Performance Comparison of Parallel Genetic Algorithms for Combinatorial Optimization Problems

Milena K. Lazarova[1], Plamenka I. Borovska[2], Shada A. Mabgar[3]

Abstract - **The goal of the paper is to investigate the scalability and compare the performance parameters of solving optimization and constraint satisfaction problems using parallel genetic algorithms based on independent island evolution of local subpopulations and best chromosomes migration in a ring topology for evolutionary based solution of three selected combinatorial optimization problems.**

Keywords - **island parallel genetic model, combinatorial optimization problems, performance comparison, profiling, scalability**

I. Introduction

An optimization problem is the problem of finding the best solution from all feasible solutions. An optimization problem consists in finding the best solution in a large set of feasible solution, where the quality of each solution is evaluated using an objective function [1, 2]. Many optimization problems are NP-hard that is an efficient (i.e. polynomial time) algorithm for their solution may not exist [3]. In such cases it is worth looking for algorithms that find approximate solution whose measure is not too far from the optimum.

Evolutionary algorithms (EAs) are search methods that take their inspiration from natural selection and survival of the fittest individuals in the biological world [4]. Several different types of evolutionary based search methods are developed including genetic programming, evolutionary programming, evolutionary strategies and genetic algorithms.

Genetic algorithms (GAs) are computational approaches for solving a variety of optimization problems [5-7]. GAs are search procedures based on the ideas of evolutionary processes in the biological individuals. They randomly create an initial population of individuals and then use genetic operators for selection, crossover and mutation to yield new offspring. GAs are successfully applied in solving optimization and constraint satisfaction problems.

Parallel genetic algorithms (PGAs) can be conveniently implemented on parallel and distributed systems. Each processor performs the genetic operations independently on an isolated subpopulation of the individuals periodically sharing its best individuals with the other processors through migration [8, 9].

[1]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

[2]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

[3]Shada Mabgar is a Ph.D. student at the Faculty of Computer Systems and Control, Department of Computer Systems, 8 Kliment Ochridsky str., 1756 Sofia, Bulgaria

The goal of the paper is to investigate the scalability and compare the performance parameters of solving optimization and constraint satisfaction problems using PGA. A parallel programming model is suggested for evolutionary based solution of three selected combinatorial optimization problems: the traveling salesman problem, the knapsack problem and the n-queens problem. The PGA model utilizes independent island evolution of local subpopulations and best chromosomes migration in a ring topology. The performance of the problem solutions is evaluated and compared based on flat (MPI) and hybrid (MPI+OpenMP) implementations of the models. Profiling and scalability analysis are also performed.

II. Evolutionary Computations of Combinatorial Optimization Problems

Several methods and parameters have to be specified when solving given problem by evolutionary algorithms:
- the size of the chromosome pool at the start of each successive generation and the number of generations that will be evolved;
- the selection method that attempts stochastically to select individuals from one generation to create the basis of the next generation providing that the fittest individuals will have a greater chance of survival than weaker ones;
- the crossover method that will allow the offspring to carry forward the important genetic material of the parents, whilst introducing enough variation to become potentially more fitter;
- the mutation occurrence strategy that is seen as an unanticipated change in a chromosome pattern of some of the individuals resulting occasionally in a much weakened or much stronger individual.

The selected case studies to be solved by evolutionary approach are two optimization problems: the traveling salesman problem and the knapsack problem, and one combinatorial constraint satisfaction problem: the N-queens problem.

The traveling salesman problem (TSP) is an NP-hard combinatorial problem that requires finding the shortest tour of a group of cities without visiting any town twice. The TSP may be presented mathematically as finding the Hamiltonian cycle of minimal weight within a weighted fully connected undirected graph G = (V, E) where the vertices present the cities, the edges denote the intercity paths and the weights of the edges represent the intercity distances. The deterministic method to solve the TSP problem involves traversing all possible routes, evaluating corresponding tour distances and finding out the tour of minimal distance. The total number of possible routes traversing n cities is n! therefore in cases of

large values of n it becomes impossible to find the cost of all tours in polynomial time.

TSP can be solved using genetic approach by representing each tour as a chromosome that is the sequence of visiting the towns by the salesman [10]. In our case permutation encoding is used where every chromosome is a string of numbers that represent a position in a sequence. For the chromosomes permutation coding is used which is the best method for coding ordering problems. The fitness represents the length of the tour. The selection is performed following the rules of the roulette wheel method – the individuals of the highest fitness are selected for parents. The method of recombination is that of one crossover point – one part of the first parent and other part of the second parents is taken with special care not to repeat a city in the tour. The mutation applied is of the normal random type and involves changing of the city order.

The knapsack problem is one of the classical optimization problems recognized to be NP-hard. It arises whenever there is resource allocation with some constraints. The problem can be stated as follows: given a set of items each having certain cost and value, to determine the items that total cost does not exceed some given cost and the total value is as large as possible. Let the knapsack capacity is denoted as $c > 0$ and there are N classes of items. The number of items in each class is unlimited. Each item in given class has a value $v_i > 0$ and a weight $w_i > 0$. All classes have different values and weights. The goal is to find the most valuable set of items that fit in a knapsack of the fixed capacity:

$$\sum_i \delta_i w_i \le c, \quad V = \sum_i \delta_i v_i \qquad (1)$$

where $\delta_i = 1$ when the i^{th} item is selected, 0 otherwise and V is the total value have to be maximized.

There are different variations of the knapsack problem but the typical formulation in practice is the 0/1 knapsack problem, where each item must be put entirely in the knapsack or not included at all. This 0/1 property makes the knapsack problem hard for a simple greedy algorithm to find the optimal selection. There are two main approaches for solving this problem: branch and bound and dynamic programming. If N is the total number of classes then 2^N subsets of the item collection should be evaluated in order to find the optimal solution using the brute-force approach. An exhaustive search for a solution to the knapsack problem generally takes exponential running time and therefore is infeasible. Some dynamic programming techniques also have exponential running time although have proven useful in practice.

The chromosomes in the case of solving the knapsack problem with GA [11-13] will have a length equal to the number of the classes of items to be put in the knapsack. Binary chromosomes will be utilized that is value 1 in given position will means the item will be selected and value 0 means the item will not be selected for the subset of the items in the knapsack. The fitness of each chromosome is the total weight of all items in the knapsack. A population is initially randomly created as certain number of possible solutions. The selection is based on the roulette wheel approach that is the fitness of the chromosomes determine the probability that a chromosome is selected to survive in the next generation. Crossover uses one point of mixing of two parent solutions in order to produce an offspring. Mutation is based random flip of a bit in a chromosome that happens with certain probability.

The N-queens problem is a classical combinatorial constraint satisfaction problem formulated as solving the task to place N queens on a N×N chessboard in such a way that no queens attack each other. The difficulty of the N-queens problem arises from the fact that the search space of possible solutions is an incredibly large even for small values of N. The complexity of the N-queens problem is estimated as O(N!) and the problem belong to the class of NP-complete problems requiring a brute-force algorithm to guarantee that the solution can be found for any value of N. The basic classes of strategy for solving the N-queens problem are systematic search strategies and repair strategies. A depth-first search backtracking algorithm can solve the N-queens problem in reasonable time but only for small values of N.

In the case of solving N-queens problem by GA [14-16] initial population will comprises randomly generated placements of the queens on the board represented as permutations of an N-tuple (1, 2, 3, …, N). A chromosome i shows the column where the queen in row i is placed. The fitness of each individual measures how close it is to the problem solution. Since a solution to the N-queens problem requires no queens to attack each other the fitness is calculated as the number of conflicts between queens. The individuals in the population are evaluated and sorted according to their fitness. A crossover process creates new individuals combining two parents. Mutation involves a random change of an individual and is implemented as exchange of the positions of two queens. Genetic algorithm finishes the search either if a solution of the queens placement on the board is found or a predefined number of iterations is accomplished.

III. Parallel Computational Model of GA for Combinatorial Optimization Problems

All targeted combinatorial problems can be solved using the suggested parallel computational model (fig.1). The model utilizes parallelization method that divides the population into some number of demes (subpopulations) that are separated and evolve independently on each of the processors of the multicomputer platform. The parallel computational model utilizes a parallel algorithmic paradigm “synchronous iterations”. Each process evolves a subpopulation performing the genetic operations selection, crossover and mutation. Iterations of independent genetic evolution on each of the processes for certain number of generations are followed by a communication stage for migration of the best solutions found so far between the processes. The processes are organized in a logical ring and each process sends its local best individuals to one of its neighbors while receiving migrants from its other neighbor process. The migrants are compared with the local subpopulation and the local worst solutions according to their fitness are replaced. Thus each process will check for incoming data from its neighbor process while evolving its subpopulation and when the data are received will utilize them replacing its worst chromosomes.

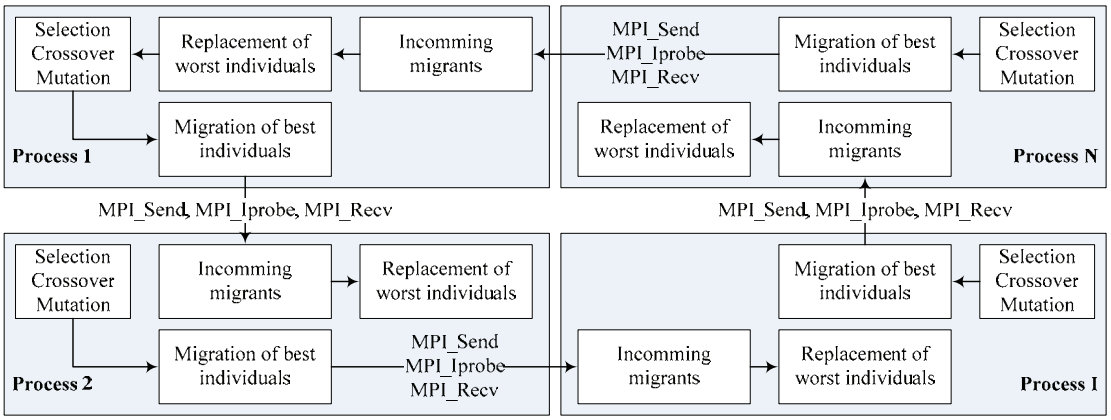

Fig.1. Parallel computational model for PGA for solving combinatorial optimization problems

Each process performs the following tasks:
- randomly generates a population of predefined size and evolves it certain generations;
- packs the best local chromosomes and sends them (MPI_Send) to one of its neighbors in the logical ring, i.e. the process with rank I+1 (the last process send migrants to the first one);
- checks for an incoming message (with MPI_Iprobe) from its other neighbor, i.e. the process with rank I-1, with a message tag MPI_MIGRATION (the first process receives migrants from the last one) and receives the data (MPI_Recv) to perform migration replacements.

The main difference in the utilization of the suggested computational model for the targeted problems is the GA termination criteria. For the optimization problems (TSP and knapsack) the evolutionary operations terminate after certain number of generations are evolved and the best solution is determined by a global reduction operation on the local optimal solutions. Due to the diversification introduced by the independent parallel evolutions the more parallel processes are employed the less number of iterations of local evolution will be accomplished. When solving the constrain satisfaction combinatorial problem of N-queens GA terminates when a solution is found. For this reason additional communications are required for sending termination message to all process when any of them founds a solution. Thus each process performs the following additional operations:
- checks for incoming message (with MPI_Iprobe) from any other process (MPI_ANY_SOURCE) with message tag MPI_STOP and if such message have been received terminates itself;
- if solution is found in the local population sends a message to all other processes with a termination tag MPI_STOP.

A parallel genetic algorithm library is developed in order to allow the implementation of the above described parallel programming model for solving the selected combinatorial problems based on evolutionary computations. The library consists of data structures, classes and functions required for implementation of the genetic operators for selection, crossover, mutation and migration.

Certain data structure comprising the data to be transferred during the migration of chromosomes between the subpopulations is used in order to speed up the communication transactions.

IV. Experimental Framework and Genetic Parameters

The genetic parameters used in the experimental evaluation of the performance of PGA for solving the selected combinatorial problems are given in Table I. The population size depends on the problem size, the subpopulation size depends on population size and the number of processors and the number of migrants depends on the population size (4% from the population size).

TABLE I. GENETIC PARAMETERS

Parameter	Description
Number of generations	200
Subpopulation size	population size / number of the processors
Migration topology	circular
Migration frequency	20 generations
Number of migrants	4% from the population
Mutation rate	various mut. rate: $0.01 \leq \mu \leq 0.2$ incremental step $\mu+=0.001$

We have estimated the parallel system performance experimentally for different workload in order to evaluate the scalability in respect to the complexity of the solved problem. For the TSP the number of cities used in the experiments is 100, 200, 300, 400, 500 and 600. The experiments for solving knapsack problem with PGA are made for three different cases of 100, 250 and 500 classes of items that have to be collected in a knapsack of capacity 1000. The PGA for solving N-queens problem is tested at several different workloads: board size of 12, 14, 16 and 17 elements.

The experimental multicomputer platform consists of 10 workstations (Intel Pentium 4, 3.2GHz, 1G RAM, Hyper-Threading) connected via Fast Ethernet Switch (100 Mbps). The parallel genetic algorithm library is implemented in C++ using Microsoft Visual Studio 2005, MPICH-2 and OpenMP.

The machine size varies from 1 to 10 workstations in order to explore the scalability of the parallel computer platform in respect to the parallel application under investigation. Each experiment is repeated 10 times and average values of the performance parameters are calculated.

V. Parallelism Profiling and Performance Analysis

The results for the speedup achieved in solving the selected combinatorial problems by evolutionary algorithm are shown in Fig.2, Fig.3 and Fig.4 for the TSP, the knapsack and the N-queens problem respectively. The results show that good speedup values are obtained for both targeted problems. The scalability in respect of the size of the parallel platform is almost linear. For the optimization problems (TSP and knapsack problem) the speedup achieved for 10 processes is 9 for 600 cities and 8.7 for 500 items. For the N-queens problem the speedup is slightly smaller (8.3 with 10 processes) due to the different termination condition introducing additional communications. The suggested programming model scales well also in respect to the workload. The suggested parallel computational model introduces very little communication overhead during the migration stage. Moreover the utilization of asynchronous communications further decreases the time spend inter- processor communications. On the other side the migration provides better subpopulation diversity and thus increases the possibility of finding better solution in less number of evolution periods.

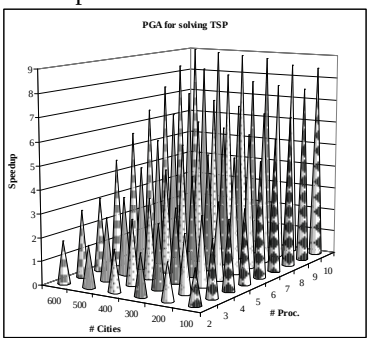

Fig.2. Speedup achieved for solving TSP by PGA

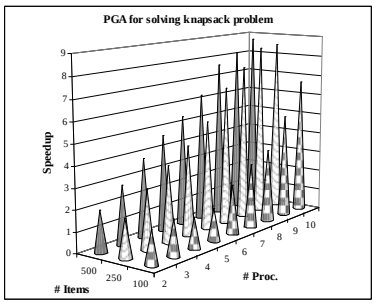

Fig.3. Speedup achieved for solving knapsack problem by PGA

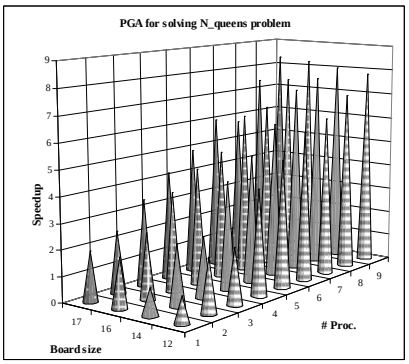

Fig.4. Speedup achieved for solving N-queens problem by PGA

VI. Conclusion

In this paper performance comparison and scalability analysis of solving optimization and constraint satisfaction problems are made using PGA. The PGA computational model is suggested and implemented based on independent island evolution and best chromosomes migration in a ring topology and is applied for three combinatorial optimization problems. The parallel system performance is evaluated experimentally for different workload and machine sizes. The results show that the model scales well both in respect to the parallel computer size and the application workload.

References

[1] P. Klein, N. Young, "Approximation algorithms for NP-hard optimization problems", *Algorithms and Theory of Computation Handbook*, CRC Press, 1999.

[2] P. Crescenzi, V. Kann, *A compendium of NP optimization problems*, http://www.nada.kth.se/nada/theory/problemlist.html

[3] G. Ausiello, *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*, Springer, 1999.

[4] P. Gray, W. Hart, L. Painton, C. Phillips, M. Trahan, J. Wagner, *A Survey of Global Optimization Methods*, http://www.cs.sandia.gov/opt/survey/ main.html

[5] Z. Michalewicz, *Genetic Algorithms + Data Stractures = Evolutioin Programs*, Springer, 1996.

[6] A. Eiben, "Evolutionary Algorithms and Constraints Satisfaction: Definitions, Survey, Methodology, and Research Directions", in L. Kallel, B. Naudts, A. Rogers (eds.), *Theoretical Aspects of Evolutionary Computing*, *Natural Computing Series*, Springer, pp.13÷58, 2001.

[7] C. Reeves, J. Rowe, *Genetic Algorithms – Principles and Perspectives: A Guide to GA Theory*, Springer, 2002.

[8] B. Wilkinson, M. Allen, *Parallel Programming Techniques and Applications Using Networked Workstations and Parallel Computers*, Pearson Prentice Hall, 2005.

[9] E. Cantu-Paz, "Migration Policies, Selection Pressure, and Parallel Evolutionary Algorithms", *Journal of Heuristics*, Vol.7, No.4, 2001, pp. 311÷334.

[10] P. Borovska, "Solving the TSP in Parallel by Genetic Algorithm on Multicomputer Cluster", Proc. of the Int. Conf. on Computer Systems and Technologies, Bulgaria, pp.II-11-1÷II-11-6, 2006.

[11] S. Khuri, T. Back, J. Heitkotter, "The Zero/One Multiple Knapsack Problem and Genetic Algorithms", *Proc. of ACM Symposium on Applied Computing*, pp.188÷193, 1994.

[12] A. Anagun, T. Sarac, "Optimization of Performance of Genetic Algorithm for 0-1 Knapsack Problems Using Taguchi Method", *Proc. of Int. Conf. on Computational Science and Its Applications (ICCSA 2006): Workshop on Optimization: Theories and Applications*, 2006, pp.678÷687.

[13] Z. Ezziane, "Solving the 0/1 knapsack problem using an adaptive genetic algorithm", *Journal Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol.16, №1, pp 23÷30, 2002.

[14] A. Kilic, M. Kaya, "A New Local Search Algorithm Based on Genetic Algorithms for the N-Queens Problem", *Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2001)*, USA, pp. 2001.

[15] K. Crawford, "Solving the N-Queens problem Using Genetic Algorithms", *Proc. of ACM/SIGAPP Symposium on Applied Computing*, USA, pp.1039÷1047, 1992