Fast Optimum-Path Forest Classification on Graphics Processors

Marcos V. T. Romero[1], Adriana S. Iwashita[1], Luciene P. Papa[2], André N. Souza[3] and João P. Papa[1]
[1]*Department of Computing, São Paulo State University, Bauru, São Paulo, Brazil*
[2]*Southwest Paulista College, Avaré, São Paulo, Brazil*
[3]*Department of Electrical Engineering, São Paulo State University, Bauru, São Paulo, Brazil*

Keywords: Optimum-Path Forest, Graphics Processing Unit.

Abstract: Some pattern recognition techniques may present a high computational cost for learning samples' behaviour. The Optimum-Path Forest (OPF) classifier has been recently developed in order to overcome such drawbacks. Although it can achieve faster training steps when compared to some state-of-art techniques, OPF can be slower for testing in some situations. Therefore, we propose in this paper an implementation in graphics cards of the OPF classification, which showed to be more efficient than traditional OPF with similar accuracies.

1 INTRODUCTION

Pattern recognition techniques have as main goal to classify a dataset based on a previous learning over some predefined samples, which are described by a set of features extracted from an observation set, defining points on a multidimensional space. Depending on the information we have about the dataset, we can face unsupervised and supervised techniques, where a fully labeled dataset can enhance the effectiveness of such approaches (Duda et al., 2000).

Supervised learning approaches allow the classifier to learn the behaviour of the dataset from a training set, and then evaluate this learning applying the knowledge over an unseen test set. Recently, a graph-based framework for supervised pattern recognition techniques was proposed by Papa et al. (Papa et al., 2009; Papa et al., 2012). The Optimum-Path Forest (OPF) classifier models pattern recognition as an optimum graph partition problem, in which a competition process among some samples generates a collection of optimum-path trees (OPTs) rooted at them. Each class can be represented by just one or several OPTs, which encode the power of connectivity between samples.

Even though OPF has obtained results comparable to the ones given by some state-of-the-art pattern recognition techniques, such as Support Vector Machines (SVM) and Neural Networks (NN), being also faster than them for training, the OPF classification has a considerable computational burden. While SVM testing phase uses only the support vectors computed in the training phase, and for NN we need only the neurons' weights, in regard to OPF it is required to compute the distance for all training set to classify just one test sample. Although Papa et al. (Papa et al., 2012) have proposed an optimization of this step, the OPF test phase may be still time consuming.

Aiming to achieve even faster classification phases, some works have presented implementations of pattern recognition techniques in CUDA (Compute Unified Device Architecture) environment, which makes use of the high number of cores embedded on the Graphic Processing Unit (GPU) of some recent graphics cads. Catanzaro et al. (Catanzaro et al., 2008), for instance, proposed a GPU-based implementation of SVM which is around $9--35\times$ faster than traditional implementation on CPU (Central Processing Unit). Oh and Jung (Oh and Jung, 2004) have introduced neural networks in the context of GPU-based programming.

Further, Jang et al. (Jang et al., 2008) proposed a NN implementation combining CUDA e OpenMP (*Open Multi-Processing*). In this work, the authors argued about the massive parallel programming in GPUs, in which we need to take care about the granularity of the problem. Therefore, the amount of source code which can be parallelized must counterbalance the system's throughput. Recently, Gainaru et al. (Gainaru et al., 2011) presented a study about some data mining algorithms implemented on GPU devices.

In this work, we present a CUDA-based OPF testing phase, as well as some experiments which shown

V. T. Romero M., S. Iwashita A., P. Papa L., N. Souza A. and P. Papa J..
Fast Optimum-Path Forest Classification on Graphics Processors.
DOI: 10.5220/0004740406270631
In *Proceedings of the 9th International Conference on Computer Vision Theory and Applications* (VISAPP-2014), pages 627-631
ISBN: 978-989-758-004-8

the gain on efficiency keeping similar recognition rates when compared with traditional OPF. As far as we know, this is the first proposal of one GPU-based implementation of OPF up to date. The remainder of this paper is organized as follows. Section 2 and 3 present OPF background and the approach approach to boost OPF testing step. Section 4 discusses the experimental results and Section 5 states conclusions.

2 OPTIMUM-PATH FOREST

Let $\mathcal{X}$ and $\mathcal{Y}$ be a set of samples and their corresponding labels, respectively, where $\mathcal{Y} \subseteq \mathcal{M}$, being $\mathcal{M}$ the set of all possible labels. Given a labeled dataset $\mathcal{D} = (\mathcal{X}, \mathcal{Y})$, the idea behind OPF is to model $\mathcal{D}$ as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ whose nodes are the samples in $\mathcal{V} = \mathcal{X}$, and the arcs are defined by an adjacency relation $\mathcal{A}$ between nodes in the feature space. The arcs are weighted by a distance function between the feature vectors of the corresponding nodes.

As a community ordered formation process, where groups of individuals are obtained based on optimum connectivity relations to their leaders, OPF employs a competition process among some key nodes (prototypes) in order to partition the graph into an optimum-path forest according to some path-cost function. By analogy, the population is divided into communities, where each individual belongs to the group which offered to it the highest reward. In addition, the dataset $\mathcal{D}$ is divided in two subsets $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$, standing for the training and test sets, respectively. Now, we have the following graphs: $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{A}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{A}_2)$, $\mathcal{V}_1 \cup \mathcal{V}_2 = \mathcal{X}$, which model $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively.

Now assume π_s be a path in the graph with terminus at sample $s \in \mathcal{D}_1$, and $\langle \pi_s \cdot (s,t) \rangle$ a concatenation between π_s and the arc (s,t). Let $\mathcal{S} \subset \mathcal{D}_1$ be a set of prototypes from all classes. Roughly speaking, the main idea of the Optimum-Path Forest algorithm is to minimize a cost map $O(t) = \min_{\forall t \in \mathcal{V}_1}\{\Psi(\pi_t)\}$, where Ψ is a path-cost function defined as:

$$\begin{aligned} \Psi(\langle s \rangle) &= \begin{cases} 0 & \text{if } s \in \mathcal{S} \\ +\infty & \text{otherwise,} \end{cases} \\ \Psi(\pi_s \cdot \langle s,t \rangle) &= \max\{\Psi(\pi_s), d(s,t)\}, \end{aligned} \quad (1)$$

in which $d(s,t)$ stands for the distance between nodes s and t.

Particularly, an optimal set of prototypes $\mathcal{S}^*$ can be found by exploiting the theoretical relation between the minimum-spanning tree (MST) (Cormen et al., 2001) and optimum-path tree for Ψ (Allène et al., 2007). By computing a MST in the complete graph $\mathcal{G}_1$, we obtain a connected acyclic graph whose nodes are all samples in $\mathcal{V}_1$. This spanning tree is optimum in the sense that the sum of its arc weights is minimum as compared to any other spanning tree in the complete graph. In addition, every pair of samples is connected by a single path, which is optimum according to Ψ. Therefore, the optimum prototypes are defined as the nodes from distinct classes that share an arc in the MST.

In the classification phase, for any sample $t \in \mathcal{V}_2$, we consider all arcs connecting t with samples $s \in \mathcal{V}_1$, as though t were part of the graph. Considering all possible paths from $\mathcal{S}^*$ to t, we find the optimum path $\mathcal{P}^*(t)$ from $\mathcal{S}^*$ and label t with the class $\lambda(\mathcal{R}(t))$ of its most strongly connected prototype $\mathcal{R}(t) \in \mathcal{S}^*$, being $\lambda(\cdot)$ a function that returns the true label of some sample. This path can be identified by evaluating the optimum cost $O(t)$ as:

$$O(t) = \min\{\max\{O(s), d(s,t)\}\}, \ \forall s \in \mathcal{V}_1. \quad (2)$$

Suppose the node $s^* \in \mathcal{G}_1$ is the one which satisfies (Equation 2). Given that label $\theta(s^*) = \lambda(\mathcal{R}(t))$, the classification simply assigns $\theta(s^*)$ as the class of t. Clearly, an error occurs when $\theta(s^*) \neq \lambda(t)$, where $\theta(\cdot)$ stands for the predicted label for some sample.

3 PROPOSED ARCHITECTURE FOR GPU-BASED OPF

In this section, we present the GPU-based implementation for OPF testing phase, as well as some definitions employed in this work, which have been also proposed to fulfill the development of the parallel version of OPF.

3.1 Matrix Association

Let T be a matrix association operation between two matrices A and B with dimensions $n_1 \times m$ and $m \times n_2$, respectively. As a result of T, we can obtain a matrix C with dimensions $n_1 \times n_2$, as follows:

$$C = T(A, B, f(x,y)), \quad (3)$$

in which $f(x,y)$ stands for a generic function, being each element c_{ij} from C defined as

$$c_{ij} = \sum_{k=1}^{m} f(a_{ik}, b_{kj}), \quad (4)$$

where a_{ik} and b_{kj} stand for elements from A and B, respectively.

In order to make it clear, a matrix multiplication, for instance, can be written as $C = T(A, B, f(x,y))$, where $f(x,y)$ is defined as follows:

$$f(x,y) = x * y. \quad (5)$$

3.2 Parallel OPF Classification on GPU Devices

The main performance issue of the OPF testing phase concerns with the distance calculation between all training nodes and the testing sample. Such process is used to find the lowest path-cost given by some training node. Therefore, we propose here to parallelize two steps of the OPF testing phase : (i) distance calculation between training and testing nodes, and (ii) finding the lowest path-cost given by some training node to each testing sample.

In regard to the former step, suppose the testing set can be modeled by a matrix M_1 with dimensions $n_1 \times n_f$, in which n_1 and n_f stand for the testing set size and the number of features, respectively. Thus, each row i of M_1 represents the features from testing node i. Similarly, the training set is then modeled by a matrix M_2 with dimensions $n_f \times n_2$, in which n_2 denotes the training set size.

Assuming the Euclidean distance as the similarity function (as employed the default distance by OPF), the distance calculation step can be modeled as a matrix association between M_1 (test set) and M_2 (training set), being $f(x,y)$ in Equation 3 given by:

$$f(x,y) = (x-y)^2. \quad (6)$$

Therefore, Equation 3 can be rewritten as follows:

$$C = T(M_1, M_2, f(x,y)). \quad (7)$$

After that, a square root must be applied to each element c_{ij} of the resulting matrix C in order to complete the Euclidean distance computation:

$$c_{ij} \leftarrow \sqrt{c_{ij}}. \quad (8)$$

In order to address the second step of OPF classification phase, i.e., to find the lowest path-cost given by some training node to each test sample, it is first necessary to compute equation below:

$$c_{ij} \leftarrow \max\{c_{ij}, cost_j\}, \quad (9)$$

where $cost_j$ contains the optimum cost of the training node j, which is computed in the training step via traditional OPF. Finally, c_{ij} contains all path-costs offered to test sample i by training node j.

In order to solve Equation 2 for testing node i, we just need to find the lowest value in line i of matrix C, which can be done in parallel with a reduction operation. For such purpose, we have used a tree-based parallel reduction operation (Harris, 2010).

4 EXPERIMENTAL RESULTS

In this section we present the experimental results conducted to show the robustness of the proposed approach. We have designed two different scenarios for performance comparisons using computers with distinct configurations: GPU 1 and GPU 2. The former is equipped with an Intel Core 2 Quad Q9300 processor with 3.24GHz, 4GB DDR3 1333MHz RAM and an EVGA GeForce GTX 550 Ti 1GB GDDR5 video card with 192 CUDA cores. GPU 2 is equipped with an Intel Core i7 3770 processor with 3.4GHz, 8GB DDR3 1600MHz RAM and a Gigabyte Geforce GTX 680 2GB GDDR5 with 1536 CUDA cores. Basically, the first configuration is a low-cost computer equipped with a not too old video card. On the other hand, GPU 2 has been equipped with one of the best Intel processors available on the market for domestic users, and also one of the best single GPU video card for domestic use.

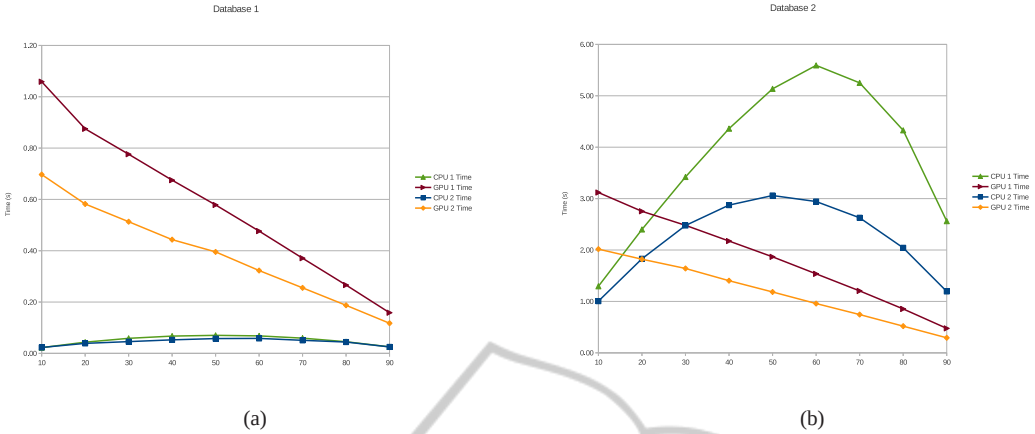
Additionally, two distinct CPU-based equipments have been used: CPU 1 and CPU 2, which have the same configurations of GPU 1 and GPU 2, respectively. The difference is that we did not employ the graphics cards, since they have ran traditional OPF. In regard to the operational system, we have used Ubuntu Linux 12.04.1 x86_64 LTS for both scenarios.

We have used datasets with different number of samples and features in order to evaluate the robustness of GPU-based OPF classification step. Database 1 and 2 refer to non-technical losses identification in power distribution systems and automatic Pterygium recognition tasks, respectively, being private datasets. Databases 3 and 4 concern with two public available hyperspectral remote sensing images: Indian Pines (Landgrebe, 2005) and Salinas (Kaewpijit et al., 2003). In this case, each pixel is described by the brightness of each spectral band. Table 1 shows the information about datasets.

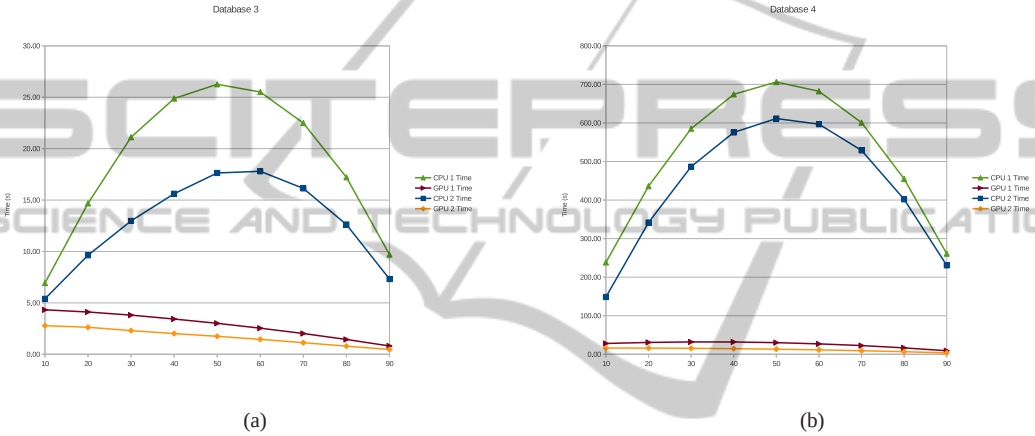
Table 1: Description of the datasets.

Database	# of nodes	# of features
Database 1	4952	8
Database 2	15302	89
Database 3	21025	220
Database 4	111104	204

In regard to the experiments, the datasets were partitioned in two parts: one for training and another one for testing phase. We used different set sizes in order to plot a curve of performance, starting from a training set with 10% of the original dataset and increasing it until 90% with a 10% step. The remaining dataset on each step has been used as the testing set. For each step size, the experiments were executed 10

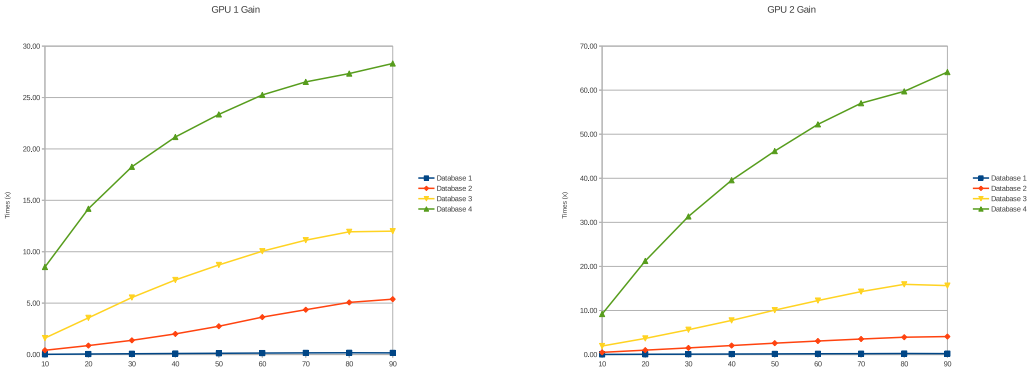


(a) (b)



(a) (b)

Figure 1: Performance curve for (a) Database 1, (b) Database 2, (c) Database 3 and (d) Database 4.



(a) (b)

Figure 2: Performance gain for (a) GPU 1 and (b) GPU 2.

times with training and testing sets randomly generated (cross-validation). Figure 1 displays the performance curve for all datasets. Tables 2 and 3 display the mean accuracy and execution times (seconds) for Computer 1 and 2, respectively.

The reader can note GPU 2 has outperformed GPU 1 and both CPUs for Databases 2, 3 and 4. For Database 1, the trade-off between GPU gain and throughput makes it inviable to the proposed algorithm, since Database 1 is not large enough. In regard to Database 2, the GPU gain can be observed at 20% of training set size. Figure 2 presents the gain results of the proposed approach for both scenarios, i.e., GPU 1 and GPU 2. One can see it is possible to achieve

Table 2: Mean accuracy and execution times (s) for Computer 1.

Data	Database 1	Database 2	Database 3
CPU Time	5.79 ± 0.29	26.13 ± 0.61	715.63 ± 4.31
CPU Acc.	56.07 ± 0.65	70.91 ± 0.73	90.98 ± 0.12
GPU Time	1.91 ± 0.05	3.05 ± 0.04	30.56 ± 0.33
GPU Acc.	55.97 ± 0.63	70.91 ± 0.73	90.99 ± 0.12
GPU Gain	3.03x	8.57x	23.42x

Table 3: Mean accuracy and execution times (s) for Computer 2.

Data	Database 1	Database 2	Database 3
CPU Time	3.03 ± 0.03	17.46 ± 0.12	613.30 ± 2.74
CPU Acc.	56.02 ± 0.77	70.92 ± 0.62	91.08 ± 0.10
GPU Time	1.18 ± 0.01	1.75 ± 0.01	13.19 ± 0.04
GPU Acc.	56.00 ± 0.79	70.92 ± 0.62	91.08 ± 0.10
GPU Gain	2.57x	9.98x	46.50x

very interesting results with the proposed approach, which can be better observed in large datasets.

5 CONCLUSIONS

This work presented a massive parallel approach for OPF testing phase, A parallel operation called matrix association, which can be seen as a generalization of a matrix multiplication procedure, has been proposed to consider this kind of data structure in order to maximize the gain of GPUs. Experimental results have shown the performance gain of the proposed approach in 3 out 4 databases, being the worst result in the smaller database, which highlights the main usage of GPU-based algorithms in applications that require a large volume data.

It is worthy noting that GPU 1 configuration can be found at $130 by the time this article was submitted, meaning that to develop parallel pattern recognition application is not exclusive only for those with expensive equipments, even with large datasets.

ACKNOWLEDGEMENTS

The authors are grateful to FAPESP grants #2009/16206-1, #2010/12697-8 and #2011/08348-0, and CNPq grants #470571/2013-6 and #303182/2011-3.

REFERENCES

Allène, C., Audibert, J. Y., Couprie, M., Cousty, J., and Keriven, R. (2007). Some links between min-cuts, optimal spanning forests and watersheds. In *Proceedings of the International Symposium on Mathematical Morphology*, pages 253–264. MCT/INPE.

Catanzaro, B., Sundaram, N., and Keutzer, K. (2008). Fast support vector machine training and classification on graphics processors. In *Proceedings of the 25th international conference on Machine learning*, pages 104–111, New York, NY, USA. ACM.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms*. The MIT Press, 2 edition.

Duda, R. O., Hart, P. E., and Stork, D. G. (2000). *Pattern Classification (2nd Edition)*. Wiley-Interscience.

Gainaru, A., Slusanschi, E., and Trausan-Matu, S. (2011). Mapping data mining algorithms on a GPU architecture: a study. In *Proceedings of the 19th international conference on Foundations of intelligent systems*, ISMIS'11, pages 102–112, Berlin, Heidelberg. Springer-Verlag.

Harris, M. (2010). Optimizing parallel reduction in CUDA.

Jang, H., Park, A., and Jung, K. (2008). Neural network implementation using CUDA and OpenMP. In *DICTA '08: Proceedings of the 2008 Digital Image Computing: Techniques and Applications*, pages 155–161, Washington, DC, USA. IEEE Computer Society.

Kaewpijit, S., Moigne, J., and El-Ghazawi, T. (2003). Automatic reduction of hyperspectral imagery using wavelet spectral analysis. *IEEE Transactions on Geoscience and Remote Sensing*, 41(4):863–871.

Landgrebe, D. (2005). *Signal Theory Methods in Multispectral Remote Sensing*. Wiley, Newark, NJ.

Oh, K. and Jung, K. (2004). GPU implementation of neural networks. *Pattern Recognition*, 37(6):1311–1314.

Papa, J. P., Falcão, A. X., Albuquerque, V. H. C., and Tavares, J. M. R. S. (2012). Efficient supervised optimum-path forest classification for large datasets. *Pattern Recognition*, 45(1):512–520.

Papa, J. P., Falcão, A. X., and Suzuki, C. T. N. (2009). Supervised pattern classification based on optimum-path forest. *International Journal of Imaging Systems and Technology*, 19(2):120–131.