Efficiency of Parallel Graph Processing on Multicomputer Platform

Plamenka Borovska[1] and Milena Lazarova[2]

***Abstract* – The paper is aimed at investigation of parallel algorithms for finding minimum spanning tree of a graph. Prim's algorithm and genetic Prim's algorithms are regarded. The impact of cluster size and workload on the parallel performance are explored. Performance profiling and analysis are presented.**

***Keywords* – minimum spanning tree, genetic algorithm, parallel computational model, multicomputer.**

I. INTRODUCTION

The Minimum Spanning Tree (MST) problem is aimed at finding a least cost spanning tree in an edge weighted graph. The problem has been well studied and many efficient polynomial-time algorithms have been developed by Dijkstra, Kruskal, Prim [1, 2]. Finding of the MST is a task with a very wide application in the field of communication network design and communication network design problems.

Genetic algorithms (GA) are part of evolutionary computation techniques, which is a rapidly growing area of artificial intelligence. GAs have been successfully applied in many research fields as biogenetics, computer science, engineering, economics, chemistry, manufacturing, mathematics, physics.

The objective of this paper is to investigate the efficiency of parallel algorithms for finding minimum spanning tree in a graph. Prim's algorithm and genetic Prim's algorithm are regarded. Parallel computational models based on "manager-worker" parallel programming paradigm are presented. The algorithms are implemented using parallel MPI program. The target parallel platform is a multicomputer. The impact of multicomputer size and computational workload on the parallel performance parameters are explored and performance profiling and analysis are presented.

II. MINIMUM SPANNING TREE OF A GRAPH

A. Prim's algorithm

Let's consider undirected graph $G = (V, E)$ where:

- $V = \{v_1, v_2, \ldots, v_n\}$ is a finite set of vertices;
- $E = \{e_1, e_2, \ldots, e_m\}$ is a finite set of undirected edges.

Each element $e_k \in E$ ($k = 1, 2, \ldots, m$) is an ordered pair (v_i, v_j), $v_i, v_j \in V$, $1 \le i, j \le n$.

A graph is weighted when a function $f(i, j)$ relates an integer value (cost) to each edge $(i, j) \in E$ with $f(i, j) = f(j, i)$.

Minimum spanning tree (MST) is a tree $T(H, D)$ of the graph G with minimum sum of the weights of all edges included in the tree, where H is the set of vertices in the tree and D is the set of edges, connecting the vertices of the MST. Prim's algorithm is iterative algorithm for finding MST. It runs (n-1) iterations for a graph with n vertices. A vertex connected with the vertices already added to the tree by an edge with a minimum weight is selected at each iteration. The procedure of building MST according to Prim's algorithm is as follows:

Start from an arbitrary vertex s: H = {s}, D = Ø
Repeat (n–1) iterations
Choose edge (i, j) ∈ E such that
i ∈ H, j ∈ V/H and f(i, j) is minimum
Include the vertex j in H and edge (i, j) in D

B. Genetic algorithm of Prim

Genetic algorithms (GA) are metaheuristic method for global search and optimization that mimics the evolution of life organisms. The tree main principles of the natural evolution are applied to the GA: reproduction with variation, natural selection based on fitness and repetition with diversification of the individuals expressed by differences between a population and the next one. GA utilizes a population of data structure, called chromosome. Chromosomes represent possible solution of the regarded problem. Each chromosome has a fitness which is a numerical value indicating the quality of the solution the chromosome represents. The GA selects chromosomes to survive or reproduce so that those with better fitness are more likely to be selected. Crossover, also called recombination, combines genetic information from two parent chromosomes. Mutation randomly modifies one parent chromosome. When enough offspring is generated the parents are replaced and the process continues so that chromosomes that represent better solutions evolve.

GA algorithms are successfully applied to solve several graph and network problems including building of minimum spanning tree [3, 4, 5].

In order to apply genetic algorithm to the problem of finding MST the following issues are considered:

- the GA will generate only MST of a given graph;
- the correspondence between chromosomes and spanning tree will be 1:1;

[1]Plamenka Borovska is with the Faculty of Computer Systems and Control, Department of Computer Systems, 8 Kliment Ochridsky str., 1756 Sofia, Bulgaria, E-mail: pborovska@tu-sofia.bg

[2]Milena Lazarova is with the Faculty of Computer Systems and Control, Department of Computer Systems, 8 Kliment Ochridsky str., 1756 Sofia, Bulgaria, E-mail: milaz@tu-sofia.bg

- chromosomes' coding, crossover and mutation operators will not depend on the problem size.

Several spanning tree representations and chromosome encodings of the possible solution of MST of given graph for GA exists [6, 7, 8]. The coding of a spanning tree of a graph for GA is based on a representation of the spanning trees as strings of numerical weights associated with the graph's vertices.

The operations used for GA for finding of MST of given graph are as follows:

- Initialization – in order to guarantee that the solution found by GA will be correct, a random population of trees is generated at the initialization stage of the GA algorithm. The generated trees have to be spanning trees of the given graph and in order to satisfy that requirement vertices and edges are added to the tree according to Prim's algorithm but without observing the requirement that the edge added is minimal. The MST will be searched in the generated population of spanning trees.
- Selection – the selection is performed based on the roulette wheel selection rule which states that parents are selected according to their fitness. The weight of each generated tree is calculated and the total sum of all weights of population is found. A random number is generated between 0 and the total sum of the weights in the population.
- Crossover – for the crossover operator the following idea is applied: the solution after crossover has to inherit as many of the parents' edges as possible. At first step the new spanning tree takes the edges that are common for both two parents. At the next step edges belonging to only one of the parents are checked if they can be included in the new solution without making cycles with the edges that have already been added. At this step edges are again randomly added to the new spanning tree as was during the initialization stage. The addition of edges stops when a full tree is built, i.e. the number of added edges is one less than the number of added vertices.
- Mutation – mutation is based on addition of randomly selected edge and deletion of an edge that lies in the cycle between the added edge and the other tree edges, going out of the vertices when the edge was added. In brief the mutation stage is as follows:
  - a random edge connecting two vertices is selected to be added to the tree;
  - a set of edges that lie between the two vertices is constructed; the edges in this set together with the new added edge make a cycle;
  - an edge from the cycle is selected to be deleted;
  - the new edge is selected while the old one is deleted.

III. Parallel algorithms for minimum spanning tree

A. Parallel computational model of Prim's algorithm

There are several possibilities for parallel calculations in the sequential Prim's algorithm for finding MST of a graph on a multicomputer platform:

- At each iteration of the algorithm finding the minimum weighted edge connecting a vertex not included in the spanning tree so far and a vertex that is part of the spanning tree – since the problem is solved utilizing several processors each processor can perform a local search for minimum weight edge only between part of the set of graph vertices and then by global reduction operation the global minimum is obtained. The result is the minimum weight of an edge to be added to the spanning tree at the current iteration. This edge connects two vertices – the first one is already part of the spanning tree built so far and the other one is the new added vertex. The processor that has found the minimum weight edge broadcasts the number of new added vertex to all other processors.
- Preparation for the next iteration – update of the edges of minimum weight connecting each vertex not included in the spanning tree and the vertices in the spanning tree. Each processor works on a strip of the graph adjacency matrix and calculates the updated minimum weight edges for part of the vertices out of the spanning tree.

Data decomposition is utilized for parallelization of the Prim's algorithm for finding MST. The set of vertices, from which a vertex to be added to the current spanning tree is selected, is distributed among the parallel working processes. Utilization of data parallelism has the advantage of load balancing between the processors. Each of *p* parallel working processes operates on either $\lceil n/p \rceil$ or $\lfloor n/p \rfloor$ vertices of total n vertices in the graph. The upper and lower indices of the strip of each process are easily calculated.

Parallel computational model of Prim's algorithm for multicomputer platform is based on the parallel algorithmic paradigm "manager-worker" (fig.1).

The manager process is responsible for the following activities:

- Sends the graph adjacency matrix to all worker processes (function MPI_Bcast);
- Sends the vertex number of the last vertex added to the spanning tree to all worker processes (function MPI_Bcast);
- Gathers values of the minimum weight edges to a vertex from the spanning tree locally calculated by the worker processes (function MPI_Gather);

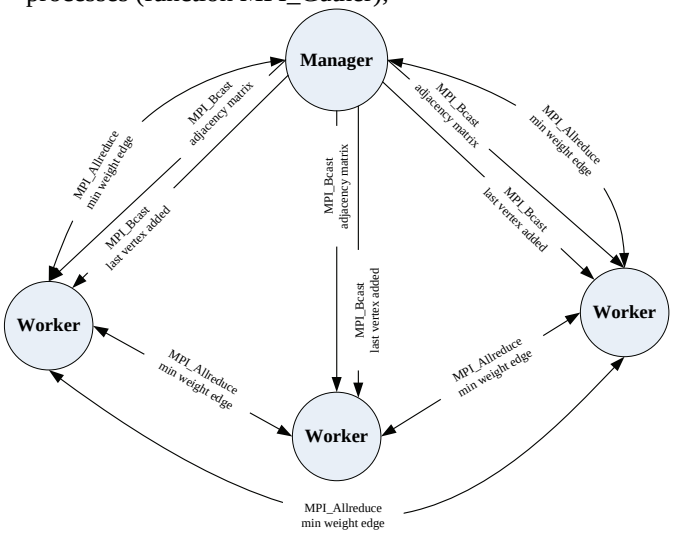

Fig.1. Parallel computational model of Prim's algorithm

- Selects a vertex to be added to the current spanning tree on the base of the results received by the worker processes.

Each worker process performs the following tasks:

- Receives the graph adjacency matrix by the manager process (function MPI_Bcast);
- Receives a vertex number of the last vertex added to the spanning tree (function MPI_Bcast);
- Working on the local set of graph's vertices, for each vertex determines an edge of a minimum weight connecting the vertex with a vertex in the spanning tree;
- Sends the calculated values to the manager process (MPI_Scatter).

B. Parallel computational model of genetic Prim's algorithm

Finding the solution of a specific problem by GA is inspired by the natural evolution and is a native parallel process that can be easily parallelized. Three main approaches of parallelizing GA exists:

- Common population, parallel calculation of the cost function of each individual by a worker process and genetic operators performed by the manager process. This approach is useful when the calculation of cost function is time consuming.
- Separation of the population into subpopulations and calculations for each subpopulation by a different process. This approach provides independent population evolution and thus higher diversification of the movements in the search space.
- Distribution of the genetic operations to different processes. This approach is an extension to the above one since genetic operators are applied only to consecutive populations and thus high diversification of the individuals is kept.

The purpose of parallelization of the GA algorithms is to reduce the time of finding the solution.

The presented genetic Prim's algorithm uses the third approach for parallelization. The crossover and mutation operators are carried out at the worker processes while initial population generation and selection are executed by the manager process.

The parallel computational model of genetic Prim's algorithm is based on parallel programming paradigm "manager-worker" (fig.2).

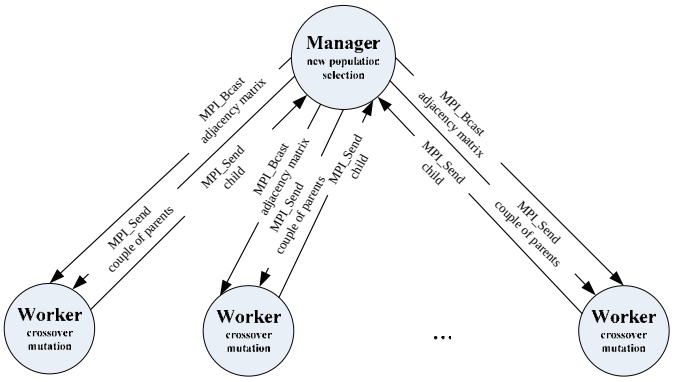

Fig.2. Parallel computational model of genetic Prim's algorithm

The manager process performs the following activities:

- Sends the graph adjacency matrix to all worker processes (function MPI_Bcast);
- Generates new population;
- Performs selection operation to select a couple of parents for a crossover operator;
- Sends the selected couple of parents to each worker process (function MPI_Send);
- Waits for a result of the crossover by any worker process (MPI_Probe);
- Receives the result from a worker process and sends to it new parents for crossover;
- Gathers all results from all worker processes for a given population
- Starts next population and repeats the above activities;
- Sends termination signals to all worker processes when the above activities are completed for all populations;

The tasks of each worker process are as follows:

- Receives the graph adjacency matrix from the manager process (function MPI_Bcast);
- Waits for a message from the manager process;
- When a message from a manager is received checks if it contains termination signal and if so then terminates;
- If the received message is not a termination signal interprets it as signal to start the crossover and mutation operators;
- Receives a couple of parents by the manager process;
- Performs the crossover of the two parents;
- Sends the result to the manager process;
- Waits for a new message;
- Performs mutation;
- Sends the result to the manager process.

B. Experimental framework

The experimental study of the parallel algorithms for finding MST described in the previous section is performed on multicomputer platform comprising 5 workstations (Intel Pentium IV 1.5GHz, 256MB RAM) connected by Fast Ethernet 100 Mbps switch. For the implementation of the algorithms Microsoft Visual Studio 2005 and MPICH-2 are used.

The speedup and efficiency of Prim's algorithm applied for finding MST of a graph with 1000 nodes are shown in fig.3 and fig.4 respectively. Gantt's diagram and communication transactions profiles of the Prim's algorithm are shown in fig.5 and fig.6 respectively.

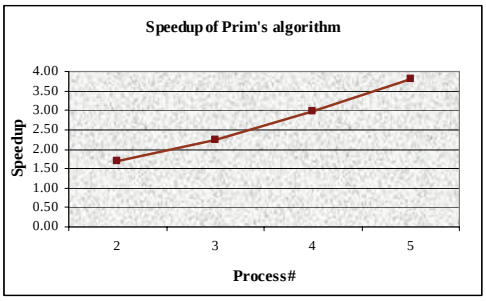

Fig.3. Speedup of parallel Prim's algorithm for finding MST on multicomputer platform

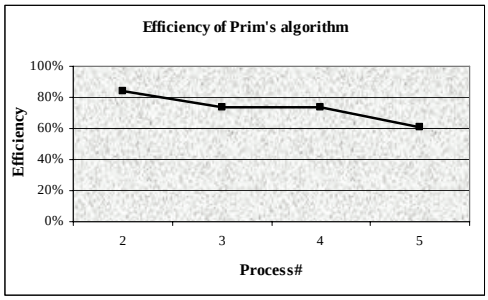

Fig.4. Efficiency of parallel Prim's algorithm

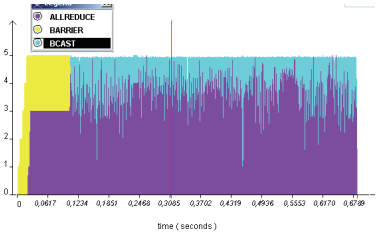

Fig.5. Communication transactions profile of parallel Prim's algorithm

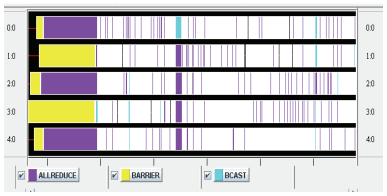

Fig.6. Gant's diagram of parallel Prim's algorithm

The speedup and efficiency of genetic Prim's algorithm applied for finding MST of a graph with 1000 nodes are shown in fig.7 and fig.8 respectively. Gantt's diagram and communication transactions profiles of the genetic Prim's algorithm are shown in fig.9 and fig.10 respectively.

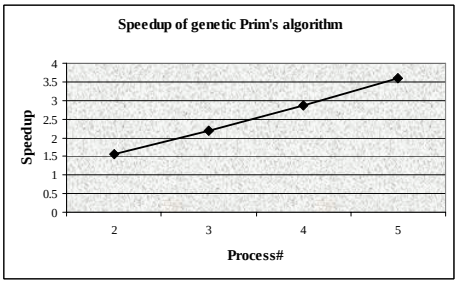

Fig.7. Speedup of parallel genetic Prim's algorithm

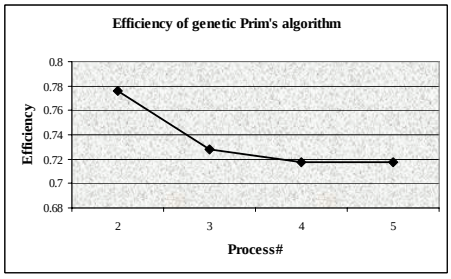

Fig.8. Efficiency of parallel genetic Prim's algorithm

The experimental results show almost linear speedup and high efficiency of the parallel calculations of MST using both Prim's and genetic Prim's algorithm.

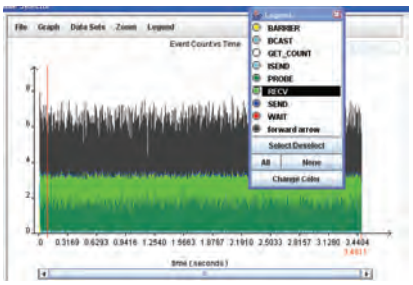

Fig.9. Communication transactions of parallel genetic Prim's algorithm

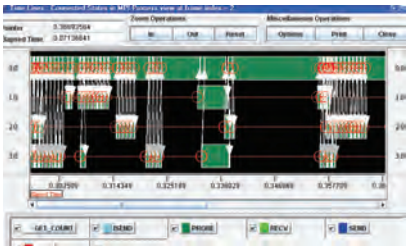

Fig.10. Gant's diagram of parallel genetic Prim's algorithm

IV. Conclusion

The paper investigates the efficiency of parallel Prim's and parallel genetic Prim's algorithms on multicomputer platform. Parallel computational models based on "manager-worker" parallel programming paradigm are presented. The algorithms are implemented using parallel MPI program. Experimental results show that both algorithm scale well with respect to the multicomputer size.

References

[1] Prim R., Shortest Connection Networks and Some Generalizations, Bell System Technical Journal, Vol.36, pp.1389÷1401, 1957.

[2] Kruskal J., On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem, Proc. of the American Mathematics Society, Vol.7, No.1, pp.48÷50, 1956.

[3] Zhou G., M. Gen, Genetic Algorithm Approach on Multi-Criteria Minimum Spanning Tree Problem, European Journal of Operational Research, Vol.114, No.1, 1999.

[4] Hanr L., Y. Wang, A Novel Genetic Algorithm for Degree-Constrained Minimum Spanning Tree Problem, IJCSNS International Journal of Computer Science and Network Security, Vol.6, No.7A, pp.50÷57, 2006.

[5] Lin L., M. Gen, Node-Based Genetic Algorithm for Communication Spanning Tree Problem, IEICE Transactions on Communications, E89-B, pp.1091÷1098, 2006.

[6] Palmer C., A. Kershenbaum, Representing Trees in Genetic Algorithms, Proc of the First IEEE Conference on Evolutionary Computation, pp.379÷384, Orlando, FL, 1994.

[7] Raidl G., B. Julstrom, A Weighted Coding in a Genetic Algorithm for the Degree-Constrained Minimum Spanning Tree Problem, 2000 ACM Symposium on Applied Computing, Como, Italy, 2000.

[8] Julstrom B., Codings and Operators in Two Genetic Algorithms for the Leaf-Constrained Minimum Spanning Tree Problem, International Journal on Applied Mathematics and Computer Science, Vol.14, No.3, pp.385÷396, 2004.