Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 100 (2015) 1634 – 1639

Procedia Engineering

www.elsevier.com/locate/procedia

25th DAAAM International Symposium on Intelligent Manufacturing and Automation, DAAAM 2014

Usage of Partial Genome Fitness Evaluation Mechanism to Get Faster Results in Genetic Algorithms

G. Janeš [a,*], Z. Car [a,b]

[a]Center for advanced computing and modeling, University of Rijeka, Radmile Matejcic 2, 51000 Rijeka, Croatia
[b]Faculty of Engineering, University of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia

Abstract

The aim of the research presented in this paper is to test the efficiency of the proposed genetic algorithm in the fast or real-time optimizations. Proposed partial genome fitness evaluation mechanism was tested on the wireless network topology optimization and job shop scheduling problem (JSSP). Wireless network topology optimization requires frequent and very fast (sometimes real time) adjustments. Job shop scheduling problem is one of the most general and difficult of all traditional scheduling problems. Search based on traditional genetic algorithms usually are not suitable in solving these problems. Long computational time and high memory usage if a large population and / or a large number of generations are used, on the other hand, a larger population and a larger number of generations usually provide better results. In comparison with two other similar and commonly used genetic the proposed algorithms have significant improvements in speed and some case solution quality.

Keywords: fast genetic algorithm; partial genome fitness; job shop scheduling problem (JSSP); wireless network topology

1. Introduction

Genetic algorithms are typically used when there is very little knowledge on the solution space or when there are far too many solutions to use standard search / optimization methods. Methods used in genetic algorithm search mechanism inspired by Darwin's theory of evolution and problems are solved by using techniques of natural evolution: crossover, inheritance, mutation and selection. They can be used to solve every optimization problem which can be described with the chromosome encoding, and it is very easy to understand and it can get multiple solutions that can be evaluated later with other methods.

* Corresponding author. Tel.: +385 51 584 812; fax: +385 51 406-588.
E-mail address: gordan.janes@cnrm.uniri.hr

doi:10.1016/j.proeng.2015.01.537

In today's world importance of real time or very fast optimization methods is increasing [1], [2] and [3]. Real-time tasks are characterized by computational activities with timing constraints and classified into two categories: a hard real-time task and a soft real-time task [4]. In a situation when real-time optimization is required, tardiness can be catastrophic. The goal of hard real-time tasks scheduling algorithms is to meet all deadlines, in other words, to keep the feasibility of scheduling through admission control. However, in the case of soft real-time tasks, slight violation of deadlines is not so critical [4] [5]. Genetic algorithms are well suited to fast solve problems like JSSP due to their adaptability and effectiveness at searching large spaces but very often commonly used genetic algorithms take a long time to produce results [8]. On the other hand, GA is rarely used in real time optimization problems such as wireless network since they are not fast enough, so we developed and tested genetic algorithms and methods to solve this problem. As it can be seen in results, the major advantage of proposed algorithm is much better speed because:

- It works with only one population - less data is being moved in computer memory than in algorithms that use two populations of possible solutions.
- Low number of calculations are required – fitness for each individual (possible solution) is only calculated when that particular individual is created or changed.
- Small amount memory content is being moved and accessed

2. Problem definition

Proposed algorithm was tested on two different problems: job shop scheduling problem and wireless network optimization:

- In wireless network topology optimization problem there are set of wireless access points $U_{WL,M} = \{W_{L1}, W_{L2}, \dots, W_{Lmap}\}$ and set of network clients $U_{WL,C} = \{C_1, C_2, \dots, C_{nc}\}$. All wireless clients are in a range of at least one access point. Quality and speed of network connection very much depends on the distance between access point and client. Maximum possible distance between client and access point was also defined. All access point have fixed network speed, but speed is also decreasing with the number of clients connected to particular access point.
- In the JSSP problem there are a set of machines $U_{JSSP,M} = \{M_1, M_2, \dots, M_{nm}\}$ and a set of operation $U_{JSSP,O} = \{O_1, O_2, \dots, O_{no}\}$. Each machine can execute one or more operations, but only one operation at a time. Each operation must be completed without interruption once it has started. The processing time $T_{P(i,j)}$ of operation O_j and on machine M_i and transport time $T_{T(i,i+1)}$ between machines i and $(i+1)$ are machine-dependent. All operations have to be processed in a given order. The problem is to assign each operation to the appropriate machine and to sequence the operations on the machines in order to minimize the makespan. To make this problem more real-life like and complex, time required to complete operations and transport time among machines were taken in consideration. In problem test in this paper, the average value of transport time between machines was 1/3 of average operation time on machines.

3. The proposed algorithm

To test proposed algorithm two separate programs (one for each problem) were made in Visual C# programing language.

3.1. Coding and fitness evaluation

The encoding method determines how the problem is structured in the algorithm. Each chromosome (individual) represents a possible solution of the problem.
JSSP: The genes of the chromosomes describe the assignment of operations to the machines in JSSP problem or network connection between. Order they appear in the chromosome describes the sequence of operations. The makespan is calculated for each chromosome and this value coincides with the fitness of that chromosome. In this JSSP problem, we are searching for individuals (solutions) with lower values of the fitness (makespan), and genetic evolution during selection will prefer chromosomes with a lower fitness.

An important requirement for a fitness function is that the closer a chromosome is to the solution, the higher the relative fitness should be. The fitness function must reward chromosomes in a manner that drives the population to the desired solution [3]. For each generation, all the chromosomes are evaluated, and the best and the worst individual in generation and the generation average fitness were recorded in this research.

3.2. Initial Population

In this research, the initial population has been produced randomly because this method provides a high level of genetic diversity. Random generation of the initial population requires longer search time (or more computational power) to obtain the solution, but, on the other hand, high diversity is important to avoid premature convergence and to escape local optimum. Just like in real life situations, in GA diversity helps a population to react quickly on changes in the environment and it allows the population faster adaptation to changes.

From empirical studies, over a wide range of function optimization problems, a population size of between 50 and 100 is usually recommended. In this research population size of 100 individuals was used in all algorithms.

After creation of the initial population, fitness of each individual in the population is calculated. Individual with the lowest value of fitness is found and marked as such.

```
program start;
  {    create initial population;
       mark individual with lowest fitness;
       repeat
              repeat
                     selection (reject the individual with lowest fitness);
                     crossover (create a new individual);
              until (created N new individuals);
              mutation;          }
       until (GA is complete);
       show results;
end
```

Fig. 1. Algorithm of Proposed modified GA.

3.3. Mutation operation

Mutation is a genetic operator used to maintain genetic diversity in genetic material (possible solution). Like in biological mutation, mutation in GA alters one or more genes. In algorithms tested here, individuals and genes that were altered were randomly selected and the probability for mutation was 3%. Since individuals are altered during process of the mutation - fitness for all altered individuals is recalculated.

3.4. Selection and Crossover

In the first-step individual with lowest value fitness is located and marked as such. After that two individuals (parents) are selected randomly and their genes are used for creation of a new individual. During the process of crossover, only one new individual is created. Newly created individual will replace one with lowest fitness value fitness for that new individual is calculated.

Removing the individual with the lowest fitness when a new individual is created is consistent with evolutionary theory: number of individuals in the habitat is designated by resources of that habitat. Individuals that are not successful enough to gain access to the resources have to leave habitat or will not survive.

Parents used to create new individual are selected randomly. All individuals in the population have the same chance to be selected for mating, except individual with the lowest fitness which will be removed from the population and replaced with the newly created individual. According to our research, usage of patents selection based on fitness and creation of offspring based on gene fitness estimation is not advisable.

In this paper the proposed crossover operator (fig. 2.) is based on the fitness evaluation of just part of the genome. First the individual with the lowest fitness in a whole population is found and removed from the population. Two individual (possible parents) that will be used as building material for new individual are randomly selected from the entire population. Size of each building genome block in the crossover operation are also selected randomly. In every step of the child creation fitness of the gene blocks are calculated for both possible parents and values are compared. Block with better fitness is used. That process is repeated until new individual is created (fig. 2).

```
mark individual with lowest fitness;
randomly select parent 1
randomly select parent 2
repeat
        randomly define genome block size
        estimate genome block fittnes (gene from parent 1)
        estimate genome block fittnes (gene from parent 2)
        comapre genome block fitheses
        use "better" genome block for offspring creation
unitl (new individual is created)
```

Fig. 2. .Algorithm of proposed modified crossover.

4. Results

Results of the proposed algorithm were compared with the results of genetic algorithms with tournament and tournament-elimination selection. These two selections are chosen because they are fast, useful, robust, and it is commonly used in genetic algorithms and GA with tournament-elimination has many similarities with algorithm proposed in this paper.

4.1. Computational time

A computational time very much depends on a code optimization and it is not easy to compare computational speed of different genetic algorithms. In this case both computer programs were made by same person.

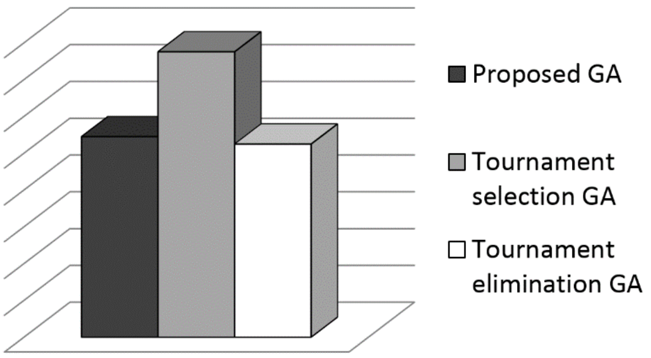

Fig. 3.Computational time.

First was tested how long it takes to run complete a hundred generations optimization for all three GA compared in this research with both computer programs. Ratio among times needed to complete tasks was same in both computer programs. As it can be seen from the Figure 3, time needed to complete a genetic algorithm with a tournament elimination and proposed algorithm is almost same, but GA with a tournament selection needs 40% more time for completion. Tournament elimination and the proposed algorithm are similar and they, as was supposed, takes the almost same time for completion.

A tournament selection genetic algorithm works with two populations and as a result much more data is accessed and moved in a computer memory than it is a case in other two compared algorithms. If code performance matters moving or accessing data in computer memory should be as low as possible because it can significantly prolong computation time. For example: to assign a constant value to a double precision real number in one-dimensional matrix takes 2-3 times more time than to do the same thing with an ordinary double precision variable.

In two-dimensional matrix (commonly used in GA) that time is 20-30 times higher (tested with MS Visual C# 2010 with integer numbers).

4.2. Performance– Results

Figure 4. shows comparison of fitness of proposed algorithm and tournament selection and tournament-elimination selection algorithm in job shop scheduling problem. Figure shows the fitness value of the best individual for 100 generation of the genetic algorithm (lower is better). First, the initial population is labelled as 1st and last as 101st generation. Proposed algorithm has similar final performance as GA with tournament selection but proposed genetic algorithm needs approximately 40% less time for completion of 100 GA generations, but it is much faster approaching to final (best) result.

As can be seen from results in figure 3. and 4. proposed algorithm outperforms other two GA for the same problem in time (fig. 3.), but also in approaching best final result.

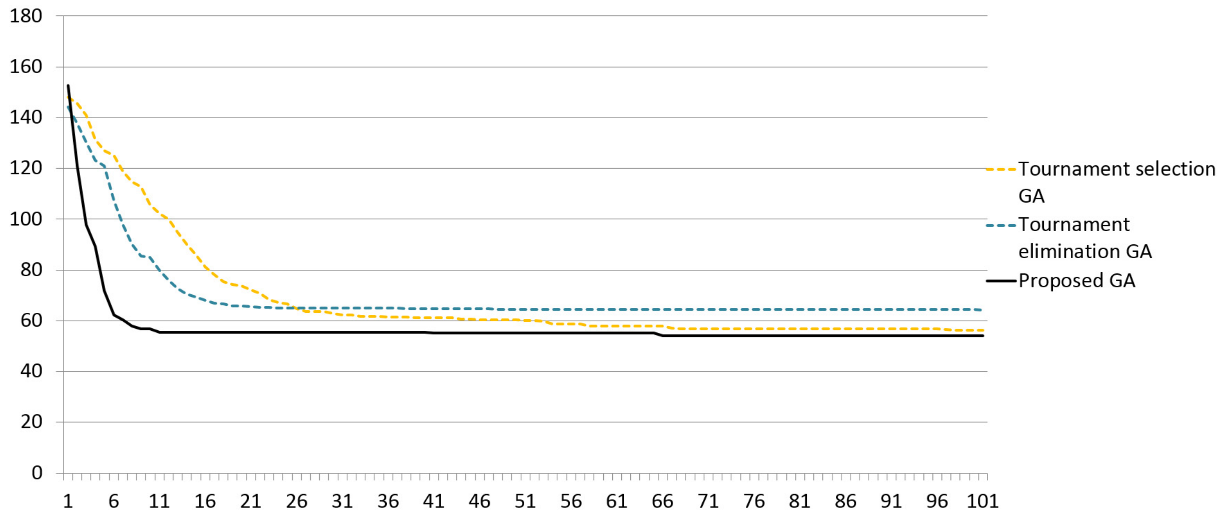

Fig. 4. Fitness.

4.3. Performance – Genetic diversity

Creation of genetically diversified individuals during the process of initial population creation and maintenance of a genetic diversity is required to ensure that the solution space is adequately searched. This is especially important in the earlier stages of the optimization process since loss of population diversity is considered as the primary reason for premature convergence. Too much selective pressure can lower the genetic diversity, on the other hand too little selective pressure can slow down converge and prevent GA to reach optimum in a reasonable time [2x]. Figure show loss of genetic diversity during GA for three tested selection methods. Tests were made with random and modified crossover method that is based on gene evaluation.

Proposed GA has fast loss of genetic diversity, as it can be seen from figure 5.

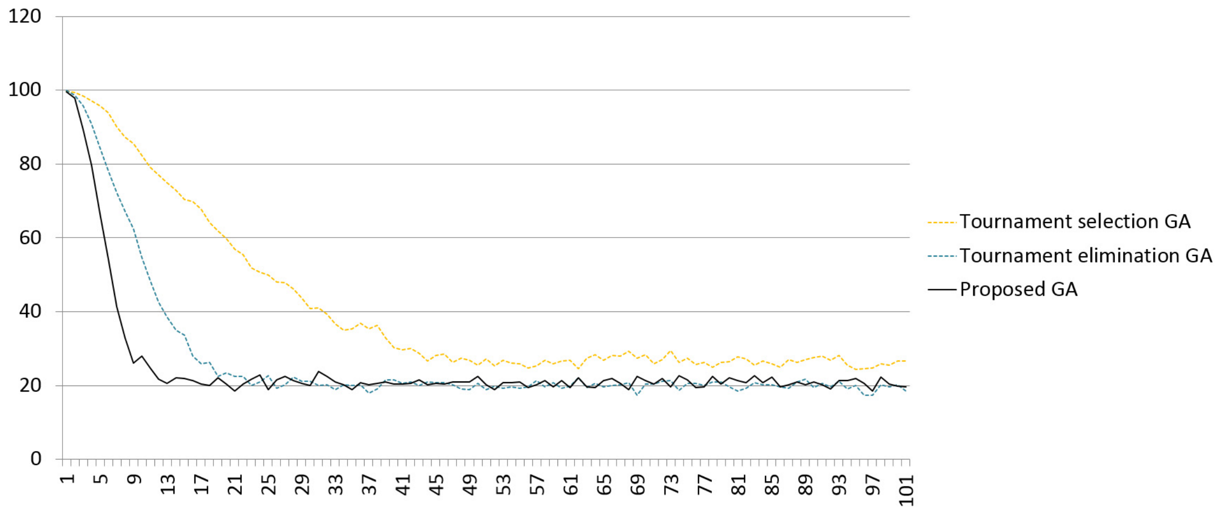

Fig. 5. Loss of genetic diversity.

As can be seen from figure 5. a tournament selection GA has the slowest loss of genetic material and the proposed algorithm has fastest loss of genetic diversity. Since the proposed algorithm was developed with the intention to have low computational time, fast loss of genetic material is expected.

In case that the proposed crossover method causes to fast loss of genetic diversity that a problem can be solved by introduction of partial random crossover since as can be seen in figure 5, a random crossover has slower loss of a genetic material.

Conclusion

In this paper, a modified genetic algorithm is proposed for solving the job shop scheduling problem (JSSP) and wireless network topology optimization. Proposed algorithm is based on tournament elimination algorithm but with modifications. Two main operations of the genetic algorithm were modified: selection and crossover operators.

The experimental results show that the performance of the proposed GA is significantly improved. Proposed genetic algorithm was also tested in wireless network optimization and job shop scheduling problems but it can be used, not only on these two types of problem, but also on other optimizations problems where fitness of single gene can be estimated. Result prove that the proposed genetic algorithm can be used in the optimization problems where speed (time) is important.

In future works, performance of proposed genetic algorithm will be tested on more complex problems and computer programs will be modified to work on computer systems capable of running large number of parallel operations.

References

[1] Joachim Wegener, Harmen Sthamer, Bryan F. Jones, David E. Eyres: Testing real-time systems using genetic algorithms, Software Quality Journal, (1997), Volume 6, Issue 2, pp 127-135.
[2] Koren, Y. : Reconfigurable manufacturing systems, Annals of the CIRP (1999), vol. 48, no. 2, p. 527-540.
[3] Simao, J.M., Stadzisz, P.C., Morel, G. Manufacturing execution systems for customized production, Journal of Materials Processing Technology, (2006), vol. 179, no. 1-3, p. 268–275.
[4] Myungryun Yoo : Real-time task scheduling by multiobjective genetic algorithm, Journal of Systems and Software, Volume 82 (2009), page 619-628.
[5] Zlatan Car, Branimir Barišić, Miroslaw Rucki : Emergent Synthesis Based Multi-Objective Design of the Manufacturing System Shop-Floor.
[6] Deepti Gupta, Shabina Ghafir : An Overview of methods maintaining Diversity in Genetic Algorithm s, International Journal of Emerging Technology and Advanced Engineering (Volume 2, Issue 5, May 2012).
[7] Bart Rylander, Computational Complexity and the Genetic Algorithm
[8] David Montana, Marshall Brinn, Sean Moore, Garrett Bidwell : Genetic Algorithms for Complex, Real-Time Scheduling (1998) , IN Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics.