IJMAO

International Journal of Mathematical Analysis and Optimization: Theory and Applications
Vol. 2020, No. 1, pp. 741 - 756

# Hybrid Metaheuristic for Just In Time Scheduling in a Flow Shop with Distinct Time Windows

Idowu, A. Gbolahan [1*] , Adamu, O. Muminu [2], Sawyerr, A. Babatunde [3]

1, Department of Mathematics Lagos State University, Lagos, Nigeria.
2, Department of Mathematics University of Lagos, Nigeria.
3, Department of Computer Sciences, University of Lagos, Nigeria.
* Corresponding author's email: gbolahan.idowu@lasu.edu.ng

**Article Info**
Received: 19 December 2019 Revised: 21 January 2020
Accepted: 08 April 2020 Available online: 19 July 2020

**Abstract**

Scheduling to meet clients' order has been a challenge in recent times therefore researches in Just-in-Time (JIT) scheduling has evolved. JIT scheduling involves reducing waste, inventory costs and making goods available as at when needed. This work addresses the JIT scheduling problem on flow shop where jobs incur penalties if they are not completed within their specific due windows. The problem is to obtain an execution sequence that will minimize the bi-criteria earliness-tardiness objective. JIT scheduling problem is prevalent in real life manufacturing applications. An hybrid of Tabu-Search and Variable Neighborhood Search (HTVNS) algorithm is proposed for the problem. To justify this hybridization a benchmark of 12500 problem instances were solved and the result obtained is compared with an algorithm in the literature. The result shows that the proposed hybrid metahueristic algorithm performs considerable better.

**Keywords And Phrases:** Earliness/Tardiness, Just-In-Time, Tabu-Search, Variable Neigbhourhood
**MSC2010: 90C27, 90C29**

## 1 Introduction

Manufacturing sector recently has shifted focus from bulk production for a single client to many clients spread all over the world. This type of manufacturing is made possible by advances in information technology where clients can order products across continents. To remain competitive, many manufacturers find themselves planning their daily production based on customers' request. Their production plaining objective is to meet different delievery dates of each customer notwithstanding the sizes of their order or delivery destinations. Meeting delivery date(s) of customer(s) translate to minimizing earliness and tardiness among the orders.

Just-In-Time (JIT) production philosophy takes into consideration of both earliness and tardiness (ET) penalties as the optimality criteria. In JIT scheduling environment, job(s) that complete early must be stored inventory until their due date(s), while job(s) that complete after their due date(s) may erode customer's confidence. Therefore, an appropriate schedule is one in which all jobs finish exactly on their assigned due-date(s). According to [1], JIT scheduling would reduce waste, inventory costs and make goods available as at when required. This concept of penalizing both earliness and tardiness was introduced in [2] and [3] which has motivated a rapidly developing trend in the field of scheduling research. The use of this combined earliness/tardiness penalties contributed to non-regular criteria and has led to new methodological issues in the design of solution procedures. The most frequently employed objective function sums the penalties due to earliness or tardiness, and the resulting objective is usually referred to as the Multi-Objective ET Scheduling Problem (MOETSP) presented in [4]. Tardiness refers to the length an order is belated while earliness is the length in which an order remains in inventory before customer's pickup or delivery.

In a typical industry, some production/assembly lines are similar to flowshop where each job undergoes predefined sequence of operations with known processing time for each operation(stage). Flowshop can be found in industries with conveyors between machines for material transfer and assembly lines performing the final assembly of bulk products as described in [5, 6, 7]. This type of factory encompasses manufacturing of cars, electronics, airplanes, food processing, clothing, packaging, heavy machinery and branding, including many service industries [8, 9].

In [10], researches spanning more than five decades of flowshop scheduling problems with due dates and its different approximate and optimal solution approach were reviewed. Many of these literature for flowshop shceduling problem are concerned with due dates related criteria. However, a single due date consideration is unrealistic and impractical in real world production planning problems. According to [11] and [12], it is often expected that jobs are completed within a certain time interval (due window) rather than at a single point in time (due date). Due window(s) constraint on job(s) is a generalization of the due date(s) concept and defines a time interval in which a job must be completed. If a job is completed within its due window, then it is considered to be on-time and no penalty will be incurred. Otherwise a penalty due to the job earliness or tardiness can be imposed, depending on whether the job was completed before or after the due window, respectively.

Variety of real life applications of due window models of JIT production scheduling can be found in manufaturing [13], semi-conductor fabrications [14], vaccine production, production of trendy, perishable and industrial products where shelf life is crucial [15]. Some research works have taken into consideration due window scheduling. [16] studied two-machines flow shop problem . They prove the problem to be NP-hard in the strong sense. A branch and bound algorithm and a heuristic were developed as a technique to tackle the problem. Also, four special cases were shown to be pseudo-polynomially solvable and NP-hard in the ordinary sense. [17] considered the problem of minimization of the maximum of the costs associated with job earliness, tardiness and due window allocation and size. They proved that for any given job sequence the optimal due window size and allocation can be obtained in polynomial time by solving appropriate linear programming model. [17] also studied other minimum-maximum due window assignment problem in which the objective is to schedule the jobs and assign due windows such that the highest cost among all job is minimized. The objective function consists of four costs: earliness, tardiness, earliest due dates and due window size. An $O(n^2 logn)$ time algorithm was proposed to solve the problem.

[18] solved approximately, the Weighted ET (WET) m-machine flow shop with a Simulated Annealing (SA) based heuristic where a job is moved forward or backward into a sequence if it has a higher priority index than its right or left adjacent neighbour. Computation of the priority index was simplified by basing it on an estimate of the job's earliest starting time on the last machine. Their approach was experimented on test instances with up to 30 jobs and 20 machines. The large number of machines used is important because it probes the quality of estimation of the priority index. [19] proposed a Genetic Algorithm (GA) for the ET m-machine permutation flow shop. The performance of their GA was compared with existing heuristics that were designed for related flow shop and single machine problems. [20] proposed three variety of hybrids of SA and Tabu Search (TS) for the hybrid flow shop problem with release date where the objective is

to minimise sequence dependent set-up and waiting times between successive jobs. [21] proposed an Ant Colony Optimization (ACO) heuristic for the ET hybrid flowshop problem. An hybrid flow shop with limited waiting and set-up dependent times was considered in [30]. They minimize the sum of earliness and squared tardiness using a colonial-based competitive algorithm. [22] presented a modified branch and bound algorithm whose tree is reduced by considering some dominance properties to minimize the sum of maximum ET. [23] design a GA for WET in a flow shop with random key generation where due dates and processing times are fuzzy.

This paper proposed an hybrid of Variable Neighbourhood Search (VNS) and Tabu-Search (TS) algorithms which are stochastic search techniques and have capacity to solve large problem instances of MOETSP. VNS and TS have been successfully applied in [24] and [25] to large combinatorial optimization problems, respectively. TS uses information about its search history to guide local search approaches in order to overcome local optimality [26]. Generally, this is achieved by a dynamic transformation of the local neighborhood, which can lead to performing deteriorating moves when all improving moves of the current neighborhood are set to tabu. The various TS strategies differ especially in the way the tabu criteria are defined, taking into consideration the information about the search history. For a given starting solution $X$ and a generic tabu criterion, which is represented by a component TabuList (Tabu Memory). A neighbour, or a corresponding move, is called admissible, if it is not tabu or if an aspiration criterion is fulfilled. The aspiration criterion may override a possibly inappropriate tabu status. VNS is proposed by [27] as a local search algorithm based on the exploration of a single neigborhood structure. This method is based on the systematic or probabilistic change of the neighborhood during the search process. VNS successes are due to its simplicity and few parameters [28]. Specifically, its philosophy derived from three the following simple principles: (1), A local minimum with respect to one neighborhood is not necessarily a local minimum for another. (2), a global minimum is a local minimum for all neighborhood structures. (3), local minima with respect to one or more neighborhoods are relatively close to each other. These three principles make the investigation of the solution space more systematic than its counterpart for other meta heuristics while providing pertinent information that can be exploited in the design of context-dependent heuristics [28]. Application of VNS to ET in JIT scheduling problems are gaining attention recently in the literature with the works of [29, 30, 19, 31, 24].

The rest of this paper is organized as follows: definition of the problem is presented in section 2. Section 3 discusses the proposed hybrid algorithm. Section 4 showcases result of the algorithm and compared it with VNS-MIP proposed in [31]. Finally, conclusions are stated in section 5

## 2 Problem Description

Considering the problem of scheduling $n$ jobs $(J_1, J_2, \dots, J_n)$ on $m$ (ordered) machines $(M_1, M_2, ..., M_m)$ to minimize the earliness/tardiness objectives. It is assumed that a machine can process at most one job at a time and all jobs are ready for processing at time $t = 0$. Each job $j$, $j = 1, ..., n$, is characterized by a set of $m$ processing times $p_{ij}$, $i = 1, ..., m$ (that is, there are different processing times on each machine for a given job). For any given non - preemptive feasible schedule $\pi$ to be a sequence of permutation of jobs $(\pi = \pi(1), \pi(2), \dots, \pi(n))$ which represents the order the jobs are to be processed on all machines. Let $C_{ij}$ be completion time of job $j$ on the $i$th machine, $a_j$ be the beginning of the due window (earliest due date), $d_j$ the end of the due window (latest due date) of job $j$ and $D_j = |d_j - a_j|$ be the size of the due window. Then, job $j$ is said to be early if $C_{ij} < a_j$, tardy if $C_{ij} > d_j$, and on time if $a_j \leq C_{ij} \leq d_j$. Let $\theta_j > 0$ and $\vartheta_j > 0$ be the penalties (weights) for scheduling job $j$ early and late respectively.

Adopting the classification of [32], the general JIT on flow shop for minimizing earliness/tardiness is:

$$Fm|a_j, d_j| \sum_{j=1}^{n} (\theta_j E_j + \vartheta_j T_j) \tag{2.1}$$

where $Ej = \max\{a_j \check{} C_{mj}, 0\}$ and $T_j = \max\{C_{mj} \check{} d_j, 0\}$

In a special case where unit penalties are incurred on both early and tardy jobs ($\theta_j = \vartheta_j = 1$). This special case is referred to as unweighted and it is represented as $Fm|a_j, d_j| \sum_{j=1}^{n}(E_j + T_j)$ for minimizing sum of earliness/tardiness and the objective function is $Z = \sum_{j=1}^{n}(E_j + T_j)$.

## 3 Proposed Algorithm

### 3.1 Solution Representation

Generally, a solution for MOETSP in a flow shop with $n$ jobs is represented by a sequence $\pi_i$ of length $n$. Each index $i = 1, 2, \ldots, n$ depicts the job to be executed at position $i$ of $\pi$. For instance, in the sequence $\pi = (3, 5, 6, 7, 1, 4, 2)$ of 7 jobs, job 3 is the first to be executed and job 2 is the last to be executed on each of the machines.

### 3.2 Construction of Initial Solution

In non-population based heuristics, selection technique of the initial candidate solution is very important because it can influence the effectiveness of an algorithm to reach the best solution for a given stopping criterion, (see [31]).

The initial solution of the permutation $\pi$ for MOETSP in a flow shop can be constructed by applying the Earliest Due Date dispatching (EDD) or probabilistic (RND) rule. Both EDD and RND rules are explained below:

EDD - This is a greedy constructive dispatching rule, often applied in the literature to scheduling problems with distinct due dates, as employed in [33, 31, 34]. It begins with an empty sequence. The jobs with the earliest due dates of its time window of those not yet sequenced is inserted at the end of the current sub-sequence, in iteration. Ties are broken arbitrarily if jobs have equal due dates. The construction procedure stops when there are no more jobs outside of the execution sequence.

RND - The sequence with least value of the objective function $\phi(\pi)$ out of twenty randomly generated permutations of the n jobs. This rule can only be applied to problems where $n \geq 4$ jobs. It is pertinent to note that the least number of jobs considered in this work is $n = 10$.

### 3.3 Evaluation of A Candidate Solution

For any feasible job sequence ($\pi$), we can produce a solution to MOETSP in a flow shop by means of computation of Boolean matrix $B^T = \{y_j : c_{m,\pi_j} < a_j \; or \; c_{m,\pi_j} > d_j\}$ and Integral matrix $D^T = \{x_j : \max(0, c_{m,\pi_j} - d_j) + \max(0, a_j - c_{m,\pi_j})\}$.

For both matrices $B^T$ and $D^T$, the completion times matrix ($c_{i,\pi_j}$) of the $j - th$ job in the sequence $\pi$ on the $ith$ machine can be computed as follows:

$$c_{i,\pi_j}(\pi) = c_{i,\pi_{j-1}}(\pi) + p_{i,\pi_j} \; i = 1, \ldots, m, \; j = 1, \ldots, n \tag{3.2}$$

where

$$c_{i,\pi_0}(\pi) = 0 \; \forall \; i = 1, \ldots, m \tag{3.3}$$

The feasible solution to MOETSP ($\phi$) objective is the sum of the values in matrix obtained by $(B^T)D$ that is, $\phi = \sum_{j=1}^{n} B^T D$. For example, if $B^T = (1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1)$ and $D^T = (4 \quad 5 \quad 6 \quad 3 \quad 3 \quad 2 \quad 1)$ and for the sequence $\pi = \{3, 5, 6, 7, 1, 4, 2\}$ of 7 jobs; jobs 3, 1 and 2 finished early/tardy therefore $\phi(\pi) = \sum_{j=1}^{7} B^T D = 8$.

## 3.4 Neighbourhood of A Solution

Exploring the solution space, two kinds of permutations are considered:

1. Shift reallocation;
2. Pairwise interchange;

These permutations state the neighbourhoods $\pi_1$, and $\pi_2$ used in algorithm 3.5, respectively, and they are described in details below.

### Neighbourhood $\pi_1$

In a sequence of n jobs, each one is reallocated to $n-1$ positions. A job in position i may be reallocated to position $i+1$ or in reverse order. Thus, there are possible $(n-1)^2$ distinct neighbors in $\pi_1$. A example of a neighbor of $\pi = \{2, 5, 4, 7, 6, 1, 3\}$ in the neighborhood of $\pi_1$ is $\pi\prime = \{2, 5, 4, 7, 6, 3, 1\}$.

### Neighbourhood $\pi_2$

For a given solution $\pi$ of n jobs, the position of each job is swapped with that of the remaining $n-1$ jobs. However, swapping the position of kth job in the sequence with that of the jth job is correspondent to swapping the position of jth job with that of the kth job. Thus, there are $\frac{n(n-1)}{2}$ possible different neighbors in respect to $\pi$. For instance, a neighbor of $\pi = \{2, 5, 4, 7, 6, 1, 3\}$ in the neighborhood of $\pi_2$ is $\pi' = \{2, 5, 4, 1, 6, 7, 3\}$. $\pi'$ is obtained from $\pi$ by swapping the execution order of jobs 1 and 7.

## 3.5 Proposed Hybrid Tabu Variable Neighbourhood Search (HTVNS)for Minimizing Earliness/Tardiness

This section presents a hybrid of Variable Neighbourhood search (VNS) and Tabu Search (TS) metaheuristics proposed in this study to minimize ET objective in JIT scheduling with distinct due windows. Both VNS and TS are known in the literature to have been applied to solve general flow shop scheduling problems and have performed well. To the best of the knowledge of this researcher, the hybrid of both algorithms are yet to be applied specifically to this problem therefore an arduous effort is taken to contribute this likely combination to the body of knowledge. The pseudocode of this hybrid is highlighted in algorithm below. Its implementation has a rudimentary short-term memory (Tsize) and contains no intermediate or long-term memory structures.

The term $f(X')$ refers to an evaluation of candidate solution, as expressed in an objective function for mathematical programming model. *TMax* corresponds to the maximum number of iterations allowed without an improvement of the best known solution. It also, set the stopping criterion of the algorithm.

## 3.6 Variable Neighbourhood Decent Search (VND)

The local search VNS algorithm is performed using the Variable Neighbourhood Descent (VND) procedure in [28]. In this research, VND utilizes two sequence of local searches, the first based on random decent using neighborhood $\pi_1$ and second on random decent using neighborhood $\pi_2$.

In the first local search, two jobs are randomly selected, and their positions in the sequence are swapped. If the new sequence gives an improved solution, then it is accepted and becomes the current solution. Otherwise, another random swap in the current solution is evaluated. This search ends when VNMax successive swaps occur without an improvement in the current solution. When the first local search finishes,

**Algorithm 1:** Detail of Hybrid Tabu-Search and Variable Neighbourhood Search

```
Input: $(I, n, \pi_1, \pi_2, TMax, Tsize, VNMax)$
Output: $f^*, X'$

$X = InitialSolution(I, n)$ ;          // Generate as described in Section 3.2
$f^* = f(X)$ ;                          // Evaluate as described Section 3.3
$X' = X$;
$t = 0$;
$tabuList = \emptyset$;
$tabuListSize = 0$;
while $t < TMax$ do
    $t = t + 1$;
    $k = 1$;
    $X'' = null$;
    while $k \leq 2$ do
        Generate Neighbours $\pi_i \in \pi(X)$ ;       // As described in Section 3.4
        if $(\pi_i \notin tabuList)$ then
            $X'' = VND(X', f, \pi_1, \pi_2, VNMax)$;   // As Described in Section 3.6
            if $f(X'') < f(X')$ then
                $tabulist \cup X'$;
                $tabulistsize = tabulistsize + 1$;
                $X' = X''$;
                $f^* = f(X')$;
                $k = 1$;
            else
                $k = k + 1$;
            end
        end
        if $(tabulistsize > Tsize)$ then
            $tabulist \setminus X'$;
        end
    end
end
return $f^*, X'$;
```

the next local search is applied.

In the second local search, a job of the sequence and a new position for this job are randomly selected. If the new sequence gives an improve solution, then it becomes the current solution, and VND switches back to the first local search. Otherwise, another random reallocation in the current solution is evaluated. This search is interrupted when VNMax successive reallocation occur without improvement in the current solution. In this case, VND is stopped, and the best solution is returned.

In order to prevent returning to previously analyzed neighbors and to reduce computational cost, all previous moves are avoided by placing them in a tabu list. In addition, the order of selection for the exploration of neighborhoods explores increasingly far distant neighborhood of the current incumbent solution in terms of computational complexity, as proposed by [35].

## 4 Computational Experiment And Results

The proposed algorithm HTVNS presented in Sections 3.5 and VNS-MIP in [31] were implemented using java programing language with Netbeans 8.2 as Integrated Development Environment (IDE). The experiment is performed on a computer with $Intel^{®}$ $Core^{TM}$ 2 Duo CPU P8800 running at 2.66Ghz of processor speed, 4GB of RAM, and the $Windows^{®}$ 10pro 64-bit operating system. This low grade computer was adopted to observe the worst-case experimental characteristics of the method.

The instances/test-data used to assess HTVNS and VNS-MIP are described in Section 4.1. In section 4.2, statistical test employed to compare implemented algorithms are presented.

### 4.1 Problem Instances/Test-Data Generation

In order to evaluate the algorithm, test-data were generated using the methodology of [36, 33]. This method is shown in Table 1 and described below.

For each job $j$, the processing time $p_{i,j}$ on machine $i$ are randomly generated integers from the uniform distribution within the interval $[1, 99]$. The time windows of job $j$, that is, the earliest due date $a_j$ and latest due dates $d_j$ are also, random integers from the same distribution within the interval $[(1-\theta-\frac{\vartheta}{2})\sum_i p_{i,j}, (1-\theta+\frac{\vartheta}{2})\sum_i p_{i,j}]$ where $\sum_i p_{i,j}$ is the total processing time of each job $j$ on machine $i$; the tardiness factor is $\theta$ and $\vartheta$ is the relative range of the time windows. Sets of instance are generated for small and large problems with their parameters as shown in Table 2 using $\theta = \{0.2, 0.4, 0.6, 0.8\}$ and $\vartheta = \{0.4, 0.6, 0.8, 1.0\}$, where $\theta < \vartheta$.

Table 1: Proposed instance/data generation scheme

| Parameter | Distribution function | Indices (independent) |
|---|---|---|
| Processing time | $p_{i,j} \approx U[1, 99]$ | $i = 1, 2, \dots m$ |
| | | $j = 1, 2, \dots n$ |
| Due dates | $a_j \approx U(1-\theta-\frac{\vartheta}{2})\sum_i p_{i,j}$ | $j = 1, 2, \dots n$ |
| | $d_j \approx U(1-\theta+\frac{\vartheta}{2})\sum_i p_{i,j}$ | $j = 1, 2, \dots n$ |

For each problem combination 50 independent test-data were generated, thus, a total of 17,250 problem instances were solved.

Table 2: Parameters of test problems

| Parameter | Problem type | |
|---|---|---|
| Number of | Small | Large |
| Machines | 3, 5, 7, 9, 10 | 3, 5, 7, 9, 10, 15, 20, 25, 30, 40, 50, 60, 70, 80, 100 |
| Jobs | 10, 20, 30, 40, 50 | 90, 100, 200, 300, 500, 900, 1000, 1500, 2000, 2500 |
| Replications | 50 | 50 |
| Total replication in each | $45 \times 50 = 2250$ | $300 \times 50 = 15000$ |
| Total replications | $2250 + 15000 = 17250$ | |

## 4.2 Statistical Test

For each instance, the result obtained by the algorithms in this study are compared with existing algorithm in the literature. These comparisons are performed with two-way hypothesis test for two independent samples with significant level of $\alpha = 0.05$. A well known non-parametric Jonckhere-Terpstra (JT) test is adopted for the test since it is a non-parametric testing procedure that does not require that the data set be normally distributed.

For the algorithms HTVNS and VNS-MIP, the following hypotheses are formulated to compare the means of their solutions/CPU run-times on generated instances:

Null Hypothesis ($H_0$): The mean ET of $HTVNS = VNS - MIP$.

Alternative Hypothesis ($H_1$): The mean ET of $HTVNS \neq VNS - MIP$.

Also, similar hypothesis are followed for the comparison of the average CPU times for the algorithms.

Consequently, the null hypothesis implies that there is no statistical significance that the mean of the solutions obtained by using algorithm HTVNS is diifferent from the mean of the solution obtained by using VNS-MIP. However, accepting the alternative hypothesis and rejecting the null hypothesis suggests that at the level of significance adopted, there is statistically significant prove that the solution obtained by using a particular method is better on the average than others.

## 4.3 Parameter Setting

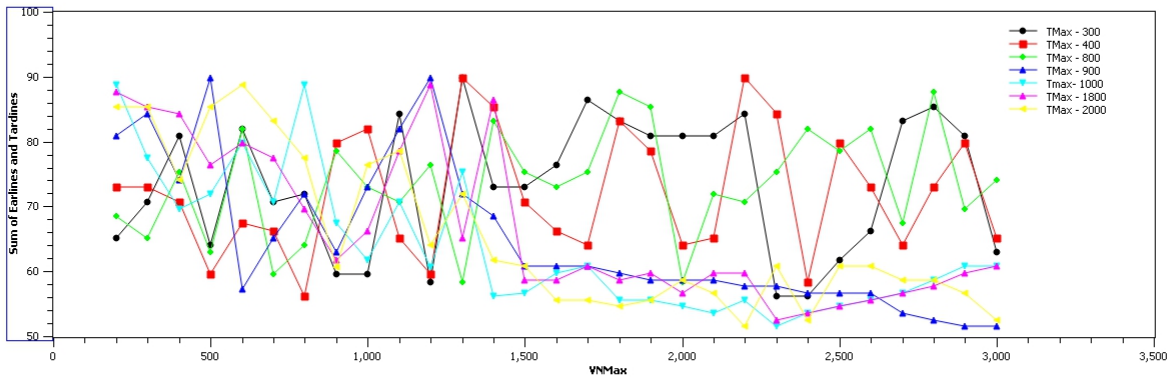
Due to the stochastic nature of metaheuritics, the values of the parameters have significant effect on the performance of the algorithm. The parameters should be calibrated before implementation and comparison with the optimal or best performing algorithms in this area of study. In this paper, the calibration is done through statistical experimental design technique. The approached used is the full factorial design in [37].

Table 3: Value of Parameter

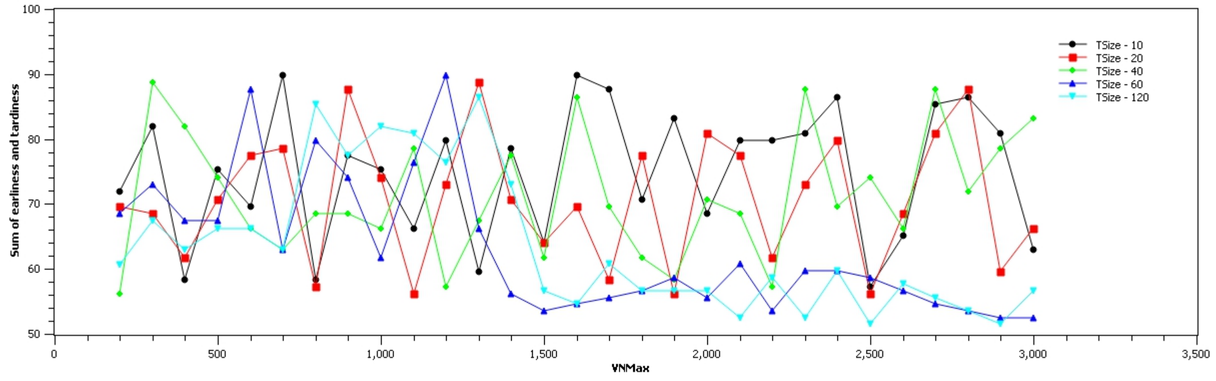
| Parameters | Values | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Tsize | 10 | 20 | 40 | 60 | 120 | | | |
| VNMax | 200 | 300 | 600 | 1000 | 1500 | 2000 | 2500 | 3000 |
| Tmax | 300 | 400 | 800 | 900 | 1000 | 1800 | 2000 | |

For the proposed algorithm, the following parameters are considered as controllable parameters; TMax, Tsize and VNMax. Different values considered for these parameters are displayed in Table 3. The best value for each of these parameters would be obtained by full factorial approach.

Full factorial approach results in 1140 different treatments which were run for 30 times. Results are displayed in Figure 1, which shows performance of HTVNS according to different levels of parameters at $m = 5$ and $n = 200$. Based on the average values, Tsize value of 60 is better than that of other values and 900 for Tmax is preferred while VNMax of 2500 obtained a support than the rest values. As shown, it is observed during experimentation that the number of iterations beyound 2500 shows no significant improvement in the values of ET but increased the CPU running times. Similar procedure is performed for the VNS-MIP its parameter tuning.



(a) Parameter tunning of Tmax



(b) Parameter tunning of TSize

Figure 1: Parameter tunning of HTVNS as VNMax increases at m = 5, n = 200

## 4.4 Results and Discussion

In this section, the CPU running time and quality of solution obtained when HTVNS and VNS-MIP are applied to minimize ET on flow shop are discussed. VNS-MIP is an hybrid of variable neighbourhood search algorithm and Mixed Integer programming proposed in [31].

As stated before, each $n \times m$ combination is evaluated for 50 different problem instances. Therefore, HTVNS and VNS-MIP were run 50 times for each instance of $n \times m$ combination and the average values of ET and CPU running time in seconds are presented in Tables 5 and 6, respectively.

In Sections 4.5 and 4.6, the quality of solutions and CPU run-time obtained by these algorithms are compared.

## 4.5 Comparison of Quality of Solution of HTVNS with VNS-MIP

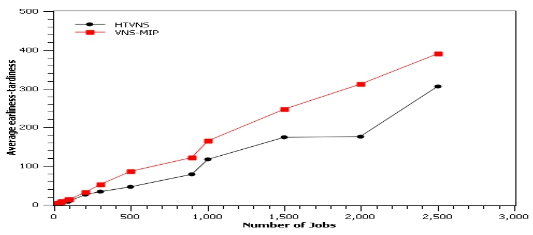
Each problem of the $50$ replications of all instances described in Table 2 is solved using HTVNS and VNS-MIP. Table 5, shows the average quality of ET returned by each algorithm when applied to all set of instances. The first column shows the number of jobs while second and third column represent results obtained by HTVNS and VNS-MIP, respectively. The table of results is stacked using the range of jobs where the first row represents the number of machines.

From Table 5, it can be observed that in most of the problem instances, the solution obtained with HTVNS is better than that of VNS-MIP. Statistical investigation using the JT method described in Section 4.2 as displayed in Table 4 shows that there is significant difference in the proposed HTVNS and VNS-MIP. Therefore, for a fixed level of $m$ and $n$, the metaheuristics are statistically different at $5\%$ level. Consequently, the null hypothesis of equal mean is rejected for the average ET values. Furthermore, Figure 2 shows a plot of average ET obtained by HTVNS and VNS-MIP for $m = 3, 10, 70$ and $100$ against the number of jobs. The graph shows that average ET for each machine increases as the the number of job increases for both algorithms. Also, both algorithms performed almost equally when the number of jobs is less than 300; whereas, a significant difference is observed as the number of jobs increases. Therefore, it can be asserted that the mean solution of HTVNS is better than that of VNS-MIP in many of the instances.

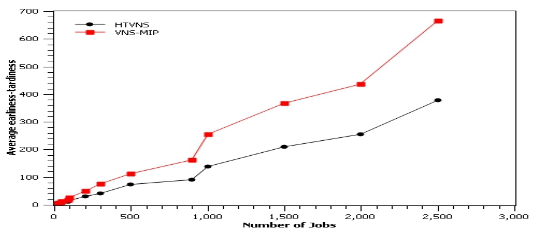
## 4.6 Comparison of Computational Times of HTVNS with VNS-MIP

Table 6 shows the average CPU running times returned by each algorithm when applied to all set of instances. Like in Table 5, it is a stack table where the first rows represents the number of machines. The first column shows number of jobs while second and third column represent results obtained by HTVNS and VNS-MIP, respectively. It can be observed that there is no obvious difference between the CPU running times of HTVNS and VNS-MIP but HTVNS has considerable difference as problem instances increase in size.

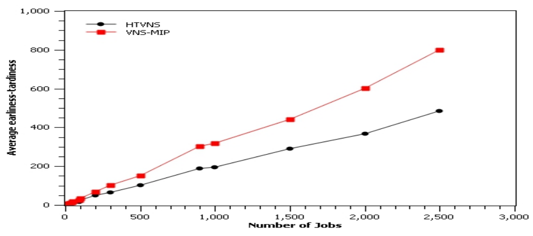
It is worthy of mention to state that collaborative use of TS and VNS has a considerable effect on the performance of HTVNS especially for the large instances. This is because of the TS searches for the best solutions in association with VNS which guides search process of HTVNS in the best direction when TS is trapped in a neighbourhood.



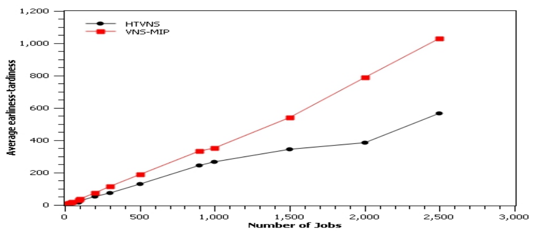
(a) Average Earliness-Tardiness when m=3



(b) Average Earliness-Tardiness when m=10

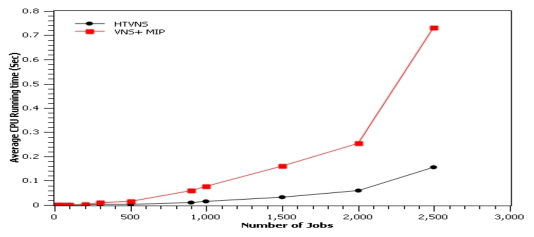


(c) Average Earliness-Tardiness when m=70

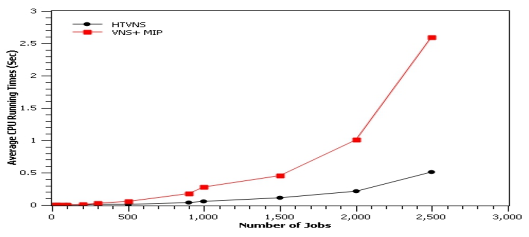


(d) Average Earliness-Tardiness when m=100

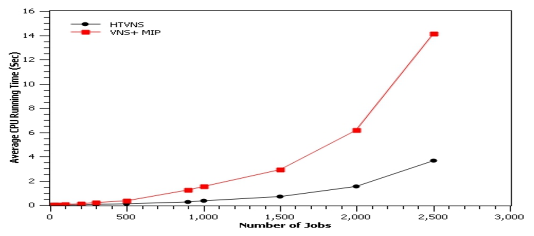
Figure 2: Comparison of Average Earliness-Tardiness values obtained by HTVNS and VNS-MIP



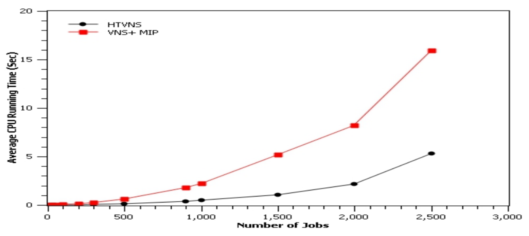
(a) Average CPU Running Times in Seconds when m=3



(b) Average CPU Running Times in Seconds when m=10



(c) Average CPU Running Times in Seconds when m=70



(d) Average CPU Running Times in Seconds when m=100

Figure 3: Comparison of Average Earliness-Tardiness values obtained by HTVNS and VNS-MIP

Consequently, a statistical test using JT is also employed to investigate the difference in the proposed HTVNS and VNS. JT analysis shown in Table 4 confirms that the CPU running times are statistical different at 5% significant level. Also, Figure 3 shows how HTVNS outperforms VNS-MIP as the number of job increases at each machine level.

Table 4: Summary of test of hypothesis

| S/N | Null Hypothesis | p-value | Decision |
|---|---|---|---|
| 1 | Average ET of HTVNS = VNS-MIP | 0.001 | Reject |
| 2 | Average of CPU Running times of HTVNS = VNS-MIP | 0.000 | Reject |

## 5 Conclusion

In this paper, an hybrid algorithm which investigates the effect of using tabu-search along with variable neighbourhood search to minimizing the sum of earliness and tardiness of jobs with due windows on flow shop. The algorithm implements a variable neighborhood search as a guild to redirect the search procedure when Tabu-search is trapped in a solution space. The VNS-MIP is adapted to present a comparison of the algorithm's performance. The algorithm is proved to perform better in terms of values of objective and computational time.

## Acknowledgements

The authors wish to thank the Editor and the anonymous reviewers for their constructive comments, which have greatly improved the contents of the paper.

## Competing financial interests

The authors declare no competing financial interests.

## References

[1] Josefowska, J., Jurisch, B. & Kubiak, W. Scheduling shops to minimize the weighted number of late jobs. *Operations Research Letters*, 10, 27–33, (1994).

[2] Ohno, T. *Toyota production system: beyond large-scale production*. Productivity Press, (1988).

[3] Ohno, T. *Just-In-Time for today & tomorrow*. Productivity Press, (1988).

[4] Murata, T., Ishibuchi, H. & Tanaka, H. Multi- objective genetic algorithm and its applications to flow shop scheduling. *Computers & Industrial Engineering*, 30(4), 957-967, (1996).

[5] Monden, Y. Toyotal production system. *Institute of Industrial Engineering Press, Norcross, G.A*, (1983).

[6] Kim, Y. & Kim, J. A coevolutionary algorithm for balancing & sequencing in mixed model assembly lines. *Applied Inteligience*, 13(3), 247–258, (2000).

[7] Cheng, T. C. E., Gupta, J. N. D. & Wang, G. A review of flow shop scheduling research with setup times. *Production and Operations Management*, 9, 262–282, (2000).

[8] French, S. *Sequencing and scheduling: an introduction to the mathematics of the job shop*. Ellis Harwood, England, (1982).

[9] Pinedo, M. *Scheduling: Theory, Algorithms & Systems*. Fourth edition, Springer, (2015).

[10] González, P., Torres J. F., González, P., León, J., & Usano, R. R. Flowshop scheduling problems with due date related objectives: A review of the literature. In *XIII Congreso de Ingeniería de Organización*, 1488–1497, (2009).

[11] Kramer, F. J. & Lee, C. Y. Common due-window scheduling. *Production & Operations Management*, 2(4), 262–275, (1993).

[12] Wan, G. & Yen, B. P. C. Tabu search for single machine scheduling with distinct due windows & weighted earliness/tardiness penalties. *European Journal of Operational Research*, 142, 271–281, (2002).

[13] Wu, C. , Lee, W. & Wang, W. A two-machine flowshop maximum tardiness scheduling problem with a learning effect. *The International Journal of Advanced Manufacturing Technology*, 31(7-8), 743–750, (2007).

[14] Lee, C. Earliness–tardiness scheduling problems with constant size of due date window. *Research Report No. 91-17, Industrial & Systems Engineering Department, University of Florida, Gainesville*, 4, 262–275, (1991).

[15] Koulamas, C. On the complexity of two-machine flowshop problems with due date related objectives. *European journal of operational research*, 106(1), 95–100, (1998).

[16] Yeung, W. K, Oğuz, C., & Cheng, T. C. E. Two-stage flowshop earliness & tardiness machine scheduling involving a common due window. *International Journal of Production Economics*, 90(3), 421–434, (2004).

[17] Mosheiov, G. & Sarig, A. Scheduling with a common due-window: Polynomi-ally solvable cases. *Information Sciences*, 180(8), 1492–1505, (2010).

[18] Zegordi, S. H., Itoh, K. & Enkawa T. A knowledgeable simulated annealing scheme for the early-tardy flow shop scheduling problem. *International Journal of Production Research*, 33, 1449–1466, (1995).

[19] Schaller, J. & Valente, J. M. S. A comparison of metaheuristic procedures to schedule jobs in a permutation flow shop to minimise total earliness & tardiness. *International Journal of Production Research*, 51, 772–779, (2013).

[20] Janiak, A., Kozan, E., Lichtenstein, M. Oğuz, & C. Metaheuristic approaches to the hybrid flow shop scheduling problem with a cost-related criterion. *International journal of production economics*, 105(2), 407–424, (2007).

[21] Khalouli, S., Ghedjati, F, & Hamzaoui, A. A meta-heuristic approach to solve a jit scheduling problem in hybrid flow shop. *Engineering Applications of Artificial Intelligence*, 23(5), 765–771, (2010).

[22] Moslehi, G., Mirzaee, M., Vasei, M., Modarres, M. & Azaron, A. Two-machine flow shop scheduling to minimize the sum of maximum earliness & tardiness. *International Journal of Production Economics*, 122(2), 763–773, (2009).

[23] Huang, C., Huang, Y. & Lai, P. Modified genetic algorithms for solving fuzzy flow shop scheduling problems and their implementation with cuda. *Expert Systems with Applications*, 39(5), 4999–5005, (2012).

[24] M'Hallah, R. An iterated local search variable neighborhood descent hybrid heuristic for total earliness tardiness permutation flow shop. *International Journal of Production Research*, 52(13),3802–3819, (2014).

[25] Liao, L. & Huang, C. Tabu search for non-permutation flowshop scheduling problem with minimizing total tardiness. *Applied Mathematics & Computation*, 217(2), 557–567, (2010).

[26] Glover, F. & Laguna, M. Tabu search. In *Handbook of combinatorial optimization*. Springer, 2093–2229, (1998).

[27] Mladenovic, N. & Hansen, P. Variable neighborhood search. *Computers & Operations Research*, 24, 1097–1100, (1997).

[28] Hansen, P., Mladenovic, N. & Moreno-Perez, J. A. Variable neighbourhood search: Methods and applications. *Annals of Operations Research*, 175(4), 367–407, (2010).

[29] Behnamian, J. & Ghomi, S. F. A survey of multi-factory scheduling. *Journal of Intelligent Manufacturing*, 27(1), 231–249, (2016).

[30] Behnamian, J. & Zandieh, M. Earliness and tardiness minimizing on a realistic hybrid flowshop scheduling with learning effect by advanced metaheuristic. *Arabian Journal for Science and Engineering*, 38(5), 1229–1242, (2013).

[31] M'Hallah, R. Minimizing total earliness & tardiness on permutation flow shop using vns & mip. *Computers & Industrial Engineering*, 75, 142–156, (2016).

[32] Graham, R. L., Lawler, E. L. , Lenstra, T. K., & Rinnooy-Kan, A. H. G. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5, 287–326, (1979).

[33] Adamu, M.O., Budlender, N., & Idowu, G. A . A note on just in time scheduling on flow shop machines. *Journal of the Nigerian Mathematical Society*, 33, 321–331, (2014).

[34] Rosa, B. F., Souza, M. J. F., De Souza, S. R. , Filho, M. F., Ales, Z. & Michelon, P. Y. P. Algorithms for job scheduling problems with distinct time windows & general earliness/tardiness penaltiese. *Computers & Operations Research*, 81, 203–215, (2017).

[35] Hansen, P., Mladenovic, N., & Prez J. A. M. Variable neighborhood search: methods and applications. *European Journal of Operational Research*, 6(4),319–316, (2008).

[36] Bulfin, R. L. & M'Hallah, R. Minimizing the weighted number of tardy jobs on a two-machine flow shop. *Computers and Operational Research*, (30), 1887–1900, (2003).

[37] Cochran, W. O. & Cox, G. M. *Experimental designs*. 2nd Edition. Wiley, (1992).

Table 5: Comparison of Average Sum of Earliness and Tardiness Values Obtained by HTVNS and VNS-MIP

| n | m = 3 | | m = 5 | | m = 7 | | m = 10 | | m = 15 | | m = 20 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HTVNS | VNS | HTVNS | VNS | HTVNS | VNS | HTVNS | VNS | HTVNS | VNS | HTVNS | VNS |
| 10 | 0.88 | 1.74 | 0.95 | 1.74 | 1.27 | 2.12 | 1.42 | 2.56 | 1.42 | 2.57 | 1.74 | 2.63 |
| 20 | 2.45 | 2.71 | 2.54 | 2.93 | 2.54 | 3.55 | 2.68 | 4.48 | 2.77 | 4.59 | 3.23 | 4.59 |
| 30 | 3.52 | 5.08 | 3.74 | 5.85 | 4.22 | 7.29 | 4.36 | 7.52 | 4.40 | 7.56 | 4.49 | 7.87 |
| 40 | 4.12 | 6.51 | 4.22 | 7.26 | 4.89 | 8.83 | 5.26 | 9.35 | 5.26 | 9.51 | 5.85 | 9.78 |
| 50 | 5.76 | 8.57 | 6.14 | 9.63 | 6.31 | 12.08 | 7.20 | 13.17 | 8.19 | 13.23 | 8.52 | 13.81 |
| 90 | 8.70 | 13.87 | 9.70 | 14.06 | 9.78 | 17.56 | 10.24 | 17.66 | 10.30 | 19.65 | 10.71 | 20.14 |
| 100 | 8.27 | 13.83 | 8.76 | 17.33 | 10.04 | 22.96 | 14.63 | 24.89 | 14.92 | 28.74 | 16.94 | 29.42 |
| 200 | 25.91 | 32.11 | 27.40 | 33.10 | 27.44 | 42.19 | 30.09 | 49.71 | 30.62 | 55.71 | 30.90 | 57.70 |
| 300 | 34.37 | 52.09 | 34.65 | 55.21 | 37.22 | 73.78 | 41.27 | 74.81 | 45.13 | 76.21 | 46.57 | 80.59 |
| 500 | 46.96 | 87.00 | 52.88 | 87.33 | 59.88 | 106.56 | 73.12 | 112.42 | 75.89 | 121.80 | 77.15 | 125.11 |
| 900 | 78.49 | 122.19 | 79.48 | 130.97 | 82.59 | 158.56 | 89.89 | 161.78 | 108.31 | 170.04 | 117.74 | 183.95 |
| 1000 | 117.27 | 165.82 | 123.23 | 175.11 | 126.11 | 236.53 | 138.36 | 254.13 | 148.76 | 277.68 | 152.68 | 284.13 |
| 1500 | 175.67 | 247.55 | 183.17 | 248.37 | 203.77 | 324.15 | 210.17 | 366.85 | 211.22 | 368.89 | 213.94 | 369.06 |
| 2000 | 174.86 | 312.28 | 238.06 | 356.17 | 254.50 | 427.20 | 255.65 | 437.35 | 255.86 | 445.58 | 284.73 | 451.36 |
| 2500 | 306.29 | 390.03 | 340.08 | 406.09 | 363.86 | 560.14 | 377.63 | 665.31 | 389.46 | 668.56 | 404.02 | 703.57 |

| n | m = 25 | | m = 40 | | m = 50 | | m = 60 | | m = 70 | | m = 100 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HTVNS | VNS | HTVNS | VNS | HTVNS | VNS | HTVNS | VNS | HTVNS | VNS | HTVNS | VNS |
| 10 | 1.87 | 2.34 | 1.91 | 2.35 | 1.95 | 2.83 | 2.03 | 3.12 | 2.13 | 3.43 | 2.42 | 3.78 |
| 20 | 3.30 | 3.90 | 3.56 | 3.93 | 3.83 | 5.60 | 4.10 | 6.11 | 4.13 | 6.41 | 4.50 | 7.51 |
| 30 | 4.58 | 6.83 | 4.94 | 7.34 | 5.02 | 9.15 | 5.42 | 9.60 | 5.87 | 9.89 | 6.65 | 10.49 |
| 40 | 5.96 | 8.40 | 6.35 | 9.17 | 7.48 | 11.50 | 7.59 | 12.79 | 8.41 | 13.25 | 10.33 | 15.07 |
| 50 | 8.73 | 11.94 | 9.27 | 13.07 | 9.53 | 15.74 | 10.65 | 16.15 | 10.68 | 17.63 | 12.77 | 18.27 |
| 90 | 11.17 | 17.59 | 12.57 | 18.35 | 13.03 | 25.15 | 14.41 | 25.72 | 15.54 | 26.13 | 16.39 | 29.83 |
| 100 | 18.46 | 25.28 | 21.70 | 26.46 | 23.48 | 31.75 | 23.94 | 35.22 | 25.23 | 35.38 | 26.87 | 36.38 |
| 200 | 38.65 | 48.91 | 42.16 | 51.87 | 44.24 | 64.26 | 47.25 | 65.99 | 48.41 | 67.22 | 50.45 | 73.09 |
| 300 | 48.44 | 74.59 | 59.08 | 77.52 | 60.26 | 95.24 | 60.84 | 95.41 | 63.86 | 102.69 | 75.02 | 113.38 |
| 500 | 92.50 | 107.17 | 93.85 | 111.18 | 99.18 | 136.05 | 101.32 | 145.44 | 102.71 | 152.55 | 128.97 | 190.26 |
| 900 | 119.22 | 160.37 | 158.50 | 198.22 | 171.08 | 238.33 | 186.76 | 242.73 | 189.19 | 303.64 | 245.76 | 333.13 |
| 1000 | 169.06 | 239.61 | 179.01 | 241.73 | 180.35 | 292.62 | 190.10 | 303.61 | 194.92 | 316.73 | 266.63 | 351.97 |
| 1500 | 230.03 | 313.56 | 273.46 | 348.43 | 283.44 | 418.42 | 290.92 | 438.26 | 291.15 | 440.71 | 343.31 | 541.82 |
| 2000 | 290.49 | 381.61 | 317.83 | 400.17 | 332.77 | 495.59 | 338.62 | 523.12 | 366.67 | 601.34 | 386.13 | 787.84 |
| 2500 | 407.08 | 610.07 | 440.19 | 615.90 | 451.82 | 767.70 | 465.31 | 782.54 | 483.12 | 799.75 | 568.49 | 1030.60 |

Table 6: Comparison of Average CPU Running Times (Seconds) of HTVNS and VNS-MIP

| | m = 3 | | m = 5 | | m = 7 | | m = 10 | | m = 15 | | m = 20 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n | HTVNS | VNS | HTVNS | VNS | HTVNS | VNS | HTVNS | VNS | HTVNS | VNS | HTVNS | VNS |
| 10 | 0.00000 | 0.00001 | 0.00000 | 0.00002 | 0.00001 | 0.00003 | 0.00001 | 0.00004 | 0.00001 | 0.00006 | 0.00002 | 0.00007 |
| 20 | 0.00001 | 0.00003 | 0.00001 | 0.00004 | 0.00002 | 0.00007 | 0.00002 | 0.00010 | 0.00003 | 0.00013 | 0.00004 | 0.00017 |
| 30 | 0.00001 | 0.00006 | 0.00002 | 0.00011 | 0.00003 | 0.00013 | 0.00004 | 0.00022 | 0.00006 | 0.00027 | 0.00009 | 0.00037 |
| 40 | 0.00003 | 0.00011 | 0.00004 | 0.00020 | 0.00006 | 0.00027 | 0.00008 | 0.00039 | 0.00012 | 0.00057 | 0.00016 | 0.00073 |
| 50 | 0.00003 | 0.00017 | 0.00006 | 0.00027 | 0.00008 | 0.00040 | 0.00011 | 0.00048 | 0.00018 | 0.00092 | 0.00022 | 0.00119 |
| 90 | 0.00012 | 0.00049 | 0.00019 | 0.00096 | 0.00027 | 0.00115 | 0.00036 | 0.00162 | 0.00056 | 0.00263 | 0.00073 | 0.00393 |
| 100 | 0.00021 | 0.00095 | 0.00034 | 0.00167 | 0.00051 | 0.00259 | 0.00065 | 0.00328 | 0.00097 | 0.00552 | 0.00140 | 0.00638 |
| 200 | 0.00061 | 0.00311 | 0.00099 | 0.00450 | 0.00147 | 0.00717 | 0.00208 | 0.01027 | 0.00298 | 0.01325 | 0.00406 | 0.02103 |
| 300 | 0.00186 | 0.00889 | 0.00331 | 0.01281 | 0.00449 | 0.01925 | 0.00661 | 0.03000 | 0.00970 | 0.04150 | 0.01215 | 0.05658 |
| 500 | 0.00361 | 0.01510 | 0.00569 | 0.02713 | 0.00820 | 0.03984 | 0.01123 | 0.05753 | 0.01640 | 0.08289 | 0.02234 | 0.11553 |
| 900 | 0.01064 | 0.05986 | 0.01910 | 0.08120 | 0.02462 | 0.11390 | 0.03690 | 0.17202 | 0.05626 | 0.28714 | 0.07288 | 0.35536 |
| 1000 | 0.01518 | 0.07765 | 0.02463 | 0.10953 | 0.03733 | 0.14950 | 0.05368 | 0.27695 | 0.07855 | 0.37430 | 0.10915 | 0.45610 |
| 1500 | 0.03233 | 0.15957 | 0.04907 | 0.26227 | 0.07303 | 0.36520 | 0.10890 | 0.45367 | 0.14793 | 0.82340 | 0.19597 | 1.07977 |
| 2000 | 0.05955 | 0.25480 | 0.11070 | 0.44905 | 0.13985 | 0.58435 | 0.20855 | 1.01005 | 0.32800 | 1.45510 | 0.42955 | 1.85470 |
| 2500 | 0.15630 | 0.73130 | 0.26850 | 1.05410 | 0.37840 | 1.56360 | 0.50600 | 2.58850 | 0.75340 | 3.97880 | 1.10010 | 5.34770 |

| n | m = 25 | | m = 40 | | m = 50 | | m = 60 | | m = 70 | | m = 100 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 10 | 0.00002 | 0.00010 | 0.00003 | 0.00015 | 0.00004 | 0.00021 | 0.00005 | 0.00022 | 0.00006 | 0.00027 | 0.00008 | 0.00035 |
| 20 | 0.00006 | 0.00024 | 0.00008 | 0.00043 | 0.00010 | 0.00043 | 0.00013 | 0.00053 | 0.00014 | 0.00069 | 0.00022 | 0.00088 |
| 30 | 0.00010 | 0.00044 | 0.00017 | 0.00069 | 0.00020 | 0.00094 | 0.00027 | 0.00127 | 0.00030 | 0.00130 | 0.00042 | 0.00178 |
| 40 | 0.00022 | 0.00093 | 0.00033 | 0.00144 | 0.00040 | 0.00205 | 0.00052 | 0.00255 | 0.00061 | 0.00269 | 0.00079 | 0.00406 |
| 50 | 0.00031 | 0.00121 | 0.00046 | 0.00227 | 0.00056 | 0.00282 | 0.00066 | 0.00307 | 0.00085 | 0.00365 | 0.00115 | 0.00617 |
| 90 | 0.00089 | 0.00462 | 0.00151 | 0.00734 | 0.00178 | 0.00949 | 0.00238 | 0.01049 | 0.00253 | 0.01264 | 0.00360 | 0.01718 |
| 100 | 0.00164 | 0.00761 | 0.00266 | 0.01289 | 0.00346 | 0.01505 | 0.00418 | 0.02141 | 0.00488 | 0.02526 | 0.00671 | 0.03677 |
| 200 | 0.00506 | 0.02199 | 0.00779 | 0.03762 | 0.01010 | 0.04638 | 0.01238 | 0.06288 | 0.01442 | 0.07720 | 0.02129 | 0.10865 |
| 300 | 0.01618 | 0.06845 | 0.02574 | 0.10557 | 0.03106 | 0.14680 | 0.03835 | 0.18009 | 0.04646 | 0.20097 | 0.06155 | 0.25682 |
| 500 | 0.02966 | 0.12867 | 0.04912 | 0.20537 | 0.05911 | 0.29057 | 0.06991 | 0.36368 | 0.07813 | 0.36483 | 0.11149 | 0.59621 |
| 900 | 0.08924 | 0.46458 | 0.15106 | 0.74916 | 0.19476 | 0.99310 | 0.22124 | 1.10498 | 0.26364 | 1.21874 | 0.38978 | 1.79514 |
| 1000 | 0.12998 | 0.59305 | 0.22128 | 0.95920 | 0.26073 | 1.07908 | 0.32915 | 1.45838 | 0.35330 | 1.51928 | 0.50058 | 2.23933 |
| 1500 | 0.26413 | 1.12950 | 0.39377 | 1.88013 | 0.51020 | 2.66307 | 0.60173 | 2.76283 | 0.71590 | 2.91720 | 1.02820 | 5.21437 |
| 2000 | 0.50080 | 2.24820 | 0.82550 | 4.42890 | 1.09190 | 5.18330 | 1.21295 | 5.26505 | 1.53150 | 6.16985 | 2.15775 | 8.21070 |
| 2500 | 1.25260 | 6.32110 | 2.13470 | 10.31500 | 2.45020 | 10.33970 | 3.18450 | 10.65330 | 3.64100 | 14.11220 | 5.28940 | 15.94750 |