Parallelization of the Particle Competition and Cooperation approach for Semi-Supervised Learning

Raul Moreira de Souza
Department of Statistics, Applied Mathematics and Computation(DEMAC)
Institute of Geosciences and Exact Sciences (IGCE)
São Paulo State University (UNESP)
Rio Claro, Brazil
raul_ms02@yahoo.com.br

Fabricio Breve
Department of Statistics, Applied Mathematics and Computation(DEMAC)
Institute of Geosciences and Exact Sciences (IGCE)
São Paulo State University (UNESP)
Rio Claro, Brazil
fabricio@rc.unesp.br

***Abstract*—This work emphasis the development of fast ways to perform the particle competition and cooperation model, which is a nature inspired algorithm of semi-supervised learning. To do so, the algorithm was parallelized on multi-core CPU and GPU(Graphics Process Unit). A modified version of the model aiming to increase its velocity was proposed, developed and tested. This paper contains explanations of how the proposed algorithm works, how it was implemented for parallel processing, how the modified version works and experiments comparing the different implementations of the particle competition and cooperation algorithm on real world data sets, followed by analysis of the results.**

Keywords—Semi-supervised learning; particles competition and cooperation; network-based methods; label propagation; parallelism; GPU implementation;

I. INTRODUCTION

One of the most important branches of computer science is the parallelism, and its importance is growing up with the new multi-core processors and GPU's(Graphic Processing Units) technologies, so the parallelization of algorithms has become very crucial nowadays [1], [2], [3]. Other important field of computer science is the nature inspired computing, which is the development of computer techniques based on natural behaviors [4]. This paper describes the work and results of a research whose objective was to develop faster ways to perform a semi-supervised learning and nature inspired algorithm proposed by Breve et. Al[5] without injure its efficiency, through the implementation of parallel processing, focusing on it's use has an automatic data classifier.

II. THE COMPETITION AND COOPERATION PARTICLE ALGORITHM

This section is an overview of the particle competition and cooperation algorithm proposed by Breve et al.[5]. The proposed model is a graph based semi-supervised learning algorithm inspired by natural behaviors, such as competition for resources between animals, exploration of territory by humans and animals, cooperation among individuals of the same specie or group, etc. This model contains mechanisms of competition and cooperation combined into a single schema. Particles representing the same class form teams and walk in a cooperative way to propagate their label on the network. At the same time, particles of different classes (teams) compete for determining the edge of classes.

Given a set of labeled data, each of its instances becomes a particle. Each particle has a strength attribute which determines how its going to change the domination levels of a node and it is initially 1, a home node which corresponds to a labeled node of the network, a team which the particle will cooperate and a table of distances of the particle to all the data set nodes.

Each network node has a vector of elements where each element represents the level of domination of a team on that node. As the system runs, the particles go through the network and increases the level of domination of its team on the nodes, while decreases the level of domination of other teams. Furthermore, each particle also has a power level which increases when it visits a node dominated by its team, and decreases when it visits a node dominated by another team. This force is important because the change that particle cause in a node (increasing its team dominance level) is proportional to the strength that it currently has. This mechanism ensures that the particles become stronger when they are in their neighborhood, protecting it, and they get weaker when they try to invade other teams territories.

At each iteration, each particle chooses one of two walking types randomly with predefined probabilities:

- Random walking: the particle chooses randomly, with equals probabilities, any of the neighbor nodes of the node where it is in. This walking is used to reproduce the animal behavior of territories exploration, in this case the particles will explore the graph nodes.
- Greedy walking: the particle chooses randomly any of the neighbor nodes of the node where it is, with probabilities calculated proportionally to the dominance levels of the particle's team in each node.

In that way the particles tends to walk on the nodes that are dominated by its team protecting them from invasion by other teams, so the greedy walking generates a defensive behavior on the particles. The Fig.1 illustrates how the algorithm works.

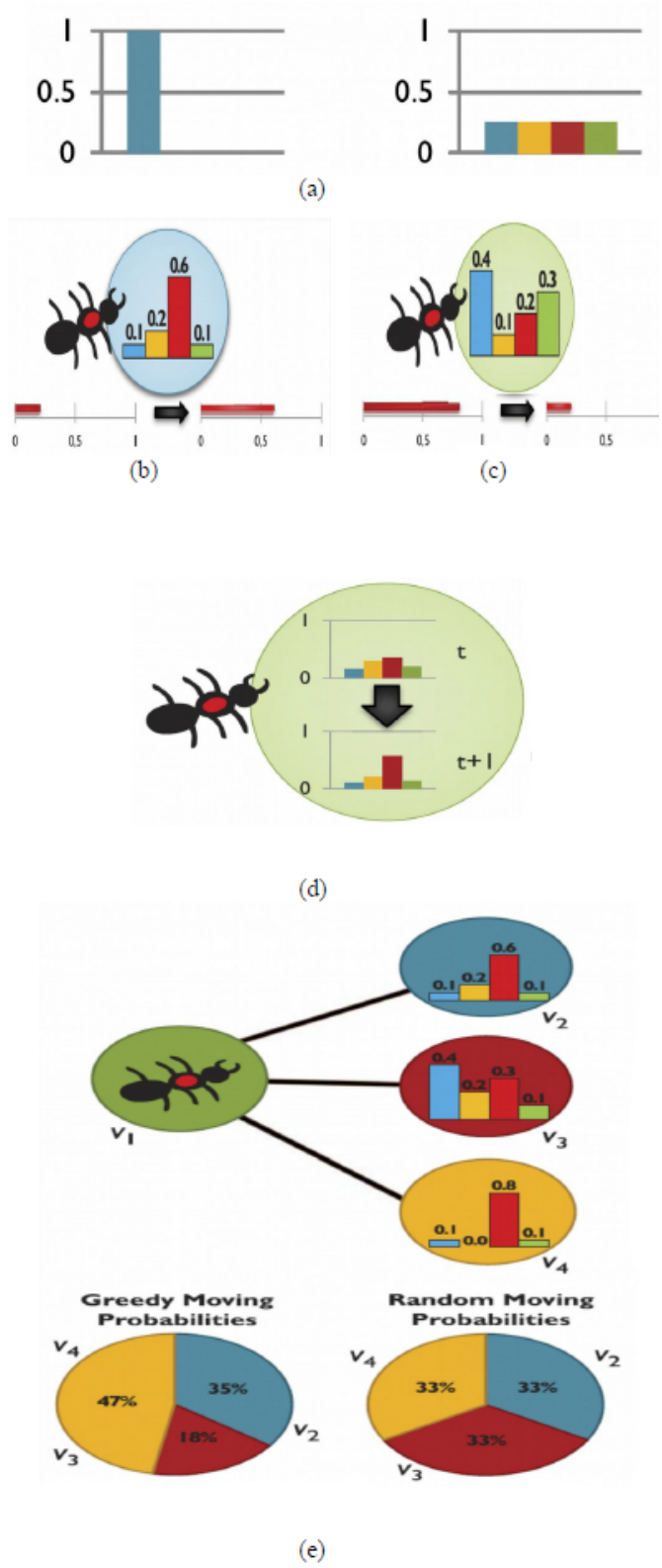

Fig. 1. Illustrations of node and particle dynamics. (a) initial domination levels for a node corresponding to a labeled sample (left) and an unlabeled node (right) in a problem of four classes; (b) a particle gets stronger as it is targeting a node being dominated by its own team; (c) a particle gets weaker as it is targeting a node being dominated by another team; (d) a particle increases its team domination level in the target node while decreasing domination level of other teams; and (e) node probabilities of being chosen by a particle with greedy and random movement, all candidate nodes have the same distance from the particle home node[5].

The main goal of this work was to implement the particle competition and cooperation algorithm with parallel processing making the particles walk synchronously with different threads processing different particles.

III. The Parallelization of the Proposed Algorithm

A. The Parallelization on CPU

To make the algorithm work in parallel process on the CPU threads, the particles were divided into smaller groups. Each thread process the walking of one particular particle group during the entire main loop. Synchronizations points were applied on the particle walking in order to avoid that one particle walks much more than another at the same time. To the parallelization on CPU the Open Multi-Processing(OpenMP) library was chosen to be used because it is an easy to use and efficient tool for parallelism [6].

B. The Parallelization on GPU

The implementation of the algorithm in GPU is much more complex than in multi-core CPU, there are specific requirements that need to be accomplished to take advantage of the graphic processing unit, such as the occupancy and device memory transfer. The GPU implementation was entirely done with the CUDA architecture, to do so, it was necessary to study and understand its concepts [7].

A large amount of threads is required in order to obtain a better efficiency in GPU. To adapt the algorithm to this requisite each thread process the walking of one single particle. As the random and greedy walking steps have different process complexity, the synchronization of the particles become harder and more synchronization points were applied to this version, in order to avoid that one particle walks much more than another at the same time.

The main problem on the adaptation to the GPU is the occupancy factor, if a process cannot be divided into a number of threads that will keep all the GPU core processors occupied, the GPU potential is not fully used, and that number can only be reached with a large amount of particles.

Another main factor is the device memory which is more limited than CPU memory, if the device memory is exceeded it is necessary to use host allocated memory, which makes the processing slower. In the proposed algorithm most of the data needs to be transferred from the CPU to the GPU, which also consumes time and compromises the performance.

IV. The New Proprosed Approach

In order to get a better performance in terms of fastness a new approach of the algorithm was proposed and implemented. In this method each instance of the labeled subset of data generates a set of particles instead of just one, in all experiments of this approach, in this paper, each set has 30 particles. The particles of the set generated by the same instance share the same home node and are equal at the initial state, each particle of this set will randomly visit one of the current node neighbor at each iteration propagating its label, like in the original model.

There is no greedy walking in this version, the idea is to instantiate many particles based on one single data instance, instead of calculate which node is more dominated, the particles go randomly through the nodes filling most of the possible ways. In the original algorithm the greedy walk prevents particles of becoming too weak, in this one, some of then will become stronger and others weaker, depending on its course.

This version doesn't require a table of distance which make it consume less memory, even with the increased number of particles, and although that increased number of particles makes each iteration much longer, the number of iterations required by the algorithm to reach optimal results is much smaller, making it faster than the parallelized version of the original algorithm.

This method was developed aiming to GPU, it solves desynchronization issues caused by greedy walking, having more particle means it can be separated into more threads, fully occupying all the GPU's cores and consumes less memory(which is more limited in graphic cards). However, the experiments shows that it is also applicable for CPU, this new approach has the same accuracy of the original on small data sets and tends to have better accuracy on the larger ones.

V. THE RESULTS

The implementations were tested on real world data sets [8] with different aspects aiming to obtain a more detailed understanding of each one's behavior. The Table I. describes the details of each data set used.

TABLE I. DATA SETS USED FOR THE TESTS

	Data Set Aspects		
Data set Name	***Nº of Instances***	***Nº of attributes***	***Nº of classes***
Iris	150	4	3
Wine	178	13	3
Satellite Image	6435	36	7
Pen-based Recognition	10992	16	10
Statlog(Shuttle)	58000	9	7
Skin Segmentation	245057	3	2

The Tables II to XI contain computational experiments results of the data sets, with measurements of time and classification accuracy achieved by each of the implementations. The implementations are: (I)single core CPU, (II)parallel processing on multi-core CPU, (III)parallel processing on GPU, (IV)new approach with parallel processing on multi-core CPU and (V)new approach with parallel processing on GPU. The time measurements are the execution time of the particle's walk only, the graph formation time was not counted, since for each data set was used the same graph for all experiments, and all the graph are KNN(k-Nearest neighbor) undirected and unweighted. All the results showed in the tables are the average accuracy and execution time of 100 runs, the samples were chosen randomly among the entire data set and all the tests were made in the same computer with the following specs :

- Processor: Intel(R) Core(TM) i7-2600K CPU @ 3.4GHz 3.7 GHz;
- RAM: 32.0 GB;
- GPU: two NVidia GEForce GTX 560Ti using Nvidia SLI Technology with 2.0GB of device memory each;

TABLE II. CLASSIFICATION ACCURACY IN THE IRIS DATA SET

% of labeled samples	***Method***				
	I	***II***	***III***	***IV***	***V***
5,3	80,25	80,48	80,11	80,41	81,12
10,7	88,98	89,07	88,47	88,76	88,22
21,3	92,13	92,14	91,13	92,70	91,14

TABLE III. CLASSIFICATION ACCURACY IN THE WINE DATA SET

% of labeled samples	***Method***				
	I	***II***	***III***	***IV***	***V***
4,5	76,45	75,98	75,87	77,45	76,82
9,0	90,32	90,49	89,84	90,09	89,75
18,0	93,93	94,05	93,10	93,93	93,51

There is no table of time execution for the *Iris* and *Wine* because with these data sets a comparison of time is pointless, sometimes the execution is faster than what could be measured by the time measuring method used, making its variance too large, and in the GPU an acceptable execution time could not be reached due to the size of these data sets. those data sets were tested in order to compare the accuracy.

TABLE IV. CLASSIFICATION ACCURACY IN THE SATELLITE IMAGE DATA SET

% of labeled samples	***Method***				
	I	***II***	***III***	***IV***	***V***
2,0	83,16	83,69	82,88	84,37	83,25
4,0	86,29	85,98	84,91	86,36	86,02
8,0	88,05	88,18	87,94	87,83	87,49

TABLE V. TIME EXECUTION ON THE SATELLITE IMAGE DATA SET IN MILLISECONDS

% of labeled samples	***Method***				
	I	***II***	***III***	***IV***	***V***
2,0	450	160	561	22	25
4,0	945	256	538	49	28
8,0	1941	489	492	89	59

TABLE VI. CLASSIFICATION ACCURACY IN THE PEN-BASED RECOGNITION DATA SET

% of labeled samples	*Method*				
	I	***II***	***III***	***IV***	***V***
1,2	90,89	90,85	89,76	93,72	93,48
2,3	94,86	94,03	93,84	96,14	96,02
4,7	96,01	96,11	96,05	97,86	97,79

TABLE VII. TIME EXECUTION ON THE PEN-BASED RECOGNITION DATA SET IN MILLISECONDS

% of labeled samples	*Method*				
	I	***II***	***III***	***IV***	***V***
1,2	426	209	630	76	64
2,3	893	309	607	129	42
4,7	1799	540	534	222	83

In the Tables V and VII there are some interesting results to discuss how the GPU implementation works, with a bigger number of particles it can run faster or almost with the same time than with a smaller number particles, even that the number of iterations was the same for all tests in those tables. That happens due to the GPU occupancy, with a small number of particles the algorithm cannot be divided into as many threads as the required number to keep the device fully occupied.

TABLE VIII. CLASSIFICATION ACCURACY IN THE STATLOG(SHUTTLE) DATA SET

% of labeled samples	*Method*		
	I	***II***	***IV***
0,4	84,25	84,52	85,95
0,9	89,47	90,01	92,84
1,8	95,02	95,55	96,83

TABLE IX. TIME EXECUTION ON THE STATLOG(SHUTTLE) DATA SET IN MILLISECONDS

% of labeled samples	*Method*		
	I	***II***	***IV***
0,4	1416	434	108
0,9	3064	790	204
1,8	7154	1633	359

TABLE X. CLASSIFICATION ACCURACY IN THE SKIN SEGMENTATION DATA SET

% of labeled samples	*Method*		
	I	***II***	***IV***
0,4	75,47	75,43	77,59
0,6	79,28	79,43	83,30
0,8	81,29	81,63	84,33

TABLE XI. TIME EXECUTION ON THE SKIN SEGMENTATION DATA SET IN MILLISECONDS

% of labeled samples	*Method*		
	I	***II***	***IV***
0,4	262230	63862	1309
0,6	392881	97574	2027
0,8	527601	130266	2709

The data sets *Statlog* and *Skin Segmentation* were not tested on GPU because its processing exceeded the graphic device memory, this is a major problem of the implementation of the proposed model on GPU, the consume of memory grow proportionally to the number of particles, so the amount of particles necessary to fully occupy the graphic device will probably exceeds its memory.

VI. CONCLUSION

As shown by the results, the parallelization was successfully applied, reaching faster execution times without loss of classification accuracy.

The new approach seems to have better classification accuracy on large data sets, with a faster execution time.

The particle competition and cooperation algorithm has proven itself very parallelizable in a wide range of data set sizes in CPU. It was not advantageous in most of the cases in GPU due to the difficulties of dividing the process into the required amount of threads without exceeding the device memory.

The new approach of the method takes more advantage of the GPU parallelism, as discussed in the Section III. In CPU it can be used with larger data sets than the classic approach due to less use of memory, since there is no table of distances or calculation of probabilities.

In general, all the implementations achieved proper classification accuracy with a small amount of labeled samples, showing that this graph based algorithm is a good option for real world problems and very capable of being used in the newer processors.

REFERENCES

[1] Mitchell, M. Oldham, J. Samuel, A. Advanced Linux Programming. Indianapolis, USA: Newriders,2001.

[2] Lakshmanan K, Kato S, Rajkumar RR (2010) Scheduling parallel real-time tasks on multi-core processors. In: RTSS'10: proceedings of the 30th IEEE real-time systems symposium, pp 259–268.

[3] Anderson JH, Calandrino JM (2006) Parallel real-time task scheduling on multicore platforms. In: RTSS'06: proceedings of the 27th IEEE real-time systems symposium, pp 89–100.

[4] Leandro Nunes de Castro, Fundamentals of Natural Computing (Chapman & Hall/Crc Computer and Information Sciences), Chapman & Hall/CRC, 2006 .

[5] Breve F. A.; Zhao L.; Quiles M. G.; Pedrycz W.; Liu J. "Particle competition and cooperation in networks for semisupervised learning". *IEEE Transactions on Knowledg and Data Engineering (PrePrints)*, 2012.DOI10.1109/TKDE.2011.119; http://doi.ieeecomputersociety.org/10.1109/TKDE.2011.119

[6] OpenMP (2011) OpenMP: open multi-processing. http://openmp.org

[7] NVIDIA, C. U. D. A. Programming guide. 2008.

[8] Bache, K. & Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[9] Mitchell, T. M. Machine learning. USA: McGraw-Hill Series in Computer Science, McGraw-Hill Companies, 1997.

[10] Alpaydin, E. (2004). Introduction to machine learning. MIT Press.

[11] Quiles, M. G., Zhao, L., Alonso, R. L., & Romero, R. A. F. (2008). Particle competition for complex network community detection. Chaos, 18(3), 033107.

[12] Zhu, X. (2005). Semi-Supervised Learning Literature Survey. Technical Report 1530 Computer Sciences, University of Wisconsin-Madison.

[13] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.