International Journal of Production Research

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/tprs20

Turret-index optimisation with mathematical programming and metaheuristic approaches

Adil Baykasoglu, Elif Yoruk & Seyda Topaloglu Yildiz

To cite this article: Adil Baykasoglu, Elif Yoruk & Seyda Topaloglu Yildiz (12 Sep 2024): Turret-index optimisation with mathematical programming and metaheuristic approaches, International Journal of Production Research, DOI: 10.1080/00207543.2024.2399711

To link to this article: https://doi.org/10.1080/00207543.2024.2399711

Published online: 12 Sep 2024.

Submit your article to this journal

Article views: 108

View related articles

View Crossmark data

Full Terms & Conditions of access and use can be found at
https://www.tandfonline.com/action/journalInformation?journalCode=tprs20

INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH
https://doi.org/10.1080/00207543.2024.2399711

Turret-index optimisation with mathematical programming and metaheuristic approaches

Adil Baykasoglu, Elif Yoruk and Seyda Topaloglu Yildiz

Faculty of Engineering, Department of Industrial Engineering, Dokuz Eylül University, İzmir, Turkey

ABSTRACT
This paper addresses the turret-index optimisation problem by proposing two mathematical programming approaches based on quadratic programming (QP) and constraint programming (CP). In addition, a metaheuristic algorithm based on weighted superposition attraction (WSA) is developed due to the combinatorial complexity of the problem. Despite attempting to linearise the QP formulation for problem resolution, both the QP and Linearized QP models prove ineffective for medium and larger-sized instances, providing solutions only for small-sized problems. In the second mathematical programming approach, a novel CP model is introduced in the literature. While CP can rapidly offer optimal solutions for small-sized problems, it only provides satisfactory solutions for medium and larger-sized problems within the predetermined computational time limits and does not guarantee optimal results. On the other hand, through computational analysis and relevant statistical tests, the proposed WSA algorithm provides the best solutions for all test problems within a reasonable computational time.

ARTICLE HISTORY
Received 13 February 2024
Accepted 27 August 2024

KEYWORDS
SDG 9; industry; turret-index optimization; constraint programming; integer programming; weighted superposition attraction algorithm

1. Introduction

The turret-index optimisation problem was initially identified by Dereli et al. (1998) during their investigations on a manufacturing process planning. They aimed to reduce non-machining times further when creating optimised process plans for mechanical components (Dereli and Baykasoglu 2005). Indexing within the automatic tool changer (ATC) of CNC (Computerized Numerical Control) machines involves the automatic positioning and/or changing of tools on the ATC in response to commands within the part programme. This definition may vary slightly depending on the type of ATC apparatus used, such as turret, disk, drum, or chain mechanisms. This study specifically focuses on turret-type ATC magazines. In turret-type ATCs, there is no need to manually load or unload cutting tools after they have been loaded into the turret magazine of a CNC machine (Dereli and Filiz 2000). Turret-indexing refers to the positioning of the turret, and turret-indexing time (from tool-to-tool or face-to-face) can be defined as the duration for a turret magazine to move between two adjacent slots (stations) or tools, as illustrated in Figure 1. The turret-index optimisation problem is different from the minimisation of tool switches problem (MTSP). In MTSP, the capacity of the ATC is limited, which means that the total number of required tools exceeds this capacity. As a result, in order to accommodate the requirements of upcoming parts, some tools may need to be substituted or switched with others, while keeping some tools in their current positions. The MTSP addresses two interconnected problems: sequencing and tooling. The main goal is to determine the optimal sequence of parts in order to minimise the frequency of tool switches. On the other hand, the turret-index optimisation problem (also known as the ATC indexing problem) focuses on finding the best allocation of the required cutting tools in the slots of a turret magazine on a CNC machine (Baykasoglu and Ozsoydan 2017). Therefore, the turret-index optimisation problem specifically deals with deciding the slot indexing of tools, and it differs from the MTSP.

Dereli et al. (1998) mentioned and discussed that the turret-index optimisation problem is combinatorically difficult and resembles the famous travelling salesperson problem. Therefore, they presented a solution approach by making use of the genetic algorithm implementation on an example problem. Subsequently, Dereli and Filiz (2000) further refined their genetic algorithm approach and demonstrated its applicability in integrating the turret-index optimisation problem into the generation of process plans for mechanical components. After

CONTACT Adil Baykasoglu adil.baykasoglu@deu.edu.tr Dokuz Eylül University, Faculty of Engineering, Department of Industrial Engineering, Tinaztepe Campus, Adatepe Mah. Dogus Cad. No: 207-E, 35397 Buca, Izmir, Turkey

© 2024 Informa UK Limited, trading as Taylor & Francis Group

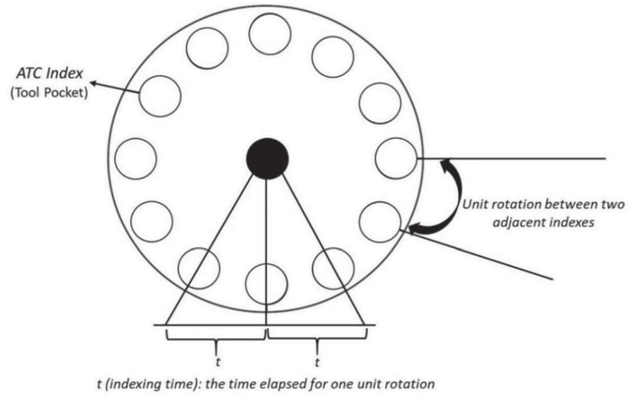

ATC Index
(Tool Pocket)
Unit rotation between two adjacent indexes
t
t
t (indexing time): the time elapsed for one unit rotation

Figure 1. Tool-indexing on a turret (Dereli and Filiz 2000).

these preliminary studies, a few authors also considered the present turret-index optimisation problem. Krishna and Rao (2006) presented an ant colony algorithm for the problem by providing a single example. Ghosh (2016a-c) formulated the turret-index optimisation problem as a single-row facility layout problem and applied several metaheuristic algorithms like tabu search and genetic algorithm for its solutions by making use of several data sets, including those provided by Anjos, Kennings, and Vannelli (2005) and Anjos and Yen (2009). Atta, Mahapatra, and Mukhopadhyay (2019) presented several versions of the harmony search algorithm for solving the present turret-index optimisation problem. They also presented a comparative study with other metaheuristic algorithms that were developed by Ghosh (2016a-c) and reported the best-known solutions so far in the literature.

The present turret-index optimisation problem was also extended by Baykasoglu and Dereli (2004) by considering cutting tool duplications. Later, Baykasoglu and Ozsoydan (2016) improved this study by introducing a new method for calculating the objective function via a shortest-path algorithm. Amouzgar, Nourmohammadi, and Ng (2021) also considered this extended problem and presented a multi-objective mathematical model and a genetic algorithm for its solution. We need to mention here that tool duplications are not considered in this study.

The present study contributes to the literature by presenting a constraint-programming model and a new metaheuristic algorithm based on Weighted Superposition Attraction (WSA), both for the first time in the literature. Detailed computation studies with mathematical programming models and WSA are also performed, and some of the best-known results are improved. Moreover, it is shown that WSA is able to provide the best results for all test problem instances. Results are also verified with statistical tests.

The remainder of this paper is organised as follows: Section 2 presents the problem definition, Section 3 outlines modelling and solution approaches, computational studies are reported in Section 4, and finally paper concludes in Section 5.

2. Problem definition

The turret-index optimisation problem can be described as identifying the optimal positions for cutting tools within the tool magazines of a CNC machine tool, such as an ATC or Turret Magazine (Dereli et al. 1998). Finding the most suitable positions for cutting tools within the turret or magazine of CNC machine tools is an important task for developing optimal process plans in component machining. As an example, consider the component presented in Figure 2, which contains 16 operations (O1-O16). These operations need ten different cutting tools (T1-T10), as also depicted in Figure 2. A suitable process plan for this component is shown in Figure 3, which contains information about operation sequences, operation numbers, and corresponding cutting tools (Dereli 1998). This component is planned to be manufactured on a CNC machine, which has a 10-station turret. Rotation of the turret is necessary as the component has 16 operations, and some consecutive operations need different cutting tools. As can be seen from Figure 3, 11 rotations are required (rotations are indicated by curved arrows in Figure 3). In order to develop optimisation models for the turret-index optimisation problem, frequency and distance matrices need to be generated from the aforementioned process-planning information.

Generating frequency and distance matrices: Frequency matrices are generated from the process plans of components by counting consecutive usage of different cutting tool types (see Figure 3). The frequency matrix of the example component that is shown in Figure 2 with its process plan in Figure 3 is depicted in Table 1.

For example, Operation 2 (O2) uses Tool Type 1 (T1), while O3 uses T5. Therefore, the cutting tool is changed from T1 to T5 after completing O2. Since there are no additional transitions between T1 and T5 in the process plan, the frequency of transitions between these tool types remains at 1. If, in the process plan (Figure 3), the tool type to be used in O12 were T7 instead of T9, there would be a total of two transitions from T9 to T7 (O9-O10 and O11-O12). Consequently, the frequency of transitions between T9 and T7 would be 2, reflecting this value in the frequency matrix shown in Table 1.

To compute the distance (i.e. the minimum number of rotations) between slots in turrets with bi-directional indexing capability, we need to consider the absolute differences in calculating the number of unit rotations from

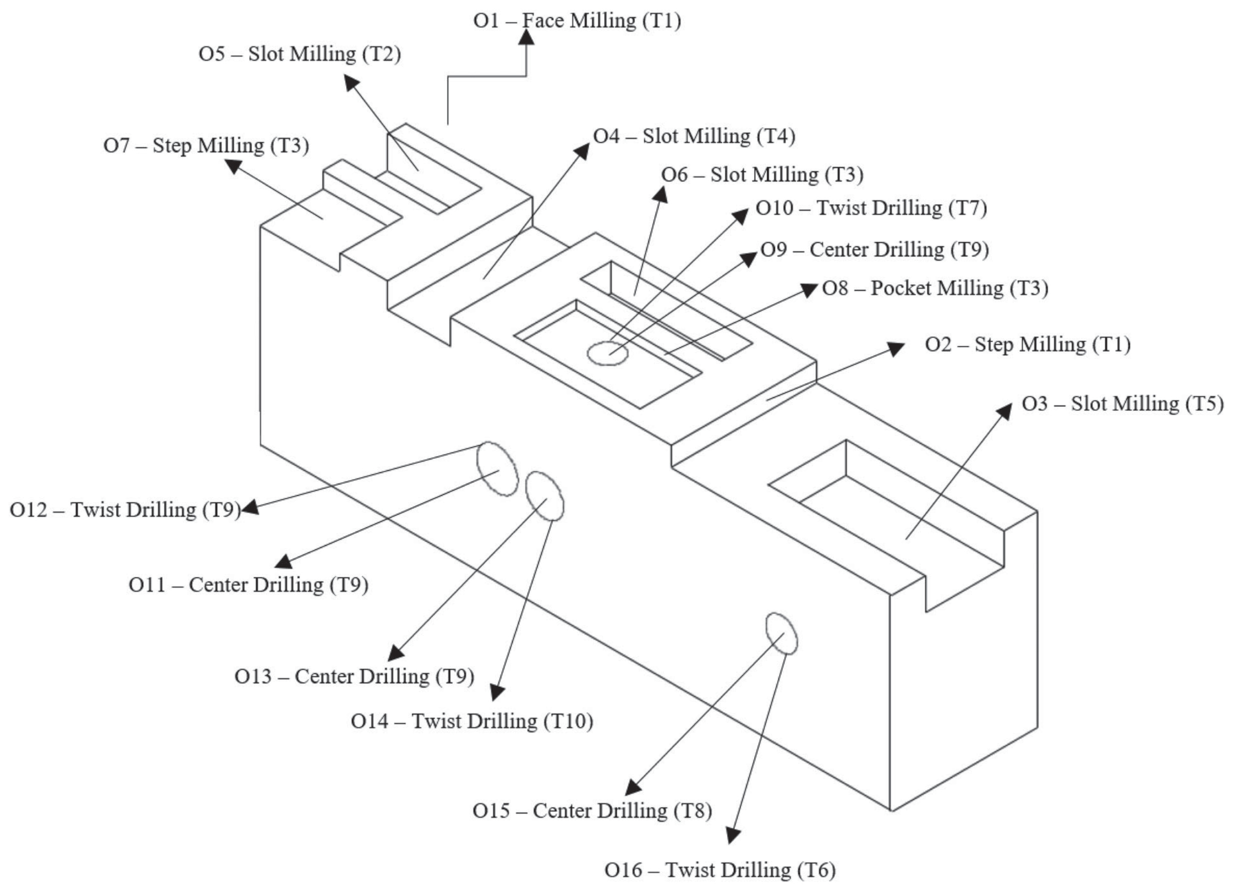
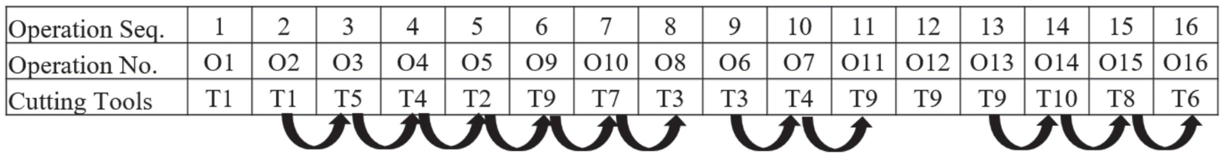

O1 – Face Milling (T1)
O5 – Slot Milling (T2)
O7 – Step Milling (T3)
O4 – Slot Milling (T4)
O6 – Slot Milling (T3)
O10 – Twist Drilling (T7)
O9 – Center Drilling (T9)
O8 – Pocket Milling (T3)
O2 – Step Milling (T1)
O3 – Slot Milling (T5)
O12 – Twist Drilling (T9)
O11 – Center Drilling (T9)
O13 – Center Drilling (T9)
O14 – Twist Drilling (T10)
O15 – Center Drilling (T8)
O16 – Twist Drilling (T6)

Figure 2. Example component with operations and cutting tools (adapted from (Dereli 1998)).



Operation Seq.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Operation No.	O1	O2	O3	O4	O5	O9	O10	O8	O6	O7	O11	O12	O13	O14	O15	O16
Cutting Tools	T1	T1	T5	T4	T2	T9	T7	T3	T3	T4	T9	T9	T9	T10	T8	T6

Figure 3. Process plan for the example component.

the current tool to the target tool. The difference between the two consecutive slot indexes (i and j) must be calculated in such a way that its value should be equal to or smaller than half of the turret capacity (n). For instance, for a turret with 10 slots ($n = 10$), the minimum distance between slot index 1 (S1) and slot index 7 (S7) is 4 and the maximum distance between S1 and S7 is 6 due to its bi-directional indexing capability. Since the half of the turret capacity is 5, the distance between S1 and S7 becomes 4 for a turret with 10 slots. Using these definitions, the distance matrix ($dist_{ij}$) for the example turret (bi-directional indexing capability) with n slots can be determined by

Table 1. Frequency matrix of the example component.

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
T1	0	0	0	0	1	0	0	0	0	0
T2	0	0	0	0	0	0	0	0	**1**	0
T3	0	0	0	**1**	0	0	0	0	0	0
T4	0	**1**	0	0	0	0	0	0	**1**	0
T5	0	0	0	**1**	0	0	0	0	0	0
T6	0	0	0	0	0	0	0	0	0	0
T7	0	0	**1**	0	0	0	0	0	0	0
T8	0	0	0	0	0	**1**	0	0	0	0
T9	0	0	0	0	0	0	**1**	0	0	**1**
T10	0	0	0	0	0	0	0	**1**	0	0

Table 2. Distance matrix between slots (S_i, S_j) for the example bi-directional turret.

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
S1	0	1	2	3	4	5	4	3	2	1
S2	1	0	1	2	3	4	5	4	3	2
S3	2	1	0	1	2	3	4	5	4	3
S4	3	2	1	0	1	2	3	4	5	4
S5	4	3	2	1	0	1	2	3	4	5
S6	5	4	3	2	1	0	1	2	3	4
S7	4	5	4	3	2	1	0	1	2	3
S8	3	4	5	4	3	2	1	0	1	2
S9	2	3	4	5	4	3	2	1	0	1
S10	1	2	3	4	5	4	3	2	1	0

using Equation (1).

$$dist_{ij} = \min\{|j-i|, |i+n-j|\} \qquad (1)$$

The distance matrix for $n = 10$ is presented in Table 2.

3. Modelling and solution approaches

Three solution approaches are proposed to solve the present turret-index optimisation problem. The first approach implements a quadratic assignment formulation that was inspired by the mathematical model of the single-row facility layout problem (Ghosh 2016a, 2016b, 2016c). This model is then linearised and used for the problem solution. The second approach employs a constraint programming formulation for the turret-index optimisation for the first time in the related literature. Due to the combinatorial complexity of the problem (Dereli and Filiz 2000), a new solution approach that is based on a recent metaheuristic algorithm, which is known as WSA, is utilised as the third optimisation strategy. Each employed optimisation approach is explained in the following sub-sections. Table 3 describes the notations used in the solution approaches.

3.1. Quadratic assignment formulation of the turret-index optimisation problem

The present turret-index optimisation problem can be formulated as a quadratic assignment problem similar to the single-row facility layout problem, as also mentioned by Ghosh (2016a). The model is given by Equations (2)-(5):

$$\min \sum_{i\in S}\sum_{\substack{j\in S\\ j\neq i}}\sum_{k\in T}\sum_{\substack{l\in T\\ l\neq k}} t dist_{ij} f_{kl}\, y_{ik}\, y_{jl} \qquad (2)$$

Subject to

$$\sum_{i\in S} y_{ik} = 1\ \forall k\in T \qquad (3)$$

$$\sum_{k\in T} y_{ik} \le 1\ \forall i\in S \qquad (4)$$

Table 3. Nomenclature for indices, parameters and decision variables.

m	Number of tools
n	Number of slots, $m \le n$
T	Set of cutting tools, $T = \{1,\dots,m\}$
S	Set of slots, $S = \{1,\dots,n\}$
i, j	The indices for slots
k, l	The indices for tools
f_{mxm}	Frequency matrix for the tools, denoting the frequencies of all the pairs
$dist_{ij}$	The minimum distance (i.e. the minimum number of rotations) between the slot $i \in S$ and the slot $j \in S$
t	Indexing time between two adjacent slots
y_{ik}	1, if the slot $i \in S$ is occupied by the tool $k \in T$; 0, otherwise
z_{ijkl}	1, if the slot $i \in S$ is occupied by the tool $k \in T$ and slot $j \in S$ is occupied by the tool $l \in T$; 0, otherwise
H_1	Set of the first half of the slots on the turret magazine $H_1 \subset S, H_1 = \{1, 2, \dots, n/2\}$
H_2	Set of the second half of the slots on the turret magazine $H_2 \subset S, H_2 = \{n/2+1, n/2+2, \dots, n\}$
u_k	The slot index where the tool $k \in T$ is occupied
x_i	1, if the slot $i \in S$ is occupied by a cutting tool; 0, otherwise
τ	Intensification and diversification parameter of WSA

$$y_{ik} \in \{0,1\}\ \forall i\in S,\ \forall k\in T \qquad (5)$$

The objective function Equation (2) minimises the total tool-indexing time. Constraint set (3) guarantees that each tool is assigned to one slot, and constraint set (4) ensures that each slot can accommodate at most one tool. Constraint set (5) defines the domain of the decision variables.

The objective function (2) is in the quadratic form; therefore, it is non-linear. However, it can be linearised by using various methods. In this study, we employ the approach of Christofides, Mingozzi, and Toth (1980). The linearised model is represented by Equations (3)-(8).

$$\min \sum_{i\in S}\sum_{\substack{j\in S\\ j\neq i}}\sum_{k\in T}\sum_{\substack{l\in T\\ l\neq k}} t\, dist_{ij} f_{kl}\, z_{ijkl} \qquad (6)$$

Subject to

$$z_{ijkl} \ge y_{ik} + y_{jl} - 1 \quad \forall i\in S,\ \forall j\in S,\ \forall k\in T,\ \forall l\in T \qquad (7)$$

$$z_{ijkl} \in \{0,1\} \quad \forall i\in S,\ \forall j\in S,\ \forall k\in T,\ \forall l\in T \qquad (8)$$

& constraints (3)-(5).

The additional constraint set (7) defines the value of the decision variable z_{ijkl} in relation to the values of y_{ik} and y_{jl}. Constraint set (8) defines the domain of the decision variable z_{ijkl}.

3.2. Constraint programming formulation of the turret-index optimisation problem

Due to the effectiveness of constraint programming methods in addressing challenging combinatorial optimisation problems (Schiex and De Givry 2019), this study proposes a constraint programming formulation for the turret-index optimisation problem for the first time in the literature. The equations for the constraint programming formulation are outlined in Equations (9)-(16).

$$\min \sum_{k\in T}\sum_{\substack{l\in T\\ k\neq l}} t\,dist_{u_k u_l} f_{kl} \tag{9}$$

Subject to

$$all\_different\{u_1, u_2, .., u_m\} \tag{10}$$

$$\text{maximum}\{u_1, u_2, .., u_m\} = m \tag{11}$$

$$\text{minimum}\{u_1, u_2, .., u_m\} = 1 \tag{12}$$

$$x_{u_k} = 1 \quad \forall k \in T \tag{13}$$

$$\sum_{i\in S} x_i = m \tag{14}$$

$$\sum_{i\in H_2} x_i \leq \sum_{j\in H_1} x_j \tag{15}$$

$$x_i \in \{0,1\} \quad \forall i \in S \tag{16}$$

The objective function Equation (9) minimises the total tool-indexing time. Constraint set (10) represents a global constraint that mandates the allocation of each tool to a distinct slot, where u_k is the slot index where the tool $k \in T$ is occupied. This constraint guarantees that each tool is assigned to only one slot, and each slot is occupied by at most one tool. Constraints (11)-(15) have been redundantly added to further narrow the solution space by enhancing constraint propagation. Constraints (11) and (12) gurantee that each tool is consecutively positioned on the turret magazine. Constraints (13) and (14) serve to optimise the propagation of constraints and enhance solution efficiency. Constraint set (13) determines whether a slot is loaded or not, while constraint (14) ensures that the total number of loaded slots equals the number of tools. Constraint (15) functions as a symmetry-breaking constraint, and constraint set (16) defines the domain set of the decision variable x.

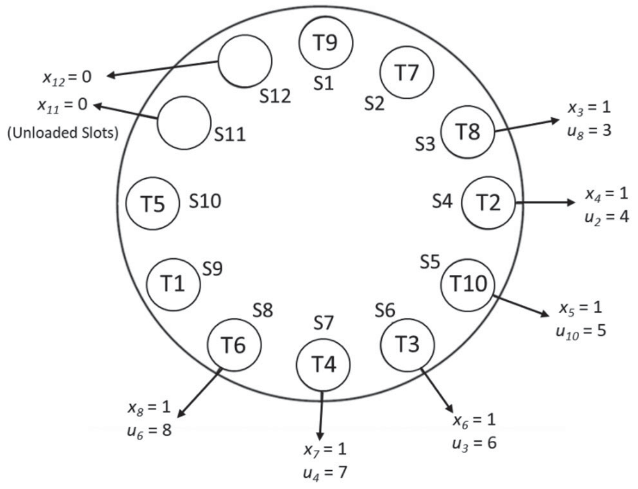


Figure 4. Illustration of the optimal solution of CP for instance O-10_t (10 tools and 12 slots.

3.2.1. An example solution for the CP formulation

The CP formulation is solved for the instance (O-10_t) that contains 10 tools and 12 slots. The values of the decision variables in the optimal solution are given in Table 4 and an illustration of the optimal solution is depicted in Figure 4.

3.3. WSA for the turret-index optimisation problem

WSA is a member of swarm intelligence-based metaheuristic algorithms designed to tackle complex optimisation problems. It has gained prominence since its introduction by Baykasoglu and Akpinar (2015). WSA has been applied to various combinatorial optimisation problems, including resource-constrained project scheduling, permutation flow-shop scheduling, facility layout etc. (Baykasoglu and Senol 2019; Baykasoglu and Subulan 2020). The underlying principles of WSA are rooted in physics-inspired concepts, specifically '*superposition*' and the '*attracted movement of agents*.' Superposition involves a weighted composition of solution vectors, where higher-quality solutions have greater weights, reflecting higher probabilities of contributing to the superposition. After the superposition of solution vectors is determined, solution vectors may be drawn towards superposition based on their attractiveness. In the WSA algorithm, a solution vector undergoes a random move if its fitness surpasses that of the superposition; otherwise, it adjusts its position towards the superposition.

3.3.1. Main steps of the WSA algorithm:

i. *initialization*: During this stage, a user-specified number of solutions are randomly generated, and their corresponding fitness values are calculated.

Table 4. The optimal result of CP for instance O-10_t (10 tools and 12 slots).

Objective Value	2544											
Decision Variable, *x*	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
Optimal Value	1	1	1	1	1	1	1	1	1	1	0	0
Decision Variable, *u*	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}		
Optimal Value	9	4	6	7	10	8	2	3	1	5		

The parameters for the WSA algorithm, including population size, maximum number of iterations, and τ, are also defined at this stage. It is important to mention that the only algorithm-specific parameter for WSA is τ. In the context of the turret-index optimisation problem, a solution vector is represented as a permutation of integers. Refer to Figure 5 for an illustration of a solution vector, where each entry denotes the assignment of a cutting tool to the corresponding turret index (slot).

ii. *Fitness function calculation*: Equation (2) is used to calculate the fitness of solution vectors.

iii. *Determination of superposition:* The procedure for determining the superposition in combinatorial search spaces involves a biased randomised sampling method (Baykasoglu and Senol 2019). The process unfolds as follows:

- Begin with an empty vector, which will later become the superposition vector.
- Generate a random number vector in the range [0, 1], with the length of this vector matching the length of the solution/superposition vector. Each position in the random vector corresponds to a position in the solution/superposition vector.
- Compare the previously calculated rank-based weights of each solution vector with the elements of the random vector. The rank-based weights are computed as $weight(i) = i^{-\tau}$, where i represents the rank of the solution vector (*the solution vector with the best fitness value obtains the first rank in determining the superposition*). τ is a critical control parameter of the WSA algorithm, typically set to 0.8 to strike a good balance between intensification and diversification (Baykasoglu 2024). Higher τ values lead to a more greedy behaviour, favouring high-quality solutions in the superposition, while lower values introduce more randomness.
- If a solution vector's rank-based weight is greater than the corresponding element of the random vector, the related solution vector's element becomes a candidate for the empty position in the superposition vector. This process is repeated for all elements of the solution vector.
- Once the candidate list for the empty position is completed, reevaluate these candidates using a roulette wheel approach to determine the winning candidate element. The selected winner is marked as chosen and blocked in all solution vectors in the population, preventing it from being selected again for the remaining empty positions in the superposition vector.
- Repeat this iterative process until all empty positions in the superposition vector are filled. In some iterations, it may be impossible to find a candidate element for assignment. In such cases, a random element is chosen from the unchecked elements for the corresponding empty position.
- This iterative procedure provides a probabilistic chance for solution vectors to determine the superposition vector based on their fitness. Fitter solution vectors have a higher probability of occupying more positions in the superposition vector.

iv. *Neighborhood generation:* After the composition of the superposition vector, its fitness value is calculated. Essentially, each solution vector decides its next move by evaluating its fitness in relation to the fitness of the superposition vector. Subsequently, each solution vector determines whether to move towards the superposition or not. This decision is based on comparing the fitness value of the superposition vector with that of the respective solution vector. If the superposition vector exhibits a superior fitness compared to the solution vector being considered, the latter will move towards it through the execution of a specially devised move operator known as linear order-crossover (LOX). Conversely, if the superposition vector's fitness is inferior, the solution vector will move randomly using a move operator. In this study, various well-known random move operators, including '*single swap*,' '*two-block swap*,' '*three-block swap*,' '*inversion*,' '*ejection chain*,' '*single insertion*,' and '*block insertion*' are employed (Baykasoglu and Senol 2019). After all solution vectors have been relocated to their new positions through these procedures, a new superposition vector is determined. The iterations of the WSA algorithm then continue in pursuit of finding a better solution.

LOX crossover: The LOX crossover is deliberately chosen as the move mechanism in WSA due to its capability

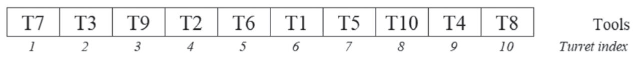


Figure 5. Solution representation for the turret-index optimisation problems.

to copy and transfer segments, enabling solution vectors to converge towards (*become more similar to*) the superposition vector. The LOX crossover method was developed by Falkenauer and Bouffouix (1991). In LOX, a segment chosen from the superposition vector is copied into the offspring (*the new position of the solution vector*). The empty places in the offspring are then filled with the missing elements while preserving the relative order of the solution vector, scanning from left to right. An illustrative example of the LOX crossover move is shown in Figure 6.

In this study, the length of the segment is randomly chosen between 1 and half of the superposition vector's length. The main steps of the WSA algorithm are shown in a flow chart in Figure 7.

An illustrative example for superposition determination: A detailed numerical example related to the superposition determination for the turret-index optimisation problem is presented in Figure 8.

Assume that there are six cutting tools to be assigned to six alternative turret indexes. For example, T3 is assigned to the second index in the first solution vector, which is also the fittest solution vector. In the ranking process, solutions are ordered from 'best to worst.' The parameter τ is set to 0.80, and random number for each dimension is [0.34, 0.52, 0.45, 0.62, 0.54, 0.48] for the empty places of the superposition vector, respectively. Based on these randomly generated threshold values, solution vectors that exceed these thresholds are defined and considered eligible for inclusion in the roulette wheel selection process. For instance, taking the first dimension as an example, the first three solution vectors are included in the roulette wheel selection. Assuming the first solution vector emerges as the winner, the first dimension of the superposition vector becomes T1. To prevent T1 from being selected again (*which would create infeasibility for the problem*), all instances of T1 in all solution vectors are disqualified as candidates (highlighted in grey in Figure 8, step-2). Applying a similar logic, the second dimension of the superposition vector is determined to be T3. This process is iterated for the remaining dimensions to ascertain the complete superposition vector. It is important to note that the first-ranked solution vector always qualifies for roulette wheel selection, as any power of unity equals unity. Therefore, elements from the fittest solution vector always stand a chance of occupying positions in the superposition vector. Probabilities for other candidates are regulated by τ. As mentioned earlier, lower values of τ increase the likelihood of poor-quality solution vectors contributing elements to the superposition vector, resulting in a more randomised search. Conversely, higher values of τ lead to a greedier search characteristic.

4. Computational study

In this section, we apply the quadratic assignment model (QP), the linearised quadratic assignment model (LQP), the newly proposed constraint-programming model (CP), and the proposed WSA metaheuristic algorithm to the present turret-index optimisation problem. Moreover, valid inequalities (11)-(15) from the CP model have been introduced into the LQP model as Constraint (17).

$$\sum_{k\in T} y_{ik}=0 \quad i\in S,\ i>m \tag{17}$$

Adding valid inequalities to the mathematical model can effectively reduce the solution space and enhance computational performance. Constraint (17) ensures that no tool is placed on slot indexes greater than the number of available tools. The linearised model with valid inequalities (LQP_vi) is represented by Equations (3)-(8) and constraint (17).

The QP, LQP, LQP_vi and CP models are implemented using IBM ILOG CPLEX OPL 22.1 and run on a PC with an Intel Core i7 (4.00 GHz) processor, 16 GB RAM, and a Windows 10 operating system. The IBM ILOG CPLEX OPL implementation codes for the QP, LQP- LQP_vi and CP models are provided in Appendix A-C, respectively. The WSA algorithm is coded using the Matlab 2020a software, and optimisation results are obtained using the same computer. We use several test instances available from the following web page (http://kucse.in/tip) to test and compare the performances of the proposed modelling and solution approaches. These test problems were generated by Anjos, Kennings, and Vannelli (2005) and Anjos and Yen (2009) for the single-row facility layout problems and adapted to the present turret-index optimisation problem by Ghosh (2016a, 2016b, 2016c) and Atta, Mahapatra, and Mukhopadhyay (2019). Atta, Mahapatra, and Mukhopadhyay (2019) also developed two efficient harmony search-based metaheuristics for the problem and provided a comparative study with several other metaheuristics (Tabu Search & Tabu Search with Lin Kernighan neighbourhood structures (Ghosh 2016a, 2016b); Genetic Algorithm (Ghosh 2016c)) by reporting the best-known results in the literature. The best results in the literature were generated by Atta, Mahapatra, and Mukhopadhyay (2019) harmony search algorithm with a harmony refinement strategy (HS_RS).

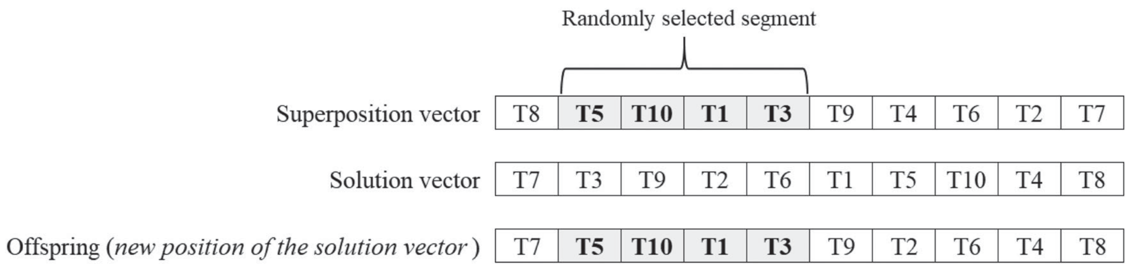

Randomly selected segment
Superposition vector
T8 T5 T10 T1 T3 T9 T4 T6 T2 T7
Solution vector
T7 T3 T9 T2 T6 T1 T5 T10 T4 T8
Offspring (new position of the solution vector)
T7 T5 T10 T1 T3 T9 T2 T6 T4 T8

Figure 6. LOX crossover move.

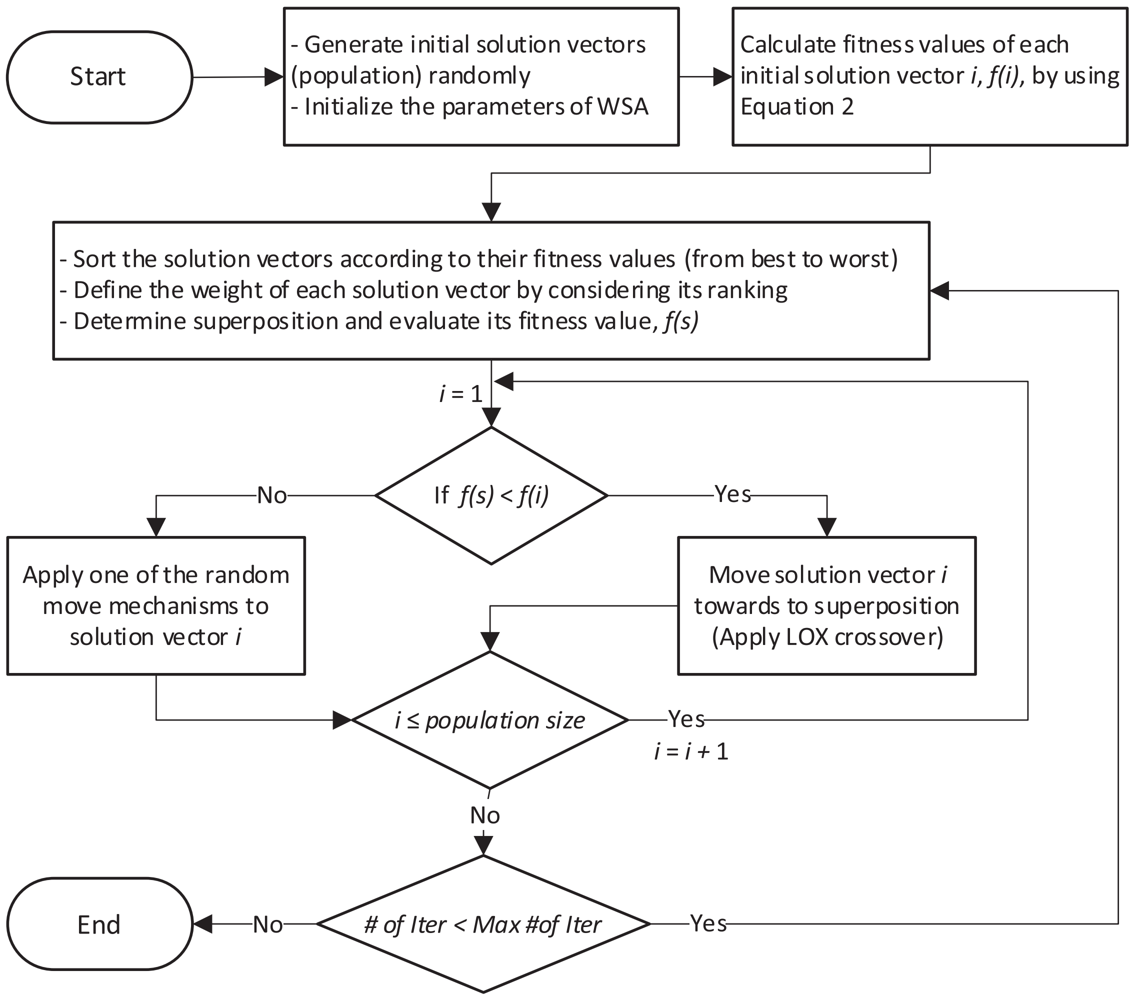
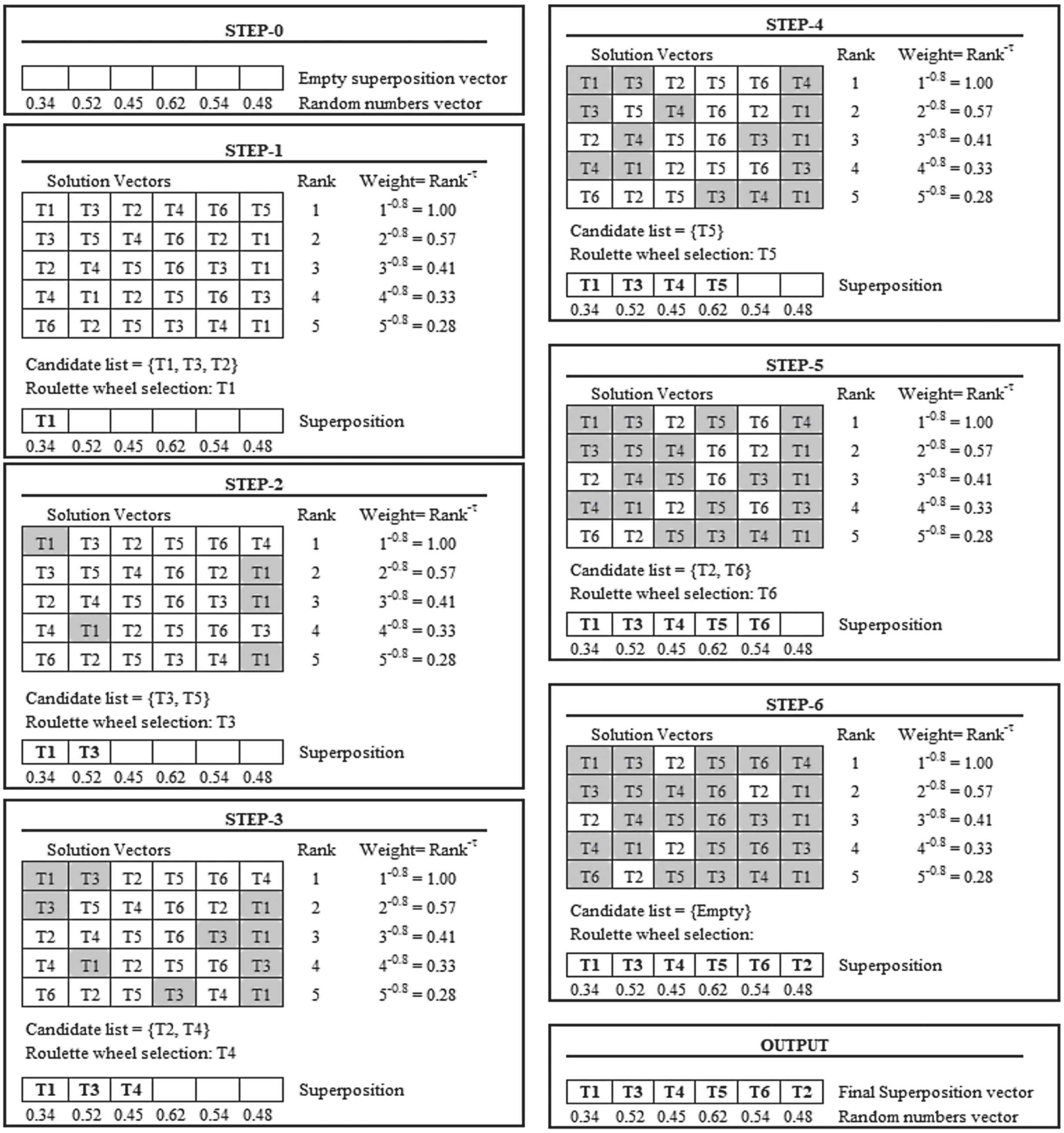

Start
- Generate initial solution vectors (population) randomly
- Initialize the parameters of WSA
Calculate fitness values of each initial solution vector i, f(i), by using Equation 2
- Sort the solution vectors according to their fitness values (from best to worst)
- Define the weight of each solution vector by considering its ranking
- Determine superposition and evaluate its fitness value, f(s)
i = 1
If f(s) < f(i)
No
Yes
Apply one of the random move mechanisms to solution vector i
Move solution vector i towards to superposition (Apply LOX crossover)
i ≤ population size
Yes
i = i + 1
No
# of Iter < Max #of Iter
No
Yes
End

Figure 7. The flowchart of the WSA.

The proposed WSA algorithm is run with the following parameters for all test problems: $\tau = 0.8$, Population size $= 15$, maximum number of iterations $= 10000$. Several other parameter settings are also used; however, it was observed that WSA provides the same results if τ is set between 0.8-1.5 and population size between 10-25. We did not stop the WSA algorithm after convergence due to considerable differences in test problem sizes. We wait until the maximum number of iterations is met; nevertheless, WSA quickly converges especially for small and medium-sized test problems. We also need to mention here that we took 30 runs for each problem instance. The WSA exhibits a remarkably high converging capability owing to its ability to attract solutions towards superposition (Baykasoglu and Baykasoglu 2020, 2021). Therefore, as noted by Baykasoglu and Baykasoglu (2020, 2021) the standard deviation is usually small as also observed in this study. We report the best results, along with the standard deviations and computational times, after 10,000 iterations. Moreover, we set a 120 minutes time limit for



STEP-0

						Empty superposition vector
0.34	0.52	0.45	0.62	0.54	0.48	Random numbers vector

STEP-1

Solution Vectors						Rank	Weight= Rank$^{-\tau}$
T1	T3	T2	T4	T6	T5	1	$1^{-0.8} = 1.00$
T3	T5	T4	T6	T2	T1	2	$2^{-0.8} = 0.57$
T2	T4	T5	T6	T3	T1	3	$3^{-0.8} = 0.41$
T4	T1	T2	T5	T6	T3	4	$4^{-0.8} = 0.33$
T6	T2	T5	T3	T4	T1	5	$5^{-0.8} = 0.28$

Candidate list = {T1, T3, T2}
Roulette wheel selection: T1

T1						Superposition
0.34	0.52	0.45	0.62	0.54	0.48	

STEP-2

Solution Vectors						Rank	Weight= Rank$^{-\tau}$
T1	T3	T2	T5	T6	T4	1	$1^{-0.8} = 1.00$
T3	T5	T4	T6	T2	T1	2	$2^{-0.8} = 0.57$
T2	T4	T5	T6	T3	T1	3	$3^{-0.8} = 0.41$
T4	T1	T2	T5	T6	T3	4	$4^{-0.8} = 0.33$
T6	T2	T5	T3	T4	T1	5	$5^{-0.8} = 0.28$

Candidate list = {T3, T5}
Roulette wheel selection: T3

T1	T3					Superposition
0.34	0.52	0.45	0.62	0.54	0.48	

STEP-3

Solution Vectors						Rank	Weight= Rank$^{-\tau}$
T1	T3	T2	T5	T6	T4	1	$1^{-0.8} = 1.00$
T3	T5	T4	T6	T2	T1	2	$2^{-0.8} = 0.57$
T2	T4	T5	T6	T3	T1	3	$3^{-0.8} = 0.41$
T4	T1	T2	T5	T6	T3	4	$4^{-0.8} = 0.33$
T6	T2	T5	T3	T4	T1	5	$5^{-0.8} = 0.28$

Candidate list = {T2, T4}
Roulette wheel selection: T4

T1	T3	T4				Superposition
0.34	0.52	0.45	0.62	0.54	0.48	

STEP-4

Solution Vectors						Rank	Weight= Rank$^{-\tau}$
T1	T3	T2	T5	T6	T4	1	$1^{-0.8} = 1.00$
T3	T5	T4	T6	T2	T1	2	$2^{-0.8} = 0.57$
T2	T4	T5	T6	T3	T1	3	$3^{-0.8} = 0.41$
T4	T1	T2	T5	T6	T3	4	$4^{-0.8} = 0.33$
T6	T2	T5	T3	T4	T1	5	$5^{-0.8} = 0.28$

Candidate list = {T5}
Roulette wheel selection: T5

T1	T3	T4	T5			Superposition
0.34	0.52	0.45	0.62	0.54	0.48	

STEP-5

Solution Vectors						Rank	Weight= Rank$^{-\tau}$
T1	T3	T2	T5	T6	T4	1	$1^{-0.8} = 1.00$
T3	T5	T4	T6	T2	T1	2	$2^{-0.8} = 0.57$
T2	T4	T5	T6	T3	T1	3	$3^{-0.8} = 0.41$
T4	T1	T2	T5	T6	T3	4	$4^{-0.8} = 0.33$
T6	T2	T5	T3	T4	T1	5	$5^{-0.8} = 0.28$

Candidate list = {T2, T6}
Roulette wheel selection: T6

T1	T3	T4	T5	T6		Superposition
0.34	0.52	0.45	0.62	0.54	0.48	

STEP-6

Solution Vectors						Rank	Weight= Rank$^{-\tau}$
T1	T3	T2	T5	T6	T4	1	$1^{-0.8} = 1.00$
T3	T5	T4	T6	T2	T1	2	$2^{-0.8} = 0.57$
T2	T4	T5	T6	T3	T1	3	$3^{-0.8} = 0.41$
T4	T1	T2	T5	T6	T3	4	$4^{-0.8} = 0.33$
T6	T2	T5	T3	T4	T1	5	$5^{-0.8} = 0.28$

Candidate list = {Empty}
Roulette wheel selection:

T1	T3	T4	T5	T6	T2	Superposition
0.34	0.52	0.45	0.62	0.54	0.48	

OUTPUT

T1	T3	T4	T5	T6	T2	Final Superposition vector
0.34	0.52	0.45	0.62	0.54	0.48	Random numbers vector

Figure 8. Superposition determination for turret-index optimisation problem.

QP, LQP, and CP models. In order to make direct comparison with the results reported in the literature, the indexing time between two adjacent slots, *t* is set to 1.

4.1. Testing and comparing solution approaches for small and medium sized problem instances

Selected instances from the benchmark data sets (O, S, Y) of Atta, Mahapatra, and Mukhopadhyay (2019) are used to analyse and compare the performances of the QP, LQP, LQP_vi, the proposed CP model, the proposed WSA algorithm and the best results reported in the literature. Relatively smaller test instances are selected from benchmark data sets (S and Y) and all (O) instances, because, the CPLEX OPL solver is unable to provide results for larger-sized problem instances in a reasonable time for QP, LQP, and LQP_vi models Results of the computational tests (under a time limit of 120 minutes) for exact solution methods, along with the best results reported in the literature, are presented in Table 5. The exact solution methods are also evaluated under a shorter time limit to assess their performance. The QP, LQP, and LQP_vi models are executed within the time when the CP model first detects the best solution (in seconds) and within 5 minutes (300 s). The size of the models and computational test results under these time limits are shown in Table 6 for QP, LQP, LQP_vi, and CP models. Results for the metaheuristic approaches, along with the best results reported in the literature, are presented in Table 7. (The best results are highlighted in bold in Tables 5–7).'

As can be seen from Table 5, the proposed CP model generated the best results among exact solution approaches for 31 test problems. The CP model reported optimal results for 20 test problems out of 31 test problem instances, whereas QP reported 18 optimal results, LQP and LQP_vi reported 15 and 18 optimal results, respectively. Valid inequalities enable LQP_vi to achieve superior results in less computational time compared to LQP. CP is able to provide optimal solutions in considerably shorter computational times compared to QP, LQP, and LQP_vi.

The gap metric calculates the relative percentage deviation between the objective values obtained from the CP, QP, LQP, and LQP_vi methods, and the best values reported in the literature. A larger positive value for this gap signifies a poorer performance of the respective solution method. The gap can be calculated by using Equation (18).

$$GAP = \frac{Method_{obj} - BestObj}{BestObj} \times 100 \qquad (18)$$

As shown in Table 6, the CP model, QP, LQP, and LQP_vi methods all reported one optimal result for the time when the best solution was first detected (in seconds). The QP model reported 10 best solutions for that time limit out of 31 instances, while LQP and LQP_vi reported 2 best solutions. Under a 5-minute time limit, QP reported 14 optimal and 23 best results, LQP reported 14 best (optimal) results, and LQP_vi reported 15 best (optimal) results. The CP model reported optimal results for 20 test problems and best results for all 31 instances within the defined times. Since the CP model provides the same results within the first solution time and within 5 minutes, the objective function values are depicted in a single column. The corresponding first solution times are also given in the last column of Table 6. It can be seen that the CP model demonstrates the highest performance among the exact solution methods

As can be seen from Table 7, since the objective values of the HS_RS and WSA algorithms are the same (with *gap* values equalling 0), these gaps are not displayed in Table 7. Comparing Tables 5–7, it is evident that WSA provides solutions in the shortest computational times when compared to exact solution methods (CP, QP, LQP, and LQP_vi).

On the other hand, CP, QP, LQP and LQP_vi were not able to provide optimal results for 11, 12, 16 and 13 instances, respectively. We can conclude that the proposed CP model and the proposed WSA algorithm are eligible to solve small/medium-sized turret-index optimisation problems as they can produce the best-known results in reasonable computational times reported in the literature.

4.2. Testing and comparing CP and WSA for larger problem instances

Selected larger instances from the benchmark data sets (Y) of Atta, Mahapatra, and Mukhopadhyay (2019) and data sets (Anjos and sko) of Anjos, Kennings, and Vannelli (2005) and Anjos and Yen (2009) are used to analyse and compare the performances of the proposed CP model, the proposed WSA algorithm, and the best results reported in the literature (HS_RS). The OPL CPLEX solver is unable to provide results for these problem instances in a reasonable time for QP, LQP and LQP_vi models. The computational results for these test problem instances are reported in Table 8, with the best results are bolded.

As shown in Table 8, the proposed WSA algorithm generated the best results for all 34 test problem instances. Additionally, the WSA algorithm improves results for 15 problem instances. On the other hand, the CP model only provided a better result than HS_RS for only one problem instance (sko-36). For three problem instances (Y-30_t, Y-35_t, Anjos-60-01) CP and HS_RS provided the same results. For the remaining problem instances HS_RS provided better results than CP. Moreover, WSA demonstrated much better computational times compared to CP. The *gap* values of the HS_RS, CP and WSA are calculated using Equation (17) and the objective value of WSA is used as the best objective value in the equation. Consequently, the *gap* value of the WSA is 0, while the average gaps of the HS_RS and CP are 0.01 and 0.26, respectively.

In Appendix D, we present the performance comparisons of the proposed CP model and WSA algorithm with other metaheuristics in the literature for Anjos and sko instances. Specifically, we have included the results obtained by Ghosh (2016a, 2016b, 2016c) using various metaheuristic techniques in the columns TS, LS-LK, LS-LK1, TS-LK1, and DFGA. The TS column shows the outcomes obtained through the Tabu Search algorithm, while the LS-LK and LS-LK1 columns represent the results achieved by two heuristics based on Lin-Kernighan neighbourhood structures. In addition, the TS-LK1 column displays the results obtained by utilising a Tabu Search algorithm with the Lin-Kernighan neighbourhood structure, and the DFGA column showcases the results obtained using a customised Genetic

Table 5. Computational results of exact solution methods for small and medium sized test problem instances under a time limit of 120 min.

			The best reported solution via metaheuristics (HS_RS)	Quadratic Programming (QP) Model		Linearized (LQP) Model		LQP with valid inequalities (LQP_vi) Model		Constraint Programming (CP) Model			GAP Values			
Instances	Number of tools (*m*)	Number of slots (*n*)	Obj. Val.	Obj.Val.	Sol.Time	Obj.Val.	Sol. Time	Obj.Val.	Sol. Time	Obj.Val.	Sol.Time	First Sol. Time	QP	LQP	LQP_vi	CP
0-5_t	5	5	**248**	**248**∗	0.03	**248**∗	0.05	**248**∗	0.05	**248**∗	0.04	–	0	0	0	0
0-5_t	5	10	**300**	**300**∗	0.55	**300**∗	0.17	**300**∗	0.06	**300**∗	0.05	-	0	0	0	0
0-6_t	6	10	**584**	**584**∗	3.47	**584**∗	0.25	**584**∗	0.16	**584**∗	0.09	-	0	0	0	0
0-7_t	7	10	**936**	**936**∗	15.7	**936**∗	0.69	**936**∗	0,42	**936**∗	0.5	-	0	0	0	0
0-8_t	8	10	**1448**	**1448**∗	68.61	**1448**∗	2.08	**1448**∗	1.34	**1448**∗	3.89	1.85	0	0	0	0
0-9_t	9	10	**1732**	**1732**∗	189.38	**1732**∗	5.56	**1732**∗	35.63	**1732**∗	39.24	1.79	0	0	0	0
0-10_t	10	10	**2264**	**2264**∗	12.75	**2264**∗	21.34	**2264**∗	16.88	**2264**∗	339.45	1.95	0	0	0	0
0-10_t	10	12	**2544**	**2544**∗	76.13	**2544**∗	87.41	**2544**∗	227	**2544**∗	586.84	2.38	0	0	0	0
0-10_t	10	15	**2756**	**2756**∗	515.48	**2756**∗	5965.42	**2756**∗	244.7	**2756**∗	716.54	2.25	0	0	0	0
0-10_t	10	20	**2804**	**2804**∗	931.94	**2804**	TLim	**2804**∗	179.34	**2804**∗	851.18	2.22	0	0	0	0
0-10_t	10	25	**2804**	**2804**∗	3740.47	**2804**	TLim	**2804**∗	186.02	**2804**∗	995.66	2.65	0	0	0	0
0-10_t	10	30	**2804**	**2804**∗	4121.67	**2804**	TLim	**2804**∗	221.23	**2804**∗	1411.77	5.5	0	0	0	0
0-15_t	15	20	**9636**	**9636**	TLim	9960	TLim	9816	TLim	**9636**	TLim	7.7	0	3.36	1.87	0
0-15_t	15	30	**10268**	**10268**	TLim	10436	TLim	10428	TLim	**10268**	TLim	24.5	0	1.64	1.56	0
O-20_t	20	30	**25200**	26056	TLim	26320	TLim	26204	TLim	**25200**	TLim	545	3.4	4.44	3.98	0
S-12_t	12	20	**8818**	**8818**	TLim	8858	TLim	8832	TLim	**8818**	TLim	2.65	0	0.45	0.16	0
S-13_t	13	20	**11562**	**11562**	TLim	11730	TLim	11654	TLim	**11562**	TLim	4.1	0	1.45	0.80	0
S-13_t	13	30	**11794**	**11794**	TLim	11954	TLim	11840	TLim	**11794**	TLim	3.5	0	1.36	0.39	0
Y-6_t	6	6	**2126**	**2126**∗	0.22	**2126**∗	0.14	**2126**∗	0.16	**2126**∗	0.05	-	0	0	0	0
Y-6_t	6	10	**2744**	**2744**∗	5.36	**2744**∗	0.41	**2744**∗	0.26	**2744**∗	0.09	-	0	0	0	0
Y-7_t	7	10	**3466**	**3466**∗	23.59	**3466**∗	1.22	**3466**∗	0.48	**3466**∗	0.59	-	0	0	0	0
Y-8_t	8	10	**4176**	**4176**∗	92.84	**4176**∗	86.91	**4176**∗	5.25	**4176**∗	5.56	1.78	0	0	0	0
Y-9_t	9	10	**4864**	**4864**∗	258.08	**4864**∗	28.11	**4864**∗	72.88	**4864**∗	37.64	1.77	0	0	0	0
Y-10_t	10	10	**5422**	**5422**∗	296.33	**5422**∗	48.94	**5422**∗	48.81	**5422**∗	385.98	1.86	0	0	0	0
Y-11_t	11	11	**6166**	**6166**	TLim	**6166**	TLim	**6166**	TLim	**6166**∗	602.58	2.16	0	0	0	0
Y-11_t	11	12	**6700**	**6700**	TLim	6706	TLim	**6700**	TLim	**6700**∗	620.58	3.7	0	0.09	0	0
Y-12_t	12	12	**7354**	**7354**	TLim	7372	TLim	7372	TLim	**7354**	TLim	3.66	0	0.25	0.25	0
Y-13_t	13	13	**8392**	**8392**	TLim	8408	TLim	8408	TLim	**8392**	TLim	2.55	0	0.19	0.19	0
Y-14_t	14	14	**9852**	**9852**	TLim	9936	TLim	9902	TLim	**9852**	TLim	4.7	0	0.85	0.51	0
Y-15_t	15	30	**14718**	**14718**	TLim	14984	TLim	14830	TLim	**14718**	TLim	3.45	0	1.81	0.76	0
Y-20_t	20	30	**23770**	24686	TLim	24790	TLim	24556	TLim	**23770**	TLim	55	3.85	4.29	3.31	0

Obj. Val.: objective function value; Sol. Time: solution time (second); First Sol. Time: the time when the best solution first detected (second); TLim: Time limit (120 min); ∗: the optimal result

Table 6. Model sizes and computational results of exact solution methods for small and medium sized test problem instances under short time limits.

			Quadratic Programming (QP) Model				Linearized (LQP) Model				LQP with valid inequalities (LQP_vi) Model				Constraint Programming (CP) Model			
Instances	# of tools (*m*)	# of slots (*n*)	#NV	#NC	Obj.Val. (Time of CP)	Obj.Val. (5 min)	#NV	#NC	Obj.Val. (Time of CP)	Obj.Val. (5 min)	#NV	#NC	Obj.Val. (Time of CP)	Obj.Val. (5 min)	#NV	#NC	Obj.Val.	First Sol. Time
0-5_t	5	5	25	10	**248∗**	**248∗**	650	635	272	**248∗**	650	635	272	**248∗**	10	221	**248∗**	–
0-5_t	5	10	50	15	**300**	**300∗**	2550	2515	392	**300∗**	2550	2520	348	**300∗**	15	221	**300∗**	–
0-6_t	6	10	60	16	**584**	**584∗**	3660	3616	712	**584∗**	3660	3620	712	**584∗**	16	324	**584∗**	–
0-7_t	7	10	70	17	948	**936∗**	4970	4917	*NoS*	**936∗**	4970	4920	*NoS*	**936∗**	17	447	**936∗**	–
0-8_t	8	10	80	18	**1448**	**1448∗**	6480	6418	**1448∗**	**1448∗**	6480	6420	**1448∗**	**1448∗**	18	590	**1448∗**	1.85
0-9_t	9	10	90	19	**1732**	**1732∗**	8190	8119	**1732**	**1732∗**	8190	8120	**1732**	**1732∗**	19	753	**1732∗**	1.79
0-10_t	10	10	100	20	**2264**	**2264∗**	10100	10020	2304	**2264∗**	10100	10020	2304	**2264∗**	20	936	**2264∗**	1.95
0-10_t	10	12	120	22	2596	**2544∗**	14520	14422	2600	**2544∗**	14520	14424	2604	**2544∗**	22	936	**2544∗**	2.38
0-10_t	10	15	150	25	2784	**2756**	22650	22525	2880	2780	22650	22530	2816	2776	25	936	**2756∗**	2.25
0-10_t	10	20	200	30	2860	**2804**	40200	40030	3012	2856	40200	40040	2904	**2804∗**	30	936	**2804∗**	2.22
0-10_t	10	25	250	35	2828	**2804**	62750	62535	2972	2836	62750	62550	2904	2828	35	936	**2804∗**	2.65
0-10_t	10	30	300	40	2892	**2804**	90030	90040	3036	2900	90300	90060	2900	2844	40	936	**2804∗**	5.5
0-15_t	15	20	300	35	9884	**9636**	90300	90035	10368	10016	90300	90040	10340	10216	35	2151	**9636**	7.7
0-15_t	15	30	450	45	10524	10496	202950	202545	11032	10580	202950	202560	10736	10736	45	2151	**10268**	24.5
O-20_t	20	30	600	50	26708	27560	360600	360050	26844	27004	360600	360060	26480	26480	50	3866	**25200**	545
S-12_t	12	20	240	32	8950	8818	57840	57632	9690	8886	57840	57640	8950	8890	32	1362	**8818**	2.65
S-13_t	13	20	260	33	11666	11562	67860	67633	12038	11796	67860	67640	11900	11772	33	1605	**11562**	4.1
S-13_t	13	30	390	43	11992	11820	152490	152143	13194	12106	152490	152160	12020	12020	43	1605	**11794**	3.5
Y-6_t	6	6	36	12	**2126**	**2126∗**	1332	1308	2188	**2126∗**	1332	1308	2188	**2126∗**	12	324	**2126∗**	-
Y-6_t	6	10	60	16	**2744**	**2744∗**	3660	3616	*NoS*	**2744∗**	3660	3620	2832	**2744∗**	16	324	**2744∗**	-
Y-7_t	7	10	70	17	3488	**3466∗**	4970	4917	*NoS*	**3466∗**	4970	4920	3728	**3466∗**	17	447	**3466∗**	-
Y-8_t	8	10	80	18	4186	**4176∗**	6480	6418	4218	**4176∗**	6480	6420	4186	**4176∗**	18	590	**4176∗**	1.78
Y-9_t	9	10	90	19	4874	**4864∗**	8190	8119	4884	**4864∗**	8190	8120	4870	**4864∗**	19	753	**4864∗**	1.77
Y-10_t	10	10	100	20	5428	**5422∗**	10100	10020	5464	**5422∗**	10100	10020	5464	**5422∗**	20	936	**5422∗**	1.86
Y-11_t	11	11	121	22	**6166**	**6166**	14762	14663	6216	6172	14762	14663	6216	6172	22	1139	**6166∗**	2.16
Y-11_t	11	12	132	23	**6700**	**6700**	17556	17447	6768	6710	17556	17448	6732	6710	23	1139	**6700∗**	3.7
Y-12_t	12	12	144	24	7354	**7354**	20880	20760	7438	7394	20880	20760	7438	7394	24	1362	**7354**	3.66
Y-13_t	13	13	169	26	8392	**8392**	28730	28587	8494	8438	28730	28587	8494	8438	26	1605	**8392**	2.55
Y-14_t	14	14	196	28	9904	9884	38612	38444	10056	9994	38612	38444	10056	9994	28	1868	**9852**	4.7
Y-15_t	15	30	450	45	14930	14810	202545	202950	15938	15046	202950	202560	15102	15058	45	2151	**14718**	3.45
Y-20_t	20	30	600	50	26050	25170	360050	360600	26416	26142	360600	360060	24990	24990	50	3866	**23770**	55

#NV: number of decision variables; #NC: number of constraints; Obj. Val. (Time of CP): objective function value with a time limit of the first solution time of CP; Obj. Val. (5 min): objective function value with a time limit of 5 minutes (300 s); NoS: no solution is obtained**;** Obj. Val.: objective function value; First Sol. Time: the time when the best solution first detected (second); TLim: Time limit (120 min); ∗: the optimal result.

Table 7. Computational results of metaheuristic approaches for small and medium sized test problem instances.

			The best reported solution via metaheuristics (HS_RS)			WSA Algorithm		
Instances	Number of tools (*m*)	Number of slots (*n*)	Obj. Val.	Standard Dev.	Sol. Time	Obj.Val	Standard Dev.	Sol. Time
0-5_t	5	5	**248**	0.00	0.72	**248**	0.00	7.71
0-5_t	5	10	**300**	0.00	1.55	**300**	0.00	7.63
0-6_t	6	10	**584**	0.00	1.40	**584**	0.00	8.07
0-7_t	7	10	**936**	1.40	1.57	**936**	0.03	8.6
0-8_t	8	10	**1448**	1.40	1.84	**1448**	0.03	9.56
0-9_t	9	10	**1732**	2.50	1.90	**1732**	0.03	10
0-10_t	10	10	**2264**	N/A	1.22	**2264**	0.05	10.43
0-10_t	10	12	**2544**	N/A	1.50	**2544**	0.06	10.44
0-10_t	10	15	**2756**	N/A	1.69	**2756**	0.69	10.42
0-10_t	10	20	**2804**	16.10	3.00	**2804**	0.83	10.37
0-10_t	10	25	**2804**	N/A	2.01	**2804**	0.77	10.34
0-10_t	10	30	**2804**	N/A	2.33	**2804**	0.87	10.32
0-15_t	15	20	**9636**	38.50	3.32	**9636**	1.13	12.61
0-15_t	15	30	**10268**	N/A	2.59	**10268**	1.45	12.57
O-20_t	20	30	**25200**	72.40	4.31	**25200**	2.39	15.24
S-12_t	12	20	**8818**	13.10	3.18	**8818**	0.34	12.15
S-13_t	13	20	**11562**	22.30	3.12	**11562**	0.82	13.38
S-13_t	13	30	**11794**	N/A	2.47	**11794**	1.08	13.34
Y-6_t	6	6	**2126**	0.00	0.70	**2126**	0.00	8.26
Y-6_t	6	10	**2744**	0.00	1.51	**2744**	0.00	8.14
Y-7_t	7	10	**3466**	3.70	1.91	**3466**	0.04	8.5
Y-8_t	8	10	**4176**	2.00	0.46	**4176**	0.03	9.54
Y-9_t	9	10	**4864**	0.50	0.46	**4864**	0.04	9.96
Y-10_t	10	10	**5422**	N/A	1.23	**5422**	0.02	10.36
Y-11_t	11	11	**6166**	N/A	1.47	**6166**	0.09	10.8
Y-11_t	11	12	**6700**	N/A	1.58	**6700**	0.14	10.77
Y-12_t	12	12	**7354**	N/A	1.63	**7354**	0.20	11.12
Y-13_t	13	13	**8392**	N/A	1.85	**8392**	0.24	11.66
Y-14_t	14	14	**9852**	N/A	1.84	**9852**	0.33	12.12
Y-15_t	15	30	**14718**	N/A	2.62	**14718**	0.77	12.67
Y-20_t	20	30	**23770**	9.90	4.26	**23770**	1.13	15.13

Obj. Val.: objective function value; Standard Dev.: standard deviation; N/A: not available; Sol. Time: solution time (second)

Algorithm (GA) proposed by Dereli et al. (1998) and Dereli and Filiz (2000). The GA column presents the results derived from a GA employing different crossover and mutation operators. Finally, the columns labelled HS_woRS and HS_RS denote the Harmony Search without Harmony Refinement and Harmony Search with Harmony Refinement, respectively, as proposed by Atta, Mahapatra, and Mukhopadhyay (2019). The ga*p* values of each method based on the best results in the literature are reported in Appendix E. It is worth noting that the proposed WSA algorithm demonstrates the most favourable results among all the algorithms mentioned in the literature.

To ascertain whether the performance of WSA is significantly better than HS_RS and CP, we conducted the Friedman and Wilcoxon signed-rank tests as recommended by Derrac et al. (2011). The Friedman test assigns rankings to multiple algorithms, while the Wilcoxon signed-rank test determines whether there is a statistically significant difference between the performances of the compared algorithms. Table 9 reveals that, *based on the Friedman test results*, the WSA algorithm generates the best results among the other solution approaches ($p < 0.05$). Moreover, based on the Wilcoxon signed-rank test, the WSA algorithm performs significantly better than both HS_RS and CP as all p values less than 0.05.

5. Conclusions

In this paper, we conduct the modelling and solution of the turret-index optimisation problem using two mathematical programming approaches and a metaheuristic algorithm. The mathematical programming approaches are based on the QP formulation, which is adapted from the single row facility layout problem by extracting frequency and distance matrices from the process plans of the manufactured components. Despite linearising the QP formulation for problem resolution, both the QP and Linearized QP models prove to be inefficient for medium and larger-sized instances, providing solutions only for small-sized problems.

In the second mathematical programming approach, a new CP model is developed for the turret-index optimisation problem, which is the first of its kind in the literature. It is observed that CP is capable of providing optimal solutions for small-sized problems much more quickly, and it can also provide good solutions for medium and larger-sized problems within the specified computational time limits. Therefore, we can conclude that CP can be

Table 8. Computational results for larger sized test problem instances.

			The best reported solution via metaheuristics (HS_RS)			Constraint Programming (CP) Model					WSA Algorithm			GAP Values		
Instances	Number of tools (*m*)	Number of slots (*n*)	Obj.Val.	Standard Dev.	Sol. Time	#NV	#NC	Obj.Val.	Sol. Time	First Sol. Time	Obj.Val.	Standard Dev.	Sol. Time	HS_RS	CP	WSA
sko-36	36	60	20578	503.10	5.01	96	12714	**20574**	TLim	1360	**20574**	12.23	26.29	0.02	0.00	0.00
sko-42	42	60	**48812**	249.60	5.41	102	17352	48914	TLim	6820	**48812**	4.13	32.1	0.00	0.21	0.00
sko-49	49	60	73076	251.70	5.58	109	23673	73244	TLim	1960	**73034**	5.76	40.68	0.06	0.29	0.00
sko-56	56	60	105406	296.10	5.75	116	30974	105734	TLim	6780	**105330**	7.72	49.8	0.07	0.38	0.00
sko-64	64	100	190382	707.80	8.80	164	40518	190612	TLim	3840	**190380**	18.62	118.34	0.00	0.12	0.00
sko-72	72	100	265226	1058.80	9.58	172	51342	267780	TLim	7195	**265136**	23.95	78.13	0.03	1.00	0.00
sko-81	81	100	367868	824.30	10.19	181	65049	373098	TLim	5500	**367776**	31.11	171.81	0.03	1.45	0.00
sko-100	100	100	578138	1474.90	12.71	200	99306	580742	TLim	7150	**577966**	45.17	271.23	0.03	0.48	0.00
Y-30_t	30	60	**55346**	44.60	5.30	90	8796	**55346**	TLim	2150	**55346**	1.29	21.62	0.00	0.00	0.00
Y-35_t	35	60	**76028**	161.60	5.59	95	12011	**76028**	TLim	2410	**76028**	5.22	25.53	0.00	0.00	0.00
Y-40_t	40	60	92666	135.20	5.40	100	15726	92798	TLim	1830	**92652**	5.12	29.87	0.02	0.16	0.00
Y-45_t	45	60	117630	164.10	6.32	105	19941	117832	TLim	6520	**117620**	5.25	35.18	0.01	0.18	0.00
Y-50_t	50	60	146642	217.20	6.15	110	24656	147010	TLim	7070	**146630**	6.96	43.94	0.01	0.26	0.00
Y-60_t	60	100	222438	380.70	7.74	160	35586	223850	TLim	2770	**222412**	7.38	55.85	0.01	0.65	0.00
Anjos-60-01	60	100	**108106**	83.10	6.85	160	35586	**108126**	TLim	7110	**108106**	2.91	56.62	0.00	0.02	0.00
Anjos-60-02	60	100	**62548**	84.50	6.80	160	35586	62558	TLim	7032	**62548**	1.87	57.25	0.00	0.02	0.00
Anjos-60-03	60	100	47020	93.40	6.99	160	35586	47246	TLim	7070	**47018**	2.39	59.02	0.00	0.48	0.00
Anjos-60-04	60	100	**23182**	123.60	7.45	160	35586	23206	TLim	5425	**23182**	3.15	62.17	0.00	0.10	0.00
Anjos-60-05	60	100	**30336**	140.60	7.36	160	35586	30364	TLim	5500	**30336**	4.37	65.3	0.00	0.09	0.00
Anjos-70-01	70	100	**84592**	130.20	7.43	170	48516	84866	TLim	1410	**84592**	6.21	88.17	0.00	0.32	0.00
Anjos-70-02	70	100	**103446**	94.60	7.49	170	48516	103524	TLim	2445	**103446**	3.06	85.99	0.00	0.08	0.00
Anjos-70-03	70	100	**87588**	117.60	7.89	170	48516	87882	TLim	5190	**87588**	3.53	89.69	0.00	0.34	0.00
Anjos-70-04	70	100	55404	163.60	8.37	170	48516	55498	TLim	6990	**55402**	4.55	86.81	0.00	0.17	0.00
Anjos-70-05	70	100	**268476**	215.50	7.29	170	48516	268724	TLim	6650	**268476**	7.14	85.58	0.00	0.09	0.00
Anjos-75-01	75	100	**133260**	178.60	7.96	175	55731	133810	TLim	6735	**133260**	3.58	101.47	0.00	0.41	0.00
Anjos-75-02	75	100	**223612**	209.90	7.60	175	55731	223744	TLim	3650	**223612**	6.08	95.75	0.00	0.06	0.00
Anjos-75-03	75	100	**76302**	233.10	8.19	175	55731	76486	TLim	4040	**76302**	3.96	96.1	0.00	0.24	0.00
Anjos-75-04	75	100	**212682**	234.10	8.57	175	55731	212710	TLim	7090	**212682**	7.07	97.48	0.00	0.01	0.00
Anjos-75-05	75	100	**94034**	167.20	8.05	175	55731	94090	TLim	2680	**94034**	3.58	97.89	0.00	0.06	0.00
Anjos-80-01	80	100	**108926**	225.20	8.67	180	63446	109400	TLim	6240	**108926**	8.01	109.42	0.00	0.44	0.00
Anjos-80-02	80	100	105706	101.70	8.44	180	63446	105978	TLim	7090	**105702**	5.92	108.35	0.00	0.26	0.00
Anjos-80-03	80	100	**190182**	340.20	8.44	180	63446	190242	TLim	5710	**190182**	10.17	106.72	0.00	0.03	0.00
Anjos-80-04	80	100	**201656**	190.40	9.15	180	63446	202034	TLim	6260	**201656**	5.73	103.53	0.00	0.19	0.00
Anjos-80-05	80	100	72434	210.80	8.78	180	63446	72598	TLim	5645	**72428**	6.34	101.13	0.01	0.23	0.00

Obj. Val.: objective function value; Standard Dev.: standard deviation; N/A: not available; Sol. Time: solution time (second)

Table 9. Friedman and Wilcoxon signed-rank test results.

Average rankings achieved by Friedman test *p*-value = 1.0795E-12		Wilcoxon signed-rank test between WSA and other solution approaches	
Solution Approaches	Sum of Ranks	WSA vs.	*p*-value
WSA	1.3235 (1)	HS_RS	6.1035E-05
HS_RS	1.7794 (2)	CP	1.1735E-06
CP	2.8971 (3)		

considered as a suitable alternative solution approach for most practical applications, especially since many CNC machine tools have turrets with 16 or fewer slots.

Conversely, larger CNC machine tools often have larger turrets, some with chain-type configurations accommodating over 50 slots. In such cases, it is observed that mathematical programming approaches, especially QP and linearised QP, are not viable alternatives for solving the turret-index optimisation problem. As a result, a new metaheuristic solution approach based on the WSA algorithm is proposed in this study. Through comprehensive computational analysis and relevant statistical evaluations, the WSA algorithm demonstrates its ability to yield the best solutions for all test problems within a reasonable computational time. Furthermore, WSA outperforms some of the best-known results in the literature. Friedman and Wilcoxon signed-rank tests confirm the significant superiority of the proposed algorithm over CP and other metaheuristics.

As a prospect for future research, different metaheuristic algorithms can be utilised to solve the turret-index optimisation problem, allowing for a comparative study. In addition, the implementation of parallel programming can be employed to reduce computational times. Moreover, various versions of the turret-index problem, such as turret-index optimisation with setups and cutting tool duplications, can be addressed by extending the methodologies presented in this study.

Data availability statement

The data that support the findings of this study are openly available at http://kucse.in/tip.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Notes on contributors

Adil Baykasoglu received B.Sc., M.Sc., and PhD degrees in mechanical and industrial engineering areas from Gaziantep and Nottingham. He is presently a full professor at the Industrial Engineering Department of Dokuz Eylül University. He has published numerous academic papers, three books and edited several conference books on operational research, computational intelligence, and manufacturing systems design. He acts as editor and referee for many scientific journals. He is also an active member of many academic and professional institutions including the International Society of Agile Manufacturing, Turkish Chamber of Mechanical Engineers, Turkish Operational Research Association, etc. He has received awards from TUBA, TUBITAK, METU, and several other institutions for his scientific contributions.

Elif Yoruk received B.Sc., and M.Sc., degrees in industrial engineering areas from Yasar University and Dokuz Eylul University (DEU). She is presently a research assistant at the Industrial Engineering Department of DEU and working for her PhD thesis. Her research interests include mathematical modelling, intelligent optimisation, logistics and disaster management.

Seyda Topaloglu Yildiz is Professor of Industrial Engineering at Dokuz Eylul University (DEU), Türkiye. She received her Master and Ph.D.degrees in Industrial Engineering from DEU. Her research interests include mathematical modelling, constraint programming, metaheuristics, and their applications mainly on scheduling, assembly line balancing, and vehicle routing problems.

ORCID

Adil Baykasoglu http://orcid.org/0000-0002-4952-7239
Elif Yoruk http://orcid.org/0000-0001-5616-2189
Seyda Topaloglu Yildiz http://orcid.org/0000-0001-6827-126X

References

Amouzgar, K., A. Nourmohammadi, and A. H. C. Ng. 2021. “Multi-Objective Optimisation of Tool Indexing Problem: A Mathematical Model and a Modified Genetic Algorithm.” *International Journal of Production Research* 59 (12): 3572–3590. https://doi.org/10.1080/00207543.2021.1897174.

Anjos, M. F., A. Kennings, and A. Vannelli. 2005. “A Semidefinite Optimization Approach for the Single-row Layout Problem with Unequal Dimensions.” *Discrete Optimization* 2 (2): 113–122. https://doi.org/10.1016/j.disopt.2005.03.001.

Anjos, M. F., and G. Yen. 2009. “Provably Near-Optimal Solutions for Very Large Single-row Facility Layout Problems.” *Optimization Methods & Software* 24 (4–5): 805–817. https://doi.org/10.1080/10556780902917735.

Atta, S., P. R. S. Mahapatra, and A. Mukhopadhyay. 2019. “Solving Tool Indexing Problem Using Harmony Search Algorithm with Harmony Refinement.” *Soft Computing* 23:7407–7423. https://doi.org/10.1007/s00500-018-3385-5.

Baykasoglu, A. 2024. “Performance Analyses of Weighted Superposition Attraction-Repulsion Algorithms in Solving

Difficult Optimization Problems." *Network: Computation in Neural Systems*, 1–57. https://doi.org/10.1080/0954898X.2024.2367481.

Baykasoglu, A., and S. Akpinar. 2015. "Weighted Superposition Attraction (WSA): A Swarm Intelligence Algorithm for Optimization Problems - Part 2: Constrained Optimization." *Applied Soft Computing* 37:396–415. https://doi.org/10.1016/j.asoc.2015.08.052.

Baykasoglu, A., and C. Baykasoglu. 2020. "Optimal Design of Truss Structures Using Weighted Superposition Attraction Algorithm." *Engineering with Computers* 36 (3): 965–979. https://doi.org/10.1007/s00366-019-00744-x.

Baykasoglu, A., and C. Baykasoglu. 2021. "Weighted Superposition Attraction-Repulsion (WSAR) Algorithm for Truss Optimization with Multiple Frequency Constraints." *Structures* 30:253–264. https://doi.org/10.1016/j.istruc.2021.01.017.

Baykasoglu, A., and T. Dereli. 2004. "Heuristic Optimization System for the Determination of Index Positions on CNC Magazines with the Consideration of Cutting Tool Duplications." *International Journal of Production Research* 42 (7): 1281–1303. https://doi.org/10.1080/00207540310001622557.

Baykasoglu, A., and F. B. Ozsoydan. 2016. "An Improved Approach for Determination of Index Positions on CNC Magazines with Cutting Tool Duplications by Integrating Shortest Path Algorithm." *International Journal of Production Research* 54 (3): 742–760. https://doi.org/10.1080/00207543.2015.1055351.

Baykasoglu, A., and F. B. Ozsoydan. 2017. "Minimizing Tool Switching and Indexing Times with Tool Duplications in Automatic Machines." *The International Journal of Advanced Manufacturing Technology* 89:1775–1789. https://doi.org/10.1007/s00170-016-9194-z.

Baykasoglu, A., and M. E. Senol. 2019. "Weighted Superposition Attraction Algorithm for Combinatorial Optimization." *Expert Systems with Applications* 138:112792. https://doi.org/10.1016/j.eswa.2019.07.009.

Baykasoglu, A., and K. Subulan. 2020. "Capability-based Distributed Layout Formation with or Without Demand and Process Flow Information." *Applied Soft Computing* 94:106469. https://doi.org/10.1016/j.asoc.2020.106469.

Christofides, N., A. Mingozzi, and P. Toth. 1980. "Contributions to Quadratic Assignment Problem." *European Journal of Operational Research* 18 (4): 243–247. https://doi.org/10.1016/0377-2217(80)90108-3.

Dereli, T. 1998. *Development of a Process Planning System for Prismatic Parts." Unpublished PhD Thesis*. University of Gaziantep.

Dereli, T., and A. Baykasoglu. 2005. "OPPS-PRI 2.0: An Open and Optimized Process Planning System for Prismatic Parts to Improve the Performance of SMEs in the Machining Industry." *International Journal of Production Research* 43 (5): 1039–1087. https://doi.org/10.1080/0020754042000298548.

Dereli, T., A. Baykasoglu, N. N. Z. Gindy, and I. H. Filiz. 1998. "Determination of Optimal Turret-Index Positions of Cutting Tools by Using Genetic Algorithms." *Proceedings of the Second International Symposium on Intelligent Manufacturing Systems*, 743–750.

Dereli, T., and I. H. Filiz. 2000. "Allocating Optimal Index Positions on Tool Magazines Using Genetic Algorithms." *Robotics and Autonomous Systems* 33 (2-3): 155–167. https://doi.org/10.1016/S0921-8890(00)00086-5.

Derrac, J., S. García, D. Molina, and F. Herrera. 2011. "A Practical Tutorial on the use of Nonparametric Statistical Tests as a Methodology for Comparing Evolutionary and Swarm Intelligence Algorithms." *Swarm and Evolutionary Computation* 1 (1): 3–18. https://doi.org/10.1016/j.swevo.2011.02.002.

Falkenauer, E., and S. Bouffouix. 1991. "A Genetic Algorithm for Job Shop." *Proceedings of IEEE International Conference on Robotics and Automation* pp. 824–829.

Ghosh, D. 2016a. *Allocating Tools to Index Positions in Tool Magazines Using Tabu Search*. Ahmedabad: Indian Institute of Management Ahmedabad, Research and Publication Department.

Ghosh, D. 2016b. *Exploring Lin Kernighan Neighborhoods for the Indexing Problem*. Ahmedabad: Indian Institute of Management Ahmedabad, Research and Publication Department.

Ghosh, D. 2016c. *A new Genetic Algorithm for the Tool Indexing Problem*. Ahmedabad: Indian Institute of Management Ahmedabad, Research and Publication Department.

Krishna, A. G., and K. M. Rao. 2006. "Optimal Allocation of Index Positions on Tool Magazines Using An Ant Colony Algorithm." *The International Journal of Advanced Manufacturing Technology* 30:717–721. https://doi.org/10.1007/s00170-005-0099-5.

Schiex, T., and S. De Givry, eds. 2019. *Principles and Practice of Constraint Programming: 25th International Conference, CP* 2019, Stamford, CT, USA, Proceedings. Vol. 11802. Springer Nature.

Appendices

Appendix A: CPLEX OPL implementation for QP model

```
int numofTools=...;
int numofSlots=...;
range Tool=1..numofTools;
range Slot = 1..numofSlots;
// Parameters
int F[Tool][Tool]=...; // frequency matrix
int dist[Slot][Slot];
float t=...; // Indexing Time Between Two Adjacent Slots
// Calculate the distance btw slots
execute{
for (var i=1; i<numofSlots; i++) {
       for (var j=i+1; j<numofSlots+1; j++) {
              dist[i][j]=Math.min(Math.abs(j−i),Math.abs(i+numofSlots−j));
              dist[j][i]=dist[i][j];}}}
// Decision Variables
dvar boolean y[Slot][Tool];
// Objective Function
minimize sum(i in Slot, j in Slot: i!=j, k in Tool, l in Tool: l!=k)
t∗dist[i][j]∗F[k][l]∗y[i][k]∗y[j][l];
// Constraints
subject to{
c1:
forall (k in Tool)
      sum(i in Slot) y[i][k]==1;
c2:
forall (i in Slot)
       sum(k in Tool) y[i][k]<=1;}
```

Appendix B: CPLEX OPL implementation for LQP and LQP_vi models

```
int numofTools=...;
int numofSlots=...;
range Tool=1..numofTools;
range Slot = 1..numofSlots;
// Parameters
int F[Tool][Tool]=...; // frequency matrix
int dist[Slot][Slot];
float t=...; // Indexing Time Between Two Adjacent Slots
// Calculate the distance btw slots
execute{
for (var i=1; i<numofSlots; i++) {
       for (var j=i+1; j<numofSlots+1; j++) {
              dist[i][j]=Math.min(Math.abs(j−i),Math.abs(i+numofSlots−j));
              dist[j][i]=dist[i][j];}}}
// Decision Variables
dvar boolean y[Slot][Tool];
dvar boolean z[Slot][Slot][Tool][Tool];
// Objective Function
minimize sum(i in Slot, j in Slot: i!=j, k in Tool, l in Tool: l!=k)
t∗dist[i][j]∗F[k][l]∗z[i][j][k][l];
// Constraints
subject to{
c1:
forall (k in Tool)
      sum(i in Slot) y[i][k]==1;
c2:
forall (i in Slot)
```

```
      sum(k in Tool) y[i][k]<=1;
c3
forall (i in Slot, j in Slot, k in Tool, l in Slot)
      z[i][j][k][l]>=y[i][k]+y[j][l]−1;}
//END OF THE LQP MODEL

c4: //Valid Inequalities (Additional constraint for LQP_vi model)
forall (i in Slot: i>numofTools)
      sum(k in Tool) y[i][k]==0;}
//END OF THE LQP_vi MODEL
```

Appendix C: CPLEX OPL implementation for CP model

```
using CP;
int numofTools=...;
int numofSlots=...;
range Tool=1..numofTools;
range Slot = 1..numofSlots;
range nSlot = 1..numofTools;
range half1Slot= 1..(numofSlots div 2)+numofSlots % 2;
range half2Slot= (numofSlots div 2)+numofSlots % 2+1..numofSlots;
// Parameters
int F[Tool][Tool]=...; // frequency matrix
int dist[Slot][Slot];
float t=...; // Indexing Time Between Two Adjacent Slots
// Calculate the distance btw slots
execute{
for (var i=1; i<numofSlots; i++) {
       for (var j=i+1; j<numofSlots+1; j++) {
              dist[i][j]=Math.min(Math.abs(j−i),Math.abs(i+numofSlots−j));
              dist[j][i]=dist[i][j];}}}
// Decision Variables
dvar int u[Tool] in nSlot;
dvar boolean x[Slot];
dexpr float cost = sum (i in Tool, j in Tool: i!=j)
dist[u[i]][u[j]]*F[i][j]*t;
// Objective Function
minimize cost;
// Constraints
subject to{
c1:
allDifferent(u);
c2:
max(i in Tool)u[i]==numofTools;
c3:
min(i in Tool)u[i]==1;
c4:
forall (i in Tool)
        x[u[i]]==1;
c5:
sum (i in Slot)x[i]==numofTools;
c6:
sum (i in half1Slot) x[i] >=sum (j in half2Slot) x[j];
}
```

Appendix D: Computational results of metaheuristic algorithms and the proposed approaches in the literature for Anjos and sko instances

	TS Ghosh (2016a)		LS-LK Ghosh (2016b)		LS-LK1 Ghosh (2016b)		TS-LK1 Ghosh (2016b)		DFGA Ghosh (2016c)		GA Ghosh (2016c)		HS_woRS Atta et al. (2019)		HS_RS Atta et al. (2019)		CP		WSA	
Instances	Obj. Val.	Sol. Time	Obj. Val.	Sol. Time	Obj. Val.	Sol. Time	Obj. Val.	Sol. Time	Obj. Val.	Sol. Time.	Obj. Val.	Sol. Time	Obj. Val.	Sol. Time	Obj. Val.	Sol. Time	Obj. Val.	Sol. Time	Obj. Val.	Sol. Time
Anjos-60-01	**108106**	179.47	**108106**	9.17	**108106**	8.95	**108106**	350.30	152200	21.06	108126	288.17	123998	33.04	**108106**	6.85	**108126**	TLim	**108106**	56.62
Anjos-60-02	62556	180.11	62556	9.20	62556	9.22	62556	350.08	88930	21.39	62562	291.67	70522	45.04	**62548**	6.80	62558	TLim	**62548**	57.25
Anjos-60-03	47020	180.11	47028	8.55	47020	8.59	47020	350.31	67606	21.16	47028	280.62	52420	45.44	47020	6.99	47246	TLim	**47018**	59.02
Anjos-60-04	23184	178.24	23184	8.84	23184	8.80	23184	350.23	35294	21.02	23184	282.52	27950	43.30	**23182**	7.45	23206	TLim	**23182**	62.17
Anjos-60-05	30364	178.23	30364	8.92	30364	8.91	30364	349.50	45082	21.13	**30336**	289.67	37132	37.52	**30336**	7.36	30364	TLim	**30336**	65.3
Anjos-70-01	84594	178.45	84598	9.81	84594	9.88	84594	350.64	114526	21.09	84624	394.47	95306	42.23	**84592**	7.43	84866	TLim	**84592**	88.17
Anjos-70-02	**103446**	178.05	**103446**	10.31	**103446**	10.22	**103446**	349.33	137216	21.05	**103446**	405.02	120480	41.19	**103446**	7.49	103524	TLim	**103446**	85.99
Anjos-70-03	87590	178.50	87590	9.92	87590	10.03	87590	349.75	112602	21.14	**87588**	406.52	99164	38.71	**87588**	7.89	87882	TLim	**87588**	89.69
Anjos-70-04	55410	178.84	55410	10.17	55410	10.45	55410	349.42	72814	21.31	55454	405.55	69246	36.58	55404	8.37	55498	TLim	**55402**	86.81
Anjos-70-05	268538	181.48	268538	9.89	268538	9.81	268538	349.72	340024	21.17	268526	396.53	291096	39.71	**268476**	7.29	268724	TLim	**268476**	85.58
Anjos-75-01	133312	182.20	133312	10.52	133312	10.36	133312	350.55	165296	21.14	133372	457.41	146820	37.68	**133260**	7.96	133810	TLim	**133260**	101.47
Anjos-75-02	**223612**	181.45	223700	10.77	**223612**	10.61	**223612**	348.64	278546	21.13	223628	454.02	241516	40.93	**223612**	7.60	223744	TLim	**223612**	95.75
Anjos-75-03	76316	181.86	76316	10.77	76316	10.36	76316	350.05	97890	21.25	76320	453.67	86328	44.08	**76302**	8.19	76486	TLim	**76302**	96.1
Anjos-75-04	**212682**	179.53	**212682**	11.55	**212682**	11.34	**212682**	349.74	260248	21.36	**212682**	478.86	232120	47.25	**212682**	8.57	212710	TLim	**212682**	97.48
Anjos-75-05	94062	179.14	94062	11.08	94062	10.97	94062	350.45	117538	21.23	94066	463.59	105022	39.09	**94034**	8.05	94090	TLim	**94034**	97.89
Anjos-80-01	**108926**	178.86	**108926**	11.06	**108926**	11.03	**108926**	349.31	138896	21.16	**108926**	519.59	121202	45.46	**108926**	8.67	109400	TLim	**108926**	109.42
Anjos-80-02	105706	179.02	105706	11.27	105706	11.63	105706	349.52	128586	21.16	105742	522.31	115204	44.73	105706	8.44	105978	TLim	**105702**	108.35
Anjos-80-03	**190182**	182.61	**190182**	11.61	190182	11.33	190182	348.45	230408	21.11	190186	519.72	205896	46.57	**190182**	8.44	190242	TLim	**190182**	106.72
Anjos-80-04	201900	184.23	201902	11.37	201900	11.06	201900	348.53	240862	21.14	201696	520.14	215882	45.58	**201656**	9.15	202034	TLim	**201656**	103.53
Anjos-80-05	72454	185.53	72456	11.39	72456	11.31	72454	349.70	89934	21.33	72466	530.20	81466	39.34	72434	8.78	72598	TLim	**72428**	101.13
sko-36	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	22922	49.49	20578	5.01	**20574**	TLim	**20574**	26.29
sko-42	48816	38.28	48816	1.27	48816	1.23	48816	73.41	61678	8.48	48820	52.31	50456	47.98	**48812**	5.41	48914	TLim	**48812**	32.1
sko-49	73164	38.08	73164	1.42	73164	1.48	73164	73.11	84726	8.48	73304	65.83	75446	54.90	73076	5.58	73244	TLim	**73034**	40.68
sko-56	105948	38.03	105948	1.67	105948	1.69	105948	72.94	117956	8.45	105854	81.27	109232	50.43	105406	5.75	105734	TLim	**105330**	49.8
sko-64	190674	182.06	190674	10.39	190674	10.28	190674	349.73	247118	21.20	191088	387.56	215998	42.07	190382	8.8	190612	TLim	**190380**	118.34
sko-72	267214	182.56	267264	11.31	267214	11.16	267214	349.87	324622	21.17	265742	497.02	297128	41.14	265226	9.58	267780	TLim	**265136**	78.13
sko-81	371704	183.08	371704	12.53	371704	12.11	371704	349.64	426844	21.11	369058	587.31	401214	42.13	367868	10.19	373098	TLim	**367776**	171.81
sko-100	580992	185.00	581034	14.95	580992	15.33	580992	350.19	642436	21.28	578896	835.08	611072	43.69	578138	12.71	580742	TLim	**577966**	271.23

Obj. Val.: objective function value; N/A: not available; Sol. Time: solution time (second); TLim: Time limit (120 min)

Appendix E: Gap values of the proposed CP and metaheuristic algorithms in the literature according to the best results for Anjos and sko instances

Instances	TS Ghosh (2016a)	LS-LK Ghosh (2016b)	LS-LK1 Ghosh (2016b)	TS-LK1 Ghosh (2016b)	DFGA Ghosh (2016c)	GA Ghosh (2016c)	HS_woRS Atta, Mahapatra, and Mukhopadhyay (2019)	HS_RS Atta, Mahapatra, and Mukhopadhyay (2019)	CP	WSA
Anjos-60-01	0.000	0.000	0.000	0.000	40.788	0.019	14.700	0.000	0.019	0.000
Anjos-60-02	0.013	0.013	0.013	0.013	42.179	0.022	12.749	0.000	0.016	0.000
Anjos-60-03	0.004	0.021	0.004	0.004	43.787	0.021	11.489	0.004	0.485	0.000
Anjos-60-04	0.009	0.009	0.009	0.009	52.247	0.009	20.568	0.000	0.104	0.000
Anjos-60-05	0.092	0.092	0.092	0.092	48.609	0.000	22.402	0.000	0.092	0.000
Anjos-70-01	0.002	0.007	0.002	0.002	35.386	0.038	12.666	0.000	0.324	0.000
Anjos-70-02	0.000	0.000	0.000	0.000	32.645	0.000	16.467	0.000	0.075	0.000
Anjos-70-03	0.002	0.002	0.002	0.002	28.559	0.000	13.216	0.000	0.336	0.000
Anjos-70-04	0.014	0.014	0.014	0.014	31.428	0.094	24.988	0.004	0.173	0.000
Anjos-70-05	0.023	0.023	0.023	0.023	26.650	0.019	8.425	0.000	0.092	0.000
Anjos-75-01	0.039	0.039	0.039	0.039	24.040	0.084	10.176	0.000	0.413	0.000
Anjos-75-02	0.000	0.039	0.000	0.000	24.567	0.007	8.007	0.000	0.059	0.000
Anjos-75-03	0.018	0.018	0.018	0.018	28.293	0.024	13.140	0.000	0.241	0.000
Anjos-75-04	0.000	0.000	0.000	0.000	22.365	0.000	9.139	0.000	0.013	0.000
Anjos-75-05	0.030	0.030	0.030	0.030	24.995	0.034	11.685	0.000	0.060	0.000
Anjos-80-01	0.000	0.000	0.000	0.000	27.514	0.000	11.270	0.000	0.435	0.000
Anjos-80-02	0.004	0.004	0.004	0.004	21.650	0.038	8.989	0.004	0.261	0.000
Anjos-80-03	0.000	0.000	0.000	0.000	21.151	0.002	8.263	0.000	0.032	0.000
Anjos-80-04	0.121	0.122	0.121	0.121	19.442	0.020	7.055	0.000	0.187	0.000
Anjos-80-05	0.036	0.039	0.039	0.036	24.170	0.052	12.479	0.008	0.235	0.000
sko-36	N/A	N/A	N/A	N/A	N/A	N/A	11.412	0.019	0.000	0.000
sko-42	0.008	0.008	0.008	0.008	26.358	0.016	3.368	0.000	0.209	0.000
sko-49	0.178	0.178	0.178	0.178	16.009	0.370	3.303	0.058	0.288	0.000
sko-56	0.587	0.587	0.587	0.587	11.987	0.497	3.705	0.072	0.384	0.000
sko-64	0.154	0.154	0.154	0.154	29.803	0.372	13.456	0.001	0.122	0.000
sko-72	0.784	0.803	0.784	0.784	22.436	0.229	12.066	0.034	0.997	0.000
sko-81	1.068	1.068	1.068	1.068	16.061	0.349	9.092	0.025	1.447	0.000
sko-100	0.524	0.531	0.524	0.524	11.155	0.161	5.728	0.030	0.480	0.000
Average	0.137	0.141	0.138	0.137	27.936	0.092	11.429	0.009	0.271	0.000