




Bottom-up Job Shop Scheduling with Swarm Intelligence in Large Production Plants

M. Schranz¹ ^a, M. Umlauf¹ ^b and W. Elmenreich² ^c

¹Lakeside Labs GmbH, Klagenfurt, Austria

²Institute of Networked and Embedded Systems, University of Klagenfurt, Klagenfurt, Austria
{schranz, umlauf}@lakeside-labs.com, wilfried.elmenreich@aau.at

Keywords:

Swarm Intelligence, Multi-Agent Modeling, Cyber-Physical Systems, Job Shop Scheduling

Abstract:

In production plants organized by the job shop principle, the factory-wide scheduling problem is NP-hard and can become extremely large. Traditional optimization methods like linear optimization reach their limits in these settings due to excessive computation time. Therefore, we propose this industrial setting as a novel field of application for swarm intelligence using bottom-up algorithms that do not require the infeasible calculation of an overall solution but depend only on local information. We consider the example of the semiconductor industry producing logic and power integrated circuits where a diverse range of highly specialized but low volume products are fabricated in the same plant. This paper shows how to select and model swarm members, swarms, and their interactions for use in real-world production plants. There are multiple possibilities for the modeling of the agents: a swarm member could be a single machine or a set of machines (workcenter), a product or group of products of the same/similar type, or a more abstract agent like a process. In particular, we consider criteria for selecting appropriate swarm members and potential candidate swarm algorithms inspired by hormones and ants.


1 Introduction


Scheduling in production plants organized by the job shop principle is a challenging problem in Industry 4.0. The main issues are the given constraints (e.g., from machines) together with the global objective in production plants (e.g., maximizing of machine utilization, throughput time, delivery reliability, or minimizing the work in progress (WIP)). High product diversity together with historical growth of the industrial plant increase complexity even further.


A typical example for such a highly dynamic process is the production of integrated circuits (ICs) in the semiconductor industry (Geng, 2018). In particular, we consider the so-called front end of line processing where the wafers are processed to create the ICs. For this paper, we consider the requirements of the leading semiconductor manufacturer Infineon Technologies¹. Between 400 and 1200 different stations need to be scheduled in such a production plant (fab). Typical process steps include lithography, doping, oxidation, etching, and measuring (Geng,

2018). They typically produce more than 1500 different products in around 300 different process steps. These processes are characterized by a multitude of technological and logistical boundary conditions, like, e.g., equipment tooling, the need for secondary resources, and constraints like time couplings and batch processing. Furthermore, the chain of necessary process steps contains loops. The job shop scheduling problem is NP-hard (Garey et al., 1976). Existing dispatching rules are based on heuristics. Due to excessive computation time, Linear optimization methods can not cope with the large, and dynamic search space (Lawler et al., 1993). These methods can only be used on subsets of the plant, and thus, do not exploit the full optimization potential. Therefore, bottlenecks cannot be prevented, and 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).

Other than using swarm intelligence as an optimization algorithm (Ghumare et al., 2015), we apply a novel approach modeling the production plant as a self-organizing system of agents that work together, each equipped with a set of local rules. A self-organizing system is highly non-linear because of feedback relations. Thus,

^a  <https://orcid.org/0000-0002-0714-6569>

^b  <https://orcid.org/0000-0000-0000-0000>

^c  <https://orcid.org/0000-0001-6401-2658>

¹<https://www.infineon.com>

it can adapt to changing environmental conditions (Heylighen, 2001), e.g., tool downs. We consider nature-inspired swarm algorithms, since many of them are able to produce near-optimal solutions for NP-hard problems in feasible computation time. Thus, the problem is transformed from finding an overall solution to defining a distributed algorithm that finds the solution from the bottom up. The considered production plant requirements match the advantages of swarm intelligence: adaptivity, robustness, and scalability. In this paper we analyze the problem setting in such a highly complex production plant (Sec. 2), introduce how swarm modeling could be used to address the problem (Sec. 4), show two examples of swarm algorithms and their application in this setting (Sec. 5), and discuss our approach in comparison to the related work in the field (Sec. 6).

2 Semiconductor Fab Setting

Semiconductor production uses an industry standard of 25 wafers that are combined to form a **lot** and then transported in so-called FOUPs² or hordes between the individual process steps. Every lot has a transponder with a unique identification number used to track a lot through the fab. Lots follow a specific recipe, which prescribes which process steps to take in which order but not the concrete machine on which to perform the required next process step. **Machines** can be single wafer tools which work wafer by wafer, or batch-oriented and process several lots at once. Machines are often spread across several factory buildings. Nevertheless, transport times between machines are negligible while workload and waiting times before machines are the determining factor. A set of machines which can run one or more similar process classes form a **workcenter**. These sets of machines are exhaustive and disjoint. A dedication matrix specifies which processes are allowed on which machines. Load in a workcenter is distributed via a solver which calculates scheduling and local optimization for the workcenter for a week in advance.

In the considered production plant about 10% of the machines have a low utilization (e.g., 60% or even 30%), because they are only used for some special products. There is a difference between utilization of a workcenter and the utilization of a machine. For example, for a specific machine a utilization of 80% is very high, while for a workcenter 80% would be quite low.

²Front Opening Unified Pod

3 Problem Statement

Currently, optimization per workcenter works well, but its influence on the following workcenters and machines is unclear and does not automatically generate a global optimum. It is not understood how local optimization affects the overall efficiency of the fab or whether optimizing one workcenter will create bottlenecks in others. Existing dispatching rules are based on heuristics. Since job shop scheduling is an NP-hard problem and the number of involved machines and lots is considerably large, optimization can only be calculated for a subset of the machines in the fab (e.g., within a workcenter). Bottlenecks arise dynamically due to a wide variety of reasons, e.g., tool downs, or changed product mix. Downtimes always cause capacity loss. Bottlenecks cannot be prevented, and generate WIP waves.

Different machine types have different logistic requirements (batch vs. single processing). Especially after a batch process, lot arrival times vary considerably (because lots leave the process in batches) while theory suggests that for single-step processing machines high utilization is only possible when arrival times are uniform (Stidham Jr, 2002). The main challenge is that this switch between batch and single-step processing introduces so-called WIP waves between machines.

Lots that traverse a loop of machines several times can add to the challenge: how to prioritize a lot against other lots of the same product which are one or several cycles ahead? If only the lots with the nearest due dates are processed, the lots with later due dates might violate their time coupling requirements. If too many of the same product with later due dates are prioritized, the products with the nearest due dates might violate theirs. In the current setting, lots of the same product are spread over the week when introduced into the fab to avoid WIP waves.

4 Swarm Modeling of the Problem

To model the production plant as a swarm, we consider a set of potential swarm members and their characteristics. In this consideration, the swarm can be homogeneous (agents of the same type, e.g., lots), or heterogeneous (agents of different types, e.g., lots and machines). Furthermore, we identify several criteria to determine whether an entity is eligible to be a swarm member in the fab according to the related work (see Sec. 6 for more details). A swarm member should be able to work in a swarm per se, thus, a reasonable number of other swarm members should

exist (e.g., the lithography workcenter only exists once in the fab and on its own would not represent a good swarm member), show a suitable level of abstraction to be modeled, detect dynamic information from the (local) environment, react to information from the local neighborhood (e.g., take decisions), and be plausible and understandable to enable trust in the proposed solution.

4.1 Challenges

When considering the introduction of a swarm-based algorithm in a production plant, the following challenges arise:

What should be modeled as agents and what is the right level of abstraction? A swarm consists of agents that interact with each other. In our case, a swarm member could be a machine, a lot or a process representing an abstract entity. Modelling a swarm algorithm requires an early decision on what entity should be used as a swarm member. As described in the beginning of Sec. 4 we consider several criteria. Nevertheless, the level of abstraction is not always clear. For example, if we consider machines as agents, we could also map a higher abstraction level and consider workcenters as agents, or we map an abstraction level lower and use process steps as agents.

How to deal with inhomogeneities among entities? Intrinsically, machines are heterogeneous by themselves. There is no unified type of a specific machine, and even on nearly identical machines there is the possibility of defining different processes. Thus, each machine has a set of different parameters that influence its operation. The same holds for lots, as they have different recipes and thus, different requirements in their route through the fab.

How to implement the necessary swarm communication paradigms in the setting of the production plant? Swarm algorithms typically implement one of two communication paradigms—direct communication where agents send messages or indirect communication (stigmergy), where agents leave information in the environment. The entities of the fab are already represented as digital twins in the central fab computer system. Therefore, agent communication can easily be implemented as local messages within the computer system or in local memory for stigmergy. As a result, it is *not* necessary to equip machines or lots with additional computational intelligence or communication devices.

How to implement a solution on top of a working environment? Current fab mechanisms include a

number of different priority classes (e.g., hot lots, developer lots). Generally, lots with priority class “hot” will sometimes be single processed due to time constraints, and developer lots will be single processed because new recipes/processes are tried out on them. Nearly 20% of lots are developer lots. They are used for research and development, machine control, and maintenance. Thus, it might not be possible to combine them with others into a swarm. The majority of these priority classes is historically driven. To model the lots as swarm members, the priority class is an important parameter. Nevertheless, by the introduction of swarm-based local rules and interactions, the priority classes could be reduced to a single one: the due date. This would free the system from reserving time for hot lots, but the question is if the treatment of hot lots is still satisfactory under such an approach. Another important consideration are time-coupled lots: if these can not be processed by the next machine within a certain time frame, the lots would be damaged and need to be discarded. To take this into account, the urgency for time-coupled lots needs to be additionally considered.

How to validate the approach? Testing and predicting the performance of a given algorithm is difficult. Although historical data exists for each lot (e.g., the logical transactions from machines including move-in and move-out operations), it is hard to get WIP and dedication matrix data that would allow predictions for new rules. Furthermore, the performance of the manufacturing process is also dependent on external factors like machine downs, operator sickness, or even weather conditions like lightning strikes.

4.2 Swarm Member Candidates

The model for the problem consists of a production plant with machines, queues, processes, lots, and recipes. The production plant \mathfrak{P} contains several sets or workcenters of machines $W^m = \{M_1^m, M_2^m, \dots\}$, where m is the machine type. Each machine M_i^m has a queue Q_i^m . Every machine in a machine set/workcenter W^m can perform a process P^m .

A set of lots $L = \{l_1^t, l_2^t, \dots\}$ need to be processed in the plant, where t is the product type. Each product type t is defined by a recipe R^t which prescribes the sequence of processing steps necessary to manufacture this product. The lot l_n^t can choose which of the suitable machines M_i^m to use for each necessary process step P^m .

Therefore, the recipes imply a directed graph

$G = (V, E)$ of possible movements between the machines of the plant, where the nodes V consist of all machines M_i^m and the edges E are defined between two machines M_i^m and M_j^p if there exists a lot l_n^t with a recipe R^t containing processes P^m and P^p in direct succession. A route \mathfrak{R} is an ordered list of machines which can execute the corresponding processes in the recipe. The taken routes are a sub-graph of G with $G_\tau \subseteq G$.

Out of this formal definition, we identified a number of possible components in a fab that can act as swarm members.

Machines M know their processes and their utilization. There are many different machines in the fab that could form either multiple cooperating homogeneous swarms, or a heterogeneous swarm with different capabilities. The neighborhood is locally and dynamically defined by the recipes of the incoming and outgoing lots. Machines can take decisions locally and can, e.g., re-order their queue and thereby select which lot to process next. Furthermore, they can communicate with other machines, and could ask for lots of a specific processing type.

Workcenters W , with $W \subset M$ have attributes very similar to single machines, because a workcenter is simply a set of related machines. It can calculate when lots will be processed internally and can use makespan information (the time that elapses from the start to the end of a lot's production) to calculate when the lots currently being processed will be finished.

Lots L can also form either homogeneous (lots of the same product type), or heterogeneous (lots of multiple product types) swarms. Lots follow a specific recipe, which prescribes which process steps to take in which order but not the concrete machine on which to perform the required next process step. As there are typically multiple machines which can perform the same process, lots could decide which machine to take next. The concrete path taken through the machines of the fab is called a route. Typically, lots of similar product types will share parts of their recipes. Given a stable load situation in the machine park, lots of the same product type and lots of similar product types can therefore be expected to share parts of their routes. Furthermore, lots could manipulate their own prioritization.

Processes P could be virtual swarm members. They see all machines that they can potentially be run on plus the current and total workload. Additionally, they could forecast the workload, the times for re-tooling, and have information on batching requirements.

5 Candidate Algorithms

In the following we present two candidate algorithms for the job shop problem using bottom-up approaches. They have been selected because they present an embodied approach, thus, each agent in the algorithm can be modeled from a "real" entity in the production plant. These swarm members are already represented as digital twins. For the swarm approach we can add supporting attributes (e.g., pheromones), and observe the virtual, but local environment of the selected swarm members. Thus, instead of a global computation of the overall fab, the digital twins can work and interact with local information.

5.1 Hormone Algorithm

In a semiconductor fab, a network for propagating an artificial hormone can be constructed by the planned processing steps (recipes) of lots. An artificial hormone system can help to express the urgency of a lot and the need for new lots at a machine type. Artificial hormone produced at machines can diffuse through the production system via the lot recipes. Lots act as swarm members that are attracted by hormone. The approach is inspired by the usage of artificial hormones to reorganize agents (Sobe et al., 2015) in a self-organizing systems for technical applications (Elmenreich and de Meer, 2008; Elmenreich et al., 2009). For implementation, the artificial hormone system is realized as a software layer spread over the processing queues of all machines in a fab. Machines are assumed to have artificial glands that can produce hormone. The hormone algorithm for optimizing lot processing in a semiconductor fab can be decomposed into six mechanisms:

Production: Machines M_i^m produce hormone h_i^m according to the number of lots in their processing queue. Machine that are about to run out of lots produce more hormone. Each machine set/workcenter W^m has its own type of hormone.

$$H^m = \frac{1}{|Q_i^m| + \beta}, \quad (1)$$

where H_i^m is the hormone corresponding to workcenter W^m at machine M_i^m , β is a smoothing factor and $|Q_i^m|$ is the number of waiting lots in the queue for machine M_i^m .

Evaporation: Hormone levels at an agent evaporate exponentially with a given rate α :

$$H_{i,t+1}^m = H_{i,t}^m \cdot (1 - \alpha) \quad (2)$$

where $H_{i,t+1}^m$ and $H_{i,t}^m$, represent the state of hormone at machine M_i^m before and after a discrete evaluation step.

Diffusion: Hormone diffuses from machine to machine based on lot's recipes that connect the machines. Hormone diffuses upstream, i.e. from a later machine in a recipe to a machine that comes before that machine in the recipe. Thus, hormone diffusion follows the transpose graph of G , G^T . If there is an edge in G^T between process P^m and a process P^p (P^p being the predecessor of P^m in a recipe R_t), the amount of hormone moving upstream from machine M_i^m is defined by

$$\Delta H = H_i^m \cdot \gamma \quad (3)$$

$$H_i^m - = \Delta H \quad (4)$$

where γ is a parameter setting the motility of hormone. The link strength $l^{m,p}$ between two machines M_i^m and M_j^p is equivalent to the number of recipes R_t containing processes P^m and P^p in direct succession. Each machine connected upstream receives a proportional part of the upstream hormone:

$$H_j^p + = \Delta H \frac{l^{m,p}}{\sum_r l^{m,r}}, \quad (5)$$

where $\sum_r l^{m,r}$ represents the sum of all upstream links from P^m .

Diffusion through lot movement: Incoming lots following an edge of G generate a flow of hormone back to where the lots came from via the corresponding backwards edge in the transpose graph of G .

$$\Delta H = H_i^m \cdot \delta \quad (6)$$

$$H_i^m - = \delta H \quad (7)$$

$$H_j^p + = \delta H \quad (8)$$

where ΔH defines the amount of hormone that moves with the lot, calculated from the amount of available hormone H_i^m at a machine M_i^m . The lot moved from machine M_i^m to M_j^p , thus adding to H_j^p . M_j^p can be also a machine from a different workcenter than W^m .

Attraction: Lots are attracted by hormone, if those match the machines types required for their next steps in their recipe. The amount of attraction decreases exponentially based on the number of steps ahead. The attraction force is applied to lots waiting in a processing queue Q_i^m and can make an attracted lot move forward in the queue.

$$attraction = \sum_{i,m} H_i^m \cdot \varepsilon^n, \quad (9)$$

where H_i^m is the hormone amount at a machine that is n edges away and ε is a factor < 1 defining the degradation of the hormone attraction over edge distance in graph G . Attraction is used to reorder waiting lots, but other factors such as remaining raw process time and remaining process cycle time also influence lot urgency. The diffusion by lots is a self-reinforcing process that is balanced by hormone production based on the shortest processing queues. In this way, a path through the system where lots are processed quickly—the paths we would like to have in a productive system—is marked with hormone. Each mechanism comes with a parameter indicating the strength of each part, that is evaporation rate α , hormone production factor β , upstream diffusion factor γ , hormone distribution factor δ and attraction factor ε . A possible configuration of these parameters is stated in (Elmenreich et al., 2021). Due to the interaction between each of the mechanisms forming feedback control loops, the algorithm can operate with a broad set of possible parameter settings.

5.2 Ant Algorithm

Ant algorithms are inspired by the foraging behavior of ants which can find near-optimal paths to food sources without global knowledge by marking trails with pheromones. They have been successfully applied to route optimization of the traveling salesman problem (Caro and Dorigo, 1998; Di Caro et al., 2005). As shown in Sec. 4.2 the plant can be seen as a graph $G = (V, E)$ and lot progression through the fab as a routing problem. Lots are mapped to ants and machines are mapped to network nodes. The approach can then be decomposed as follows:

Trail following: Lots choose the next machine M_i^m probabilistically out of the set of possible machines W^m based on the local pheromone values for that machine and a local heuristic of its current load (ie. the queue length of the machine) as shown in Equation 10.

$$P_{i,j} = \frac{\tau_{i,j,d} + \alpha \eta_{i,j}}{1 + \alpha(N_i - 1)} \quad (10)$$

with

$$\eta = 1 - \frac{q_{i,j}}{\sum q} \quad (11)$$

the relative queue length of the machine and N_i the number of possible machines. Instead of a “destination node” as in the original AntNet variant of the algorithm, we use the next following step of the recipe for the destination d . Thus, a

machine with a high amount of pheromone and a short queue length becomes more attractive for a lot. The influence of pheromone vs. local heuristic can be adjusted via a parameter.

Trail laying: Pheromone values are updated after a lot has progressed. In distributed ant routing algorithms for packet networks, this is done by sending a backward ant to travel the same route in reverse after the original ant has reached its final destination. Backward ants update the pheromone values at each hop according to the time it took the original ant to traverse the respective link. In our approach, pheromone updates are triggered *not* by sending a “reverse lot” but rather by communication between machines. To facilitate the pheromone update, each lot keeps a memory of the time it took to traverse an edge—i.o.w. how long it took to wait in the machine’s queue. Since complete production of a lot takes weeks, backward communication and pheromone update is performed as soon as a lot has traversed a workcenter. For a chosen machine M_x^m the pheromone value is updated as follows:

$$\tau_{x,d} \leftarrow \tau_{x,d} + r(1 - \tau_{x,d}) \quad (12)$$

while for all potential machines M_n^m which were *not* chosen, the pheromone values are updated according to Equation 13

$$\tau_{n,d} \leftarrow \tau_{n,d} - r\tau_{n,d} \quad (13)$$

with reinforcement r depending on the time the lot took to traverse this “hop” and the current congestion status according to the local heuristic:

$$r \equiv r(T, \mathcal{M}_i^m) \quad (14)$$

Evaporation: At regular time intervals, pheromone values are evaporated with a rate p so that trails which have gotten worse (e.g., due to machine downs) are removed (Equation 15).

$$\tau(t+1) = \tau(t)(1-p) \quad (15)$$

6 Related Work

Production scheduling aims at arranging and controlling work and workloads in a production process so that a given optimality criterion, such as minimizing makespan, mean flow time, idle machine time, or total tardiness is addressed (Blazewicz et al., 2019). It is an NP-hard problem, thus belonging to the class of hardest problems according to computational complexity theory (Garey et al., 1976). To address large problem sizes, various techniques are suggested

including differential evolution algorithms (Yuan and Xu, 2013), genetic algorithms and tabu search (Li and Gao, 2016), large-neighborhood search (Pacino and Van Hentenryck, 2011), or a defining scheduler upon general purpose declarative problem solvers (Da Col and Teppan, 2016). Although several approaches already exist, they typically make the calculations centrally and beforehand (similar to existing linear optimization approaches).

In swarm intelligence complex optimizations problems are solved by using local rules inspired by swarms of birds, fish, and ants (Almufti et al., 2019; Hassanien and Emary, 2018). When modeling a swarm, the choice of agents in the system needs to be defined carefully. The two main aspects comprise the agent’s properties (internal and external states) and the agent’s behavior. In our example, the internal or external states would describe, e.g., the agent’s current and next process step, process time, or utilization. Through its behavior, the agent can change states from the environment, other agents, or the internal state (Schranz et al., 2020). Furthermore, we can differentiate between three types of agents: mobile, stationary or connecting agents (Wilensky and Rand, 2015). How to choose an agent for a swarm is application-specific. Nevertheless, (van Ast et al., 2008) present a swarm framework to formally define dynamic agents, their neighborhood, environment, and interactions. (Schranz et al., 2018) model a swarm member by focusing on cyber-physical systems and their local memory (e.g., the current x and y position, available energy), behaviour, physical aspects, security, and human interaction interfaces using the modeling tools Modelio and FREVO (Sobe et al., 2012) to evolve a swarm algorithm solving it. Swarm algorithms are of high interest for the domain of job shop scheduling as it is well-known that optimal solutions cannot be found in feasible time with linear optimization (Zhang et al., 2009). Nature-inspired algorithms have been confirmed to be excellent in addressing complex optimization problems in job shop scheduling (Gao et al., 2019). The majority of related work on the application of swarm concepts in production scheduling build upon particle swarm optimization (Ghumare et al., 2015). Additionally, several contributions are using the artificial bee colony algorithm (Sharma and Pant, 2017). Digital hormone models for self-organization have been suggested in (Shen et al., 2002) and have been applied to task allocation problems (Renteln et al., 2008; Brinkschulte et al., 2007), synthesis

of robot controllers (Hamann et al., 2010; Wilson et al., 2019), or large-scale IoT systems (Sinha and Chaczko, 2017). Artificial hormone systems have also been proposed as the basic mechanism of middleware for distributed systems (Sobe et al., 2015). Ant algorithms have originally been proposed in (Dorigo et al., 1996) and (Dorigo and Gambardella, 1997). They are a bio-inspired meta-heuristic based on the foraging behavior of ants (e.g., the Black Garden ant *Lasius niger* or the Argentine ant *Iridomyrmex humilis*) and have been applied successfully to the NP-hard problem of finding the shortest path in the traveling salesman problem (Lawler et al., 1985). A thorough introduction to ant algorithms can be found in (Dorigo and Stützle, 2004) and in (Bonabeau et al., 1999); there exist currently over a hundred variants of these algorithms. Ant algorithms have also been investigated for use as a routing protocol in packet routing networks in, e.g., (Caro and Dorigo, 1998) or (Di Caro et al., 2005) where they are implemented as distributed algorithms using only local information at each network node. They have been shown to converge quickly with reasonable overhead. Compared to the related work, we do not create a swarm that operates in a solution space of possible schedules as solutions to a given job-shop scheduling set, but instead derive a bottom-up approach, where embodied agents represent physical entities in the fab (instead of representing a single solution in the solution space), and work with local rules from which a global behavior emerges.

7 Conclusion

Job shop scheduling is an NP-hard problem which makes it impossible to find an optimal solution of a large scale system in reasonable time. While previous approaches often aimed at reducing the problem size by optimizing subsets of the system, we aim for a bottom-up approach that uses swarm-based modeling to establish local interaction rules between related parts of the system. The proposed approach is sketched via two different algorithms, one inspired by the natural endocrine system and the other inspired by eusocial insects. We depicted a mapping between machines and lots in the job shop system to agents of a swarm system.

As a next step we will elaborate on the efficiency of the proposed candidate algorithms using a simulation approach based on an abstracted version of a wafer fab. An open question remains if an improvement is measurable at workcenter level as approximate scheduling solvers already

exist for several workcenters. This brings up another question that needs to be answered: how to integrate with and where to set the boundary between currently used solvers and the swarm approach in the running system? One possibility would be that the swarm algorithms provide local information to the solver which takes this information into account during its calculation.

Acknowledgement

This work was performed in the course of project SWILT³ (Swarm Intelligence Layer to Control Autonomous Agents) supported by FFG under contract number 867530.

REFERENCES

- Almufti, S., Marqas, R., and Ashqi, V. (2019). Taxonomy of bio-inspired optimization algorithms. *Journal of Advanced Computer Science & Technology*, 8(2):23–31.
- Blazewicz, J., Ecker, K., Pesch, E., Schmidt, G., and Weglarz, J. (2019). *Handbook on Scheduling*. Springer.
- Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm Intelligence – From Natural to Artificial Systems*. Oxford University Press.
- Brinkschulte, U., Pacher, M., and Von Renteln, A. (2007). Towards an artificial hormone system for self-organizing real-time task allocation. *Software Technologies for Embedded and Ubiquitous Systems*, Springer, pages 339–347.
- Caro, G. D. and Dorigo, M. (1998). Antnet: Distributed stigmergy control for communications networks. *Artificial Intelligence Research*, 9:317–365.
- Da Col, G. and Teppan, E. C. (2016). Declarative decomposition and dispatching for large-scale job-shop scheduling. In *Proceedings of the Joint German/Austrian Conference on Artificial Intelligence*, pages 134–140. Springer.
- Di Caro, G., Ducatelle, F., and Gambardella, L. M. (2005). Anthocnet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks. *European Transactions on Telecommunications*, 16(5):443–455.
- Dorigo, M. and Gambardella, L. M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Trans. on Evolutionary Computation*, 1(1):53–66.
- Dorigo, M., Maniezzo, V., and Colomi, A. (1996). Ant system: Optimization by a colony of cooperating agents. *IEEE Trans. on Systems, Man, and Cybernetics, Part B*, 26(1):29–41.
- Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimization*. A Bradford Book, The MIT Press.

³<https://swilt.aau.at/>

- Elmenreich, W. and de Meer, H. (2008). Self-organizing networked systems for technical applications: A discussion on open issues. In *Proceedings of the 3rd International Workshop on Self-Organizing Systems*, pages 1–9. Springer.
- 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 *Self-Organizing Systems*, volume 5918 of *LNCS*, pages 37–49. Springer.
- Elmenreich, W., Schnabl, A., and Schranz, M. (2021). An artificial hormone-based algorithm for production scheduling from the bottom-up. In *Proceedings of the 13th International Conference on Agents and Artificial Intelligence*. SciTePress.
- Gao, K., Cao, Z., Zhang, L., Chen, Z., Han, Y., and Pan, Q. (2019). A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems. *IEEE/CAA Journal of Automatica Sinica*, 6(4):904–916.
- 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., editor (2018). *Semiconductor Manufacturing Handbook*. McGraw-Hill Education.
- Ghumare, M., Bewoor, L., and Sapkal, S. (2015). Application of particle swarm optimization for production scheduling. In *Proc. 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 *Proceedings of the Artificial Life XII*, pages 773–780.
- Hassani, A. E. and Emary, E. (2018). *Swarm Intelligence: Principles, Advances, and Applications*. CRC Press.
- Heylighen, F. (2001). The science of self-organization and adaptivity. *The Encyclopedia of Life Support Systems*, 5(3):253–280.
- 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.
- Lawler, E. L., Lenstra, J. K., Rinnooy-Kan, A. H. G., and Shmoys, D. B., editors (1985). *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. JohnWiley & Sons.
- Li, X. and Gao, L. (2016). An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. *International Journal of Production Economics*, 174:93–110.
- Pacino, D. and Van Hentenryck, P. (2011). Large neighborhood search and adaptive randomized decompositions for flexible jobshop scheduling. In *Proc. of the 22nd International Joint Conference on Artificial Intelligence*, pages 1997–2002.
- Renteln, A. V., Brinkschulte, U., and Weiss, M. (2008). Examining task distribution by an artificial hormone system based middleware. In *Proc. of the 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing*, pages 119–123.
- Schranz, M., Bagnato, A., Brosse, E., and Elmenreich, W. (2018). Modelling a CPS swarm system: A simple case study. In *Proc. of the 6th International Conference on Model-Driven Engineering and Software Development*, pages 615–624.
- Schranz, M., Umlauf, M., Sende, M., and Elmenreich, W. (2020). Swarm robotic behaviors and current applications. *Frontiers in Robotics and AI*, 7:36.
- Sharma, T. K. and Pant, M. (2017). Shuffled artificial bee colony algorithm. *Soft Computing*, 21(20):6085–6104.
- Shen, W. M., Chuong, C. M., and Will, P. (2002). Digital hormone models for self-organization. In *Proceedings of the 8th International Conference on Artificial Life*, pages 116–120.
- Sinha, S. and Chaczko, Z. (2017). Concepts and observations in artificial endocrine systems for iot infrastructure. In *Proceedings of the 25th International Conference on Systems Engineering*, pages 427–430.
- Sobe, A., Elmenreich, W., Szkaliczki, T., and Böszörményi, L. (2015). SEAHORSE: Generalizing an artificial hormone system algorithm to a middleware for search and delivery of information units. *Computer Networks*, 80:124–142.
- Sobe, A., Fehérvári, I., and Elmenreich, W. (2012). FREVO: A tool for evolving and evaluating self-organizing systems. In *Proceedings of the 1st International Workshop on Evaluation for Self-Adaptive and Self-Organizing Systems*.
- Stidham Jr, S. (2002). Analysis, design, and control of queueing systems. *Operations Research*, 50(1):197–216.
- van Ast, J., Babuska, R., and De Schutter, B. (2008). A general modeling framework for swarms. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 3795–3800.
- Wilensky, U. and Rand, W. (2015). *An Introduction to Agent-Based Modeling: Modeling Natural, Social, and Engineered Complex Systems with NetLogo*. MIT Press.
- Wilson, J., Timmis, J., and Tyrrell, A. (2019). An amalgamation of hormone inspired arbitration systems for application in robot swarms. *Applied Sciences*, 9(17).
- Yuan, Y. and Xu, H. (2013). Flexible job shop scheduling using hybrid differential evolution algorithms. *Computers & Industrial Engineering*, 65(2):246–260.
- 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.