

Distributed Manufacturing as a Scheduling Problem



Jarosław Pempera , Czesław Smutnicki , and Robert Wójcik 

Abstract In this chapter, we consider a distributed production system modeled and analyzed by means of the extended open-shop scheduling problem with transport times and the makespan criterion. The stated problem is more general than those considered by us so far as well as those found in the literature. The stated problem has been decomposed into two scheduling sub-problems: processing of jobs in each factory and transfer of jobs between factories. These schedules can be represented by two job orders: “processing order” and “transfer order”. The overall aim of the optimization task is stated to find the optimal/best job orders. We provide the mathematical model to find the non-delay schedule for the given job orders. Next, we formulate a graph model and introduce special properties, suitable to check the feasibility of job orders as well as to calculate the makespan value. The properties have been used in the neighborhood dedicated for local search methods. We have implemented the tabu search algorithm and tested experimentally its performance.

1 Introduction

We consider so-called distributed production system, modeled and analyzed by using tools from the scheduling area, namely the extended open-shop scheduling problem

J. Pempera (✉)

Department of Control Systems and Mechatronics, Faculty of Electronics,
Wrocław University of Science and Technology, 27 Wybrzeże Wyspiańskiego St., 50-372
Wrocław, Poland
e-mail: jaroslaw.pempera@pwr.edu.pl

C. Smutnicki · R. Wójcik

Department of Computer Engineering, Faculty of Electronics,
Wrocław University of Science and Technology, 27 Wybrzeże Wyspiańskiego St., 50-372
Wrocław, Poland
e-mail: czeslaw.smutnicki@pwr.edu.pl

R. Wójcik

e-mail: robert.wojcik@pwr.edu.pl

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2021
G. Bocewicz et al. (eds.), *Performance Evaluation Models for Distributed Service Networks*, Studies in Systems, Decision and Control 343,
https://doi.org/10.1007/978-3-030-67063-4_3

with transport times and the makespan criterion. The extension consists in using various sets of operations for jobs and operation-dependent transport times. Formulated problem is more general than these considered up-to-now by us and more general than those considered in the literature. The novelty in the problem formulation is a mix of open-shop and job-shop with transport times. The novelty in the solution methods is the graph as well as a labelling procedure, which performs Critical Path Method (CPM) with detector of graph cycle. The stated problem have been decomposed into two sub-problems of scheduling: processing of jobs in each factory and transfer of jobs between factories. We show that these schedules can be represented by two job orders called respectively: “processing order” and “transfer order”. We propose a labelling procedure which transforms orders into appropriate non-delay schedules. The goal of optimization is to find the optimal/best job orders. We provide the mathematical model to find the non-delay schedule for the given job orders. Next, we formulate a graph model and introduce several special properties, which allows us to check the feasibility of job orders as well as to calculate the makespan value. The properties have been used in the neighborhood dedicated for local search methods. We have implemented the tabu search algorithm and tested experimentally its performance.

In the distributed manufacturing system, the production process is carried out in many locations mutually distant. The transport time between dispersed production stages has significant influence on the schedule designed for these cases, [3]. Currently, many companies are moving away from the traditional production method in an enterprise located in one place to distributed production implemented in many locations. There are many examples of industries in which such a process takes place, ranging from companies producing food products in which some production stages must be carried out at a safe distance from sanitary reasons and ending with the automotive industry in which final products in the form of cars are produced from many components produced in various locations. Another example of distributed production systems is the construction of buildings, in which buildings are built from prefabricated elements manufactured in many company branches located near the raw materials for production [16] and/or construction sites. Appropriate location of company branches increases the level of production efficiency while reducing production expectations and reducing, among others, production costs.

Two basic models of distributed systems are considered in the literature, i.e. distributed flexible flow shop (DFFS) and distributed flexible job shop (DFJS) systems. This is due to the fact that the vast number of production systems the technological route for each product is precisely defined. The least often considered are the production systems with open-shop policy in which the order of performing operations within one task can be any.

The flow production systems are the most common research object of scientists. They model the vast majority of manufacturing systems actually encountered. In addressing distributed flowshop scheduling problem where are many algorithms based on advanced local search methods (for example a tabu search algorithm Gao, Chen and Deng [8], a scatter search algorithm Naderi and Ruiz [20], a hybrid genetic algorithm Gao and Chen [7], iterated greedy algorithms Ruiz, Pan and Naderi [24]).

Determining an efficient schedule for job shop production system problems is definitely much more difficult. This is due to its NP-hardness and a huge number of solutions from which a significant fraction is unfeasible. As in the case of flow shop problems, for instances of problems with medium and large sizes, many heuristics algorithms based on local search are proposed. One of the first optimisation algorithms for distributed job shop problem is genetic algorithm proposed by Jia, Fuh, Nee and Zhang [14]. A hybrid genetic algorithm Hao-Chin and Tung-Kuan [12] and an agent-based fuzzy algorithm have recently been proposed Hsu, Kao, Ho and Lai [13].

The open-shop scheduling problem is a topic that has already been studied by many authors. Due to the NP-hardness of the problem for regular criteria, mainly heuristic algorithms based on various methods are proposed. One of the simplest algorithm of this type for total completion time minimisation is the polynomial algorithm proposed by Kyparisis and Koulamas (1997) [15]. Gueret and Prins (1998) [11] developed an original list scheduling heuristic algorithm enhanced by a local search improvement procedure.

Nowadays, the best heuristic algorithms provide good solutions for benchmark instances in short computation time. Liaw (2000) [17] introduces a hybrid genetic algorithm incorporates a local improvement procedure based on tabu search, Sha and Hsu (2008) [26] proposed a particle swarm optimization algorithm.

Other work focuses on criteria such as: late work Blazewicz et al. (2004) [2], total tardiness Naderi et al. (2010) [19], total tardiness and makespan Noori-Darvish and Tavakkoli-Moghaddam (2012) [25], makespan with release dates Bai and Tang (2013) [1] and cycle time Pempera and Smutnicki (2018) [23].

The chapter is organized as follows. Section 2 provides the mathematical description of the problem. Section 3 presents graph model and some properties of the problem. Optimisation algorithm is presented in Sect. 4, whereas the result of computation experiments shown in Sect. 5.

2 Mathematical Model

We begin from a brief descriptive introduction to the problem. The distributed manufacturing system working with open-shop policy (DMSO) one can consider as an ensemble of cooperating factories engaged to provide various products. Factories are located far away each other. A product corresponds to a production job composed of operations, each of which is processed in the different factory. Operations included in the job can be performed in any order, then the route of a job through factories is variable. Each factory can perform only one operation at a time.

We introduce the following mathematical model, see Table 1 for the list of symbols and their meaning. We denote by

$$\mathbf{M} = \{1, \dots, m\}$$

Table 1 List of symbols

Denotation	Meaning
Data	
N	Set of jobs $N = \{1, 2, \dots, n\}$
M	Set of manufacturing plants $M = \{1, 2, \dots, m\}$
n_j	Number of operations in job $j \in N$
p_i	Processing time of operation $i \in O$
v_i	Manufacturing plant for operation $i \in O$
d_{kl}	Distance between plants k and l , $k, l \in M$
Definitions of objects	
O_j	Set of operations of job j , $O_j = \{l_{j-1} + 1, \dots, l_j\}$
l_j	Number of operations of the first j jobs, $l_j = \sum_{s=1}^j n_s$
O	Set of all operations
M_k	Operations in plant $k \in M$, $M_k = \{i \in O : v_i = k\}$
o	Total number of operations
Variables	
S_i	Start time of operation $i \in O$
C_i	Completion time of operation $i \in O$
τ	Processing order of visiting plants, $\tau = (\tau_1, \dots, \tau_n)$
σ	Processing order of operations in a plant, $\sigma = (\sigma_1, \dots, \sigma_m)$

the collection of geographically distributed manufacturing plants (factories). The set of tasks (jobs) which have to be performed using M is denoted by

$$N = \{1, 2, \dots, n\}.$$

The job j is represented by the set of n_j independent operations indexed consecutively as follows

$$O_j = \{l_{j-1} + 1, \dots, l_j\},$$

where

$$l_j = \sum_{s=1}^j n_s$$

is the number of operations of the first j jobs. Note that n_j can be various for different jobs. *Set* means that all operations from O_j have to be processed, in an order, but only one operation at the time. We define the set of all operations in the natural way

$$O = \bigcup_{j \in N} O_j,$$

where the total number of operations is $|O| = o = l_n$. Operation $i \in O$ is processed in factory $v_i \in M$ in the duration $p_i > 0$. v_i and p_i are known. We know also the distance (transfer time) defined by a matrix

$$D = [d_{kl}]_{m \times m},$$

where d_{kl} is a distance between k -th and l -th plant, $k, l \in M$. Interruption of an operation performed in the factory is not allowed. Each factory processes at most one operation at a time. Buffers before and after factory have infinite capacity.

Overall schedule is defined by vector of start times

$$S = (S_1, S_2, \dots, S_o)$$

and completion times

$$C = (C_1, C_2, \dots, C_o).$$

Actually, since $C_i = S_i + p_i$, then either S or C is sufficient to represent the schedule. The schedule S can be found having defined the following processing orders

- order of performing operations inside the job (order in which job visits factories),
- order of performing operations in each factory.

Both types of orders can be expressed by a permutation or/and their composition. The processing order of visiting factories of the job $j \in N$ can be represented by the permutation

$$\tau_j = (\tau_j(1), \tau_j(2), \dots, \tau_j(n_j))$$

on the set O_j . We denote in the sequel

$$\tau = (\tau_1, \tau_2, \dots, \tau_n).$$

The processing order of operations in factory $k \in M$ can be represented by the permutation

$$\sigma_k = (\sigma_k(1), \sigma_k(2), \dots, \sigma_k(m_k))$$

of m_k operations from the set

$$M_k = \{j \in O : v_j = k\}.$$

We denote by analogy

$$\sigma = (\sigma_1, \sigma_2, \dots, \sigma_m).$$

The pair (σ, τ) introduces constraints on time events in the following form

$$S_{\tau_j(i)} \geq C_{\tau_j(i-1)} + d_{v_{\tau_j(i-1)}, v_{\tau_j(i)}}, \quad i = 2, \dots, n_j, \quad j = 1, \dots, n, \quad (1)$$

Table 2 Data for the example

Job	i	p_i	v_i	i	p_i	v_i	i	p_i	v_i	i	p_i	v_i
1	1	90	1	2	50	3	3	64	2	4	65	4
2	5	47	4	6	92	3	7	73	1	8	26	2
3	9	27	4	10	94	2	11	48	1	12	93	3
4	13	43	3	14	76	1	15	87	2	16	65	4

$$S_{\sigma_k(i)} \geq C_{\sigma_k(i-1)}, \quad i = 2, \dots, m_k, \quad k = 1, \dots, m, \quad (2)$$

$$S_i \geq 0, \quad i = 1, \dots, o, \quad (3)$$

Inequality (1) expresses that the start time of operation $\tau_j(i)$ has to be later then the completion time of its precedence operation $\tau_j(i-1)$ plus the transport time from factory $v_{\tau_j(i-1)}$ to factory $v_{\tau_j(i)}$. Inequality (2) describes relations between events of start and completion of operations processed in the same factory and means that start of the operation $\sigma_k(i)$ is possible only after completion of its precedence operation $\sigma_k(i-1)$. The constraint (3) is obvious. Equality (4) ensures that processing of operation $i \in \mathcal{O}$ cannot be interrupted

$$C_i = S_i + p_i, \quad i = 1, \dots, o. \quad (4)$$

We call processing order (σ, τ) feasible if exists schedule S and/or C so that (1)–(4) hold. For the given (σ, τ) we define the makespan as follows

$$C_{\max}(\sigma, \tau) = \max_{i \in \mathcal{O}} C_i. \quad (5)$$

The overall aim of the distributed open shop scheduling problem is to find σ^* and τ^* for which $C_{\max}(\sigma, \tau)$ is the smallest among all feasible (σ, τ)

$$C_{\max}(\sigma^*, \tau^*) = \min_{\sigma, \tau} C_{\max}(\sigma, \tau). \quad (6)$$

The problem can be perceived as the mixed open-shop/job-shop scheduling problem. Indeed, assuming $n_j = m$, $j \in N$ and transport time zero we obtain the traditional definition of the open-shop existed in the scheduling area. Assuming that transport time is zero and τ_j is fixed but can be various for various jobs, we obtain the well-known definition of the job-shop problem. Since both special cases, namely the open-shop and job-shop are NP-hard, the considered problem is also NP-hard. In order to show the problem in detail, we provide the following example.

Example. Four jobs have to be scheduled in a distributed open-shop system consisting of four factories. Each job is processed in each factory. The job indexes, operation indexes, processing times and factories assigned to the operations are given in

Table 3 Distances between factories

<i>Factory</i>	1	2	3	4
1	0	5	10	15
2	5	0	5	10
3	10	5	0	5
4	15	10	5	0

Table 2 (the data derive from 5th Taillard's benchmark of the open-shop [27]). The distances between factories are given in Table 3.

Three Gantt charts are displayed in Figs. 1A–C for various processing orders (σ, τ) and various transport times D . Chart A provides the optimal schedule with the makespan $C_{\max}(\sigma, \tau) = 295$ for data given in Table 2 and processing orders

$$\begin{aligned}\sigma &= ((11, 7, 1, 14), (3, 15, 10, 8), (13, 12, 6, 2), (5, 4, 16, 9)) \\ \tau &= ((3, 4, 1, 2), (5, 7, 6, 8), (11, 12, 10, 9), (13, 15, 16, 14))\end{aligned}\quad (7)$$

assuming zero transport time (case $d_{ij} = 0$). Chart B shows the schedule with the makespan $C_{\max}(\sigma, \tau) = 320$ built for the processing order (7) (the same as in case A) and distances taken from Table 3. Chart C shows the best found schedule with $C_{\max}(\sigma, \tau) = 306$ for the job order

$$\begin{aligned}\sigma &= ((11, 14, 1, 7), (3, 8, 10, 15), (13, 2, 6, 12), (5, 9, 16, 4)) \\ \tau &= ((3, 2, 1, 4), (5, 8, 6, 7), (11, 9, 10, 12), (13, 14, 16, 15))\end{aligned}\quad (8)$$

and distances from Table 3.

3 Application of the Graph

In order to introduce some special properties of the problem we convert mathematical model (1)–(4) to certain graph. For a fixed (σ, τ) we define a directed planar graph in the Activity-on-Node (AoN) modelling style, namely

$$G(\sigma, \tau) = (O, E(\sigma) \cup F(\tau)), \quad (9)$$

with the set of nodes O and the set of arcs $E(\sigma) \cup F(\tau)$. The node $i \in O$ represents an operation and has weight p_i . The set of arcs $E(\sigma) \cup F(\tau) \subseteq O \times O$ represents preceding constraints and consists of two subsets:

- following from the changeable order τ of visiting plants by a job,

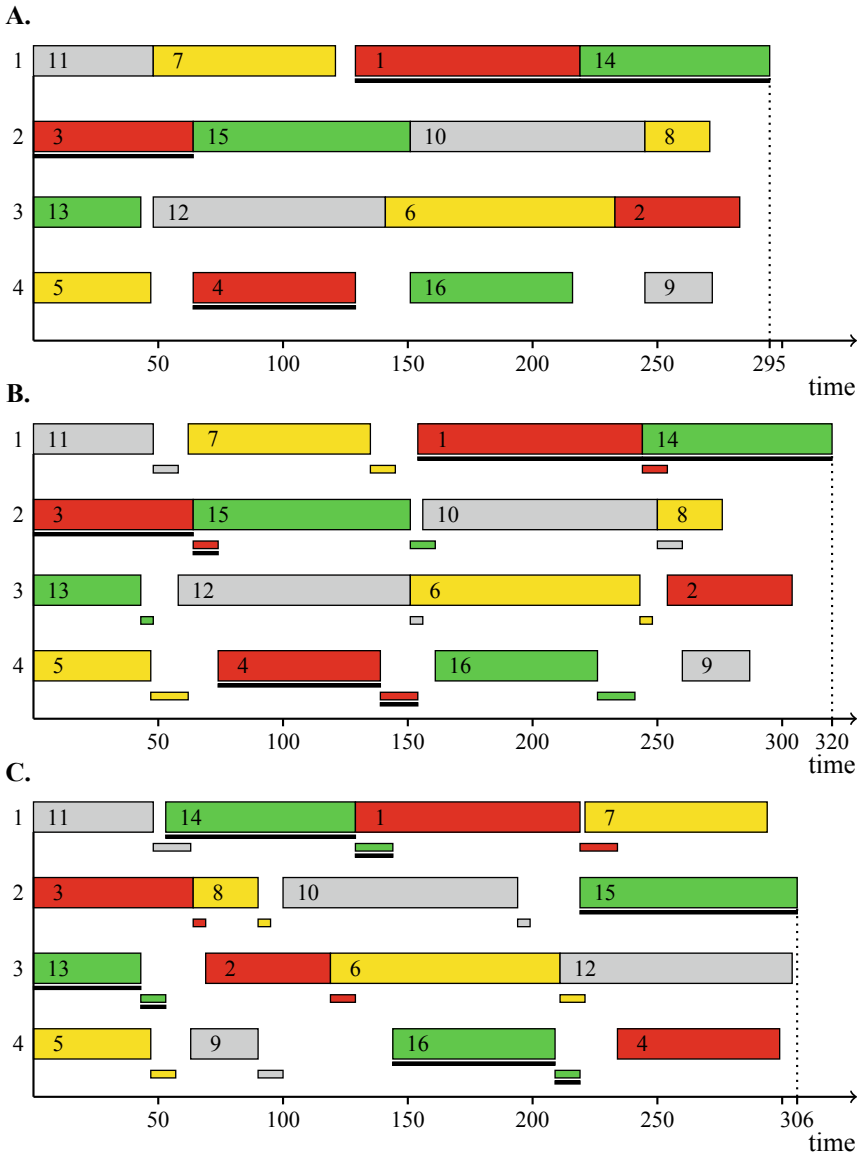


Fig. 1 Gantt charts for the Example 2: **A** optimal for order (7) and zero transport time ($d_{kl} = 0$), the makespan is 295. **B** certain for order (7) and transport time from Table 3, the makespan is 320. **C** optimal for order (8) and transport time given in Table 3, the makespan is 306

$$F(\tau) = \bigcup_{j=1}^n \bigcup_{i=2}^{n_j} \{(\tau_j(i-1), \tau_j(i))\}, \quad (10)$$

which corresponds to constraints (1), each arc $(\tau_k(i-1), \tau_k(i)) \in F(\tau)$ has weight $d_{\tau_k(i-1), \tau_k(i)}$,

- following from the changeable processing order σ of operations in factories

$$E(\sigma) = \bigcup_{k=1}^m \bigcup_{i=2}^{m_k} \{(\sigma_k(i-1), \sigma_k(i))\}, \quad (11)$$

which corresponds to constraints (2); each arc $((\sigma_k(i-1), \sigma_k(i)) \in E(\sigma)$ has weight zero.

Graph $G(\sigma, \tau)$ is used for checking feasibility, finding the schedule S and the makespan value for the given processing order (σ, τ) . Appropriate features follow from properties shown in the sequel.

Property 1 *For processing order (σ, τ) exists feasible schedule S , C from (1)–(4) if and only if the graph $G(\sigma, \tau)$ does not contain a cycle.*

Property 2 *For processing order (σ, τ) , the starting time S_i of an operation $i \in \mathcal{O}$ equals the length of the longest path going to the node $i \in \mathcal{O}$ in the graph $G(\sigma, \tau)$.*

Checking “if the graph has a cycle” in Property 1 can be done in the time $O(o)$. Longest paths going to each node in Property 2 can be found by running Critical Path Method (CPM); it requires also the time $O(o)$. Nevertheless, we propose an alternative labelling procedure, which combines features enumerated in Properties 1 and 2. Its running time is $O(o)$. Its pseudo-code is shown in Fig. 2.

Notice, Step 1.2 implements constraint (1) and Step 1.3 implements constraint (2). Then, before performing Step 1.4 in S_i we have the earliest possible start time of operation $i \in \mathcal{O}$. Step 1.4 implements Eq. (4). Performing Steps 1.2, 1.3 and 1.4 is possible only for those operations for which completion times of all predecessors are known. Operations with known completion time for all predecessors in a given iteration are stored in the queue Q . Queue Q is initialized with operations having no predecessors (see Step 0.2). For each operation we store in np_i , $i \in \mathcal{O}$ the number of predecessors with unknown (not determined yet) completion time. Each time when is determined completion time of some operation the np_i value for job and machine successors are updated and successors are stored in Q if np_i reach zero (Steps 1.5.1 and 1.6.1). If the completion time is not determined for all operations then the processing orders are unfeasible. This fact is detected by counting the number of operations for which the moment of completion is determined (see variable *itercount*).

The problem (6) refers to huge number of processing orders, theoretically $n^m m^n$, although the majority of them are unfeasible. Taking into account the cost of checking feasibility we are interested in the potential property allowing us to generate only

```

Step 0:      Initialization
Step 0.1:    Determine  $np_i$  for each operation  $i \in O$ 
Step 0.2:    Each operation  $i \in O$  with  $np_i = 0$  add to  $Q$ 
Step 0.3:     $itercount = 0$ 
Step 1:      While  $Q \neq \emptyset$ 
Step 1.1:     $i \leftarrow Q, S_i = 0$ 
Step 1.2:    if  $jp_i \neq 0$  then  $S_i = \max(S_i, C_{jp_i} + d_{jp_i, i})$ 
Step 1.3:    if  $fp_i \neq 0$  then  $S_i = \max(S_i, C_{fs_i})$ 
Step 1.4:     $C_i = S_i + p_i$ 
Step 1.5:    if  $js_i \neq 0$  then
Step 1.5.1:   $np_{js_i} = np_{js_i} - 1$ , if  $np_{js_i} = 0$  then  $Q \leftarrow js_i$ 
Step 1.5:    if  $fs_i \neq 0$  then
Step 1.6.1:   $np_{fs_i} = np_{fs_i} - 1$ , if  $np_{fs_i} = 0$  then  $Q \leftarrow fs_i$ 
Step 1.7:     $itercount = itercount + 1$ 
Step 2:      if ( $itercount < n$ )
Step 2.1:    return processing orders  $\sigma$  and  $\tau$  are unfeasible
              else
Step 2.2:    return  $S$  and  $C$ 

```

Fig. 2 Pseudocode of schedule determining algorithm for given (σ, τ)

feasible processing orders. The next two facts following from graph model, describe the conditions of generating the new feasible and better processing orders.

Let U be the arbitrary selected longest path in $G(\sigma, \tau)$. The maximum sub-path of U consisting of nodes representing operations performed in the same factory will be called the operation block or, briefly the block. The maximum sub-path of U consisting of nodes representing operations that belong to the same job will be called the job-block.

Schedule (σ, τ)

indirect variables (variables following from σ and τ):

- jp_i - job predecessor i.e. operation preceding operation i in τ ,
- js_i - job successor i.e. operation succeeding operation i in τ ,
- fp_i - factory predecessor i.e. operation preceding operation i in σ ,
- fs_i - factory successor i.e. operation succeeding operation i in σ ,

note that $jp_i = 0, js_i = 0, fp_i = 0, fs_i = 0$ if operation i has not appropriate predecessors or/and successors.

current variables

- np_i - number of predecessors of operation $i \in O$ with unknown (not yet determined) completion time,
- Q - queue of operations with determined completion times all of predecessors,
- $itercount$ - number of operations with determined completion times all of predecessors.

Property 3 *Let (σ, τ) be a feasible solution. All processing orders created by swapping the order of two adjacent operations in the blocks are feasible.*

Property 3 allows us to generate from a feasible processing order a sequence (subset) of feasible processing orders. It can be applied in an algorithm based on the B&B scheme as well as in various metaheuristics.

Property 4 *Let (σ', τ') be the feasible solution obtained from (σ, τ) so that*

$$C_{\max}(\sigma', \tau') < C_{\max}(\sigma, \tau) \quad (12)$$

then at least one condition occurs

- *at least at one operation from at least at one block is performed before the first operation of the block,*
- *at least at one operation from at least at one block is performed after the last operation of the block,*
- *at least at one operation from at least at one job-block is performed before the first operation of the job-block,*
- *at least at one operation from at least at one job-block is performed after the last operation of the job-block,*
- *the operation from at least at one internal job-block is performed in other order.*

Property 4 is a realization of so-called block approach. It defines sufficient conditions necessary to generate the processing order with better makespan. Block properties were successfully used to construct efficient algorithms for flow-shop scheduling problem [22], job-shop scheduling problem [10], flexible job-shop scheduling problem [5] and many others.

4 Optimization Algorithm

We refer to the metaheuristic algorithm called tabu search, see [9] for foundations of the method. Briefly, it is a modification of the well-known fast descend search approach. The search trajectory is generated by checking successive local neighborhoods. The algorithm uses a short-term memory (called tabu list) to prevent wondering around in the solution space and to guide the search into promising regions of this space. The best non-forbidden solution in the neighborhood is selected and it becomes current solution for the next iteration.

Since determining the C_{\max} value for a given solution order consumes a considerable amount of time, in the interest of computation time, one should search only feasible and/or promising improvement of the objective function solutions.

The most important element of local search algorithms is the neighborhood definition of the current solution. One of the most effective ways to generate the solution neighborhood for job-shop problem is the method proposed by Smutnicki and Nowicki [21]. Watson, Howe and Whitley (2005) [28] experimentally proved its natural

convergence to local minima regardless of the local search method used. Roughly speaking, the neighbor solution is generated by swapping the first operation in the block with the next or the last operation in the block with the second to last one.

Bearing in mind the similarity of the jobshop scheduling problem to the openshop scheduling problem (jobshop is a special case of openshop) and the fact that this method of generating fulfill the conditions of Property 4 (for blocks and job-blocks) and Property 3 (only for blocks) we have decided to implement this type of neighborhood generation method. Precisely, let $B = (B_1, \dots, B_b)$ be the sequence of blocks and job-blocks. The neighborhood $N(\sigma, \tau)$ of solution (σ, τ) consists of two subsets:

$$N(\sigma, \tau) = N_1(\sigma, \tau) \cup N_2(\sigma, \tau). \quad (13)$$

The former set $N_1(\sigma, \tau)$ contains processing orders created from (σ, τ) by swapping the two first operations of the block or job-block i.e. operations $B_s(1)$ and $B_s(2)$, $s = 1, \dots, b$. The latter set $N_2(\sigma, \tau)$ contains of processing orders created from (σ, τ) by swapping the two last operations operations of the block or job-block i.e. operations $B_s(b_s - 1)$ and $B_s(b_s)$, $s = 1, \dots, b$, where b_s is the number of operations in block or job-block s .

The tabu list TL were constructed and used in the following ways:

- contains the pairs of operations,
- has limited length L and is serviced by FIFO rule,
- the pair (a, b) of TL prohibits swapping operations a and b .

The algorithm terminates after *maxiter* iterations. In addition, in order to diversify the search process, after each *nonimptiter* iterations without improving the best found solution, there is a random “jump” from this solution. The jump is also performed if a search cycle is detected. In the jump, *jumpiter* iterations are performed. In each iteration of the jump, the neighbor solution is randomly selected and it becomes the current solution in the next iteration. The first solution in this procedure is the best solution found so far by the TS algorithm.

5 Experimental Results

We set three research goals:

1. evaluate the quality of solutions generated by TS,
2. evaluate the impact of transport times on the value of the objective function,
3. measure the calculation time.

The benchmark set consist of 40 modified open-shop Taillard instances. For each original Taillard’s instance we generate symmetric distance matrix. The benchmark set contains 4 group of instances of size: $n \times m = 4 \times 4, 5 \times 5, 7 \times 7, 10 \times 10$. Each group contains 10 instances. TS algorithm were implemented in C# under Visual Studio 2010 environment. The machine used was a a PC with Intel I7 2.4GHz, under Windows 8.1 operating system with 8GB RAM.

In the TS, there are several control parameters. In the preliminary computer tests of efficiency of algorithm we determine the following value of parameters:

- $nonimpiter = 1,000$,
- $L = 7, \dots, 13$ - length of tabu cyclically increase by one each time jump is performed,
- $jumpiter = 5$.

Algorithm TS was run on $maxiter = 10,000$ iterations.

Relative percentage deviation is employed to evaluate the quality of solution generated by proposed algorithm.

$$PRD = \frac{C_{\max}(\sigma^{TS}, \tau^{TS}) - LB}{LB} \times 100\%, \quad (14)$$

were (σ^{TS}, τ^{TS}) is solution generated by TS algorithm and LB is lower bound of C_{\max} value for given instance.

We introduce a simple lower bound calculated as the maximum of factory workloads i.e.

$$LB = \max_{1 \leq k \leq m} \sum_{i \in O: v_i=k} p_i. \quad (15)$$

In the initial experiment we evaluated the quality of solution generated by our TS using original Taillard's instances of the open-shop scheduling problem. Note the the "classical" open-shop is a special case of DOSP i.e. with distances between factories equal zero. Optimal or near optimal values are known for the Taillard's instances. So, the PRD in this tests, can refer to optimal makespans.

The optimal or best found values of C_{\max} for Taillard's instances are depicted in column *Ref* in Table 4. The columns C_{\max} reports C_{\max} values for solution generated by TS and *PRD* columns percentage relative deviation. For small problem size (group 4×4), the average PRD is equal 0.82 and slightly increases with increasing size of problem, especially with increasing number of machines. The biggest average $PRD = 4.11$ we observe for group 7×7 . Quality of TS in pure open-shop we evaluate vicariously and briefly, because is is not our main topic. Average PRD of TS for all Taillard's instances is 2.64 and it is similar to 2.54 obtained by the best algorithm mentioned in [18]. Nevertheless, for the pure open-shop still exists more advanced algorithms, with PRD close to zero, [6].

Each Taillard's benchmark for the open-shop problem contains up to nines diverse jobs, therefore is considered as hard for many heuristics. On the other hand, especially in the mass production system we observe small, medium or large series of identical products. This incline us to introduce another benchmark set.

In order to examine the impact of travel time between factories and series size on the C_{\max} values, we generated 4 groups of instances with different distances. The distances are generated random in the range $(1, frac \cdot LB)$, were $frac \in \{0\%, 5\%, 10\%, 20\%\}$. Note, that distances for $frac = 10\%, 20\%$ are proportional to random generated distances for $frac = 5\%$. For each of group we

Table 4 Relative percentage deviation for open-shop benchmark instances

Instance	<i>Ref</i>	C_{\max}	<i>PRD</i>	Instance	<i>Ref</i>	C_{\max}	<i>PRD</i>
Group 4×4				Group 5×5			
tail01	193	193	0.00	tail11	300	307	2.33
tail02	236	239	1.27	tail12	262	269	2.67
tail03	271	272	0.37	tail13	323	333	3.10
tail04	250	252	0.80	tail14	310	321	3.55
tail05	295	295	0.00	tail15	326	332	1.84
tail06	189	193	2.12	tail16	312	318	1.92
tail07	201	203	1.00	tail17	303	308	1.65
tail08	217	217	0.00	tail18	300	304	1.33
tail09	261	268	2.68	tail19	353	363	2.83
tail10	217	217	0.00	tail20	326	328	0.61
Average			0.82	Average			2.18
Group 7×7				Group 10×10			
tail21	435	448	2.99	tail31	652	670	2.76
tail22	443	463	4.51	tail32	588	617	4.93
tail23	468	496	5.98	tail33	611	627	2.62
tail24	463	483	4.32	tail34	577	598	3.64
tail25	416	433	4.09	tail35	657	669	1.83
tail26	451	478	5.99	tail36	538	565	5.02
tail27	422	443	4.98	tail37	616	630	2.27
tail28	424	435	2.59	tail38	595	621	4.37
tail29	458	478	4.37	tail39	595	615	3.36
tail30	398	403	1.26	tail40	604	628	3.97
Average			4.11	Average			3.48

generate five subgroups (series) denoted $\{1, 2, \dots, 5\}$, respectively; in subgroup x each job from original Taillard instances is duplicated x times. Finally, we receive 20×40 instances of DOSP.

Table 5 presents the results obtained for different groups of instances. Results for “Series 1” confirm the supposition that transport between factories enlarges the makespan. The average *PRD* increases with increasing *frac* value (with increasing time of travel) and with increasing number of factories. For group 5% the *PRD* value varies from 8.95% to 20.97% while for group 20% the *PRD* value varies from 25.08% to 58.23%. Taking into account the number of factories, it can easily be seen that the increase in the value of *PRD* is near proportional to the increase in the number of factories. For example for group 5% and 5 factories the $PRD = 10.63\%$ and for 10 factories is near two times greater i.e. $PRD = 20.97\%$. Surprising result can be seen for “Series 2.5”. For the majority of these groups, the *PRD* value is 0. This means that the makespan is limited only by certain bottleneck factory and time

Table 5 Average percentage relative deviation and computation time for instance groups

	$n \times m$	Group 0%		Group 5%		Group 10%		Group 20%	
		<i>PRD</i>	<i>CPU</i>	<i>PRD</i>	<i>CPU</i>	<i>PRD</i>	<i>CPU</i>	<i>PRD</i>	<i>CPU</i>
Series 1	4 × 4	0.82	0.3	8.94	0.3	13.65	0.3	25.08	0.1
	5 × 5	2.18	0.6	10.63	0.5	17.55	0.5	31.80	0.2
	7 × 7	4.11	1.3	13.27	1.2	24.15	1.3	39.68	0.3
	10 × 10	3.48	3.0	20.97	3.1	32.81	1.0	58.23	0.3
	Average	2.65		13.45		22.04		38.69	
Series 2	8 × 4	0	0.3	0	0.3	0	0.3	0	0.5
	10 × 5	0	0.3	0	0.3	0	0.3	0	0.3
	14 × 7	0	0.4	0	0.4	0	0.5	0.02	0.5
	20 × 10	0	0.7	0	0.8	0.46	1.1	13.96	1.6
	Average	0		0		0.12		3.49	
Series 3	12 × 4	0	0.3	0	0.3	0	0.3	0	0.3
	15 × 5	0	0.4	0	0.4	0	0.4	0	0.4
	21 × 7	0	0.5	0	0.4	0	0.6	0	0.5
	30 × 10	0	0.8	0	1.1	0	1.3	3.13	3.0
	Average	0		0		0		0.78	
Series 4	16 × 4	0	0.4	0	0.3	0	0.4	0	0.4
	20 × 5	0	0.4	0	0.4	0	0.4	0	0.4
	28 × 7	0	0.6	0	0.5	0	0.5	0	0.6
	40 × 10	0	1.5	0	1.3	0	2.0	2.59	2.9
	Average	0		0		0		0.65	
Series 5	20 × 4	0	0.4	0	0.4	0	0.4	0	0.4
	25 × 5	0	0.4	0	0.5	0	0.4	0	0.4
	35 × 7	0	0.6	0	0.7	0	0.7	0	0.6
	50 × 10	0	1.7	0	1.9	0	2.2	1.89	3.5
	Average	0		0		0		0.47	

of other travels does not affect it. Nonzero *PRD* values are observed in groups of 10% and 20% for a large number of factories. *PRD* values quickly decreases if the Series index increases.

At the beginning of discussion about the computation time, it should be noted that the this time depends on the number of neighbors in the neighborhood in each iteration. This number, in order, depends on the distribution of blocks and job-blocks and is peculiar for each particular solution. In the “Series 1”, we observe that if the distance between factories increases, the calculation time decreases. This is due to the fact that the number of blocks and task blocks decreases as the impact of travel times on the total completion of all jobs increases. In the remaining series, in each group of examples of the same size, the calculation time is approximately the same. This time is relatively short and follows from a small number of blocks due to the dominance of the bottleneck factory. The computation time increases with increasing

number of jobs. This relationship is approximately linear and shows that with the increase in the number of jobs, the average number of blocks changes only slightly.

The largest observed computation time, for instances with largest sizes, does not exceed 4 s and it is accepted in practical applications.

6 Conclusions

A distributed manufacturing system has been studied, modelled and analysed as the mixed open-shop/jobshop scheduling problem with transport times and the makespan criterion. We decomposed the problem of finding the best schedule into two sub-problems: finding optimal schedule for given processing order and finding the best processing order. To solve the former sub-problem we use the specific graph, which allow us to formulate several special properties: feasibility of processing order, fast computing of the makespan and elimination of non-perspective solution by so-called block properties. The block property defines the necessary conditions for the processing order to find better order than the current one without the direct calculation of the makespan value. In this way, non-promising solutions easily can be eliminated.

To solve the latter sub-problem we propose the metaheuristic algorithm based on tabu search (TS) method. Block properties have been used to construct the neighborhood of TS. Computer experiments show that the proposed algorithm provides solutions close to optimal in a short time already for test instances (taken from the literature) of the open-shop case. For the distributed manufacturing case the similarity of jobs has an impact on the quality of generated solutions. For instances with series of identical jobs, algorithm TS frequently generates an optimal schedule. Moreover, we observe for job series that transport time has rather weak impact on the selection of the best solution.

The used approach can be extended to cover more complex distributed manufacturing structures and other manufacturing policies. Further natural extension of the research will be parallel variants of the proposed algorithm, [4].

References

1. Bai, D., Tang, L.: Open shop scheduling problem to minimize makespan with release dates. *Appl. Math. Model.* **37**(4), 2008–2015 (2013)
2. Blażewicz, J., Pesch, E., Sterna, M., Werner, F.: Open shop scheduling problems with late work criteria. *Discret. Appl. Math.* **134**(1), 1–24 (2004)
3. Bocewicz, G., Wójcik, R., Banaszak, Z.: Agvs distributed control subject to imprecise operation Times. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4953 LNAI, 421–430 (2008). https://doi.org/10.1007/978-3-540-78582-8_43
4. Bożejko, W., Wodecki, M.: Parallel genetic algorithm for the flow shop scheduling problem. *Lecture Notes in Computer Science* 3019, pp. 566–571. Springer (2004)

5. Bożejko, W., Uchroński, M., Wodecki, M.: Parallel hybrid metaheuristics for the flexible job shop problem. *Computers & Industrial Engineering* **59**(2), 323–333 (2010)
6. Fardin, A., Mehdi, H., F.: A novel hybrid genetic algorithm for the open shop scheduling problem. *The International Journal of Advanced Manufacturing Technology* **62** (5) 775–787, (2012)
7. Gao, J., Chen, R.: A hybrid genetic algorithm for the distributed permutation flowshop scheduling problem. *International Journal of Computational Intelligence Systems* **4** (4) 497–508, (2011)
8. Gao, J., Chen, R., Deng, W.: An efficient tabu search algorithm for the distributed permutation flowshop scheduling problem *International Journal of Production Research* **51** (3) 641–651, (2013)
9. Glover, F.: Tabu Search - Part I. *ORSA Journal on Computing* **1**(3), 190–206 (1989). <https://doi.org/10.1287/ijoc.1.3.190>
10. Grabowski, J., Wodecki, M.: A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion. *Comput. Oper. Res.* **31**(11), 1891–1909 (2004)
11. Guéret, C., Prins, C.: Classical and new heuristics for the open-shop problem: a computational evaluation. *Eur. J. Oper. Res.* **107**(2), 306–314 (1998)
12. Hao-Chin, C., Tung-Kuan, L.: Optimisation of distributed manufacturing flexible job shop scheduling by using hybrid genetic algorithms. *J. Intell. Manuf.* **28**(8), 1973–1986 (2017)
13. Hsu, C., Kao, B., Ho, V.L., Lai, K.R.: Agent-based fuzzy constraint-directed negotiation mechanism for distributed job shop scheduling. *Eng. Appl. Artif. Intell.* **53**, 140–154 (2016)
14. Jia, H., Fuh, J., Nee, A., Zhang, Y.: Integration of genetic algorithm and Gantt chart for job shop scheduling in distributed manufacturing systems. *Comput. Ind. Eng.* **53**(2), 313–320 (2007)
15. Kyparisis, G.J., Koulamas, C.: Open shop scheduling with maximal machines. *Discret. Appl. Math.* **78**(1), 175–187 (1997)
16. Li, J., Bai, S., Duan, P., Sang, H., Han, Y., Zheng, Z.: An improved artificial bee colony algorithm for addressing distributed flow shop with distance coefficient in a prefabricated system. *Int. J. Prod. Res.* **57**(22), 6922–6942 (2019)
17. Liaw, C.: A hybrid genetic algorithm for the open shop scheduling problem. *Eur. J. Oper. Res.* **124**(1), 28–42 (2000)
18. Naderi, B., FatemiGhomi, S.M.T., Aminnayeri, M., Zandieh, M.: A contribution and new heuristics for open shop scheduling. *Comput. Oper. Res.* **37**, 213–221 (2010)
19. Naderi, B., Ghomi, S.F., Aminnayeri, M.Z.M.: ieh: a study on open shop scheduling to minimise total tardiness. *Int. J. Prod. Res.* **49**(15), 4657–678 (2011)
20. Naderi, B., Ruiz, R.: A scatter search algorithm for the distributed permutation flowshop scheduling problem. *Eur. J. Oper. Res.* **239**(2), 323–334 (2014)
21. Nowicki, E., Smutnicki, C.: A fast taboo search algorithm for the job shop problem. *Manag. Sci.* **42**(6), 783–938 (1996)
22. Nowicki, E., Smutnicki, C.: A fast tabu search algorithm for the permutation flow-shop problem. *Eur. J. Oper. Res.* **91**(1), 160–175 (1996)
23. Pempera, J., Smutnicki, C.: Open shop cyclic scheduling. *Eur. J. Oper. Res.* **269**(2), 773–781 (2018)
24. Ruiz, R., Pan, Q., Naderi, B.: Iterated Greedy methods for the distributed permutation flowshop scheduling problem. *Omega* **83**, 213–222 (2019)
25. Samaneh, N., Reza, T.: Minimizing the total tardiness and makespan in an open shop scheduling problem with sequence-dependent setup times. *J. Ind. Eng. Int.* **8**(1), 25 (2012)
26. Sha, D., Hsu, C.: A new particle swarm optimization for the open shop scheduling problem. *Comput. Oper. Res.* **35**(10), 3243–3261 (2008)
27. Taillard, E.: Benchmarks for basic scheduling problems. *Eur. J. Oper. Res.* **64**(2), 278–285 (1993)
28. Watson, J., Howe, A.E., Whitley, L.D.: Deconstructing Nowicki and Smutnicki's i-TSAB tabu search algorithm for the job-shop scheduling problem. *Comput. Oper. Res.* **33**(9), 2623–2644 (2006)