# Heuristic Algorithms For Scheduling Jobs On Identical Parallel Machines Via Measures Of Spread

M.I. Gualtieri, P. Pietramala and F. Rossi *

*Abstract*— **Constructive heuristics for the scheduling problem of $n$ independent jobs on $m$ identical parallel machines with minimum makespan objective are described. The proposed algorithms, which are $n \log n$ algorithms as the *LPT* algorithm of Graham, iteratively combine partial solutions that are obtained by partitioning the set of jobs into suitable families of subsets. The procedure used to partition the jobs in partial solutions and the rule used for selecting the partial solutions to combine, are designed to reduce the measure of spread of the processing times $m$-set associated to the partial solution. The algorithms were tested using different families of instances taken from the literature and results compared with other algorithms.**

***Keywords:*** ***Combinatorial optimization, scheduling, heuristic algorithm, measures of spread.***

## 1 Introduction

In this paper, we study the scheduling problem of $n$ independent jobs on $m$ parallel machines. Each job $i$ must be processed without interruption by only one of the $m$ machines (non-preemptive environment); as the machines are identical, the processing time $p_i$ of the job $i$ is independent of the processing machine (environment of identical parallel processors). Our goal is to minimize the makespan, i.e., the total time required to complete all jobs. Using the standard three field classification scheme of Graham et al. (1979), this problem is usually denoted as $P||C_{max}$ and it is well known to be NP-hard in strong sense for an arbitrary $m \geqslant 2$, see Garey and Johnson (1978), and Ullman (1976). For large instances, one needs to rely on good heuristic procedures to provide solutions that are probably close to the optimum. Heuristic algorithms are classified into constructive algorithms and improvement algorithms. The list scheduling family of Graham (1966 and 1969), which includes the Largest Processing Time ($LPT$), and the MultiFit Decreasing ($MFD$) scheduling algorithm of Coffman et al. (1978), belongs to first category. Hochbaum and Shmoys (1987) presented a polynomial approximation scheme ($PTAS$) that seems to be, in some sense, the best possible.
Improvement algorithms have been proposed, for example, by França et al. (1994), Anderson et al. (1997) and Frangioni et al. (2004).
Surveys on the heuristic algorithms for parallel machine scheduling problems have been provided by Cheng and Sin (1990), by Lawler et al. (1993) and by Chen et al. (1998). An overview of existing results and of recent research areas is presented in the handbook edited by Leung (2004).

In this paper, we present some constructive $n \log n$ algorithms. These algorithms, from an analysis of the computational results with respect to the relative error, are more accurate than the $n \log n$ $LPT$ algorithm of Graham and seem comparable in several cases to the two considered improvement algorithms. As in Paletta and Pietramala (2007) and Gualtieri et al. (2008), our algorithms are based on the idea of combining iteratively partial solutions until a feasible solution for the scheduling problem is obtained. The initial partial solutions are calculated by partitioning the set of jobs into $z$ subsets such that the elements of related $m$-set of processing times are as close to each other as possible. In order to do this, we make use of the measures of spread, that are commonly employed techniques in statistical problems.
Our paper improves the results of Gualtieri et al. (in which only the gap between the maximum and the minimum processing time is considered), because it sheds light on the fact that the key point in our algorithm is to minimize the variability between all the elements of the processing times $m$-set associated with a partial solution. In order to compare the heuristics with other algorithms, we used different families of instances, taken from the literature, for our computational investigation.

The paper is organized as follows. Section 2 presents the partitions that are used to design the algorithms and summarizes the basic elements of statistical meth-

*We thank F.M. Muller and E. Necciari for having provided the generators and the instances used for the computational tests. Maria Italia Gualtieri, Dipartimento di Matematica, Università della Calabria, 87036 Arcavacata di Rende, Cosenza, Italy, Email: mig.gualtieri@unical.it; Paolamaria Pietramala, Dipartimento di Matematica, Università della Calabria, 87036 Arcavacata di Rende, Cosenza, Italy, Email: pietramala@mat.unical.it; Ferdinando Rossi, Dipartimento di Sociologia e Scienza Politica, Università della Calabria, 87036 Arcavacata di Rende, Cosenza, Italy Email: ferdinando.rossi@unical.it.

ods. Section 3 contains the description of the algorithms and the results of the computational investigation concerning $P||C_{max}$.

## 2 Partial Solutions and Measures of Spread

Let $I = \{1, ..., i, ..., n\}$ be the set of $n$ independent jobs, $M = \{1, ..., j, ..., m\}$ be the set of $m$ identical parallel machines and $A = \{p_1, ..., p_i, ..., p_n\}$ be the set of processing times of the jobs.
We associate to a partition of $m \cdot (z-1) + s$ subsets $\Im = \{I_1^1, ..., I_j^1, ..., I_m^1, ..., I_1^r, ..., I_m^r, ..., I_1^z, ..., I_j^z, ..., I_s^z\}$, $s \leqslant m$ of the set $I$, the family of $z$ partial solutions $\mathfrak{P} = \{\Im^1, \ldots, \Im^r, \ldots, \Im^z\}$. Here $\Im^r = \{I_1^r, \ldots, I_j^r, \ldots, I_m^r\}$, $r = 1, ..., z-1$, represents the $r$th partial solution and $I_j^r$ the set of jobs that are performed by the machine $j$; in the $z$th partial solution $\Im^z = \{I_1^z, \ldots, I_j^z, \ldots, I_s^z, \emptyset_{s+1}, \ldots, \emptyset_m\}$, $s \leqslant m$, the notation $\emptyset_j$ means that the machine $j$ is not performing any jobs.
Let $p_j^r := \sum_{i \in I_j^r} p_i$ be the sum of the processing times of the jobs belonging to $I_j^r$, $r = 1, ..., z-1$ and $j = 1, ..., m$, let $p_j^z := \sum_{i \in I_j^z} p_i$ be the sum of the processing times of the jobs belonging to $I_j^z$ for $j = 1, ..., s$ and $p_j^z := 0$ for $s < j \leqslant m$. Each partial solution $\Im^r$, $r = 1, ..., z-1$, has, associated with it, the processing times $m$-set $G^r = \{p_1^r, \ldots, p_j^r, \ldots, p_m^r\}$, whereas $\Im^z$ has, associated with it, the processing times $m$-set $G^z = \{p_1^z, \ldots, p_s^z, 0, \ldots, 0\}$, $s \leqslant m$.
A partial solutions $\Im^r$ is called an *ordered partial solution* if the elements of $G^r$ are sorted in not increasing order with respect to their size, that is,

$$p_1^r \geqslant \ldots \geqslant p_j^r \geqslant \ldots \geqslant p_m^r.$$

A family of $z$ partial solutions $\mathfrak{P} = \{\Im^1, \ldots, \Im^r, \ldots, \Im^z\}$ is called *ordered $z$-family of partial solutions* if every $\Im^r$ is an ordered partial solution.
Let $\Im^r$ and $\Im^q$ be two ordered partial solutions. The *combination* among $\Im^r$ and $\Im^q$ is a new partial solution defined as the $m$-family

$$\Im^r \uplus \Im^q := \{I_1^r \cup I_m^q, \ldots, I_j^r \cup I_{m-j+1}^q, \ldots, I_m^r \cup I_1^q\}.$$

where the set $I_j^r \cup I_{m-j+1}^q$, $j = 1, \ldots, m$, represents the jobs performed by the machine $j$. The total processing time needed for the machines to perform all the jobs belonging to $\Im^r \uplus \Im^q$ is computed by using the *sum* between the processing times $m$-sets $G^r$ and $G^q$ defined as the $m$-set

$$G^r \oplus G^q := \{p_1^r + p_m^q, \ldots, p_j^r + p_{m-j+1}^q, \ldots, p_m^r + p_1^q\}.$$

Thus $p_j^r + p_{m-j+1}^q$ represents the total processing time required to perform all the jobs belonging to $I_j^r \cup I_{m-j+1}^q$. Note that $\Im^r \uplus \Im^q$ is a partial solution that is not necessarily ordered since the elements of $G^r \oplus G^q$ are not sorted in non-increasing order with respect to their size.

The *ordered sum* among $G^r$ and $G^q$ is the ordered $m$-set $Ord(G^r \oplus G^q)$ whose elements are the elements of $G^r \oplus G^q$ sorted in non-increasing order with respect to their size and the *ordered combination* among $\Im^r$ and $\Im^q$ is the $m$-family $Ord(\Im^r \uplus \Im^q)$ whose sets are those of $\Im^r \uplus \Im^q$ sorted such that the $j$-th element of $Ord(G^r \oplus G^q)$ represents the total processing time of the $j$-th job-set of $Ord(\Im^r \uplus \Im^q)$.

In this paper we want to construct an ordered $z$-family of partial solutions for the scheduling problem and combine them until a feasible solution is obtained.
Since the objective is to minimize the makespan of the solution, one tries to minimize the makespan of every partial solution. This is done by making the $m$ elements of the processing times sets very "close" to each other. In order to measure this "closeness" we utilize the statistical measures of spread. A *measure of variability or spread* quantifies how dispersed the values in a data set tend to be. It is a real number that is zero if all the data are identical and increases as the data become more diverse. If the spread is small, most of the data are nearly equal; if the spread is large, there are large differences among the data.
In the study of the variability of a data set, various types of measures of spread are used. Let $X = \{x_1, x_2, ..., x_j\}$ be a set of real values.

The *average or arithmetic mean or mean* of $X$ is defined by

$$\mu(X) = \frac{1}{j} \sum_{i=1,\ldots,j} x_i.$$

The *pth percentile* of $X$ is the smallest number that is at least as large as $p\%$ of the numbers in $X$. Some percentiles have special names: the *lower quartile of $X$ or* $Q_1(X)$ is the $25th$ *percentile*, the *median of $X$ or* $Q_2(X)$ is the $50th$ *percentile* and the *upper quartile of $X$ or* $Q_3(X)$ is the $75th$ *percentile*.
The mean and the median of $X$ are *measures of location*: they are, depending upon the definition of proximity of two numbers, "as close as possible" to all the elements of $X$.

The simplest measures which characterize the amount of spread are the *range of $X$*, or $R(X)$, which is the difference between the maximum and the minimum value in the set $X$, and the *interquartile range of $X$*, or $IQR(X)$, which is the difference between upper and lower quartiles of $X$:

$$R(X) := \max X - \min X, \quad IQR(X) := Q_3(X) - Q_1(X).$$

They do not measure the spread of the majority of values in the data set, they only measure the spread between two values.

The *variance of $X$* is a measure of the degree of dispersion

of the elements in $X$ from the mean value of $X$, given by

$$\sigma^2(X) := \frac{1}{j} \sum_{i=1,...,j} (x_i - \mu(X))^2.$$

This is a "natural" measure of dispersion if the center of the data is measured about the mean, because the function $f(\theta) = \frac{1}{j} \sum_{i=1,...,j}(x_i - \theta)^2$ has a unique minimum at $\theta = \frac{1}{j} \sum_{i=1,...,j} x_i$.

The *standard deviation of X*, defined as the square root of the variance

$$\sigma(X) := \sqrt{\sigma^2(X)} = \sqrt{\frac{1}{j} \sum_{i=1,...,j} (x_i - \mu(X))^2}$$

is easier to handle from a mathematical point of view and has the same unity of measurement as the data.

The *mean absolute deviation of X*

$$\alpha(X) := \frac{1}{j} \sum_{i=1,...,j} |x_i - Q_2(X)|.$$

is a "natural" measure of dispersion if the center of the data set is measured about the median, because the function $f(\theta) = \frac{1}{j} \sum_{i=1,...,j} |x_i - \theta|$ has a unique minimum at the median of $\{x_1, x_2, ..., x_j\}$.

The variance, the standard deviation and the mean absolute deviation of $X$ are measure of variability that describe how much the values of $X$ are close to a typical value of $X$.

The *Gini's mean difference* of $X$, instead, is a measure that reflects how the values in $X$ differ from each other, that is,

$$GMD(X) := \frac{1}{j(j-1)} \sum_{i=1}^{j-1} \sum_{k=i+1}^{j} |x_i - x_k|.$$

The various measures of spread are independent from each other, but all of these statistical describers are influenced by the "extreme" values of the data.

## 3 Algorithms. Computational Results

*Algorithms*

The proposed *PSC* (Partial Solution Combination) algorithm, firstly partitions the jobs by using the *IPS* (Initial Partial Solution) procedure, in order to obtain an ordered $z$-family of partial solutions $\mathfrak{P} = \{\mathfrak{I}^1, \ldots, \mathfrak{I}^r, \ldots, \mathfrak{I}^z\}$. Secondly, the algorithm selects two ordered partial solutions whose processing times $m$-sets have bigger measures of spread and combines them with the ordered combination operator. The algorithm continues to iterate (exactly $z-1$ times) until a feasible solution of the scheduling problem is obtained.

Generalizing the approach of Gualtieri et al., both the procedure used to partition the jobs into an ordered $z$-family of partial solutions, and the rule used for selecting which two partial solutions are to be combined, are designed in order to reduce as much as possible the measure of spread of the processing times $m$-sets $G^r$ so that the $m$ elements of $G^r$ are as close as possible to each other and the makespan is minimized.

The *IPS* procedure, firstly orders the jobs so that $p_1 \geqslant p_2 \geqslant \ldots \geqslant p_n$. Secondly, *IPS* computes the *minimum* number $z$ of partial solutions, in which the first element is a singleton, as the greatest index of the jobs for which

$$\sum_{i=1,...,z \leqslant n} p_i \leqslant \max\Big\{p_1, p_m + p_{m+1}, \frac{1}{m} \sum_{i=1,...,n} p_i\Big\}$$

holds. Thirdly, by using as seeds the first $z$ jobs, the procedure initializes $z$ partial solutions. Finally, *IPS* processes the remaining jobs as follows. Suppose that the job $i$ must be inserted. Thus, *IPS* selects an ordered partial solution to which the greatest value of measure of spread is associated, say $\mathfrak{I}^r$. If $p^r_m + p_i \leqslant p^r_1$, the job $i$ is inserted in the last job-set of $\mathfrak{I}^r$; otherwise, *IPS* uses the job $i$ as a seed to initialize a new partial solution.
Let us denote by $V^r = V(G^r)$ the measure of variability of the processing times $m$-set $G^r$ associated with the ordered partial solution $\mathfrak{I}^r$.

*IPS* Procedure

Initialization

- Order the jobs so that $p_1 \geqslant \ldots \geqslant p_i \geqslant \ldots \geqslant p_n$. Set $z$ the greatest index of the jobs so that $\sum_{i=1,...,z \leqslant n} p_i \leqslant \max\{p_1, p_m + p_{m+1}, \frac{1}{m} \sum_{i=1,...,n} p_i\}$;
- Set $\mathfrak{I}^1 = \{I^1_1 = \{1\}, I^1_2 = \emptyset, \ldots, I^1_j = \emptyset, \ldots, I^1_m = \emptyset\}$, $\mathfrak{I}^2 = \{I^2_1 = \{2\}, I^2_2 = \emptyset, \ldots, I^2_j = \emptyset, \ldots, I^2_m = \emptyset\}$,..., $\mathfrak{I}^z = \{I^z_1 = \{z\}, I^z_2 = \emptyset, \ldots, I^z_j = \emptyset, \ldots, I^z_m = \emptyset\}$, and $G^1 = \{p^1_1 = p_1, p^1_2 = 0, \ldots, p^1_j = 0, \ldots, p^1_m = 0\}$, $G^2 = \{p^2_1 = p_2, p^2_2 = 0, \ldots, p^2_j = 0, \ldots, p^2_m = 0\}$,..., $G^z = \{p^z_1 = p_z, p^z_2 = 0, \ldots, p^z_j = 0, \ldots, p^z_m = 0\}$;
- Set $V^r = V(G^r)$, $r = 1, ..., z$;
- Set $\mathfrak{P} = \{\mathfrak{I}^1, \ldots, \mathfrak{I}^r, \ldots, \mathfrak{I}^{z-1}, \mathfrak{I}^z\}$ ($\mathfrak{P}$ is ordered so that $V^1 \geqslant \ldots \geqslant V^r \geqslant ... \geqslant V^z$).

Construction

For $i = z+1, \ldots, n$

- select $\mathfrak{I}^1$ (since $V^1 = \max_{r=1,...,z} V^r$);
- If $p^1_m + p_i \leqslant p^1_1$ then

- set $I_m^1 = I_m^1 \cup \{i\}$ and $p_m^1 = p_m^1 + p_i$;
- sort the elements of $G^1$ so that $p_1^1 \geqslant \ldots \geqslant p_j^1 \geqslant \ldots \geqslant p_m^1$;
- arrange $\mathfrak{I}^1$ so that $p_j^1$ is the total time required by the jobs belonging to $I_j^1$, for $j = 1, ..., m$;
- set $V^1 = V(G^1)$ and sort $\mathfrak{P}$ so that $V^1 \geqslant V^2 \geqslant ... \geqslant V^z$;

Otherwise

- set $z = z + 1$, $\mathfrak{I}^z = \{I_1^z = \{i\}, I_2^z = \emptyset, \ldots, I_j^z = \emptyset, \ldots, I_m^z = \emptyset\}$, $\mathfrak{P} = \mathfrak{P} \cup \mathfrak{I}^z$;
- set $G^z = \{p_1^z = p_i, p_2^z = 0, \ldots, p_j^z = 0, \ldots, p_m^z = 0\}$;
- set $V^z = V(G^z)$, and sort $\mathfrak{P}$ so that $V^1 \geqslant \ldots \geqslant V^r \geqslant ... \geqslant V^z$;

End If $p_m^1 + p_i \leqslant p_1^1$

End For $i$.

*PSC* Algorithm

Initialization

- Use the *IPS* procedure to obtain an ordered $z$-family of partial solutions $\mathfrak{P}$. If *IPS* returns with only one partial solution then Stop (the algorithm provides an optimal solution);

Construction

For $j = 1, \ldots, z-1$

- select two ordered partial solutions belonging to $\mathfrak{P}$ whose processing times $m$-sets have bigger measures of spread (say $\mathfrak{I}^l$ and $\mathfrak{I}^k$);
- compute $G^{z+j} = Ord(G^l \oplus G^k)$, $\mathfrak{I}^{z+j} = Ord(\mathfrak{I}^l \uplus \mathfrak{I}^k)$ and $V^{z+j} = V(G^{z+j})$;
- set $\mathfrak{P} = (\mathfrak{P} \backslash \{\mathfrak{I}^l, \mathfrak{I}^k\}) \cup \mathfrak{I}^{z+j}$;

End For $j$.

It is routine to show that the *PSC* algorithm runs in $O(nlog(n))$-time, which is the running time of the *IPS* procedure, when the measures range, interquartile range, variance, standard deviation and mean absolute deviation are utilized. When Gini's mean difference is used, the *PSC* algorithm runs in $O(nlog(n) + m^2)$-time.

*Implementation Outcome*

We have implemented and tested our heuristic with five different measures of variability, on three families of instances, in order to verify whether some choice is dominant with respect to the accuracy of the feasible solution.

We report all the cases; in fact, our investigation has shown that good results are obtained by using the range, the variance, the standard deviation, the mean absolute deviation and Gini's mean difference, whereas the inter-quartile range did not lead to significant performance results. In general, the variance, the standard deviation, the mean absolute deviation and Gini's mean difference showed to be the best choices. This result is not surprising, because such measures of spread involve all the elements of a partial solution. The results on the range could be surprising. However, this is explained by the fact that, as showed by Paletta and Pietramala, the smaller is the range associated with the initial ordered partial solutions, the smaller becomes the range associated with the feasible solution and, therefore, the smaller is the heuristic error.

*The Instances*

Three families of instances, which posses a very different structure, are taken from the literature.

In the first two families the number of machines $m$ is $5, 10, 25$, the number of jobs $n$ is $50, 100, 500, 1000$ (the case $m = 5$ and $n = 10$ is also tested), and the integer processing times belong to the intervals $[1, 100]$, $[1, 1000]$ and $[1, 10000]$. Ten instances are randomly generated for each choice of $m, n$ and of the processing times interval, with a total of 390 instances within each family.

These two families differ in shape of the distribution of processing times. In the UNIFORM family, presented by França et al., the processing times are generated by using an uniform distribution. The generator of the NONUNIFORM family, presented by Frangioni et al. (1999 and 2004), given an interval $[a, b]$ of the processing times, produces instances where 98% of the processing times are uniformly distributed in the interval $[(b-a)0.9, b]$, while the remaining processing times fall within the interval $[a, (b-a)0.02]$. Both generators are available at the URL www.di.unipi.it/di/groups/optimize/Data/index.html

The last family of instances (BINPACK) is obtained from bin packing instances, which are available at the OR-Library of J.E. Beasley, from

http://mscmga.ms.ic.ac.uk/jeb/orlib/binpackinfo.html

In this family the processing times are uniformly distributed in $[20, 100]$ and the number of machines $m$ is the number of bins in the best known solution of the bin packing instances.

*Plan of the Experimentation*

Our algorithms were tested on these three families of instances and were compared with the classical *LPT* heuristic of Graham, the *3-PHASE* heuristic of França et al. (1994) and the local search *K-SPT* algorithm of Frangioni et al. (2004). We exploit the results of Frangioni et al. (1999) for the *3-PHASE* and *K-SPT* algorithms, since we use the same instances. The results of Frangioni et al. do not include the number of instances solved by optimality.

Our results were averaged for a group of 10 instances and were given in terms of the relative error with respect to the lower bound

$$L_2 = \max\left\{\left\lceil \frac{1}{m}\sum_{i=1,\ldots,n} p_i \right\rceil, p_1, p_m + p_{m+1}\right\},$$

where $p_1 \geqslant p_2 \geqslant \ldots \geqslant p_n$.

In Tables 1-6 we report the results of the five algorithms. For sake of simplicity, our algorithms are referred with the name of the used measure of spread. In Tables 7 and 8, the columns *LPT*, *3-PH*, *K-SPT* and *BEST* describe, respectively, the results of the *LPT* heuristic, of the algorithm of França et al., of the algorithm of Francioni et al. and of best result between our five variants. Since our results were averaged for groups of ten instances and the ten results can be from different variants, Tables 7 and 8 provide no information on the choice made. On the other hand, no superiority of a specific measure has been observed, so that we are unable to make a choice of the suitable spead measure before the algorithm starts.

The number of instances in which the algorithms obtain the makespan equal to the lower bound (which represent the number of instances solved to optimality) is reported in column opt.

*Computational Results*

Experiments with different measures of spread on these three family of instances have shown that quality of the solutions does not depend strongly on the chosen measure. Moreover, a choice of a particular measure does not lead a significant reduction of the run time. The BIN-PACK instances seem not suitable for a statistical methods approach; in fact, only in the case when the range is used, the obtained solutions were better than the ones provided by *LPT*. Therefore, we present only the results for the other two families of instances. The results for UNIFORM instances are shown in Table 1-3 and 7 for the three subsets of instances with processing times in $[1, 100]$, $[1, 1000]$, and $[1, 10000]$. UNIFORM instances are known to be efficiently approached with *LPT*, *3-PHASE* and *K-SPT* algorithms. *LPT* usually obtains low gaps, and solves to optimality a fair number of instances, while *3-PHASE* and *K-SPT* offer more accurate results than *LPT*. Our algorithms offer significant improvements over *LPT* (Table 7); in 15 (resp. in 23) out of 39 cases the average relative error of *BEST* is comparable with respect to the more accurate *3-PHASE* (resp. *K-SPT*). For example, when the processing times belong to $[1, 100]$, the average relative errors of *BEST* and *3-PHASE* are equal in 8 out of 13 cases and when the processing times belong to $[1, 10000]$, the average relative errors of *BEST* and *K-SPT* are equal in 8 out of 13 cases. The results for NONUNIFORM instances are shown in Table 4-6 and 8 for the three subsets of instances with processing times in $[1, 100]$, $[1, 1000]$, and $[1, 10000]$. These instances are more difficult than the UNIFORM instances, as greater gaps remain in all of the four algorithms examined. *BEST* consistently outperformed the *LPT* heuristic (Table 5), generating much smaller gaps. In addition, compared with the *3-PHASE* (resp. *K-SPT*) algorithm, *BEST* obtained smaller or equal gaps in 30 (resp. in 23) out of 39 cases.

| | | $\mathbf{p}_j \in [1, 100]$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **R** $(\cdot)$ | | | **G**$MD(\cdot)$ | | | $\sigma^2(\cdot)$ | | | **I**$QR(\cdot)$ | | | **M**$AD(\cdot)$ | | |
| **m** | **n** | **gap** | | **opt** | **gap** | | **opt** | **gap** | | **opt** | **gap** | | **opt** | **gap** | | **opt** |
| 5 | 10 | 3.76e-02 | | 3 | 3.76e-02 | | 3 | 3.76e-02 | | 3 | 7.99e-02 | | 3 | 1.01e-01 | | 2 |
| 5 | 50 | 1.38e-03 | | 3 | 1.81e-03 | | 5 | 9.99e-04 | | 5 | 1.13e-02 | | | 1.06e-02 | | |
| 5 | 100 | 1.05e-04 | | 9 | 3.02e-04 | | 7 | 2.02e-04 | | 8 | 1.75e-03 | | 3 | 3.19e-03 | | 1 |
| 5 | 500 | 0.00e+00 | | 10 | 0.00e+00 | | 10 | 0.00e+00 | | 10 | 1.21e-04 | | 5 | 2.02e-05 | | 9 |
| 5 | 1000 | 0.00e+00 | | 10 | 0.00e+00 | | 10 | 0.00e+00 | | 10 | 3.03e-05 | | 7 | 0.00e+00 | | 10 |
| 10 | 50 | 7.53e-03 | | 2 | 3.51e-02 | | | 8.41e-03 | | | 4.20e-02 | | | 4.20e-02 | | |
| 10 | 100 | 1.16e-03 | | 5 | 5.04e-03 | | 1 | 1.15e-03 | | 5 | 1.07e-02 | | | 1.07e-02 | | |
| 10 | 500 | 0.00e+00 | | 10 | 3.27e-04 | | 6 | 0.00e+00 | | 10 | 3.91e-04 | | 6 | 3.91e-04 | | 6 |
| 10 | 1000 | 0.00e+00 | | 10 | 1.43e-04 | | 6 | 0.00e+00 | | 10 | 0.00e+00 | | 10 | 0.00e+00 | | 10 |
| 25 | 50 | 3.35e-02 | | 2 | 2.10e-01 | | 1 | 1.74e-01 | | 1 | 2.15e-01 | | 1 | 2.15e-01 | | 1 |
| 25 | 100 | 2.40e-02 | | | 7.25e-02 | | | 4.19e-02 | | | 5.55e-02 | | | 5.55e-02 | | |
| 25 | 500 | 1.03e-04 | | 9 | 5.86e-03 | | 2 | 7.04e-04 | | 4 | 2.33e-03 | | | 2.33e-03 | | |
| 25 | 1000 | 0.00e+00 | | 10 | 1.57e-03 | | 2 | 3.04e-04 | | 6 | 3.55e-04 | | 4 | 3.55e-04 | | 4 |

Table 1 - Computational results for UNIFORM instances.

| | | $\mathbf{p}_j \in [1, 1000]$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **R** $(\cdot)$ | | | **G**$MD(\cdot)$ | | | $\sigma^2(\cdot)$ | | | **I**$QR(\cdot)$ | | | **M**$AD(\cdot)$ | | |
| **m** | **n** | **gap** | | **opt** | **gap** | | **opt** | **gap** | | **opt** | **gap** | | **opt** | **gap** | | **opt** |
| 5 | 10 | 4.39e-02 | | 2 | 5.74e-02 | | 2 | 4.39e-02 | | 2 | 1.17e-01 | | 1 | 1.43e-01 | | |
| 5 | 50 | 1.68e-03 | | | 2.79e-03 | | | 1.82e-03 | | | 5.81e-03 | | | 9.13e-03 | | |
| 5 | 100 | 2.48e-04 | | 1 | 6.11e-04 | | 1 | 3.70e-04 | | | 2.22e-03 | | | 2.47e-03 | | |
| 5 | 500 | 2.02e-06 | | 9 | 2.06e-06 | | 9 | 0.00e+00 | | 10 | 1.42e-05 | | 4 | 3.06e-05 | | 2 |
| 5 | 1000 | 0.00e+00 | | 10 | 0.00e+00 | | 10 | 0.00e+00 | | 10 | 0.00e+00 | | 10 | 1.39e-05 | | 3 |
| 10 | 50 | 1.67e-02 | | | 5.01e-02 | | | 1.41e-02 | | | 4.00e-02 | | | 4.00e-02 | | |
| 10 | 100 | 3.01e-03 | | | 1.45e-02 | | | 4.83e-03 | | | 1.14e-02 | | | 1.14e-02 | | |
| 10 | 500 | 5.33e-05 | | 4 | 3.48e-04 | | | 2.07e-05 | | 6 | 2.51e-04 | | | 2.51e-04 | | |
| 10 | 1000 | 5.94e-06 | | 7 | 1.61e-05 | | 4 | 1.99e-06 | | 9 | 5.21e-05 | | | 5.21e-05 | | |
| 25 | 50 | 2.46e-02 | | 5 | 1.65e-01 | | 2 | 1.76e-01 | | 2 | 1.78e-01 | | 2 | 1.78e-01 | | 2 |
| 25 | 100 | 3.01e-02 | | | 7.99e-02 | | | 6.74e-02 | | | 4.13e-02 | | | 4.13e-02 | | |
| 25 | 500 | 6.99e-04 | | | 6.68e-03 | | | 7.26e-04 | | | 2.39e-03 | | | 2.39e-03 | | |
| 25 | 1000 | 1.77e-04 | | 1 | 1.68e-03 | | | 1.96e-04 | | 1 | 1.02e-03 | | | 1.02e-03 | | |

Table 2 - Computational results for UNIFORM instances.

| | | $\mathbf{p}_j \in [1, 10000]$ | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **R** $(\cdot)$ | | **G**$MD(\cdot)$ | | $\sigma^2(\cdot)$ | | **I**$QR(\cdot)$ | | **M**$AD(\cdot)$ | |
| **m** | **n** | **gap** | **opt** | **gap** | **opt** | **gap** | **opt** | **gap** | **opt** | **gap** | **opt** |
| 5 | 10 | 6.78e-03 | 8 | 6.78e-03 | 8 | 6.78e-03 | 8 | 3.22e-02 | 6 | 4.31e-02 | 5 |
| 5 | 50 | 1.97e-03 | | 2.98e-03 | | 1.97e-03 | | 8.96e-03 | | 1.32e-02 | |
| 5 | 100 | 4.10e-04 | | 4.80e-04 | | 3.82e-04 | | 1.36e-03 | | 3.07e-03 | |
| 5 | 500 | 3.16e-06 | 3 | 2.96e-06 | 3 | 2.76e-06 | 2 | 2.74e-05 | | 7.06e-05 | |
| 5 | 1000 | 4.01e-07 | 6 | 2.98e-07 | 7 | 2.00e-07 | 8 | 1.50e-06 | 3 | 1.77e-05 | |
| 10 | 50 | 1.59e-02 | | 3.99e-02 | | 1.39e-02 | | 3.63e-02 | | 3.63e-02 | |
| 10 | 100 | 2.54e-03 | | 9.88e-03 | | 2.27e-03 | | 1.41e-02 | | 1.41e-02 | |
| 10 | 500 | 6.78e-05 | | 3.20e-04 | | 5.50e-05 | | 2.21e-04 | | 2.21e-04 | |
| 10 | 1000 | 4.61e-06 | 1 | 8.34e-05 | | 1.55e-05 | | 5.39e-05 | | 5.39e-05 | |
| 25 | 50 | 2.72e-02 | 2 | 1.94e-01 | 2 | 1.35e-01 | 2 | 1.98e-01 | 2 | 1.98e-01 | 2 |
| 25 | 100 | 2.25e-02 | | 9.89e-02 | | 4.57e-02 | | 7.22e-02 | | 7.22e-02 | |
| 25 | 500 | 6.60e-04 | | 1.27e-02 | | 1.04e-03 | | 2.35e-03 | | 2.35e-03 | |
| 25 | 1000 | 8.66e-05 | | 1.98e-03 | | 4.73e-04 | | 8.63e-04 | | 8.63e-04 | |

Table 3 - Computational results for UNIFORM instances.

| | | $\mathbf{p}_j \in [1, 100]$ | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **R** $(\cdot)$ | | **G**$MD(\cdot)$ | | $\sigma^2(\cdot)$ | | **I**$QR(\cdot)$ | | **M**$AD(\cdot)$ | |
| **m** | **n** | **gap** | **opt** | **gap** | **opt** | **gap** | **opt** | **gap** | **opt** | **gap** | **opt** |
| 5 | 10 | 7.52e-03 | 2 | 7.52e-03 | 2 | 7.52e-03 | 2 | 7.52e-03 | 2 | 7.52e-03 | 2 |
| 5 | 50 | 7.50e-03 | 0 | 7.50e-03 | 0 | 8.79e-03 | 0 | 7.50e-03 | 0 | 7.50e-03 | 0 |
| 5 | 100 | 6.75e-03 | 0 | 6.75e-03 | 0 | 4.07e-03 | 0 | 6.75e-03 | 0 | 6.75e-03 | 0 |
| 5 | 500 | 0.00e+00 | 10 | 0.00e+00 | 10 | 0.00e+00 | 10 | 1.06e-05 | 9 | 2.13e-05 | 8 |
| 5 | 1000 | 0.00e+00 | 10 | 0.00e+00 | 10 | 0.00e+00 | 10 | 5.32e-06 | 9 | 1.06e-05 | 8 |
| 10 | 50 | 1.44e-02 | 0 | 1.44e-02 | 0 | 1.61e-02 | 0 | 1.76e-02 | 0 | 1.44e-02 | 0 |
| 10 | 100 | 1.21e-02 | 0 | 1.21e-02 | 0 | 1.37e-02 | 0 | 1.72e-02 | 0 | 1.21e-02 | 0 |
| 10 | 500 | 4.43e-03 | 0 | 4.43e-03 | 0 | 1.49e-04 | 4 | 1.00e-02 | 0 | 4.43e-03 | 0 |
| 10 | 1000 | 0.00e+00 | 10 | 0.00e+00 | 10 | 0.00e+00 | 10 | 3.93e-04 | 1 | 0.00e+00 | 10 |
| 25 | 50 | 5.30e-03 | 4 | 5.30e-03 | 4 | 5.30e-03 | 4 | 5.82e-03 | 4 | 5.30e-03 | 4 |
| 25 | 100 | 1.55e-02 | 0 | 1.55e-02 | 0 | 1.61e-02 | 0 | 1.69e-02 | 0 | 1.58e-02 | 0 |
| 25 | 500 | 7.61e-03 | 0 | 7.61e-03 | 0 | 8.35e-03 | 0 | 9.79e-03 | 0 | 7.61e-03 | 0 |
| 25 | 1000 | 6.99e-03 | 0 | 6.99e-03 | 0 | 6.24e-03 | 0 | 9.30e-03 | 0 | 6.99e-03 | 0 |

Table 4 - Computational results for NONUNIFORM instances.

| | | $\mathbf{p}_j \in [1, 1000]$ | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **R** $(\cdot)$ | | **G**$MD(\cdot)$ | | $\sigma^2(\cdot)$ | | **I**$QR(\cdot)$ | | **M**$AD(\cdot)$ | |
| **m** | **n** | **gap** | **opt** | **gap** | **opt** | **gap** | **opt** | **gap** | **opt** | **gap** | **opt** |
| 5 | 10 | 6.13e-03 | 1 | 6.13e-03 | 1 | 6.13e-03 | 1 | 6.13e-03 | 1 | 6.13e-03 | 1 |
| 5 | 50 | 8.70e-03 | 0 | 8.69e-03 | 0 | 1.01e-02 | 0 | 8.68e-03 | 0 | 8.69e-03 | 0 |
| 5 | 100 | 7.71e-03 | 0 | 7.71e-03 | 0 | 5.48e-03 | 0 | 7.71e-03 | 0 | 7.71e-03 | 0 |
| 5 | 500 | 6.39e-06 | 6 | 5.33e-06 | 7 | 0.00e+00 | 10 | 0.00e+00 | 10 | 6.18e-05 | 2 |
| 5 | 1000 | 0.00e+00 | 10 | 0.00e+00 | 10 | 0.00e+00 | 10 | 1.59e-06 | 7 | 3.03e-05 | 2 |
| 10 | 50 | 1.49e-02 | 0 | 1.47e-02 | 0 | 1.66e-02 | 0 | 1.81e-02 | 0 | 1.47e-02 | 0 |
| 10 | 100 | 1.30e-02 | 0 | 1.29e-02 | 0 | 1.41e-02 | 0 | 1.78e-02 | 0 | 1.29e-02 | 0 |
| 10 | 500 | 4.80e-03 | 0 | 4.80e-03 | 0 | 3.00e-04 | 0 | 1.02e-02 | 0 | 4.80e-03 | 0 |
| 10 | 1000 | 4.25e-06 | 8 | 3.08e-05 | 1 | 0.00e+00 | 10 | 4.14e-04 | 0 | 5.10e-05 | 0 |
| 25 | 50 | 4.03e-03 | 4 | 4.03e-03 | 4 | 4.03e-03 | 4 | 4.34e-03 | 2 | 4.03e-03 | 4 |
| 25 | 100 | 1.71e-02 | 0 | 1.67e-02 | 0 | 1.74e-02 | 0 | 1.80e-02 | 0 | 1.66e-02 | 0 |
| 25 | 500 | 7.78e-03 | 0 | 7.77e-03 | 0 | 8.64e-03 | 0 | 1.00e-02 | 0 | 7.77e-03 | 0 |
| 25 | 1000 | 7.00e-03 | 0 | 7.00e-03 | 0 | 6.34e-03 | 0 | 9.38e-03 | 0 | 7.00e-03 | 0 |

Table 5 - Computational results for NONUNIFORM instances.

| | | $\mathbf{p}_j \in [1, 10000]$ | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **R** $(\cdot)$ | | **G**$MD(\cdot)$ | | $\sigma^2(\cdot)$ | | **I**$QR(\cdot)$ | | **M**$AD(\cdot)$ | |
| **m** | **n** | **gap** | **opt** | **gap** | **opt** | **gap** | **opt** | **gap** | **opt** | **gap** | **opt** |
| 5 | 10 | 6.17e-03 | 1 | 6.17e-03 | 1 | 6.17e-03 | 1 | 6.17e-03 | 1 | 6.17e-03 | 1 |
| 5 | 50 | 8.85e-03 | 0 | 8.85e-03 | 0 | 1.02e-02 | 0 | 8.83e-03 | 0 | 8.83e-03 | 0 |
| 5 | 100 | 7.81e-03 | 0 | 7.82e-03 | 0 | 5.73e-03 | 0 | 7.81e-03 | 0 | 7.81e-03 | 0 |
| 5 | 500 | 3.82e-05 | 1 | 3.16e-05 | 2 | 1.06e-07 | 9 | 1.38e-06 | 4 | 6.70e-05 | 1 |
| 5 | 1000 | 8.61e-06 | 5 | 4.95e-06 | 5 | 0.00e+00 | 10 | 3.03e-06 | 5 | 3.56e-05 | 2 |
| 10 | 50 | 1.50e-02 | 0 | 1.48e-02 | 0 | 1.67e-02 | 0 | 1.81e-02 | 0 | 1.48e-02 | 0 |
| 10 | 100 | 1.31e-02 | 0 | 1.30e-02 | 0 | 1.42e-02 | 0 | 1.78e-02 | 0 | 1.30e-02 | 0 |
| 10 | 500 | 4.87e-03 | 0 | 4.87e-03 | 0 | 2.93e-04 | 0 | 1.02e-02 | 0 | 4.87e-03 | 0 |
| 10 | 1000 | 1.01e-05 | 1 | 3.21e-05 | 1 | 0.00e+00 | 10 | 4.14e-04 | 0 | 4.75e-05 | 0 |
| 25 | 50 | 4.09e-03 | 3 | 4.09e-03 | 3 | 4.09e-03 | 3 | 4.31e-03 | 2 | 4.09e-03 | 3 |
| 25 | 100 | 1.72e-02 | 0 | 1.67e-02 | 0 | 1.75e-02 | 0 | 1.81e-02 | 0 | 1.67e-02 | 0 |
| 25 | 500 | 7.82e-03 | 0 | 7.82e-03 | 0 | 8.69e-03 | 0 | 1.01e-02 | 0 | 7.82e-03 | 0 |
| 25 | 1000 | 7.04e-03 | 0 | 7.04e-03 | 0 | 6.38e-03 | 0 | 9.40e-03 | 0 | 7.04e-03 | 0 |

Table 6 - Computational results for NONUNIFORM instances.

| | | $p_j \in [1, 100]$ | | | | | | | | $p_j \in [1, 1000]$ | | | | | | | | $p_j \in [1, 10000]$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | LPT | | 3-PH | | K-SPT | BEST | | | LPT | | 3-PH | | K-SPT | BEST | | | LPT | | 3-PH | | K-SPT | BEST | | |
| m | n | gap | o | gap | o | gap | gap | o | | gap | o | gap | o | gap | gap | o | | gap | o | gap | o | gap | gap | o | |
| 5 | 10 | 0.00e-00 | 10 | 0.00e-00 | 10 | 0.00e-00 | 0.00e-00 | 10 | | 0.00e-00 | 10 | 0.00e-00 | 10 | 0.00e-00 | 0.00e-00 | 10 | | 0.00e-00 | 10 | 0.00e-00 | 10 | 0.00e-00 | 0.00e-00 | 10 | |
| | 50 | 4.23e-03 | | 1.96e-04 | | 1.96e-04 | 6.18e-04 | 7 | | 4.27e-03 | | 2.48e-04 | | 2.28e-04 | 1.15e-03 | | | 5.44e-03 | | 5.44e-04 | | 2.60e-04 | 1.17e-03 | | |
| | 100 | 1.46e-03 | 3 | 0.00e-00 | 3 | 0.00e-00 | 1.05e-04 | 9 | | 1.22e-03 | | 6.03e-05 | | 7.16e-05 | 1.60e-04 | 2 | | 9.84e-04 | | 7.78e-05 | | 4.19e-05 | 2.19e-04 | | |
| | 500 | 0.00e-00 | 10 | 0.00e-00 | 10 | 0.00e-00 | 0.00e-00 | 10 | | 2.05e-05 | 4 | 0.00e-00 | 4 | 0.00e-00 | 0.00e-00 | 10 | | 4.49e-05 | | 3.96e-07 | | 3.96e-06 | 1.38e-06 | 4 | |
| | 1000 | 0.00e-00 | 10 | 0.00e-00 | 10 | 0.00e-00 | 0.00e-00 | 10 | | 6.00e-06 | 4 | 0.00e-00 | 4 | 0.00e-00 | 0.00e-00 | 10 | | 1.03e-05 | | 0.00e-00 | | 1.40e-06 | 9.88e-08 | 9 | |
| 10 | 50 | 1.60e-02 | 1 | 2.35e-03 | 1 | 1.86e-03 | 6.33e-03 | 2 | | 2.80e-02 | | 4.43e-03 | | 4.30e-03 | 1.01e-02 | | | 2.07e-02 | | 4.81e-03 | | 2.84e-03 | 9.82e-03 | | |
| | 100 | 4.93e-03 | 2 | 0.00e-00 | 2 | 5.97e-04 | 5.71e-04 | 8 | | 4.76e-03 | | 3.43e-04 | | 7.27e-04 | 2.67e-03 | | | 5.63e-03 | | 2.48e-04 | | 5.77e-04 | 1.59e-03 | | |
| | 500 | 4.10e-05 | 9 | 0.00e-00 | 9 | 0.00e-00 | 0.00e-00 | 10 | | 1.61e-04 | | 0.00e-00 | | 2.87e-05 | 1.23e-05 | 8 | | 1.59e-04 | | 2.38e-06 | | 2.57e-05 | 3.58e-05 | | |
| | 1000 | 0.00e-00 | 10 | 0.00e-00 | 10 | 0.00e-00 | 0.00e-00 | 10 | | 4.01e-05 | | 0.00e-00 | | 1.02e-05 | 1.99e-06 | 9 | | 4.81e-05 | | 0.00e-00 | | 8.81e-06 | 4.41e-06 | 1 | |
| 25 | 50 | 0.00e-00 | 10 | 0.00e-00 | 10 | 0.00e-00 | 0.00e-00 | 10 | | 0.00e-00 | 10 | 0.00e-00 | 10 | 0.00e-00 | 0.00e-00 | 10 | | 0.00e-00 | 10 | 0.00e-00 | 10 | 0.00e-00 | 0.00e-00 | 10 | |
| | 100 | 2.52e-02 | | 3.62e-03 | | 1.15e-02 | 2.08e-02 | | | 2.23e-02 | | 4.04e-03 | | 8.82e-03 | 2.61e-02 | | | 3.26e-02 | | 4.33e-03 | | 6.67e-03 | 1.99e-02 | | |
| | 500 | 5.04e-04 | 5 | 0.00e-00 | 5 | 4.07e-04 | 0.00e-00 | 10 | | 1.06e-03 | | 0.00e-00 | | 3.41e-04 | 5.03e-04 | | | 1.31e-03 | | 2.28e-05 | | 2.06e-04 | 4.26e-04 | | |
| | 1000 | 1.01e-04 | 8 | 0.00e-00 | 8 | 0.00e-00 | 0.00e-00 | 10 | | 2.81e-04 | | 0.00e-00 | | 7.53e-05 | 1.06e-04 | 1 | | 3.55e-04 | | 3.52e-06 | | 6.10e-05 | 7.89e-05 | | |

Table 7 - Computational results for UNIFORM instances

| | | $p_j \in [1, 100]$ | | | | | | | | $p_j \in [1, 1000]$ | | | | | | | | $p_j \in [1, 10000]$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | LPT | | 3-PH | | K-SPT | BEST | | | LPT | | 3-PH | | K-SPT | BEST | | | LPT | | 3-PH | | K-SPT | BEST | | |
| m | n | gap | o | gap | o | gap | gap | o | | gap | o | gap | o | gap | gap | o | | gap | o | gap | o | gap | gap | o | |
| 5 | 10 | 0.00e-00 | 10 | 0.00e-00 | 10 | 0.00e-00 | 0.00e-00 | 10 | | 0.00e-00 | 10 | 0.00e-00 | 10 | 0.00e-00 | 0.00e-00 | 10 | | 0.00e-00 | 10 | 0.00e-00 | 10 | 0.00e-00 | 0.00e-00 | 10 | |
| | 50 | 1.69e-02 | | 1.54e-02 | | 8.80e-03 | 7.50e-03 | | | 1.75e-02 | | 1.59e-02 | | 1.01e-02 | 8.62e-03 | | | 1.76e-02 | | 1.58e-02 | | 1.03e-02 | 8.78e-03 | | |
| | 100 | 1.67e-02 | | 6.05e-03 | | 2.12e-04 | 4.07e-03 | | | 1.70e-02 | | 6.52e-03 | | 3.71e-05 | 5.48e-03 | | | 1.70e-02 | | 6.64e-03 | | 4.87e-05 | 5.73e-03 | | |
| | 500 | 5.86e-04 | | 2.03e-03 | | 0.00e-00 | 0.00e-00 | 10 | | 6.35e-04 | | 0.00e-00 | | 4.24e-06 | 0.00e-00 | 10 | | 6.42e-04 | | 0.00e-00 | | 1.27e-06 | 0.00e-00 | 10 | |
| | 1000 | 1.44e-04 | | 1.25e-03 | | 0.00e-00 | 0.00e-00 | 10 | | 1.78e-04 | | 1.88e-04 | | 3.18e-06 | 0.00e-00 | 10 | | 1.78e-04 | | 0.00e-00 | | 1.59e-07 | 0.00e-00 | 10 | |
| 10 | 50 | 1.76e-02 | | 1.42e-02 | | 1.61e-02 | 1.44e-02 | | | 1.82e-02 | | 1.42e-02 | | 1.61e-02 | 1.47e-02 | | | 1.82e-02 | | 1.42e-02 | | 1.63e-02 | 1.48e-02 | | |
| | 100 | 1.72e-02 | | 7.31e-03 | | 6.25e-03 | 1.21e-02 | | | 1.77e-02 | | 7.18e-03 | | 5.14e-03 | 1.29e-02 | | | 1.78e-02 | | 7.25e-03 | | 5.24e-03 | 1.29e-02 | | |
| | 500 | 1.00e-02 | | 6.63e-03 | | 1.89e-03 | 1.49e-04 | | | 1.01e-02 | | 6.46e-03 | | 6.75e-05 | 3.00e-04 | | | 1.02e-02 | | 6.65e-03 | | 6.78e-03 | 2.93e-04 | | |
| | 1000 | 3.83e-04 | 1 | 2.74e-03 | 1 | 6.36e-05 | 0.00e-00 | 10 | | 4.12e-04 | | 4.53e-04 | | 5.74e-05 | 0.00e-00 | 10 | | 4.10e-04 | | 0.00e-00 | | 2.01e-05 | 0.00e-00 | 10 | |
| 25 | 50 | 0.00e-00 | 10 | 0.00e-00 | 10 | 0.00e-00 | 0.00e-00 | 10 | | 0.00e-00 | 10 | 0.00e-00 | 10 | 0.00e-00 | 0.00e-00 | 10 | | 0.00e-00 | 10 | 0.00e-00 | 10 | 0.00e-00 | 0.00e-00 | 10 | |
| | 100 | 1.77e-02 | | 7.44e-03 | | 9.30e-03 | 1.53e-02 | | | 1.81e-02 | | 7.50e-03 | | 8.88e-03 | 1.65e-02 | | | 1.81e-02 | | 7.58e-03 | | 8.83e-03 | 1.66e-02 | | |
| | 500 | 9.79e-03 | | 6.43e-03 | | 3.66e-03 | 7.61e-03 | | | 1.00e-02 | | 6.59e-03 | | 8.49e-04 | 7.77e-03 | | | 1.01e-02 | | 6.58e-03 | | 3.86e-04 | 7.81e-03 | | |
| | 1000 | 9.30e-03 | | 6.16e-03 | | 5.04e-04 | 6.24e-03 | | | 9.38e-03 | | 6.18e-03 | | 4.03e-04 | 6.34e-03 | | | 9.40e-03 | | 6.20e-03 | | 1.88e-04 | 6.38e-03 | | |

Table 8 - Computational results for NONUNIFORM instances

## 4 Conclusions

In this paper, we proposed $n \log n$ algorithms for solving parallel machine scheduling problem to minimize the makespan. These algorithms produce less average relative errors than the *LPT* algorithm, and, usually, the average relative errors are comparable with those generated by the *3-PHASE* and *K-SPT* improvement heuristics. Our algorithms are based on five different measures of spread, that are techniques commonly employed in statistical problems. While the use of each measure possesses some degree of success, no overwhelming superiority of a single measure has been determined.

## References


[1] Anderson E.J., Glass C.A., Potts C.N., "Machine scheduling", *Local Search in Combinatorial Optimization*, Aarts and Lenstra eds., Wiley, New York, pp. 361-414, 1997.

[2] Chen B., Potts C.N., Woeginger G.J., "A review of machine scheduling: Complexity, algorithms and approximability," *Handbook of Combinatorial Optimization*, Du D.Z. and Pardalos P. eds., Kluwer Academic Publishers, Dordrecht, pp. 21-169, 3/98.

[3] Cheng T., Sin C., "A state of the art review of parallel machine scheduling research," *European J. of Oper. Res.*, pp. 271-292, 47/90.

[4] Coffman E.G. Jr., Garey M.R., Johnson D.S., "An application of bin-packing to multiprocessor scheduling," *SIAM J. Comput.*, pp. 1-17, 7/78.

[5] França P.M., Gendrau M., Laporte G., Muller F.M., "A composite heuristic for the identical parallel machine scheduling problem with minimum makespan objective," *Computers Ops. Res.*, pp. 205-210, 21(2)/94.

[6] Frangioni A., Scutellà M.G., Necciari E., "Multi-exchange algorithms for the minimum makespan machine," *Technical Report:99-22*, Dipartimento di Matematica, University of Pisa, Italy, 1999.

[7] Frangioni A., Necciari E., Scutellà M.G., "A multi-exchange neighborhood for minimum makespan machine scheduling problems," *J. Comb. Optim.*, pp. 195-220, 8/04.

[8] Garey M.R., Johnson D.S., *Computers and Intractability: A guide to the Theory of NP-Completeness*, Freeman, San Francisco, CA, 1979.

[9] Graham R.L., "Bounds for certain multiprocessing anomalies," *Bell System Tech. J.*, pp. 1563-1581, 45/66.

[10] Graham R.L., "Bounds on multiprocessing timing anomalies," *SIAM J. Appl. Math.*, pp. 416-429, 17/69.

[11] Graham R.L., Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G., "Optimization and approximation in deterministic sequencing and scheduling. A survey," *Ann. Discrete Math.*, pp. 287-326, 5/79.

[12] Gualtieri M.I., Paletta G., Pietramala P., "A new $n \log n$ algorithm for the identical parallel machine scheduling problem," *Int. J. Contemp. Math. Sciences*, pp. 25-36, 3(1)/08.

[13] Hochbaum D.S., Shmoys D.B., "Using dual approximation algorithms for scheduling problems: theoretical and practical results," *J. Assoc. Comput. Mach.*, pp. 144–162, 34/87.

[14] Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G., Shmoys D.B., "Sequencing and scheduling: algorithms and complexity in logistics of production and inventory," *Handbooks in Operation Research and Management Science*, Elsevier Science Publishers, pp. 445-522, 4/93.

[15] Leung J. (Editor), *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, Chapman & Hall/CRC, 2004.

[16] Paletta G., Pietramala P., "A new approximation algorithm for the non-preemptive scheduling of independent jobs on identical parallel processors," *SIAM J. Discrete Math.*, pp. 313-328, 21/07.

[17] Ullman J.D., "Complexity of sequencing problems," *Computer and Job Shop Scheduling Theory*, E.G. Coffman (ed.), Wiley, New York, pp. 139-164, 1976.