





# Elimination Properties for a Probabilistic Scheduling Problem

Wojciech Bożejko <sup>1,\*</sup>, Paweł Rajba <sup>2</sup>, Mariusz Uchroński <sup>1</sup> and Mieczysław Wodecki <sup>3</sup>

<sup>1</sup> Department of Control Systems and Mechatronics, Faculty of Electronics, Wrocław University of Science and Technology, Wyb. Wyspińskiego 27, 50-370 Wrocław, Poland

<sup>2</sup> Institute of Computer Science, University of Wrocław, Joliot-Curie 15, 50-383 Wrocław, Poland

<sup>3</sup> Department of Telecommunications and Teleinformatics, Faculty of Electronics, Wrocław University of Science and Technology, Wyb. Wyspińskiego 27, 50-370 Wrocław, Poland

\* Correspondence: wojciech.bozejko@pwr.edu.pl

**Abstract:** In many areas of the economy, we deal with random processes, e.g., transport, agriculture, trade, construction, etc. Effective management of such processes often leads to optimization models with random parameters. Solving these problems is already very difficult in deterministic cases, because they usually belong to the *NP-hard* class. In addition the inclusion of the uncertainty parameters in the model causes additional complications. Hence these problems are much less frequently studied. We propose a new customized approach to searching the solutions space for problems with random parameters. We prove new, strong properties of solutions, the so-called block elimination properties, accelerating the neighborhood search. They make it possible to eliminate certain subsets of the solution space containing worse solutions without the need to calculate the value of the criterion function. Blocks can be used in the construction of exact and approximate algorithms, e.g., metaheuristics such as tabu search, significantly improving their efficiency.

**Keywords:** scheduling; single machine; random parameters



**Citation:** Bożejko, W.; Rajba, P.; Uchroński, M.; Wodecki, M. Elimination Properties for a Probabilistic Scheduling Problem. *Appl. Sci.* **2023**, *13*, 5304. <https://doi.org/10.3390/app13095304>

Academic Editor: Richard C. Millham

Received: 28 January 2023

Revised: 16 March 2023

Accepted: 20 March 2023

Published: 24 April 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Studies of discrete optimization problems that have been conducted for many years relate mostly to deterministic models, in which the basic assumption is the uniqueness of the parameters. In order to solve these types of problems that mostly belong to classes of strongly NP-hard problems, a number of very effective approximate algorithms have been developed. The solutions designated by these algorithms are only slightly different from the optimal ones. In practice, however, during the process realization (according to the adopted schedule), it often turns out that some parameters (e.g., operation times) are different from those initially adopted values. In the absence of the stability of solutions, it happens that it loses not only optimality but also acceptability. However, in many applications there are great difficulties in defining process parameters in a clear way or data coming from imprecise measuring devices. They bear a certain error, thus they are uncertain.

The choice of approach to modeling and analysis results from such issues as system features, the ability to perform data measurements, data reliability, the power of theoretical tools, etc. The knowledge of all the aforementioned elements is necessary for the efficient solving of practical problems (Shang, You [1], Zhang [2]). For example, the data regarding the duration of the activity can be accepted as deterministic (e.g., normative), measure their random characteristics (take a series of measurements and verify the hypothesis about the type of distribution and its parameters), make a measurement to approximate a deterministic value (if the variance is small enough) or designating the membership function for the fuzzy representation, determine the membership function based on expert opinion. The complexity of the problems and computational problems already for deterministic cases results in the fact that problems with uncertain data are much less formulated and analyzed.

Many problems related to the decision-making process come down to solving some scheduling problems on machines. Scheduling problems with uncertain data can be solved

using methods based on elements of probability (Shaked and Shandhkumard [3], Zhu and Cai [4], Van den Akker and Hoogeveen [5,6], Vondrák [7], Dean [8]), Soroush [9], He [10] or fuzzy set theory (Prade, [11], Ishii [12], Iscibuchi et al. [13], Itoh and Ishii [14], Bocewicz et al. [15]). The accuracy (quality) of such an algorithm is not determined on the basis of individual instances of the problem (as in deterministic data), but on a certain family of examples generated randomly according to a certain probability of distributions. Such specific accuracy will be called stability of an algorithm. Research carried out in recent years, in which uncertainty is taken into account, is promising not only at the stage of model construction but also during the algorithm design (Rajba and Wodecki [16], Bożejko et al. [17]).

In this paper, we consider the *strongly NP-hard* single-machine task scheduling problem with critical lines and minimizing the total weighted tardiness tasks cost on a single machine. Task execution times are random variables. We present methods of an intermediate review of solutions, the so-called ‘block properties of the problem,’ which we use in the tabu search algorithm (TS). This algorithm is one of the best for solving the deterministic version of the problem under consideration (Bożejko et al. [18]). It is deterministic and guarantees the repeatability of calculations. Computational experiments were conducted mainly to:

1. check the effectiveness of using blocks, i.e., their impact on the time and quality of the determined solutions; and
2. testing the stability of algorithms, i.e., their resistance to data disturbances.

The computational experiments show that the solutions obtained in the probabilistic model including blocks are stable and allow us to generate better solutions in less time.

It follows from the conducted computational experiments that the solutions obtained in the probabilistic model are stable, i.e., not very sensitive to random data disturbances.

## 2. Single-Machine Total Weighted Tardiness Tasks Scheduling Problem

Tasks scheduling problems on a single machine with cost goal functions have a very long (over 50 years) history (the first work of Rinnoy Kan et al. [19] appeared in 1975). However, despite the simplicity of formulation, they mostly belong to the class of strongly NP-hard problems. They are important both from the point of view of theory and practice. Such problems often constitute a significant part of more extensive production systems. Different variants of scheduling tasks on a single machine are still intensively tested and the results obtained are the inspiration for much more complex research of multi-machine problems. While describing of the problem considered in this section we will use some definitions, designations and properties presented in the following works: Bożejko et al. [18,20], Rajba and Wodecki [16].

A single-machine Total Weighted Tardiness Problem, abbreviated as TWT, will be defined as follows.

**TWT Problem:** Let

$$\mathcal{J} = \{1, 2, \dots, n\},$$

be a set of tasks to be performed, without interruption, on the executing machine performing at a given moment at most one task at a time. For task  $i \in \mathcal{J}$  ( $i = 1, \dots, n$ ) we define:

- $p_i$ —execution time;
- $w_i$ —penalty for tardiness;
- $d_i$ —demanded completion time..

Let  $\pi = (\pi(1), \pi(2), \dots, \pi(n))$  be any permutation of elements from  $\mathcal{J}$ , and  $\Phi$  a set of all such permutations. Then,

$$C_{\pi(i)} = \sum_{j=1}^i p_{\pi(j)} \tag{1}$$

is a moment of completion of task  $\pi(i) \in \mathcal{J}$ , executed as  $i$ -th in a sequence. If  $C_{\pi(i)} \leq d_{\pi(i)}$ , then the task is called an early one, otherwise a tardy one.

Tardiness of task

$$T_{\pi(i)} = \max\{0, C_{\pi(i)} - d_{\pi(i)}\}, \tag{2}$$

and cost of its execution  $f(\pi(i)) = w_{\pi(i)}T_{\pi(i)}$ . By

$$\mathcal{F}(\pi) = \sum_{i=1}^n w_{\pi(i)}T_{\pi(i)} \quad (3)$$

we designate cost of execution of all tasks (weight of permutation  $\pi$ ).

The problem under consideration (which will be also called a deterministic) problem involves determining permutation  $\pi^* \in \Phi$  with the smallest weight, such that

$$\mathcal{F}(\pi^*) = \min\{\mathcal{F}(\pi) : \pi \in \Phi\}. \quad (4)$$

In the literature, this problem is denoted by  $1||\sum w_iT_i$  and belongs to the strongly NP-hard class of problems. Optimal algorithms for solving it based on the dynamic programming method were presented by Lawler, Moore [21], Sahni [22], those based on the branch and bound method were presented by Potts, Van Wassenhove [23] Villareal, Bulfin [24], Urgo [25] and Wodecki [26]. They enable the effective determination of optimal solutions for examples whose number of tasks does not exceed 80. Due to the time-consuming characteristic of exact algorithms, in practice, approximate algorithms (mainly metaheuristics) are usually applied.

In the best metaheuristic algorithms, where the task is solving multi-machine task scheduling problems, so-called 'block elimination properties' are used (e.g., for flow shop problems such algorithms are presented in the works of Nowicki and Smutnicki [27] and Grabowski and Wodecki [28]). Blocks enable both the reduction of calculation time, as well as improvement of the values determined by the algorithm solutions. For the single-machine Total Weighted Tardiness Problem TWT presented in this section, in the works of Uchroński [29], Wodecki [26,30] there were two types of blocks presented: the so-called early and tardy blocks, which were successfully used in tabu search algorithms.

### 3. Random Task Execution Times

In the literature we can find many ways in which uncertainty is modeled. First, we can distinguish proactive and reactive approaches where the former assumes that all preparations for handling uncertainty are completed before the algorithm starts its execution, while the latter assumes the opposite, i.e., uncertainty needs to be handled during the actual algorithm execution. Of course, we can also find many types of combination of those two. Moreover, we can also distinguish an online approach where the problem instance is not known in advance. Next, uncertainty can be modeled in different ways and even though there are several descriptions in the field, we can distinguish 3 main categories: probabilistic (or stochastic), where random variables are involved, and a fuzzy description and bound form where fixed ranges are involved. Sometimes by stochastic the authors refer to certain types of online algorithms and solutions in the bound form are sometimes referred to as robust, by fixed ranges we can introduce the guarantee on the upper or bottom bound of the provided solutions (however, in general the robustness description is not limited to the bound form). Having the above, in this paper we investigate the proactive version of the problem with uncertainty modeled by a probability approach with random variables; that is, in this section, we consider the probabilistic version of the aforementioned single-machine Total Weighted Tardiness Problem and we assume that the times of completing the tasks are independent random variables.

They are most often described in the literature as scheduling problems with uncertain parameters. In practice, there are great difficulties in establishing the probability distribution of random parameters. Especially when we are dealing with unique processes, therefore, there is no representative statistical data.

An extensive review of methods and algorithms for solving the problems of combinatorial optimization with random parameters was presented by Vondrák in monograph [7] and in the more recent work by Xiaoqiang et al. [31]. Some practical problems are also considered in the works of Božejko et al. [20,32–34]. The last of these works concerns the

implementation of construction projects under uncertainty. We will now introduce the necessary definitions and designations.

If  $X$  is a continuous random variable, we will use the following designations later in this work:

- $F_X$ —cumulative distribution function of random variable  $X$ ;
- $f_X$ —density function of random variable  $X$ ;
- $E(X)$ —expected value of random variable  $X$ .

We are considering, as described in Section 2, a probabilistic version of the TWT problem, in which the task execution times  $\tilde{p}_i$  are independent random variables, and the remaining task parameters  $w_i$  and  $d_i$  ( $i = 1, 2, \dots, n$ ) are deterministic. This problem will be denoted in short by PTWT.

If task execution times  $\tilde{p}_i$  are random variables, then for any order of execution of tasks  $\pi \in \Phi$ , the completion of the task  $\pi(k)$

$$\tilde{C}_{\pi(k)} = \tilde{p}_{\pi(1)} + \tilde{p}_{\pi(2)} + \dots + \tilde{p}_{\pi(k)},$$

tardiness  $\tilde{T}_{\pi(k)} = \max\{0, \tilde{C}_{\pi(k)} - d_{\pi(k)}\}$  (Equivalent (2)) and criterion function (Equivalent (3))

$$\tilde{\mathcal{F}}(\pi) = \sum_{i=1}^n w_{\pi(i)} \tilde{T}_{\pi(i)}. \tag{5}$$

are also random variables.

In algorithms for solving optimization problems, it is necessary to compare the values of the criterion function for various acceptable solutions (e.g., permutations). In a case when this function is a random variable (5) we will be using its expected value. Therefore, as comparative criteria of solutions the following function will be used:

$$\mathcal{L}(\pi) = E(\tilde{\mathcal{F}}(\pi)) = \sum_{i=1}^n w_{\pi(i)} E(\tilde{T}_{\pi(i)}). \tag{6}$$

By

$$f(\pi(k)) = w_{\pi(k)} E(\tilde{T}_{\pi(k)}) \tag{7}$$

we designate the cost of execution of task  $\pi(k)$ .

Later in this work we present the methods for calculating the value of criterion function (6). Following elements in sequence from  $\pi \in \Phi$

$$\beta = (\pi(a), \pi(a + 1), \dots, \pi(b)),$$

where  $1 \leq a \leq b \leq n$  will be called **subpermutation** of permutation  $\pi$ . The cost of performing tasks from subpermutation  $\beta$

$$\mathcal{L}(\beta) = \sum_{i=a}^b w_{\pi(i)} E(\tilde{T}_{\pi(i)}). \tag{8}$$

By  $\mathcal{Y}(\beta)$  we denote the set of sub-permutation elements  $\beta$ , i.e.,

$$\mathcal{Y}(\beta) = \{\pi(a), \pi(a + 1), \dots, \pi(b)\}. \tag{9}$$

#### 4. Blocks of Tasks

We are considering permutation  $\pi \in \Phi$ , i.e., some kind of solution to the PTWT problem. If the expected value of the completion time of the task  $\pi(i)$

$$E(\tilde{C}_{\pi(i)}) \leq d_{\pi(i)},$$

then the task  $\pi(i)$  it is called an *early* one, otherwise, i.e., when

$$E(\tilde{C}_{\pi(i)}) > d_{\pi(i)},$$

a *tardy* one.

Later in this section, we present a method of breaking  $\pi$  into subpermutations (called blocks) containing only early or tardy tasks. Next, we will determine the optimal order of tasks in each of these subpermutations. A permutation generated in this way has the following property: *any change in the order of elements in any block does not generate a solution with a smaller value of the criterion function (6)*. This is the so-called **block elimination property**. Blocks for problems with a deterministic demanded completion time (due-dates) are considered by Uchroński [29] and Bożejko et al. [35], and with a probabilistic one, by Bożejko et al. [36].

#### 4.1. Blocks of Early Tasks

**Definition 1.** Subpermutation of tasks  $\pi^T$  in permutation  $\pi \in \Phi$  is called **blocks of early tasks** (in short **T-block**), if:

- (a) every task  $j \in \pi^T$  is early and  $d_j \geq E(\tilde{C}_{last})$ , where  $\tilde{C}_{last}$  is a random variable—date of completion of execution of the last task from  $\pi^T$ ,
- (b)  $\pi^T$  is the maximum subpermutation meeting the constraint (a).

**Corollary 1.** If  $\pi^T = (\pi(a), \pi(a + 1), \dots, \pi(b))$  is a T-block in a permutation  $\pi$ , then

- 1. an inequality  $\min\{d_j : j \in \pi^T\} \geq E(\tilde{C}_{last})$  is fulfilled,
- 2. the cost of execution of tasks from  $\pi^T$ ,

$$\mathcal{L}(\pi^T) = \sum_{i=a}^b w_{\pi(i)} E(\tilde{T}_{\pi(i)}) = 0. \tag{10}$$

Using this property, we determine the first T-block in permutation  $\pi$ . After minor modifications (expected values of random variables  $E(\tilde{C}_i)$  instead of deterministic values  $C_i$ ), one can apply the **AT-block algorithm** presented in the work of Uchroński [29]. The computational complexity of this algorithm is  $O(n)$ . Considering further elements of permutation, after the last element of the first T-block, we can determine the next block of early tasks.

**Lemma 1.** If permutation  $\beta$  was generated from  $\pi \in \Phi$  by changing the order of elements in some block of early tasks in permutation  $\pi$  to

$$\mathcal{L}(\beta) = \mathcal{L}(\pi).$$

**Proof.** The proof results directly from the definition of the early task block and equality (10).  $\square$

#### 4.2. Blocks of Tardy Tasks

**Definition 2.** Subpermutation of tasks  $\pi^D$  in a permutation  $\pi \in \Phi$  is called a **D-block of tardy tasks**, if:

- (a') every task  $j \in \pi^D$  is tardy and  $d_j < E(\tilde{S}_{first} + \tilde{p}_j)$ , where  $\tilde{S}_{first}$  is a random variable—date of starting the execution of the first task from  $\pi^D$ ,
- (b')  $\pi^D$  is a maximum subpermutation meeting the constraint (a').

The following properties result directly from the tardy task block definition.

**Corollary 2.** If  $\pi^D = (\pi(a), \pi(a + 1), \dots, \pi(b))$  is a D-block in permutation  $\pi$ , then

1. the following inequality is met

$$\max\{d_j : j \in \pi^{\mathcal{D}}\} < E(\tilde{S}_{first}) + \min\{E(\tilde{p}_i) : i \in \pi^{\mathcal{D}}\}.$$

2. any task (belonging to  $\pi^{\mathcal{D}}$ ), swapped in the first position, in permutation  $\pi$  is tardy.

Similarly as in the  $\mathcal{T}$ -block, we determine the first  $\mathcal{D}$ -block in permutation  $\pi$ . After modifications (expected values of random variables  $E(\tilde{C}_i)$  instead of deterministic values  $C_i$  and  $E(\tilde{S}_{first})$  instead of  $S_{first}$ ), one can apply the **AD-block algorithm** presented in the work of Uchroński [29]. The computational complexity of this algorithm is  $O(n)$ . Considering further elements in permutation, after the last of the first  $\mathcal{D}$ -blocks, we can designate the next block of tardy tasks.

**Theorem 1** ([26]). *For any permutation  $\pi \in \Phi$  there is such division  $\pi$  into subpermutations in which each of them is:*

- (i)  $\mathcal{T}$ -block, or
- (ii)  $\mathcal{D}$ -block.

The algorithm for division of an  $n$ -elemental permutation into blocks, based on the proof of Theorem 1, has a computational complexity  $O(n)$ .

It follows from the block definition and the Theorem 1 that after the division of permutation into blocks:

1. Every task belongs to some  $\mathcal{T}$  or  $\mathcal{D}$  block.
2. Different blocks have different elements.
3. Two  $\mathcal{T}$  or  $\mathcal{D}$  blocks can appear directly next to each other.
4. A block can contain only one task.
5. The division of permutations into blocks is not interchangeable.

It is easy to show that the order of occurrence of tasks in the  $\mathcal{D}$ -block is not optimal due to the criterion (6). Later in this section we present, we are presenting a method for determining the optimal order of elements in the  $\mathcal{D}$ -block. First, we will prove some auxiliary lemma.

**Lemma 2.** *Let us assume that permutation  $\delta$  was generated from  $\pi \in \Phi$  by swapping the position of two random neighboring elements in the permutation. If for every  $i$  ( $i = 2, 3, \dots, n$ ) there is:*

$$f_{\pi(i-1)}(E(\tilde{C}_{\pi(i-1)})) + f_{\pi(i)}(E(\tilde{C}_{\pi(i)})) \leq f_{\pi(i-1)}(E(\tilde{C}_{\pi(i)})) + f_{\pi(i)}(E(\tilde{C}_{\pi(i)} - \tilde{p}_{\pi(i-1)})), \tag{11}$$

then

$$\mathcal{L}(\delta) \geq \mathcal{L}(\pi).$$

**Proof.** For the determination of attention, we assume that  $\delta$  was created from permutation  $\pi$  by swapping the positions of task  $\pi(i)$  z  $\pi(i - 1)$  ( $2 \leq i \leq n$ ), then

$$\delta(j) = \pi(j), \text{ therefore } E(\tilde{C}_{\delta(j)}) = E(\tilde{C}_{\pi(j)}),$$

$$\text{for } j = 1, 2, \dots, i - 2, i + 1, \dots, n$$

and

$$\delta(i - 1) = \pi(i), \delta(i) = \pi(i - 1),$$

$$E(\tilde{C}_{\delta(i-1)}) = E(\tilde{C}_{\pi(i)} - \tilde{p}_{\pi(i-1)}), E(\tilde{C}_{\delta(i)}) = E(\tilde{C}_{\pi(i)}).$$

Difference in the criterion value:

$$\begin{aligned} \mathcal{L}(\delta) - \mathcal{L}(\pi) &= \sum_{j=1}^n f_{\delta(j)}(E(\tilde{C}_{\delta(j)})) - \sum_{j=1}^n f_{\pi(j)}(E(\tilde{C}_{\pi(j)})) = \\ & \sum_{j=1}^{i-2} f_{\delta(j)}(E(\tilde{C}_{\delta(j)})) + f_{\delta(i-1)}(E(\tilde{C}_{\delta(i-1)})) + \\ & f_{\delta(i)}(E(\tilde{C}_{\delta(i)})) + \sum_{j=i+1}^n f_{\delta(j)}(E(\tilde{C}_{\delta(j)})) - \\ & - \left[ \sum_{j=1}^{i-2} f_{\pi(j)}(E(\tilde{C}_{\pi(j)})) + f_{\pi(i-1)}(E(\tilde{C}_{\pi(i-1)})) + \right. \\ & \left. f_{\pi(i)}(E(\tilde{C}_{\pi(i)})) + \sum_{j=i+1}^n f_{\pi(j)}(E(\tilde{C}_{\pi(j)})) \right] = \\ & f_{\delta(i-1)}(E(\tilde{C}_{\delta(i-1)})) + f_{\delta(i)}(E(\tilde{C}_{\delta(i)})) - \\ & f_{\pi(i-1)}(E(\tilde{C}_{\pi(i-1)})) - f_{\pi(i)}(E(\tilde{C}_{\pi(i)})). \end{aligned}$$

Using the permutation definition  $\delta$ , we obtain:

$$\begin{aligned} \mathcal{L}(\delta) - \mathcal{L}(\pi) &= f_{\delta(i-1)}(E(\tilde{C}_{\delta(i-1)})) + f_{\delta(i)}(E(\tilde{C}_{\delta(i)})) - \\ & f_{\pi(i-1)}(E(\tilde{C}_{\pi(i-1)})) - f_{\pi(i)}(E(\tilde{C}_{\pi(i)})) - \\ & = f_{\pi(i)}(E(\tilde{C}_{\pi(i)} - \tilde{p}_{\pi(i-1)})) + f_{\pi(i-1)}(E(\tilde{C}_{\pi(i)})) - \\ & f_{\pi(i-1)}(E(\tilde{C}_{\pi(i-1)})) - f_{\pi(i)}(E(\tilde{C}_{\pi(i)})) \geq 0. \end{aligned}$$

The last inequality follows from the assumption (11).  $\square$

**Corollary 3.** *If tasks  $\pi(i - 1), \pi(i)$  belong to some  $\mathcal{D}$ -block in permutation  $\pi$ , then inequality (11) takes the form:*

$$\frac{w_{\pi(i-1)}}{E(\tilde{p}_{\pi(i-1)})} \geq \frac{w_{\pi(i)}}{E(\tilde{p}_{\pi(i)})}. \tag{12}$$

**Lemma 3.** *Let  $B = (\pi(a), \pi(a + 1), \dots, \pi(b)), 1 \leq a < b \leq n$  be the  $\mathcal{D}$ -block in permutation  $\pi$ . If each pair of elements adjacent to  $B$  satisfies the relation (12), then the order in  $B$  is optimal for the tasks of the set  $\mathcal{Y}(B)$ , i.e.,*

$$\mathcal{L}(B) = \min\{\mathcal{L}(\gamma) : \gamma - \text{permutation of elements of the set } \mathcal{Y}(B)\}.$$

**Proof.** The proof of the Lemma 2 and the property (12) should be used, which by assumption is satisfied by every pair of elements from subpermutation  $B$ .  $\square$

**Definition 3.** *Let  $\mathcal{B}$  be the partition of permutation  $\pi$  into blocks. The  $\pi$  permutation is **ordered** (in short  $\mathcal{D}$ -OPT), if every  $\mathcal{D}$ -block of tasks satisfies the relation (12), so they appear in optimal order.*

**Corollary 4.** *Changing the order of tasks in any block of ordered permutation  $\pi$  does not generate permutations of a lower value of criterion function.*

We will now prove a theorem containing the necessary conditions, which must be met in order to generate a solution of the smaller criterion function value.

**Theorem 2.** Let  $\pi \in \Phi$  be the  $\mathcal{D}$ -OPT permutation. If  $\beta \in \Phi$  and

$$\mathcal{L}(\beta) < \mathcal{L}(\pi), \tag{13}$$

then in the permutation  $\beta$  at least one task of some block from division  $\pi$  was swapped before the first or last task of this block.

**Proof.** Let  $\mathcal{B} = [B^1, B^2, \dots, B^k]$  be a division of the ordered permutation  $\pi \in \Phi$  into blocks. Every block is a sequence of tasks

$$B^i = (\pi(a^i), \pi(a^i + 1), \dots, \pi(b^i)), \text{ for } i = 1, 2, \dots, k, \text{ where}$$

$$1 \leq a^1 \leq b^1 < a^2 \leq b^2 < \dots < a^k \leq b^k.$$

By

$$\mathcal{Y}^i(\pi) = \{\pi(a^i), \pi(a^i + 1), \dots, \pi(b^i)\},$$

we denote a set of tasks from block  $B^i$ .

Let permutation  $\beta \in \Phi$  and  $\mathcal{L}(\beta) < \mathcal{L}(\pi)$ . Let us assume indirectly that in permutation  $\beta$  no task from any block  $B^1, B^2, \dots, B^k$  was swapped before the first or last task of this block. Thus

$$\mathcal{Y}^i(\pi) = \mathcal{Y}^i(\beta), \quad i = 1, 2, \dots, k.$$

Therefore the sequence of tasks  $(\pi(a^i), \pi(a^i + 1), \dots, \pi(b^i))$  in permutation  $\pi$  and  $(\beta(a^i), \beta(a^i + 1), \dots, \beta(b^i))$  in  $\beta$  are permutations of the same subset of tasks  $\{\pi(a^i), \pi(a^i + 1), \dots, \pi(b^i)\}$ . It follows from Lemma 1 and 3 that in this case  $\mathcal{L}(\beta) \geq \mathcal{L}(\pi)$ , which is contrary to the assumption (13) of Theorem 3.  $\square$

The above theorem is the basis for the construction of a subneighborhood in algorithms based on the local improvement method.

### 5. Tasks Execution Times with Normal Distribution

We now consider the probabilistic Total Weighted Tardiness Problem PTWT. In order to simplify the notation, it was assumed that the order of execution of tasks is a natural permutation, i.e.,  $\pi = (1, 2, \dots, n)$ . Let  $(\tilde{p}_i, w_i, d_i)$  ( $i \in \mathcal{J}$ ) be an example of the problem data, where  $\tilde{p}_i \sim N(m_i, s_i)$  are random variables with normal distribution, and  $w_i$  and  $d_i$  are certain numbers. Using Graham's notation, this problem can be symbolically presented as  $1|\tilde{p}_i \sim N(m_i, s_i)|\sum w_i T_i$ . When determining the value of the criterion function (6) of the PTWT problem, one must compute the expected tardiness value  $\tilde{T}_i$ , i.e.,  $E(\tilde{T}_i)$ . First, we present some facts concerning distributions of random variables.

**Fact 1.** If task execution times are independent random variables with normal distribution  $\tilde{p}_i \sim N(m_i, s_i)$   $i \in \mathcal{J}$ , the dates of completing tasks are also random variables with normal distribution  $\tilde{C}_i \sim N(\mu_i, \sigma_i)$ , where

$$\mu_i = \sum_{j=1}^i m_j \quad \text{and} \quad \sigma_i = \sqrt{\sum_{j=1}^i s_j^2}. \tag{14}$$

In this case, tardiness is a random variable defined as follows:

$$\tilde{T}_i = \begin{cases} \tilde{C}_i - d_i, & \text{if } \tilde{C}_i > d_i, \\ 0, & \text{if } \tilde{C}_i \leq d_i. \end{cases}$$

**Lemma 4.** Cumulative distribution  $F_{\tilde{T}_i}$  of distribution of random variable  $\tilde{T}_i$  is expressed by the formula:

$$F_{\tilde{T}_i}(x) = \begin{cases} 0, & x \leq 0, \\ F_{\tilde{C}_i}(d_i + x) - F_{\tilde{C}_i}(d_i + x)F_{\tilde{C}_i}(d_i) + F_{\tilde{C}_i}(d_i), & x > 0, \end{cases}$$



**Proof.** Using the definition of the random variable  $\tilde{T}_i$  we consider two cases.

1.  $x > 0$ . Then

$$F_{\tilde{T}_i}(x) = P(\tilde{T}_i < x) = P(\tilde{T}_i < x | \tilde{C}_i - d_i > 0)P(\tilde{C}_i - d_i > 0) + P(\tilde{T}_i < x | \tilde{C}_i - d_i \leq 0)P(\tilde{C}_i - d_i \leq 0).$$

We can see two facts:

$$P(\tilde{T}_i < x | \tilde{C}_i - d_i \leq 0) = 1$$

and

$$P(\tilde{T}_i < x | \tilde{C}_i - d_i > 0) = P(\tilde{C}_i - d_i < x).$$

Next we have:

$$\begin{aligned} F_{\tilde{T}_i}(x) &= P(\tilde{C}_i - d_i < x)P(\tilde{C}_i - d_i > 0) + P(\tilde{C}_i - d_i \leq 0) = \\ &= P(\tilde{C}_i < d_i + x)P(\tilde{C}_i > d_i) + P(\tilde{C}_i \leq d_i) = \\ &= F_{\tilde{C}_i}(d_i + x)(1 - F_{\tilde{C}_i}(d_i)) + F_{\tilde{C}_i}(d_i) = \\ &= F_{\tilde{C}_i}(d_i + x) - F_{\tilde{C}_i}(d_i + x)F_{\tilde{C}_i}(d_i) + F_{\tilde{C}_i}(d_i). \end{aligned}$$

2.  $x < 0$ . It is obvious that  $F_{\tilde{T}_i}(x) = 0$ .  $\square$

In order to calculate the density function  $f_{\tilde{T}_i}(x)$  of random variable  $\tilde{T}$ , it is enough to calculate the derivative of the cumulative distribution function  $F_{\tilde{T}_i}(x)$ . Thus

$$f_{\tilde{T}_i}(x) = \begin{cases} 0, & \text{for } x \leq 0, \\ f_{\tilde{C}_i}(d_i + x) - F_{\tilde{C}_i}(d_i)f_{\tilde{C}_i}(d_i + x), & \text{for } x > 0. \end{cases}$$

Using the above lemma we will prove the theorem enabling the calculation of the expected value of the random variable  $\tilde{T}_i$ .

**Theorem 3.** *Expected value of the random variable  $\tilde{T}_i$*

$$E(\tilde{T}_i) = (1 - F_{\tilde{C}_i}(d_i)) \left( \frac{\sigma}{\sqrt{2\pi}} e^{-\frac{(d_i - \mu_i)^2}{2\sigma_i^2}} + (\mu_i - d_i) \left( 1 - F_{N(0,1)}\left(\frac{d_i - \mu_i}{\sigma_i}\right) \right) \right).$$

**Proof.** By definition of the expected value of a random variable

$$\begin{aligned} E(\tilde{T}_i) &= \int_{-\infty}^{\infty} x f_{\tilde{T}_i}(x) dx = \\ &= \int_0^{\infty} x (f_{\tilde{C}_i}(d_i + x) - F_{\tilde{C}_i}(d_i) f_{\tilde{C}_i}(d_i + x)) dx \\ &= \int_0^{\infty} x f_{\tilde{C}_i}(d_i + x) dx - \int_0^{\infty} x F_{\tilde{C}_i}(d_i) f_{\tilde{C}_i}(d_i + x) dx = \\ &= (1 - F_{\tilde{C}_i}(d_i)) \int_0^{\infty} x f_{\tilde{C}_i}(d_i + x) dx. \end{aligned} \tag{15}$$

By introducing the appropriate substitutions, we obtain:

$$E(\tilde{T}_i) = (1 - F_{\tilde{C}_i}(d_i)) \int_{d_i}^{\infty} (y - d_i) f_{\tilde{C}_i}(y) dy =$$

$$(1 - F_{\tilde{C}_i}(d_i)) \left( \int_{d_i}^{\infty} y f_{\tilde{C}_i}(y) dy - d_i \int_{d_i}^{\infty} f_{\tilde{C}_i}(y) dy \right).$$

The following equations are easy to prove:

$$\int_{d_i}^{\infty} y f_{\tilde{C}_i}(y) dy = \frac{\sigma_i}{\sqrt{2\pi}} e^{-\frac{(d_i - \mu_i)^2}{2\sigma_i^2}} + \mu_i \left( 1 - F_{N(0,1)}\left(\frac{d_i - \mu_i}{\sigma_i}\right) \right)$$

and

$$d_i \int_{d_i}^{\infty} f_{\tilde{C}_i}(y) dy = d_i \left( 1 - F_{N(0,1)}\left(\frac{d_i - \mu_i}{\sigma_i}\right) \right).$$

By inserting the above equations into the Expression (15) we obtain the theorem thesis:

$$E(\tilde{T}_i) = (1 - F_{\tilde{C}_i}(d_i)) \left( \frac{\sigma_i}{\sqrt{2\pi}} e^{-\frac{(d_i - \mu_i)^2}{2\sigma_i^2}} + (\mu_i - d_i) \left( 1 - F_{N(0,1)}\left(\frac{d_i - \mu_i}{\sigma_i}\right) \right) \right),$$

which ends the proof of the theorem.  $\square$

The above proved theorem enables quick calculation of the expected value of tardiness for random dates of the tasks' completion. Thanks to this, the cost of permutation  $\pi \in \Phi$  (the value of criterion function (6)) is:

$$\begin{aligned} \mathcal{L}(\pi) = E(\tilde{\mathcal{F}}(\pi)) &= \sum_{i=1}^n w_{\pi(i)} E(\tilde{T}_{\pi(i)}) = \\ &= \sum_{i=1}^n w_{\pi(i)} \left( 1 - F_{N(0,1)}\left(\frac{d_i - \mu_i}{\sigma_i}\right) \right) \left( \frac{\sigma_i}{\sqrt{2\pi}} e^{-\frac{(d_i - \mu_i)^2}{2\sigma_i^2}} + \right. \\ &\quad \left. + (\mu_i - d_i) \left( 1 - F_{N(0,1)}\left(\frac{d_i - \mu_i}{\sigma_i}\right) \right) \right). \end{aligned} \tag{16}$$

### 6. Tabu Search Algorithm

For solving NP-hard discrete optimization problems, approximate algorithms are almost exclusively used. The solutions determined by these algorithms are, from an application point of view, fully satisfactory (they often differ from the best solutions by less than 1%). They belong mostly to the local search methods, whose operation boils down to iterative browsing of a certain subset of acceptable solutions (neighborhood) and determining the best neighbor. One of the best implementations of this approach is tabu search algorithm. The work [18] presents such an algorithm solving the TWT problem. The tabu search algorithm is deterministic, so it guarantees repeatability of results. The application of block properties in the construction of an algorithm significantly improved its efficiency. For solving examples of a single-machine problem with random times of tasks execution PTWT we used a simplified version of this algorithm. Below, we briefly describe its basic elements.

#### 6.1. Moves and Neighborhoods

In each iteration of an algorithm based on a local search method using the neighborhood (moves) generator, a subset of a set of solutions, neighborhood is determined. If the solutions are permutations, most often the swap-type (*s-move*) and insert (*i-move*) are used. The first move swaps positions of several elements in permutation, and the second moves the element to a different position.

Let

$$\mathcal{B} = [B^1, B^2, \dots, B^v],$$

be a division of the ordered ( $\mathcal{D}$ -OPT) permutation  $\pi$  into blocks. The  $m$  move made on permutation  $\pi$  is called improving if it generates permutation  $m(\pi)$  with a lower value of criterion function. It follows from Theorem 2 that moves consisting of swapping the order of elements in any block of  $\mathcal{D}$ -OPT permutation are not improving moves. We now consider task  $\pi(j)$  belonging to a certain block from the division  $\mathcal{B}$  of permutation  $\pi$ . Moves that can bring improvement consist of swapping the task  $\pi(j)$  (*before*) the first or (*after*) the last task of this block. Let  $\mathcal{M}_{bf}^j(\pi)$  and  $\mathcal{M}_{af}^j(\pi)$  be sets of these moves. By

$$\mathcal{M}(\pi) = \bigcup_{j=1}^n \mathcal{M}_{bf}^j(\pi) \cup \bigcup_{j=1}^n \mathcal{M}_{af}^j(\pi), \tag{17}$$

we designate the set of all moves that can bring improvement, i.e., moves before or after blocks of some  $\pi$  permutation. Since the procedure for splitting permutations into blocks is not unambiguous, hence the set of moves  $\mathcal{M}(\pi)$  is not explicitly defined and depends on the considered split. Formally, in the definition (17) there should be a symbol identifying a specific division. To simplify the description, it is omitted.

For a fixed split  $\mathcal{D}$ -OPT of permutation  $\pi \in \Phi$  into blocks, the set of solutions

$$\mathcal{N}(\pi) = \{m(\pi) : m \in \mathcal{M}(\pi)\} \tag{18}$$

is *subneighborhood*, generated with the use of “block elimination properties”.

The procedure for determining subneighborhood (including the elimination of certain moves generating permutations that do not improve the value of the objective function) has the complexity of  $O(n^2)$ . Its use has significantly improved the efficiency of the algorithm solving the PTWT problem.

### 6.2. Tabu List

In order to prevent a cycle (returning to the same permutation after a certain number of algorithm iterations), some attributes of each move are stored on the list of prohibited moves  $L_{TS}$ . It operates on the principle of a FIFO queue. Performing the move, swapping the element  $\pi(r)$  to the position  $j$ , generating from  $\pi \in \Phi$  permutation  $\beta$  we save the attributes of this move on the tabu list threefold  $(\pi(r), j, \mathcal{L}(\beta))$ .

Let us assume we are considering the  $m \in \mathcal{M}(\beta)$ , swapping task  $\beta(k)$  to position  $l$ , generating from  $\beta \in \Phi$  permutation  $\gamma$ . If on the list there is threefold  $(r, j, \Psi)$  such that  $\beta(k) = r, l = j$  and  $W(\gamma) \leq \Psi$ , such a move is prohibited and removed from the set  $\mathcal{M}(\beta)$ . The only parameter of the list is its length, i.e., the number of remembered elements. In the literature, there are many descriptions of implementations of the tabu list (Algorithm 1).

**Algorithm 1** Tabu Search (TS)**Input:**

$\pi$ —start permutation;  
 $L_{TS} = \emptyset$ —tabu list;

**Output:**

$\pi^*$ —best solution;

**repeat**

Determine the subneighborhood  $\mathcal{N}(\pi)$  of permutation  $\pi$  due to Equation (18);

Delete from  $\mathcal{N}(\pi)$  permutations forbidden by the list  $L_{TS}$ ;

Determine permutation  $\beta \in \mathcal{N}(\pi)$ , such that

$$\mathcal{L}(\beta) = \min\{\mathcal{L}(\delta) : \delta \in \mathcal{N}(\pi)\};$$

**if** ( $\mathcal{L}(\beta) < \mathcal{L}(\pi^*)$ ) **then**

$$\pi^* := \beta;$$

Place attributes  $\beta$  on the list  $L_{TS}$ ;

$$\pi := \beta$$

**until** (the completion condition is fulfilled).

Termination condition:

1. calculation time,
2. maximum number of iterations.

**7. Computational Experiments**

This section introduces a method of random generation of data, a measure of algorithm stability and the calculation results of two algorithms:

$TS_{\mathcal{P}}$ —probabilistic tabu search algorithm with entire neighborhood generated by the insert;  
 $TS_{\mathcal{P}+\mathcal{B}}$ —modified algorithm  $TS_{\mathcal{P}}$ , with the additional application of block elimination properties in the procedure of neighborhood generation.

The algorithms have been implemented in C++ and run on the “Bem” cluster in the Wrocław Network Supercomputer Center working under 64-bit operating system Scientific Linux 6.10 (Final) equipped with Intel Xenon processor a E5-2670 (2.3 GHz). Calculations were made on suitably modified examples of reference data for the problem  $1 || \sum w_i T_i$  included with the best currently known solutions on the OR-Library website [37]. The data was randomly generated for the number of tasks  $n = 40, 50$  i  $100$ . For each  $n$  there were 125 examples with varying degrees of difficulty designated. When running each of the three algorithms, the following assumptions were made:

- starting permutation:  $\pi = (1, 2, \dots, n)$ ,
- length of the tabu list:  $n$ .

A set of these 375 examples, called *deterministic data*, will be denoted by  $\mathcal{D}$ .

**7.1. Generating of Test Data**

Computational experiments were conducted on examples that were generated according to the standard method of generating popular benchmarks, proposed by Potts and Van Wassenhove [23], available on the website [37].

For a fixed number of tasks,  $n$  example of deterministic data

$$\eta = ((p_1, w_1, d_1), (p_2, w_2, d_2) \dots, (p_n, w_n, d_n))$$

is a sequence of  $n$  triples (task execution time, tardiness cost factor, requested completion date). Based on this, we set examples of *probabilistic data*

$$\tilde{\eta} = ((\tilde{p}_1, w_1, d_1), (\tilde{p}_2, w_2, d_2) \dots, (\tilde{p}_n, w_n, d_n)),$$

where task execution times  $\tilde{p}_i$  ( $i = 1, 2, \dots, n$ ) are independent random variables of normal distribution  $\tilde{p}_i \sim N(p_i, c \cdot p_i)$  ( $i = 1, 2, \dots, n$ ), and a coefficient  $c = 0.05, 0.1, 0.2$ , where  $c$  is

the random data equivalent of the *RDD* parameter in the method used to generate data for the single-machine problem with the total weighted tardiness criterion available on the OR-Library [37] website. For one deterministic example, we generate three examples of probabilistic data from set  $\mathcal{D}$ . In total, *probabilistic data set*  $\tilde{\mathcal{P}}$  has 1125 examples.

For probabilistic data, task execution times are random variables. The probabilistic algorithm (i.e., the algorithm PTWT problem), for a fixed example of probabilistic data, sets a solution task execution order (task permutation). Therefore we set the order of the tasks, but we do not know the specific values of the execution times because they are the implementation of random variables of normal distribution. Thus, they can have different values. We introduce a certain measure (stability is covered in the next chapter) solution resistance (more generally the algorithm with which the solution was determined). For this purpose, the so-called sets of *disturbed data* were generated.

Let

$$\tilde{\eta} = ((\tilde{p}_1, w_1, d_1), (\tilde{p}_2, w_2, d_2) \dots, (\tilde{p}_n, w_n, d_n)),$$

( $\tilde{\eta} \in \tilde{\mathcal{P}}$ ) be an example of probabilistic data. For this example, there were 100 examples of *disturbed data* generated by randomly determining task execution times. The set of these examples is denoted by  $\mathcal{Z}(\tilde{\eta})$ . An example of disturbed data  $\theta \in \mathcal{Z}(\tilde{\eta})$  takes the form of

$$\theta = ((p'_1, w_1, d_1), \dots, (p'_n, w_n, d_n)),$$

where the task execution time  $p'_i$  ( $i = 1, \dots, n$ ) is implementation of random variable  $\tilde{p}_i$  of normal distribution, i.e.,  $\tilde{p}_i \sim N(p_i, c \cdot p_i)$  ( $i = 1, 2, \dots, n$ )  $c = 0.05, 0.1, 0.2$ , where  $c$  is the random data equivalent of the *RDD* parameter in the standard method of generating popular single-machine benchmarks from the [37] website. A set of disturbed data will be denoted by  $\mathcal{Z}$ ,  $|\mathcal{Z}| = 112500$ .

### 7.2. Algorithm Stability

Let  $F$  be the solution value determined by the tested algorithm and  $F^*$  the value of the reference solution. Relative error of the solution  $F$

$$\delta = \frac{F - F^*}{F^*} 100\% \tag{19}$$

indicates by how many percent the solution determined by the algorithm is worse or better than the reference one. Taking into consideration a set of examples, we can then assess the quality of solution designated by the algorithm solutions, and thus indirectly—the quality of the algorithm.

Let  $\tilde{\eta}$  be an example of probabilistic data,  $\mathcal{Z}(\tilde{\eta})$  data set generated from  $\tilde{\eta}$  by disturbance of task execution times according to the assumed schedule. We have further:

$A_{ref}$ —algorithm designating reference solutions,

$A$ —algorithm whose resistance we are testing (in our case  $A_{\mathcal{P}}$  or  $A_{\mathcal{P}+\mathcal{B}}$ ),

$\pi_{(A,x)}$ —solution designated by algorithm  $A$  for data  $x$ ,  $F(\pi_{(A,x)}, y)$ —value of criterion function of solution  $\pi_{(A,x)}$  for the example  $y$ .

Then

$$\Delta(A, \tilde{\eta}, \mathcal{Z}(\tilde{\eta})) = \frac{\sum_{\varphi \in \mathcal{Z}(\tilde{\eta})} F(\pi_{A, \tilde{\eta}}, \varphi) - \sum_{\varphi \in \mathcal{Z}(\tilde{\eta})} F(\pi_{(A_{ref}, \varphi)}, \varphi)}{\sum_{\varphi \in \mathcal{Z}(\tilde{\eta})} F(\pi_{(A_{ref}, \varphi)}, \varphi)},$$

we call *resistance of solution*  $\pi_{(A, \tilde{\eta})}$  (designated by algorithm  $A$  for the example  $\tilde{\eta}$ ) on the set of disturbed data  $\mathcal{Z}(\tilde{\eta})$ .

If  $\tilde{\mathcal{P}}$  is a set of examples of probabilistic data of the examined problem, then the expression

$$S(A, \tilde{\mathcal{P}}) = \frac{1}{|\tilde{\mathcal{P}}|} \sum_{\tilde{\eta} \in \tilde{\mathcal{P}}} \Delta(A, \tilde{\eta}, \mathcal{Z}(\tilde{\eta})) \tag{20}$$

we call a *resistance coefficient* of algorithm A on set  $\tilde{\mathcal{D}}$ . The smaller its value, the more stable the solutions determined by the algorithm, i.e., random disturbances of data results in less change in the cost of the solution.

### 7.2.1. Algorithms' Efficiency

Computations of probabilistic algorithm  $TS_{\mathcal{P}}$  and its version with blocks  $TS_{\mathcal{P}+\mathcal{B}}$  were made on the examples from set  $\mathcal{D}$  (see Section 7). The received results were compared with the best currently known. For each example, the relative error  $\delta$ , as well as calculation time, were calculated. Average relative deviation values  $\delta_{apprd}$ , for individual groups of examples, are shown in Table 1. The computations time for a run was fixed at 60 s in each instance.

**Table 1.** Computational results for deterministic data [37] ( $t = 60$  s).

Instance Group	wt40		wt50		wt100	
	$TS_{\mathcal{P}}$	$TS_{\mathcal{P}+\mathcal{B}}$	$TS_{\mathcal{P}}$	$TS_{\mathcal{P}+\mathcal{B}}$	$TS_{\mathcal{P}}$	$TS_{\mathcal{P}+\mathcal{B}}$
000–025	6.66	0.12	1.43	0.04	1.77	0.22
026–050	5.27	0.75	17.07	0.09	230.38	23.62
050–075	1.93	0.80	2.28	1.29	2.16	1.23
076–100	42.77	5.92	9.25	3.70	64.54	7.81
101–125	0.45	1.23	1.27	2.85	0.36	0.39
Average	11.42	1.76	6.26	1.60	59.84	6.65

As expected for all groups of examples, the average relative error is smaller for the algorithm with blocks. The difference is particularly noticeable for large-scale examples. For  $n = 100$  tasks, the relative error of the algorithm without blocks is 59.84%, and with blocks only 6.65%, for the same calculation time of one minute for a single test instance.

### 7.2.2. Algorithms' Stability

The main purpose of the conducted computational experiments was to examine the individual stability of individual algorithms, i.e., resistance of solutions determined by these algorithms for random changes (disturbances) of parameters.

Let  $\mathcal{D}$  be a set of deterministic data,  $\tilde{\mathcal{D}}$  the corresponding set of probabilistic data, and  $\mathcal{Z}$  the set of disturbed data. Based on this data, the stability coefficients of individual algorithms were designated. Comparative results are shown in Tables 2–4 (best values in a group are marked in bold).

**Table 2.** Computational results  $S(A, \tilde{\mathcal{D}})$  for the wt40 instance group.

Instance Group	$c = 0.05$		$c = 0.1$		$c = 0.2$	
	$TS_{\mathcal{P}}$	$TS_{\mathcal{P}+\mathcal{B}}$	$TS_{\mathcal{P}}$	$TS_{\mathcal{P}+\mathcal{B}}$	$TS_{\mathcal{P}}$	$TS_{\mathcal{P}+\mathcal{B}}$
000–025	0.0175	<b>−0.0170</b>	0.0183	<b>0.0044</b>	0.0912	<b>0.0850</b>
026–050	<b>0.0087</b>	0.0221	3.7938	<b>3.1483</b>	0.2869	<b>0.2023</b>
050–075	0.0009	<b>−0.0036</b>	0.0247	<b>0.0177</b>	2.4646	<b>2.2704</b>
076–100	0.0009	<b>−0.0288</b>	20.6039	<b>18.0087</b>	0.2451	<b>0.1578</b>
101–125	<b>0.0013</b>	0.0033	<b>0.4440</b>	0.7487	13.2248	<b>12.6221</b>
Average	0.0059	<b>−0.0048</b>	4.9769	<b>4.3856</b>	3.2625	<b>3.0675</b>

In general, the Tabu search algorithm with blocks is much more resistant to disturbed data disorders. The comparative results generated for 3 groups of examples with the size  $n = 40$  (Table 3),  $n = 50$  (Table 4), and  $n = 100$  (Table 5) clearly indicate that for almost every group of examples the value of the resistant coefficient is lower for the  $TS_{\mathcal{P}+\mathcal{B}}$  algorithm compared to the value of this coefficient for the  $TS_{\mathcal{P}}$  algorithm. The only exceptions are the groups of examples 026–050,  $c = 0.05$  and  $c = 0.1$ ; 010–125,  $c = 0.05$  and  $c = 0.1$  for  $n = 40$ ; 026–050,  $c = 0.05$ ,  $c = 0.1$  and 101–125,  $c = 0.05$  for  $n = 50$ ; 101–125,  $c = 0.1$  and 076–100,

$c = 0.2$  for  $n = 100$ , but they do not affect significantly the overall value of the immunity coefficient. The negative value of the coefficient is related to receiving better than reference solutions by the tested algorithm.

**Table 3.** Computational results  $S(A, \tilde{P})$  for the *wt50* instance group.

Instance Group	$c = 0.5$		$c = 0.1$		$c = 0.2$	
	$TS_{\mathcal{P}}$	$TS_{\mathcal{P}+\mathcal{B}}$	$TS_{\mathcal{P}}$	$TS_{\mathcal{P}+\mathcal{B}}$	$TS_{\mathcal{P}}$	$TS_{\mathcal{P}+\mathcal{B}}$
000–025	0.0049	−0.0070	0.0263	<b>0.0169</b>	<b>0.0826</b>	0.0862
026–050	<b>0.0148</b>	0.2466	<b>0.1426</b>	0.1878	0.3270	<b>0.1983</b>
050–075	0.0146	−0.0064	0.0595	<b>0.0078</b>	0.2674	<b>0.1840</b>
076–100	0.0054	−0.0256	0.0077	−0.0037	0.1847	<b>0.1720</b>
101–125	<b>0.0138</b>	0.0217	0.0277	<b>0.0265</b>	0.6870	<b>0.6292</b>
Average	<b>0.0107</b>	0.0459	0.0528	<b>0.0471</b>	0.3097	<b>0.2540</b>

**Table 4.** Computational results  $S(A, \tilde{P})$  for the *wt100* instance group.

Instance Group	$c = 0.5$		$c = 0.1$		$c = 0.2$	
	$TS_{\mathcal{P}}$	$TS_{\mathcal{P}+\mathcal{B}}$	$TS_{\mathcal{P}}$	$TS_{\mathcal{P}+\mathcal{B}}$	$TS_{\mathcal{P}}$	$TS_{\mathcal{P}+\mathcal{B}}$
000–025	0.0028	−0.0040	0.0199	<b>0.0157</b>	0.0855	<b>0.0761</b>
026–050	15.0490	<b>5.3410</b>	0.0482	<b>0.0478</b>	0.3703	<b>0.3258</b>
050–075	0.0131	−0.0017	0.0282	<b>0.0122</b>	19.6031	<b>17.7367</b>
076–100	0.0642	−0.0294	0.8347	<b>0.7095</b>	<b>0.3933</b>	0.4549
101–125	0.0596	<b>0.0345</b>	<b>0.3063</b>	0.3850	1.1980	<b>1.0438</b>
Average	3.0377	<b>1.0681</b>	0.2475	<b>0.2341</b>	4.3300	<b>3.9275</b>

A summary of the results obtained is given in Table 5 (the best values are marked in bold). The stability coefficients of both algorithms increase with the increase of parameter  $c$ . This parameter has a direct impact on the variance of the distribution from which the data are drawn. The  $TS_{\mathcal{P}+\mathcal{B}}$  algorithm has a lower stability factor, so the solutions determined by this algorithm are more resistant to data disturbances. In addition, this algorithm, at the same time of calculations, determines much better solutions than the  $TS_{\mathcal{P}}$  algorithm (see Table 1). For the number of tasks  $n = 100$ , the relative error is almost 10 times smaller. Summing up, the use of blocks in the tabu search algorithm, compared with the algorithm without blocks, gives a significant improvement in the value of the determined solutions and their stability.

**Table 5.** Comparative computational results of  $S(A, \tilde{P})$ —summary.

	$c = 0.05$		$c = 0.1$		$c = 0.2$	
	$TS_{\mathcal{P}}$	$TS_{\mathcal{P}+\mathcal{B}}$	$TS_{\mathcal{P}}$	$TS_{\mathcal{P}+\mathcal{B}}$	$TS_{\mathcal{P}}$	$TS_{\mathcal{P}+\mathcal{B}}$
<i>wt40</i>	0.0059	−0.0048	4.9769	<b>4.3856</b>	3.2625	<b>3.0675</b>
<i>wt50</i>	<b>0.0107</b>	0.0459	0.0528	<b>0.0471</b>	0.3097	<b>0.2540</b>
<i>wt100</i>	3.0377	<b>1.0681</b>	0.2475	<b>0.2341</b>	4.3300	<b>3.9275</b>
Average	1.0181	<b>0.3697</b>	1.7591	<b>1.5556</b>	2.6341	<b>2.4163</b>

### 8. Summary

The paper presents a certain method representing uncertain data in discrete optimization problems, using independent random variables of normal distribution. This distribution is in practice widely used for solving many non-deterministic decision problems. Consideration of uncertain data leads to great difficulty in terms of the calculations of practical optimization problems which, however, significantly better describe reality than deterministic models.

Also presented is the construction of the algorithm based on the tabu search method for the problem of scheduling tasks on a single machine with minimization of the sum of penalties for tasks not completed on time. Random execution times tasks are represented by independent random variables of normal distribution. Since the objective function is a random variable, thus, when choosing an element from the neighborhood as a comparative criterion the expected value was assumed. To accelerate the calculation, blocks are used, i.e., certain specific methods of an intermediate review of solutions for the problem under consideration. Computational experiments were conducted out in order to test the stability of the algorithms, i.e., the impact of altering task parameters on changes in the value of a criterion function. The obtained results clearly indicate that probabilistic algorithms are much more stable, i.e., algorithms in which the randomness of parameters is taken into account. The use of blocks significantly accelerated the calculations and greatly improved the stability of the algorithm. The described methods and algorithms can be directly applied to other probability distributions and also other task parameters (not only execution times). They can also be adopted in methods that solve problems with uncertain parameters represented by fuzzy numbers, especially in case of the so-called “triangular fuzzy numbers”.

Further research directions will be focused on adapting the developed properties to solve cyclical problems, minimizing the cycle time, as the most frequent case of large-scale manufacturing in practice.

**Author Contributions:** Conceptualization, W.B., P.R. and M.W.; methodology, M.W., and P.R.; software, M.U.; validation, W.B.; formal analysis, M.W.; investigation, M.W., and M.U.; resources, P.R.; data curation, M.U.; writing—original draft preparation, M.W.; writing—review and editing, W.B.; supervision, P.R.; project administration, M.U.; funding acquisition, W.B., and M.U. All authors have read and agreed to the published version of the manuscript.

**Funding:** The calculations were carried out using resources provided by the Wrocław Centre for Networking and Supercomputing (<http://wcss.pl>, accessed on 28 January 2023), grant No. 96.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following notations are used in this manuscript:

$n$	number of tasks
$\mathcal{J}$	set of tasks
$p_i$	task's execution time
$w_i$	task's tardiness cost ratio
$d_i$	demand task's execution date
$\pi$	tasks permutation
$\Phi$	a set of all elements' permutations from $\mathcal{J}$
$\mathcal{N}(\pi)$	solution neighborhood $\pi$
$S_i$	start date of task $i \in \mathcal{J}$
$C_i$	end date of task $i \in \mathcal{J}$
$\tilde{p}_i$	random variable of execution time of task $i$
$\tilde{T}_i$	random variable of tardiness of task $i$
$\pi^{\mathcal{T}}$	semi-block of early tasks
$\pi^{\mathcal{D}}$	semi-block of tardy tasks
$\mathcal{L}(\pi)$	sum of tardiness costs (criterion)

## References

1. Shang, C.; You, C. Distributionally robust optimization for planning and scheduling under uncertainty. *Comput. Chem. Eng.* **2018**, *110*, 53–68. [[CrossRef](#)]
2. Zhang, L.; Lin, Y.; Xiao, Y.; Zhang, X. Stochastic single-machine scheduling with random resource arrival times. *Int. J. Mach. Learn. Cybern.* **2018**, *9*, 1101–1107. [[CrossRef](#)]
3. Shaked, M.; Shandhkumard, R. (Eds.) *Stochastic Order*; Academic Press: San Diego, CA, USA, 1994.



4. Zhu, X.; Cai, X. General Stochastic Single-Machine Scheduling with Regular Cost Functions. *Math. Comput. Model.* **1997**, *26*, 95–108. [[CrossRef](#)]
5. Van den Akker, M.; Hoogeveen, R. *Minimizing the Number of Late Jobs in Case of Stochastic Processing Times with Minimum Success Probabilities*; Technical Report; Institute of Information and Computation Science, Utrecht University: Utrecht, The Netherlands, 2004.
6. Van den Akker, M.; Hoogeevee, R. Minimizing the number of late jobs in a stochastic setting using chance constraint. *J. Sched.* **2008**, *11*, 59–69. [[CrossRef](#)]
7. Vondrák, J. Probabilistic Methods in Combinatorial and Stochastic Optimization. Ph.D. Thesis, MIT, Cambridge, MA, USA, 2005.
8. Dean, B.C. Approximation Algorithms for Stochastic Scheduling Problems. Ph.D. Thesis, MIT, Cambridge, MA, USA, 2005.
9. Soroush, H.M. Scheduling stochastic job on a single machine minimize weighted number of tardy jobs. *Kiwait J. Sci.* **2013**, *40*, 123–147.
10. He, X.X.; Yao, C.; Tang, Q.H. Robust Single Machine Scheduling with Stochastic Processing Times Based on Event Point. *Appl. Mech. Mater.* **2014**, *668–669*, 1641–1645. [[CrossRef](#)]
11. Prade, H. Using fuzzy set theory in a scheduling problem. *Fuzzy Sets Syst.* **1979**, *2*, 153–165. [[CrossRef](#)]
12. Ishii, H. Fuzzy combinatorial optimization. *Jpn. J. Fuzzy Theory Syst.* **1992**, *4*, 31–40. [[CrossRef](#)]
13. Iscibuch, H.; Yamamoto, N.; Misaki, S.; Tanaka, H. Local Search Algorithm for Flow Shop Scheduling with Fuzzy Due-Dates. *Int. J. Prod. Econ.* **1994**, *33*, 53–66. [[CrossRef](#)]
14. Itoh, T.; Ishii, H. Fuzzy due-date scheduling problem with fuzzy processing times. *Int. Trans. Oper. Res.* **1999**, *6*, 639–647. [[CrossRef](#)]
15. Bocewicz, G.; Nielsen, I.E.; Banaszak, Z.A. Production flows scheduling subject to fuzzy processing time constraints. *Int. J. Comput. Integr. Manuf.* **2016**, *29*, 1105–1127. [[CrossRef](#)]
16. Rajba, P.; Wodecki, M. Stability of scheduling with random processing times on one machine. *Appl. Mathematica* **2012**, *39*, 169–183. [[CrossRef](#)]
17. Bożejko, W.; Hejducki, Z.; Wodecki, M. Flowshop scheduling of construction processes with uncertain parameters. *Arch. Civ. Mech. Eng.* **2019**, *19*, 194–204. [[CrossRef](#)]
18. Bożejko, W.; Grabowski, J.; Wodecki, M. Block approach-tabu search algorithm for single machine total weighted tardiness problem. *Comput. Ind. Eng.* **2006**, *50*, 1–14. [[CrossRef](#)]
19. Rinnooy Kan, A.H.G.; Lageweg, B.J.; Lenstra, J.K. Minimizing total costs in one-machine scheduling. *Oper. Res.* **1975**, *25*, 908–927. [[CrossRef](#)]
20. Bożejko, W.; Rajba, P.; Wodecki, M. Stable scheduling of single machine with probabilistic parameters. *Bull. Polish Acad. Sci. Tech. Sci.* **2017**, *65*, 219–231. [[CrossRef](#)]
21. Lawler, E.L.; Moor, J.M. A Functional Equation and its Applications to Resource Allocation and Sequencing Problems. *Manag. Sci.* **1969**, *16*, 77–84. [[CrossRef](#)]
22. Sahni, S.K. Algorithms for Scheduling Independent Jobs. *J. Assoc. Comput. Match.* **1976**, *23*, 116–127. [[CrossRef](#)]
23. Potts, C.N.; Van Wassenhove, L.N. Single Machine Tardiness Sequencing Heuristics. *IIE Trans.* **1991**, *23*, 346–354. [[CrossRef](#)]
24. Villareal, F.J.; Bulfin, R.L. Scheduling a Single Machine to Minimize the Weighted Number of Tardy Jobs. *IEE Trans.* **1983**, *15*, 337–343. [[CrossRef](#)]
25. Urgo, M.; Váncza, J. A branch-and-bound approach for the single machine maximum lateness stochastic scheduling problem to minimize the value-at-risk. *Flex. Serv. Manuf. J.* **2018**, *31*, 472–496. [[CrossRef](#)]
26. Wodecki, M. A Branch-and-Bound Parallel Algorithm for Single-Machine Total Weighted Tardiness Problem. *Adv. Manuf. Technol.* **2008**, *37*, 996–1004. [[CrossRef](#)]
27. Nowicki, E.; Smutnicki, C. A Fast tabu serach algorithm for permutation flow shop problem. *Eur. J. Off. Oper. Res.* **1996**, *91*, 160–175. [[CrossRef](#)]
28. Grabowski, J.; Wodecki, M. A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion. *Comput. Oper. Res.* **2004**, *31*, 1891–1909. [[CrossRef](#)]
29. Uchroński, M. Parallel Algorithm with Blocks for a Single-Machine Total Weighted Tardiness Scheduling Problem. *Appl. Sci.* **2021**, *11*, 2069. [[CrossRef](#)]
30. Wodecki, M. A block approach to earliness-tardiness scheduling problems. *Adv. Manuf. Technol.* **2009**, *40*, 797–807. [[CrossRef](#)]
31. Xiaoqiang, C.; Wu, X.; Zhou, X. *Optimal Stochastic Scheduling*; Springer-Verlag New York Inc.: New York, NY, USA, 2014.
32. Bożejko, W.; Hejducki, Z.; Rajba, P.; Wodecki, M. Project management In building process with uncertain tasks Times. *Manag. Prod. Eng. Rev.* **2011**, *2*, 3–9.
33. Bożejko, W.; Uchroński, M.; Wodecki, M. Block approach to the cyclic flow shop scheduling. *Comput. Ind. Eng.* **2015**, *81*, 158–166. [[CrossRef](#)]
34. Bożejko, W.; Wodecki, M. Solving Permutational Routing Problems by Population-Based Metaheuristics. *Comput. Ind. Eng.* **2009**, *57*, 269–276. [[CrossRef](#)]
35. Bożejko, W.; Pempere, J.; Uchroński, M.; Wodecki, M. Parallel Block-Based Simulated Annealing for the Single Machine Total Weighted Tardiness Scheduling Problem. In Proceedings of the 16th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2021), Bilbao, Spain, 22–24 September 2021; Advances in Intelligent Systems and Computing Book Series (AISC); Springer: Berling/Heidelberg, Germany, 2022; Volume 1401, pp. 758–765.

36. Bożejko, W.; Rajba, P.; Wodecki, M. Blocks of jobs for solving two-machine flow shop problem with normal distributed processing times. In Proceedings of the 15th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2020), Burgos, Spain, 16–18 September 2020; Advances in Intelligent Systems and Computing Book Series (AISC); Springer: Berlin/Heidelberg, Germany, 2021; pp. 289–298.
37. OR Library. Available online: <https://www.brunel.ac.uk/~mastjib/jeb/info.html> (accessed on 22 March 2023).

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.