A Novel Approach of Operation Sequencing Problem in Computer Aided Process Planning

Béla Illés[1], Imre Piller[2], Sándor Radeleczki[2], Tibor Tóth[3†], György Wagner[3]

[1]Logistical Institute, University of Miskolc, Miskolc-Egyetemváros, 3515 Miskolc, Hungary, altilles@uni-miskolc.hu

[2]Mathematical Institute, University of Miskolc, Miskolc-Egyetemváros, 3515 Miskolc, Hungary, matip@uni-miskolc.hu, matradi@uni-miskolc.hu

[3]Institute of Informatics, University of Miskolc, Miskolc-Egyetemváros, 3515 Miskolc, Hungary, toth@ait.iit.uni-miskolc.hu, wagner@iit.uni-miskolc.hu

Abstract: The paper presents a novel method for finding quasi-optimal linear sequences of operations in manufacturing processes. The objective is to produce a part by minimizing the sum of machine, setup and tool change costs. The considered approach is based on a fuzzy clustering process, where the clustering algorithm divides the problem into smaller, locally solvable parts, forming disjoint groups of manufacturing tasks. Whenever the groups are already formed, then the order between their elements is established by using a kind of sorting algorithm, inserting their elements into some chains. Joining these chains according to the induced ordering of the groups we obtain a linear ordering of the tasks of the manufacturing processes. The total cost of this manufacturing process represents a good approximation of the (expected) minimal cost. Some threshold values are used in the fuzzy classification algorithm, which results a more flexible, adaptable tool for similar kind of problems. A numerical example is also presented, which follows and illustrates the mentioned theoretical background and the calculation steps. The article provides a brief overview of a possible GNU/Octave implementation of the proposed method.

Keywords: CAM; CAPP; operation sequencing; linear order; precedence constraint; fuzzy clustering; order-congruence

1 Introduction and Preliminaries

The problem considered in our paper has an interdisciplinary character, being a problem of Computer Aided Process Planning (CAPP), which appears also in Dynamic Process-Planning and in Logistic Information Systems (see [3], [16] or [1]). CAPP is considered the key technology for computer aided design/manufacturing (CAD/CAM) integration. It is also a basic technology in

robotics. See for instance a paper about the generic sequencing problem in this domain, where the sequencing of various tasks is accomplished together with making choices on how the tasks are performed [35]. In any CAPP system, selection of the operation sequence is a basic activity for manufacturing a part economically [29]. The essence of operation sequencing is determining in what order to perform a set of selected operations such that the resulting order satisfies the precedence constraints established by both the part and operations [28], to produce a part by minimizing the sum of machine, setup and tool change costs. The combination of different choices and constraints makes process sequencing a combinatorial problem [26].

To determine the optimal sequence, the literature contains various techniques like branch and bound methods, linear programming, dynamics programming (see, e.g., Lin and Wang [17] or Koulamas [18]). In [13] an approach based on particular clusters in a partially ordered set is developed. As the operations ordering problem involves various interdependent constraints, it is difficult to solve this problem using only the mentioned techniques. In fact, it is classified as an (NP)-complete problem. Recently, most works applied metaheuristics for solving process sequencing problems. Bhaskara R. et al. [19], Mojtaba et al. [25], and Li et al. [20] applied genetic algorithms to generate the optimal sequence of manufacturing operations. Foerster et al. [21] and G. Nallakumarasamy and et. al. [27] used simulated annealing for order spread minimization in sequencing cutting patterns which can be considered a kind of generalized Traveling-Salesman Problem (TSP). Li et al. [22] investigated the application of constrained-based tabu search approach for optimization of process plans. Further, the problem was investigated by Krishna et al. [23] using ant colony algorithm and found that the computational time has considerably reduced. Guo et al. [24] applied particle swarm optimization for operation-sequencing problem and concluded that there is still potential for further improvement in computation efficiency and optimality if introducing new operators and characteristics of other algorithms. In [11] a clustering algorithm based on the transition cost between two operations is combined with a precedence constraint checking algorithm to obtain an optimal operation sequencing.

The paper considers a specific case of manufacturing sequencing problems. It assumes that we can fulfil manufacturing steps in sequential order, without any overlapping between the operations, and that there is a transitional cost between the operations. In practice, this cost can mean setup or logistic cost depending on any kind of resource, (for instance time or energy, see e.g. [28]). For simplicity, we regard a cost as an aggregated value of the possible specific cost types. This can describe the forbidden precedencies in the cost matrix by infinite values. The presented algorithm can be successfully applied also to solve problems of logistic nature, replacing the setup and tool changing costs with some shipping costs.

We note that manufacturing and logistics are nowadays inseparable activities [34] due to the globalized market, globalized production and service, and related supplier and distribution activities. The large-scale expansion of the products structure, their

dynamic change, the shortening of the product life cycles and the rapid increasing of the number of the suppliers on the global market, all these factors require prompt response, as well as reaching the necessary economic efficiency and improving the market position (see [31] or [32]). The resolution for the challenges of the markets from the part of manufacturing and logistics was provided by the innovations of Information Technology and Manufacturing Technology, moreover, the applications of new software solutions [30]. The Industry 4.0 and Logistics 4.0 make available high scale flexibility in manufacturing and logistics via the benefits of digitalization in the aim of fulfilling the needs of the market (see, e.g. [4], [5], [6]).

In our paper, a heuristic mathematical method is presented for realizing a rapid and economical response of small or medium-sized companies to market needs in the field of manufacturing activity sequencing. By using this method, the following production-specific parameters become manageable:

- minimization of the setup times and hence of the turnaround time of the production process based on specific criteria;
- minimization of material handling work during the production on the basis of specific technological aspects;
- optimization of the production costs incurred during the production process.

The final benefit of the presented method is that it can improve the competitive position of small and medium-sized businesses with relatively simple computational solutions. It is important to emphasize that the main advantage of our method is not the exceptional minimization of the costs, but the flexibility of the program.

The considered concrete issue is the following: We have to insert in a linear order the tasks of a manufacturing process which consists of different manufacturing operations (steps). Our goal is to form an "optimal linear order" of these tasks (cf. [33]). Let $\mathcal{W}$ stand for the n-element set of the necessary tasks and let $a \leq b$ express the fact that the task $b \in \mathcal{W}$ must be preceded in time by the task $a \in \mathcal{W}$. Of course, if $a \neq b$, then we can also use the strict order $a < b$. It is easy to see that in this way we obtain a partially ordered set $(\mathcal{W}, \leq)$. Let denote by $\prec$ the *covering relation* in this ordered set. Thus $a \prec b$ means that $a < b$ holds, and there is no $c \in \mathcal{W}$ satisfying $a < c < b$. A *solution* of the scheduling problem is a linear order of the elements of $\mathcal{W}$ (in the physical time) which extends the initially given partial order $\leq$. This order can be represented by a sequence R: $x_1, x_2, \dots, x_n$, where $x_1, x_2, \dots, x_n$ are the elements of $\mathcal{W}$. As consecutive members x_i and x_{i+1} of this order can be chosen only such operations which can be effectuated one after the other. This means that either $x_i \prec x_{i+1}$ is satisfied, or x_i and x_{i+1} are not comparable with respect to $\leq$ (i.e., their order of execution is indifferent). Conversely, if $a, b \in \mathcal{W}$ are operations which either satisfy $a \prec b$ or they are not comparable, then there exists a linear extension R of the partial order $\leq$ such that

a and b are consecutive elements in this linear order R (see e.g. [14]).

Any allowed transition from the task (manufacturing step) x_i on the task x_j always has a transaction cost (setup cost, or shipping cost) $c_{i,j}$ which can be expressed by a nonnegative real number. In case of the above linear order R let us denote these numbers by $c_{1,2}$, $c_{2,3}, \dots, c_{n-1,n}$, and let $c_{1,1}$ stand for the setup cost necessary to begin the first task x_1. Then the total cost corresponding to this linear extension R is the following:

$$T_R := c_{1,1} + c_{1,2} + c_{2,3} + \cdots + c_{n-1,n} = c_{1,1} + \sum_{i=1}^{n-1} c_{i,i+1}.$$

Our aim is to find a solution with a minimal total cost T_R, or more precisely, to find such a linear extension R of $\leq$ which corresponds approximates the minimal total cost "as much as possible" – and also satisfies at the same time some other necessary technological constraints. (The meaning of the expression "as much as possible" will be specified later.)

2 The Principles of the Proposed Solution

There are more methods for solving such a linear sequencing problem of manufacturing process planning. For instance, the application of restricted traveller salesman algorithm is a common part of these solutions (see, e.g. [13] or [12]). In this case, the costs of the allowed transitions are proportional with the distances between places (shops) visited one after the other. Unfortunately, this method has an exponential time complexity. Another common feature of these methods (which often are the parts of proprietary algorithms) is that they use a clustering process preceding the ordering (see [11], [7] or [13]). For instance, in case of this variant of restricted traveling salesman algorithm, the basic idea can be illustrated as follows.

Suppose that the shops visited by the traveling salesman are grouped in different (neighbouring) cities. For instance, suppose that the places $a_1, a_2, \dots, a_l$ are in the city A, the places $a_{l+1}, \dots, a_{l+p}$ are in another city B, ..., etc. In this case, it is worth to visit first the places situated in the same city (say A) and then the places situated in another city (say B), ..., and so on.

Analogously, we will partition the elements of the set $\mathcal{W}$ in some disjoint groups $C_1, C_2, \dots, C_k$ ensuring that two requirements are fulfilled. Namely,

- the groups contain tasks with relatively small setup costs (shipping cost) between them,
- the extension of the (initial) ordering $\leq$ induces a linear ordering between the groups $C_1, C_2, \dots, C_k$.

Of course, some further requirements can be also formulated, for instance, since

more workpieces will be processed, it could be advantageous if the processing times of the manufacturing items corresponding to a group do not differ significantly (see, e.g. [3]).

We will see that this grouping can be obtained only in two steps. If the groups are already formed, then the order between their elements will be established by using a kind of sorting algorithm (Algorithm 1). In this way the elements of the groups are inserted in some chains. Joining these chains according to the induced ordering of the groups we will obtain a linear ordering of the elements of $\mathcal{W}$, whose costs represents a good approximation of the (expected) minimal cost.

3 Preliminary Steps

As a first step we generate a square table $\mathcal{M}$ the rows and columns of which are headed by the elements of $\mathcal{W}$. (Hence $i = 1, \dots, n$ and $j = 1, \dots, n$, where $n = |\mathcal{W}|$). If the order of the tasks x_i and x_j is indifferent (they being incomparable with respect to the machining order $\leq$) or x_i has to precede directly x_j-t (i.e. $x_i \prec x_j$ a holds in the partially ordered set $(\mathcal{W}, \leq)$) then in the intersection of the row of x_i and of the column x_j is inserted the setup cost $c_{i,j}$ of the transition from x_i on x_j, i.e. we set $\mathcal{M}(i,j) := c_{i,j}$. In all other cases - i.e., in each case when x_i and x_j cannot be performed consecutively (this being not possible or not allowed,) in position (i,j) is inserted ∞, i.e., we set $\mathcal{M}(i,j) := \infty$. The definition of the partial order $\leq$ and of the covering relation $\prec$ on $\mathcal{W}$ will be explained in Section Implementation.

As it was mentioned before, the order of elements in each final group C_p is established by using a kind of sorting algorithm. For this we will need a minor $Y^{(p)}$ of the matrix $\mathcal{M}$, which is determined by the elements of the group C_p. (Here $1 \leq p \leq k$.) Let $C_p = \{x_{p_1}, x_{p_2}, \dots, x_{p_{m(p)}}\}$, then $Y^{(p)}$ is a matrix of size $m(p) \times m(p)$ (with $m(p) \leq n$), which is defined as follows:

$$Y^{(p)}(i,j) = \mathcal{M}(p_i, p_j), \text{ for all } i,j = 1, \dots, m(p). \tag{1}$$

The elements $Y^{(p)}(i,j)$ of $Y^{(p)}$ are considered as some weighted edges connecting the elements of C_p. We are considering all those possible permutations π of the elements of C_p which represent a linear extension of $\leq$ (more precisely, of the restriction of $\leq$) on the set C_p; The elements $x_{k_1}, x_{k_2}, \dots, x_{k_{m(p)}}$ of a mentioned permutation are selected in the following algorithm (see, e.g. [14]):

Algorithm 1

Set $X_1 = C_p$, $P_1 = (X_1, \leq)$, $M_1 = \min(P_1)$,

For $i = 1,2, \dots, m(p) - 1$

Choose $x_{k_i} \in M_i$,

Set $X_{i+1} = X_i - \{x_{k_i}\}$, $P_{i+1} = (X_{i+1}, \leq)$, and $M_{i+1} = \min(P_{i+1})$

End

Notice that $\min(P)$ is a nonempty set of all minimal elements of the poset $(P, \leq)$ and, in view of [14], any such a linear order $x_{k_1} \leq_R x_{k_2} \leq_R \ldots \leq_R x_{k_{m(p)}}$ is an extension of $\leq$. The cost $c(\pi)$ of such a permutation π is computed as the sum of their consecutive edges, i.e.: $c(\pi) = \sum_{i=1}^{m(p)-1} \mathcal{M}(k_i, k_{i+1})$.

If an edge with cost ∞ is met in the above sum, then $c(\pi)$ is not computed.

4 The Fuzzy Clustering Process

The reviewed literature shows clearly that we need a flexible clustering method. In what follows, we will use a fuzzy clustering method as a heuristic estimate to guide the search to a good sequence by capturing how "cost-friendly" are the transitions between tasks.

Based on the matrix $\mathcal{M}$, we will implement a fuzzy clustering algorithm, defining first a so-called fuzzy tolerance relation on the set $\mathcal{W}$.

First, we construct a symmetric matrix $\mathcal{K}$ with the same size n, which is defined as follows:

$$\mathcal{K}(i,j) = \mathcal{K}(j,i) := \min\{\mathcal{M}(i,j), \mathcal{M}(j,i)\}, \quad i,j = 1,\ldots,n. \tag{2}$$

Of course, this matrix $\mathcal{K}$ will contain the value $\mathcal{M}(i,j) = \infty$ exactly in that case when the jobs x_i and x_j are comparable, but neither $x_i \prec x_j$, nor $x_j \prec x_i$ holds (i.e., they cannot be executed consecutively, one just after the other). This is because, in the case when x_i and x_j are not comparable, a finite value is assigned both to $\mathcal{M}(i,j)$ and $\mathcal{M}(j,i)$ by the construction of $\mathcal{M}$.

Now we define another symmetric matrix $\mathcal{H}$ of size $n \times n$, whose elements will be generated from the elements of the matrix $\mathcal{K}$ as follows:

$$\mathcal{H}(i,j) = \begin{cases} 1, & \text{if } i = j; \\ 0, & \text{if } \mathcal{K}(i,j) = \infty; \\ a^{-\mathcal{K}(i,j)}, & \text{otherwise,} \end{cases} \tag{3}$$

where $a > 1$ is a fixed real number. Since the elements of this matrix $\mathcal{H}$ are from the interval $[0,1]$, we can attach to it a binary fuzzy relation ρ defined on the set $\mathcal{W} = \{x_1, x_2, \ldots, x_{n,}\}$, whose membership function $\mu_\rho: \mathcal{W} \times \mathcal{W} \longrightarrow [0,1]$ is defined as follows:

$$\mu_\rho(x_i, x_j) := \mathcal{H}(i,j). \tag{4}$$

Since μ_ρ is symmetric and $\mu_\rho(x_i, x_i) = 1,\ i = 1, \dots, n$, this is a so-called fuzzy tolerance relation on $\mathcal{W}$. The determining method of the least fuzzy equivalence containing relation ρ is well known and it can be effectuated in a favourable time. To explain it, we will need some related notions.

4.1 Elements of the Theory of Fuzzy Relations

A *binary fuzzy relation* ρ between the elements of a (crisp) set $X \neq \emptyset$ is defined as a pair (X, μ_ρ), where $\mu_\rho: X \times X \longrightarrow [0,1]$ is a function. The value $\mu_\rho(x, y)$ express the strength" of the fuzzy relation ρ between the elements $x, y \in X$, and μ_ρ is called the *membership function* of ρ.

A *fuzzy tolerance relation* is a binary fuzzy relation $\rho = (X, \mu_\rho)$ satisfying the properties:

$$\mu_\rho(x, x) = 1, \text{ for all } x \in X, \tag{5}$$

$$\mu_\rho(x, y) = \mu_\rho(y, x), \text{ for all } x, y \in X. \tag{6}$$

If ρ in addition satisfies the inequality

$$\mu_\rho(x, z) \geq \min\{\mu_\rho(x, y), \mu_\rho(y, z)\}, \text{ for all } x, y, z \in X, \tag{7}$$

it is called a *fuzzy equivalence* (see [2] or [9]). Let $\alpha \in [0,1]$. An α-cut of a fuzzy relation $\rho = (X, \mu_\rho)$ is a crisp relation ρ_α defined as follows:

$$\rho_\alpha := \{(x, y) \in X \times X | \mu_\rho(x, y) \geq \alpha\}.$$

If $\rho = (X, \mu_\rho)$ is a fuzzy equivalence, then ρ_α is a (crisp) equivalence relation on the set X. Let Π_α stand for the partition induced by ρ_α on X. It is easy to see that for any $\beta \in [0,1]$ with $\beta \geq \alpha$, the partition Π_β is a refinement of Π_α. Therefore, to any sequence $0 \leq \alpha_1 < \alpha_2 < \cdots < \alpha_k \leq 1$ we may attach a nested sequence of partitions $\Pi_{\alpha_1}, \Pi_{\alpha_2}, \dots, \Pi_{\alpha_k}$ of the set X. The *composition* of two fuzzy relations $\varphi = (X, \mu_\varphi)$ and $\theta = (X, \mu_\theta)$ is defined as a fuzzy relation $\varphi \circ \theta = (X, \mu_{\varphi \circ \theta})$, where for all $(x, z) \in X \times X$,

$$\mu_{\varphi \circ \theta}(x, z) := \sup\left\{\min\left(\mu_\varphi(x, y), \mu_\theta(y, z)\right) \middle| y \in X\right\}. \tag{8}$$

The *m-th power* of a fuzzy relation $\rho = (X, \mu_\rho)$ is defined inductively as $\rho^m := \rho^{m-1} \circ \rho$, where $m > 1$ and $\rho^1 := \rho$. The *transitive closure* of a fuzzy relation $\rho = (X, \mu_\rho)$ is the least fuzzy relation $\overline{\rho} = (X, \mu_{\overline{\rho}})$ satisfying the inequality (7) with $\mu_{\overline{\rho}}$ and $\rho \leq \overline{\rho}$. If ρ is fuzzy tolerance on X, then $\overline{\rho}$ always exists and it is a fuzzy equivalence. Now let X be a finite set with n elements and let $\rho = (X, \mu_\rho)$ be a fuzzy tolerance on X. It is well known that in this case there exists a number $k \leq n$ such that $\rho^k = \overline{\rho}$, and we have also $\rho^k = \rho^{k+m}$, for any positive integer m (see, e.g. [9], [15] or [10]).

In our case $X = \mathcal{W} = \{x_1, x_2, \dots, x_{n,}\}$, and we consider the fuzzy tolerance ρ

corresponding to the $n \times n$ type matrix $\mathcal{H}$, defined by the membership function $\mu_\rho: \mathcal{W} \times \mathcal{W} \longrightarrow [0,1]$,

$$\mu_\rho(x_i, x_j) := \mathcal{H}(i,j),\ 1 \leq i,j \leq n. \tag{9}$$

Computing the consecutive powers $\rho^2, \rho^3, \dots, \rho^k$, $(k \leq n)$ until $\rho^k = \rho^{k-1}$ we obtain the transitive closure $\overline{\rho} = \rho^k$ of the fuzzy tolerance ρ. We denote $\Phi = \overline{\rho} = \rho^k$. Clearly, in our case, the fuzzy equivalence relation Φ is obtained by calculating the consecutive powers $\mathcal{H}^1, \mathcal{H}^2, \dots, \mathcal{H}^k$, of the similarity matrix, where $\mathcal{H}^1 := \mathcal{H}$, and where the usual matrix product is replaced with the following "max-min product" (cf. [15]):

$$\mathcal{H}^{l+1}(i,j) = (\mathcal{H}^l \cdot \mathcal{H})(i,j) := \max_{1 \leq p \leq n} \{\min(\mathcal{H}^l(i,p), \mathcal{H}(p,j))\}. \tag{10}$$

This calculation is performed until repeating powers appear, i.e., $\mathcal{H}^k = \mathcal{H}^{k+1}$. The obtained fuzzy equivalence Φ is defined by the matrix $\mathcal{H}^k$, i.e., its membership function is:

$$\mu_\Phi(x_i, x_j) := \mathcal{H}^k(i,j),\ 1 \leq i,j \leq n. \tag{11}$$

In fact, Φ is expressing how closed are the tasks x_i and x_j, in other words, how "cost-friendly" is the transition from x_i to x_j. Clearly, any cut Φ_α of Φ corresponding to a number $\alpha \in [0,1]$ is a crisp equivalence on $\mathcal{W}$.

5 The Second Step: Generating a Linear Order-Congruence

Our aim is to insert the machining tasks (i.e., the elements of $\mathcal{W}$) into disjoint groups $C_1, C_2, \dots, C_k$, where inside these groups they might be ordered in several ways. However, the groups follow each other in a well-determined linear order, which is "compatible" with the initial partial order $\leq$. In other words, the order $\leq$ induces between the groups a linear order. Of course, in this case $\{C_1, C_2, \dots, C_k\}$ being a partition of the set $\mathcal{W}$, determines a unique equivalence relation ρ on $\mathcal{W}$-n, that equivalence whose classes are $C_1, C_2, \dots, C_k$. Hence the factor set of ρ is $\mathcal{W}/\rho = \{C_1, C_2, \dots, C_k\}$. Let us consider now that map $\varphi: \mathcal{W} \longrightarrow \mathcal{W}/\rho$, which maps any $a \in \mathcal{W}$ into the class C_a containing a. The mathematical meaning of the requirement that the order $\leq_\rho$ defined between the classes $C_1, C_2, \dots, C_k$ is "compatible" with the initial partial order $\leq$, is the condition that for any $a, b \in \mathcal{W}$ with $a \leq b$, the relation $C_a \leq_\rho C_b$ also holds, that is, $a \leq b$ implies $\varphi(a) \leq_\rho \varphi(b)$. An equivalence relation which satisfies such a property is called an order-congruence of the partially ordered set - its exact definition (see [8]) is given below:

Definition 1 (i) Let $(P, \leq)$ be a partially ordered set and $\rho \subseteq P \times P$ an equivalence on P. Denote by $[x]_\rho$ the class of ρ containing $x \in P$. If there exists

a partially order $\leq_\rho$ on the factor set P/ρ with the property that the mapping $\varphi: P \longrightarrow P/\rho,\ \varphi(x) = [x]_\rho$ is order-preserving with respect to $\leq_\rho$, that is, for any $a, b \in P,\ a \leq b$ implies $\varphi(a) \leq_\rho \varphi(b)$, then ρ is called an *order-congruence* and $\leq_\rho$ is said to be the *partial order induced on the factor set* P/ρ. In this case, $a\rho b \Leftrightarrow \varphi(a) = \varphi(b)$, in other words, the equivalence ρ is the same as the "kernel equivalence" of the function φ (i.e., $\rho = \mathrm{Ker}\varphi$).

(ii) The order congruence ρ is called *linear* if $\leq_\rho$ is a linear order on P/ρ.

In [8] is proved also that any block $[x]_\rho$ of an order congruence is convex, i.e., $a, b \in [x]_\rho$ and $a \leq c \leq b$ for any $c \in P$ imply $c \in [x]_\rho$.

In view of this definition, we are looking for linear order-congruences of the poset $(\mathcal{W}, \leq)$ of machining tasks. We have elaborated a method for generating them. This generalizes a method of László Eszes. The essence of the algorithm in [13] is the choosing of a set of minimal elements M_1. Then, a set M_2 of minimal elements is selected from $(\mathcal{W} \setminus M_1, \leq)$, and so on, until the set $\mathcal{W}$ is exhausted. It is not hard to prove that the equivalence corresponding to the partition $\mathcal{W} = M_1 \cup \ldots \cup M_k$ generated by this method is an order-congruence of $(\mathcal{W}, \leq)$. An *order ideal* of a partially ordered set $(P, \leq)$ is a nonempty subset $A \subseteq P$ with the property that for any $a \in A$ and $x \in P,\ x \leq a$ implies $x \in A$. Notice, that any order-ideal is a convex subset. The previous algorithm is modified by using the fuzzy equivalence obtained in the first step of our method as follows:

First, we consider a cut Φ_{α_1} corresponding to a number $\alpha_1 \in [0,1]$ of our fuzzy equivalence Φ, and take the $\Phi_{\alpha_1}[m_1]$ class of a minimal element m_1 from the poset $(\mathcal{W}, \leq)$. As set B_1 will be chosen such an order ideal of the poset $(\mathcal{W}, \leq)$, which contains m_1 and it is included in $\Phi_{\alpha_1}[m_1]$, i.e., $m_1 \in B_1$. The number α_1 and the class $\Phi_{\alpha_1}[m_1]$ is chosen in that way, to obtain a set B_1 as large as possible, but at least with three elements.

We consider a linear order of the elements of B_1 and denote its maximal element by m_1^*, and its predecessor element by m_1' ($m_1' \prec m_1^*$). Next, we restrict the fuzzy equivalence relation to $\mathcal{W}_1 = (\mathcal{W} \setminus B_1) \cup \{m_1^*\}$. We try to find a new α_2 value and a $\Phi_{\alpha_2}[m_1^*]$ class of $(\mathcal{W}_1, \leq)$, which provides the largest $B_2 \subseteq \Phi_{\alpha_2}[m_1^*]$. In the next step, we consider the set $\mathcal{W}_2 = \big(\mathcal{W} \setminus (B_1 \cup B_2)\big) \cup \{m_2^*\}$. We continue this procedure, while the set $\mathcal{W}$ has any element. It results the $C_1 = B_1 \setminus \{m_1^*\}, C_2 = B_2 \setminus \{m_2^*\}, \ldots, C_k = B_k$ disjoint clusters. Clearly, $\mathcal{W} = C_1 \cup C_2 \cup \cdots \cup C_k$ is a partition of $\mathcal{W}$. In Proposition 3, we will show that the sets $C_i,\ i = 1, \ldots, k$ are the blocks of an order-congruence of $(\mathcal{W}, \leq)$. As we mentioned before, we have determined the order of elements within the sets B_i by computing the weights of all possible linear orders (extending the initial partial order) and choosing the optimal one, and thereafter we joined the resulted chains. This means that, we join the *maximal element* m_i' of $C_i = B_i \setminus \{m_i^*\}$ (in step $i \in \{1, \ldots, k-1\}$), with the *minimal element* $m_{i+1}^{\#}$ of the chain C_{i+1}.

Clearly, in this way we obtain a linear order. Observe that choosing an order ideal B_j from the fuzzy equivalence class $\Phi[m_j]$, and ordering the elements of the group B_j in a linear order, can be done by the same algorithm, namely a variant of the Algorithm 1, as follows:

Algorithm 1a

Set $X_1 = \mathcal{W}_{j-1}$, $P_1 = (X_1, \leq)$, $M_1 = \{m_j\}$,

For $i = 1,2, \dots, |B_j|$ do

If $M_i = \emptyset$ print $x_1 < \cdots < x_{i-1}$ and Stop

Else Choose $x_i \in M_i$,

Set $X_{i+1} = X_i - \{x_i\}$, $P_{i+1} = (X_{i+1}, \leq)$, and

$M_{i+1} = \min(P_{i+1}) \cap \Phi[m_j]$

End

We note that whenever B_j has a small, limited size (e.g., $s = |B_j| \leq 6$), the linear orderings of its elements can be obtained by a simplified variant of this method: First, all the permutations of the elements of B_j are generated. Then we select those permutations $\pi = (x_{k_1}, x_{k_2}, \dots, x_{k_s})$ which have the property that each element x_{k_i}, $1 \leq i < s$ of them is a minimal one in the subposet $(P_i, \leq)$, where $P_i = \{x_{k_i}, x_{k_{i+1}}, \dots, x_{k_s}\}$. (Finally, as optimal permutation is chosen that one with the minimal cost.)

Algorithm 1b

Set $\pi = (x_{k_1}, x_{k_2}, \dots, x_{k_s}), P_1 = (\{x_1, x_2, \dots, x_s\}, \leq), d = \text{true}$

For $i = 1,2, \dots, |s| - 1$ do

If $x_{k_i} \in \min(P_i)$

Set $X_{i+1} = X_i - \{x_{k_i}\}$, $P_{i+1} = (X_{i+1}, \leq)$

Else Set $d = \text{false}$

End

Since the elements $x_1, \dots, x_{i-1}$ within the group B_j are generated in fact by Algorithm 1, the order $x_1 < \cdots < x_{i-1}$ always is a linear order which extends the restriction of $\leq$ to $\{x_1, \dots, x_{i-1}\}$. Let us observe that the final group B_j is an order ideal in the partially ordered set $(\mathcal{W}_{j-1}, \leq)$:

Indeed, let $B_j = \{x_1, \dots, x_s\}$, $s = |B_j|$, and take $b \in \mathcal{W}_{j-1}$ such that $b < x_k$, for some $x_k \in B_j$. Since x_k is a minimal element in the ordered set $P_k = \mathcal{W}_{j-1} \setminus \{x_1, \dots, x_{k-1}\}$, this is possible only in the case $b \in \{x_1, \dots, x_{k-1}\} \subseteq B_j$. Hence B_j is an order ideal in $(\mathcal{W}_{j-1}, \leq)$.

Now, we have to prove only that this linear order R induced by our method on $\mathcal{W}$ is an extension of the partial order $\leq$ defined initially on $\mathcal{W}$. This will be done in Proposition 3(ii). Observe, that by definition, $m_i^* R m_{i+1}^{\#}$ $i = 1, \dots, k-1$ always holds. We also claim that any element m_i^* *is maximal in the poset* $(B_i, \leq)$*, and it is minimal in the poset* $(\mathcal{W}_i, \leq)$, *where* $\mathcal{W}_i = \left(\mathcal{W} \setminus \left(\bigcup_{k=1}^{i} B_k\right)\right) \cup \{m_i^*\}$. Thus $\Phi_{\alpha_i}[m_i^*]$ is always an equivalence class of a minimal element in $\mathcal{W}_i$.

Indeed, suppose by contradiction that there is an element $b \in B_i$ such that $m_i^* < b$. Since Algorithm 1a preserves the existing order, we get $m_i^* R b$, in contradiction to the maximality of m_i^* with respect to R in B_i. Similarly, assume that there exists an element $a \in \mathcal{W}_i$ such that $a < m_i^*$. Since $m_i^* \in B_i$, $a \in \mathcal{W}_i \subseteq \mathcal{W}_{i-1}$ and B_i is an order ideal in $\mathcal{W}_{i-1}$, we obtain $a \in B_i \setminus \{m_i^*\}$, and the latter yields $a \notin \mathcal{W}_i$, which is a contradiction.

Proposition 3 (i) *The clusters* $C_1, C_2, \dots, C_k$ *are the classes of a linear order-congruence on* $(\mathcal{W}, \leq)$. (ii) R *is an extension of* $\leq$.

Proof. First, observe that our algorithm inserts the groups $C_p \subseteq \mathcal{W}$, $1 \leq p \leq k$ in a linear order $C_1 <_\rho C_2 <_\rho, \dots, <_\rho C_k$, corresponding to their indices. Denote by ρ the equivalence corresponding to the partition $\mathcal{W} = C_1 \cup C_2 \cup \cdots \cup C_k$, and let $\varphi: P \to P/\rho$ be the map that assigns to any $x \in \mathcal{W}$ the cluster containing it, i.e., its $[x]_\rho$ class.

(i) Take $a, b \in \mathcal{W}$ such that $a \leq b$, and suppose that $\varphi(a) = [a]_\rho = C_i$ and $\varphi(b) = [b]_\rho = C_j$. Then either $i \leq j$ or $j < i$ holds. In the first case $\varphi(a) = C_i \leq_\rho C_j = \varphi(b)$. If this holds for each $a \leq b$, then ρ is an order-congruence. We prove that the case $j < i$ is excluded. Indeed, $b \in C_j \subseteq \mathcal{W}_{j-1}$, hence for $j < i$ we get $a \in \mathcal{W}_{i-1} \setminus C_j$ and $\mathcal{W}_{i-1} \subset \mathcal{W}_{j-1}$. Because B_j is an order ideal in $(\mathcal{W}_{j-1}, \leq)$ and m_j^* is a maximal element in B_j, we get that $C_j = B_j \setminus \{m_j^*\}$ is also an order ideal in $(\mathcal{W}_{j-1}, \leq)$. As $b \in C_j$, $a \in \mathcal{W}_{j-1}$, now the hypothesis $a \leq b$ yields $a \in C_j$, a contradiction.

(ii) Take any $a, b \in \mathcal{W}$ with $a \leq b$, and $\varphi(a) = C_i$, $\varphi(b) = C_j$. If $i = j$ i.e., $C_i = C_j$, then the order of a and b is determined within the group C_i by Algorithm 1 which preserves $\leq$. Thus, we obtain aRb. If $i \neq j$, then in view of (i) in the proof above, we get $i < j$ and $C_i <_\rho C_j$. Since $a \in C_i$ and $b \in C_j$, we get $a R m_i'$ and $m_j^{\#} R b$. Then $a R m_i'$, $m_i' R m_{i+1}^{\#}$, $m_{i+1}^{\#} R m_{i+1}', \dots, m_{j-1}' R m_j^{\#}$, $m_j^{\#} R b$ imply aRb. Thus R is an extension of $\leq$. □

6 Implementation

For experimentation, we have implemented a simple user interface. Its main purpose is to provide a more convenient way for defining the order of manufacturing operations in the technological matrix. The user interface has been implemented in

JavaScript with HTML5 Canvas. It helps check that the provided relation is a partial order or not. The proposed algorithm has implemented in GNU/Octave. Therefore, it is compatible with MATLAB. The relations are represented in a matrix. The language makes available the infinity symbol (inf) for denoting the infinite weights in the matrix.

At the first step, we must set a partial order. In fact, as a first input a binary relation ρ is defined between the elements of $\mathcal{W}$: $x_i \rho x_j$ means that the task x_i must precede the task x_j in the technological process. Hence, ρ can be represented by a directed graph. Its vertices are the elements of $\mathcal{W}$. Then the reflexive- transitive closure R of this relation ρ is formed – this will be a partial order whenever the graph is acyclic. Thus, the input is considered correct only in this acyclic case (see, e.g. [7]). Now, by using the adjacency matrix of the partial order R, generating the covering relation (given in a so-called Hasse-diagram) corresponding to R is a well-known function.

We have to insert the technologically justified weights (or we can generate random weights for testing purposes) for the covering pairs of tasks. We must also provide some weights for the incomparable tasks. In our example, we have generated uniformly distributed random values on the interval $[1,120] \subset \mathbb{R}$.

We calculate the order from the weights of the technological matrix. We assume that the transitions between manufacturing operations are possible, where the weights are finite. All further steps of the algorithm are organized into separate functions. The purposes of them are the following.

- Calculate the symmetric matrix from the initial technological matrix.
- Calculate the similarity matrix from the symmetric matrix.
- Calculate the transitive closure.
- Calculate the appropriate threshold value according to the number of the resulted clusters.
- Use the threshold value for clustering. Find equivalence classes to find minimal indices.
- Eliminate row and column (when removing a task from matrix representation).
- Find shortest path starting from the given (minimal) task.
- Evaluate the cost of the linear order.

6.1 Some Illustrative Examples

We must note that, the proposed algorithm works properly without the elimination of unnecessary weights (which are not part of the Hasse diagram). We can see the matrix representation of the partial order in Table 1 and the corresponding weight matrix in Table 2. (We note that there is no restriction on the effectuation order of

the task O, therefore, it appears in the Hasse diagram as an element incomparable with the others.) The example below is from [12], pages 160-161, and it is related to a manufacturing process of a ventilation screw, where the effectuated operations were the following:

A, B: Longitudinal turning (roughing respectively smoothing)
C, E, N: Insertion
D, J: Edge chamfer
F: Stabbing
G, H, I, O: Drilling
K: Thread cutting
L: Thread rolling
M: Thread turning
Q: Thread drilling
P: Rubbing
R: Dosage-collision

Table 1
The adjacency matrix of the considered partial order

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
A	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	0
B	0	1	1	1	1	1	0	0	1	1	1	1	1	0	0	1	1	0
C	0	0	1	0	1	1	0	0	1	1	1	1	1	0	0	1	1	0
D	0	0	0	1	1	1	0	0	1	1	1	1	1	0	0	1	1	0
E	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	1	0
F	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
G	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0
H	0	0	0	0	1	1	0	1	1	1	1	1	1	0	0	1	1	0
I	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	1	0
J	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	1	0
K	0	0	0	0	1	1	0	0	1	1	1	0	0	0	0	1	1	0
L	0	0	0	0	1	1	0	0	1	1	0	1	0	0	0	1	1	0
M	0	0	0	0	1	1	0	0	1	1	0	0	1	0	0	1	1	0
N	0	0	0	0	1	1	0	0	1	1	1	1	1	1	0	1	1	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
P	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0
Q	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0
R	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1

Table 2
Edge weights in matrix form

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
A	∞	4	∞	∞	∞	∞	19	5	∞	∞	∞	∞	∞	6	47	∞	∞	96
B	∞	∞	11	12	∞	∞	∞	71	∞	∞	∞	∞	∞	47	88	∞	∞	∞
C	∞	∞	∞	67	∞	∞	∞	86	∞	∞	13	14	15	16	76	∞	∞	∞
D	∞	∞	55	∞	∞	∞	∞	7	∞	∞	16	17	18	34	102	∞	∞	∞

E	∞	∞	∞	∞	∞	∞	∞	∞	103	100	∞	∞	∞	∞	74	41	40	∞
F	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	60	∞	∞	∞
G	14	7	∞	∞	∞	∞	∞	8	∞	∞	∞	∞	∞	9	90	∞	∞	∞
H	∞	25	51	68	∞	∞	∞	∞	∞	∞	19	21	22	6	45	∞	∞	∞
I	∞	∞	∞	∞	90	∞	∞	∞	∞	44	∞	∞	∞	∞	120	45	44	∞
J	∞	∞	∞	∞	38	∞	∞	∞	43	∞	∞	∞	∞	∞	31	43	42	∞
K	∞	∞	∞	∞	30	∞	∞	∞	28	29	∞	65	54	∞	24	∞	∞	∞
L	∞	∞	∞	∞	33	∞	∞	∞	31	32	20	∞	17	∞	32	∞	∞	∞
M	∞	∞	∞	∞	38	∞	∞	∞	35	36	88	59	∞	∞	108	∞	∞	∞
N	∞	107	75	9	∞	∞	∞	107	∞	∞	24	25	26	∞	68	∞	∞	∞
O	14	48	37	96	6	38	87	36	58	19	109	75	18	17	∞	60	57	88
P	∞	∞	∞	∞	∞	47	∞	∞	∞	∞	∞	∞	∞	∞	107	∞	49	∞
Q	∞	∞	∞	∞	∞	48	∞	∞	∞	∞	∞	∞	∞	∞	44	51	∞	∞
R	119	∞	∞	∞	∞	∞	3	∞	∞	∞	∞	∞	∞	∞	16	∞	∞	∞

The *dual of the Hasse diagram* of the considered partially ordered set (that is, the down-directed graph of the covering relation $\prec$) is from [12] (see Figure 1).

The result of the proposed algorithm for this numerical example is the following linear extension:

$A \prec R \prec G \prec B \prec D \prec H \prec N \prec C \prec K \prec$
$L \prec M \prec E \prec J \prec O \prec I \prec P \prec Q \prec F.$

The total cost of the resulted linear extension is 670.

6.2 Comparing with Simulated Annealing

The Simulated Annealing (SA) heuristic provides an alternative approach for finding the optimal path regarding to the cost matrix [21]. The method uses a temperature value (denoted by T) for limiting the magnitude of the local search. The SA method uses the following basic operations at parameters.

- It sets the initial T value and decrease it in each iteration. It chooses an initial permutation (as the path) denoted by π_0. It contains a permutation generator, which alters the actual permutation to achieve a new state in the search space.
- When the cost of the generated permutation is lower than the previous one ($f(\pi_{i+1}) < f(\pi_i)$) the algorithm updates the minimal cost and the path estimation to $f(\pi_{i+1})$ and π_{i+1}.
- In other case, it decides (based on the T value) to accept a worse cost and path.
- The stop condition of the SA algorithm is a limit for the T value.

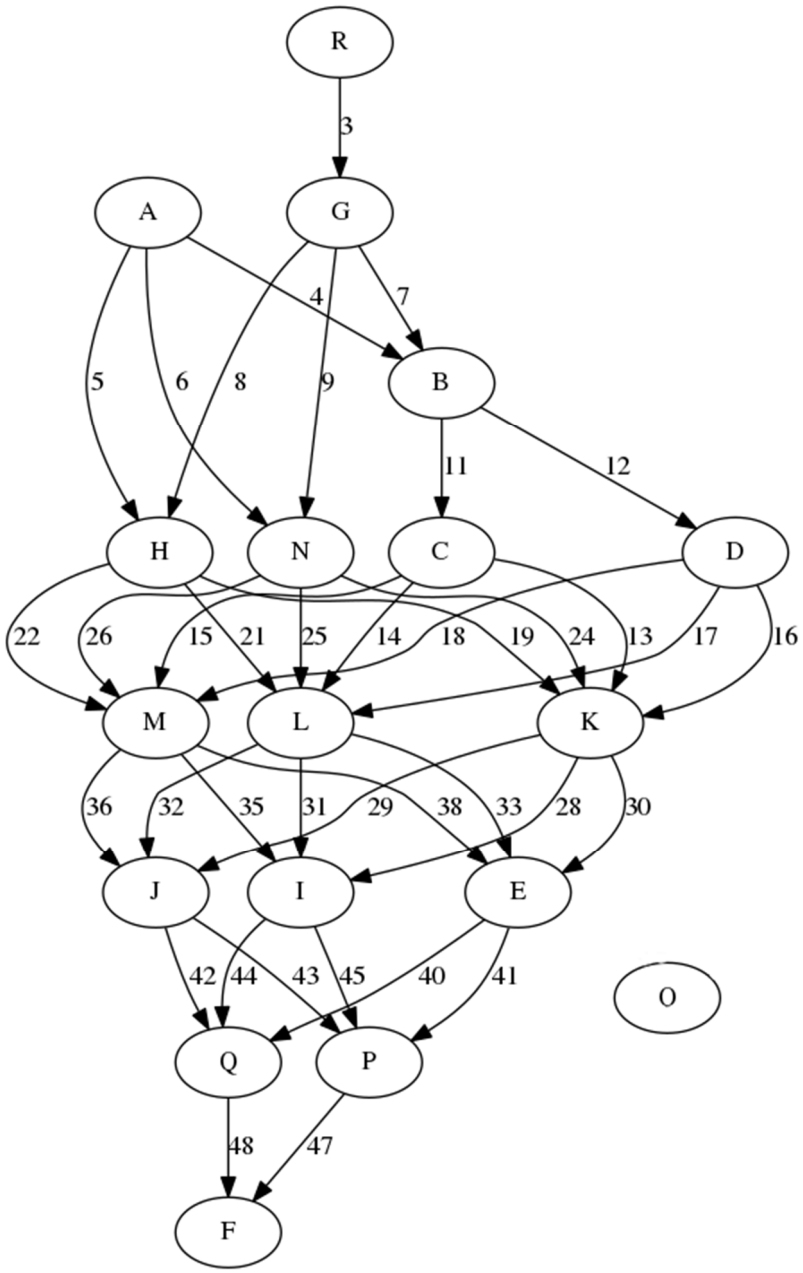

Figure 1
The dual of the Hasse diagram of the considered numerical example.
It also contains the weights on the edges of the graph.

At first as an experimentation, we tried to use a random search and check the provided $f(\pi_i)$ values. Unfortunately, it was unable to provide viable solution in a reasonable time. (The reason is the sparse cost matrix, where the probability of random linear extension is low.)

Next, we have improved the algorithm by providing a sub optimal linear extension. Instead of random permutations, the algorithm chooses two different indices in the path randomly and tried to swap them for a lower cost value. The resulted path is the following: A, R, G, B, D, H, C, N, L, K, J, E, O, M, I, P, Q, F. The cost of this path is 573.

The experimentation with the SA algorithm shows that a relatively simple algorithm with local search can improve the result, but it had difficulties with finding the initial solution.

It is important to check the effectiveness of the algorithm by some statistical experimentations. In this way, we can see the results of the mentioned algorithms for larger set of examples, and the nondeterministic nature of the simulated annealing method also can be managed properly. In the followings, we show measurements which try to demonstrate how the algorithms performs for different sequencing problems.

We have chosen three other examples where the main operations were the following: Various kinds of turning and drilling, coarse grinding, transporting between workplaces and rubbing. Their adjacency matrix had sizes: 10, 12 and 16.

For comparing the efficiency of the Fuzzy Clustering and Simulated Annealing method, we generated 100 random weight matrices for each sample. (The count 100 is an experimental value.)

The generation of the weight matrix uses uniform distributed values from the range [0, 10). The algorithm sets infinity weights for the transitions which are not possible according to the adjacency matrix.

We have estimated the minimal costs both by using the proposed Fuzzy Clustering and the Simulated Annealing methods. We can see the distributions of the minimal costs on Figure 2. (The ranges of estimated minimal costs and the frequencies have been set to the same interval for each histogram, for helping the visual comparison easier.)

This statistical experiment shows that the Fuzzy Clustering method provides better estimation of the minimal costs.

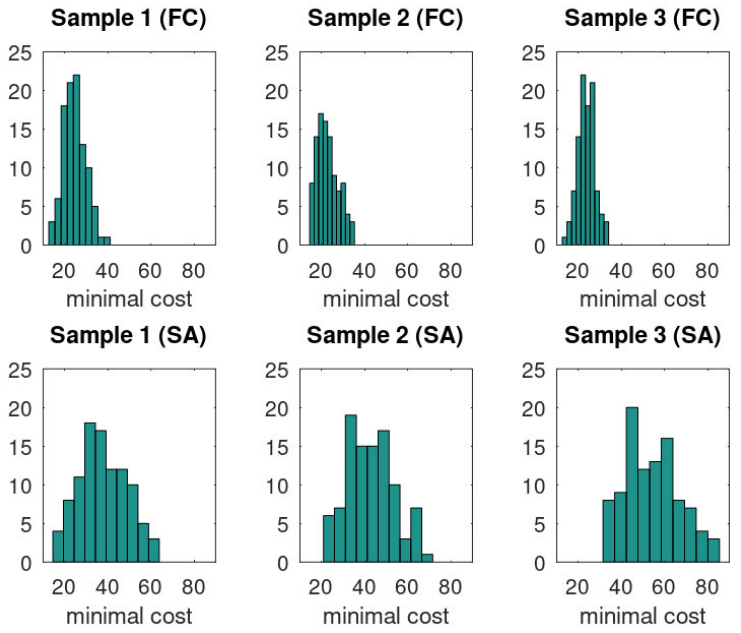

Figure 2

Histograms of the distribution of estimated minimal costs for our method using the Fuzzy Clustering (FC) and for the Simulated Annealing (SA) method

Conclusions

The method discussed in the paper is a heuristic procedure which can solve the job scheduling problem. It consists of a fuzzy clustering algorithm and a procedure for finding the optimal linear extensions of the given partial order inside the clusters. If the size of a cluster K is not too big (for instance, $|K| \leq 6$), then the second phase can be effectuated by a complete enumeration, i.e., in an exact way, as given in Algorithm 1b. Because the procedure is intended for medium-sized manufacturing processes, the size of the formed clusters is small, in general – therefore, our method (which uses Algorithm 1b) is a locally exact heuristic method.

The method has been implemented in GNU/Octave, and has been extended by a JavaScript application, which can help the engineers by guaranteeing that the input matrix is mathematically correct (antisymmetric and acyclic). The Octave source code contains many functions for making possible to reuse or improve some parts of the proposed algorithm.

The results of the proposed algorithm have been illustrated on a small-sized problem; however, this size is typical in real-world applications. Some logistical scheduling problems belong also to the application area of our algorithm.

It is hard to compare the achieved results with state-of-the-art solutions of the manufacturing industry based on some modifications of the travelling salesman method. In fact, the restricted travelling salesman method has several semi-heuristic variants which are applied to large scale scheduling problems, but most of them are proprietary. Their time complexity is generally better than our method, however, for medium-sized job sequences the latter is more flexible. This means that in the case of modification of the technological process the input of our algorithm can be adapted easily. This flexibility is due to the fuzzy clustering procedure, which makes possible to change some threshold values appropriately during the optimization process.

Comparing our method with Eszes procedure, we can state that the results obtained by our method represent better approximations of the optimal solutions because the formed clusters are more realistic. Both proposed fuzzy clustering method and the SA method have many parameters. We must conclude that, our preferred method is more robust in general, and suitable for problems where the manual tuning of the parameters (fuzzy clustering) is beneficial. It does not seem to be relevant to measure the running times of the program, because the Octave software has been chosen for gaining experiments of the algorithm conveniently, and it has terminated in few seconds on modern desktop computers.

Acknowledgment

We dedicate our paper to the memory of Professor Tibor Tóth who passed away in 2020. This research was initiated by him, and each of the authors is grateful to profit from numerous inspiring and friendly discussions with this outstanding researcher. In particular, the fifth author is indebted to Tibor as PhD supervisor.

The authors would like to thank the reviewers for their valuable comments and suggestions which significantly improved the presentation of the paper.

References

[1] Al-wswasi, M., Ivanov, A., and Makatsoris, H.: A survey on smart automated computer-aided process planning (ACAPP) techniques. The International Journal of Advanced Manufacturing Technology, 97(1), (2018) pp. 809-832

[2] Dubois, D., Prade, H.: Fuzzy Sets and Systems, Theory and Applications; Mathematics in Science and Engineering, Vol. 144, Academic Press, New-York, p. 393 (1980) ISBN-13: 978-0122227509

[3] Halevi, G. and Weill, R. D.: Principles of Process Planning, A logical approach, Springer Science+Business Media, Dordrecht (1995) ISBN: 978-0-412-54360-9

[4] Hompel, M., Vogel-Heuser, B. and Bauernhansl, T.: Industry 4.0 handbook Bd.1: Produktion, Springer Reference Technik, Berlin (2017) ISBN 978-3-662-45278-3 [in German]

[5] Hompel, M., Vogel-Heuser, B. and Bauernhansl, T.: Industry 4.0 handbook

Bd.2: Automatisierung, Springer Reference Technik, Berlin (2017) ISBN 978-3-662-53247-8 [in German]

[6] Hompel, M., Vogel-Heuser, B. and Bauernhansl, T.: Industry 4.0 handbook Bd.3: Logistik, Springer Reference Technik, Berlin (2017), ISBN 978-3-662-53250-8 [in German]

[7] Huang, W., Lin, W. and Xu, S.: Application of graph theory and hybrid GA-SA for operation sequencing in a dynamic workshop environment. Computer-Aided Design and Applications, 14(2), (2017) pp. 148-159

[8] Körtesi, P., Radeleczki, S. and Szilágyi, Sz.: Congruences and isotone maps on partially ordered sets, Math. Pannonica, 16/1 (2005), pp. 39-55

[9] Radeleczki, S.: A classification method based on fuzzy contexts, Bul. Stiint. Univ. Baia-Mare, Ser. B, Matematica-Informatica, Vol. XVIII (2002), Nr. 2, pp. 319-324

[10] Radeleczki, S, Tóth, T.: Fuzzy methods in Group Technology.: Research report, OTKA 2348/91, (4.2/6) Univ. of Miskolc, Inst. of Informatics (1992), pp. 1-34 (In Hungarian)

[11] Phung, L. X., Van Tran, D., Hoang, S. V., and Truong, S. H.: Effective method of operation sequence optimization in CAPP based on modified clustering algorithm. Bulletin of the JSME, Journal of Advanced Mechanical Design, Systems, and Manufacturing, 11(1), (2017), JAMDSM0001-JAMDSM0001

[12] Tóth, T.: Production Systems and Processes, Fundamentals of Production Informatics, University of Miskolc Publishing House (Miskolci Egyetemi Kiadó) (2004) p. 464, ISBN: 9636616302 (in Hungarian)

[13] Tóth, T., Detzky, I. and Eszes, L.: Optimization of Discrete Technology Processes Using a Method Traced Back to Constrained Travelling Salesman Problem. Proceedings of the 27th International MATADOR Conference, Manchester, 21. April (1988), editor Davis B. J., edited by University of Manchester in collaboration with MacMillan Education Ltd. pp. 145-153

[14] Trotter, W. T.: Combinatorics and Partially Ordered Sets, Dimension Theory, John Hopkins University Press, Baltimore and London, (1992), ISBN-13: 978-0801869778

[15] Xu, H. and Wang, H. P.: Part family formation for GT applications based on fuzzy mathematics, International Journal of Production Research, Vol. 27, No. 9 (1989), pp. 1637-1651

[16] Xu, X., Wang, L. and Newman, S. T.: Computer-aided process planning - a critical review of recent developments and future trends. International Journal of Computer Integrated Manufacturing, 24(1) (2011) pp. 1-31

[17] Lin C. J, Wang H. P.: Optimal operation planning and sequencing minimization of tool change overs. International Journal of Production

Research, 31 (1993) pp. 311-324

[18] Koulamas C.: Operations sequencing and machining economics. Int J Prod Res 31(4), (1993) pp. 957-975

[19] Bhaskara Reddy SV, Shunmugam MS, Narendran TT.: Operation sequencing in CAPP using genetic algorithms. Int J Prod Res 37(5), (1999), pp. 1063-1074

[20] Li L, Fuh JYH, Zhang YF, Nee AYC.: Application of genetic algorithm to computer-aided process planning in distributed manufacturing environments. Robot Comput-Integr Manuf. 21 (2005) pp. 568-578

[21] Foerster H, Wischer G.: Simulated annealing for order spread minimization in sequencing cutting patterns. Eur J. Oper. Res. 110 (1998) pp. 272-281

[22] Li WD, Ong SK, Nee AYC.: Optimization of process plans using a constraint-based tabu search approach. Int J. Prod Res, 42 (10), (1998), pp. 1955-1985

[23] Krishna AG, Rao KM.: Optimisation of operations sequence in CAPP using an ant colony algorithm. Int J Adv Manuf Technol 29, (2006), pp. 159-164

[24] Guo YW, Mileham AR, Owen GW, Li WD: Operation sequencing optimization using a particle swarm optimization approach. Proc Inst Mech Eng, B, J. Eng Manuf. 220 (B12), (2006), pp. 1945-1958

[25] Mojtaba S, Reza T-M.: Application of genetic algorithm to computer-aided process planning in preliminary and detailed planning. Eng Appl Artif Intell. 2009.04.05, doi:10.1016/j.engappai

[26] Korde UP, Bora BC, Stelson KA, Riley DR (1992) Computer aided process planning for turned parts using fundamental and heuristic principles. J Eng Ind, (1992), pp. 114:31-40

[27] Nallakumarasamy, G., Srinivasan, P. S. S., Raja, K. V., and Malayalamurthi, R.: Optimization of operation sequencing in CAPP using simulated annealing technique (SAT). Int J Adv Manuf Technol, 54(5-8), (2011), pp. 721-728

[28] Ssemakula, M. E., and Rangachar, R. M.: The prospects of process sequence optimization in CAPP systems. Computers and Industrial Engineering, 16(1), (1989) pp. 161-170

[29] Rétfalvi, A., and Stampfer, M.: The key steps toward automation of the fixture planning and design. Acta Polytechnica Hungarica, 10(6), (2013), pp. 77-98

[30] Azemi, F., Šimunović, G., Lujić, R., Tokody, D., and Rajnai, Z.: The use of advanced manufacturing technology to reduce product cost. Acta Polytechnica Hungarica, 16(7), (2019), pp.115-131

[31] Bechler, G.: Demand-oriented product line optimization, eBook, Springer,

(2019), ISBN 978-3-658-25542-8 [in German]

[32] Krapp, S.: The successful design of logistics in cooperative groups: An empirical study, eBook, Springer, (2018), ISBN: 978-3-658-23888-9 [in German]

[33] Scholz, D.: Mixed-integer linear layout planning model, In: Innerbetriebliche Standortplanung, Gabler Verlag, (2010) ISBN 978-3-8349-2277-9 [in German]

[34] Wehking, K.-H., Yousefifar, R., Nortwyck, R.: Computer-aided planning, technical handbook for logistics, Springer Verlag, (2010) ISBN 978-3-662-60868-5 [in German]

[35] Zahorán, L. and Kovács, A.: ProSeqqo: A generic solver for process planning and sequencing in industrial robotics. Robotics and Computer-Integrated Manufacturing, 78, (2022), 102387, (DOI: 10.1016/j.rcim.2022.102387)