The Implementation of Parallel Ant Colony Optimization Algorithm based on MATLAB

Wan Baocheng
Information Technology College, Jilin Agricultural University, Changchun 130118, China
Email: wanbaocheng@163.com

Wang Tiane
Information Technology College, Jilin Agricultural University, Changchun 130118, China

Wang Zenghui
Information Technology College, Jilin Agricultural University, Changchun 130118, China

***Abstract*—At first the relationship between the volume of the data transmitted and the transmission time is tested and the analysis of the data shows that there is a significant linear relationship between the two in MATLAB Distributed Computing Engine. Then we give an implementation solution of the parallel ant colony optimization algorithm , and we also carried on the computation of a TSP example which shows a higher speedup and a better performance. All these show that the it is efficient and effective to use MATLAB to develop distributed computing application program.**

***Keywords*-distributed computing; ant colony optimization; MATLAB**

I. Introduction

MATLAB is an outstanding scientific and engineering computing software owned by MathWorks Inc. It becomes the standouts in computing software fields because its high-speed calculation, reliable, rich features and convenience programming. With the popularity of computer networks, most computers are connected into a network, and distributed computing has entered the times of pc(personal computer). Developing distributed and parallel applications based on MATLAB would take full advantage of the rich functions of Matlab and greatly reduce the difficulty and cost of development, and highly improve efficiency[1]. Mathworks Corporation has timely promoted of the distributed computing engine and toolbox[2][3], moreover Matlab is a cross-platform product, and therefore Matlab actually constitute the distributed computing engine and toolbox in heterogenous environment. Message Passing interface (MPI)[4], the most important parallel program tool at present, has already become the industry standard in parallel programming. Matlab has also provided the basic support for MPI functions. The distributed computing engine and toolbox, the commercial product pro-moted by the MathWorks Inc, has some unmatched advantages compared with other matlab development environment[5].

As an important member in the Evolutionary Algorithms (EA) family and meta-heuristic optimization method, ACO shows good performance in a great deal of complex optimization problems[6][7][8]. Solving some practical issues will slow down the process of the sequential ACO due to more number of individuals and substantial calculation in need, this will make it is hard to meet the practical require-ment. Therefore parallel the ACO becomes a study hotspot[9][10].

Aimed at this study hotspot, we easily develop the parallel ACO make use of two toolbox together. The rest of the paper is organized as follows. Section 2 introduces some contents on MDCE. Section 3 gives a detailed description of the implementation process of the PACO followed by a practical example to test the validity of the algorithm. Finally, a brief summary is made.

II. Matlab Distributed Computing Engine

Before carrying out the distributed computing, we need to configure the computing environment. The basic process is: first, start the MDCE service in each computer involved, and then start Job Manager, start Worker leech on to one of the Job Manag-ers (Matlab Session involved in computing in the background will be started at the meanwhile). All the Workers leeched onto one of the Job Manage constitute a Matlab distributed computing environment, and several such environments can be constructed. Having started the Matlab in some computer, created a Job Manager example, we could carry out distributed and parallel computing take advantage of them.

The workflow is as follow: Discomposing the task into several and then submitted them to Job Manager, then Job Manager will appropriately distribute them for evalua-tion to workers according to the number of the workers and how many workers are available. After workers complete the tasks, results will be return to the Job Manager. After all the workers complete the tasks distributed to them, the job manager returns the results of all the tasks in the job to the client session.

Next, we will configure the Matlab distributed computing environment with two computers as an example. Similarly we can extend to more computers. We choose Windows XP SP2, node1 and node2 as our host computers' names. MDCE and tool-box are both version 3.1. Create a Job Manager on node1, then create a Worker in both nodes. Do

978-0-7695-4860-9/12 $26.00 © 2012 IEEE
DOI 10.1109/GCIS.2012.62

as follow: click the start button-¿run, type the command cmd, a DOS window will appear, then input the following commands:

(1) Install MDCE
mdce install
(2) Start MDCE
mdce start
(3) Create Job Manager
start jobmanager-name jm-romotehost node1
(4) Create Worker
startjobmanager -name w1 -jobmanager jm -jobmanagerhost node1 -romotehost node1
startjobmanager -name w2 -jobmanager jm -jobmanagerhost node1 -remotehost node2

Now, start the Windows Task Manager at the nodes, you will see both processes mdced.exe and matlab.exe in the process option card. Note that close the firewall in the process of configuration and running MDCE.

III. Realization of the parallel ACO

At present, there are two main types of parallel ACOs : fine-grained strategy and coarse-grained strategy. In this paper, we give a realization of a coarse-grained parallel ACO based on MATLAB. The description of the algorithm is as follows.

```
Algorithm PACO:
Randomly generate P ant colonies
Send each subcolony and the parameter
  values to all nodes
for K cycles
   dopar(on nodes)
      for loop=1 to P do
         Run sequentially ACO
      endfor
   endpar
   Send best individuals to the
     neighboring node
endfor(for cycles)
```

Parameters involved above are explained as follow.
P: number of nodes. K: times for the iteration of the sequentially ACO.

In order for convenient use, the paralleled ACO is realized in the form of function. The PACO function is as follow:

```
function solu = paco(jm, dis, options)
```

The first input parameter jm is a jobmanager object, input the following command in the MATLAB command window.

```
jm = findResource('scheduler',
  'configuration', 'jobmanager',
  'name', 'jm').
```

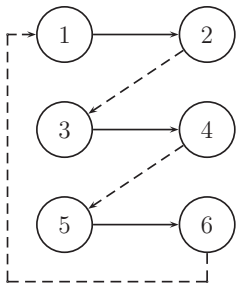

Figure 1. A Workflow of Distributed Computing

Parameter options could be attained from the function a-cooptimset. Two fields(P and K) will be added.

This function mainly creates a parallel task object and sets some parameters related. The key is the task pacos established on it, which is the parallel executed m function file placed on the workers of each node. Now submit the file to the parallel job object and run it. Then fetch all the results ,which are made some appropriate process and returned to the output parameters. Finally, the parallel job object was destroyed.

Note: To make the function pacos run on both nodes, the FileDependencies field of the parallel job object should be set first, and then temporarily send the paco to both nodes. Of course, copy the file pacos.m to both nodes advance will do, too. The form of the function is as follow:

```
function solu = pacos(dis, options).
```

This function is the core of the PACO , whose body is designed and developed by the PACO algorithm mentioned above and the MPI function provided by MATLAB. It mainly make use of the two MPI functions labSend and labRecieve. With circle structure, migration for the evolved subpopulation is made. Note that the sending command is standard form, which means that block occur or not depends on the state of the system. However, the receiving command is block form.

When a ring migration happens in the nodes, the number of the nodes should be even so that the blocking time can be reduced. In order to achieve that, the data should be transmitted as follow: as an example of six nodes, the data will be transmitted first between nodes linked by solid lines, then between nodes linked by dotted lines. No matter how many nodes there are, the total transmission time is nearly 2 times of single transmission time. Show in figure 1.

IV. Testing examples and Performance Analysing

Testing environment: Intel Core 2 Quad CPU Q8300 2.5GHz, the memory on the nodes that run jm is 2G, and the others are 1G.

First we test the relationship between transmission data volume and transmission time carried out by the MPI

Table I
TRANSMISSION DATA VOLUME AND TRANSMISSION TIME

transmission data volume(kb)	time(ms)	transmission data volume(kb)	time(ms)
703	51	1653	118
800	58	1800	128
903	65	1953	139
1013	73	2113	150
1128	81	2278	161
1250	89	2450	173
1378	99	2628	186
1513	108	2813	198

Table II
RESLUTS OF REGRESSION MODEL

coef	value	with 95% confidence bounds	R-square
p_1	0.06981	(0.06952, 0.07011)	0.9999
p_2	2.224	(1.705, 2.743)	

function labReceive provided by MATLAB. Transmis-sion time will be averaged between ten times. Datum are showed in Table I.

The model is $f(x) = p_1x + p_2$ using linear regression, x represents the volume of data transmitted, while $f(x)$ represents transmission time. The results returned by MATLAB are showed in table II.

The result indicates that the data transmitted by MATLAB obey a linear relation-ship. The increase 1 kb, about 0.07 millisecond will be cost. The linear relationship supplies a basis for the analysis of the performance of the MATLAB parallel program.

Take TSP Att48[11] as an example, distance of the shortest path is 33524 in this problem. PACO is runing on the cluster constituted by one computer, two computers,four computers and eight computers respectively.

In order to test the state of the executive time of each section of the algorithm, some points in paco are chosen. Start the stopping clock on each nodes where the implementation of paco begins, point before the migration of the population or after the evolution of the subpopulation, point after evolution of the population or before evolution of the subpopulation, point where the PACO ended on each node. The operation result statistic is showed in table III.

The total time consumed is about 2 seconds less than the paco consumed, as has nothing to do with the factors such as the number of nodes. It may be considered that the starting time of the whole parallel job is constant. Not including starting time, the paco speedup is showed in table IV.

Table III
TEST RESULTS OF PACO

Number of node	1	2	4	8
Optimum value	35031	34812	33829	33621
Total time(s)	8.6811	6.887	4.6532	3.6134
Pacos time(s)	/	5.4103	3.2951	2.1324
Start time(s)	/	0.9214	0.9189	0.9057

Table IV
SPEEDUP(WITHOUT STARTING TIME)

Number of nodes	speedup	speedup(without start time)
1	1	1
2	1.6	1.9
4	2.7	3.7
8	4.5	7.2

V. RESULTS AND DISCUSSION

From the passage above we can see that developing distributed and parallel algorithm based on MATLAB platform is very favorable for scientific and engineering computing applications users. Experiments are not carry out with relatively more nodes, where situation may change a little. Multi-nodes case is the future research direction.

ACKNOWLEDGMENT

Project supported by the Foundation for Youths of Jilin Province (Grant No. 201201095).

REFERENCES

[1] J. Kepner and S. Ahalt. MatlabMPI. Journal of Parallel and Distributed Computing: 2004,64, 997-1005.

[2] The Mathworks, Inc.: MDCE3.1 System Administrator's Guide.

[3] The Mathworks, Inc.: Distributed Computing Toolbox 3.1 User's Guide

[4] Message Passing Interface Forum: MPI: A Message-Passing Interface Standard. November 15, 2003. http://www.mpi-forum.org/docs

[5] J. P.Hoffbeck, M. Sarwar and E. J. Rix. Interfacing MATLAB with a parallel virtual processor for matrix algorithms. The Journal of Systems and Software:2001, 56, 77-80.

[6] M. Dorigo and T. Stutzle. Ant Colony Optimization. Bei Jing, China: Tsing Hua University Press, 2007.

[7] Duan Haibin. Ant colony algorithm: theory and applications. Bei Jing, China: Science Press, 2005.

[8] Li Shiyong. Ant colony algorithms with applications. Harbin, China: Harbin Institute of Technologys Press, 2004.

[9] B. Bullnheimer and G. Kotsis. Parallelization strategies for the ant system. In R.D. Linear Optimization: 1998, 24, 87-100.

[10] M. Middendorf and F. Reischle. Multi colony algorithms. Journal of Heristics: 2002,8(3), 305-320.

[11] . Jiao Licheng and Du Haifeng. Immune Optimization. Bei Jing, China: Sciense Press, 2006.