Abstract—This paper proposes approaches for combining Iterated Greedy techniques, as state-of-the-art methods, with bacterial evolutionary algorithms based on a hybrid technique involving the Multi-Threaded Iterated Greedy heuristic and a memetic algorithm in order to efficiently solve the Permutation Flow Shop Problem on parallel computing architectures. In the present work three novel approaches are proposed by combining a variant of the Bacterial Memetic Algorithm and the recently proposed Bacterial Iterated Greedy technique with the mentioned hybrid multi-threaded approach.

The techniques thus obtained are evaluated via simulation runs carried out on a series of data from the well known Taillard’s benchmark problem set. Based on the experimental results the hybrid methods are compared to each other and to the original techniques (i.e. to the techniques without bacterial algorithms).

Index Terms—Bacterial methods, Memetic algorithms, Hybrid Iterated Greedy techniques, Combinatorial optimization, Permutation Flow Shop Problem, Parallel architectures, Multiple threads

I. INTRODUCTION

One of the most intensively studied combinatorial optimization problems is the Permutation Flow Shop Problem (PFSP) [1]. In this problem there are given \( n \) jobs and \( m \) machines. All the jobs should be processed by all the machines one after another. The machines are deployed in a line and a machine can handle one single job at once. That is, the processing of the jobs is pipeline-like. There is also given an \( n \times m \)-size processing time matrix defining the necessary amount of time a job has to stay on a machine, for each job-machine pair. A job can be processed on a machine only if the machine is free (the preceding job has finished on the machine) and the job has already processed on the preceding machine.

The task is to find a permutation (a sequence) of the jobs, in case of which the total processing time of all the jobs on all the machines (i.e. the so called makespan) is minimal.

This problem is known to be NP-hard [2], thus there are no efficient algorithms to solve this task (and there is not much hope to find one). It means that every method guaranteeing optimal solutions has impractically long computational time for even moderate problem sizes. Hence only heuristics resulting in so called quasi-optimal solutions are viable. In the past few decades a number of such heuristics are invented and published (e.g. [3], [4]).

Since due to the nature of the PFSP problem these heuristics cannot be evaluated analytically, their evaluation and their comparison to other techniques can be made experimentally, i.e. based on results of simulation runs carried out on standard reference tasks, called benchmark problems. Several such comparisons have been made involving a large part of the so far proposed methods (e.g. [3], [4]). These comparative studies are mostly based on the well known Taillard’s benchmark problem set [5].

In one of our previous work [6] a uniform approach was proposed for applying various types of chromosome based evolutionary algorithms for the PFSP problem. The proposal included two ways for individual representation and the corresponding evolutionary operators built up from three so called atomic operators.

Our recent work [7] proposed approaches for combining the Iterated Greedy techniques as state-of-the-art methods with bacterial evolutionary algorithms to efficiently solve the Permutation Flow Shop Problem. The best resulting Bacterial Iterated Greedy method clearly outperformed the Iterated Greedy heuristic.

Studies about parallel Iterated Greedy techniques involving Memetic Algorithm (e.g. [8]) showed that in case of multi-threaded Iterated Greedy methods the combination with evolutionary algorithms was able to improve the performance of simple multi-threaded Iterated Greedy algorithms.

These results motivated the idea that it might be also worth to try to replace the genetic algorithm based memetic method in [8] with the Bacterial Memetic Algorithm, which appears
to be more effective for the PFSP task (cf. [6]) and which shows better properties in other fields of optimization, too (see e.g. [9], [10]).

Therefore, this paper proposes approaches for combining multi-threaded Iterated Greedy techniques as state-of-the-art methods with bacterial evolutionary algorithms to efficiently solve the Permutation Flow Shop Problem on parallel computing architectures. In the present work three novel approaches are proposed by combining a variant of the Bacterial Memetic Algorithm and the recently proposed Bacterial Iterated Greedy technique with the mentioned hybrid multi-threaded approach.

The techniques obtained are evaluated via simulation runs carried out on the well known Taillard’s benchmark problem set. Based on the experimental results the hybrid methods are compared to each other and to the original techniques applied e.g. in [11] (i.e. to the techniques without bacterial algorithms).

The next section gives a formal definition to the PFSP problem. Within this, the search space and the makespan function as the objective function are defined. Then, the third section briefly describes the bacterial evolutionary and memetic techniques being combined with the iterated greedy algorithms, furthermore it gives a brief overview of the single and multi-threaded Iterated Greedy methods, since these techniques appear in the hybrid approaches. The basic concept and the main steps of the algorithms are also presented. The new combination approaches for multi-threaded heuristics are proposed in section four. The experimental results and the observed characteristics are presented in section five. Finally, in the last section our work is summarized and some conclusions are drawn.

II. THE PERMUTATION FLOW SHOP PROBLEM

As it was described in the Introduction, in this problem there is given the number of jobs \( n \), the number of machines \( m \) and an \( n \times m \)-size processing time matrix \( P \) defining the necessary amount of time a job has to stay on a machine, for each job-machine pair. That is, the elements of the matrix are positive and an element \( p_{i,j} \) denotes the time the \( i \)th job stays on the \( j \)th machine.

All the jobs should be worked by all the machines one after another. The machines are deployed in a line and each machine can handle one single job at once. That is, the processing of the jobs is pipeline-like. A job can be processed on a machine only if the machine is free (the preceding job has finished on the machine) and the job has already processed on the preceding machine.

The task is to find a permutation (a sequence) of the jobs, in case of which the total processing time of all the jobs on all the machines (i.e. the so called makespan) is minimal.

For example, if there are three jobs the permutation (2, 3, 1) denotes the case when the second job goes first, the third goes next, and finally the first goes last.

Clearly, the search space is the set of the \( n \)-order permutations \( S_n \), and the objective function is defined over this search space and its range is the set of positive numbers.

Formally, the objective or makespan function \( f \) can be defined as follows (see e.g. [1]).

\[
f: S_n \rightarrow \mathbb{R}^+ \\
f(\sigma) = t(n, m, \sigma) \\
t(0, j, \sigma) = 0 \\
t(i, 0, \sigma) = 0 \\
t(i,j,\sigma) = \max(t(i,j-1,\sigma),t(i-1,j,\sigma)) + p_{\sigma(i),j} \tag{1}
\]

The task is to find a permutation for which the makespan is optimal (i.e. minimal).

III. OVERVIEW OF THE TECHNIQUES APPLIED

The purpose of this section is to enumerate and shortly describe the techniques applied in the establishment of the hybrid algorithms. Thus, in the first part of this section after a brief introduction to chromosome based evolutionary techniques, the skeleton of the Genetic and Bacterial Evolutionary Algorithms will be presented followed by the idea of memetic algorithms. Then, the uniform approach for applying chromosome based techniques, like the Bacterial Memetic Algorithm, to the PFSP problem is described, which is proposed and deeply discussed in [6]. This includes the description of both the encoding methods and the evolutionary operators applied in this research. In the second part of the section the well known Iterated Greedy method together with its bacterial hybridization and multi-threaded variant will be presented shortly.

Due to space limitations the approaches will only be outlined very briefly. For further details the reader should refer to the cited literature.

A. Chromosome based evolutionary algorithms

A famous, frequently studied and applied family of iterative stochastic optimization techniques is called chromosome based evolutionary algorithms. These methods, like the Genetic Algorithm (GA) [12] or the Bacterial Evolutionary Algorithm (BEA) [13], imitate the abstract model of the evolution of populations observed in the nature. Their aim is to change the individuals in the population (set of individuals) by the evolutionary operators to obtain better and better ones.

1) Genetic Algorithm: One of the most (if not the most) widely applied chromosome based evolutionary techniques is the Genetic Algorithm (GA) [12]. Due to its notoriety its further description is omitted here.

2) Bacterial Evolutionary Algorithm: Compared to GA, a somewhat different evolutionary technique is called Bacterial Evolutionary Algorithm (BEA) [13]. BEA has proved a rather efficient method among chromosome based techniques for various optimization tasks, including the PFSP problem (cf. e.g. [9], [10], [6]).

BEA comprises the following steps:

1) Initialization
2) Bacterial mutation
3) Gene transfer
3) Memetic algorithms: The techniques causing minor modifications to the candidate solutions iteration-by-iteration and thus exploring only the ‘neighborhood’ of particular elements of the search space are called local search methods. Since evolutionary algorithms explore the whole objective function, i.e. they are global search techniques in contrast with local search approaches, evolutionary and local search methods may be combined resulting in memetic algorithms [14]. For example, if in each iteration for each individual one or more local search steps are applied. Expectedly, this way the advantages of both local search and evolutionary techniques can be exploited.

4) Encoding methods: The encoding method for the PFSP problem is an indirect, real value based approach [6], which is an obvious extension of those representations applied for numerical optimization problems. Although, the operators modify the values of real valued vectors — since the objective function is defined over permutations, the chromosomes represent permutations actually — there is a need to convert the real valued vectors to permutations somehow. This can be done by ordering the genes according to the values they have.

5) Evolutionary operators: The short description of the applied evolutionary operators is as follows [6]:

- **Mutation:**
  When a gene is mutated, it is set to a random real value. Thus, the permutation represented by the chromosome changes, because the order of the real values in the chromosome changes.

- **Gene transfer:**
  The real value of the selected gene in the target individual is set to the real value of the corresponding gene of the source individual.

- **Local search:**
  One iteration cycle of the local search is the following. First of all, a random order of the elements of the permutation from the first to the last but one is selected. Then, following this order the neighboring elements according to the permutation represented by the chromosome are tried to change their values with each other so that if according to the random order the current element is the \(i\)th, then it is tried to change its value with the \((i + 1)\)th. After each change between the neighbors if the resulting permutation is better (i.e. it has a higher fitness value), the change is kept. Otherwise, the change is rolled back.

**B. Iterated Greedy methods**

The Iterated Greedy (IG) technique [11] is a very simple and intuitive but rather efficient heuristic for the PFSP problem. The basic method will be described below followed by the bacterial hybrid version and the multi-threaded variant.

1) **The Iterated Greedy technique:** Basically, the Iterated Greedy (IG) technique [11] comprises four steps:

1) **Initialization:**
   An initial permutation is created by using the deterministic NEH heuristic [15] (which is not described in this paper). This permutation is stored as best.

2) **Destruction phase:**
   A predefined number of jobs are selected randomly, and they are removed from the permutation.

3) **Construction phase:**
   The removed jobs are reinserted into the destructed permutation to those places in case of which the partial permutations have the lowest makespan values.

4) **Acceptance check:**
   If the newly created permutation is better than the original one, it is kept. If it is better than the ever best, it is stored as the new best solution. If the newly created permutation is worse than the original one, it may also be kept with a probability depending on a so-called ‘temperature parameter’ and on the difference between the makespan value of the original and the new solution. Otherwise the new permutation is ignored.

The main iteration loop of the algorithm contains steps 2 – 4. The algorithm stops, if at the end of an iteration one of the termination criteria fulfills (iteration limit reached, time limit exceeded, etc.). After termination the stored best permutation represents the quasi-optimal solution.

Local search steps can also be applied within the Iterated Greedy method. Usually, in this case the so called Iterative Insertion Improvement algorithm (III) [11] is used between steps 3 and 4. During one iteration of this local search a destruction and a construction phase take place for one single job at once, but this is performed for each job in the permutation. The local search iterates until the last iteration shows no improvement.

2) **The Bacterial Iterated Greedy algorithm:** In our recent work [7] a number of hybrid bacterial iterated greedy variants were established, however in this paper only the best one will be described and involved in the multi-threaded approaches.

In the (best established) Bacterial Iterated Greedy (BIG) algorithm the above discussed Bacterial Memetic Algorithm is embedded into the Iterated Greedy heuristic. Since the Iterated Greedy method considers only one candidate solution at once, whereas the bacterial algorithms maintain a whole population of the permutations, the population of the bacterial algorithm is created by the multiple execution of the destruction phase. Apparently, after each destruction the corresponding construction phase follows in order to have valid permutations before the embedded bacterial heuristic starts. Here comes the main loop of the embedded technique, which iterates a predefined number of times on the created population. After that, the iterated greedy method continues with the acceptance check immediately involving the best individual from the bacterial population.

Thus, the hybrid algorithm comprises the following steps:

1) **Initialization**
2) **Multiple destruction phase**
3) **Construction phase**
4) **Embedded bacterial algorithm**
   a) Bacterial mutation
   b) Gene transfer
5) The best individual is selected from the bacterial population for further usage and the rest of the population is disregarded.

6) Acceptance check

The main iteration loop of the algorithm contains steps 2 – 7. The algorithm stops, if at the end of an iteration one of the termination criteria fulfills (iteration limit reached, time limit exceeded, etc.). After termination the stored best permutation represents the quasi-optimal solution.

3) Multi-threaded Iterated Greedy techniques: Considering multi-processor computer systems parallel Iterated Greedy algorithm types running on multiple threads were also invented. A simple parallel extension of the IG method is the Multi-threaded Iterated Greedy (MIG) technique [8]. In this case an instance of the IG algorithm is running on each thread until the termination condition is fulfilled. Then, the best one among the permutations given by the threads will be the quasi-optimal solution. The steps of MIG are the following:

1) Initialization of the parallel threads:
   An initial permutation is created for each thread by using the deterministic NEH heuristic [15].

2) Parallel execution of iterated greedy methods:
   Every thread executes an iterated greedy technique until a termination condition is fulfilled.

3) Selection of the best candidate solution:
   The resulting permutations given by the parallel executed methods are collected and the best one is selected as the result of the optimization process.

A more sophisticated parallel extension of the IG method is a combination of a genetic algorithm based memetic technique and the MIG algorithm. This is referred to as MA+MIG (Memetic Algorithm + Multi-threaded Iterated Greedy) method by the inventors [8]. In this heuristic one thread is executing a genetic algorithm based memetic technique, while the others are running the IG method. During the optimization process the memetic algorithm and the IG threads are communicating with each other via a migration pool, where a larger number of individuals take place. This way, candidate solutions are continuously migrating between the threads. The steps of MA+MIG are the following:

1) Initialization of the parallel threads:
   The migration pool is filled with individuals partly by using initial heuristics and partly by generating random permutations. The threads take their initial candidate solutions from this pool.

2) Parallel execution:
   A genetic algorithm based memetic technique is running on one thread, while each other thread is executing an iterated greedy technique concurrently until a termination condition is fulfilled. During this process in case of the fulfillment of certain conditions, the threads exchange (migrate) their candidate solutions by using the migration pool.

3) Selection of the best candidate solution:
   The resulting permutations given by the parallel executed methods are collected and the best one is selected as the result of the optimization process.

IV. HYBRID MULTI-THREADED BACTERIAL APPROACHES

Unlike in the single threaded case, if multiple threads are considered, i.e. there are more than one algorithms running parallel, even if the original techniques are executed on every thread, a hybrid method can be obtained by running different methods on different threads. An example to such a hybrid multi-threaded heuristic is the above described MA+MIG method, where the original algorithms are running on all the threads, mostly IG techniques parallel, but there is one exceptional thread, where the genetic algorithm based memetic method is executed.

This paper proposes similar (but hopefully better) hybrid techniques by exchanging the heuristics on the threads. Considering our recent work on chromosome based evolutionary methods [6] and the previously discussed single-threaded techniques three hybrid bacterial approaches are proposed for multi-threaded PFSP optimization. They are originated from the MA+MIG method by making the following changes, respectively:

1) The genetic algorithm based memetic heuristic is exchanged with the bacterial memetic technique and the iterated greedy threads are left untouched (MBIG_1).

2) The genetic algorithm based memetic thread is left untouched and the iterated greedy method is exchanged with a hybrid single-threaded bacterial iterated greedy technique on every thread (MBIG_2).

3) Both heuristics are exchanged, i.e. the bacterial memetic technique is executed on one thread, while a hybrid single-threaded bacterial iterated greedy method is applied on the other threads (MBIG_3).

During the optimization the candidate solutions migrate between the threads via the migration pool and at the beginning of the optimization process the threads are also initialized with the individuals from the pool. The size of the migration pool equals to the sum of the size of the bacterial population and the number of iterated greedy threads. Thus, the corresponding individuals can be determined for every thread, which assignment is used during the migration.

The migration between the threads and the pool occurs according to a clock. When a certain amount of time elapsed after the last migration each thread overwrites the individual(s) in the pool assigned to the algorithm instance. Then, they take new permutations from the pool randomly. Although, the threads take new individuals, they keep their best ever permutations. The time gap between migrations is a parameter of the multi-threaded algorithm.

On the threads arbitrary single-threaded bacterial iterated greedy algorithm can be applied.

V. EVALUATION OF THE OBTAINED TECHNIQUES

Simulation runs were carried out in order to evaluate and to compare the efficiency of the proposed approaches and the
established algorithms.

For this purpose, a dozen problems were applied from the well known Taillard’s benchmark set. Exactly one problem from each available problem sizes.

In the simulations the parameters had the following values, because after a number of test runs these values seemed to be the most suitable.

In the bacterial algorithms the number of individuals in a generation was 8, the number of clones was 2 and 1 gene transfer was carried out in each generation. In the iterated greedy methods 4 jobs were selected to remove in each generation and the temperature parameter was 5 (see [11]). The run of the embedded techniques took 3 iterations. The strength parameter for the parameterized distributions was 0.99.

The multi-threaded methods used 8 threads and the time gaps between migrations were calculated according to the following formula:

\[ \text{time\_gap} = \frac{\text{time\_limit}}{\text{num\_of\_threads}} \times 10 \]  

(2)

where \( \text{time\_limit} \) is the length of the simulation and \( \text{num\_of\_threads} \) denotes the number of threads the algorithm uses.

The simulation was carried out on a PC with E8500 3.16 GHz Intel Core 2 Duo CPU and using Windows Vista Business 64-bit operating system.

In case of all the algorithms for all benchmark problems 10 runs were carried out. Then the mean of the obtained values were taken.

The means of the resulting values were collected in tables. In Table I and Table II under the ‘Problem’ label the ‘ID’ columns show the identifier of the tasks in Taillard’s benchmark problem set [5] and ‘Size’ denotes the size of the benchmark problem (in the form of “number of jobs \times number of machines”). The best known makespan values according to the website of Taillard\(^1\) (which was last updated in 2005) are collected in columns labeled by ‘B.k.m.v.’. ‘Time limit’ shows the length of the simulation runs in seconds. The time limits were chosen according to test runs to values, after which the techniques did not show significant improvements. Under the algorithm labels the ‘Results’ columns present the mean of the makespan values produced by the techniques, ‘Rel. diff.’ shows the mean of the relative differences of these makespan values compared to the known best ones:

\[ \frac{1}{5} \sum_{i=1}^{5} \frac{(\text{Result}_i - \text{Best known value})}{(\text{Best known value})}, \]

(3)

whereas ‘Std. dev.’ denotes the standard deviation values of the results.

A. Experimental results for the hybrid methods

The three novel multi-threaded algorithm types and the MA+MIG multi-threaded hybrid heuristic [8] were compared based on the previously applied 12 benchmark problems from the Taillard’s set. The BIG variant was applied as single-threaded algorithm on the threads besides the memetic thread in MBIG\(_2\) and MBIG\(_3\) methods. The results of the runs are collected in Table I and Table II. The average of the relative difference values (ARD) of the MBIG versions are shown in Table III. The ARD value of the technique reaching the highest performance is heightened.

Considering Table I – Table III the following observations can be made. Surprisingly, the methods applying BIG (MBIG\(_2\) and MBIG\(_3\)) performed worse than the original MA+MIG algorithm. However, one of the newly proposed hybrid techniques, namely MBIG\(_1\), outperformed the MA+MIG heuristic, thus it appears to be the best established multi-threaded hybrid bacterial iterated greedy heuristic. Although, the superiority of MBIG\(_1\) over MA+MIG does not appear in case of every problem instance as Table I shows, it can be observed that MBIG\(_1\) performed better than MA+MIG (cf. Table I and Table III).

VI. Conclusion

In this paper approaches were proposed for combining a multi-threaded iterated greedy technique [8] as a state-of-the-art method with an adapted version of the Bacterial Memetic Algorithm [6] and the recently proposed Bacterial Iterated Greedy technique [7] in order to efficiently solve the Permutation Flow Shop Problem on parallel computing architectures.

The obtained techniques were evaluated via simulation runs carried out on the well known Taillard’s benchmark problem set. Based on the experimental results the hybrid methods are compared to each other and to the original multi-threaded technique. During the experimental analysis the following conclusion could be made.

The best hybrid method appeared to be more efficient than the original MA+MIG heuristics. That is, it is definitely worth to apply bacterial techniques in combination with the hybrid multi-threaded iterated greedy method. However, as the variegation of the performances given by different combinations show, the hybridization manner must be selected carefully.

Similarly like in this work, the bacterial techniques can also be combined with other techniques. Thus, future work may aim to establish more hybrid methods involving other state-of-the-art optimization algorithms for the PFSP problem, including parallel multi-threaded methods as well, and to compare them with the ones proposed in our recent and present work.

ACKNOWLEDGMENT

The research is supported by the National Development Agency and the European Union within the frame of the project TÁMOP-4.2.2-08/1-2008-0021 at the Széchenyi István University entitled “Simulation and Optimization — basic research in numerical mathematics", the National Scientific Research Fund Grant OTKA K105529, a Széchenyi István University Main Research Direction Grant and the Social Renewal Operation Programmes TÁMOP-4.2.2 08/1-2008-0021 and 421 B.
TABLE I
RESULTS FOR THE MBIG TECHNIQUES

<table>
<thead>
<tr>
<th>ID</th>
<th>Size</th>
<th>B.k.m.v.</th>
<th>Time limit</th>
<th>Result</th>
<th>Rel. diff.</th>
<th>Std. dev.</th>
<th>Result</th>
<th>Rel. diff.</th>
<th>Std. dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ta001</td>
<td>20x5</td>
<td>1278</td>
<td>10</td>
<td>1278</td>
<td>0.00%</td>
<td>0.00</td>
<td>1278</td>
<td>0.00%</td>
<td>0.00</td>
</tr>
<tr>
<td>ta011</td>
<td>20x10</td>
<td>1582</td>
<td>10</td>
<td>1583.4</td>
<td>0.99%</td>
<td>1.43</td>
<td>1584.2</td>
<td>0.14%</td>
<td>1.55</td>
</tr>
<tr>
<td>ta021</td>
<td>20x20</td>
<td>2297</td>
<td>10</td>
<td>2305.1</td>
<td>0.35%</td>
<td>4.79</td>
<td>2304.1</td>
<td>0.31%</td>
<td>5.84</td>
</tr>
<tr>
<td>ta031</td>
<td>50x5</td>
<td>2724</td>
<td>50</td>
<td>2724</td>
<td>0.00%</td>
<td>0.00</td>
<td>2724</td>
<td>0.00%</td>
<td>0.00</td>
</tr>
<tr>
<td>ta041</td>
<td>50x10</td>
<td>2991</td>
<td>50</td>
<td>3078.5</td>
<td>2.93%</td>
<td>8.03</td>
<td>3065</td>
<td>2.47%</td>
<td>10.38</td>
</tr>
<tr>
<td>ta051</td>
<td>50x20</td>
<td>3847</td>
<td>50</td>
<td>3967.1</td>
<td>3.12%</td>
<td>7.87</td>
<td>3957.6</td>
<td>2.87%</td>
<td>13.08</td>
</tr>
<tr>
<td>ta061</td>
<td>100x5</td>
<td>5493</td>
<td>200</td>
<td>5493</td>
<td>0.00%</td>
<td>0.00</td>
<td>5493</td>
<td>0.00%</td>
<td>0.00</td>
</tr>
<tr>
<td>ta071</td>
<td>100x10</td>
<td>5770</td>
<td>200</td>
<td>5814.8</td>
<td>0.78%</td>
<td>9.57</td>
<td>5803</td>
<td>0.52%</td>
<td>13.13</td>
</tr>
<tr>
<td>ta081</td>
<td>100x20</td>
<td>6202</td>
<td>200</td>
<td>6426</td>
<td>3.61%</td>
<td>15.34</td>
<td>6409</td>
<td>3.34%</td>
<td>19.64</td>
</tr>
<tr>
<td>ta091</td>
<td>200x10</td>
<td>10862</td>
<td>1000</td>
<td>10862</td>
<td>0.00%</td>
<td>0.00</td>
<td>10899.3</td>
<td>0.35%</td>
<td>10.44</td>
</tr>
<tr>
<td>ta101</td>
<td>200x20</td>
<td>11181</td>
<td>1000</td>
<td>11480.2</td>
<td>2.68%</td>
<td>15.32</td>
<td>11478.8</td>
<td>2.66%</td>
<td>24.30</td>
</tr>
<tr>
<td>ta111</td>
<td>500x20</td>
<td>26059</td>
<td>5000</td>
<td>26059</td>
<td>2.47%</td>
<td>47.48</td>
<td>26469.2</td>
<td>1.57%</td>
<td>23.13</td>
</tr>
</tbody>
</table>

TABLE II
RESULTS FOR THE MBIG TECHNIQUES (CONTINUED)

<table>
<thead>
<tr>
<th>ID</th>
<th>Size</th>
<th>B.k.m.v.</th>
<th>Time limit</th>
<th>Result</th>
<th>Rel. diff.</th>
<th>Std. dev.</th>
<th>Result</th>
<th>Rel. diff.</th>
<th>Std. dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ta001</td>
<td>20x5</td>
<td>1278</td>
<td>10</td>
<td>1278</td>
<td>0.00%</td>
<td>0.00</td>
<td>1278</td>
<td>0.00%</td>
<td>0.00</td>
</tr>
<tr>
<td>ta011</td>
<td>20x10</td>
<td>1582</td>
<td>10</td>
<td>1596.4</td>
<td>0.91%</td>
<td>9.81</td>
<td>1591</td>
<td>0.57%</td>
<td>4.74</td>
</tr>
<tr>
<td>ta021</td>
<td>20x20</td>
<td>2297</td>
<td>10</td>
<td>2320.7</td>
<td>1.03%</td>
<td>9.62</td>
<td>2316.2</td>
<td>0.84%</td>
<td>7.08</td>
</tr>
<tr>
<td>ta031</td>
<td>50x5</td>
<td>2724</td>
<td>50</td>
<td>2724</td>
<td>0.00%</td>
<td>0.00</td>
<td>2724</td>
<td>0.00%</td>
<td>0.00</td>
</tr>
<tr>
<td>ta041</td>
<td>50x10</td>
<td>2991</td>
<td>50</td>
<td>3087.6</td>
<td>3.23%</td>
<td>10.38</td>
<td>3067.8</td>
<td>2.57%</td>
<td>11.17</td>
</tr>
<tr>
<td>ta051</td>
<td>50x20</td>
<td>3847</td>
<td>50</td>
<td>3982</td>
<td>3.51%</td>
<td>10.84</td>
<td>3966.7</td>
<td>3.11%</td>
<td>5.68</td>
</tr>
<tr>
<td>ta061</td>
<td>100x5</td>
<td>5493</td>
<td>200</td>
<td>5493</td>
<td>0.00%</td>
<td>0.00</td>
<td>5493</td>
<td>0.00%</td>
<td>0.00</td>
</tr>
<tr>
<td>ta071</td>
<td>100x10</td>
<td>5770</td>
<td>200</td>
<td>5807.1</td>
<td>0.64%</td>
<td>7.89</td>
<td>5804.9</td>
<td>0.60%</td>
<td>9.11</td>
</tr>
<tr>
<td>ta081</td>
<td>100x20</td>
<td>6202</td>
<td>200</td>
<td>6439.6</td>
<td>3.83%</td>
<td>15.45</td>
<td>6417.4</td>
<td>3.47%</td>
<td>17.14</td>
</tr>
<tr>
<td>ta091</td>
<td>100x20</td>
<td>10862</td>
<td>1000</td>
<td>10862</td>
<td>0.00%</td>
<td>0.00</td>
<td>10885.4</td>
<td>0.00%</td>
<td>0.00</td>
</tr>
<tr>
<td>ta101</td>
<td>200x20</td>
<td>11181</td>
<td>1000</td>
<td>11496.5</td>
<td>2.82%</td>
<td>26.84</td>
<td>11480.2</td>
<td>2.68%</td>
<td>24.00</td>
</tr>
<tr>
<td>ta111</td>
<td>500x20</td>
<td>26059</td>
<td>5000</td>
<td>26059</td>
<td>1.71%</td>
<td>47.48</td>
<td>26469.2</td>
<td>1.57%</td>
<td>23.13</td>
</tr>
</tbody>
</table>

TABLE III
AVERAGE OF THE RELATIVE DIFFERENCE VALUES OF THE MULTI-THREADED TECHNIQUES

<table>
<thead>
<tr>
<th>Version</th>
<th>ARD</th>
</tr>
</thead>
<tbody>
<tr>
<td>MA+MIG</td>
<td>1.2787%</td>
</tr>
<tr>
<td>MBIG_1</td>
<td>1.9100%</td>
</tr>
<tr>
<td>MBIG_2</td>
<td>1.4920%</td>
</tr>
<tr>
<td>MBIG_3</td>
<td>1.3030%</td>
</tr>
</tbody>
</table>

REFERENCES