Ilkogretim Online - Elementary Education Online, 2021; Vol 20 (Issue 6): pp. 1446-1451
http://ilkogretim-online.org
doi: 10.17051/ilkonline.2021.06.146

Solution Of The Problem Of The Horse Tour Based On The Problem Of The Traveling Agent On A Distributed Computing Platform

Roberto Manuel Poveda Chaves, Universidad Distrital Francisco José de Caldas, [3]
https://orcid.org/0000-0002-6694-7673

Orlando García Hurtado, Universidad Distrital Francisco José de Caldas,
https://orcid.org/0000-0002-4155-4515

Javier Felipe Moncada Sánchez, Universidad Distrital Francisco José de Caldas,
https://orcid.org/0000-0003-1863-8144

ABSTRACT

This document describes the solution of the Horse Tour problem from a parallel genetic algorithm designed to solve the classic combinatorial problem of the Traveling Agent (TSP) on a distributed computing platform.

The problem of the Tour del Caballo reaches satisfactory solutions from the basic genetic operators that solve the TSP, but the solution improves in terms of speed and precision if the operators are modified with others of the same type but more sophisticated.

Keywords: Parallel Genetic Algorithms, Horse Tour Problem, Traveling Agent, Distributed Computing, Parallel Virtual Machines (PVM).

INTRODUCTION

The TSP is about a traveler who visits each of the n given cities exactly once and returns to the initial city. The solution to the problem consists in finding the sequence of cities visited (the cycle) that minimizes the total distance of the traveler.

The problem of the traveling agent has been tackled through various techniques, among which the Evolutionary Algorithms stand out. Gregory Gutin in his book, referenced in [1], delves into each of these techniques.

The traveling agent problem is today one of the most prominent problems in combinatorial optimization as well as one of the most important NP-Complete type

problems, since all known detrministic algorithms for their solution require either an exponential or factorial time. in the worst case for n cities) [2].($\frac{(n-1)!}{2}$

The Traveling Agent problem is widely applicable to a great variety of assignment, route optimization and planning problems, as well as to combinatorial didactic problems such as piece movement problems on a chessboard, which is the reason for this document.

The Knight's tour problem is the first problem with a statement similar to the Traveling Agent problem. The Knight problem was raised by Leonard Euler in 1759, Euler was interested in finding the shortest cycle traveled by a knight on a chessboard, that is, finding the 64 squares of the board visited by the knight only once.

Numerous investigations have been conducted since the problem was popularized by the RAND Corporation in 1948 [3]. The TSP is characterized above all by having a relatively simple statement but being quite difficult to solve.

Genetic Algorithms (GA's) is one of the most outstanding approaches in the field of Evolutionary Algorithms and they are defined as iterative general-purpose adaptive search procedures with the virtue of abstractly and rigorously describing the collective adaptation of a population of individuals. to a particular environment, based on a behavior similar to a natural system. Goldberg in [4] and Michalewicz in [5] provide a formal study on this topic.

AG's result in the vast majority of cases more robust and advantageous than enumerative and calculation-based optimization techniques for solving particular problems.

AG's are suitable to be implemented in a parallel way in a distributed computing system, first for the reason of trying to “save” time distributing workloads and second for the natural behavior of parallelism on spatially distributed populations; Tomassini in [6] highlights this goodness.

The model developed is a Parallel Genetic Algorithm (AGP) that uses two of the most important AGP models: Islands and Grids, as well as a local search optimization procedure such as the Lin-Kernighan 2-opt heuristic method.

Poveda and others in [7] describe these procedures in detail for the TSP solution.

Methodology

The AGP developed proceeds as follows: A master process (client process) is in charge of distributing the tasks (executing a Genetic Algorithm) to several slave processes (server processes). Each slave process (associated with a single processor) works simultaneously with eventual information exchanges.

An integer coding is used for the strings that make up each of the individuals in the population, since this is the most advisable way to proceed when you want to face a graph problem such as the traveling agent problem.
For example, the string [4 6 8 1 7 5 3 0 2 4] represents an individual in the population that highlights a particular trajectory for a 9-city problem. The trajectory that this individual represents is the cycle: [4 -> 6 -> 8 -> 1 -> 7 -> 5 -> 3 -> 0 -> 2-> 4].
The selection operator is implemented on each randomly determined initial population (different initial population in each processor) with the purpose of working from now on with the individuals with the greatest aptitude.
Each population of individuals (managed by each slave process) is distributed in a two-dimensional mesh.
The crossover rate (0 <<1) is the same in each processor and is constantly managed throughout the execution of the Genetic Algorithm, its complete operation is described in [7].$p_c p_c$
In a certain number of iterations of the algorithm there is a partial exchange of information between processors. The crossing for an individual of a population managed by the i-th processor is carried out according to the previous procedure but the couple selected for the crossing is an individual that belongs to another population managed simultaneously by the processor (i + 1) (module n), for all i, i = 1... n where n represents the number of available processors. It is appreciated that this achieves an exchange of partial genetic material between individuals belonging to different populations managed by Genetic Algorithms that work in parallel.
The new individuals thus generated enter into conflict with the individuals of the current population to determine which individuals make up the new population; again, the individuals with the highest fitness will make up that new population.
After this "new" population has been formed, a usual crossing between these individuals with the same crossing rate described above is carried out locally (in the corresponding Genetic Algorithm managed by each processor).p_c
The mutation rate, 0 <1 is implemented to be provided differently in each processor, but it is handled like the crossover constantly throughout the model execution. The mutation operator also evolves locally in each Parallel Genetic Algorithm, [7] again describes the detail of this procedure. In itself, the intention of the mutation is to avoid falling into local optimum.$p_M p_M$ <2

This is where the 2-opt local optimization heuristic procedure is applied. This procedure attempts to optimize sub-trajectories by finding shorter paths between two given values avoiding “edge crossings” that generally cause much greater distances in the total route.
The model now turns to the Islands model to continue its course.

Again, every certain number of iterations (frequency), an exchange of information takes place between processors. The aptitude of the population resulting from the previous procedures is evaluated, the best individuals are determined as well as the worst individuals in each Genetic Algorithm executed by each processor and

the worst k individuals in processor i are replaced by the best k individuals of the managed population by processor (i + 1) (modulo n) for all i, i = 1... n (n: number of available processors.
Finally, each Genetic Algorithm calculates the fitness of the individuals resulting from each population and the model continues its course in a next iteration. The number of iterations is determined by the user.
The horse tour problem is a particular case of the TSP problem [8] where each square on the chessboard corresponds to a city on the TSP and the distance matrix corresponds to the movement of the hair on the board.
The knight tour problem (for a kxk board) is solved by finding a permutation σ such that:

$$\sum_{i=0}^{n-2} d_{\sigma(i)\sigma(i+1)} + d_{\sigma(n-1)\sigma(0)} = 0$$

With $n = k^2$. For example in figure 1, or among others.$d_{27,10} = 0 d_{27,21} = 0$

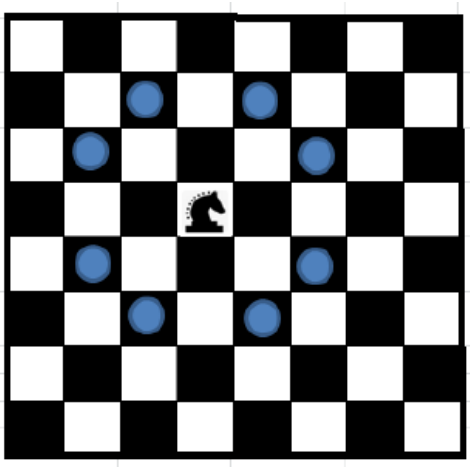

Figure 1. Horse Tour.

In general the distance matrix is declared as:

$$d_{rs} = \begin{cases} 0 \ \text{si}(r-2 \geq 0 \ \text{y} \ s+1 < m) \ \text{o} \ (r-1 \geq 0 \ \text{y} \ s+2 < m) \text{o} \\ (r+1 < m \ \ y \ \ s+2 < m) \text{o} (r+2 < m \ \ y \ \ s+1 < m) \text{o} \\ (r+2 < m \ \ y \ s-1 \geq 0) \text{o} (r+1 < m \ \ y \ \ s-2 \geq 0) \text{o} \\ (r-1 \geq 0 \ \text{y} \ s-2 \geq 0) \text{o} (r+2 \geq 0 \ \text{y} \ s-1 \geq 0) \text{o} \\ \\ 1 \ \text{elsewhere} \end{cases}$$

The genetic operators originally implemented in our model were the basic operators, see [7]. A subsequent study on sophisticated genetic operators for the Traveling Agent problem yielded much more satisfactory results [9], the following combinations found were the best in this study (RSS-MOX-CM and RSS-OX-ISM), where RSS: Selection Stochastic Surplus. MOX: Modified Cross Order. CM: Mutation

Cycle. OX: Cross Order. ISM: Insertion Mutation. These operators, along with others, are described in detail in [9].

Results

In [7] we solve different TSP problems using 5 independent processors on the PVM distributed environment [10], these problems are test instances present in the TSPLIB standard library [11].
The following table shows the number of iterations and time required to reach the optimal solution (0) in the horse tour problem through our model with basic genetic operators and "sophisticated" genetic operators for a board of size 8x8.

Operators	#of Iterations	Time (seconds)
Basics: RWS-MOX-EM	fifteen	65
RSS-MOX-CM	13	52
RSS-OX-ISM	10	fifty

Conclusions

Our model solves large instances of the TSP and multiple applications of it, such as routing and assignment problems [12], going through the solution of piece movement problems on chess boards to balancing problems in electrical charge distribution [13].

Therefore, it can be conjectured that our model with appropriate variants (those stated in this work and those presented in future works) is seen as an efficient model in solving various combinatorial optimization problems.

Bibliography

1. G. Gutin and A. Punnen (Eds.). Combinatorial Optimization. The traveling Salesman Problem and Its Variations.
2. E. Cantú Paz. A Survey of Parallel Genetic Algorithms. Technical Report 97003, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana Champaign, Urbana, IL, 1997.
3. P. Larrañaga, CMH Kuijpers. Genetic Algorithms for the Traveling Salesman Problem Review of Representations and Operator. Springer Volume 13 Number 2 199 pp. 129-170.
4. D. Goldberg. Genetic Algorithms in Search, Optimizations and Machine Learning. Addison-Wesley, 1989.
5. Z. Michalewicz. Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, 2nd edition, 1994.
6. M. Tomassini. A Survey of Genetic Algorithms. Volume III of Annual Reviews of Computational Physics, World Scientific.
7. R. Poveda, J. Gómez, and E. León Dept. of Computer Engineering, Universidad Nacional de Colombia Bogotá, Colombia. Grisland: A Parallel Genetic Algorithm for Finding Near Optimal Solutions to the Traveling

Salesman Problem published GECCO'09, July 8–12, 2009, Montreal Quebec, Canada. ACM 978-1-60558- 505-5 / 09/07.
8. C. Moon. (2002).An efficient genetic algorithm for the traveling salesman problem with precedence constraints. European Journal of Operational Research, 606-617.
9. R. Poveda, J. Gómez, and O. García Dept. of Computer Engineering, Universidad Nacional de Colombia Bogotá, Colombia. comparative study of genetic operators for the grisland model. Jokull Journal. Vol 64, No. 11; Nov 2014.
10. A. Geist, A. Beguelin, J. Dongarra, and W. Jiang, PVM: Parallel Virtual Machine: A Users'Guide and Tutorial for Network Parallel Computing (Scientific and Engineering Computation). Paperback - Nov 8, 1994.
11. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/
12. R. Poveda, O. García, and E. Cárdenas Faculty of Engineering "Francisco José de Caldas" District University, Bogotá, Colombia. Assignation Problems' Solutions by Parallel and Distributed Genetic Algorithms. Applied Mathematical Sciences, Vol. 8, 2014, no. 104, 5185 - 5194
13. R. Poveda, F. Guerrero, and N. Chaparro Faculty of Engineering, Francisco José de Caldas District University, Bogotá, Colombia. Optimization of Power Distribution Networks Using Parallel Genetic Algorithms. Applied Mathematical Sciences, Vol. 8, 2014, no. 156 , 7785-7796