40.014: Engineering Systems Architecture

# Operations Scheduling

Project Report 1

**ESD CH01 Group 5 Project 6**

|  |  |
|---|---|
| Oon Eu Kuan Eugene | 1006637 |
| Kong Le'ann Norah | 1007000 |
| Georgia Karen Lau | 1007153 |
| Tan Chong Hao | 1006915 |
| Long Yan Ting | 1006944 |
| Lee Peck Yeok | 1006968 |
| Nathan Ansel | 1007492 |

**Course Instructors:**

Rakesh Nagi

Sun Zeyu

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

## Contents

## 1 Define the Problem

Many organisations face the challenge of meeting customer expectations while reducing production costs in a complex and varied environment. Our challenge lies in optimising assembly line scheduling within manufacturing operations to determine an efficient task sequence, allocate resources accurately, and adhere to operational constraints. Since the 1960s, the focus has shifted from single-criteria optimisation to considering multiple conflicting objectives. The multi-objective problem that we will be solving is minimising makespan (arguably the most important), work-in-process (WIP) cost, number of tardy jobs, and runtime.

In real-world scenarios, assembly line scheduling problems are often complex and grow exponentially with problem size. To address these challenges, we must account for resource constraints, sequence dependencies, dynamic environments, and trade-offs between conflicting objectives. Heuristic approaches are essential due to computational limitations, and integrating Bills-of-Materials (BOMs) information further complicates the task. In this report, we summarise our design process into eight major steps:

1. **Define:** Identify and define the problem
2. **Measure:** Measure the needs and set performance targets
3. **Explore:** Explore various design space
4. **Optimise:** Optimise the selection of design alternatives
5. **Develop:** Develop the system's framework
6. **Validate:** Validate the design requirements
7. **Execute:** Execute the design into action
8. **Iterate:** Iterate and refine the design process

### 1.1 Mission Statement

Considering these pain points, our team has came up with the following mission statement:

*"To develop an intuitive and efficient web application that incorporates heuristic methods to optimise large-scale assembly line scheduling for manufacturing companies, tailored to their specific objective(s)."*

This solution will enable the creation of the near-optimal sequence of tasks, addressing the complexities and scale of large-scale assembly line scheduling challenges. The application will enhance productivity, reduce operational inefficiencies, thereby significantly improving overall manufacturing efficiency through streamlining work flow.

### 1.2 Tailor the Process

To meet tight deadlines and multiple milestones of this project, it is essential to strike an optimal balance between planning and executing the project. Hence, we will use the eight major steps as outlined in [Jackson, 2009] to avoid unnecessary design flaws and wasting the team's precious time and effort. We began by defining our project and establishing the system context for our products, which are heuristic approaches and web application for visualisation purposes, including customer feedback and identifying functional requirements for the algorithms and the web application features. Next, we set measurable performance targets for our assembly line scheduling problem. Afterwards, we explore various algorithms to address our objectives (such as minimising makespan, minimising work-in-process costs, minimising number of tardy jobs, and/or minimising algorithm runtime). We will then optimise these algorithms, potentially selecting multiple algorithms for our multi-objective problem. Detailed functional behaviours and subsystems will be defined, considering any physical constraints. A master test plan will be developed to ensure the web application will meet its intended purpose, such as generating Gantt charts and BOM and solving the optimisation problem using heuristic approaches. Finally, we will execute the web application at the end of the project deadline and remain open to iterating the process if resources allow.

To enhance project efficiency and minimise miscommunication, it is crucial to clearly define system boundaries. This involves articulating our internal and external entities, ensuring a shared understanding of project scope and limitations. Clear boundaries will foster coordination, teamwork, and a unified project framework.

**Internal Entity:**
The internal entity comprises multiple subsystems that need to be decomposed. These internal entities are the components or elements within the system that are directly involved in its processes and operations, and therefore fall within the scope of our design.

Our internal entity can be divided into two subsystems: the front end and the back end. Our project aims to integrate both subsystems into a seamless web application that offers a comfortable user experience and efficiently produces an optimized schedule based on inputted objectives.

These are the components that make up the back end subsystem:

- **Bill-of-Materials (BOMs):** For our assembly line challenge, a Bill of Materials (BOM) is essential. This document outlines the sequential assembly of components, ensuring a logical and efficient production process. From listing basic parts and quantities to detailed multi-level structures, including part numbers and manufacturing specifics, the BOM guides our production strategy effectively.
- **Heuristic:** Further details on our selected heuristic will be explored in Section 3 [Exploring Decision Space].

As for the front end subsystem, these are some of the components:

- **Web Application:** Our final product will be a web application enabling users to apply our heuristics for assembly line scheduling, providing near-optimal solutions, Gantt charts, and Bills-of-Materials (BOMs) information.
- **Gantt Chart:** This chart provides a clear, compact tool for scheduling tasks, promoting accountability and clarity in task assignments. It also aids management by highlighting deviations from the plan and attributing responsibility for outcomes.

**External Entity:**
External entities are users, systems, or any components outside the boundary of the system being designed or modified. Entities that are external to the system have their own goals, functions, behaviour patterns, and interfaces. The external entities identified in our case are:

- **Manufacturing companies, Stakeholders:** These entities operate outside the boundary of the system being designed. Manufacturing companies are the end-users who seek to optimize their production processes, stakeholders provide critical input, feedback, and approval without interacting directly with the system's internal processes.

- **Order database:** The order database, while crucial for operations, exists outside the system's direct control. It stores data needed by the system but is managed and maintained separately.

## 1.3 Context Diagram

With a clear understanding of our internal and external entities, we create a context diagram. This diagram captures relationships between our system and external entities, providing a visual overview of our system's scope, boundaries, and interfaces with other systems.

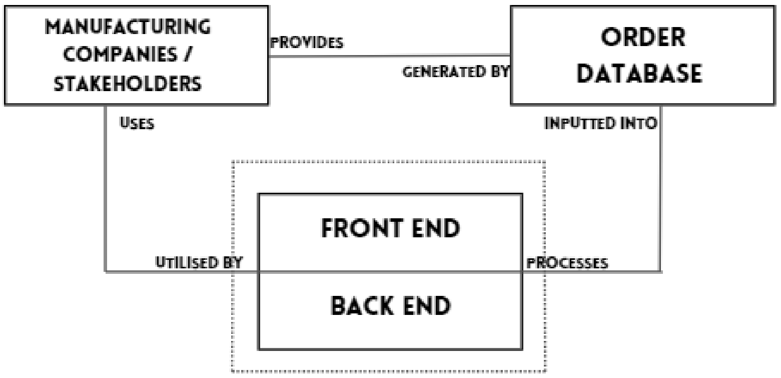

Figure 1: Context Diagram

## 1.4 Affinity Process

The final step in defining the context involves summarising project (product) objectives. This enables our team to structure the wealth of information gathered throughout the various stages of design thinking into coherent, associative relationships. It helps organise this information to effectively address customer needs.

| **Timeliness and Scheduling** | **Resource Management** | **User Experience & Integration** |
|---|---|---|
| Are we completing production within the designated start and end times for each job? | Are we efficiently allocating resources such as materials, equipment, and labour to tasks? | Can the system handle large-scale operations without performance degradation? |
| How can we minimize the total time needed to complete all tasks, and what is the near-optimal job sequence? | Are we including the time needed for machines to begin production on new tasks, considering set-up requirements? | Is our web application easy to navigate and use? |
| Are we respecting dependencies and performing tasks in the correct order? | How can we distribute tasks evenly across resources to prevent bottlenecks and overloading? | - |

## 1.5 Identify the Owners, Customers, and Users

It is a good start to have a clear idea of who is involved and what their point of view is before collecting the user comments and the Voice of Customers (VoC). In the context of our project, assembly line scheduling, we distinguish our individuals into 3 different major roles at this stage. The roles are elaborated below:

- **Owner:** Individual or entity responsible for setting the overall design goals and making major design decisions. They have the authority to authorise project initiatives and allocate resources. In our case, we divide the roles of owner into 3 roles (Client, Architect, and Builder).
  - **Client:** represents the high-level goals of the system and measures its success. They express their desires and requirements to the development team, which is our team, and they are critical for the project success. In our case, our clients are Professor Rakesh and Zeyu.

    Companies who want to develop web applications for their assembly line scheduling problem can also be our potential clients. For example, companies like:
    - **Engineering and Project Management Firms:** Surbana Jurong and Worley.
    - **Manufacturing Companies:** Micron Technology and HP Singapore.
    - **Construction Companies:** Dragages Singapore and GS Engineering & Construction.
  - **Architect:** translates the client's desires into detailed specifications for the system. They bridge the gap between high-level goals and technical implementation. Our team will define the system's structure, components, and interactions. These decisions will guide the builder's work.
  - **Builder:** constructs the system based on the architect's specifications. They implement the design, write algorithm codes, and create the system components. While they have the authority to make some low-level design choices, they are highly constrained by the architect's guidelines. Therefore, our team will act as the builder as well and ensure that the system aligns with the intended design and purpose.
- **Customer:** Individual or organisation that approves the purchase of the system. They take ownership of the system once it is ready for deployment to the market. Additionally, their satisfaction with the final product is crucial. In our case, Professor Rakesh and Zeyu can be our customers.

  Based on our research, these are some real life companies in Singapore that might require Gantt chart website application for their business operations and can be our potential customer as well:
  - **Electronics Manufacturing:** Flextronics, Venture Corporation, and ST Electronics.
  - **Precision Engineering:** JEP Precision Engineering and UMS Holdings.
  - **Aerospace:** Singapore Technologies Engineering (ST Engineering) and Rolls-Royce Singapore.
- **User:** Individual who directly interacts with the system for its intended purpose. They are the ones who use the system's features and functionalities. Understanding their needs and ensuring usability is essential. For our project, the production managers are the primary users.
- **Unintended user:** Individual who are affected by the system's existence or operation, even if they are not direct users. These could be employees within the company impacted by the system changes. Considering unintended users helps us avoid negative consequences.

## 1.6 Collect Customer Comments and Create Voice of the Customers

Before setting targets and measuring needs, we must collect customer feedback related to the problem we aim to solve. This will help us gain a deeper understanding of the customer needs and create the products. Below is the voice of customers (VoC) that we have collected via interviews and information gathering from the internet.

- **Criticality of scheduling efficiency**
  - We need a tool that can significantly improve our scheduling efficiency to minimise idle time and optimise the use of resources.
  - Our current scheduling process is manual and time-consuming. Automation through a web app could save us considerable time and reduce human errors.
  - Efficient scheduling is crucial for meeting tight production deadlines and maintaining customer satisfaction.
- **User Interface preferences or accessibility**
  - The interface must be intuitive and accessible, allowing users of varying technical expertise to use the tool effectively.
  - We want the interface to be as simple as possible and easy to navigate.
  - Accessibility features such as keyboard shortcuts and screen reader compatibility are important for some of our employees.
- **Optimisation performance**
  - The tool must provide high optimisation performance, delivering accurate and efficient scheduling results quickly.
  - The tool must utilise various existing algorithms and provide comparison for the solutions and give the best result possible.
- **Success measurement for the Webapp (e.g. more user-friendly, aesthetic)**
  - Success can be measured by the user-friendliness and aesthetic appeal of the web application, as well as its ability to deliver accurate scheduling.
  - Metrics like reduction in scheduling time and improvements in on-time delivery rates will help us measure the effectiveness of the tool.
- **Webapp medium preferences**
  - The application should be responsive in various devices and browsers to ensure accessibility and usability in different environments.
  - We have no preferences about the medium for the webapp. We are satisfied as long as it does their job and shows the results we want.
  - Fast loading times and responsive design are critical for ensuring a smooth user experience across different browsers.
- **Data security and Privacy**
  - We need assurances that our data will be secure and that the application complies with data privacy regulations.
  - Regular security updates and compliance with standards like GDPR are important for protecting sensitive information.
  - Role-based access control and encryption of data both at rest and in transit are essential security features.
- **Scalability and Flexibility**

- The solution should be scalable to accommodate growth and flexible to adapt to different types of operations and industries.
- We need a tool that can handle increased workloads as our business expands.
- Customizable features and modular design will allow us to adapt the tool to our specific needs over time.

## 2 Measure the Needs and Set Targets

### 2.1 Define Requirements

We define our requirements by two categories: functional and performance. We exclude interface and crosscutting requirements as it does not apply to this project. Afterwards, we validate our requirements in Section 2.2 using the six steps taught in ESA.

#### 2.1.1 Functional Requirements

1. The algorithms shall be able to return a production schedule formed using bill of materials (BOM) and order data.
2. The algorithms used shall be able to return a production schedule that achieves their respective objectives.
3. The algorithms shall achieve their respective objectives by utilising heuristics.
4. The algorithms shall satisfy capacity constraints related to the availability of work centres.
5. The algorithms shall generate a production schedule that complies with BOM precedence constraints.
6. The algorithms shall be able to work on large databases, where the number of operations and machines go up to 200 each.
7. The web application shall utilise Gantt charts to represent the schedule that is produced.
8. The web application shall give users a choice of which objective they want to prioritise.
9. The web application shall be responsive to different screen sizes. When the screen width is below 800px, the layout should adapt by switching from a horizontal desktop view to a vertical mobile view.
10. The web application shall have clear labeling of buttons and menus.

#### 2.1.2 Performance Requirements

1. The algorithm shall have a runtime of less than 5 minutes to generate a schedule for a small number of operations (within 20 operations) and a runtime of less than 30 minutes to generate a schedule for a larger number of operations (within 200 operations).
2. The algorithm shall be able to scale to large dimensions of up to 200 operations with 20 machines.
3. The algorithm shall be able to return a near optimal production schedule which prioritises the objective selected by the user.

4. The algorithm shall ensure all feasible cases have less than 20% of its products with an end time later than its due date.
5. The web application shall display signs of the algorithm running within 10 seconds of submitting a query to the server.
6. The homepage of the web application shall load within 5 seconds.
7. The web application shall not exceed 500mb of memory usage per user session.

### 2.2 Validate Requirements

The process of validating requirements can be broken into six questions.

**Qn 1:** Are the requirements written correctly?
Ans: Requirements are written in 'shall' statement format and grammatically accurate.

**Qn 2:** Are the requirements technically correct?
Ans: Requirements were formed using valid assumptions, and have bidirectional traceability to the stakeholder expectations.

**Qn 3:** Do the requirements satisfy stakeholders?
Ans: All relevant stakeholders identified are satisfied with end product features.

**Qn 4:** Are the requirements feasible?
Ans: All requirements make technical sense and are possible to achieve.

**Qn 5:** Are the requirements verifiable?
Ans: Requirements have a clear criteria and output to verify whether it has been met.

**Qn 6:** Are the requirements redundant or over-specified?
Ans: Requirements are individually unique and are not redundant to any other requirements.

## 3 Explore the Design Space

### 3.1 Discover and Explore Different Potential Alternative Solutions or Approaches

Heuristics are a class of problem-solving techniques that prioritise finding "good enough" solutions within a reasonable timeframe. They achieve this by using readily available information and following a set of guiding principles rather than exhaustively exploring every possible option.

In contrast to traditional optimisation methods that struggle with large-scale assembly line scheduling, heuristics offer a powerful alternative. Their computational efficiency allows for quick generation of good quality solutions, crucial for real-time decision making. Additionally, their flexibility enables customisation based on line constraints, task priorities, or performance objectives, leading to a more suitable approach for dynamic environments. Ultimately, heuristics provide a valuable balance between solution quality and computational effort, offering near-optimal results that significantly outperform random scheduling while remaining achievable within practical timeframes.

In the context of assembly line scheduling, our team will explore different heuristic methods and algorithms such as Scheduling Algorithm (LETSA), Lagrangian Relaxation (LR) and Shifting Bottleneck Procedure. We also aim to explore metaheuristic methods like genetic algorithms, simulated annealing, and tabu search, together with the multistart technique, which offer viable alternatives by providing near-optimal solutions efficiently. While we mainly focus on the heuristics listed below, we acknowledge the importance of optimisation approaches and consider them if they were to complement heuristic methods. This ensures an identification of the most effective strategies for assembly line scheduling.

#### 3.1.1 Methods

- **Lead Time Evaluation and Scheduling Algorithm (LETSA)**
  [Agrawal et al., 1996] developed a heuristic to generate a close-to-optimal schedule by prioritizing the execution of jobs which belong to a dynamically computed critical path. The proposed method, evaluated based on cumulative lead-time and work-in-process (WIP) inventory costs, outperforms various scheduling heuristics developed by past researchers, be it simple sequencing rules such as SPT (Shortest Processing Time), EDD (Earliest Due Date), FCFS (First-Come-First-Served) or more sophisticated algorithms such as SIMFACTORY, a commercially available shop-floor simulation tool. LETSA largely outperforms these heuristics in terms of producing a near-optimal solution, as well as minimizing WIP holding costs via a just-in-time production approach.

- **Lagrangian Relaxation (LR)**
  [Xu and Nagi, 2013] present an approach utilising lagrangian relaxation to solve the MILP formulation for the assembly scheduling problem. What sets apart the lagrangian relaxation, apart from being able to produce fast and good solutions, is the method's ability to produce the solution's gap towards optimality. This makes it easier to determine the algorithm's stopping criteria for problems where the optimal solution is strenuous to find.

- **Shifting Bottleneck Procedure (SBP)**
  [Liu and Kozan, 2012] and [Adams et al., 1988] present a scheduling algorithm which specialises in solving assembly scheduling problems where bottleneck machines complicate the search for an optimal solution. This heuristic outperforms other methods in terms of solving bottleneck situations, but perhaps not as well when it comes to solving other objectives such as minimising makespan.

- **Memetic Algorithms (MA)**
  Memetic algorithms are one of the most effective and flexible metaheuristic approaches for tackling hard optimisation problems. [Hasan et al., 2009] and [Cotta et al., 2018] present a heuristic based on memetic algorithms to solve the assembly scheduling problem. MA searches the global and local solution space in order to reduce the chance of a local convergence. This enables the algorithm to efficiently explore and exploit the search space, providing high-quality solutions for complex optimisation problems.

- **Genetic and Evolutionary Algorithms (GA & EA)**
  [Zhang et al., 2011] and [Mesghouni et al., 2004] present genetic and evolutionary algorithms to solve the job shop scheduling problem. Their algorithms draw inspiration from biological processes, namely genetic mutations and evolution, to find a near-optimal solution. These algorithms simulate the process of natural selection whereby only the "fittest" survives: that is, weak solutions are discarded as the algorithm runs. This approach enables the genetic algorithm to generate an optimal solution effectively and efficiently.

- **Variable Neighbourhood Search Algorithm (VNS)**
  Search-based approach centred on exploring more than one type of neighbourhood structure [Roshanaei et al., 2009]. VNS iterates in an analogous way over some neighbourhood structure until some stopping criterion is met. Either a local minimum of one neighbourhood structure

or the global optimum of all neighbourhood structures are observed. VNS starts from an initial solution produced by Shortest Processing Times rule for job shops, and utilises 3 types of Neighbourhood Search Structure (NSS) that are based on insertion neighbourhood. A significant advantage of VNS compared to other local search-based metaheuristic algorithms is that VNS is equiped with several neighbourhood search structures which diversifies the search space. This prevents the metaheuristic from getting stuck in the local optima and hence a more accurate solution.

- **Multi-type Particle Swarm Optimisation (MPSO)**
  MPSO enhances traditional Particle Swarm Optimisation algorithm and incorportates multi-type individual enhancement schemes [Lin et al., 2010]. This prevents the solution from converging to the local optima prematurely. MPSO adopts the real space as the search space called random-key (RK) and does not require a heuristic algorithm to initialise a population to speed the convergence rate. Implementing a heuristic algorithm increases the computational load. MPSO is able to reach the optimal area in the search space with a smaller population size and iterations.

- **Simulated Annealing (SA)**
  Simulated annealing is inspired by the annealing process in metallurgy [Kirkpatrick et al., 1983]. A control parameter, temperature, is used to determine the probability of accepting non-improving solutions. The temperature is gradually decreased according to a cooling schedule such that few non-improving solutions are accepted at the end of the search such that it converges to an optimal or near-optimal solution. Simulated annealing has the flexibility to handle a wide range of optimisation problems and is generally more capable of escaping the local optima, giving it a better chance of finding the global optimum. However, the cooling schedule needs to be tuned, if it is too fast, the algorithm may converge to a local optimum, and if it is too slow, the algorithm may take too long to converge. It may also require a large number of iterations to find a good solution with increased complexity.

- **Tabu Search (TS)**
  Tabu search enhances the performance of local search by relaxing constraints to explore a broader solution space [Glover et al., 1993]. Unlike hill climbing algorithms that only accept improving solutions, tabu search allows for non-improving solutions to escape from local optima, allowing the algorithm navigate out of local optima when all neighbouring solutions are non-improving. A tabu table is introduced to prohibit the revisitation of recently explored solutions, thus leading the search away from previously visited solutions. This approach can lead to cycles where the solution repeats itself within a local optimum. To mitigate this, a larger tabu list can be used (determined by varying the table size or allocated a dynamic tabu list). This maintains a history of a broader set of visited solutions, preventing the formation of cycles. By doing so, the tabu search algorithm can explore new areas of the solution space more effectively, thereby enhancing the overall performance in finding a high-quality solution.

For the front end itself, there are three alternatives that we considered and their respective reasons:

- **ReactJS:** React, widely used by developers globally, excels in creating interactive user interfaces. It's a dominant library among web developers, with over 40% reporting extensive usage based on 2023 Stack Overflow surveys. React's declarative views enhance code predictability and simplify debugging. Additionally, its encapsulated components, managing their own state, promote better code comprehension for team members.

- **RShiny:** RShiny, built upon the widely acclaimed R language for statistical modeling and data analysis, provides a viable option. It enables the development of interactive web applications that execute R code on the back end.

- **Docker:** Docker offers a platform that streamlines the deployment and management of applications. Its ability to isolate applications and their dependencies within containers ensures consistent performance across diverse environments. This feature makes Docker a compelling candidate for enhancing our front-end development processes.

## References

[Adams et al., 1988] Adams, J., Balas, E., and Zawack, D. (1988). The shifting bottleneck procedure for job shop scheduling. *Management science*, 34(3):391–401.

[Agrawal et al., 1996] Agrawal, A., Minis, I., and Nagi, R. (1996). 'just-in-time' production of large assemblies. *IIE Transactions*, 28(8):653–667. `https://doi.org/10.1080/15458830.1996.11770710`.

[Cotta et al., 2018] Cotta, C., Mathieson, L., and Moscato, P. (2018). *Memetic Algorithms*, pages 607–638. Springer International Publishing, Cham. `https://doi.org/10.1007/978-3-319-07124-4_29`.

[Glover et al., 1993] Glover, F., Taillard, E., and Taillard, E. (1993). A user's guide to tabu search. *Annals of Operations Research*, 41(1):1–28. `https://doi.org/10.1007/BF02078647`.

[Hasan et al., 2009] Hasan, S. M. K., Sarker, R., Essam, D., and Cornforth, D. (2009). Memetic algorithms for solving job-shop scheduling problems. *Memetic computing*, 1(1):69–83. `http://dx.doi.org/10.1007/s12293-008-0004-5`.

[Jackson, 2009] Jackson, P. (2009). *Getting Design Right: A Systems Approach*. CRC Press.

[Kirkpatrick et al., 1983] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680. `https://doi.org/10.1126/science.220.4598.671`.

[Lin et al., 2010] Lin, T.-L., Horng, S.-J., Kao, T.-W., Chen, Y.-H., Run, R.-S., Chen, R.-J., Lai, J.-L., and Kuo, I.-H. (2010). An efficient job-shop scheduling algorithm based on particle swarm optimization. *Expert Systems with Applications*, 37(3):2629–2636. `https://doi.org/10.1016/j.eswa.2009.08.015`.

[Liu and Kozan, 2012] Liu, S. and Kozan, E. (2012). A hybrid shifting bottleneck procedure algorithm for the parallel-machine job-shop scheduling problem. *The Journal of the Operational Research Society*, 63(2):168–182. `http://dx.doi.org/10.1057/jors.2011.4`.

[Mesghouni et al., 2004] Mesghouni, K., Hammadi, S., and Borne, P. (2004). Evolutionary algorithms for job-shop scheduling. *International Journal of Applied Mathematics and Computer Science*, 14:91–103. `https://api.semanticscholar.org/CorpusID:3063358`.

[Roshanaei et al., 2009] Roshanaei, V., Naderi, B., Jolai, F., and Khalili, M. (2009). A variable neighborhood search for job shop scheduling with set-up times to minimize makespan. *Future Generation Computer Systems*, 25(6):654–661. `https://doi.org/10.1016/j.future.2009.01.004`.

[Xu and Nagi, 2013] Xu, J. and Nagi, R. (2013). Solving assembly scheduling problems with tree-structure precedence constraints: A lagrangian relaxation approach. *IEEE Transactions on Automation Science and Engineering*, 10(3):757–771. `http://dx.doi.org/10.1109/TASE.2013.2259816`.

[Zhang et al., 2011] Zhang, G., Gao, L., and Shi, Y. (2011). An effective genetic algorithm for the flexible job-shop scheduling problem. *Expert Systems with Applications*, 38(4):3563–3573. `https://doi.org/10.1016/j.eswa.2010.08.145`.