IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 6, 2019, accepted June 11, 2019, date of publication June 21, 2019, date of current version July 9, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2924218

Hybrid Micro Genetic Multi-Population Algorithm With Collective Communication for the Job Shop Scheduling Problem

MARCO ANTONIO CRUZ-CHÁVEZ[1], MARTÍN H. CRUZ ROSALES[2], JOSÉ CRISPÍN ZAVALA-DÍAZ[2], JOSÉ ALBERTO HERNÁNDEZ AGUILAR[2], ABELARDO RODRÍGUEZ-LEÓN[3], JUAN CARLOS PRINCE AVELINO[4], MARTHA ELENA LUNA ORTIZ[5], AND OSCAR H. SALINAS[6]

[1]Research Center in Engineering and Applied Sciences (CIICAP), Autonomous University of Morelos State, Cuernavaca 62209, México
[2]Faculty of Accounting, Administration and Informatics (FCAeI), Autonomous University of Morelos State, Cuernavaca 62209, México
[3]Department of Systems and Computing (SyC), Veracruz Institute of Technology, Veracruz 2779, México
[4]Department of Mechanic Metal (DMM), Veracruz Institute of Technology, Veracruz 2779, México
[5]Department of Research and Technological Development (IDT), Emiliano Zapata Technological University of Morelos State, Emiliano Zapata 62760, México
[6]Academic Division of Information and Communication Technologies (DATIC), Emiliano Zapata Technological University of Morelos State, Emiliano Zapata 62760, México

Corresponding author: Marco Antonio Cruz-Chávez (mcruz@uaem.mx)

This work was supported in part by the Thematic Collaboration Networks of PRODEP through the SA-DDI-UAEM/15/451 Project, from 2015 to 2016.

ABSTRACT This paper presents a hybrid genetic algorithm with collective communication (HGACC) using distributed processing for the job shop scheduling problem. The genetic algorithm starts with a set of elite micro-populations created randomly, where the fitness of these individuals does not exceed a tuned upper bound in the makespan value. The computational processes distribute the micro-populations collectively. In the micro-populations, each individual's search for good solutions is directed toward the solution space of the fittest individual, identified by an approximation of genetic traits. In each generation of the genetic algorithm, the best individual from each micro-population migrates to another micro-population to maintain diversity in populations. Changes in the genetic sequence are applied to each individual by the simulated annealing algorithm (iterative mutation). In this paper, the results obtained show that the genetic algorithm achieves excellent results, as compared to other genetic algorithms. It is also better than other non-genetic meta heuristics or competes with them.

INDEX TERMS Genetic approximation, hamming distance, processes, barrier, landscape, micro-population.

I. INTRODUCTION

The Job Shop Scheduling Problem (JSSP) has a variety of applications in the manufacturing industry, one of which is found in the maquiladora industries. The problem is to find the optimal scheduling with fixed limited resources. The resources are the machines that execute the jobs required by the manufacturing system. Each job requires a certain number of operations. Finding an optimum scheduling for these machines, enables great savings in the use of resources, this can increase production while requiring less resource investment. This means manufacturing companies can increase productivity. In complexity theory, the JSSP is classified as an NP-complete problem [1], making it a very interesting problem for the scientific community.

The associate editor coordinating the review of this manuscript and approving it for publication was Shankarachary Ragi.

There are several strategies to accelerate the search for good solutions in the JSSP which use hybrid genetic algorithms (GA) or GA with distributed processes. In [2] include priority rules within the genetic evolution process with a hybrid heuristic, in order to obtain the shortest processing time to reduce the search space. They also include a local search to improve GA performance. In [3] present a hybrid genetic algorithm, which constructs schedules using a priority rule. The priorities are defined by GA itself, where the applied crossover is via a procedure called parameterized active schedule. This procedure prevents a premature GA convergence and applies a local search heuristic to improve solutions. In [4] propose several genetic operators for a hybrid genetic algorithm. They propose a new crossover operator which works as a function of the machines present in the JSSP. They also propose a mutation operator which works as a function of the critical path to increase diversity in the micro-

This work is licensed under a Creative Commons Attribution 3.0 License. For more information, see http://creativecommons.org/licenses/by/3.0/

population, which helps local optimal solutions stand out. They also propose a local search operator that improves the performance of GA. In [5] propose a hybrid genetic algorithm with a multi-parent crossover which recombines more than two parents to obtain a single new offspring. In [6] present a simulation of a parallel quantum genetic algorithm model, in which they propose a migration scheme that applies a control strategy between universes (sub-populations). They also apply a quantum crossover strategy, where they transfer fitness information of one or more individuals from universe j to universe i. This transfer influences the evolution of universe i and uses a coarse grained parallel model in the GA. In [7] present an agent-based parallel approach with a genetic algorithm, where the initial micro-population and GA are executed by agents of JADE middleware. The communication between agents is via message passing within the island model. The micro-population is divided into small subpopulations and the best individuals migrate between islands. In [8] present a hybrid parallel micro genetic algorithm based on the combination of a GA with asynchronous colony and a GA with autonomous immigration. An autonomous migration operator is applied to exchange the best solution in the system. It uses two types of populations, large and small, and applies a coarse grained parallel model (island model). It adapts the system, on each island setting up three asynchronous colonies and one master colony. In [9] present a parallel hybrid GA with specialized crossover and mutation operators which use path-relinking concepts and taboo search. The GA uses model islands (independent subpopulations) of an initial population of size N. It creates a subset L which consists of the best individuals from each island, evaluated in every iteration of the GA on each island. If the fitness stagnates, the Hamming distance is calculated for all the individuals of L. In this case, two subpopulations are merged which reduces the number of subpopulations. The thread continues until there is a single population of size N. The results presented are benchmarks of small, medium and large size, and they are quite good. They have less than 2.5% RE for big problems, but do not present a study of the efficiency (speedup) of the parallel algorithm. In [10] present a parallel and agent-based local search genetic algorithm. The researcher applies a parallel local search by means of a multi agent system, each agent possessing a special behavior. Asadzadeh uses JADE middleware to build agents that communicate with each other in a peer-to-peer manner. The island model is used to implement the GA. The micro-population is divided into small subpopulations and the migration of the best individuals between islands is realized. The local search procedure uses the neighboring search variable method. In [11] present a hybrid island model genetic algorithm and proposes a naturally inspired self-adaptation phase strategy, which obtains a better balance between exploration and exploitation in the search within the solution space. The best individuals are selected to conduct a local search process and the worst individuals to conduct a global search process. The roulette operator is applied for selection in the migration of individuals. A coarse grained parallel model is used in the GA. In [12] present an island model genetic algorithm and proposes an inspired evolution model with different evolution methods. A naturally inspired migration selection mechanism is proposed, where the worst individuals are the first to migrate to other populations. This is done to improve the search for better solutions and to delay convergence. In [13] present a parallel artificial bee colony algorithm, where there are several colonies in different hosts of the network and exists communication between colonies by a dynamic migration strategy to determine when a colony must communicate with each other. In [14] present an island model memetic algorithm with a new naturally inspired cooperation phase to improve the exploitation capabilities of an island model. Several techniques are applied to improve the exploration capabilities, such as a diversity-based population, an incest prevention-based tournament selection method, and a similarity-and-quality based replacement method. In [15] present a hybrid algorithm that combines three heuristics, ant colony (AC), simulated annealing (SA) and GA, and it is executed in parallel with OpenMP directives. AC generates initial population, while in SA the perturbation is performed using the crossover genetic operator. Each offspring replaces parents using acceptance criterion in SA.

In this paper, a hybrid micro parallel genetic algorithm with collective processes communication is applied using the Message Passing Interface (MPI). The parallel model used is the master-slave. The crossing in each micro-population is done using a genetic approach oriented toward the best individual of the micro-population. Iterative mutation is applied to each individual by means of the simulated annealing algorithm.

This paper is structured in the following manner. Section one gives a brief introduction. Section two describes the mathematical model and disjunctive formulation of the job shop scheduling problem that is used. Section three presents the proposed Hybrid Micro Genetic Algorithm Multi-Population with Collective Communication. Section four shows the experimental results for some benchmarks of JSSP. Finally, the conclusions drawn by the present work are explained.

II. MATHEMATICAL MODEL AND DISJUNCTIVE FORMULATION OF JSSP

The mathematical model considers one objective function and various sets of constraints. The objective is to minimize the makespan, which is defined based on starting time s and processing time p of the last system operation j. It can be expressed as (1). There are several sets of constraints: a set J of n jobs, where $\boldsymbol{J} = \{J_1, J_2, \ldots J_n\}$; a set $\boldsymbol{M}$ of m machines where $\boldsymbol{M} = \{M_1, M_2, \ldots, M_m\}$; and a set $\boldsymbol{O}$ of operations where $\boldsymbol{O} = \{1, 2, 3, \ldots\}$. These operations form k subsets of operations for each one of the jobs ($J_k \subseteq \boldsymbol{O}$) and machines ($M_k \subseteq \boldsymbol{O}$).

Each operation j has a threading time of p_j. In a job J_k, each pair of operations i, j possesses a precedence relationship which is represented ($i \prec j$). Only one operation performed

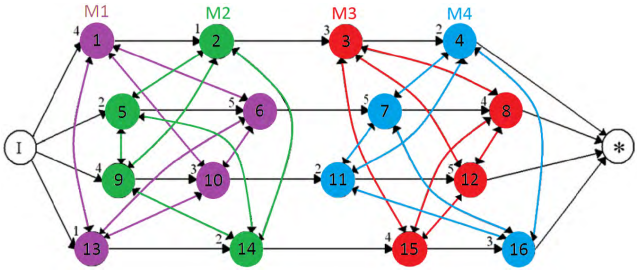


FIGURE 1. Representation of a JSSP with four jobs and four machines using a disjunctive graph.

by a machine M_k can be executed at any given point in time. The mathematical model can be represented in the following manner:

$$\text{Min } f = \left[\max_{j \in \boldsymbol{O}} \left(S_j + p_j\right)\right] \tag{1}$$

$$s_j \geq 0 \qquad \forall\, j \in \boldsymbol{O} \tag{2}$$

$$s_i + p_i \leq s_j \quad \forall\, i, j \in \boldsymbol{O},\ (i < j) \in J_k \tag{3}$$

$$s_i + p_i \leq s_j \vee s_j + p_j \leq s_i \quad \forall\, i, j \in \boldsymbol{O},\ (i, j \in M_k) \tag{4}$$

The constraints in (2) indicate that the operation's start time, *j*, must be greater than or equal to zero; meaning only positive values are accepted. Constraint (3) is a precedence constraint. It indicates that within one job which contains operations *i* and *j*, in order for *j* to begin, *i* must be completed. The constraints in (4) are disjunctive. These constraints ensure that two operations, *i* and *j*, which are performed by the same machine are not carried out simultaneously.

Figure 1 shows the disjunctive graph model $G = (A, E, O)$ for a JSSP of 4x4 (four machines and four jobs). This disjunctive graph is formed by three sets. The operations set, *O*, is made up of the nodes *G*, numbered one to sixteen. The threading time appears before each operation. The beginning and ending operations (I and $*$ respectively) are fictitious, with threading times equal to zero. Set *A* is composed of conjunctive arcs, each one of these arcs unites a pair of operations that belong to the same job. Operations 1, 2, 3 and 4 are connected by a conjunctive arc and therefore form job one. Operations 5, 6, 7 and 8, are connected by another conjunctive arc and therefore form job two. Jobs three and four are made up of the operations 9, 10, 11, 12, and 13, 14, 15, 16, respectively. Each arc of *A* represents a precedence constraint. For example, in job one, operation two must finish before operation three begins. Set *E* is composed of disjunctive arcs. Each arc that belongs to *E* unites a pair of operations that belong to the same machine. It can be seen that operations 1, 6, 10 and 13 are executed by machine one and united by disjunctive arcs. Operations 2, 5, 9 and 14 are executed by machine two and united by disjunctive arcs. Likewise, machines three and four execute operations 3, 8, 12, 15 and 4, 7, 11, 16 respectively. Each machine forms a clique (a subset of *E* that is completely connected). Each arc of *E* represents a resource capacity constraint between a pair of operations that belong to the same machine. This type of restriction indicates that the machine cannot execute more than one operation during the same time interval.

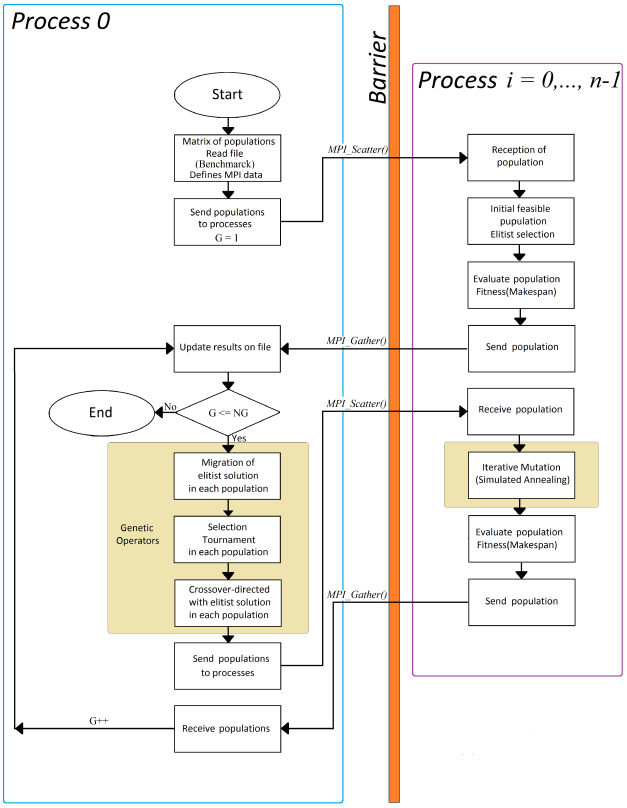


FIGURE 2. Hybrid micro genetic multi-population algorithm with collective communication (HGACC) for the JSSP.

III. HYBRID MICRO GENETIC MULTI-POPULATION ALGORITHM WITH COLLECTIVE COMMUNICATION (HGACC)

The HGACC works under the parallel master-slave model and performs communication by SPMD message passing. Under this parallel model, the most costly algorithm calculations (initial elite micro-population and iterative mutation) are performed by the slave process. The sharing of knowledge between populations (migration and approximation of genetic traits) is performed by the master process. The communication between computational nodes is collective. This communication requires synchronous sending of algorithm information, from the slave nodes to the master node as well as from the master node to the slave nodes. A communication block exists while data is being sent from the slave to the master nodes. This small block is released the instant the last process executed in the slave nodes finishes sending its information to the master node.

A. DESCRIPTION OF HGACC

Figure 2 shows the flow sequence of the hybrid micro genetic algorithm multi-population with collective communication:

1. Initially, the algorithm begins its execution at zero process. It defines the types of MPI data that are necessary

for the management of an array of structure matrices. With the definition of the new MPI data type, an MPI data matrix is created that will contain a set of micro-populations. Each row *i* defines a micro-population, and each coordinate (row *i,* column *j*) defines an individual (a solution to the problem). The genetic representation of an individual is shown in Figure 4, where the work and machine characteristics are presented. The file is read. The file contains information on the problem (benchmark) which is stored in the matrix of micro-populations in each coordinate (*i, j*).

2. The first generation is defined and the data is sent to the *n* processes collectively, by means of the *MPI_Scatter()* function. The micro-populations are distributed evenly among the processes, one micro-population per process.
3. In each process *i*, a micro-population of feasible individuals is generated through an elitist selection. This selection is bound by a maximum aptitude value, which should not be exceeded (see section III.B).
4. The fitness of individuals in all micro-populations is evaluated.
5. In each process *i,* the micro-population is sent to process zero collectively with the other *n-1* processes by means of the *MPI_Gather()* function.
6. Micro-populations are synchronically sent to process zero from all n-1 processes. The *MPI_Barrier()* function is used for synchronization.
7. The necessary results from each micro-population are stored in the output file, as is the fitness value, the hamming distance and the execution time.
8. The stop criterion is evaluated. If the number of generations has been reached, then the algorithm ends.
9. Genetic operators are applied independently in each micro-population: elite migration (see section III C), tournament selection, and genetic trait approximation (*crossover-directed*, see section III D). In each micro-population, the tournament selection randomly chooses a pair of individuals, compares them, and chooses the one with the best fitness. Then the *crossover-directed* is conducted with the individual that has the best fitness in the micro-population, and the winner of the tournament. This selection is made to complete the size of the micro-population.
10. After applying the genetic operators, the micro-populations obtained in the *n* processes are sent collectively by means of the *MPI_Scatter()* function. They are evenly distributed among the micro-populations with the *n* processes, one micro-population per process.
11. In each process *i*, the iterative mutation operator is applied to each individual of the micro-population, by means of simulated annealing quenching (see section III E).
12. The fitness of individuals in all micro-populations is evaluated.

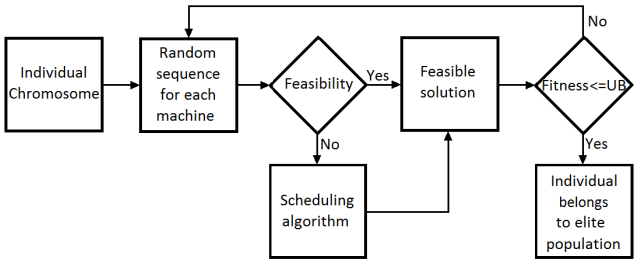


FIGURE 3. Creation of individuals for the initial elite micro-population.

13. In each process *i,* the micro-population is sent to process zero collectively with the other *n-1* processes by means of the *MPI_Gather()* function.
14. Information is sent to zero process for all *n-1* processes. The sending is synchronized using the *MPI_Barrier()* function.
15. The new generation is counted.
16. Algorithm execution returns to step 7.

B. INITIAL CONSTRUCTION OF ELITE MICRO-POPULATIONS

Due to the time it takes to construct each initial elite micro-population, the work is distributed among slave processes. Figure 3 presents the construction of a feasible individual to be part of an initial elite micro-population.

Each individual in the micro-population is randomly generated. The scheduling algorithm [16], is used to repair the population that was created randomly at the beginning. Therefore, it does not affect the performance of HGACC, i.e., it only repairs solutions that are infeasible to have an initial population completely feasible. Afterwards, there is no need to repair anymore since all solutions generated by HGACC are then feasible.

The scheduling algorithm works with a symbolic representation of integer values to represent an individual [16]. The representation does not require a binary encoding, so a decoding of a genotype to a phenotype is not needed either, which makes HGACC a more efficient algorithm because it reduces time consumption in every generation. In [16], two representations are employed: an integer (phenotype) and a binary type (genotype), but in this work it is exclusively used the integer representation. The genetic representation of an individual is generated by two matrices with integer values (Figure 4). There are other convenient representations for JSSP, such as those explained in [17]. The integer representation [16], is suitable for applying the algorithm of scheduling which is implemented in all HGACC; first, to repair the initial population by converting infeasible solutions to feasible ones and then, to merely calculate the value of fitness of each individual since it is no longer required to repair any individual.

To accept the new individual feasible randomly generated as part of the elite micro-population, its fitness value must be less than or equal to an upper bound. This upper

	J/M	Operations for J				Operations for M			
Chromosome 1	1	1	2	3	4	13	1	10	6
Chromosome 2	2	5	6	7	8	5	2	9	14
Chromosome 3	3	9	10	11	12	15	3	12	8
Chromosome 4	4	13	14	15	16	7	11	16	4

FIGURE 4. Representation of an individual and its chromosomes for a 4x4 JSSP problem.

bound is defined by a tuning, which is a function of the time needed to obtain the total number of required micro-populations. This time should not exceed 5 minutes. Once the new individual has a fitness less than or equal to the upper bound, it becomes part of an initial elite micro-population. Obtaining an elite micro-population allows the HGACC to initiate in a good solution space. There is great diversity in the genetic traits among individuals in any given micro-population, because the creation of each individual is random. The creation of micro-populations benefits the efficiency of HGACC, because the number of individuals is small. There is a maximum of 5 individuals per micro-population, which also allows for the convergence of each micro-population without considering the length of the chromosome that forms the individuals [18]. The migration operator avoids premature convergence in the HGACC.

C. MIGRATION ELITE

In HGACC, diversification is performed by iterative simulated annealing and the exploitation of the search space is made through genetic approximation. Yet, the diversification and exploitation is improved in HGACC with the migration of elite solutions in each population, which improves convergence towards better solutions.

The migration process is performed between micro-populations in the master process. Access to all the micro-populations is obtained, which avoids the transfer of data between nodes of the computational cluster. This transfer could otherwise lead to slack time in the algorithm, because HGACC works synchronously. In addition, the communication process could be less efficient because there would be small information packets transferred between computational nodes. This inefficiency would be reflected in the speed up of the algorithm. Migration applies to only one individual from each micro-population, to the individual with the best fitness value (elite individual). With this type of migration, each elite individual transfers its genetic traits to another micro-population by applying the *crossover-directed* procedure. In the next generation, the new elite individual from each micro-population migrates to another micro-population chosen randomly but in a balanced way. This elite migration procedure maintains the diversity of each micro-population and avoids early convergence of the algorithm.

D. APPROXIMATION OF GENETIC TRAITS

The genetic approximation process is performed independently in each micro-population via the master process. This genetic approximation is performed by pairs of individuals, applying the *crossover-directed* operator (elite individual, individual *i*), to the *n-1* individuals of the micro-population.

The crossover consists of the following steps. From the elite individuals, all possible feasible neighbors (offspring) are generated. They are generated using the neighborhood function N_1 [19] that performs the permutation of pairs of adjacent operations which belong to the critical path generated in the individual. Each feasible neighbor k is generated by a single permutation of the elite individual. The Hamming distance of each neighbor k is evaluated with respect to the individual i. Neighbor k is chosen because it has greater resemblance to individual i, demonstrated by its smaller Hamming distance. Neighbor k substitutes individual i, hence k is integrated into the micro-population. The genetic approximation between individuals allows each micro-population to perform a search directed toward a good solution space.

Genetic approximation makes an intensified search, choosing the minimum Hamming distance of the offspring k in regard to the individual i. Thus, the resulting offspring will acquire the biggest resemblance to the individual i by obtaining greater genetic characteristics of it, but it also maintains the genetic characteristics of the elite individual. This can be seen as an implicit crossover in which the procedure selects an offspring k without taking into account its fitness value, but keeping the genetic characteristics of a solution where the elite individual is selected. By employing this procedure, the premature convergence of the algorithm is prevented and the exploration of a much larger space of solutions is allowed as well, all by keeping good genetic characteristics. At this point of the algorithm it is in some cases sacrificed part of its performance when lower quality solutions are chosen for maintaining a further exploration to preserve a good performance on a global basis. Preliminary tests in HGACC showed that if only offsprings k are chosen to improve the solution, the algorithm converges prematurely to solutions of much lower quality than those obtained in this work. The genetic approximation manages a tabu list that prevents from choosing the last 8 offspring k(tuned value) previously selected from the same elite individual.

There are other types of neighborhoods, such as N4, N5, N6 and the one proposed in [20]. However, N1 is selected to generate small changes in the genetic characteristics of the new offspring obtained from the elite individual. Hence, the elite individual presents a very large neighborhood when applying N1, which makes it easier to find offspring with closer genetic characteristics to the individual i, due to a wider range of offspring options to compare and select.

The crossover is an implicit process since once the offspring is picked, it shows genetic characteristics of both the elite individual and the individual i. The advantage of the implicit crossover is that the chosen offspring will never be an

infeasible solution, and therefore will not require any type of repair to make it feasible, so this is what makes the algorithm more efficient in its execution.

E. ITERATIVE MUTATION WITH SIMULATED ANNEALING

Iterative mutation is applied to each individual in each micro-population by the simulated annealing algorithm (SA). The iterative mutation allows an exploitation of the solution space through neighborhood searches in each individual. The applied mutation uses the neighborhood function N_1 and the critical path generated in the individual. From this path, pairs of adjacent operations are exchanged. Each permutation of operation pairs is identified as a mutation on one of the chromosomes of an individual. Figure 4 shows the representation of the genetic information of a solution ($S_{individual}$) of a feasible individual for the JSSP problem in Figure 1, with 4 machines and 4 jobs. Each row defines one of the individual's chromosomes. It provides information about the operations required to execute each job according to the precedence order. It also specifies information about the operations executed by each machine, consistent with the precedence order defined by the problem solution. For example, on chromosome 1, the first column indicates the job number (J_1) and machine number (M_1). Columns 2 through 5 indicate the operations required by J_1. According to the precedence order defined by the JSSP, operation O_1, is executed first, then O_2, followed by O_3, and finally O_4. Columns 6 through 9 indicate the operations that have to be performed by M_1. According to the feasible solution that represents the individual, the precedence order of operations for M_1 requires that O_{13} is executed first, then O_1, followed by O_{10}, and finally O_6. The permutation of a pair of adjacent operations is done only on machines. For example, the adjacent pair (O_{13}, O_1), which belongs to chromosome 1, can be swapped if both operations belong to the critical path. This would be a mutation in the individual ($S_{individual}$), as shown in Figure 4. The mutation generates a new solution ($S'_{individual}$). The simulated annealing algorithm performs the mutation in an iterative way, as presented in Algorithm 1.

The iterative mutation using simulated annealing is presented in Algorithm 1.

1. $S_{individual}$ is the individual (solution of JSSP). The variable k counts the number of SA executed in the iterative mutation. The following are initialized: the final temperature T_f, the initial temperature T_o, the coefficient of temperature β, and the best solution S_{best}, obtained by iterative mutation (at the beginning this is a very large value).
2. The internal cycle begins, which executes the Metropolis algorithm until equilibrium is achieved, when the size of the Markov Chain (MC) is reached. A neighborhood function N_1 is used. This generates a mutation $S'_{individual}$. This mutation is accepted as a new individual if the energy in the system decreases. If the energy in the system increases, $S'_{individual}$ is accepted as a new individual according to the acceptance probability, P_{accept}, obtained by the Boltzmann function. If the new individual fitness, ($S_{individual}$), is better than the best fitness, (S_{best}), then S_{best} is upgraded.
3. The temperature is decreased with β. Step 5 continues as long as the final temperature is not reached in the SA.
4. If the number of SA (NSA) for iterative mutation has not been reached, then a new SA is begun in step 3, using the new individual $S_{individual}$ obtained.
5. S_{best} is the new individual obtained by iterative mutation that replaces the individual that initiated SA.

Algorithm 1 Simulated Annealing Algorithm That Implements Iterative Mutation

1. *Obtain* $S_{individual}$
2. *Perform an initial iteration* $k = 0$, *initialize values of* T_o, T_f, β, $S_{individual}$, S_{best}
3. *Start the annealing* $k = k + 1$:
   4. $T = T_o$
   5. *While the final temperature* T_f *is not reached,*
   6. *While equilibrium is not reached:*
      6.1. *Generate state* $S'_{individual}$ *by means of a mutation in* $S_{individual}$
      6.2. *If fitness*($S'_{individual}$) - *fitness*($S_{individual}$) $<= 0$ *the state is accepted as the current state,* $S_{individual} = S'_{individual}$
      6.3. *If fitness*($S'_{individual}$) - *fitness*($S_{individual}$) > 0 *the state is accepted with the probability*

$$P_{accept} = e^{-\left(\frac{fitness\left(S'_{individual}\right) - fitness(S_{indivudual})}{T}\right)}$$

      6.4. *Generate a random number* α, *uniformly distributed between (0,1)*
      6.5. *If* $\alpha < P_{accept}$ *the state is accepted as the current state,* $S_{individual} = S'_{individual}$
      6.6. *If fitness*($S_{individual}$) $<$ *fitness*(S_{best}) *then* $S_{best} = S_{individual}$
      6.7. *If equilibrium is not reached, return to 6*
   7. $T = T * \beta$.
   8. *If* $T >= T_f$, *return to 5*
9. *If* $k <$ ***NSA***, *return to 3 in order to begin a new annealing*

Simulated annealing does not necessarily choose a new solution that improves the individual, since it depends on the acceptance criterion involving the Boltzmann function, which also accepts poor solutions, allowing escaping from local optimal solutions. So when this strategy is employed it finds the best solutions thereafter. Hence, by mutating iteratively and using specific information of the problem, such as mutating a position in the order of operations that belong only to the critical path, the result is a new genetic material of better quality that did not exist previously.

IV. EXPERIMENTAL RESULTS

A computational cluster was used in these experimental tests, consisting of four nodes, a Motherboard with two processors, six core, Intel Xeon 3.06GHz, 48 total cores, 96GB RAM, switch InfiniBand 40Gb/s, each node with InfiniBand board 40Gb/s, O. S. Centos 5.5. gcc compiler, MPI libraries.

In this paper, HGACC is evaluated only with square instances because they are more difficult to solve than the rectangular type [21]. Fifty-four problems of different sizes were taken from the OR-Library [22]. The problem instances used were: FT06, FT10 [23], ORB01 to ORB10 [24], ABZ5, ABZ6 [25], LA16 to LA20, LA36 to LA40 [26], TA1 to TA10, TA21 to TA30 [27], YN1, YN2, YN3, YN4 [28], DMU06 to DMU10 and DMU46 [29]. HGACC is compared with twenty algorithms that are found in the literature, most of population. The algorithms are: BRK-GA. A Biased Random-Key Genetic Algorithm [30]. SAGen. Simulated Annealing Genetic [31]. ACOFT-MWR. Ant Colony Optimization with Fast Taboo - Most Work Remaining. [32]. TSSA. Tabu Search and Simulated Annealing [33]. HPSO Hybrid Particle Swarm Optimization [34]. EPPX. Extended Precedence Preservative Crossover [5]. PPSO. Parallel Particle Swarm Optimization [35]. cGA-PR. Coarse-Grained Genetic Algorithm with Path-Relinking [9]. PaGA. Parallel Agent-Based Genetic Algorithm [7]. HGAPSA. Hybridization of Genetic Algorithm with Parallel Implementation of Simulated Annealing [36]. HIMGA. Hybrid island model genetic algorithm [11]. NIMGA New island model genetic algorithm [12]. IIMMA, improved island model memetic algorithm [14]. TGA, Tabu-based Genetic Algorithm [37]. IEBO, Guided local search with Iterative Ejections of Bottleneck Operations [38]. TS/PR, Tabu Search/Path Relinking, [39]. AntGenSA, hybrid algorithm Ant Colony System-Simulated Annealing-Genetic Algorithm [15]. UPLA, Upper-Level Algorithm [40]. PABC, Parallel Artificial Bee Colony Algorithm [13]. ALSGA, Agent-based Local Search Genetic Algorithm [41].

A sensibility analysis was carried out in order to define the parameters of HGACC as Elite Upper Bound (*UB*), Generations Number (*GEN*), Micro-populations Number (*NPOP*), Micro-populations Size (*SPOP*), Crossover rate (*% CROSS*), and Number of simulated annealing (*NSA*). For simulated annealing, the following were defined: initial temperature (T_0), final temperature (T_f), coefficient of temperature (α) and Markov Chain (MC). Table 1 shows the results of this analysis for small, medium and large problems.

HGACC was executed with several different numbers of micro-populations, (each micro-population is executed by a process). Figure 5 presents, for problem YN1, the diversity that exists in individuals applying *crossover-directed*, and the influence of the elite migration operator, with different numbers of micro-populations (2, 6, 12, 24, and 48). It is observed that as the number of micro-populations increases in the algorithm and the time approaches 2 hours, the diversity of the individuals tends to decrease because the hamming distance reduces in value. This indicates that as the number of micro-populations increases, the influence of the elite migration operator also increases because the other parameters of the algorithm are applied independently in each micro-population. The only parameter that has a global influence on the micro-populations is the elite migration operator.

TABLE 1. Tuned parameters of hybrid micro genetic multi-population algorithm with collective communication (HGACC).

Problem	*GEN*	*NPOP*	*SPOP*	*%CROSS*	*NSA*	T_o	T_f	α	*MC*	*Elite UB*
Size 10X10, 15X15										
FT10, A16 to LA20, ORB01 to ORB10, ABZ05 and ABZ06,LA36 to LA40, TA01 to TA10	20	48	5	100	24	25	1	0.98	1000	2500
Size 20X20										
YN1, YN2	20	48	5	100	24	25	1	0.98	1000	2000
YN3	20	48	5	100	24	25	1	0.98	1000	1830
YN4	50	48	5	100	24	30	0.5	0.99	1000	1930
DMU06	20	48	5	100	24	25	1	0.98	420	9000
DMU07	20	48	5	100	24	25	1	0.99	420	8000
DMU08	20	48	5	100	24	25	1	0.98	420	7500
DMU09	20	48	5	100	24	25	1	0.98	420	6700
DMU10	20	48	5	100	24	25	1	0.98	420	6500
DMU46	20	48	5	100	48	350	0.01	0.98	420	8700

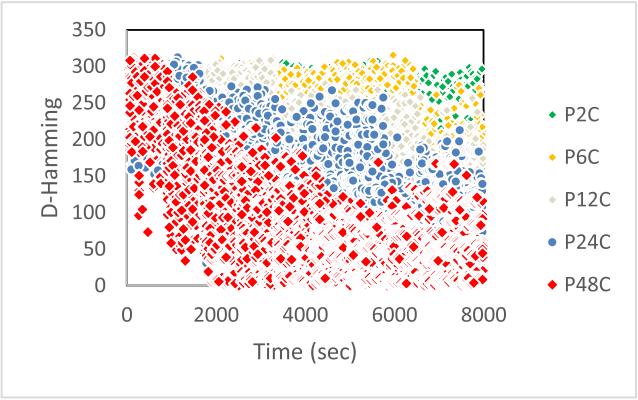
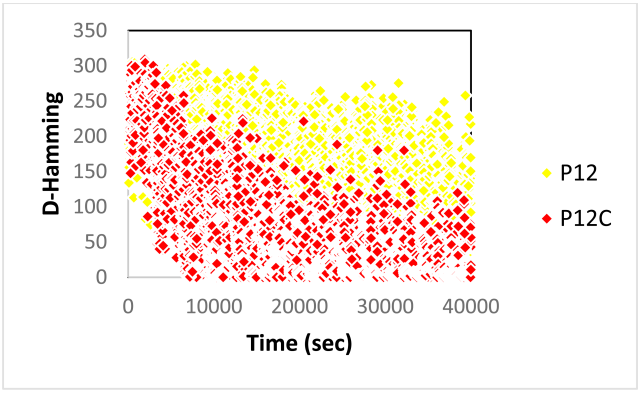
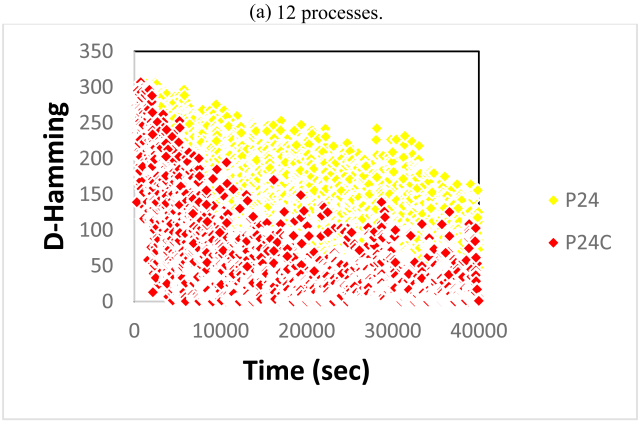


FIGURE 5. YN3-20x20. Solution diversity in HGACC with different numbers of micro-populations (processes 2, 6, 12, 24 and 48), with crossover-directed. Average of 30 tests.

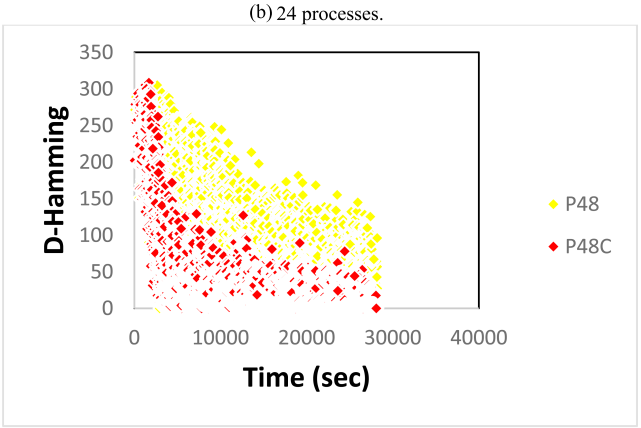
Figure 6 shows the influence of the *crossover-directed* operator for the problem YN1, when applied and not applied in micro-populations. It is observed in a time interval of 1.5 to 11 hours, when *crossover-directed* is not applied, the population diversity is larger for any number of micro-populations (processes, P12, P24 and P48) than when *crossover-directed* is applied (P12C, P24C and P48C). When *crossover-directed* is applied, diversity tends to converge faster, as shown by the Hamming distances for any number of processes (P12C, P24C and P48C). It is also observed that whether *crossover-directed* is applied or not, there is a reduction in the diversity of the micro-populations as the number of processes increases. This is due to the influence of the elite migration operator. In each iteration of the HGACC, the elite individual shares the genetic characteristics of the micro-population



(a) 12 processes.



(b) 24 processes.



(c) 48 processes.

FIGURE 6. YN1-20x20. Solution diversity in HGACC with different numbers of micro-populations, processes (a) 12, (b) 24, c (48). Average of 30 tests, with crossover-directed (C) and without *crossover-directed*.

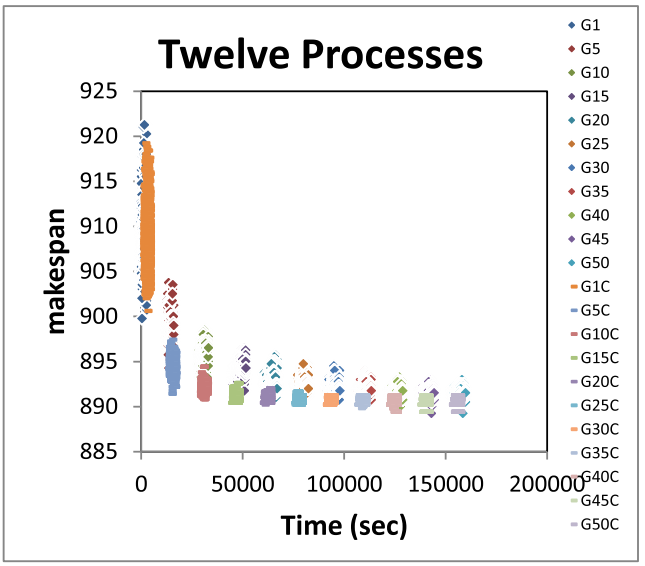


FIGURE 7. YN1-20x20. Performance of HGACC in different numbers of generations and twelve micro-populations, without crossover-directed (G1-G50) and with crossover-directed procedure (G1C-G50C). Average of 30 tests.

TABLE 2. Results obtained with HGACC for small JSSP problems. 30 tests.

Problem	Op/UB	Better	GEN	Worse	GEN	Average	%RE	σ	t(Sec)	MED	MOD
Size 10 x 10											
FT10	930	930	1	930	1	930	0	0	5.31	**930**	**930**
LA16	945	945	1	945	1	945	0	0	0.06	**945**	**945**
LA17	784	784	1	784	1	784	0	0	0.16	**784**	**784**
LA18	848	848	1	848	1	848	0	0	0.21	**848**	**848**
LA19	842	842	1	842	1	842	0	0	0.1	**842**	**842**
LA20	902	902	1	902	1	902	0	0	0.07	**902**	**902**
ORB01	1059	1059	1	1059	2	1059	0	0	0.23	**1059**	**1059**
ORB02	888	888	1	888	2	888	0	0	0.28	**888**	**888**
ORB03	1005	1005	1	1005	1	1005	0	0	1.84	**1005**	**1005**
ORB04	1005	1005	1	1005	1	1005	0	0	0.1	**1005**	**1005**
ORB05	887	887	2	887	7	887	0	0	1.76	**887**	**887**
ORB06	1010	1010	1	1010	4	1010	0	0	15.67	**1010**	**1010**
ORB07	397	397	1	397	1	397	0	0	0.08	**397**	**397**
ORB08	899	899	1	899	1	899	0	0	0.38	**899**	**899**
ORB09	934	934	1	934	1	934	0	0	10.21	**934**	**934**
ORB10	944	944	1	944	1	944	0	0	7.47	**944**	**944**
ABZ5	1234	1234	1	1234	1	1234	0	0	11.04	**1234**	**1234**
ABZ6	943	943	1	943	1	943	0	0	9.37	**943**	**943**

from which it emigrated to the micro-population to which it arrived. For this reason, a reduction in the diversity of individuals between micro-populations is observed. If there is a large number of micro-populations (number of processes), more elite individuals will emigrate to share their genetic traits with very small populations, because for every micro-population there is an elite individual who migrates to another micro-population. If populations are very large, the influence of the elite migration operator decreases.

Figure 7 presents the influence of the *crossover-directed* operator for the YN1 problem, as reflected in the performance of HGACC as the number of generations increases. For a set of 12 micro-populations, it can be observed that in any generation, the set of micro-populations is more concentrated when the *crossover-directed* operator is used and more dispersed when the *crossover-directed* operator is not used. The dispersion of the micro-populations shows that a large number of individuals do not display good results with respect to makespan. The best performance of the set of micro-populations in HGACC occurs when the *crossover-directed* operator is applied. Independently, however, in generations 45 and 50, some individuals are observed to improve

TABLE 3. Results obtained with HGACC for medium problems of JSSP. 30 tests.

Problem	Op/UB	Better	GEN	Worse	GEN	Avg.	%RE	σ	t(Sec)	MED	MOD
Size 15 x 15											
LA36	1268	1268	2	1268	2	1268	0	0	313	**1268**	**1268**
LA37	1397	1397	4	1397	12	1397	0	0	1084	**1397**	**1397**
LA38	1196	1196	1	1196	3	1196	0	0	116	**1196**	**1196**
LA39	1233	1233	2	1233	8	1233	0	0	295	**1233**	**1233**
LA40	1222	1222	3	1224	2	1224	0	0.84	2877	1224	1224
TA01	1231	1231	1	1231	2	1231	0	0	234	**1231**	**1231**
TA02	1244	1244	1	1244	8	1244	0	0	113	**1244**	**1244**
TA03	1218	1218	9	1221	4	1219	0	1.27	2353	**1218**	**1218**
TA04	1175	1175	1	1175	7	1175	0	0	62	**1175**	**1175**
TA05	1224	1224	13	1227	7	1225	0	1.55	4065	**1224**	**1224**
TA06	1238	1238	15	1240	8	1239	0	0.84	4216	1240	1240
TA07	1227	1228	1	1228	1	1228	0.08	0	59	1228	1228
TA08	1217	1217	1	1217	1	1217	0	0	62	**1217**	**1217**
TA09	1274	1274	2	1280	2	1276	0	3.10	447	**1274**	**1274**
TA10	1241	1241	1	1241	1	1241	0	0	66	**1241**	**1241**

TABLE 4. Results obtained by HGACC for 20X20, large problems of JSSP. 30 tests.

Problem	Op/UB	Better	GEN	Worse	GEN	Avg.	%RE	σ	t(Sec)	MED	MODE
Size 20 x 20											
DMU06	3244	3268	16	3381	17	3283	0.74	5.25	29868	3285	3285
DMU07	3046	3064	14	3223	2	3083	0.59	12.62	83791	3084	3084
DMU08	3188	3188	5	3385	20	3208	0	9.06	1836	3209	3199
DMU09	3092	3109	11	3231	20	3139	0.55	8.36	2379	3140	3140
DMU10	2984	2987	13	3084	10	3004	0.10	9.63	2813	3004	3004
DMU46	4035	4099	4	4189	8	4115	1.59	11.36	9973	4116	4103
YN1	884	886	13	905	6	890	0.23	0.42	13794	890	890
YN2	904	907	8	930	43	907	0.33	0	6502	907	907
YN3	892	892	6	915	1	895	0	1.43	4548	895	895
YN4	968	969	44	990	49	970	0.10	0.67	46328	970	970

when *crossover-directed* is not applied. The convergence of HGACC, is observed to start in generation 45C.

Figure 8, presents the influence of the *crossover-directed* operator for the problem YN1, as reflected in the performance of HGACC as the number of generations increases. For a set of 24 micro-populations, it can be observed that in any generation, the set of micro-populations is more concentrated when the *crossover-directed* operator is used, and more dispersed when the *crossover-directed* operator is not used. The dispersion of micro-populations shows that a large number of the individuals do not show good results with respect to makespan. The best performance of the set of micro-populations in HGACC occurs when the *crossover-directed* operator is applied. In addition, in the generations 5C to 50C, it can be observed that a large number of individuals reach the best solutions when applying this operator. The convergence of HGACC is observed starting at 25C.

Figure 9, presents the influence of the *crossover-directed* operator for the problem YN1, as demonstrated by the performance of HGACC as the number of generations increases. For a set of 48 micro-populations, it can be observed that

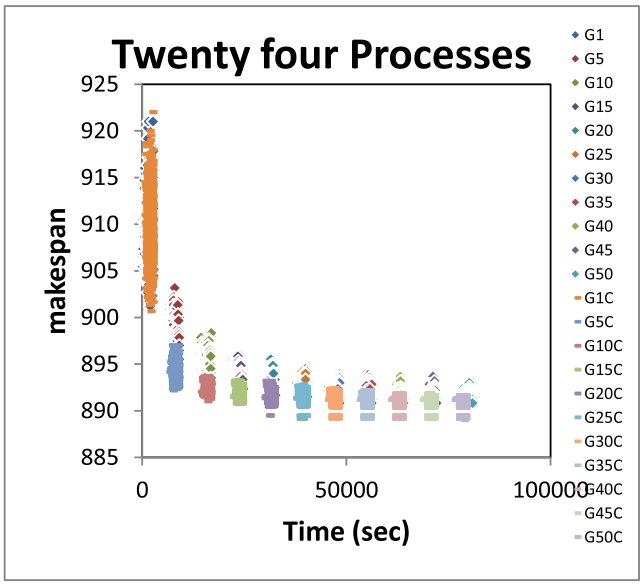


FIGURE 8. YN1-20x20. Performance of HGACC in different generation numbers and twenty four micro-populations. Without crossover-directed operator (G1-G50). With crossover-directed operator (G1C-G50C). Average of 30 tests.

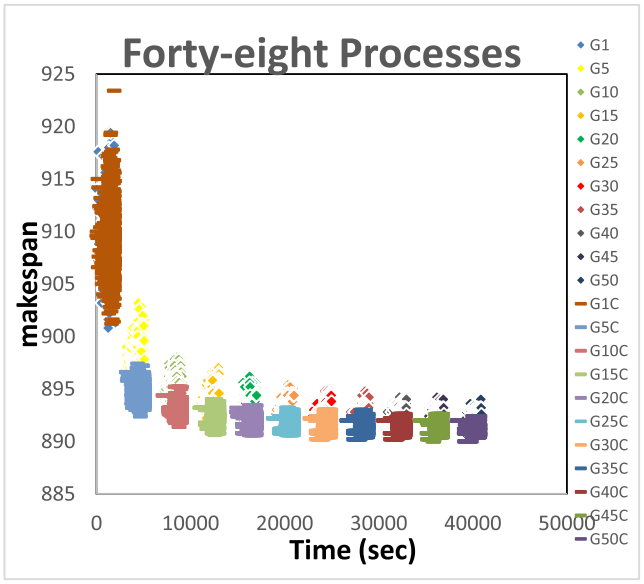


FIGURE 9. YN1-20x20. Performance of HGACC with forty-eight micro-populations and different numbers of generations. Without crossover-directed operator (G1-G50). With crossover-directed operator (G1C-G50C). Average of 30 tests.

in any generation, the set of micro-populations is more concentrated when the *crossover-directed* operator is used, and more dispersed when the *crossover-directed* operator is not used. The dispersion of the micro-populations shows that a large number of individuals do not display good results with respect to makespan. The best performance of the set of micro-populations in HGACC occurs when the *crossover-directed* operator is applied. In the 5C to 50C generations, it is observed that a large number of individuals improve with the application of the operator. The convergence of HGACC is observed starting in generation 30C.

TABLE 5. Efficiency and efficacy results for Fisher, Thompson, and Lawrence benchmarks.

Problem	Op/UB	%RE																				
		HG	tsec	B-G	tsec	SAG	tsec	A-M	tsec	TSSA	tsec	HPSO	tsec	TGA	tsec	TS/PR	tsec	AGS	tsec	UP	tsec	AGA
Size 10 x 10																						
FT10	**930**	0	5.31	0	10.1	--	--	0	65	0	3.8	0	4.1	0	0.06	0	4.75	0.9	557	0	1208	0
LA16	**945**	0	0.06	0	4.6	0	39	--	--	--	--	0	19.9	0	0.094	0	0.15	0	304	0	1458	0.10
LA17	**784**	0	0.16	0	4.6	--	--	--	--	--	--	0	19.9	0	0.016	0	0.08	--	--	0	78	0
LA18	**848**	0	0.21	0	4.6	--	--	--	--	--	--	0	19.9	0	0.015	0	0.09	--	--	0	76	0
LA19	**842**	0	0.1	0	4.6	0	35	--	--	0	0.5	0	19.9	0	0.025	0	0.16	--	--	0	1130	1.18
LA20	**902**	0	0.07	0	4.6	--	--	--	--	--	--	0	19.9	0	0.031	0	0.11	--	--	0	1304	0.55
Size 15 x 15																						
LA36	**1268**	0	313	0	21.4	0	4655	0	37	0	9.9	0	105	0	0.57	0	4.5	--	--	0.79	48387	--
LA37	**1397**	0	1084	0	21.4	0.31	4144	0	880	0	42.1	0	105	0	0.51	0	26.2	--	--	0.72	49836	--
LA38	**1196**	0	116	0	21.4	0.42	5049	0	55	0	47.8	0	105	0	1.25	0	32.6	--	--	1.59	50876	--
LA39	**1233**	0	295	0	21.4	--	--	0	66	0	28.6	0	105	0	0.5	0	11.6	--	--	1.38	50603	--
LA40	**1222**	0	2877	0	21.4	0.33	4544	0.16	941	0.16	52.1	0.16	105	0.16	0.86	0	385	--	--	0.57	50609	--

AntGenSA (AGS). Intel® Xeon® 2.3 GHz, 64GB [15]
BRK-GA (B-G). 2.2GHz AMD Opteron, [30]
SAGen (SAG). Pentium 120, 0.12 GHz, [31]
ACOFT-MWR (A-M) 1.533GHz CPU. [32]
TSSA. Pentium IV 3.0GHz, [33]
HPSO. AMD Athlon 1700+, 1.47 GHz, [34]
TGA. PC 2.2 GHz, 8Gb, [37]
TS/PR. PC AMD Athlon 3 GHz, 2 GB [39]
UPLA, (UP) Intel CoreTM i5, 2.67 GHz, 6GB , [40]
ALSGA (AGA), Intel core duo, 2.93 GHz, 2.0GB, [41]
HGACC (HG). CLUSTER, 48cores, Xeon 3.06GHz, this work

Figures 7, 8 and 9 show that when the *crossover-directed* operator is applied and the number of populations in the HGACC increases, the populations tend to be more dispersed. This is due to the fact that when there are a greater number of populations (number of processes), it is possible to carry out a greater exploration in the solutions space because there are a greater number of individuals working on the HGACC. For example, if there are 5 individuals in each micro-population, and 12 populations, there are a total of 60 individuals. If there are 48 populations, there are a total of 240 individuals. Therefore, in general, large populations obtain a greater variety in the aptitude of the individuals, which is reflected in Figure 9.

Figure 10 shows the histograms for the DMU06, DMU07, DMU08, DMU09 and DMU10 problems. In three histograms, HGACC presents frequencies in which the distribution is skewed to the right of the midpoint. In one histogram, HGACC presents frequencies in which the distribution is skewed to the left. In the other histogram, HGACC presents frequencies with central distribution.

The frequency distribution for DMU06 (Figure 10a) is an isolated peak (22 results), which is skewed to the right of the midpoint. Fitness results are in the range of 3282-3287. There are 2 results at the fitness midpoint (makespan). At the extremes of the histogram, a greater tendency is observed for obtaining quality results, with a ratio of 3 quality results to 2 poor ones.

The frequency distribution for DMU07 (Figure 10b) is skewed to the left of the midpoint. It can be seen that there are 6 results at the midpoint of the makespan. Results are in the range of 3082-3091. At the extremes of the histogram, a greater tendency is observed for obtaining quality results, with a ratio of 9 quality results to 3 poor ones.

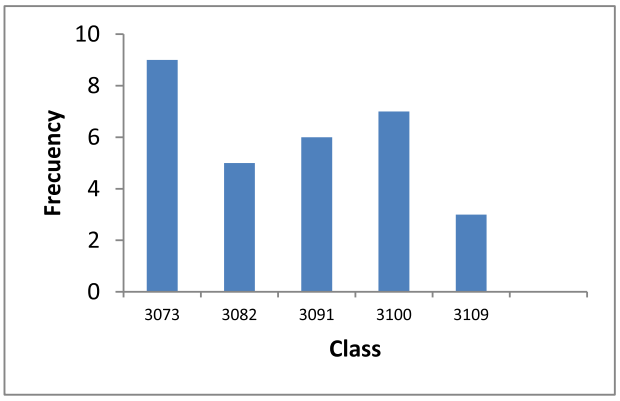
The frequency distribution for DMU08 (Figure 10c) is skewed to the right of the midpoint. It can be seen that the most frequent result, of which there are 8, is to the right of the makespan. Results are in the range of 3215-3222. It can be seen that there are 7 results at the midpoint of the makespan. At the extremes of the histogram, a smaller tendency is observed for obtaining quality results, with a ratio of 2 quality results to 8 poor ones.

DMU09 (Figure 10d) shows an isolated peak frequency distribution which is skewed to the right of the midpoint. It also shows that the most frequent result, of which there are 14, is to the right of the makespan. Results are in the range of 3135-3144. It can be seen that there are 8 results at the midpoint of the makespan. At the extremes of the histogram, a smaller tendency is observed for obtaining quality results, with a ratio of 1 quality result to 7 poor ones.

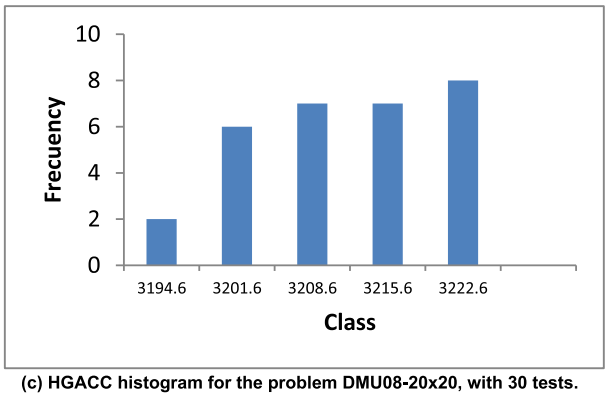
DMU10 (Figure 10e) shows a central frequency distribution. The most frequent result for the makespan is reported 9 times. Results are within the range of 3001-3008. At the extremes of the histogram, a greater tendency is observed for obtaining quality results, with a ratio of 5 quality results to 4 poor ones.

Graphs 10(a), 10(c) and 10(d) show the right skewed distribution, which indicates that more than half of the makespan obtained in the 30 tests of HGACC are poorer than the arithmetic mean, which is the midpoint of the histograms.

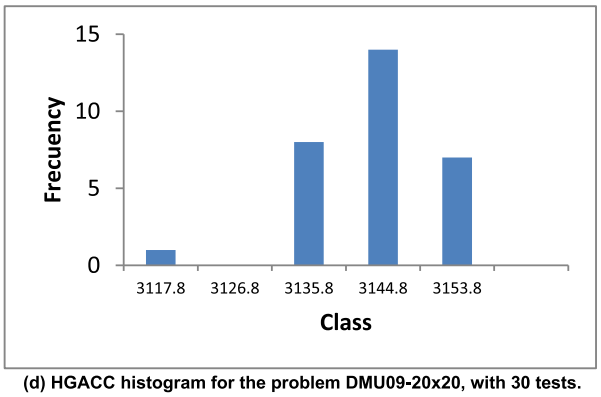
(a) HGACC histogram for the problem DMU06-20x20, with 30 tests.



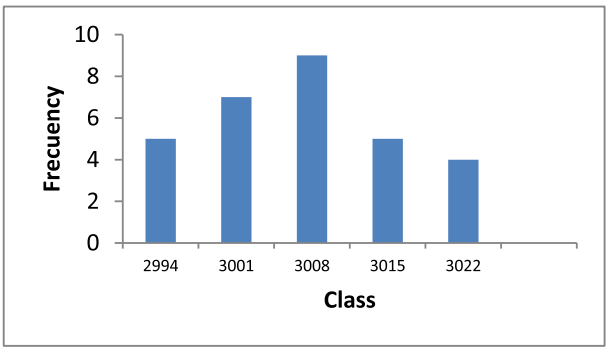
(b) HGACC histogram for the problem DMU07-20x20, with 30 tests.



(c) HGACC histogram for the problem DMU08-20x20, with 30 tests.



(d) HGACC histogram for the problem DMU09-20x20, with 30 tests.



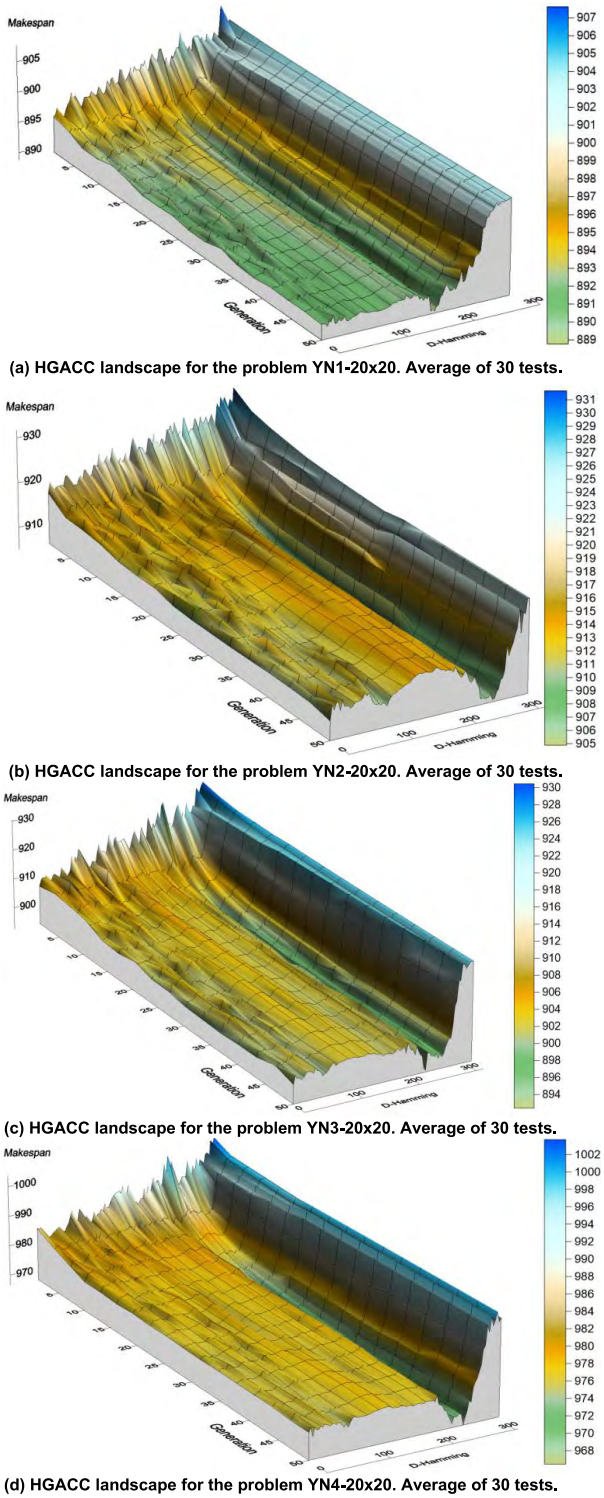
(e) HGACC histogram for the problem DMU10-20x20, with 30 tests.

FIGURE 10. HGACC histograms for the problems DMU (06-10)-20x20.

In 10(b) and 10(e), the left skew indicates that more than half of the makespan obtained in the 30 tests of HGACC are better than the arithmetic mean.

Figure 11 presents the landscapes for problems YN1, YN2, YN3 and YN4, when all genetic operators (*tournament selection*, *elite migration*, *crossover-directed*, *iterative mutation*) are applied. The values of the HGACC parameters are presented in Table 1. The graphs are presented according to three important parameters of the HGACC: the diversity between the individuals (D-Hamming), the convergence of the algorithm (number of generations), and the fitness of the individuals (makespan). These graphs are analyzed with respect to the 240 total individuals present in the 48 micro-populations.

It is observed in the four graphs that at the beginning, during the first five generations, the fitness of the individuals have the poorest values. These values improve as the number of generations increases and the diversity of individuals decreases. There is also a group of diverse individuals who always maintains a very poor aptitude value as the generations pass. For the YN1 problem, Figure 11 (a) shows that this higher percentage appears above a Hamming distance of 200. For problem YN3, it appears in a lower percentage and is observed in Figure 11 (c), where the Hamming distance is above 250. For the other two problems, YN2 and YN4, Hamming distances above 250 appear. For the four problems, it is observed that as Hamming distance decreases, there is a point



(a) HGACC landscape for the problem YN1-20x20. Average of 30 tests.

(b) HGACC landscape for the problem YN2-20x20. Average of 30 tests.

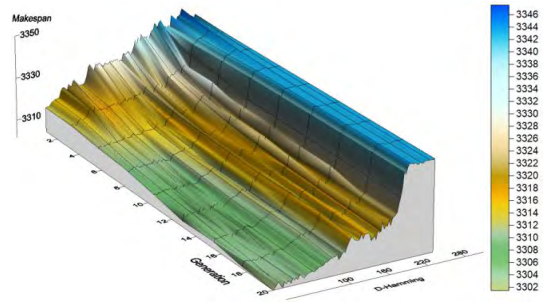
(c) HGACC landscape for the problem YN3-20x20. Average of 30 tests.

(d) HGACC landscape for the problem YN4-20x20. Average of 30 tests.

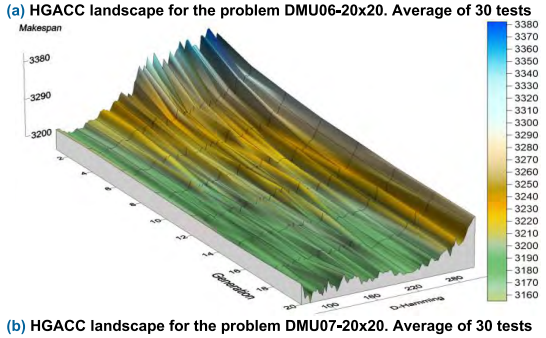
FIGURE 11. HGACC landscapes for the problems YN (1-4)-20x20.

at which the fitness of a small group of individuals improves considerably as the number of generations increases. For YN1, this improvement is close to a Hamming distance of 160; For YN2, this improvement is close to a Hamming distance of 240; For YN3, this improvement is close to a Hamming distance of 230 and for YN4, this improvement is close to a Hamming distance of 240. For the four problems, it is observed that after further decreasing Hamming distance, the fitness of the individuals tends to be poorer again. With these results and according to the presented landscape, it is possible to understand that the diversity in the solutions is not directly or inversely proportional to the effectiveness in the fitness for this type of problems (YN1 to YN2), but that there is an interval in the That there is a good diversity among individuals and in which the fitness of the individuals is of very high quality, but also there are intervals with a greater or very little diversity in which the quality of fitness is very poor or of regular quality among individuals. Diversity among individuals is evaluated by each micro-population independently. It is also evaluated by each pair of individuals that make up a micro-population, that is, each individual vs. the best individual in the micro-population.

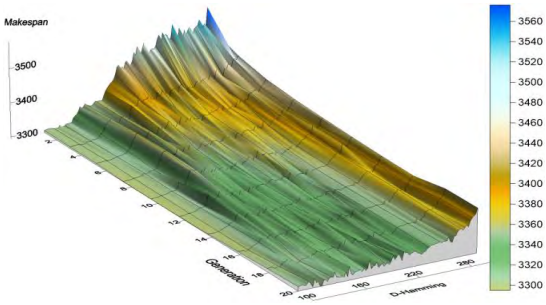
Figure 12 presents the landscape for problems DMU06, DMU07, DMU08, DMU09 and DMU10, when all genetic operators are applied (*tournament*, *elite migration*, *crossover-directed*, *iterative mutation*). The values of the HGACC parameters are presented in Table 1. The graphs are presented according to three important parameters of the HGACC, the diversity between the individuals (D-Hamming), the convergence of the algorithm (number of generations) and the fitness of individuals (makespan). The analysis of these graphs is with respect to the total 240 individuals that exist in the 48 micro-populations. It is observed in the four graphs, that during the first four generations, the fitness among individuals has the poorest value, but improves as the number of generations increases and the diversity of individuals decreases. There is also a group of very diverse individuals who always maintain a poor aptitude value as the generations pass. In Figure 12 (a) it can be seen that for the DMU06 problem, this higher percentage appears above a Hamming distance of 180. For the DMU07 problem, this phenomenon appears in a lower percentage, and can be observed in Figure 12 (d) to have a Hamming distance above 290. For the other three problems (DMU07, DMU08, and DMU10), the group of individuals with very poor aptitude values appears with a Hamming distance above 280. For all problems, it is observed that as the Hamming distance decreases and the number of generations increases, fitness, on average, tends to improve considerably. However, when studying all five problems, it is evident that not all of them achieve the best fitness quality when the diversity between individuals is close to zero and the number of generations reaches 20. For example, for problems DMU07 (Figure 12b) and DMU10 (Figure 12e), the best fitness in the 20th generation is obtained with individuals who have a Hamming distance close to 70 for both problems. Similarly, for problem DMU10, the Hamming distance is about 120. For DMU08 (Figure 12c), the best fitness in generation 20 is obtained with individuals that have a Hamming distance of about 130. For DMU06 (Figure 12a), the best fitness in generation 20 is obtained with individuals who have a Hamming distance of about 50.The case for problem DMU09 is similar, yielding a Hamming distance of about 60 (Figure 12d). The



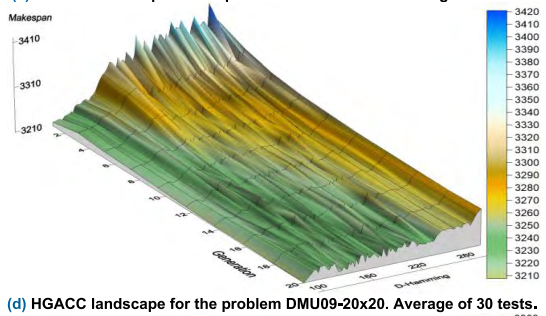
(a) HGACC landscape for the problem DMU06-20x20. Average of 30 tests



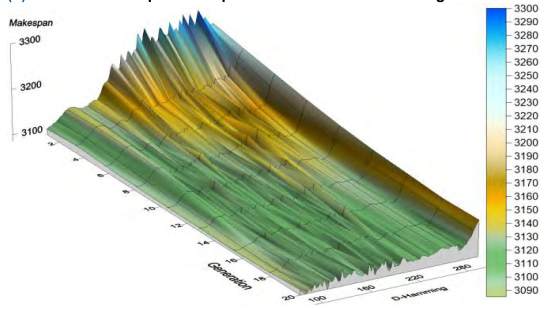
(b) HGACC landscape for the problem DMU07-20x20. Average of 30 tests



(c) HGACC landscape for the problem DMU08-20x20. Average of 30 tests.



(d) HGACC landscape for the problem DMU09-20x20. Average of 30 tests.



(e) HGACC landscape for the problem DMU10-20x20. Average of 30 tests.

FIGURE 12. HGACC landscapes for the problems DMU (06-10)-20x20.

best fitness in generation 20 is obtained with individuals who have a Hamming distance of about 90. With these results, and according to the landscape presented, it can be understood that on average, the diversity in the solutions is inversely proportional to the effectiveness in fitness for this type of problem (DMU06 to DMU10). Diversity among individuals is evaluated by each micro-population independently. It is also evaluated by each pair of individuals that make up a micro-population, that is, each individual vs. the best individual in the micro-population.

Table 2 presents the makespan results obtained with HGACC for small problems. It is noteworthy that the optimum value is obtained for all problems. This means the relative error is RE = 0. For all problems, a standard deviation of zero is obtained. Table 2 presents the best runtime for obtaining the optimal solution. The best time for the FT10 problem was about 6 seconds. The best times obtained for the LA problems did not exceed 0.21 seconds. The best times obtained for the ABZ problems did not exceed 12 seconds. In the ORB problems, the least complicated problem was ORB07, which reached optimal solution in less than 0.08 seconds. The most complicated of these problems was ORB06, for which the optimal solution was obtained in less than 16 seconds. The result of makespan that appears most often in the 30 tests is presented as mode and is shown in Table 2. For all problems, the mode is the optimal solution. The result of makespan presented as the median in the 30 executions is also the optimal solution. For all problems, the median is the optimal solution. When the median is the optimal solution, it can be understood that at least half of the 30 tests obtained the optimal solution. The problem that requires the greatest number of generations to find optimal value is ORB05, which requires 2 generations. Seventeen problems, out of eighteen, require only one generation to reach the optimum value.

Table 3 presents the results of MS obtained with HGACC for medium problems. It is noted that for almost all problems, the optimum value is obtained, resulting in a relative error RE = 0. In most problems a standard deviation of lower than 1 is obtained. Table 3 presents the best runtime for obtaining the optimal solution. The shortest time for LA problems was LA38 which took 115.9 seconds and one generation. LA40 took about 48 minutes and three generations, which was the longest time. The shortest time for the TA problems was the problem TA04 which took 61.6 seconds and one generation. The longest time was the TA06 with a time of about 71 minutes and fifteen generations. Only in problem TA07 was the optimal value not found and the RE=0.08.

The result of makespan that appears most often in the 30 tests is presented as the mode and is shown in Table 3. Twelve of fifteen medium size problems show the mode as the optimal value. In problems where the mode is not the optimal value, the greatest difference (mode-optimal) is 2, as seen in problem TA06. The smallest difference is 1, as seen in problem TA07. The result of makespan is presented as the median in 30 executions. For most problems, the median is the optimal solution. When the median is the optimal solution, it can be understood that at least half of the 30 tests obtained the optimal solution. The problem that requires the greatest number of generations to find the optimal value is the

TABLE 6. Efficiency and efficacy for Applegate-Cook, Adams, Lawrence, Taillard, Yamada-Nakano, and Demirkol et al. benchmarks.

Problem	Op/UB	%RE																			
		HG	tsec	B-G	tsec	A-M	tsec	TSSA	tsec	EPPX	tsec	TGA	tsec	IEBO	tsec	TS/PR	tsec	AGS	tsec	UP	tsec	AGA
Size 10 x 10																						
ORB01	**1059**	0	0.23	0	5.8	0	56.6	0	3.5	1.70	1.02	0	0.06	--	--	0	0.51	0	342	0	2312	3.12
ORB02	**888**	0	0.28	0	5.8	0	569	0	6.4	0.11	0.92	0	0.06	--	--	0	1.69	0.1	306	0.11	2393	0.68
ORB03	**1005**	0	1.84	0	5.8	0	404	0	13.8	1.69	1.07	0	0.15	--	--	0	1.46	1.2	330	0	2358	2.39
ORB04	**1005**	0	0.1	0	5.8	0	17.9	0	14.3	0	0.93	0	0.45	--	--	0	3.71	0	306	0	796	1.09
ORB05	**887**	0	1.76	0	5.8	0	670	0	6.6	0.34	0.98	0	0.76	--	--	0	7.28	0	366	0.23	2458	1.58
ORB06	**1010**	0	15.7	0	5.8	--	--	0	8.5	1.09	1.11	0	0.72	--	--	0	1.81	--	--	0.30	2525	1.78
ORB07	**397**	0	0.08	0	5.8	--	--	0	0.5	0	0.90	0	0.02	--	--	0	0.13	--	--	0	2096	2.02
ORB08	**899**	0	0.38	0	5.8	--	--	0	7.2	0	1.01	0	0.09	--	--	0	3.99	--	--	0	2338	1.67
ORB09	**934**	0	10.2	0	5.8	--	--	0	0.4	0	0.92	0	0.09	--	--	0	0.47	--	--	0	884	0.96
ORB10	**944**	0	7.5	0	5.8	--	--	0	0.3	0	0.98	0	0.03	--	--	0	0.09	--	--	0	817	--
ABZ5	**1234**	0	11	--	--	0	502	--	--	0	0.84	0	0.04	--	--	--	--	--	--	--	--	--
ABZ6	**943**	0	0.23	--	--	0	199	--	--	0	0.79	0	0.03	--	--	--	--	--	--	--	--	--
Size 15 x 15																						
TA01	**1231**	0	234	0	30.4	0	1531	0	11.2	--	--	--	--	0	124	0	2.93	3.1	2782	--	--	--
TA02	**1244**	0	113	0	30.4	0	685	0	30.1	--	--	--	--	0	118	0	38	--	--	--	--	--
TA03	**1218**	0	2353	0	30.4	0.16	1834	0	109	--	--	--	--	0	120	0	44	--	--	--	--	--
TA04	**1175**	0	62	0	30.4	0	1186	0	71.7	--	--	--	--	0	117	0	39	--	--	--	--	--
TA05	**1224**	0	4065	0	30.4	0.33	1493	0	10.8	--	--	--	--	0	120	0	11	--	--	--	--	--
TA06	**1238**	0	4216	0	30.4	0	1549	0	125	--	--	--	--	0	113	0	178	--	--	--	--	--
TA07	**1227**	0.08	59	0.08	30.4	0.08	1687	0.08	139	--	--	--	--	0	117	0.08	0.60	--	--	--	--	--
TA08	**1217**	0	62	0	30.4	0	968	0	27.6	--	--	--	--	0	108	0	2.43	--	--	--	--	--
TA09	**1274**	0	447	0	30.4	0	1694	0	61.3	--	--	--	--	0	127	0	19	--	--	--	--	--
TA10	**1241**	0	66	0	30.4	0	1418	0	68	--	--	--	--	0	122	0	42	--	--	--	--	--
Size 20 x 20																						
DMU06	**3244**	0.74	29868	0	145.4	--	--	--	--	--	--	--	--	--	--	0.03	823	--	--	--	---	--
DMU07	**3046**	0.59	83791	0	145.4	--	--	--	--	--	--	--	--	--	--	0	361	--	--	--	---	--
DMU08	**3188**	0	1836	0	145.4	--	--	--	--	--	--	--	--	--	--	0	296	--	--	--	---	--
DMU09	**3092**	0.55	2379	0	145.4	--	--	--	--	--	--	--	--	--	--	0.07	148	--	--	--	---	--
DMU10	**2984**	0.10	2813	0	145.4	--	--	--	--	--	--	--	--	--	--	0.03	253	--	--	--	---	--
DMU46	**4035**	1.59	9973	0	187.7	--	--	--	--	--	--	--	--	--	--	0	985	--	--	--	---	--
YN1	**884**	0.23	13794	0	105.2	--	--	0	107	3.05	12.19	0.23	92.8	0	190	0	169	0.2	15786	--	---	--
YN2	**904**	0.33	6502	0	105.2	--	--	0.33	110	4.09	11.72	0.77	13.1	0	197	0	202	4.4	14586	--	---	--
YN3	**892**	0	4548	0	105.2	--	--	0	111	4.04	14.09	0.56	37.2	0	212	0	344	1.3	16662	--	---	--
YN4	**967**	0.21	46328	0	105.2	--	--	0.21	109	4.55	10.52	0.83	114.1	0	---	0.10	321	2.17	14752	--	---	--

EPPX, 2.4GHz, Intel Xeon, [5]
AntGenSA (AGS). Intel® Xeon® 2.3 GHz, 64Gb, [15]
BRK-GA (B-G). 2.2GHz AMD Opteron, [30]
ACOFT-MWR (A-M) 1.533GHz CPU, [32]
TSSA. Pentium IV 3.0GHz, [33]
TGA. PC 2.2 GHz, 8GHz, [37]
IEBO, 2.93 GHz,Intel Xeon X5670, [38]
TS/PR. PC AMD Athlon 3 GHz, 2 Gb,[39]
UPLA (UP), Intel CoreTM i5, 2.67 GHz, 6GB, [40]
ALSGA (AGA). Intel core duo, 2.93 GHz, 2.0GB, [41]
HGACC(HG). CLUSTER, 48cores, Xeon 3.06GHz, this work

TA06 problem with 15 generations. Seven problems require only one generation to reach the optimum value.

Table 4 presents the results of makespan obtained with HGACC for big problems. It is observed that most of the obtained values were very close to the known bound, in fact for two of the problems the obtained value equaled the known bound. The greatest relative error found was RE = 1.59% for the problem DMU46 in generation 4. The lowest relative error was RE = 0 for the problems DMU08 and YN3 in generations 5 and 8 respectively. The DMU46 problem was the largest relative error found, RE = 1.59 in generation 8. The largest standard deviation was 12.62 for the problem DMU07, and the least standard deviation was 0 for the problem YN2. Table 4 presents the best execution time for the best solution found. The shortest time for YN problems in which the UB was obtained was for the YN3 problem with a time of about 76 minutes in generation 6. The longest time was for the YN4 problem with a time of about 772 minutes in generation 49. The shortest time necessary to obtain the UB was about 31 minutes, for the DMU08 problem in generation 5. The longest time, for the DMU07 problem, was about 23 hours in generation number 14. The result of makespan that appears most often in the 30 tests is presented as the mode.

TABLE 7. Parallel algorithms, efficacy for Fisher and Thompson, Applegate-Cook, Lawrence, Taillard, Yamada-Nakano, and Demirkol et al., benchmarks.

Problem	Op/UB	%RE									
		HG	PPSO	cGA-PR	PaGA	HGS	HIMGA	NIMGA	IIMMA	AGS	PABC
Size 10 x 10											
FT10	**930**	0	--	0	7.2	--	0	0	0	0.9	0
LA16	**945**	0	29	--	5.2	--	0	0.11	0	0	0
LA17	**784**	0	33	--	1.2	--	0	0	0	--	0
LA18	**848**	0	33	--	1.4	--	0	0	0	--	0
LA19	**842**	0	37	--	3.7	--	0	0	0	--	0
LA20	**902**	0	32	--	1.1	--	0	0.55	0	--	0.55
ORB01	**1059**	0	--	0	8.5	--	0	0	0	0	--
ORB02	**888**	0	--	--	4.6	--	0	0.23	0	0.1	--
ORB03	**1005**	0	--	0	12.3	--	0	2.09	0	0	--
ORB04	**1005**	0	--	0	5.7	--	0	1.39	0	0	--
ORB05	**887**	0	--	--	5.5	--	0	0.68	0	0	--
ORB06	**1010**	0	--	--	4.0	--	0	0.20	0	--	--
ORB07	**397**	0	--	--	4.8	--	0	0	0	--	--
ORB08	**899**	0	--	0	12.4	--	0	1.11	0	--	--
ORB09	**934**	0	--	--	6.4	--	0	0.86	0	--	--
ORB10	**944**	0	--	0	--	--	0	--	0	--	--
Size 15 x 15											
LA36	**1268**	0	65	--	--	0.87	0	1.97	0	--	--
LA37	**1397**	0	58	--	--	0.79	0	3.01	0	--	--
LA38	**1196**	0	68	1.0	--	1.92	0	2.17	0	--	--
LA39	**1233**	0	67	--	--	1.05	0	2.11	0	--	--
LA40	**1222**	0	68	1.31	--	1.56	0.16	1.96	0.16	--	--
Size 20 x 20											
DMU06	**3244**	0.74	--	0.52	--	--	--	--	--	--	--
DMU07	**3046**	0.59	--	1.15	--	--	--	--	--	--	--
DMU08	**3188**	0	--	0.53	--	--	--	--	--	--	--
DMU09	**3092**	0.59	--	0.13	--	--	--	--	--	--	--
DMU10	**2984**	0.10	--	0.84	--	--	--	--	--	---	--
YN1	**884**	0.23	--	2.49	--	--	1.01	--	0.22	1.4	--
YN2	**904**	0.33	--	1.77	--	--	0.99	--	0.55	1.2	--
YN3	**892**	0	--	1.01	--	--	0.90	--	0.34	0.9	--
YN4	**967**	0.21	--	1.55	--	--	1.03	--	0.31	1.76	--

PaGA. Computer network with JADE Middleware, [7]
cGA-PR. Workstation, multicore, 2.0GHz, Visual C++, [9]
HIMGA. PC, 3.4GHz, Intel®, Core(TM), i7-3770 CPU, C++, [11]
NIMGA. PC, 3.4GHz, Intel®, Core(TM), i7-3770 CPU, C++, [12]
PABC. Four computers system configuration, JAVA, [13]
IIMMA, PC, 3.4 GHz, Intel®, Core(TM), i7-3770 CPU, C++, [14]
AntGenSA (AGS). Cluster 4nodes, Intel® Xeon® 2.3 GHz, 64Gb, C, OpenMP, [15]
PPSO. Server and client machines, Logical ring topology, [35]
HGAPSA (HGS). Server and client machines, [36]
HGACC (HG). CLUSTER, 48cores, Xeon 3.06GHz, gcc, MPI Library, this work

In Table 4, it can be seen that none of the large size problems show mode with the best UB. In DMU problems, the largest difference between the mode and the UB (mode-UB) is 68, for DMU46. The smallest difference is 11, for DMU08. It can be noted that the highest relative error obtained for the large problems DMU (06-10) was RE = 0.74 and the smallest was RE = 0. This indicates that the efficacy of HGACC is competitive because RE < 1 for the known UB. For the DMU46 problem, the efficacy is not very good because 1.59 <= RE <= 3.82. The relative error obtained with mode for the YN problems shows that the RE is in the range of 0 <= RE <= 2.57. This indicates that the MS found in 30 tests and repeated in each YN problem, has a RE in this interval with respect to UB. The relative error obtained for mode in the DMU (06-10) problems shows that the RE is in the range of 0<= RE <= 4.2. This indicates that the MS found in 30 tests, and repeated in each DMU (06-10) problem, has a RE in this interval with respect to the UB. The result of makespan presented with the median obtained from the 30 tests, shows that in none of the problems was the

median the UB. The problems that caused most difficulty in obtaining good results with respect to the mode and median were problems DMU06, DMU09 and DMU46.

Table 5 presents the results for the Fisher, Thompson, and Lawrence benchmarks. The results of the proposed HGACC algorithm show that the best results reported in the literature are obtained with a BRK-GA (B-G) algorithm. HGACC, together with BRK-GA (B-G) and TS/PR, obtained the optimum for LA40. With regard to the execution time, HGACC is competitive for most benchmarks. For problems LA36 to LA40, the time is greater with respect to the other algorithms. The exception is with the genetic SAGen, which has greater execution times and does not reach the optimum value in all LA problems. ALSGA (AGA) algorithm does not report execution time. In the case of HGACC, the optimum is reached for all problems. Table 5 shows that HGACC has better performance in efficacy than most of the population-based algorithms and it competes with the BRK-GA (B-G) population algorithm. It is also observed that HGACC is competitive with algorithms that are not population-based.

Table 6 presents results for the Applegate-Cook, Adams, Lawrence, Taillard, Yamada-Nakano and Demirkol et al., benchmarks. The results for the proposed HGACC show that it obtains the best results reported in the literature for most of the benchmarks. The failure to achieve the known results are at no greater than 0.33% RE in the YN problems, 0.74% RE in the DMU06 to DMU10 problems, and 1.59% RE in the DMU46 problem. These results show that HGACC is very competitive with respect to the other algorithms presented. In terms of time, HGACC is competitive with the ACOFT-MWR (A-M) algorithm. The exception can be seen in the results obtained by HGACC, with respect to the BRK-GA (B-G) algorithm for the DMU problems. In HGACC, $0 < RE < 1.59$ and in BRK-GA (B-G), the optimum is reached in all the DMU problems. Table 6 shows that HGACC achieves better performance in efficacy than most of the presented population-based algorithms and it competes with the BRK-GA (B-G) population-based algorithm. IEBO algorithm obtains all the Op/LB of the benchmarks used, but it is not a population-based algorithm. The TS/PR algorithm has also better performance than HGACC, but it is not a population-based algorithm neither. HGACC achieves better performance in efficacy than ACOFT-MWR (A-M), EPPX, and UPLA (UP), these are not population-based algorithms.

Table 7 presents a comparison of the relative error of algorithms with the threads/processes (parallel/distributed) vs. HGACC for the Fisher and Thompson, Applegate-Cook, Lawrence, Taillard, Yamada-Nakano and Demirkol et al. benchmarks. It is difficult to compare these algorithms because very little recent information was found in literature for parallel/distributed algorithms, and the results presented did not address the variety of problems that HGACC does. Nonetheless, in all results for the proposed HGACC, it can be noted that HGACC has better performance with respect to the other algorithms in the literature that use the threads/processes which are presented in this paper.

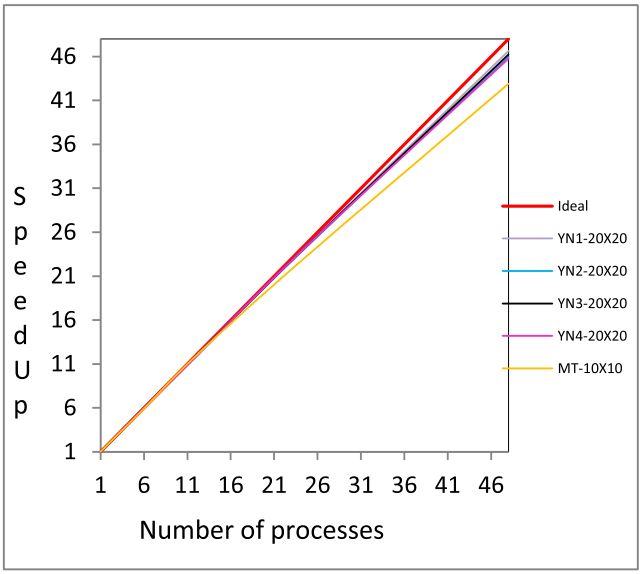


FIGURE 13. Average Speedup for the HGACC algorithm for MT10, YN1, YN2, YN3, and YN4 benchmarks.

The exceptions were the DMU06 and DMU09 problems. The relative error that HGACC presents is in the range of 0 to 0.33% without DMU problems and 0 to 0.75 in DMU problems. The relative error that PPSO presents is in the range of 29 to 68% without DMU problems. The relative error of cGA-PR is in the range of 0 to 2.49% without DMU problems and 0.13 to 1.15 in DMU problems. The relative error of PaGA is in the range of 0 to 12.4% without DMU problems. HGAPSA has a relative error in the range of 0.79 to 1.92% without DMU problems. HIMGA has a relative error in the range of 0 to 1.01% % without DMU problems. NIMGA has a relative error in the range of 0 to 3.0 % without DMU problems. IIMMA has a relative error in the range of 0 to 0.55% without DMU problems. Table 7 shows that HGACC reports better performance than the population-based parallel algorithms.

Figure 13 shows that the speedup is influenced by the communication between nodes of the computational cluster (data transfer via InfiniBand networks). An increased number of processes lead to greater data transfer because each running process needs to transfer data to the master node (see Figure 2). Therefore, it can be seen that the speedup moves away from the ideal as the number of processes increases. It is also noted that the speedup for the YN4 problem is farther from the ideal when compared to other YN problems. This may be because the YN4 problem takes more time to find feasible individuals when performing the iterative mutation. This causes the run time to increase with greater numbers of processes (micro-populations). If this is true, then YN1, YN2 and YN3 may find feasible solutions more easily in iterative mutation since they have better speedup when the number of processes increases. Regardless, it is important to recognize that the behavior of the speedup is very good for all YN problems since the speedup is close to the ideal in the evaluation interval for the four instances of large size.

For a small size problem such as the FT10 problem, it can be observed that the speedup is of lower quality when compared to the YN problems as the number of running processes increases.

Figure 13 shows that for the F10 problem, data transfer between the slave nodes and the master node becomes more inefficient with 24 processes. This is because each node in the computational cluster has 12 processing cores (each core executes one HGACC process). When transferring between two nodes (master-slave), the data sending in the HGACC becomes slower than when sending between the processes belonging to the master node (maximum 12 processes, one per core). This also happens for the YN problems. The difference is that in the FT10 problem, small datasets are sent. This is because the genetic representations of each individual contain 10 chromosomes, and each chromosome has 21 genes, for a total of 210 genes per individual. For YN problems, there are 820 genes per individual. Figure 4 illustrates the genetic representation of an individual. Based on the size of information sent for small problems and large problems, it can be understood that it is more convenient to send large datasets between master-slave nodes, because the FT10 speedup is less efficient than the speedup for large problems. There is a limit for data sending between nodes of the computational cluster. This limit is the bandwidth of 40 GB/s, which the nodes have as a maximum for data transfer.

V. CONCLUSIONS

There is a balance between the *crossover-directed* operators and elite migration, which allows for convergence control in the algorithm. According to the tuning of these parameters, a convergence behavior was obtained, which was sufficient, as shown by the high quality results of the algorithm for the benchmarks used in this work.

The influence of the *crossover-directed* operator generally allows, on average, for all individuals in each micro-population to present higher quality fitness than when the operator is not applied. It is also possible to conclude that the exploration of the solution space improves when a greater number of micro-populations are given. This is due to the diversity that exists in micro-populations. Such increase in quality of the exploration occurs when there are smaller micro-populations and a greater number of these.

Finding a quality aptitude for individuals does not always depend on the diversity of the micro-population. It also depends on the problem to be solved, as was observed in the landscape analysis of the YN and DMU problems. In general, there is a tendency to find good aptitude with lower diversity among individuals to YN and DMU problems.

The results obtained with HGACC are of excellent quality as compared with the sequential algorithms proposed in the literature. HGACC reports better performance in efficacy than most of the population-based algorithms compared in this work, and it is competitive with the BRK-GA (B-G) population algorithm. IEBO algorithm obtains all the Op/UB of the benchmarks used, but it is not a population-based algorithm. The TS/PR algorithm shows also better performance than HGACC but it is not a population-based algorithm either. It was also observed that HGACC is competitive and some times better with non-population-based algorithms presented in this work. With respect to the parallel/distributed algorithms, HGACC presents higher quality results in the majority of the problems evaluated in this work.

According to the speedup results obtained for large and small problems, it can be concluded that in order to have good efficiency in speedup, it is advisable to send large packets of information when using collective communication. This is dependent on having good bandwidth and transfer speed in the computational cluster.

The main contribution of this work is the design of a genetic procedure with micro-populations, which applies in a balanced way a crossover that leads to elite solutions and changes in the genetic sequence employing an iterative procedure. This contribution creates a great performance to explore and exploit the space of solutions through micro-population that maintain genetic diversity. It should be noted that applying genetic diversity to improve the performance of the HGACC algorithm depends on the type of problem to be solved.

REFERENCES

[1] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman Company, 1990, p. 340.

[2] H. Zhou, Y. Feng, and L. Han, "The hybrid heuristic genetic algorithm for job shop scheduling," *Comput. Ind. Eng.*, vol. 40, no. 3, pp. 191–200, Jul. 2001. doi: 10.1016/S0360-8352(01)00017-1.

[3] J. F. Gonçalves, J. J. M. Mendes, and M. G. C. Resende, "A hybrid genetic algorithm for the job shop scheduling problem," *Eur. J. Oper. Res.*, vol. 167, no. 1, pp. 77–95, Nov. 2005. doi: 10.1016/j.ejor.2004.03.012.

[4] R. Qing-dao-er-ji and Y. Wang, "A new hybrid genetic algorithm for job shop scheduling problem," *Comput. Oper. Res.*, vol. 39, no. 10, pp. 2291–2299, Oct. 2012. doi: 10.1016/j.cor.2011.12.005.

[5] N. H. Moin, O. C. Sin, and M. Omar, "Hybrid genetic algorithm with multiparents crossover for job shop scheduling problems," *Math. Problems Eng.*, vol. 2015, Sep. 2014, Art. no. 210680. doi: 10.1155/2015/210680.

[6] J. Gu, X. Gu, andM. Gu, "A novel parallel quantum genetic algorithm for stochastic job shop scheduling," *J. Math. Anal. Appl.*, vol. 355, no. 1, pp. 63–81, Jul. 2009. doi: 10.1016/j.jmaa.2008.12.065.

[7] L. Asadzadeh and K. Zamanifar, "An agent-based parallel approach for the job shop scheduling problem with genetic algorithms," *Math. Comput. Model.*, vol. 52, nos. 11–12, pp. 1957–1965, Dec. 2010. doi: 10.1016/j.mcm.2010.04.019.

[8] R. Yusof, M. Khalid, G. T. Hui, S. M. Yusof, and M. F. Othman, "Solving job shop scheduling problem using a hybrid parallel micro genetic algorithm," *Appl. Soft Comput.*, vol. 11, no. 8, pp. 5782–5792, Dec. 2011. doi: 10.1016/j.asoc.2011.01.046.

[9] A. C. Spanos, S. T. Ponis, I. P. Tatsiopoulos, I. T. Christou, and E. Rokou, "A new hybrid parallel genetic algorithm for the job-shop scheduling problem," *Int. Trans. Oper. Res.*, vol. 21, no. 3, pp. 479–499, May 2014. doi: 10.1111/itor.12056.

[10] L. Asadzadeh, "Solving the job shop scheduling problem with a parallel and agent-based local search genetic algorithm," *J. Theor. Appl. Inf. Technol.*, vol. 62, no. 2, pp. 317–324, Apr. 2014.

[11] M. Kurdi, "A new Hybrid island model genetic algorithm for job shop scheduling problem," *Comput. Ind. Eng.*, vol. 88, pp. 273–283, Oct. 2015. doi: 10.1016/j.cie.2015.07.015.

[12] M. Kurdi, "An effective new island model genetic algorithm for job shop scheduling problem," *Comput. Oper. Res.*, vol. 67, pp. 132–142, Mar. 2016. doi: doi.org/10.1016/j.cor.2015.10.005.

[13] L. Asadzadeh, "A parallel artificial bee colony algorithm for the job shop scheduling problem with a dynamic migration strategy," *Comput. Ind. Eng.*, vol. 102, pp. 359–367, Dec. 2016. doi: 10.1016/j.cie.2016.06.025.

[14] M. Kurdi, "An improved island model memetic algorithm with a new cooperation phase for multi-objective job shop scheduling problem," *Comput. Ind. Eng.*, vol. 111, pp. 183–201, Jul. 2017. doi: 10.1016/j.cie.2017.07.021.

[15] L. Hernández-Ramírez, J. Frausto-Solis, G. Castilla-Valdez, J. J. González-Barbosa, D. Terán-Villanueva, and L. M. Morales-Rodríguez , "A hybrid simulated annealing for job shop scheduling problem," *Int. J. Combinat. Optim. Problems Informat.*, vol. 10, no. 1, pp. 6–15, Jan. 2019.

[16] R. Nakano and T. Yamada, "Conventional genetic algorithm for job-shop problems," in *Proc. ICGA*, San Diego, CA, USA, 1991, pp. 474–479.

[17] R. Cheng, M. Gen, and Y. Tsujimura, "A tutorial survey of job-shop scheduling problems using genetic algorithms—I. representation," *Comput. Ind. Eng.*, vol. 30, no. 4, pp. 983–997, Sep. 1996. doi: 10.1016/0360-8352(96)00047-2.

[18] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Boston, MA, USA: Addison-Wesley, 1989, p. 432.

[19] P. J. M. Van Laarhoven, E. H. L. Aarts, and J. K. Lenstra, "Job shop scheduling by simulated annealing," *Oper. Res.*, vol. 40, no. 1, pp. 113–125 Feb. 1992.

[20] C. Zhang, P. Li, Z. Guan, and Y. Rao, "A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem," *Comput. Oper. Res.*, vol. 34, no. 11, pp. 3229–3242, Nov. 2007. doi: 10.1016/j.cor.2005.12.002.

[21] J.-P. Watson, J. C. Beck, A. E. Howe, and L. D. Whitley, "Problem difficulty for tabu search in job-shop scheduling," *Artif. Intell.*, vol. 143, no. 2, pp. 189–217, Feb. 2003. doi: 10.1016/S0004-3702(02)00363-6.

[22] J. E. Beasley, "OR-Library: Distributing test problems by electronic mail," *J. Oper. Res. Soc.*, vol. 41, no. 11, pp. 1069–1072, Nov. 1990. doi: 10.1057/jors.1990.166.

[23] H. Fisher and L. G. Thompson, "Probabilistic learning combinations of local job-shop scheduling rules," in *Industrial Scheduling*, J. F. Muth, and G. L. Thompson, Eds. Englewood Cliffs, New Jersey, USA: Prentice-Hall, 1963, pp. 225–251.

[24] D. Applegate and W. Cook, "A computational study of the job-shop scheduling instance," *ORSA J. Comput.*, vol. 3, no. 2, pp. 149–156, 1991. doi: 10.1287/ijoc.3.2.149.

[25] J. Adams, E. Balas, and D. Zawack, "The shifting bottleneck procedure for job shop scheduling," *Manage. Sci.*, vol. 34, pp. 391–401, Mar. 1988.

[26] S. Lawrence, "Resouce constrained project scheduling : An experimental investigation of heuristic scheduling techniques (Supplement)," Graduate School Ind. Admin., Carnegie-Mellon Univ., Pittsburgh, Pennsylvania, Tech. Rep., 1984.

[27] E. Taillard, "Benchmarks for basic scheduling problems," *Eur. J. Oper. Res.*, vol. 64, no. 2, pp. 278–285, Jan. 1993. doi: 10.1016/0377-2217(93)90182-M.

[28] T. Yamada and R. Nakano, "A genetic algorithm applicable to large-scale job-shop instances," in *Proc. PISFN2*, Amsterdam, Holland, 1992, pp. 281–290.

[29] E. Demirkol, S. Mehta, and R. Uzsoy, "A computational study of shifting bottleneck procedures for shop scheduling problems," *J. Heuristics*, vol. 3, no. 2, pp. 111–137, Nov. 1997. doi: 10.1023/A:1009627429878.

[30] J. F. GonÃğalves and M. G. C. Resende, "An extended Akers graphical method with a biased random-key genetic algorithm for job-shop scheduling," *Int. Trans. Oper. Res.*, vol. 21, no. 2, pp. 215–246, Mar. 2014. doi: 10.1111/itor.12044.

[31] M. Kolonko, "Some new results on simulated annealing applied to the job shop scheduling problem," *Eur. J. Oper. Res.*, vol. 113, no. 1, pp. 123–136, Feb. 1999. doi: 10.1016/S0377-2217(97)00420-7

[32] K.-L. Huang and C.-J. Liao, "Ant colony optimization combined with taboo search for the job shop scheduling problem," *Comput. Oper. Res.*, vol. 35, no. 4, pp. 1030–1046, Apr. 2008. doi: 10.1016/j.cor.2006.07.003

[33] C. Y. Zhang, P. Li, Y. Rao, and Z. Guan, "A very fast TS/SA algorithm for the job shop scheduling problem," *Comput. Oper. Res.*, vol. 35, no. 1, pp. 282–294, Jan. 2008. doi: 10.1016/j.cor.2006.02.024.

[34] D. Y. Sha and C.-Y. Hsu, "A hybrid particle swarm optimization for job shop scheduling problem," *Comput. Ind. Eng.*, vol. 51, no. 4, pp. 791–808, Dec. 2006. doi: 10.1016/j.cie.2006.09.002.

[35] A. Aitzai and M. Boudhar, "Parallel branch-and-bound and parallel PSO algorithms for job shop scheduling problem with blocking," *Int. J. Oper. Res.*, vol. 16, no. 1, pp. 14–37, 2013. doi: 10.1504/IJOR.2013.050538.

[36] T. Rakkiannan and B. Palanisamy, "Hybridization of genetic algorithm with parallel implementation of simulated annealing for job shop scheduling," *Amer. J. Appl. Sci.*, vol. 9, no. 10, pp. 1694–1705, Jan. 2012.

[37] M. Amirghasemi,and R. Zamani, "An effective asexual genetic algorithm for solving the job shop scheduling problem," *Comput. Ind. Eng.*, vol. 83, pp. 123–138, May 2015. doi: doi.org/10.1016/j.cie.2015.02.011.

[38] Y. Nagata,and I. Ono, "A guided local search with iterative ejections of bottleneck operations for the job shop scheduling problem," *Comput. Oper. Res.*, vol. 90, pp. 60–71, Feb. 2018. doi: doi.org/10.1016/j.cor.2017.09.017.

[39] B. Peng, Z. Lü, and T. C. E. Cheng, "A tabu search/path relinking algorithm to solve the job shop scheduling problem," *Comput. Oper. Res.*, vol. 53, pp. 154–164, Jan. 2015. doi: doi.org/10.1016/j.cor.2014.08.006.

[40] P. Pongchairerks, "A two-level metaheuristic algorithm for the job-shop scheduling problem," *Complexity*, vol. 2019, Mar. 2019, Art. no. 8683472. doi: doi.org/10.1155/2019/8683472.

[41] L. Asadzadeh, "A local search genetic algorithm for the job shop scheduling problem with intelligent agents," *Comput. Ind. Eng.*, vol. 85, pp. 376–383, Jul. 2015.

MARCO ANTONIO CRUZ CHÁVEZ received the Ph.D. degree in computer sciences from the Tec de Monterrey, in 2004. Since 2004, he has been a Research Professsor with the Research Center in Engineering and Applied Sciences (CIICAP), Autonomous University of Morelos State (UAEM). He is a National Researcher of Mexico (SNI). He is also a Leader of Grid Morelos Project, High Performance of Computing Laboratory. He is the Manager of High Performance Grid Morelos. He is also the Leader of Research Group Optimization and Software. He has 28 international publications and 30 national publications. Since 2005, he has been a Reviewer of international journals.

MARTÍN H. CRUZ ROSALES received the Ph.D. degree from the Autonomous University of Morelos State. He was Research Professor with the Simulation and Nuclear Energy Department, Electrical Research Institute, Morelos, Mexico. He then became a Researcher and a Lecturer with the Science Faculty, Autonomous University of Morelos Sate, Morelos. He is currently a Researcher and a Lecturer with the Faculty of Accounting, Administration and Computing, Autonomous University of Morelos State, Morelos. He has several publications and articles on scheduling algorithms, and he has also given several lectures and seminars. His research interest includes the area of combinatorial optimization.

JOSÉ CRISPÍN ZAVALA-DÍAZ received the Ph.D. degree in computer sciences from the Tec de Monterrey, in 2002. Since 2002, he has been a Research Professor with the Autonomous University of Morelos State. He is also a leader of a research group. He has several publications and articles on scheduling algorithms, and he has also given several lectures and seminars.

JOSÉ ALBERTO HERNÁNDEZ AGUILAR received the B.S. degree in computers engineering from the National Autonomous University of Mexico, the M.B.A. degree *(Cuma Sum Laude)* from the Universidad de las Americas (UDLA), A.C., in 2003, and the Ph.D. degree, in 2008. He has finished the Doctorate thesis with the Autonomous University of Morelos State, in 2007. Since 2010, he has been a full-time Professor with the School of Accounting, Management, and Computer Sciences, Autonomous University of the Morelos State. His areas of interest include artificial intelligence, optimization, and data science.

ABELARDO RODRÍGUEZ-LEÓN received the B.E. degree in computer systems engineering from the Veracruz Institute of Technology, in 1989, the M.S. degree in computer sciences from Universidad Veracruzana, in 1992, and the Ph.D. degree in computer science from the Universidad Politécnica de Valencia, Spain, in 2007. He is currently a Professor in computer science with the Computer and Systems Department, Veracruz Institute of Technology, Mexico. His research interests include grid computing, parallel programming, and optimization.

JUAN CARLOS PRINCE AVELINO received the B.Sc. degree in mechanical engineering from the Instituto Tecnologico de Vercruz, Mexico, in 1984, the M.Sc. degree in mechanical engineering from the Instituto Politecnico Nacional, Mexico, in 1987, and the Ph.D. degree from the University of Cambridge, U.K., in 1994. He has been a Professor of mechanical engineering with the Technological Institute of Veracruz, Mexico, since 1995. He has also been with the National Research of Mexico (SNI), since 1997.

MARTHA ELENA LUNA ORTIZ received the master's degree in information technology from the Technological Institute of Zacatepec, in 2007. She has been a Professor with the Emiliano Zapata Technological University of Morelos State, since 2007, attached to the Academic Division of Information Technology Networks and Telecommunications Area. She is a member of the State System of Researchers (SEI), Morelos. She is responsible for the academic body called Secure Converging Networks. She is currently the in charge of the Department of Research and Technological Development, promoting technological and innovation projects, among which is the GRID Morelos.

OSCAR H. SALINAS received the Ph.D. degree in engineering and energy area from the Universidad Nacional Autonoma de Mexico (UNAM), in 2008. He has been a Research Professor with the Emiliano Zapata Technological University of Morelos State. He has been in collaboration with the Autonomous University of the Morelos State, since 2003. He is a member of the Morelos State Researchers System (SEI), since 2009. He has received the State Saving Energy Award in 2015 and 2017, with two researching projects. He has published 23 papers in different national and international journals and also a chapter from one book. He is a Reviewer of three internationals journals. He has professional experience six years with Motorola de México.