An artificial hormone-based algorithm for production scheduling from the bottom-up

Wilfried Elmenreich[1], Alexander Schnabl[1] and Melanie Schranz[2]
[1]*Institute of Networked and Embedded Systems, Alpen-Adria-Universität Klagenfurt, Klagenfurt, Austria*
[2]*Lakeside Labs, Klagenfurt Austria*
wilfried.elmenreich@aau.at, alexander.schnabl@aau.at, schranz@lakeside-labs.at

Keywords:
Production Scheduling, Job-Shop Scheduling, Artificial Hormone Systems, Self-Organizing Systems, Swarm Systems

Abstract:
This paper presents a model for supporting a production scheduling system with an artificial hormone algorithm. The system consists of lots that have to undergo a number of processing steps on different machines. The processing steps for a lot are formalized in a recipe assigned to the lot type. Since the steps in the recipe have to be processed in order, the given system allows choice only in the context of selecting a particular machine for the next step and in changing the processing order of waiting lots at a machine. Optimization of such a job-shop scheduling system is an NP-hard problem. In the approach proposed by this paper, artificial hormone systems are used to express the urgency of a lot and the need for new lots at a machine type, thus providing a system using local information for optimization. Results indicate that the artificial hormone system provides an improvement of around 5% over a First Come-First Serve approach.

1 Introduction

The production of logic and power integrated circuits (ICs) in the semiconductor industry is a highly dynamic process (Geng, 2018). Unlike the high-volume production of memory ICs, in the logic and power sector, the wafer production has a huge product mix, dynamic changes in the system and a high number of processing steps and involved machines (Khatmi et al., 2019). Weekly workloads can involve around 10^5 operations on 10^3 machines (Teppan, 2018). Optimizing such a system for WIP (work in progress) and flow factor is an NP-hard problem (Garey et al., 1976), where existing dispatching rules and linear optimization methods cannot cope with the NP-hard search space (Lawler et al., 1993), and thus cannot consider the entire system behavior due to computational complexity. Therefore, they cannot exploit the optimization potential in job-shop scheduling in semiconductor production systems (Khatmi et al., 2019).

To overcome this issue, we model the production plant as a self-organizing system of agents that interact with each other in a non-linear way. Therefore, we aim at achieving near-optimal solutions in feasible computation time. Furthermore, such a system is also able to adapt to changes in the environment, scale with the number of agents, and is robust against a single point of failure, as the actions depend on local interactions (Heylighen, 2001). Moreover, local rules and local interactions allow overcoming the huge computing time of centrally performed linear optimization. One possibility for a self-organizing approach are algorithms modeled after biological hormone systems. Such artificial hormone systems are inspired by the biological endocrine system that is adjusting the metabolism of tissue cells in our body (Turing, 1952; Sobe et al., 2015). They are part of the class of self-organizing system with properties like scalability, adaptability, and robustness (Prehofer and Bettstetter, 2005) that can be used in networked technical applications to coordinate a set of complex agents interacting with each other (Elmenreich et al., 2009; Böszörmenyi et al., 2011; Sobe, 2012). Such algorithms are especially of interest for such large

cyber-physical systems, as, for example, in the semiconductor industry, where traditional control or scheduling mechanisms are at their limit.

In this paper, we describe an artificial hormone algorithm that can be applied to optimize the processing of lots and to balance the load between machines of the same type. Section 2 describes the target system problem statement that forms the basis to integrate an artificial hormone-based algorithm in such a system in Section 3. Section 4 evaluates the approach compared to a FIFO (first-in-first-out) principle and presents simulation results in Section 5. Finally, Section 6 presents the related work to this topic and Section 7 concludes the paper.

2 Problem Statement

A production plant has a number of machines, each being able to perform a specific process. Additionally, there can be multiple instances of a machine type. It is guaranteed that there exits at least one machine for each process step. Based on real semiconductor factories the processing time at a machine ranges between 20 minutes and 2 hours. Table 1 depicts the parameters of the system model. The notation $\mathcal{N}$ represents a Normal distributed random variable, while U represents a uniformly distributed random value. Random values are calculated once for each instance and remain unchanged for the duration of a simulation. The processing time for each machine type will be set to the outcome of a random variable $X \sim \mathcal{N}(1.16, 0.32)$ at the beginning of a simulation. This corresponds to a Normal distribution with 99% of values being between 0.33 and 2 hours. For a simulation, a virtual production plant comprising a random set of machines is generated. As it is common in the semiconductor industry, wafers are combined in groups of 25 pieces forming a so-called *lot*. Each lot is assigned a unique identification number and follows a respective recipe defining an ordered list of process steps the lot has to undergo at different machines. In our simulated problem set, lots are instantiated according to one out of 100 recipes. Based on the Raw Process Time (RPT) each individual lot gets a Planned Cycle Time (PCT) that is between 2 and 10 times higher than the RPT. Half of the machines employ batch processing, where lots are processed in parallel batches of $U(2, 8)$.

Table 1: Parameters used in the simulation model.

Parameter	**Value**
Process types	100
Machines per process type	$U(2, 10)$
Machines with batch processing	50
Batch size	$U(2, 8)$
Processing time per production step	$X \sim \mathcal{N}(1.16, 0.32)$
Number of recipes (lot types)	100
Recipe length	$U(90, 110)$
Lots per type	$U(2, 10)$
PCT	$U(2, 10)\times$ RPT
Total number of operations	$\approx 60\,000$

3 Artificial Hormone Algorithm

The artificial hormone algorithm is engineered in a bottom-up approach to express the urgency of a lot and the need for incoming lots by machines. Therefore, artificial hormones are produced at machines which can diffuse through the production system along with the processing steps of lots. The lots act as swarm members that can be attracted by the hormone level of a machine. The algorithm is based upon five parameters, which are described with their respective mechanisms in the following. Table 2 depicts an ad-hoc definition of parameter values, which will be used in first simulations. The algorithm describes the calculations and decision mechanisms that affect the processing of lots. There are several degrees of freedom for a particular implementation in the field. For example, the hormone-related computing could take either at the machines or within a layer of a networked monitoring and control system.

3.1 Hormone model

For each processing step, a corresponding hormone type exists. Hormones can be at any machine, also different hormone types can be at the same machine. Every simulation tick, hormones degrade exponentially at a given rate α.

$$hormone\_amount = hormone\_amount \cdot (1 - \alpha) \quad (1)$$

An evaporation rate of 0 means that hormones do not degrade. The maximum value is 1, where hormones degrade immediately.

3.2 Machines produce hormone to attract lots

Each machine generates a hormone of its process type. Machines that perform the same process type issue the same hormone. If a machine would be able to handle multiple process types, it would issue, in equal parts, the respective multiple hormones instead. The goal of machines is to maximize their working time. Therefore, a machine would aim for pulling a sufficient number of lots into its queue. The amount of hormone produced by one machine is therefore calculated by

$$hormone\_output = \frac{1}{lots\_in\_queue + \beta}, \quad (2)$$

where β is a smoothing factor > 0.

3.3 Machines are linked

A machine A is upstream-linked to a machine B if there exists a recipe that has a process of machine B and a process of machine A in subsequent steps. If each machine has exactly one process, the linking strength from this recipe is 1, otherwise, the link strength is 2 divided by the number of supported processes. If the same sequence appears in other recipes, the link strength adds up.

3.4 Hormone diffuses upstream

If upstream links exist, a part γ of the hormone at a given machine diffuses upstream:

$$upstream\_hormone = hormone\_amount \cdot \gamma \quad (3)$$

$$hormone\_amount = hormone\_amount - upstream\_hormone \quad (4)$$

For each machine connected upstream a proportional part of the upstream hormone is added:

$$added\_hormone = upstream\_hormone \frac{link\_strength}{\sum link\_strengths}, \quad (5)$$

where *link_strength* refers to the upstream link strength for the given hormone between the respective machines and $\sum link\_strengths$ refers to the sum of all upstream link strengths for the given hormone that emerge from the sending machine.

Table 2: Suggested algorithm parameters

Parameter	Value
α	.3
β	1
γ	.5
δ	.2
ε	.8

3.5 Incoming lots diffuse Hormone

In addition to the mechanism above, incoming lots cause a part δ of the respective hormone at a given machine to diffuse upstream.

$$upstream\_hormone = hormone\_amount \cdot \delta \quad (6)$$

$$hormone\_amount = hormone\_amount - upstream\_hormone \quad (7)$$

The machine where the lot came from receives the upstream hormone, this way a flow of lots can self-stabilize.

$$added\_hormone = upstream\_hormone \quad (8)$$

3.6 Lots are prioritized by their timing

A lot has a base priority calculated from its remaining RPT and its remaining PCT:

$$base\_priority = remaining\_RPT / remaining\_PCT \quad (9)$$

3.7 Lots are attracted by Hormone

The priority of a lot is modified by hormones that are present at the machine where the lot is currently waiting

$$attraction = \sum_{i=0} h_i \cdot \varepsilon^i, \quad (10)$$

where h_i is the hormone of the process that is i steps ahead in the lot's recipe. Therefore, h_0 is the hormone of the current process. ε is a factor indicating the strength of a hormone's influence.

The priority of a lot is then calculated with:

$$priority = base\_priority \cdot attraction \quad (11)$$

At a machine, lots are processed based on their priority. If a machine has batch processing, all lots that fit are accepted for one batch.

4 Evaluation

4.1 Evaluation Environment

The simulation was constructed in a NetLogo environment. NetLogo is an agent-based modeling software which includes the NetLogo programming language and an integrated development environment (Wilensky and Rand, 2015). An agent-based model is used to simulate the interactions of autonomous agents while also keeping track of the whole simulated system. Every autonomous agent makes decisions on its own based on predefined rules which later impact the outcome of the simulation. In the case of the problem statement, the agents are machines and lots. Before the simulation can start, the parameters have to be set. This can be achieved via the GUI in NetLogo or, for larger quantities, with configuration files. Then, the *Setup* function is called to initialize the model. It uses the specified parameters or parses the given configuration files to generate the agents. After the machines and lots have been placed in the production plant, the lots are introduced into production as shown in Figure 1. In it the machines are shown as squares, batch machines are being visualized with additional lines. The lots are numbered based on their product type and the most recent path they took is marked with a line.

4.2 NetLogo Model

After the initialization of all entities is finished, the production starts. A production step runs as follows: Every lot moves to the queue of the production type specified in its recipe. It only does so if it is currently not being processed by a machine. The algorithm then updates the hormones and sorts the queues based on the priority of the lots as described in Section 3. Listing 1 shows a simplified version of the step function and Listing 2 and 3 outline two exemplary hormone functions.

Listing 1: NetLogo process step

```
to step
    ask lots [
      if not processing? [ move-to-queue ]
    ]
    ask machines [
      produce-hormones self
      decay-hormones self
    ]
    sort-queues
    ask machines [
      pick-lots-from-queue
      do-processing
    ]
end
```

Listing 2: NetLogo hormone production function.

```
to produce-hormones [ m ]
    let q (item ([m.machine_type] of machine
        m - 1) queue_list)
    let q_len length [q.lotlist] of q
    ask machine m [ set m.hormone_amount
        m.hormone_amount + (1 / ( q_len +
        BETA)) ]
end
```

Listing 3: NetLogo hormone decay function.

```
to decay-hormones [ m ]
    ask machine m [ set m.hormone_amount
        m.hormone_amount * (1 - ALPHA) ]
end
```

After the calculations are done, the machines pick the lots with the highest priority from their queue to be processed. When a lot finishes its recipe it leaves the production plant and writes its *ProductionTime*, *RPT* and *PCT* to a file. This sequence is repeated until there are no more lots left in the production plant.

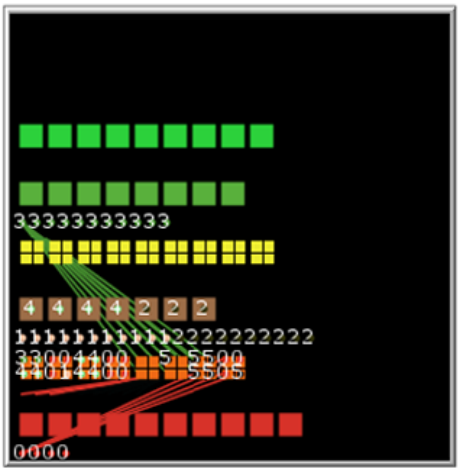

Figure 1: Screenshot from the Netlogo simulation: After initialization the lots start being processed and move to the queues specified in their recipes.

4.3 Headless Implementation

Using the technique specified above it is possible to gather large amounts of data by running multiple instances of the simulation simultaneously without a GUI in headless mode. The headless controller software was written in Python using the *NL4Py* package[1]. This package was chosen instead of other NetLogo controller software or the NetLogo BehaviourSpace because the algorithm has a very high time complexity and as soon as it gets to simulating larger quantities of machines, lots, etc. there is a significant improvement in performance. Through the *threading* module multiple simulations can be executed at the same time. The Python code was augmented with an automated configuration file generator. It creates new files for each simulation run using randomly chosen values within the limits specified in Table 3. Listing 4 shows an example workspace for multi-threading.

Listing 4: Python headless workspace.

```
def start_workspace(config_file,
    parameters):
  workspace = nl4py.
      newNetLogoHeadlessWorkspace()
  workspace.openModel(model)
  workspace.command("Setup")
  ...
  workspace.command("headless-go")

  while(processing == 0):
    time.sleep(20)
    processing = workspace.report("
        IS_FINISHED")
    ticks = str(workspace.report("
        ticks"))
    print("Ticks: " + ticks)
```

4.4 Parameter optimization

Another implementation of the algorithm was developed in Java so as to achieve a better way of finding optimal parameter values for the algorithm. This had various reasons such as performance and adaptability for future work. With the possibilities of object-oriented programming, the created simulation environment matching the problem statement is only based on 3 functions and it is now possible to simply exchange the algorithm of a production plant to compare different approaches with one another. The creation of new algorithms is also fairly easy and for data comparison, a simple first come, first served algorithm was also used.

[1]https://pypi.org/project/NL4Py/

Table 3: Evaluation parameters.

Parameter	Value
Initial RNG seed	58008
Number of simulation runs	100

5 Simulation results

The simulation results of the NetLogo model are evaluated based on average flow factor and tardiness of the lots. Table 3 details the used evaluation parameters.

$$flow\ factor = production\_time/RPT$$

$$tardiness = production\_time - RPT$$

The hormone algorithm was tested using the parameters outlined in Table 1, Table 2 and Table 3. Results are depicted in Table 4.

Table 4: Hormone algorithm average simulation results.

Average	Value
Time to produce all lots	173 h
RPT	59 h
PCT	348 h
Tardiness	114h
Flow Factor	2.93

A FCFS (first come, first served) algorithm was used to contrast the results. It ran with exactly the same test set and resulted in the values depicted in Table 5.

Table 5: FCFS algorithm average simulation results.

Average	Value
Time to produce all lots	184 h
RPT	60 h
PCT	331 h
Tardiness	124 h
Flow Factor	3.05

With the used parameters the hormone-based production approach seems to perform slightly better than an FCFS algorithm; however, considering the algorithm parameters in Table 2 are

ad-hoc values, there is a chance of improving the efficiency of the hormone algorithm by optimizing the parameters, which was done with the implementation in Java. While the NetLogo simulation ran on multiple randomly generated sample sets to gather a better general understanding of the speed of the algorithm, the Java version used the same configuration of machines, lots and products and just the parameters were altered. Table 6 and Table 7 show the fastest and the slowest simulation results and parameter values of the Java hormone algorithm. The algorithm with the optimized parameter set performs 9% better than the reference algorithm, however, it has to be noted that the optimization was done for a single test case. More importantly, while the two tables depict a potential for optimization of parameters, they also show that even with a bad parameter set the algorithm works slightly better than the FCFS algorithm. For reference, Table 8 shows the outcomes of the FCFS algorithm.

Table 6: Hormone algorithm parameters yielding the fastest result for the tested sample set.

Name	**Value**
α	0.2
β	3.0
γ	0.6
δ	0.3
ε	0.7
Time to produce all lots	152 h

Table 7: Hormone algorithm parameters yielding the slowest result for the tested sample set.

Name	**Value**
α	0.3
β	3.0
γ	0.5
δ	0.1
ε	1.0
Time to produce all lots	164 h

Table 8: FCFS Algorithm for the tested sample set

Name	**Value**
Time to produce all lots	165 h

6 Related Work

In a dynamic job shop, currently applied dispatching rules are based on heuristics with the disadvantage of not optimizing lot sequences. Although linear optimization methods are used to strengthen the scheduling procedure, they cannot cope with the highly complex, large, and dynamic search space (Lawler et al., 1993), mostly due to the excessive and unfeasible computation time in dynamic production plants. This leads to bottlenecks and work in progress (WIP) waves are generated. So far, no optimal solution for job shop scheduling has been developed using linear optimization that can be computed in polynomial time (Zhang et al., 2009).

In the course of this paper, we model the production plant as a self-organizing system of agents. Due to its non-linearity, it is able to produce near-optimal solutions for NP-hard problems in feasible computation times (Zhang et al., 2009; Dhiman and Kumar, 2017) by transforming the problem from finding an overall solution to defining a distributed algorithm that finds the solution from the bottom up. The related work on the application of self-organization in production scheduling builds upon the particle swarm optimization (Ghumare et al., 2015), and artificial bee colony algorithm (Zhang et al., 2013), and ant algorithm (Udomsakdigool and Kachitvichyanukul, 2008), to name but a few. Compared to these algorithmic approaches, we do not create a swarm that operates in a solution space to a given job-shop scheduling set. Instead, we derive a bottom-up approach, where embodied agents represent physical entities in the fab, and work with local rules from which a global behavior emerges. This presents a novel approach in the application of self-organizing principles in job-shop scheduling.

This approach can be implemented with an artificial hormone algorithm. It is a bio-inspired self-organizing algorithm to produce near optimal solutions in NP-hard, highly complex and dynamic systems based on relatively simple and local rules. The algorithm was introduced by Sobe et al. (Sobe et al., 2010) and describes the inspiration by the endocrine system of higher mammals on the technical application of sharing multimedia units. The main principle describes virtual hormones that show their interest in specific interest units with their concentration density. These hormones are created, consumed or evaporated, and forwarded by the agents in this artificial hormone system (Szkaliczki et al., 2013). Artificial hormone systems have been applied in technical systems such as task allocation (Renteln et al., 2008), synthesis of robot controller software (Hamann et al., 2010), or content delivery in dynamic networks (Schelfthout and Holvoet, 2003). Trumler et al. (Trumler et al., 2006) pro-

pose an artificial hormone system as middleware for ubiquitous computing, as for example in smart office applications. Brinkschulte et al. present an artificial hormone system for task allocation for heterogeneous processing units in a processor grid to realize self-X features (self-configuration, self-optimization, self-healing) (Brinkschulte et al., 2007; Brinkschulte et al., 2009). Sobe et al. (Sobe et al., 2015) build a middleware inspired by an artificial hormone system for search and delivery of information units. They showcase the applications of multimedia distribution at social events, and the information dissemination in smart electrical microgrids using tens of thousands of agents. The evaluation was also performed on a theoretical foundation to show that the hormone levels converge to a limit at each agent using a set of theorems on convergence conditions (Szkaliczki et al., 2016). Finally, Dong et al. (Dong et al., 2010) successfully proposed the artificial hormone algorithm for clustering in wireless sensor networks to prolong the lifetime of the sensor nodes. Further details on the origins of the artificial hormone algorithm, including the medical background of the endocrine system and its digitization, can be found in Xu and Wang (Xu and Wang, 2011).

7 Conclusion

The novel contribution of this paper is the application of an artificial hormone algorithm in the context of a job-shop scheduling system such as a large semiconductor fab from the bottom-up. The algorithm builds upon five principles, which are (i) machines produce hormone to attract lots, (ii) hormone diffuses process-upstream, (iii) incoming lots diffuse hormone, (iv) lots are prioritized by their timing, and (v) lots are attracted by hormone. Via these mechanisms, machines can balance their workload by pulling required lots towards them. The algorithm has been implemented and evaluated in a NetLogo simulation model. Simulation results indicate that the artificial hormone system provides an improvement of around 5% for overall production time and flow factor.

Acknowledgements

We would like to acknowledge Martina Umlauft for her comments on the paper and the NetLogo implementation. This work was performed in the course of project SWILT (Swarm Intelligence Layer to Control Autonomous Agents) supported by FFG – IKT der Zukunft under contract number 867530.

REFERENCES

Böszörmenyi, L., del Fabro, M., Kogler, M., Lux, M., Marques, O., and Sobe, A. (2011). Innovative Directions in Self-organized Distributed Multimedia Systems. *Multimedia Tools and Applications, Springer*, 51(2):525–553.

Brinkschulte, U., Pacher, M., and Von Renteln, A. (2007). Towards an artificial hormone system for self-organizing real-time task allocation. In *Proceedings of the IFIP International Workshop on Software Technolgies for Embedded and Ubiquitous Systems*, pages 339–347. Springer.

Brinkschulte, U., Pacher, M., and Von Renteln, A. (2009). An artificial hormone system for self-organizing real-time task allocation in organic middleware. In *Organic Computing*, pages 261–283. Springer.

Dhiman, G. and Kumar, V. (2017). Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications. *Advances in Engineering Software*, 114:48–70.

Dong, D., You, H., Zhang, Y., and Wang, X. (2010). A hormone-based clustering algorithm in wireless sensor networks. In *Proceedings of the 2nd International Conference on Computer Engineering and Technology*, volume 3, pages V3–555. IEEE.

Elmenreich, W., D'Souza, R., Bettstetter, C., and de Meer, H. (2009). A survey of models and design methods for self-organizing networked systems. In *Proceedings of the 4th International Workshop on Self-Organizing Systems*, volume LNCS 5918, page 37–49. Springer.

Garey, M. R., Johnson, D. S., and Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1(2):117–129.

Geng, H. (2018). *Semiconductor Manufacturing Handbook*. McGraw-Hill Education, 2 edition. ISBN 978-1-259-58769-6.

Ghumare, M., Bewoor, L., and Sapkal, S. (2015). Application of particle swarm optimization

for production scheduling. In *Proceedings of the International Conference on Computing Communication Control and Automation*, pages 485–489. IEEE.

Hamann, H., Stradner, J., Schmickl, T., and Crailsheim, K. (2010). Artificial Hormone Reaction Networks: Towards Higher Evolvability in Evolutionary Multi-Modular Robotics. In *Artificial Life XII (ALife XII)*, pages 773–780.

Heylighen, F. (2001). The science of self-organization and adaptivity. *The Encyclopedia of Life Support Systems*, 5(3):253–280.

Khatmi, E., Elmenreich, W., Wogatai, K., Schranz, M., Umlauft, M., Laure, W., and Wuttei, A. (2019). Swarm intelligence layer to control autonomous agents (SWILT). In *Proceedings of the Research Project Showcase at Software Technologies: Applications and Foundations (STAF-RPS19)*.

Lawler, E. L., Lenstra, J. K., Kan, A. H. R., and Shmoys, D. B. (1993). Sequencing and scheduling: Algorithms and complexity. *Handbooks in Operations Research and Management Science*, 4:445–522.

Prehofer, C. and Bettstetter, C. (2005). Self-organization in communication networks: principles and design paradigms. *IEEE Communications Magazine*, 43(7):78–85.

Renteln, A. V., Brinkschulte, U., and Weiss, M. (2008). Examinating Task Distribution by an Artificial Hormone System Based Middleware. *Proceedings of the 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing*, pages 119–123.

Schelfthout, K. and Holvoet, T. (2003). A pheromone-based coordination mechanism applied in P2P. In *Proceedings of the 2nd International Workshop on Agents and Peer-to-Peer Computing*, pages 1–12. Citeseer.

Sobe, A. (2012). *Self-organizing Multimedia Delivery*. Phd thesis, Alpen-Adria Universität Klagenfurt.

Sobe, A., Elmenreich, W., and Böszörmenyi, L. (2010). Towards a Self-organizing Replication Model for Non-sequential Media Access. In *Proceedings of the ACM MM Workshop on Social, Adaptive and Personalized Multimedia Interaction and Access*, pages 3–8.

Sobe, A., Elmenreich, W., Szkaliczki, T., and Böszörmenyi, L. (2015). SEAHORSE: Generalizing an artificial hormone system algorithm to a middleware for search and delivery of information units. *Computer Networks*, 80:124–142.

Szkaliczki, T., Sobe, A., and Elmenreich, W. (2016). Convergence and monotonicity of the hormone levels in a hormone-based content delivery system. *Central European Journal of Operations Research*, 24(4):939–964.

Szkaliczki, T., Sobe, A., Elmenreich, W., and Böszörmenyi, L. (2013). Analysis of an artificial hormone system. In *Proceedings of the 8th Japanese-Hungarian Symposium. on Discrete Mathematics and Its Applications*, Veszprém, Hungary.

Teppan, E. C. (2018). Dispatching rules revisited – a large scale job shop scheduling experiment. In *Proceedings of the IEEE Symposium Series on Computational Intelligence*, pages 561–568. IEEE.

Trumler, W., Thiemann, T., and Ungerer, T. (2006). An artificial hormone system for self-organization of networked nodes. *Biologically Inspired Cooperative Computing*, pages 85–94.

Turing, A. M. (1952). The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 237(641):37–72.

Udomsakdigool, A. and Kachitvichyanukul, V. (2008). Multiple colony ant algorithm for job-shop scheduling problem. *International Journal of Production Research*, 46(15):4155–4175.

Wilensky, U. and Rand, W. (2015). *An Introduction to Agent-Based Modeling: Modeling Natural, Social, and Engineered Complex Systems with NetLogo*. MIT Press.

Xu, Q.-z. and Wang, L. (2011). Recent advances in the artificial endocrine system. *Journal of Zhejiang University Science C*, 12(3):171–183.

Zhang, G., Shao, X., Li, P., and Gao, L. (2009). An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computers & Industrial Engineering*, 56(4):1309–1318.

Zhang, R., Song, S., and Wu, C. (2013). A hybrid artificial bee colony algorithm for the job shop scheduling problem. *International Journal of Production Economics*, 141(1):167–178.