2012 IEEE International Conference on Systems, Man, and Cybernetics
October 14-17, 2012, COEX, Seoul, Korea

A PSO-Based Document Classification Algorithm accelerated by the CUDA Platform

Jan Platos, Vaclav Snasel, Tomas Jezowicz, Pavel Kromer, Ajith Abraham
Department of Computer Science, FEECS
IT4 Innovations, Centre of Excellence
VSB-Technical University of Ostrava
17. listopadu 15, 708 33, Ostrava Poruba, Czech Republic
{jan.platos, vaclav.snasel, tomas.jezowicz, pavel.kromer}@vsb.cz, ajith.abraham@ieee.org

***Abstract*—Document classification is a well-known problem that is focused on assigning predefined labels or categories to the documents found in the searched collection. Many classical algorithms were developed for solving of this problem. They usually have large time complexity and with increasing number of documents it is necessary to find algorithm which are able to find solution in reasonable time. Such algorithms are usually inspired by biological processes. Even such meta-heuristics algorithms become too slow when the number of documents is really large and it is necessary to optimize them for faster processing. This paper describes a document classification algorithm based on Particle Swarm Optimization with implementation of one and two GPUs.**

***Index Terms*—particle swarm optimization, document classification, gpu, optimization.**

I. Introduction

Document classification [1] is a well-known problem that is focused on assigning predefined labels or categories to the documents found in the searched collection. Document classification may be applied into many areas such as information retrieval, personalized recommendation systems, and many others.

With the increasing number of documents becomes document classification even more significant. This task has many difficulties, because documents for classification are usually written in natural language but of course, documents may have different nature. Many classical algorithms were developed for solving document classification problem but many classical algorithms usually have large time complexity and with increasing number of documents it is necessary to find algorithm which are able to find solution in reasonable time. Such algorithms are usually inspired by biological processes. One of such algorithm is Particle Swarm Optimization (PSO) which is inspired by the social behavior of swarm of birds.

Even meta-heuristics algorithms such as PSO are become too slow and when the number of document is really large. Therefore it is necessary to find another algorithm or to use different style of programming. We select the second possibility and focus on graphics processing units (GPU). GPU are very promising technology, because they brings massive parallel architecture for reasonable price and becomes a new technical standard in writing applications and algorithms.

The rest of the paper is organized as follows. The section II describes a problem of document classification, the Section III introduces details about Particle Swarm Optimization algorithm. The Section sec:algorithm describes the algorithm suggested in this paper. Section V contains description of the performed experiments and results. Last section contains conclusion of this paper.

II. Document classification

Document classification is a process which decide if a document $d_i \in D$ belong to the category $c_i \in C$ according to the knowledge of the correct categories known for subset $D_T \subset D$ of training documents, where D is a collection of all documents and C is collection of all categories. Generally, each document may belong to more than one category and each category may contain more than one document [2].

The document classification task may be solved by humans or by automatic classifiers. Automatic documents classifiers need several preprocessing steps, which must convert documents into automatic classification capable form. The first task is preprocessing of the words a creation of vector representation of the documents. Each document is parsed out and list of used words with their frequency is extracted. Each word is compared with the list of stop-words, which are useless in classification process, because they are present in most of the documents and, therefore, bring no information about them. Each word is converted into it canonical form using normalization algorithm such as Porter Stemmer Algorithm [3]. When this process finished, the documents collection is represented as *document-term frequency matrix* $(Doc_i \times TF_{ij})$, where Doc_i refers to the each document in collection and TF_{ij} refers to the frequency of the j term in the document i. In this representation, only relation of the term to the individual document is concerned, but for classification across the document collection must be computed, therefore, we need to evaluate term importance in individual document according the importance of the term in the collection (it will be called weights). One of the way we may define a weight to the terms is according to TF-IDF (term-frequency inverse document frequency) was described in [4], [5]. The weights v_{ij} of the term j in document i is computed as :

$$v_{ij} = t_{ij} \times \log \frac{F}{f_i}$$

978-1-4673-1714-6/12/$31.00 ©2012 IEEE

Where t_{ij} is the number of times that the term j appears in document i, f_i is the number of times that the term t_j appears in the entire document database and F is the number of unique terms in document collection.

The previous paragraph described the process for conversion of document collection into (*document - term-weight*) matrix, but the number of terms with non-trivial weight for each document is still large (tenths of thousands). This may cause problems with automatic classifiers because of the large number of terms to process. Therefore, feature/term selection may be applied. Several approaches to feature selection were developed [6], [7]. One of the popular approach is entropy weighting scheme [8]. The entropy weighting scheme is computed to the each term as a multiplication of the local weighting scheme L_{ij} and the global weighting scheme G_j of the document i and term j. The definition of the schemes is following:

$$L_{ij} = \begin{cases} 1 + \log TF_{ij}, TF_{ij} > 0 \\ 0, otherwise \end{cases}$$

$$G_k = \frac{1 + \sum_{k=1}^{N} \frac{TF_{ij}}{F_j} \log \frac{TF_{ij}}{F_k}}{\log N}$$

Where N is number of documents in collections, TF_{ij} refers to the frequency of the j term in the document i, F_h is a frequency of term k in the entire document collection. In our work, we choose terms with the highest entropy value. This may reduce the number of processed term to several hundreds or less.

III. Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a optimization technique that is inspired by the behavior of the swarm of birds and other animals that uses a collective behavior. It was developed by Kennedy and Eberhart in 1995 [9], [10]. It was inspired by various interpretation of the movement of organisms in a bird flock or fish school [9]. The principle of the PSO is very simple. The goal is to find the optimal solution of the fitness function defined over the search space. PSO will generate a set of particles where each particle is defined by its position and velocity. Particles travel through a search space and try to find optimal value of the fitness function. The movement of the particle is influenced by the position of the best value found by given particle (so called local optimum) and the position of the best value over all particles (so called global optimum). The influence is realized as a change of the vector of velocity of each particle. The velocity v_i^{t+1} of i-th particle in iteration $t+1$ is defined as follows:

$$v_i^{t+1} = wv_i^t + c_1 r_1 \times lBest - x_i^t + c_2 r_2 \times gBest - x_i^t$$

where r_1, r_2 are randomly generated number in the interval $< 0, 1)$, c_1, c_2 are parameters of the algorithm called learning factors, x_i^t is the position of the i-th particle in iteration t and w is a inertia factor of the algorithm. $lBest$ and $gBest$ are local and global best solution found.

The PSO was successfully used in many optimization task such as dealing with the equality and inequality constraints of the economic dispatch problems with non-smooth cost functions [11], optimization of the reactive power flow in the power system network in order to minimize real power system losses and several other application in electric power systems [12], estimation of the non-linear parameters of mathematical models in chemistry [13], solving of no-wait flowshop scheduling problem [14], solving of vehicle routing problem with simultaneous pick-up and delivery [15] and many others.

A. GPU implementation

Implementation of PSO using GPU becomes very natural step due to the implicit parallel nature. Usually, PSO algorithms use tenths of hundreds of particles. But the most expensive part of the algorithm is usually the computation of the fitness value; therefore, the main task is the implementation of the problem solution according of the particle position using GPU. In the past, several works focused on PSO and GPU were published. Mussi et al. [16] uses PSO implemented using CUDA architecture for implementation of the Road Sign Detection used in Advanced Driver Assistance Systems. Rymut and Kwolek [17] uses PSO on GPU with cooperation of the Adaptive Appearance Model to tracking of the objects. Zhang et al. [18] uses PSO for simplification of the terrain. The following section describes our implementation of the PSO for document classification.

IV. Algorithm

As was mentioned above, we are solving document classification problem using Particle Swarm Optimization. First of all, we must define classification problem in the terms of the PSO. In our approach, as well as in [5], each particle represents a vector which best describes an individual category. Each particle (vector which describe a category) is compared with each vector in training collection during each iteration. The similarity between vectors is computed using standard Euclidean distance. When all comparisons are done, each particle modifies its best solution if necessary as well as they update global maximum if necessary.

The definition of the fitness function is the main task in each evolutionary algorithm. One of the most popular metrics in document classification is precision and recall. Precision (*Pr*) is defined as a probability that selected document is classified correctly and recall (*Re*) is defined as a probability that randomly selected document is assigned to the correct category [2]. Mathematical definitions are as follows:

$$Pr = \frac{TP}{TP + FP}$$

$$Re = \frac{TP}{TP + FN}$$

Where TP (true positive) is a count of correctly classified documents, FP (false positives) is count of documents incorrectly assigned into category, and FN (false negatives) is count of documents incorrectly not assigned into category.

The final fitness function F_1 is combination of the Precision and Recall and is defined according [19] as

$$F_1 = \frac{2 \times Pr \times Re}{Pr + Re}$$

Fitness function F_1 works well when only one category is used. In our case, we need to define classification vectors for more than one category (according to document collections). Therefore, we need to generalize fitness function for more categories. The solution is averaging of the precision and recall through all categories. Two basic averaging approaches exists according [2], Micro and Macro averaging. Macro-average is defined as a arithmetic mean of precision and recall and, therefore, prefer categories with low number of documents. Micro-average is proportional average according to number of documents in categories with higher number of documents.

A. GPU Classification using PSO

The basic algorithm of classification of documents using PSO was defined in the previous paragraphs. The precision of classification is depicted in the Section V. When we analyze the process of the searching for the optimal classification vectors, we will see, that the most time consuming part is comparison of category vector with all documents in the collection. Therefore, we need to optimize this part of the algorithm. We choose GPU units for this task. The reason was that GPU brings massive parallelism for reasonable price.

We define two approaches for this part of the algorithm. Both versions has common goal - compute the similarity between all category vectors and document vectors. Let C is the set of M categories and D is a collection of N documents. The size of the matrix is $M \times N$. The dimension of all vectors is D.

1) Variant 1: In the first variant, each kernel computes a comparison of a particular vector $c \in C$ with a k vectors from the collection D. Therefore, we need to run $\frac{M \times N}{k}$ threads, because each kernel compute exactly k comparisons. The situation if depicted in Fig 1.

2) Variant 2: The second variant of the algorithm uses different strategy. Each thread computes only one element of the similarity vector. Threads in a block then compute the final similarity between particular vectors. Therefore, we need to run d thread per block and $M \times N$ blocks. The situation is again depicted in Fig 2. This variant uses a shared memory for storing a temporal similarity vector, which increase the speed of the algorithm.

V. Experimental results

This section contains description of the performed experiments on several data sets. First part of the experiments was testing of the efficiency of the classification. Second part of the experimental results contains description of the speed up factor on one and two GPUs. All results are very interesting. First of all we describe tested collections.

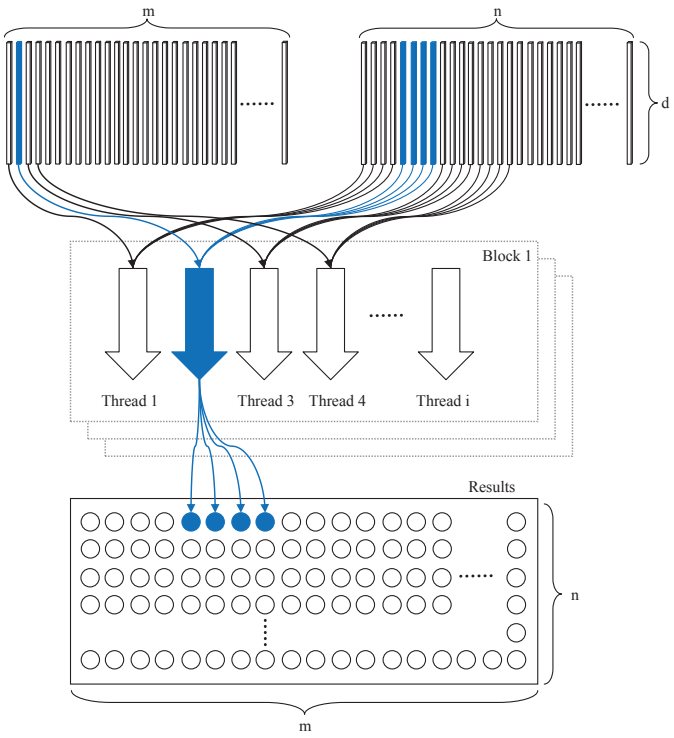

Fig. 1: Visualization of the Variant 1

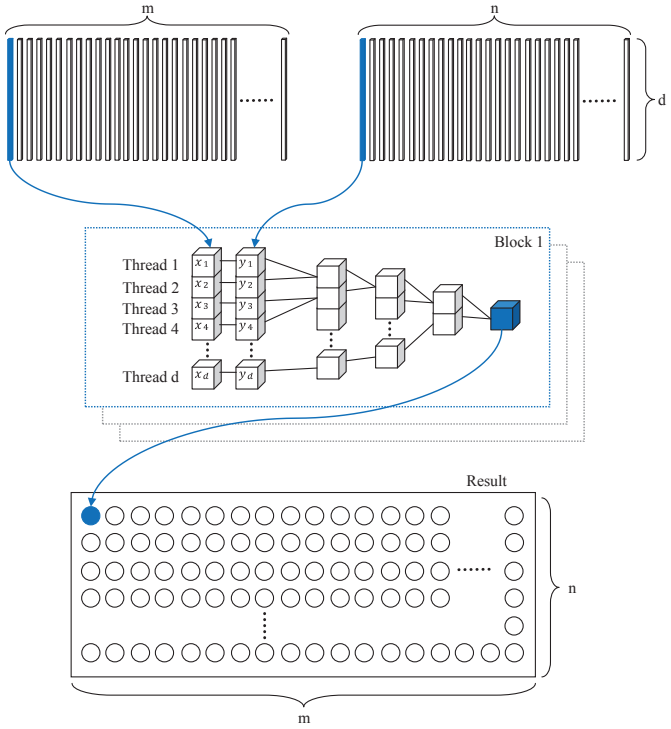

Fig. 2: Visualization of the Variant 2

A. Hardware configuration and parameters of the algorithm

In our experiments we used a server with AMD Opteron processors @GHz with Linux CentOS, two GPUs Tesla C2050 with 448 CUDA cores and 3GB of RAM each.

The parameters of the PSO algorithm were defined as follows. The number of particles was set to 10000 and algorithm uses 15 iterations. Parameters c_1 and c_2 was set to 2.0 and k was set to 0.99. Maximal speed of the particles was limited to 25.

B. Document collections

We used three different document collections. The following sections contain descriptions of the used collection with several facts and approaches of the testing.

1) Reuters-21578: First collection, Reuters-21578[1], was made in 1987 and is the most used and most popular collection for document classification problem. This collection contains 21578 papers and 135 categories, but only 120 categories has at least one document and only 57 categories has at least 20 documents. This collection also defined several partitioning into training and testing collections. The first partitioning is called *ModLewis* and contain 13625 documents in training collection and 6188 documents in testing collection. Second partitioning is called *ModApte* and contain 9603 documents in training collection and 8676 documents in testing collections. The third partitioning is called *ModHayes*, but it is not important for our paper, because we didn't use it.

In our experiments, we select ten categories with highest number of documents. All documents which belong into other categories were ignored. We perform 4 experiments - R300, Rma300, Rma1000 and R300fold3. The R means Reuters, R300 uses a partitioning ModLewis, *ma* means ModApte partitioning and the number means number of selected terms. The *fold3* is a special collection which does not use any described partitioning, but it uses 3-fold cross folding validation [2].

2) Iris collection: Iris collection [2], also known as Fisher's Iris data set, was compiled in 1936. It contains three categories, 50 documents each. Each record contains four attributes - sepal length, sepal width, petal length and petal width.

3) 20 Newsgroup: The last collection, 20 Newsgroup [3], contains 20 categories and almost 20000 documents. The distribution of the documents into categories is almost uniform. We used two setting with this collection - NG300 and NG1000. The meaning of the number is same as with the Reuters collection - the number of used terms.

C. Efficiency of the classification

First parts of the experiments were focused on efficiency of the document classification. Each table contains several columns, which are separated into 2 groups. First group contains value of Precision, Recall and function F1 computed using *micro* algorithm and second group contains same properties computed using *macro* algorithm. The higher number is better. First results which were achieved for Reuters collection are depicted in Table I.

TABLE I: Efficiency of the document classification for Reuters collection

	Micro			Macro		
Dataset	Pr	Re	F1	Pr	Re	F1
R300	0.9093	0.8608	0.8844	0.8501	0.7760	0.8114
Rma300	0.8888	0.8520	0.8700	0.7940	0.7537	0.7733
Rma1000	0.8543	0.8308	0.8424	0.8582	0.7510	0.8010
R300fold3	0.8984	0.8518	0.8745	0.8409	0.7704	0.8041

TABLE II: Efficiency of the document classification for Iris collection

Category	Pr	Re	F1
Iris-setosa	1.0000	1.0000	1.0000
Iris-versicolor	1.0000	1.0000	1.0000
Iric-virginica	1.0000	1.0000	1.0000

When we look on the results we see that the numbers are very good. In comparison with the results published in [5], we see that we have almost the same results. The differences may be cause by the different setting of the PSO algorithm which was not published in the paper.

Table II depicts the results achieved on Iris collection. We see different columns in this table because the number of categories is low and it is possible to show results for each category. As may be seen, classifiers found by PSO are perfect, because they are able to classify every document correctly.

Last collection is Newsgroup 20. We define 2 dataset from this collection, one with 300 terms selected and one with 1000 terms selected. The results are depicted in Table III. As may be seen, better results are achieved with 1000 important terms selected form the total list. But the overall results are very bad; the maximum value is 0.4417 of the F1 function which is around 50% of the efficiency achieved for the Reuters collection.

1) Speed up achieved using GPU: Description of the experiments with the GPU implementation of the PSO classification algorithm is described in this section. First experiments was focused on the processing time of the CPU version, both GPU implementation performed on one GPU and a variant 1 performed on two GPUs. As a testing collection we create artificial dataset. The GPU implementation cover only the vector comparison part of the algorithm, therefore, we need only prepare a data set that contain vectors which must be compared one-to-one. The results achieved for different input size is depicted in Fig 3. Let the input size is n then we used a 2 sets of vectors which will be compared, each with n vectors and each vector has dimension n.

The scale on the vertical axis is logarithmic. As may be seen, difference between CPU and GPU implementation increases with the size of the input. The variant 2 of the GPU implementation is faster than any other implementation from input size 16 to 256. For input sizes larger than 256 is better to use variant 1 on 2 GPUs.

The Fig 4 depicts the speed up achieved by the GPU

[1] http://www.daviddlewis.com/resources/testcollections/reuters21578/

[2] http://archive.ics.uci.edu/ml/datasets/Iris

[3] http://people.csail.mit.edu/jrennie/20Newsgroups/

TABLE III: Efficiency of the document classification for 20 Newsgroup collection

	Micro			Macro		
Dataset	Pr	Re	F1	Pr	Re	F1
NG300	0.3409	0.3702	0.3549	0.3639	0.3742	0.3690
NG1000	0.4794	0.3839	0.4264	0.5337	0.3768	0.4417

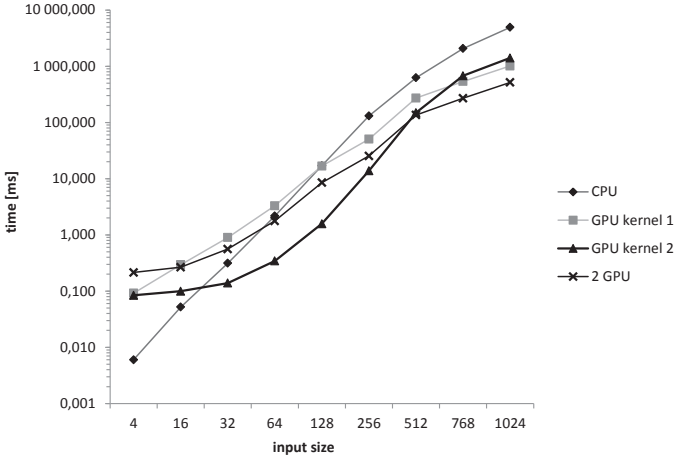

Fig. 3: Efficiency of the different implementation of vector comparison

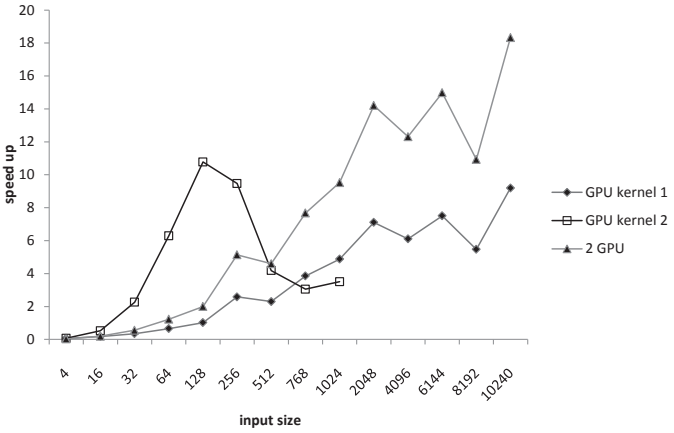

Fig. 4: Efficiency of the different implementation of vector comparison

TABLE IV: Speed up on the real testing collection

Dataset	m	n	d	Speedup
Rma300	4161	10000	300	10.45
Rma1000	7945	10000	1000	9.45
NG300	7945	10000	300	8.16
NG1000	7902	10000	1000	8.29
Iris	60	10000	4	2.54

implementations over the CPU implementation for different input sizes.

The results for the Variant 2 of the algorithm are computed only for input size up to 1024 because of the limitation of CUDA implementation which is not able to process vectors larger than 1024 elements into shared memory of the processing unit. But as may be seen, Variant 1 of the algorithm is much more efficient for input sizes larger than 512 even on single GPU. The speed of the 2 GPUs implementation is approximately two times higher than the single GPU implementation.

The real experiments on the testing collection are depicted in Table IV. Achieved speed up was higher than 8 which is very good. Therefore it is clear, that we are able to decrease time consumption almost by factor 10. Depicted results were achieved on one GPU. Implementation on 2 GPUs will be almost two times higher.

VI. Conclusion

This paper described an document classification algorithm based on PSO and implemented on GPU. The efficiency of the algorithm was tested on three collections - Reuters-21578, Iris and 20 Newsgroup. Achieved efficiency of the algorithm was very for Reuters and Iris collection and it was comparable with others document classification algorithms. The results for 20 Newsgroup were not ideal and needs to be improved with modification of the algorithm.

Second part of the paper was focused on implementation of suggested algorithm using GPU. The part that was transformed into GPU was the most time consuming part - vector comparison. Two variants of the algorithm for this task were developed. The second variant was more efficient on vectors with dimension less than 1024, which was the hardware limitation of the algorithm. The first variant was more efficient for dimension larger than 1024 and was more suitable for implementation on more than one GPU. The efficiency of the GPU implementation was tested on artificial collection at first. The maximum speed up was 20. Experiments on mentioned collections show that the real speed up over CPU implementation was 10.45 and not less than 8.16 except Iris collection which is very small.

Acknowledgment

This work was partially supported by the SGS in VSB Technical University of Ostrava, Czech Republic, under the grant No. SP2012/58, and was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070) and by the Bio-Inspired Methods: research, development and knowledge transfer project, reg. no. CZ.1.07/2.3.00/20.0073 funded by Operational Programme Education for Competitiveness, co-financed by ESF and state budget of the Czech Republic.

References

[1] C. van Rijsbergen, *Information Retrieval*. Butterworths, 1979.

[2] F. Sebastiani, "Machine learning in automated text categorization," *ACM Comput. Surv.*, vol. 34, no. 1, pp. 1–47, Mar. 2002. [Online]. Available: http://doi.acm.org/10.1145/505282.505283

[3] M. F. Porter, "Readings in information retrieval," K. Sparck Jones and P. Willett, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, ch. An algorithm for suffix stripping, pp. 313–316. [Online]. Available: http://dl.acm.org/citation.cfm?id=275537.275705

[4] V. P. Guerrero Bote, F. de Moya Anegón, and V. H. Solana, "Document organization using kohonen's algorithm," *Inf. Process. Manage.*, vol. 38, no. 1, pp. 79–89, Jan. 2002. [Online]. Available: http://dx.doi.org/10.1016/S0306-4573(00)00066-2

[5] Z. Wang, Q. Zhang, and D. Zhang, "A pso-based web document classification algorithm," in *Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing - Volume 03*, ser. SNPD '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 659–664. [Online]. Available: http://dx.doi.org/10.1109/SNPD.2007.84

[6] Y. Saeys, I. Inza, and P. Larra?aga, "A review of feature selection techniques in bioinformatics," *Bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007. [Online]. Available: http://bioinformatics.oxfordjournals.org/content/23/19/2507.abstract

[7] Y. Yang and J. Pedersen, *Feature selection in statistical learning of text categorization*. Morgan Kaufmann, 1997, pp. 412–420.

[8] S. Dumais, "Improving the retrieval of information from external sources," *Behavior Research Methods*, vol. 23, pp. 229–236, 1991, 10.3758/BF03203370. [Online]. Available: http://dx.doi.org/10.3758/BF03203370

[9] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4, nov/dec 1995, pp. 1942 –1948 vol.4.

[10] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on*, oct 1995, pp. 39 –43.

[11] J.-B. Park, K.-S. Lee, J.-R. Shin, and K. Lee, "A particle swarm optimization for economic dispatch with nonsmooth cost functions," *Power Systems, IEEE Transactions on*, vol. 20, no. 1, pp. 34 – 42, feb. 2005.

[12] M. AlRashidi and M. El-Hawary, "A survey of particle swarm optimization applications in electric power systems," *Evolutionary Computation, IEEE Transactions on*, vol. 13, no. 4, pp. 913 –918, aug. 2009.

[13] M. Schwaab, E. C. Biscaia, Jr., J. L. Monteiro, and J. C. Pinto, "Nonlinear parameter estimation through particle swarm optimization," *Chemical Engineering Science*, vol. 63, no. 6, pp. 1542 – 1552, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0009250907008755

[14] Q.-K. Pan, M. F. Tasgetiren, and Y.-C. Liang, "A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem," *Computers & Operations Research*, vol. 35, no. 9, pp. 2807 – 2839, 2008, ¡ce:title¿Part Special Issue: Bio-inspired Methods in Combinatorial Optimization¡/ce:title¿. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0305054806003170

[15] T. J. Ai and V. Kachitvichyanukul, "A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery," *Computers & Operations Research*, vol. 36, no. 5, pp. 1693 – 1702, 2009, ¡ce:title¿Selected papers presented at the Tenth International Symposium on Locational Decisions (ISOLDE X)¡/ce:title¿. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0305054808000774

[16] L. Mussi, S. Cagnoni, and F. Daolio, "Gpu-based road sign detection using particle swarm optimization," in *Intelligent Systems Design and Applications, 2009. ISDA '09. Ninth International Conference on*, 30 2009-dec. 2 2009, pp. 152 –157.

[17] B. Rymut and B. Kwolek, "Gpu-supported object tracking using adaptive appearance models and particle swarm optimization," in *Computer Vision and Graphics*, ser. Lecture Notes in Computer Science, L. Bolc, R. Tadeusiewicz, L. Chmielewski, and K. Wojciechowski, Eds. Springer Berlin / Heidelberg, 2010, vol. 6375, pp. 227–234, 10.1007/978-3-642-15907-7_28. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-15907-7_28

[18] H. Zhang, J. Sun, J. Liu, and N. Lv, "A new simplification method for terrain model using discrete particle swarm optimization," in *Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems*, ser. GIS '07. New York, NY, USA: ACM, 2007, pp. 67:1–67:4. [Online]. Available: http://doi.acm.org/10.1145/1341012.1341091

[19] Y. Yang, "An evaluation of statistical approaches to text categorization," *Information Retrieval*, vol. 1, pp. 69–90, 1999, 10.1023/A:1009982220290. [Online]. Available: http://dx.doi.org/10.1023/A:1009982220290