Parallel Genetic Algorithm Based on the MPI Environment

Jing-Jun Zhang\(^1\), Wen-Juan Liu\(^2\), Guang-Yuan Liu\(^2\)

\(^1\)Department of Scientific Research, Hebei University of Engineering, Handan, China
\(^2\)School of Information and Electrical Engineering, Hebei University of Engineering, Handan, China
*corresponding author, e-mail: santt88@163.com*, liuwenjuangu@yahoo.cn, gyuanliu@163.com

Abstract

Current genetic algorithm require both management of huge amounts of data and heavy computation, fulfilling these requirements calls for simple ways to implement parallel computing. In this paper, serial genetic algorithm was designed to parallel GA; this technology appears to be particularly well adapted to this task. Here we introduce two related mechanism: elite reserve strategy and MPI. The first can increase the possible to get the optimal solution of the population, while the message passing interface MPI support is adding to form a new coarse-grain model of distributed parallel genetic algorithm. This new algorithm is tested by the classical and effective Knapsack problem, analysis shows that, the introduction of the parallel strategies can reduce the communication between different machines and the scheduling time of the heterogeneous system, thereby accelerate the traditional genetic algorithm search process, ultimately concluded that the parallel genetic algorithm is very promising and this framework could have a wide range of applications while maintaining good computational efficiency, scalability and ease of maintenance.

Keywords: genetic algorithm; parallel genetic algorithm; MPI

1. Introduction

The genetic algorithm is a kind of probabilistic search algorithms; it was created based on natural and species evolution and population genetics, advances in this filed have led to an explosive growth of biological information. It is particularly suitable for used to solve combinatorial optimization problems of the nearly optimal solutions [1]. The implementation of the genetic algorithm is a multi-point parallel iterative process, the solution of the problem can be expressed as a binary encoded string, before the start of the search job, we need to do the following work: first presents certain population size of the chromosome group, then according to the genetic mechanism of survival of the fittest, choose the more adapt to the environment of chromosome to execute reproduction, crossover, mutation operation, terminated until it meets certain performance indicators and the convergence condition, finally, obtained the optimal solutions or satisfied solution of the problem.

When we use genetic algorithm for the specific application, the probability of success depends on whether the population scale we choose is compatible with the difficulty of the problem-solving. Compared with other deterministic algorithm, since there are multiple search points in the solution space, we need to assessment every solutions to the search space simultaneously will inevitably bring about the defects of the computational overhead and poor search performance. Usually, large-scale data integration places high demands on processing, storage and querying. Therefore, through the serial genetic algorithm with the currently licensed PC or workstation can only solve small and medium-sized application tasks. For the ultra-large-scale problems, the work of evolution process moves slowly, it is difficult to achieve the engineering calculation speed requirements, through the multigrain co-evolution to accelerate the speed and realize the parallel execution of the traditional genetic algorithm have got more and more attention.

However, it will generate a lot of communication overhead if the classic serial genetic algorithm parallelization just though the PC cluster hardware links. Over the past few years, most of the developed parallel genetic algorithms are based on a variety of parallel cluster or LAN, so to some extent it does improve the speed, but if only select the speed-up ratio as the
most fundamental considerations may not be able to ensure the accuracy of the obtained solution. In order to better optimize the division of the various sub-populations, determine the parallel cluster system communication topology to make the best individual to join without adding additional communication costs, this paper joined the support of the MPI message passing interface, formed a new kind of distributed parallel genetic algorithm. This distributes process clustered environment offers a low cost, high performance approach to processing massive amounts of instance data.

2. Research Method
2.1. Parallelization of the genetic algorithm

Parallel genetic algorithm combined with high-speed parallel computing and genetic algorithms natural parallelism; immensely promote the research and application of genetic algorithm. The introduction of parallel processing from the perspective of global develop the parallelism of the populations in the process of evolution, it is not only accelerating the search process of the genetic algorithm but also enriching and maintaining the diversity of population due to the local features which caused by the expansion of population size and the processor on the sub-population, it effectively dodged the premature phenomenon of the serial genetic algorithm and thus improve the solution quality.

In the realization of the structure of Parallel genetic algorithm, it can be divided into master-slave model, the coarse-grain model, fine grain model 3 kinds [2]. The coarse-grain model PGA also called distributed parallel genetic algorithm, it has the most widely application used range among the three parallel genetic algorithm. The biggest difference between the parallel genetic algorithm and the serial genetic algorithm is the introduction of individual migration strategy in the executing process. It just simply increase some migration subroutine in serial genetic algorithm and each parallel computer node running a copy of the file then regular exchange of several best individuals. This strategy as far as possible to reduce the communication of parallelization process and strengthen the global search ability of genetic algorithms, finally, it provides mechanisms to support distributed processing over large data and then improve the convergence speed and guarantee the solution precision.

According to the different patterns of migration of the individual sub-groups, the coarse-grain model can be divided into two categories: Island Model and Stepping Stone Model [3], as shown in Figure 1.

The biggest difference between the island model and the steppingstone model is in the migration patterns. In the island model, migration can be executed in any sub-groups, while in the steppingstone model; migration is limited to the neighboring two sub-groups. For the island model, in order to reduce the sharp increase in the bus access conflict when the number of processors is guilt, in the literature [4], it presents a new central control node of the island model, in the Figure 2, it shown the diagram of the model.

In this manner, genetic information data is broken into smaller units and each computer node process its local copy of data, every sub-groups of processing implementation the local search for the corresponding sub-groups, these results are then combined to obtain the complete answer. Experiments have shown that it can get a better speedup, at the same time, accomplishing the masker task in the whole group of the global selection. It overcomes the
shortcomings of traditional coarse-grained model’s localized select cannot ensure that the global optimal solution is correct. Obviously, this centralized control island model has a simple and efficient advantage and is also easy to implement in the existing parallel hardware system. It has been successfully deployed within scientific computing and has been shown to be a highly reliable, scalable architecture [5].

2.2. Coarse-grained Parallel genetic algorithm based on the MPI environment

MPI is the abbreviation of the Message Passing Interface; it is a user interface to write parallel programs which base on message passing, it may be executed on any node in a cluster of machines. MPI as the most popular distributed storage parallel programming environment can be installed in the Windows NT/2000 platform, the MPI library function can be embedded and called directly without any modification in C, C++ and Fortran and other programming languages. The master-slave MPI program design according to centralized control of the island model: the main process is designed to the master task, its task is to find the best individual of the current generation and send it to every other sub-processes. For the sub-processes, they performed serial GA tasks and find the best individual, passed them to the master process in addition. This framework takes care of data partitioning, scheduling, load balancing and inter-machine communication. As shown in Figure 3, it is the implementation process of the master-slave parallel genetic algorithm.

![Diagram of the implementation process of the master-slave parallel genetic algorithm](image)

Figure 3. The implementation process of the master-slave parallel genetic algorithm

The main process read-in the data and information needed by the genetic algorithm initialization, then transfer the message to each sub-process and receive messages from the sub-processes, the message includes the best individual chromosome bit string and its fitness value produced by the sub-population genetic evolution. By means of the best individual migration between the master and slave processor, make the main processor “receiving pool” always keep the optimal individuals form the processor, the master process receives the best individual of the various sub-groups, elect the best fitness solution by comparing all the data then form the next generation of solvable groups. In the last of the algorithm, through the comparison of chromosomes fitness value of the elite individual in the “receiving pool” merges
together these values to form the final output, which is just the best individual. The use of the elite reserve strategy can guaranteed to seek the optimal solution.

The algorithm description of centralized control island model

I. Initialization sub-groups

In order to shorten the message length and reduce the communication overhead, we have chosen real number encoding way to do the coding of chromosomes. Above of all, randomly generated a certain number of initial sub-populations by the main process then concurrent delivery to each child process according to the island model.

II. Each sub-group grouping independently completed the evolutionary process

Each processor serial operation the basic genetic algorithm, implementation of the selection operator, crossover operator and mutation operator, then calculate the fitness of the current generation of sub-groups and select the appropriate offspring to replace the parent individual to the formation of a new generation of sub-groups.

III. Connection topology and the communication of the sub-groups

According to the connection topology between sub-groups(which established by the centralized control of the island mode),in this paper we selected a bidirectional ring communication topological ring way for each sub-group with the adjacent two sub-groups to exchange the individual process.

IV. The migration strategy of the sub-groups

Individual migration strategy is introduced in the implementation process of the coarse-grained parallel algorithm, migration operator is responsible for the exchange of individuals between the various sub-populations the main control parameters include the migration individual number, the migration intervals and the migration choice and replace mode.

i. the migration individual number

In order to enable the algorithm to get a better acceleration and reduce the competition conflict in the bus access between the processors the selection scale should be adapt to the number of processors in the parallel processing environment. Usually the number of migration of individuals is based on the absolute number or percentage of the given sub-population size, in order to ensure the migration of individual survival probability and benign restructuring with other completely different sub-populations individuals the migration individual number we choose is 2.

ii. the migration intervals

The migration individuals can be migrated within a generation or every several generations. Generally speaking, if the interval is too small, there will have resulted in increased the communication and synchronization costs between the processors and the convergence speed is slow; if the interval is too big, the better individual cannot be timely dissemination, as a result, it will weaken its role in guidance and unfavorable to get high quality solutions. Considering these two factors the migration intervals we choose is10.

iii. migration choice and replace

In the coarse-grain model, some choose the best individual in the sub-groups to the individual “immigrants”, while some use fitness proportion or randomly selected individuals as the “immigrants”. In this paper, we take the strategies of move out of the best individuals to replace the worst individuals in the immigrant population.

V. Communication mode

In the parallel environment the communication between different machines is the basic mechanism for the exchange of information between the various processor, there are synchronous and asynchronous communication mode between the parallel machines, most of the coarse-grained parallel genetic algorithm commonly used synchronous transfer mode.

VI. Termination conditions

In the genetic algorithm, usually, there are three determination of the termination conditions: first, default the maximum iterative algebraic then terminate the operation when it reaches the preset algebra; second, the algorithm executes the terminate operation when the fitness of the chromosome and the fitness of the colony reached a given threshold or there are
no significant improvement after a number of consecutive generations; third, all the
chromosomes in the population have the same fitness, i.e. the populations have already
convergence, execution terminates.

VII. Evolution to the end, update the population of the essence to get the optimal solution

The following functions are part of the process execution code
Set generation num to 0;
Initialize Population (generation num);
Evaluate fitness of Population (generation num);
While (not termination condition)
(Generation num++);
Select Parents (generation num) from population (generation num-1);
Apply crossover to parents (generation num) to produce offspring (generation num);
Apply mutation to offspring (generation num) to get population (generation num);
Apply migration to Population (generation num);
Evaluate Population (generation num);

3. Results and Analysis
3.1. The structure of the parallel environment base on MPICH under the Windows
platform
MPI is a messaging standard for the development of distributed storage parallel
computer and cluster systems [6]. MPI provide a platform-independent message transfer library
standards, the purpose is to achieve the portability of the program, it supports PC, workstations
and almost all parallel machines. The programs written by MPI can be applied to all operating
system platform without any modification, the MPI environment initialization and end process
positive as follows: before invoking the MPI routines, each process should be executed
MPI_INIT, invoke the MPI_COMM_SIZE to get the size of the default group, then invoke the
MPI_COMM_SIZE to get the logical number of the calling process in the default group. After
completion of these, every process can send messages to other nodes or receive the messages
of other nodes according to the necessary needs. in this process, the MPI_SEND and
MPI_RECV have a higher frequency of all the functions. At last, invoked MPI_FINALIZE to
eliminate the MPI environment when it is no need to call any MPI routines, all the process as
can be ended at this time.

In order to evaluate the performance of the coarse-grained parallel genetic algorithm,
we applied 4 PC machine constitute a heterogeneous C+MPI parallel simulation grid platform.
The hardware platform includes: three Microsoft Windows XP pc, a Microsoft Windows 7 pc, a 4
port D-LINE interconnect router; the software platform includes: Microsoft Visual Studio C++6.0,
Microsoft .NET Framework 2.0, MPICH2-1.4.1p1-win-ia32.

The specific environmental build process is as follows:
i. for every pc machine installed Microsoft.NET Framework 2.0;
ii. install VC6.0 on every pc machine;
iii. install MPICH2-1.4.1p1-win-ia32 on every pc machine;
iv. after the installation is complete, added the installation directory C:\Program
Files\MPICH2\bin in every PC system environment variable path.

For every PC machines participate in the parallel computing, it will not be used until the
MPICH is installed and the following registration and configuration work is successful, the
registered work should be carried out on every computer while the configuration just executed
on the master computer. After the completion of parallel environment, we can start to the
following job: first, create a new C++ project in the C environment, stored the MPI code in the
engineering of CPP file so that it can be directly compiled; afterwards, copy the compile debug
MPI code to every JOB directory in the parallel running PC then we can free to enter the CMD
doing any one machine to perform an online operation.

MPICH offers an open source implementation parallel framework and a distributed file
system on which the parallel framework can be run. Each node in the parallel environment can
act as a master or slave. There are 3 kinds of node types.

Name-node: it is the parallel environment master, it splits data information files into
blocks and distributes them across the cluster system with replication. The name-node holds all
metadata information and acts as an interface that allows sup-groups of immigration data information to be received and preservation.

Data-nodes: they are the parallel environment slaves. They physically store data blocks, serve the read or write requests from propagate replication tasks as directed by the name-node.

Task trackers: they are the parallel environment slaves. They execute their assigned data partitions, in this paper, each of the task nodes execute serial genetic algorithm and transfer the immigration information to the name-node.

It is easy to assume that the more computer nodes are the better calculation of capacity. This is not always true. In general, in any kind of parallel or distributed problem, each additional computer node presents some overhead in exchange for extra parallelism. Tata must be transferred to that node which must bring extra load on the data access system, whether it is a shared data transfer service. More parallelism means more concurrently running jobs for both the engine and the batch system to manage, and a heavier likelihood of a node failing, which increase the dispatch latency.

One of the key features of this MPICH parallel framework is its data storage model, which allows for high-level parallelization in data access. This parallel mechanism suits well bioinformatics applications which are typically characterized by large amounts of data that are not modified during a job run.

3.2. Instance of the test

In order to investigation the accuracy and efficiency of the parallel genetic algorithm settlement, using one of the classical genetic algorithm efficiency tests of the 0-1 knapsack problems to test. 0-1 knapsack problems defined as follows: given the n items of goods and a backpack, the weight of the items is \( w_i \), its value is \( v_i \).

![Graphs comparing the average fitness of different algorithms](image)

(a) The size of the project scale were 100  
(b) The size of the project scale were 250  
(c) The size of the project scale were 500

Figure 4. The comparison of these two algorithm average fitness
The capacity of the backpack is $C$. The purpose is to make the largest total value of all the items in the backpack, as in (1).

$$f(x) = \sum_{i=1}^{n} v_i x_i$$  \hspace{1cm} (1)

At the same time, it should meet the following requirements constraints, as in (2).

$$\sum_{i=1}^{n} w_i x_i \leq C, \hspace{1cm} x = (x_1, x_2, \ldots, x_n), x_i \in \{0,1\}$$ \hspace{1cm} (2)

If $x_i = 1$, it means that the project was selected into the backpack. The comparison tests were denoted by GA and PGA knapsack problem, the size of the project scale were 100, 250 and 500, the test results shown as Figure 4 (a), (b) and (c) showed.

It not difficult to draw the following conclusions: in a shared computing environment, the system has the capable of adjusting the system to serve new workloads as they are submitted. The comparison with the GA and PGA show that the parallel genetic algorithm improved the convergence speed of the genetic algorithm and shortened the given precision time-consuming. At the same time the introduction of the elitist mechanism has a better ability to solve the problems in the process of the evolution cheating cased by the optimal solution is destruction by the follow-up genetic operation.

4. Conclusion

The innovation points of this paper are: proposed a new parallel genetic algorithm which based on the elite reserve strategy, then the algorithm is applied to solving the knapsack problem. The experimental results had shown that: MPI as a parallel programming environment although itself does not provide parallel language, but by the means of parallel program design function, it can greatly reduce the rapidly increase bus access conflicts when the processor number is larger, thus speeding up the convergence rate of the parallel genetic algorithm. This simple, efficient centralized control model can effectively reduced the occurrence probability of premature phenomenon that caused by the locality selection of the traditional coarse-grained model, its good global searching ability ensure that the solution we got to meet the global optimality.

The MPI parallel framework described above requires that at least one of the data sets fits in its entirety on each computer node. Doing this requires significant network traffic, which can make distributed solving worse and has different access controls. Delays are particularly troublesome, this limits the number of compute nodes, perhaps to zero, available to a problem for large data sets. Because of the different computing node in the parallel environmental has different computing power, thus, the data transmission and processing must be done with more care, in addition, buffer management, concurrent control remains to be further research.

Acknowledgment

The authors gratefully acknowledge the supports of Natural Sciences Foundation of Hebei under the Project number F2011402028.

References

