Fast and Parallel Genetic Algorithm-Based Feature Selection

Jichuan Zeng
Department of Computer Science and Engineering
The Chinese University of Hong Kong
jczeng@cse.cuhk.edu.hk

Jian Li
Department of Computer Science and Engineering
The Chinese University of Hong Kong
jianli@cse.cuhk.edu.hk

ABSTRACT

This project aims to explore the fast and parallel implementation of Genetic Algorithm (GA) for the problem of image feature selection, which is a fundamental task in image classification. We explore several kinds of parallel framework for GA, e.g. Master-Slave model, Island model and hybrid model. We analysis the performance of our parallel framework in the task of image feature selection problem. And finally, we provide a easy-to-use Julia toolkit for user to develop their own parallel GA algorithms.

Keywords

Genetic algorithm, Feature Selection, Parallel implementation

1. INTRODUCTION

Since the emerge of Big Data, search engine, social network and e-commerce, become indivisible parts of people daily life. Among those big data applications, imaging processing, especially feature extraction, play an important and fundamental role. High dimension feature set will degenerate the performance of image recognition system, since some of the features may be redundant and non-informative [1]. Tradition methods such as Principal Component Analysis (PCA), needs a lot of work on parameter tuning and not scale well in large datasets.

Evolution algorithms have shown to be effective in solving complex optimization problems in real-world applications. However, with the rapid development of information technology and the trend of Big Data, the increasing data volume and critical demand of read-time analysis bring a new challenge to evolution computing. Genetic algorithm (GA), is a particular type of evolution algorithm. Generally, GA is hard to provide the coverage rate and sometimes very slow in training when the search space is huge. Traditional parallel strategy to improve the runtime of GA is to evaluate the fitness of population members in parallel. This strategy is usually fruitless because of communication overhead in transferring the populations. And other kinds of parallel strategy, such as island model, cellular model are hard to implemented for machine learning researchers.

Hence, in the project, we focus on the GA-based feature selection with fast and parallel implementation. In general, we follow the idea of feature selection in [2], and implement several parallel strategies of GA algorithm, e.g. master-slave model, island model and hybrid model, for image feature selection. Then we analysis the performance of each model and provide an easy-to-use toolkit in Julia language.

SAC'15 April 13-17, 2015, Salamanca, Spain.

2. PREVIOUS WORKS

Feature extraction is the first and most critical step in image classification task. To enhance the classification accuracy, most existing research focus on optimizing the classification algorithms (e.g., SVM, KNN), with all the primitive features. However, high dimensional feature set can negatively affect the performance of pattern or image recognition systems. Tradition dimension reduction methods such as Principal Component Analysis (PCA), needs a lot of work on parameter tuning and not scale well in large datasets. GA has been known to be a very adaptive and efficient method of numerical optimization. As for feature selection, GA also show its promising characters such as user-defined functional configuration, which helps to improve the accuracy.

Evolution algorithms, including GA, conventionally has two types of distributed parallel strategies: population parallel and dimension parallel. These two parallel strategies divide individuals of population or problem dimensions (subspace) into multiple processor or computing nodes. The population parallel model can be further divide to master-slave, island (coarse-grained model), cellular (fine-grained model), hierarchical model, and pool models.

Master-slave model summarizes a distributed approach to the EA operations and domain evaluations. As the evaluations of individuals are mutually independent. The master-slave model is hence simple, in which communications only occur when the unique master sends individuals to slaves and the slaves return the corresponding fitness values back to the master in each generation.

An island model divides the global population is divided into several subpopulations, each of which is processed by one processor. Communications between the islands occur when the several best individuals in one island migrate to another at a specific interval of generation. Island model

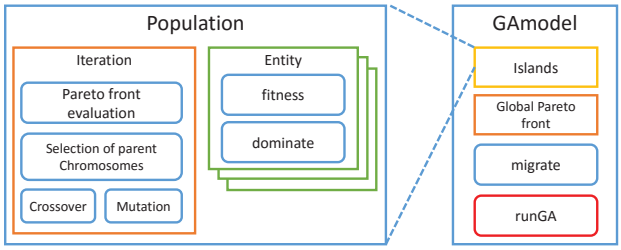

Figure 1: System framework

is spatially distributed model for parallel computation in evolutionary algorithms that largely enhance the usage on each available core and has better global exploration ability.

3. METHODOLOGY

3.1 System Overview

The framework of our automatically parallel GA-based feature selection system is shown in Fig. 1. In this framework, GAmodel controls the parallel execution of the system, which will be discussed more specifically in section 3.3. Each island contains a population, population follows the complete GA update, that is, initialization, evaluation, selection, crossover and mutation. Population control and evaluate the fitness of population members in parallel by through master-slaves model. Hence, our automatically parallel GA system leverages a hierarchical parallel framework, that is island-master-slave hybrid model. We have to mention that, this is a general purpose parallel GA framework with highly flexibility and scalability, and feature selection is just an application building on it. In this framework, user can define the Entity, fitness function, dominate function, crossover and mutation function themselves and leave everything else to the system.

3.2 Feature Extraction via GA

In this project, we developed an evolutionary learning methodology to automatically generate feature combinations for classification using genetic programming (GP). More specifically, given a group of primitive features, our GP method evolves to randomly combine a subset of them into feature descriptor, which are then fed into a KNN classifier. The evaluation criteria for each generation is the corresponding classification accuracy. After the whole evolution procedure, we can obtain the best-so-far feature descriptor. The Feature Subset Selection (FFS) is a map function Fs from an m-dimensional feature space (input space) to an n-dimensional feature space (output), which can be formulated as following

$$Fs: \ \mathbb{R}^{r\times m} \rightarrow \mathbb{R}^{r\times n} \ m \geq n \qquad (1)$$

Where $R^{r\times m}$ is the input database or matrix, containing the original feature set with r instances, $R^{r\times n}$ is the output matrix after the subset selection, containing the reduced feature set with r instances.

Based on the description above, the following optimization problem will be solved.

Given $n \in \mathbb{R}^{+}$, $n \leq m$ and $0 \leq \varepsilon \leq r$, where

(a) n is number of features in the reduced feature set.

(b) ε is the 0-1 classification error.

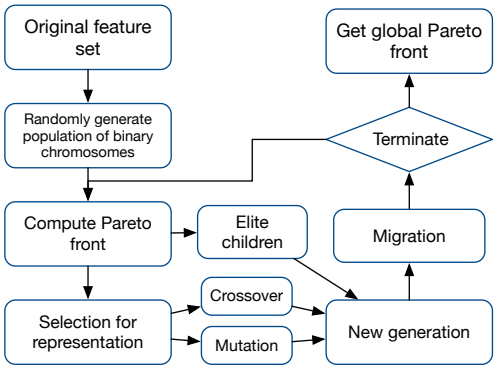

Figure 2: Algorithm workflow

Try to find a subset of feature mapping Fs, such that the objective ε and n are minimized.

In our method, the GA operates on binary search space as the chromosomes (individuals) are bit strings. In each binary chromosome, gene "1" denotes that the particular feature at the same indexed position is selected. Otherwise (i.e., gene "0"), the feature is not selected to form a subset for evaluation. The overall workflow of our algorithm design is shown in figure 2

3.3 Parallel Implementation

We first implemented master-slave model of GA because of its simplicity. In master-salve model, population are distributed into slaves After evaluation, fitness values are summarized from slaves to master, which is very efficient in the situation of computation-intensive tasks, for example, our KNN fitness function. But the master-slave model suffer the problem of the communication bottleneck and convergence of local optima solution.

Hence, We implement the island model of GA. An island model is coarse-grained, where the global population is divided into several subpopulations, each of which is processed by one processor. Our implementation is synchronous, that the best individual on each island propagates to all the other islands at a specific interval of generation. Within the time interval between communications, individuals on different islands can evolve with different directions. The island model has better global exploration ability but converges much slower when the communication frequency is lower.

In order to boost the runtime of island model, we use a hierarchical model, that is island-master-slave hybrid model as illustrated in Fig. 3. In this model, the evaluation tasks are distributed into slave processors, which can handle the computation-intensive tasks very well while keeping the variety of the population.

Julia is a programming language receiving more and more attentions recently. Julia's high performance, easy vectorization, access to low-level data types and user-friendly parallel implementation make it an ideal language for developing a GA for feature selection. Hence, we will use Julia to implement our whole system.

4. EVALUATION

In order to give a comprehensive evaluation of our framework. We evaluate our automatically parallel GA-based fea-

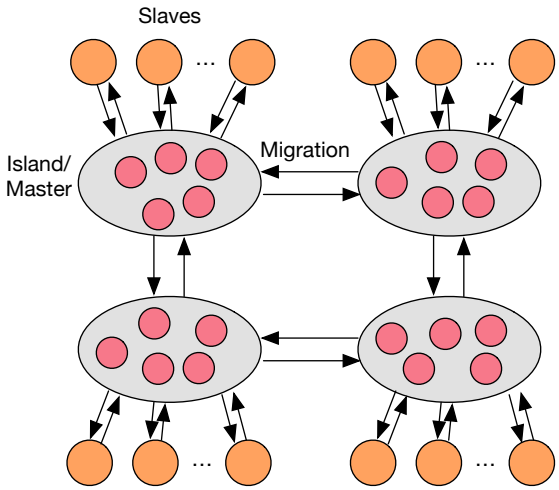

Figure 3: Island-master-slave hybrid model

P-GA Parameters	Value
Population Size	100
Genomelength	34
Population type	Bit strings
Fitness function	KNN-based classification error
Number of generations	100
Crossover	Arithmetic Crossover (XOR)
Mutation	Uniform Mutation
Mutation probability	0.1
Selection scheme	Tournament of size 2
Elite Count	2
Crossover fraction	All the Prato front
Migration probability	0.2
Migration number	2
Tolerance	0.01

Figure 4: Parameter choosing

ture selection system from several aspects: effectiveness, efficiency, scalability and availability. The data we use is the ionosphere dataset form the University College London [3]. This dataset comprises of 351 observations and 34 attributes with binary class information (bad radar and good radar returns). we use 80% of data as training set, 20% for test set. 5-fold cross validation is used in our kNN classification. And our choosing of parameters is shown in Fig. 4

4.1 Effectiveness

We first test the effectiveness of our algorithm by using Matlab code. Fig. 5 shows the convergence curve of GA-based feature selection method. Our GA terminates at generation 101. From the simulation diagram we can see that the fitness function almost remains constant from generation 38 to 101. The final feature set selected by GA is (3, 4, 5, 8, 15, 21, 27), with a mean classification error 0.0015.

4.2 Efficiency

We compare our Julia implementation with Matlab GA toolkit, From Fig. 6, we found that our system is at least 4X faster than Matlab in both single processor environment. Moreover, our system mainly focuses on the parallel execution of GA algorithm, which Matlab GA toolkit can not handle.

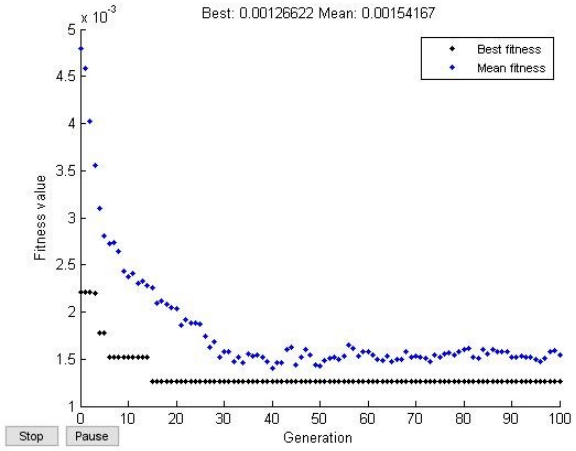

Figure 5: Convergence rate

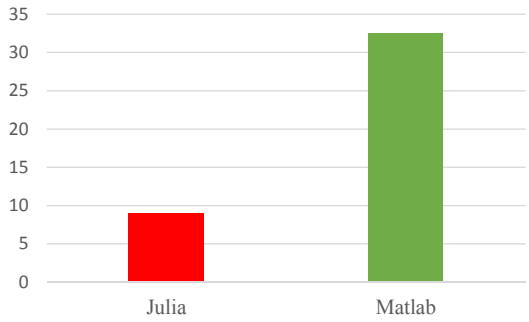

Figure 6: Run-time (sec) of Julia implementation and Matlab GA toolkit

4.3 Scalability

We test our parallel in different number of processors and islands. The speed-up is computed basing the perfect solution which means fitness is within the tolerance rate (See Fig. 4). Fig. 7 shows the speed-up under different number of processors, given the fix island number (Here we set 3 islands). Fig. 8 shows that, with the number of islands increases, the run-time of our system is also increasing. We will discuss this issue in later section.

4.4 Availability

One of the purpose of our project is to provide a easy-to-use programming interface for users to develop their own parallel GA algorithms. Our system is designed to be a general purpose GA framework, supporting user-defined fitness function, dominate function, selection, crossover and mutation strategy. We try our best to separate the application and system parallel framework. Our task, feature selection, is a light-weigh application built on our system. And we also provide a toy application of linear regression to help user to quickly enter our system. Fig. 9 give a example of how to define GA operators using our system.

Basically, there are several interfaces that has to be defined by user.

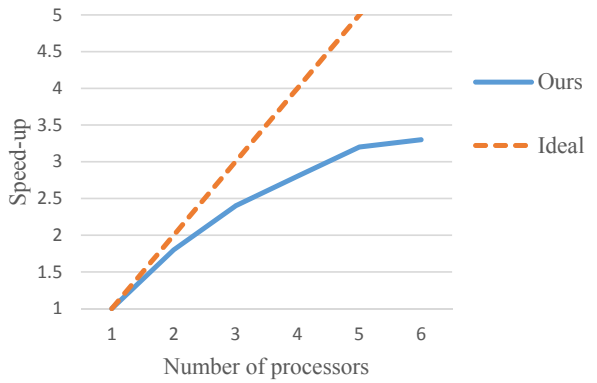

Figure 7: Speed-up under different number of processors

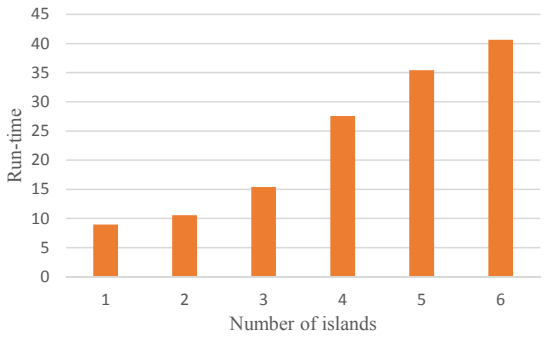

Figure 8: Run-time of different number of islands

- *User-defined_Entity*: type extends from abstract type *Entity*, containing *fitness*, *dominates* function and user-defined data structure.
- *create_entity(num)*: create entities according to the given number.
- *fitness(ent)*: fitness function, return the fitness value of given entity.
- *group_entities(pop)*: representation selection function, given the a subset of population, return the selected entity.
- *crossover(group)*: crossover function, receiving the selected group, performs crossover operation among them and returns a resulted entity.
- *mutate(ent)*: mutation function mutates the given entity.

5. DISCUSSION

5.1 Feature Selection Genetic Algorithm

Experiments on Ionosphere dataset shows that, our feature selection genetic algorithm is effective in the image classification task. Comparing to PCA-like feature reduction which completely change the global feature space, our algorithm keeps the space information, reserves the structure of original feature and has meaningful interpretation for feature selected. And our hybrid model has show good ability at finding optimal solutions than traditional GA. We also found that the Prato front is not very suitable in feature selection task, because it will easily reduce to size-one Prato front set without much varieties. We havenâĂŹt compare our algorithm to other feature selection method, such as deep NN, but it is already a part of our future work.

```
function fitness(ent)
  score = ent.abcde[1] + 2 * ent.abcde[2] +
    3 * ent.abcde[3] + 4 * ent.abcde[4] + 5 * ent.abcde[5]
  abs(score -42)
end

function group_entities(pop)
  for i in pop.paretoFront
    produce(i)
  end
end

function crossover(group)
  child = []
  count = length(group)
  sizehint!(child, length(group[1].abcde))

  parent1 = group[rand(1:count)]
  parent2 = group[rand(1:count)]

  # do crossover
  TestEntity(child)
end

function mutate(ent)
```

Figure 9: User-defined GA operators

5.2 Parallel Implementation

From Fig. 7 we can see that if the number of processor is larger than 4, the speed-up gain is tiny. Because we use all the processors to compute the fitness of population of an island and an islandś update is managed by one processor. So if the number of processor increases, the message containing population and fitness value will flood between each islandś processor and the rest processors, and the communication overhead will become a crucial issue if the the number of processors increases. This is much obvious if we increase the number islands. We can see from Fig. 8 that, with the increasing number of islands, the run-time is also increasing because of the communication overhead.

6. CONCLUSION AND FUTURE WORK

In this project, we developed a GA-based parallel system for image feature selection. Our hybrid parallel framework combines the advantages of island model and master-slave model. Our system exhibits high performance and good scalability. We also provide a easy-to-use Julia toolkit for user to develop their own parallel GA algorithms. In the future, we want test our GA-based feature selection parallel system on large data, such as ImageNet. Other popular feature selection methods, e.g. PCA, deep neural network should be compared in the future. Since time limits, we mainly do our evaluation in multi-processor single machine environment. We will also deploy our system on server clusters to find out the communication bottleneck of our algorithm in real distributed environment.

7. ADDITIONAL AUTHORS

8. REFERENCES

[1] L. Bruzzone and C. Persello. A novel approach to the selection of robust and invariant features for

classification of hyperspectral images. In *IEEE International Geoscience & Remote Sensing Symposium, IGARSS 2008, July 8-11, 2008, Boston, Massachusetts, USA, Proceedings*, pages 66–69, 2008.

[2] Y. Gong, W. Chen, Z. Zhan, J. Zhang, Y. Li, Q. Zhang, and J. Li. Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. *Appl. Soft Comput.*, 34:286–300, 2015.

[3] V. G. Sigillito, S. P. Wing, L. V. Hutton, and K. B. Baker. Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Tech. Dig*, vol. 10:262–266, 1989. in.