A bio-inspired evolution-development method for modelling and optimisation of buffer allocation in unreliable serial production line

Zhiwei Zhao [1], Paul Goodall [2], Andrew West [2], Andrew Colligan [1], Imelda Friel [1], Simon Hickinbotham [3], Mark Price [1], Yan Jin [1*]

[1] School of Mechanical and Aerospace Engineering, Queen's University Belfast, United Kingdom
[2] Wolfson School of Mechanical, Electrical and Manufacturing Engineering, Loughborough University, Loughborough, United Kingdom
[3] Department of Computer Science, University of York, York, United Kingdom
[*] y.jin@qub.ac.uk

Abstract: A buffer is an important element in a production line and its allocation influences the throughput and inventory of the line. The buffer allocation problem can be framed as a multi-objective optimisation problem and is often addressed by using meta-heuristic algorithms, such as evolutionary algorithms. However, these algorithms primarily focus on the "genetic evolution" aspect and do not take in to account the impact of the biological "organism development" process, potentially constraining the exploration of the solution space. In this paper, a bio-inspired evolution-development (evo-devo) approach for modelling and optimising buffer allocations is proposed. The organism representing a production line is defined and modelled, and the evolution and development processes of organisms are developed for researching optimised solutions. The method has been validated by a simulation of a buffer allocation optimisation in an unreliable serial production line with multi-objectives, aiming to maximize production throughput and minimize the total buffer size. Results show that the proposed method can efficiently obtain solutions, while also achieving greater exploration of the solution space than competing evolutionary algorithms such as the Non-Dominated Sorting Genetic Algorithm II. The proposed approach's functionality means that it could be applied to other areas of generative design of future factories.

1. Introduction

With the advancement of big data and intelligent technology, the landscape of manufacturing is undergoing a significant transformation from traditional automation to the paradigm of intelligent manufacturing [1, 2]. This evolution is in direct response to changing market demands, necessitating the development of intelligent factories capable of adaptable production capabilities [3, 4].

Intelligent factories aim to revolutionise manufacturing processes by enhancing resource utilisation, cost reduction, throughput improvement, and environmental impact minimisation. Leveraging the power of data analysis and intelligent decision-making, these advanced manufacturing facilities can precisely adjust their operations to align with specific production objectives. This shift towards intelligent manufacturing is applicable across various industries, including automotive and aerospace manufacturing, allowing them to stay competitive and responsive in a rapidly changing marketplace.

In intelligent factories, the production line is a crucial component. However, due to a variety of unexpected problems such as machine failure, product quality problems, etc., production lines can be disrupted, affecting the throughput [5]. To alleviate this situation, buffers are commonly incorporated into the production line. These buffers serve as storage points for in-process components in the event of a production station's unexpected stoppage. Thus, the introduction of buffers can limit the propagation of unexpected interruptions caused by machine stoppages and maintain the throughput of the entire production line. However, setting buffers in a production line will also occupy space and inventory requirements in production lines, increasing the investment and operational costs. Therefore, the proper allocation of buffers can help achieve higher production line throughput and greater reliability, whilst the design of buffer size is critical to increasing productivity and reliability at an acceptable additional cost.

The optimisation problem of buffer allocation has been the focus of academic and industrial research [6]. Various optimisation methods have been used to optimise buffer size, such as analytical methods, knowledge-based methods [7], search methods and the meta-heuristics method. The analytical method is only applicable to simple production lines, in which the performance of the solution could be calculated by evaluative approaches, e.g. Markovian analysis [8], [9]. For long production lines with a large number of machine stations and buffers, it is difficult to build a mathematical model of the buffer allocation problem, and the buffer size optimisation problem is NP-hard. Therefore, the main method of the search algorithm is to divide the overall optimisation problem into sub problems to build a mathematical model, and then a search method, such as gradient descent, address the integrated problem [10]. Xi et al. [11] decomposed a long production line into many sub-lines and obtained the a near optimal solution of the whole production line through the optimisation of the subsystems. However, buffer allocation problems usually require non-convex optimisation, and it is difficult to find optimised results by search methods when complex nonlinear constraints are involved.

For this kind of complex, discrete and nonlinear buffer allocation optimisation problem, a meta-heuristics algorithm combined with discrete event simulation [12, 13] is also an effective method. Nabil [14] proposed an extended great deluge algorithm for buffer allocation and preventive maintenance optimisation. Zhou et al. [15] presented an optimisation method combining particle swarm optimisation

and distribution estimation algorithm. Yelkenci et al. [16] proposed a hybrid evolutionary approach for multi-objectives buffer allocation problem in open serial production lines, in which a Non-dominated Sorting Genetic Algorithm (NSGA-II) and multi-objective simulated annealing are mixed to discover pareto front sets. For solving the buffer allocation problem in large unbalanced production lines, Yasmine et al. [17] proposed a multi-objective resolution approach for solving an energy-efficient buffer allocation problem, in which the weighted sum and epsilon-constraint methods as well as the elitist non-dominated sorting genetic algorithm are adapted. The advantage of meta-heuristic algorithms is that they are not dependent on complex mathematical model and can be readily optimised using appropriate rules and objective functions.

Current meta-heuristic algorithms only consider the "genetic evolution" process, such as evolutionary algorithms, but do not incorporate a biological process of "organism development". Both evolution (evo) and development (devo) are important for a population's survival in an environment, which is the focus of evo-devo approach reported within this paper. Evolution processes control how populations evolve and adapt to environments over time, by altering their genetic information to adapt to different survival conditions. Development, on the other hand, is primarily the internal process within an organism, involving gene expression and the gradual changes in individual organisms. For organisms, evolution and development are intertwined processes, evolution influences the developmental process, and in turn, the developmental process can also influence evolution [18]. It is the hypothesis of this research that considering both evolution and development processes in algorithms has the potential to enhance search capabilities and find more suitable solutions within a given environment. Recently, Simon et.al [19] proposed a evo-devo framework for structure design optimization and analysis, in which the evo-devo process is configured to be a search process to find optimal solutions in a high-dimensional design space.

In this paper, a bio-inspired evo-devo approach is proposed for modelling and optimising the buffer allocation problem. The fundamental elements of a production line are conceptualized as organisms and the rules governing organisms are defined and modelled. Within the framework of the evo-devo method, the rules and processes that dedicate to the evolution and development of organisms within a production line with buffers is described. The method has been validated using a simulation case-study, and the results indicate that the evo-devo method can effectively provide an optimised buffer allocation solution. The contributions of this paper are as follows.

(1) Modelling the buffer allocation optimisation problem as an evolution-development (evo-devo) process of a bio-organism for the first time.

(2) The evo-devo framework [19], which was used in design and structure analysis, is deployed to solve the production problem.

(3) An evo-devo algorithm is developed compared with NSGA-II algorithm for this buffer allocation problem, and the results show the evo-devo approach is superior to the NSGA-II algorithm in terms of solution volume.

(4) The modelling method and the evo-devo algorithm can be well applied to other manufacturing problems.

2. Problem description and definition

In this paper a serial production line that includes n unreliable machine stations $M_i, i \in (1, \dots, n)$ and $n-1$ buffers $B_i, i \in (1, \dots, n-1)$ is considered. As shown in Fig. 1, a buffer B_i (represented by a triangle) is the temporal material storage between two consecutive machine stations M_i and M_{i+1} (represented by circles). The arrows specify the direction of the material/part flow in the serial production line.

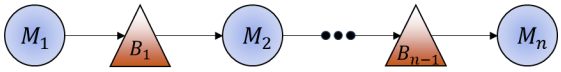

Fig. 1 *The structure of a serial production line with buffers*

Based on the above structure, the following assumptions are made.

(1) In this serial production line, the first machine station on the production line has sufficient supply of raw materials, and the last machine station does not block exits from the line.

(2) With the exception of the first machine station, each machine station requires the in-process components from its input buffer and if there is no in-process component in that buffer, the machine station is in a stagnant state.

(3) After each part is completed by a machine station, it is automatically stored in its output buffer and when the number of parts in the downstream buffer reaches its storage limit, the machine will stop production an enter stagnant state.

(4) The probability of a machine failure follows uniform distribution, i.e., the machine will randomly produce some faults in the process and the maintenance of the fault takes a predetermined time. Each machine station M_i is characterised by a failure rate f_i, a repair time r_i, and a machine time p_i.

It should be noted that when a machine station fails, the process will be interrupted, and the parts cannot be processed. At this time, the upstream (i.e. previous) machine station may become blocked when the buffer reaches its maximum capacity, and the downstream machine station may be stalled due to the lack of parts to be machined, affecting the output of the entire production line. Therefore, the purpose of increasing the buffer between these machine stations is to avoid the stagnation of the overall production line caused by machine failure, thereby improving the efficiency of the production line.

The goal is to determine the optimal buffer allocation to minimise the total buffer size whilst maximising the production line throughput. The multi-objective optimal design problem is formulated as a combinatorial optimisation problem to find the Pareto optimal set of $\boldsymbol{B}(B_1, \dots, B_{n-1})$ to

$$Maximise\ O = f(\boldsymbol{B}) \tag{1}$$

$$Minimise\ S = \sum_{i=1}^{n-1} B_i, i = 1, \dots, n-1 \tag{2}$$

$$Subject\ to\ \ 0 < B_i \leq B_{up} \tag{3}$$

where O is the throughput of the production line under a buffer design solution $\boldsymbol{B}$ calculated by discrete-event simulation, and S represents the total size of all buffers. B_{up} is the upper limit of each buffer.

3. Evo-devo method for the buffer allocation problem

The modelling process and method of buffer allocation optimisation from the evo-devo perspective is discussed in this section.

3.1. Model of cell and organism for buffer allocation problem

For the buffer allocation optimisation problem, the goal is to search for an optimised solution by varying the buffer size, i.e. to evolve and develop an optimised organism by evolving and developing the cells in the organism i.e. by defining a buffer allocation genome. In order to use the evo-devo approach the form of organism and cell are defined, as illustrated in Fig. 2.

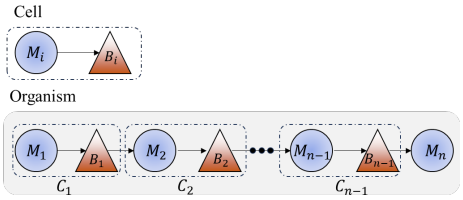

Fig. 2 *Cells and the organism in production line*

The production line to be optimised is defined as an organism, which is created by a series of cells. A cell is the fundamental component of an organism and each can be modelled in different ways according to the needs of the optimisation problem. In this paper, a cell includes a machine station node and a buffer node, both of which have different attributes. The machine station's attributes encompass station capacity, machining time, maintenance time, and station serial number. The buffer's attributes include buffer size and serial number. The cells are arranged according to the sequence of the production operations to generate the organism. The organism's evo-devo process is to optimise the size of the buffer in each cell within the constraints of minimising the overall buffer size within the organism.

3.2. Evo-devo algorithm on buffer allocation optimisation problem

In the evo-devo process, Gene Regulatory Networks (GRN) play a crucial role [21]. The GRNs are bioinformatics models used to describe and analyse how genes interact with each other to control the development and various biological processes of an organism. GRNs produce changes during the population's evolutionary process through mechanisms like gene recombination and mutations, as well as alterations in regulatory mechanisms. Subsequently, during the stages of biological development, these changes in the GRN drive the development of the organism by controlling gene expression and interactions.

To realise the evo-devo process, the GRN and rules of the evolution and development of the organism need be modelled and defined. In the evo-devo process reported in this paper, each solution of buffer allocation could be deemed as an organism developed via a GRN. In a development process, a GRN determines the final state of the organism through gene expression and interaction between genes. Meanwhile, gene and regulatory relationships in a GRN are optimised in the evolution process. Therefore, in the evo-devo process, the GRNs not only evolve, but also act on cells to determine the cell state, i.e. developing organisms. To realise the functions of GRNs, the NEAT framework, which utilizes a neural network to output cell states and evolves itself based on feedback from the environment, is adopted. In the neural network, the neurons represent the genes, and the weights associated with the connection between neurons describe the adjustment relationship.

The functional flow of the evo-devo algorithm is shown in Fig. 3. First, the parameters of the algorithm need to be initialized, which include the number of development steps, the number of generations in the evolutionary process, the population size of each generation and the number of neural network units, and these parameters are determined based on the complexity of the problem, the available computational resources and the convergence behaviour observed during preliminary experiments. The initialised parameters also include, for example, values for the weights, biases and interconnected neurons. In the development process of an organism, the GRN acts on the cells to develop the organism. There are a number of iterations based on the number of development steps. For each development step, the GRN acts on each cell, taking the state of each cell as input and outputting the value of the cell's buffer size. This process continues until the state of all cells has been updated, resulting in the formation of an organism (i.e. solution). The development process is a constant and periodic operation process, i.e., the neural network takes the current cell state as input and updates the cell state. It should be noted that the states of individual cells within the organism are different (buffer sizes are different), and growth of cells of the current state depends on the action of the GRN. In the buffer optimisation problem, the cell states input to the GRN includes station capacity, machining time, buffer size, and serial number, with output being the corresponding buffer

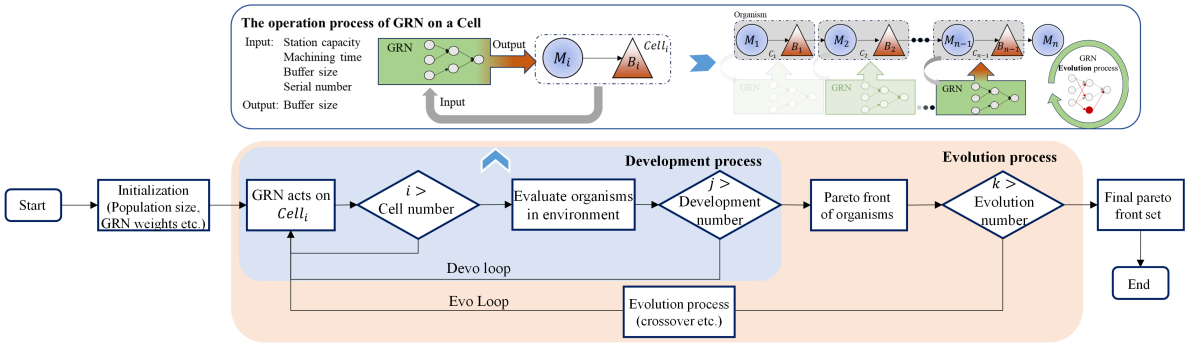

Fig. 3 *Evo-devo algorithm architecture*

size. Finally, the evolution of organisms is reflected by the evolution of the GRN. During evolution, based on the effects (calculated through a fitness function) of each organism developed by its GRN, the weights, bias and connection structures of the GRN are evolved through crossover, etc., to generate a GRN that may be better suited to the environment. This process is repeated until the evolution iterations are terminated, resulting in the Pareto front set.

4. Case study

The evo-devo method is evaluated using a simulation case and compared with the NSGA-II method in this section. The simulation environment was developed using Simpy software package, and the evo-devo was developed using NEAT-Python package, while the NSGA-II utilised the Pymoo package [22].

4.1. Description of the test case

To represent realistic behaviour of a production line, a discrete event simulation environment based on Simpy is employed. The production line consists of 10 machine stations and 9 buffers, since the first machine operates without a buffer. The upper limit of an individual buffer's size B_{up} is 30, while the lower limit is 1. The simulation time adopted 500 hours. Every machine has a fault rate determined by a random number uniformly distributed between 0 and 1. Due to the machine fault rates, the throughput of the production line has uncertainties. Therefore, for each solution, the throughput is obtained using the average value from 100 repetitions, to determine the variation of the throughput value. Table 1 displays the parameters of this production line.

Table 1 Production line parameters

Machine M_i	Machine time p_i	Fault rate f_i	Repair time r_i
M_1	1	0.35	9
M_2	1	0.38	12
M_3	1	0.31	9
M_4	1	0.33	12
M_5	1	0.35	6
M_6	1	0.4	12
M_7	1	0.42	12
M_8	1	0.29	12
M_9	1	0.26	12
M_{10}	1	0.25	12

4.2. Test results

As described in equations (1) to (3), the objective of this case is to maximise the production line throughput while minimising the overall buffer size. The proposed method was compared with the NSGA-II algorithm. The initialised neural network using NEAT consists of 12 hidden neurons evenly distributed in 4 layers, which is to allow the neural network to have a stronger fitting ability than the original NEAT framework with only one hidden layer. During the evolution process, the number of hidden neurons, connections, and weights change. In order to limit the output of the neural network between 1 and 30 (lower and upper limits of a buffer's size), the activation function is *sin* function, and the absolute value of the final output is enlarged by a factor of 30. The settings for NSGA-II were consistent with the evo-devo method, where the population size was 30 and the number of generation iterations was 10. After the optimisation process converged, Pareto front sets were obtained for each of the two methods, as illustrated in Fig. 4.

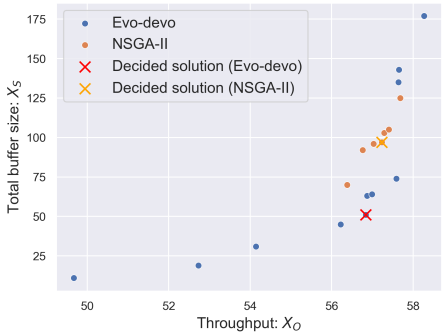

Fig. 4 *Pareto front set of different algorithms, where evo-devo obtain a wider Pareto front set compared to NSGA-II.*

From the data and annotations in ***Fig. 4***, it can be seen that the evo-devo method is effective in finding suitable solutions, i.e., generating the Pareto front. Moreover, to quantitatively evaluate the effectiveness of the optimisation methods, Hypervolumes, which provide a measure of the volume of the objective space covered by the Pareto front, were calculated for both methods. In this comparison, evo-devo achieved a Hypervolume of 0.94, surpassing NSGA-II with 0.69. This result underscores the superior performance of evo-devo in exploring a more extensive region of the Pareto front, further highlighting its efficacy in multi-objective optimisation tasks.

In order to select appropriate solutions $\boldsymbol{X}$ within the non-dominated solutions, an entropy evaluation method has been utilised. The process involves calculating the entropy of each objective in the Pareto front and then assigning a weight. For two objective attributes (x_O and x_S) of this multi-objective problem, a higher entropy indicates increased uncertainty in the attribute. Therefore, to reduce the uncertainty of the decision, the attribute with higher entropy is allocated a lower weight. The calculation is as follows:

$$p_{ij} = \frac{x_{ij}^{norm}}{\sum_{i=1}^{q} x_{ij}^{norm}}, j = (O, S); i = (1 \dots q) \tag{4}$$

$$E_j = -\ln(q)^{-1} \sum p_{ij} \ln p_{ij} \tag{5}$$

$$w_j = \frac{1-E_j}{\sum(1-E_j)} \tag{6}$$

$$X_i = w_O x_{iO}^{norm} + w_S x_{iS}^{norm} \tag{7}$$

where x_{ij}^{norm} is the normalised value of i-th solution in Pareto front set with objective j, E_j is the entropy of objective j, w_j is the weight of objective j, X_i is the score of i-th solution in Pareto front set, which is used to determine whether the solution is chosen or not.

First, the two objective values $(\boldsymbol{x_o}, \boldsymbol{x_s})$ of the Pareto front set $\boldsymbol{x} = (\boldsymbol{x_o}, \boldsymbol{x_s})$ are normalised to obtain $\boldsymbol{x}_{iO}^{norm}$ and $\boldsymbol{x}_{iS}^{norm}$ for each objective set. Then the entropy E_j for each objective set is calculated. After obtaining the entropy of the

two objectives, the weight of each objective (w_O, w_S)is calculated. Finally, from the scores of each solution, the best is that with the highest score. According to the entropy weight method, the best solution selected from the Pareto front is represented by the red crosses in Fig. 4. The capacities for buffer allocation is as shown in Table 2.

Table 2 Buffer size of the decided solution

B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9
7	7	6	6	6	5	5	5	4

Based on the buffer allocation solution, the average throughput of the production line is 56.83 with a variance of 17.8 from 100 simulation results. This is to be compared with a throughput of 48.8 when all of the buffer sizes are set to 1. The result demonstrates a 16.4% increase in throughput under the condition of a reasonable buffer allocation.

5. Discussion

Under the evo-devo framework, a new modelling and optimisation approach is for the first time developed for the buffer allocation optimisation problem. The organisms and cells used to represent a production line have been modelled, and an evo-devo algorithm is developed to deal with the optimisation problem. Through a multi-objective case, aiming to maximise production line throughput and reduce total buffer size, the feasibility of the approach is validated. Compared with NSGA-II, the proposed approach can obtain a greater hypervolume of the Pareto front with the same number of generations, showing a 49% increase. The modelling method and the evo-devo algorithm can be well applied to other manufacturing problems.

In future, the generalisation applicability of this method will be investigated, with the expectation that the GRN established based on neural networks can be readily deployed in similar environments and adapt to new environments. Additionally, application of the method in various production line problems such as the job shop scheduling problem will also be investigated.

Acknowledgments

This work was supported by EPSRC programme Grant Ref is EP/V007335/1, "RIED: Re-Imagining Engineering Design".

References

1 Morgan, J., Halton, M., Qiao, Y., Breslin, J.G.: 'Industry 4.0 smart reconfigurable manufacturing machines'*J. Manuf. Syst.*, 2021, **59**, pp. 481–506.

2 Tao, F., Qi, Q., Liu, A., Kusiak, A.: 'Data-driven smart manufacturing'*J. Manuf. Syst.*, 2018, **48**, pp. 157–169.

3 He, B., Bai, K.J.: 'Digital twin-based sustainable intelligent manufacturing: a review'*Adv. Manuf.*, 2021, **9**, (1), pp. 1–21.

4 Kang, Z., Catal, C., Tekinerdogan, B.: 'Machine learning applications in production lines: A systematic literature review'*Comput. Ind. Eng.*, 2020, **149**, (April), p. 106773.

5 Renna, P.: 'Adaptive policy of buffer allocation and preventive maintenance actions in unreliable production lines'*J. Ind. Eng. Int.*, 2019, **15**, (3), pp. 411–421.

6 Demir, L., Tunali, S., Eliiyi, D.T.: 'The state of the art on buffer allocation problem: A comprehensive survey'*J. Intell. Manuf.*, 2014, **25**, (3), pp. 371–392.

7 Vouros, G.A., Papadopoulos, H.T.: 'Buffer allocation in unreliable production lines using a knowledge based system'*Comput. Oper. Res.*, 1998, **25**, (12), pp. 1055–1067.

8 Alexandros, D.C., Chrissoleon, P.T.: 'Exact analysis of a two-workstation one-buffer flow line with parallel unreliable machines'*Eur. J. Oper. Res.*, 2009, **197**, (2), pp. 572–580.

9 Zhao, C., Li, J., Huang, N., Horst, J.A.: 'Flexible Serial Lines with Setups: Analysis, Improvement, and Application'*IEEE Robot. Autom. Lett.*, 2017, **2**, (1), pp. 120–127.

10 Weiss, S., Matta, A., Stolletz, R.: 'Optimization of buffer allocations in flow lines with limited supply'*IISE Trans.*, 2018, **50**, (3), pp. 191–202.

11 Xi, S., Chen, Q., MacGregor Smith, J., Mao, N., Yu, A., Zhang, H.: 'A new method for solving buffer allocation problem in large unbalanced production lines'*Int. J. Prod. Res.*, 2020, **58**, (22), pp. 6846–6867.

12 Alon, G., Kroese, D.P., Raviv, T., Rubinstein, R.Y.: 'Application of the cross-entropy method to the buffer allocation problem in a simulation-based environment'*Ann. Oper. Res.*, 2005, **134**, (1), pp. 137–151.

13 Amiri, M., Mohtashami, A.: 'Buffer allocation in unreliable production lines based on design of experiments, simulation, and genetic algorithm'*Int. J. Adv. Manuf. Technol.*, 2012, **62**, (1–4), pp. 371–383.

14 Nahas, N.: 'Buffer allocation and preventive maintenance optimization in unreliable production lines'*J. Intell. Manuf.*, 2017, **28**, (1), pp. 85–93.

15 Zhou, B.H., Liu, Y.W., Yu, J. Di, Tao, D.: 'Optimization of buffer allocation in unreliable production lines based on availability evaluation'*Optim. Control Appl. Methods*, 2018, **39**, (1), pp. 204–219.

16 Yelkenci Kose, S., Kilincci, O.: 'A multi-objective hybrid evolutionary approach for buffer allocation in open serial production lines'*J. Intell. Manuf.*, 2020, **31**, (1), pp. 33–51.

17 Alaouchiche, Y., Ouazene, Y., Yalaoui, F.: 'Multi-Objective Optimization of Energy-Efficient Buffer Allocation Problem for Non-Homogeneous Unreliable Production Lines'*IEEE Access*, 2022, **10**, (1), pp. 3320–3335.

18 Arthur, W.: 'What is Evo-Devo and Why is it Important?'*Underst. Evo-Devo*, 2021, pp. 1–18.

19 Hickinbotham, S., Dubey, R., Friel, I., Colligan, A., Price, M., Tyrrell, A.: 'Evolving Design Modifiers'*2022 IEEE Symp. Ser. Comput. Intell.*, 2022, pp. 1052–1058.

20 Stanley, K.O., Miikkulainen, R.: 'Evolving Neural Networks through Augmenting Topologies'*Evol. Comput.*, 2002, **10**, (2), pp. 99–127.

21 DiFrisco, J., Jaeger, J.: 'Beyond networks: mechanism and process in evo-devo'*Biol. Philos.*, 2019, **34**, (6), p. 54.

22 Blank, J., Deb, K.: 'Pymoo: Multi-Objective Optimization in Python'*IEEE Access*, 2020, **8**, pp. 89497–89509.