

Sharded Ledgers for
Micro-transactions and Automated
Assembly Line Planning

Christoffer Fink

Cyber-Physical Systems

Sharded Ledgers for Micro-transactions and Automated Assembly Line Planning

Christoffer Fink

Dept. of Computer Science, Electrical and Space Engineering
Luleå University of Technology
Luleå, Sweden

Supervisors:

Olov Schelén, Ulf Bodin, Johan Kristiansson

To Ada, Bibbi, and Helen.

ABSTRACT

This thesis addresses two research areas: scalable distributed ledgers for micro-transactions, and the automation of assembly planning in manufacturing industries.

Established blockchain solutions are robust and reliable. Being distributed and decentralized, they avoid a single point of failure, and fault-tolerant consensus mechanisms ensure that the system works as intended even when some participants are faulty or malicious. However, their main weakness is scalability. The two most popular and well-known blockchain solutions, Bitcoin and Ethereum, require all nodes to store all transactions, and their transaction throughput is far too low to compete with traditional, centralized transaction processing systems. To improve scalability, systems have been developed that split the network nodes into groups that can process transactions in parallel, a technique known as sharding. We propose a sharded system called ScaleGraph that uses a novel architecture with one transaction per block and one shard per account, designed to maximize parallelism. The design is inspired by concepts from distributed hash tables, particularly to define shards based on a logical distance metric for node IDs and account IDs. Nodes store and process only transactions involving those accounts that are close to the node according to the distance metric. This greatly reduces the storage burden on each node and allows any number of transactions involving distinct accounts to be validated in parallel. We also design a new cross-shard transaction commit protocol for this architecture. The protocol offers global serializability and inevitable atomic commit, without the need for an abort path. This is achieved using only shard-local consensus and certificate exchange, rather than global or joint cross-shard consensus.

Manufacturing is a highly complex process in many industries and involves many different planning problems where increasing automation has the potential to make manufacturing more efficient. This thesis presents a proof-of-concept solution to the kitting layout problem, where a list of parts has to be placed on a kitting wagon for delivery to an assembly line station. However, some problems have proven difficult to automate in practice, despite decades of research. One such problem, assembly line balancing, is analyzed in depth. We identify fundamental challenges that make the goal of complete automation implausible in some industries, such as automotive manufacturing. Human intervention is thus unavoidable, suggesting that bridging the gap between theory and practice requires decision support systems for assisted, iterative, and interactive planning. The thesis also includes preliminary work on the product sequencing problem, limited to framing the use case, assumptions, and requirements. Subsequent ongoing work suggests strong parallels to assembly line balancing, indicating that the identified challenges and possibilities for addressing them reflect a broader pattern in industrial planning automation.

CONTENTS

CHAPTER 1 – INTRODUCTION	1
CHAPTER 2 – MICRO-TRANSACTIONS	3
2.1 Background	3
2.2 Motivating Challenges	11
CHAPTER 3 – ASSEMBLY LINE PLANNING	13
3.1 Background	14
3.2 Motivating Challenges	20
CHAPTER 4 – RESEARCH QUESTIONS	21
CHAPTER 5 – CONTRIBUTIONS	23
5.1 Paper A – ScaleGraph	23
5.2 Paper B – Cross-Shard Commit Protocol	24
5.3 Paper C – Layout Planning	25
5.4 Paper D – Assembly Line Balancing	25
5.5 Paper E – Flexible Termination Criteria	26
5.6 Paper F – Product Sequencing	27
CHAPTER 6 – RESEARCH METHODS	29
6.1 Constructive, Analytical, and Evaluative Work	29
6.2 Synthesis, Requirements, and Conceptual Framing	31
6.3 Relations Between Methods and Question Types	31
6.4 Summary	32
CHAPTER 7 – CONCLUSIONS, DISCUSSION, AND FUTURE WORK	35
7.1 Conclusions	35
7.2 Discussion and Future Work	36
REFERENCES	39
Part II	45
PAPER A	47
1 Introduction	49
2 Critical event ordering	52
3 Overview of ScaleGraph	53

4	Consensus in ScaleGraph	57
5	Security Analysis	60
6	Simulation Experiment	64
7	Background and Related Work	72
8	Discussion and Future Work	82
9	Concluding Remarks	86
PAPER B		93
1	Introduction	95
2	System Model	98
3	Cross-Shard Protocol	99
4	Correctness Properties	106
5	Formal Specification and Model-Checking	109
6	Scalability	109
7	Related Work	119
8	Discussion and Future Work	121
9	Concluding Remarks	123
10	Acknowledgments	123
PAPER C		125
1	Introduction	127
2	Layout Planning	129
3	Constraint programming	131
4	Implementation	133
5	Evaluation setup	136
6	Results	139
7	Discussion and future work	142
8	Conclusions	143
PAPER D		147
1	Introduction	149
2	Explaining the Theory-Practice Gap	151
3	Fundamental challenges	158
4	Decision Support Systems bridging the Theory-Practice Gap	161
5	Solver features relevant in a DSS context	166
6	Discussion and future work	171
7	Summary and conclusion	172
PAPER E		183
1	Introduction	185
2	Use case	186
3	Termination criteria	188
4	Experiment: Rigid vs Flex	190
5	Discussion and Future Work	196
6	Summary and Conclusion	197

PAPER F	201
1 Introduction	203
2 Related Work	204
3 Use Case	206
4 Proposed approach	206
5 Discussion	209

ACKNOWLEDGMENTS

I am grateful to my supervisors, Olov Schelén, Ulf Bodin, and Johan Kristiansson for their guidance and flexibility. I also want to thank my family and girlfriend for their support. Finally, I thank Volvo and SEB for their collaboration.

This work was funded by KDT JU 2022 IA Arrowhead fPVN (grant 2023-00450), Arrowhead fPVN (Chips JU grant 101111977), Arrowhead Tools under grant agreement no. 826452 (ECSEL Joint Undertaking), the POAM prestudy (Vinnova FFI grant 2021-05071), the EUREKA ITEA4 ArtWork project (Vinnova grant 2023-00970), STAMINA (Vinnova grant 2024-0644), and SIM (Vinnova grant 2025-01060).

CHAPTER 1

Introduction

This thesis contains two research threads: blockchain systems scalable enough to handle large transaction volumes, e.g. micro-transactions, and automating planning problems in manufacturing industries, such as assembly line balancing, product sequencing, and kitting. Each subject is introduced separately and in more detail in the following two chapters. These are two distinct domains, and the thesis does not attempt to force a strong conceptual unification of the two threads. However, both can be related to Cyber-Physical Systems (CPS), which involve interactions between computation, communication, and physical processes. Manufacturing industries deal with planning and control in complex industrial systems where different planning problems are interconnected and where digital information and physical processes are tightly coupled. Distributed ledgers for micro-transactions are relevant to CPS through machine-economy scenarios in which autonomous devices transact without human involvement. The two themes also touch each other more directly in industrial settings where, for example, tools or other equipment could act as autonomous participants that can charge for their use.

Micro-transactions require systems that can handle very large transaction volumes. Traditional distributed ledger systems have generally not been designed for this level of scalability, which motivates the question of **how a distributed ledger can be made scalable enough to support such workloads**. This is the starting point for the first research thread of the thesis.

Manufacturing involves many planning problems, with a is clear potential to improve both planning and production efficiency through increased automation. At the same time, some planning problems have proven difficult to automate in practice, despite decades of research. This motivates the second research thread of the thesis, which concerns the **challenges and possibilities for automating assembly line planning**.

Chapter 2 introduces blockchain systems and provides the necessary distributed-systems background. Chapter 3 does the same for assembly planning. Chapter 4 elaborates on the two research questions. Chapter 5 lists and summarizes the included publications. Chapter 6 describes the research methods used to address the questions posed in the included papers. Finally, Chapter 7 discusses the conclusions that can be drawn from the included work and the future research directions it suggests.

CHAPTER 2

Micro-Transactions

One of the use cases that typically places a high demand on transaction throughput is micro-transactions. Micro-transactions involve very small amounts but are instead expected to occur with high frequency, resulting in large transaction volumes. A system for micro-transactions should therefore have two properties:

1. The transaction throughput must be high enough to handle the expected volume of transactions.
2. Transaction fees must be low enough to make it economically feasible to transfer a small amount in each transaction.

The throughput requirement can potentially be met by designing the system to have sufficient capacity in absolute terms. Another approach is to design it to be scalable, so that the capacity can be increased as required to meet the demand. Scalability, in particular horizontal scalability, can also allow the system to consist of a large number of lower-performance nodes¹, keeping the cost of the infrastructure and therefore transaction fees lower.

This thesis proposes a solution that aims to achieve scalability via a novel approach to (full) sharding. But to appreciate the challenges and relevant trade-offs involved, some background needs to be established.

2.1 Background

We will start by explaining some basic concepts and defining important terms.

¹It is sufficient to think of a “node” as one of the networked computers that make up the distributed system.

2.1.1 Basic Concepts

How to Share a Cookie

Alice and Bob are siblings. They have a cookie that they would like to divide fairly so that each gets an equal share. What they really want is to have the whole cookie, or at least as much of it as possible. But sharing it equally is an acceptable compromise and avoids giving either one an excuse to resort to violence.

If they are both trustworthy, any of them can be relied on to split the cookie as evenly as possible. But what if they would rather not rely on that assumption? Perhaps they could involve some third party. For example, a parent could split the cookie for them. What if there is no parent around? Is there some way Alice and Bob can share the cookie fairly, without trusting each other or a third party?

Yes. Either one of them is arbitrarily assigned the role of leader, whose job it is to split the cookie into two pieces. The other person then chooses one of the pieces, and the leader gets the remaining piece.

This protocol is self-enforcing because it is in the leader's interest to make the split as even as possible. Otherwise, the other party would simply choose the bigger piece. The protocol can also be extended to any number of participants.

Trust

The story about Alice and Bob sharing a cookie used the words “trust” and “trustworthy” very carefully. Colloquially, we often use “trust” to express a subjective judgement about someone's trustworthiness. In designing protocols and systems, such subjective judgements are usually not helpful. More relevant is whether, objectively speaking, some component of a system is required to be trustworthy. A trusted component is one that is relied on to function as intended, or else the system will break. If that component were not in a position to break the system, what would it even mean to trust it?

If Alice delegates the decision about how to divide the cookie to Bob, it would be meaningless to say that she does not trust him. She may or may not believe him to be trustworthy, but that subjective fact has no bearing on the situation, which is that she is in fact relying on him being trustworthy. If he is not, she will not get a fair share, and the system has failed. Similarly, if they ask a third party to divide the cookie for them, that is in fact a trusted third party. In that case, they do not trust each other – whether or not they believe each other to be trustworthy is irrelevant – but they do trust the third party, whether or not they believe it to be trustworthy.

The beauty of the self-enforcing cookie sharing protocol is that there is no need for Alice and Bob to trust each other or a third party. The protocol works in a *trustless* system; it makes trust unnecessary. This is a crucial property in many distributed and, more importantly, decentralized systems. It means that trust assumptions can be replaced by protocol design.

Distributed and Decentralized Systems

The concepts *distributed* and *decentralized* are often confused and even used interchangeably. They are related but separate.

Distributed systems are computer systems whose intercommunicating components are located on different networked computers. For example, fault tolerance and performance can be improved by running the same software on multiple machines. This adds computing resources and redundancy. Another example would be an application that is constructed using a micro-service architecture. While these would both qualify as distributed systems, they need **not** be decentralized.

The concept of (de)centralization has to do with authority and control. This distinction is especially clear when a system consists of a relatively small number of computers in a single location, owned and maintained by e.g. a single company. This would be an example of a distributed but centralized system.

Unfortunately, the word “centralized” seems to be the natural antonym for both “decentralized” and “distributed”, and a better alternative has not been established. This means that “centralized” has two different meanings, which helps explain why the two concepts are so easily confused.

A distributed system is in many cases also distributed over multiple physical locations, but this need not be the case. Hence a system that is distributed in the architectural sense may or may not be distributed in the geographical sense. Generally speaking, the main characteristics and challenges of distributed systems do not depend specifically on geographical distribution.

Finally, rather than a binary and absolute classification, it is meaningful to make a distinction between more/less decentralized [15].

Permissionless versus Permissioned

Permissionless systems allow anyone to participate; no special permission is required. Hence they are also called open, or public. These can be fully decentralized. Permissioned systems do require special permission and are also called closed, or private. Since “someone” has to give permission, permissioned systems are more centralized. However, access can be controlled by a federation or consortium, which would place the system in the gray area between centralized and decentralized.

Byzantine-Fault Tolerance

Computers crash, network cables get cut by excavators, and power fails. A distributed system must therefore cope with a remote process failing in some way; it must be fault-tolerant. One failure model is crash-fault, or stop-fault. In this model, a process either works correctly or not at all. However, the stop-fault model does not account for programs that are buggy and, rather than crashing, start behaving in strange and unexpected ways. Nor does it account for malicious adversaries.

Especially in a decentralized and permissionless system where participants cannot be trusted, a Byzantine-fault model is more appropriate. In this model, a process may

not only stop working, but it may also misbehave in arbitrary ways, which includes intentionally malicious behavior. It may selectively drop or delay messages, provide false or inconsistent information, etc. Faulty processes (or nodes/participants) that exhibit such misbehavior are referred to as Byzantine [25], or sometimes as dishonest or traitors². Correct, non-Byzantine processes are also sometimes called honest or loyal². Byzantine-fault tolerance (BFT) is a property of systems that are designed to work even in the presence of faulty (and possibly malicious) participants. The cookie sharing protocol is an example of a BFT protocol and illustrates how protocol design can replace trust assumptions.

Consensus

A common problem in distributed systems is consensus, i.e. agreeing on some value. If a service is distributed over multiple nodes, it may be important for all nodes to agree on the current state so that requests can be handled consistently. This is certainly the case in the context of processing financial transactions.

BFT consensus protocols are designed to reach consensus even when some nodes are Byzantine. Many consensus protocols involve some form of voting among nodes in a committee. This is precarious in a permissionless system. One issue is knowing which nodes belong to a committee and are expected to participate in the protocol. Another issue is the potential for Sybil attacks, where a single entity masquerades as multiple entities by generating many fake identities. Multiple voting nodes may then be controlled by a single, possibly malicious entity.

Depending on the design of the protocol and the assumptions that can be made, different fractions of Byzantine nodes can be tolerated. Some protocols can withstand up to $1/3$ of the nodes being Byzantine, while others can tolerate up to $1/2$.

Other mechanisms than explicit vote counting are also possible. For example, Bitcoin (see Section 2.1.2) can tolerate that up to $1/2$ of the computational resources in the network are controlled by a single entity. The “number of Byzantine nodes” per se is therefore largely irrelevant in that case.

Distributed Hash Tables

A traditional hash table is a data structure that maps keys to values. Key-value pairs can be added, and a value can later be retrieved (or “looked up”) by providing the corresponding key. The implementation of the data structure is part of the program that uses it and thus runs in a local process on one machine.

It is possible to offer the functionality of a hash table as a service on the network. This is essentially a specialized database, usually referred to as a key-value store. While this involves a separate machine and network communication, it is not a distributed hash table, because the actual implementation is still localized to a single machine.

²However, this description based on ascribed intent can also be misleading and confusing. Ultimately, each process is either faulty or correct; faulty or not faulty.

A distributed hash table (DHT) is instead implemented collectively by multiple networked machines, or nodes. It is a distributed system where the work of storing, adding, and looking up key-value pairs is distributed among the nodes. Common DHTs, such as Kademia, are also decentralized and permissionless, and the nodes cooperate to maintain the data structure while arbitrary nodes may join or leave the network at any time.

2.1.2 A Brief History of Cryptocurrencies

How to Prevent Spam

While e-mail spam is still a problem, it used to be worse. Spam is a problem because there is essentially no cost to the attacker, and so even a minuscule success rate can make it profitable. Suppose it cost a negligible amount of money to send an e-mail. For legitimate users, who send a small number of messages per day, the cost could be ignored. For a spammer, who sends millions of messages, the cost would no longer be negligible.

This was the concept behind Hashcash [2]. The sender would have to expend resources to solve a computational problem. This problem has no clever and simple solution and can only be solved by brute force. For a legitimate sender, solving the problem for a reasonable number of messages is a negligible burden, but, for a spammer, the cost would be too high. Once a solution is found, it is easy to check that it is valid. By attaching the solution to the message, the sender can prove that a certain amount of effort was invested into sending the message, like a form of costly signaling. Because receivers can easily filter out messages that do not come with such a proof-of-work (PoW), spamming is no longer profitable.

Or at least that was the idea. While this solution did not take off, it became crucial inspiration for the design of Bitcoin.

Double-Spending

The single most important problem for cryptocurrencies is arguably the double-spending problem. Physical money is presumed difficult to copy, and each coin can only have one owner at a time. When Alice gives a physical coin to Bob, she no longer has it. But digital money only consists of information, bits, which computers are famously good at copying. How then can we make sure that users cannot simply copy coins?

Alice would arouse suspicion if she gave Bob 10 identical coins. For any individual transaction, it would be easy to check that all the coins are unique. But if Alice can give a coin to Bob in one transaction and then give the same coin to Carol in a different transaction, she would have spent the same coin twice – a double-spending attack.

eCash/DigiCash

The first proposed cryptocurrency, eCash³ [12], predates Bitcoin by 25 years. It solved the double-spending problem using a bank as a trusted third party. The bank kept track of past transactions and, in particular, the coins that had been spent. Every transaction had to involve the bank so it could be validated and the bank could update its record

of spent coins. This is a centralized system because there is a central, trusted authority. While the participants do not need to trust each other, they do need to trust the bank, which becomes a single point of failure.

Bitcoin

Bitcoin [37] was the first cryptocurrency to solve the double-spending problem in a *decentralized* system. It essentially reduced the double-spending problem to a distributed consensus problem. As long as everyone can agree on which transactions have happened, double-spending can easily be prevented.

In Bitcoin, there is a single, global ledger that records all transactions. It is a *distributed* ledger since all⁴ nodes have a copy and collectively maintain the global ledger by updating their local copy. Since all of history is available to each node, checking that only valid transactions are added is straightforward. The challenge is instead how independent nodes can all agree on the same ledger, i.e. reach global consensus, without trusting each other. In particular, if Alice can make a transaction to Bob and later change the ledger so that this transaction is erased, then she can spend the same coin again. It is therefore critical that everyone can agree on which transactions did or did not happen.

Multiple transactions are validated at the same time and bundled into blocks. Blocks also contain the cryptographic hash of the previous block, which links them together, forming a chain of blocks. Any node in the network can propose the next block of transactions to be added to the blockchain. But it has to be accompanied by a PoW, which means that making valid proposals is expensive. Proposals that do not carry a valid PoW are easily checked and discarded. Just as Hashcash used PoW to prevent e-mail spam, Bitcoin uses PoW to filter out spurious proposals.

When other nodes see a valid proposal, they all add the block to their local copy of the chain. But if two blocks are generated simultaneously, there is a fork in the chain, resulting in two competing ledgers. To reach global consensus, one more piece is needed, which is the rule that the longest chain is the canonical, i.e. the “real” one. The next block is highly likely to break the tie and resolve the ambiguity. Sooner rather than later, consensus can eventually be reached.

An attempt to rewrite history by changing an older block would change the hash of that block and make the chain obviously invalid. It would therefore be necessary to solve the PoW again for the altered block *and all later blocks*. Assuming most nodes are honest, they would keep working on extending the canonical chain, and the only way to convince the rest of the network to accept a new and modified chain is to make it the longest chain. An attacker would therefore need enough computational resources (a.k.a. hashing power) to generate PoWs much faster and out-pace all other nodes in the network.

³The concept was proposed by David Chaum, who then started the DigiCash company to commercialize it.

⁴For simplicity, we can ignore that there are different types of nodes and not all store the entire ledger.

The longest chain is the one that has had the most work invested in it. By making it (computationally) expensive to create valid blocks, and always favoring the longer chain, the only way an attacker can alter the ledger is to control a majority of the hashing power in the network⁵.

Bitcoin only requires the assumption that at least half of the total hashing power is controlled by honest nodes. Other than that, there is no trust involved. Like Alice and Bob sharing a cookie, there is no need for the participants to trust each other or some central authority.

Later Systems

Bitcoin has two major disadvantages. One is that the reliance on PoW means that the system consumes staggering amounts of electricity [54], making it unattractive from a sustainability standpoint. More relevant to this thesis is that the maximum throughput is approximately 7 transactions per second (TPS) [14].

Many new cryptocurrencies and blockchain systems have been created after Bitcoin. One of them is Ethereum [8], the second-most popular system [44] after Bitcoin. At approximately 30 TPS [41], its throughput is only slightly higher. Ethereum initially relied on PoW as well, but later switched to a proof-of-stake (PoS) mechanism [22, 9]. While PoS is much more environmentally friendly, it did not improve throughput; nor was it meant to.

These systems do not come close to the kind of throughput that would be required for a widely adopted payment system, not to mention micro-transactions. Their transaction fees are also too high to make micro-transactions economically feasible [22].

Another newer system is Solana [56], which was explicitly designed to handle a much higher transaction throughput. Its claimed capacity rivals that of traditional centralized payment systems. Observable real-world performance is at least several thousand TPS [39], orders of magnitude higher than Bitcoin and Ethereum. Whether it is suitable for micro-transactions is debatable, and it is not clear how strong its theoretical foundation is [47, 45].

There are also systems specifically targeting micro-transactions, such as Bitcoin Lightning Network [42], IOTA [43], and NANO [27].

2.1.3 Scalability

Scalability is sometimes confused with *performance*. While scalability can be a way to achieve high performance, it is a distinct concept.

Scalability refers to the ability to increase performance by adding hardware resources. It is a property that a system, by virtue of its design, may or may not have. But hardware can be added in two different ways. Vertical scalability means that performance can be increased by upgrading an individual node. Horizontal scalability means that performance increases when more nodes are added.

⁵This is technically a slight oversimplification, but it is broadly accurate.

Bitcoin is explicitly designed to not be scalable in either sense. The difficulty of the PoW problem is set so that the block interval, i.e. the time it takes to produce a block, is approximately 10 minutes. If the total hashing power of the network increases, the difficulty is adjusted to maintain the same ~ 10 -minute block interval and the same ~ 7 TPS. Ethereum is similarly limited to a fixed block interval and TPS.

Solana is designed specifically to take advantage of high-performance hardware and be vertically scalable. However, the goal is not horizontal scalability, and adding more nodes would not improve performance.

In addition to performance metrics such as transaction confirmation latency and throughput, storage capacity is also relevant. In most current systems, including Bitcoin, Ethereum, Solana, and many others, adding nodes, and therefore storage space, does not increase the storage capacity of the network. This is because all nodes must agree on a single global state and store the entire ledger. A new node simply uses its storage space to keep yet another copy and does not offload other nodes.

What makes horizontal scalability attractive is that, as the network grows, so does its capacity to handle larger workloads. At the same time, the demand on individual nodes can potentially be lower.

2.1.4 Sharding

One strategy for achieving horizontal scalability is *sharding* [28]. In this context, sharding refers to dividing the nodes into groups, or *shards*. The burden of processing transactions can then be spread among these shards. In particular, different subsets of nodes can process transactions in parallel, more or less independently of each other. If different shards are also responsible for storing different subsets of transactions and network communication is limited to the relevant nodes, the system is *fully* sharded. In this case, the throughput and storage capacity of the network both scale up with the number of nodes.

Other approaches [18] include second-layer solutions such as side channels, where transactions can be processed “off-chain”, as in the lightning network. However, this thesis focuses on first-layer full sharding for scalability. The promise of this approach is that, as more nodes are added to the network, the capacity of the system as a whole is increased and can support more users.

2.1.5 Global Serializability

Serializability means that the effect of a collection of concurrent transactions is equivalent to some serial execution. Global serializability means that this consistency property holds across the whole system, i.e. not only within each shard but when considering all shards together. It is a desirable property because it creates a strong and simple correctness model and aids auditability. Each transaction can then be viewed as a single, atomic event that occurred at some well-defined point in a globally coherent history. Some events are comparable in the sense that one occurred before another. Concurrent events cannot be compared in this way but, because they are concurrent, their order is not

important. What matters is that an order could be assigned to them such that the concurrent execution is equivalent to a consistent serial order. Hence the order is defined when meaningful and is undefined when it is not meaningful. In other words, a partial order is sufficient. Traditional (unsharded) blockchains achieve global serializability by recording all transactions atomically in a single global chain, thereby recording a single sequential log of totally ordered atomic events. A total order is unnecessarily strict and requires global coordination, which limits parallelism and horizontal scalability.

2.2 Motivating Challenges

While sharding sounds like a promising and almost obvious solution to the scalability problem, there are several challenges that must be overcome to successfully implement such a system.

The first challenge is defining the shards. That is, it must somehow be decided which nodes belong to which shards, and transactions that are to be validated must be routed to the appropriate nodes. This means that which nodes belong to which shards must be well-defined and discoverable by any node.

Another challenge with sharding is cross-shard validation. Suppose Alice sends a coin to Bob in a transaction that is validated by one shard, and then sends the same coin to Carol in another transaction that is validated by a different shard. How do the two shards communicate with each other to prevent this double-spending attempt, but without paying a performance cost that negates most of the gain from sharding? Many sharded blockchain designs address this by recording the shard-local effects (e.g. withdrawal and deposit) of a cross-shard transaction separately in each affected shard and coordinating them only loosely. This can preserve eventual atomicity, but it sacrifices global serializability because transactions are no longer recorded as single, atomic events.

A third challenge is maintaining security. Sharding inevitably reduces the global fault tolerance of the network, since each transaction is validated only by a (relatively small) subset of all the nodes. Therefore, an attacker could potentially compromise a shard while only controlling a small number of nodes⁶. This is a fundamental trade-off between security and performance.

The main challenges can be summarized as follows:

- Defining shards and routing transactions accordingly.
- Cross-shard validation that preserves global serializability.
- Maintaining security despite sharding.

⁶Or, in the case of proof-of-work, it is sufficient to control a small fraction of the total computing power as long as it is a majority of the computing power within the shard.

Assembly line planning

Manufacturing is a highly complex process in many industries, involving many planning problems. To illustrate this complexity, some example planning problems are listed below:

- **Assembly sequencing** – dividing the assembly process into a sequence of tasks.
- **Assembly line layout** – the shape and length of the assembly line.
- **Buffer allocation** – placing and sizing buffer areas between stations or connection points for line segments.
- **Assembly line balancing** – assigning tasks to stations.
- **Worker assignment** – scheduling who should do what, where, and when.
- **Kitting layout** – how to place parts on kitting wagons for delivery to stations.
- **Kitting facade** – how to organize parts storage for kitting, i.e. picking parts.
- **Lot sizing** – how many products of each type (model) should be assembled on a multi-model assembly line.
- **Product sequencing** – in which order products of different types (models) should be manufactured on a mixed-model line.

Using optimization algorithms to automatically solve these problems has three potential advantages. First, the machine is able to examine many millions of solutions and is likely to find better solutions compared to purely manual planning. This would allow manufacturing to be more efficient. Second, even if the automated solution is not necessarily significantly better compared to manual planning, the automation can make the planning process itself more efficient and reduce costs before production even begins. Finally, if solving the planning problem is more efficient and less costly, it becomes feasible to update the plans more often, which in turn allows more fine-grained and short-term

adaptation to changing circumstances. If a time-consuming manual effort eventually results in a near-optimal solution that would not be improved through automation, then it would still be infeasible to frequently reconsider the solution. In other words, the second advantage leads right back to the first advantage by making production more efficient as a result of making planning more efficient.

This thesis addresses three of these problems: Kitting layout, assembly line balancing, and product sequencing. Assembly lines in the automotive industry will be assumed as the context, but much of the discussion can be generalized to other industries and manufacturing more broadly.

3.1 Background

3.1.1 Basic Concepts

Assembly Lines

The point of an assembly line is that work can be parallelized by creating a pipelining effect. The workpiece (i.e. the product being assembled) moves from one station to the next. While tasks are performed at one station, other tasks can simultaneously be performed on other workpieces at all the other stations. The pace at which a new workpiece is launched on the line – which typically means each workpiece currently on the line moves to the next station – is the takt time. The total time required to assemble a product is largely unaffected¹, but a finished product leaves the final station each takt. In other words, the latency (i.e. the time between starting work on a new workpiece at the first station and finishing at the last station) is more or less the same, but the throughput can be greatly increased.

The takt time is a function of production targets, which is a function of consumer demand. It is how frequently a product **must** be started/finished in order to meet the desired throughput. The takt time is however limited by the cycle time, which is how long it takes to complete the work at stations and therefore how quickly the workpiece **can** advance along the line. By dividing the work among a larger number of stations, the cycle time can be decreased, which allows the takt time to be decreased and the throughput to be increased.

However, exactly how the work is divided among stations is crucial for determining the cycle time. This is what the assembly line balancing problem is all about, which will be introduced shortly. (See Section 3.1.2)

Single-, Multi-, and Mixed-Model

A single-model assembly line (figure 3.1a) mass-produces a single kind of product; a single model. This greatly simplifies planning because the work at every station is fixed.

¹The total time *can increase*, especially if the work is distributed unevenly among stations so there is a lot of idle time.

Each station always works on the same kind of workpiece and always performs the same tasks. Therefore the tasks that will be performed at each station and how much time is needed is known.

A multi-model line (figure 3.1b) produces batches of varying models. Each batch consists of a single model, but the model differs between batches. Therefore, stations will work on different kinds of workpieces and perform different tasks, and so each station may require different amounts of time depending on the model. However, since there is a finite and relatively small set of models, each station performs one of a few sets of tasks and requires an amount of time that is one of a few different possibilities. More importantly, the set of tasks and the time are known for each batch. Transitioning from a batch of one model to another may incur some setup cost. This situation gives rise to a lot sizing problem, i.e. how many of each model to produce in each batch.

A mixed-model line (figure 3.1c) produces a heterogeneous mix of different models and model variants. Equivalently, we can think of a mixed-model line as a multi-model line with a fixed lot size of 1. Mixed-model lines are common in the automotive industry because there is too much variation to model production as a multi-model line. While there may be a manageable number of base models, customization options effectively create infinite variation. In mixed-model lines, each station may perform a large number of different sets of tasks, each requiring different amounts of time. Since each individual station may experience considerable variation from one product to the next, the variation for the whole line can be extreme, and the combination of tasks and times at all stations at one moment may be completely unique. We no longer have a lot sizing problem, but other planning problems instead become more difficult. This situation also creates a new problem, which is to decide in what order to produce different models so that the variation in work content does not cause trouble on the line. This product sequencing problem will be described in more detail shortly. (See Section 3.1.2)

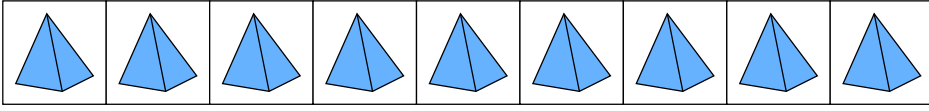
The differences between single-/multi-/mixed-model production are illustrated in Figure 3.1.

Line Pacing and Workpiece Movement

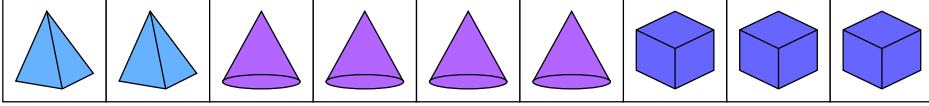
An assembly line can be paced or unpaced. In a paced line, workpieces advance at a set takt time. In an unpaced line, workpieces can all advance synchronously in unison or individually and asynchronously. The latter requires buffer space [51] between stations to absorb the variation by allowing workpieces to temporarily accumulate. Finally, workpieces move in discrete steps and come to rest at each station in a stop-and-go line, while they advance continuously at a constant velocity in a driven line.

Assembly Line Configurations

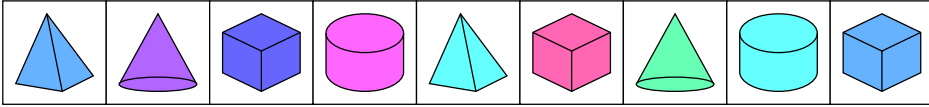
Assembly lines do not have to be limited to a linear sequence of stations. They can have parallel stations or segments [40, 29], which allows the cycle time to be reduced even below the duration of the most time-consuming task. There are also U-shaped lines [34], which provide more flexibility because workpieces pass by the same station



(a) A possible sequence of products for a single-model line. Every product is the same model.



(b) A possible sequence of products for a multi-model line. The models are arranged in batches.



(c) A possible sequence of products for a mixed-model line. Each product may be a unique variant.

Figure 3.1: Comparing single-, multi-, and mixed-model production.

twice. Another common option is the fishbone layout, where feeder lines [49] produce sub-assemblies that are fed to the main line. This enables both parallelism and flexibility. The feeder lines work in parallel with the main line, and variation can be pushed to the feeder lines so that work at the main line can be more uniform. Feeder lines typically operate asynchronously with the main line and have buffers at the connection points [23].

Sides and Work Areas

For industries that work on large workpieces, such as the automotive industry, a station is not atomic but can be divided into smaller areas. That is, it is not sufficient to talk only about the tasks that are performed merely “at the station”, because different tasks may be performed in parallel on the left and right side of the workpiece at the same station, as in two-sided assembly lines [3]. The station and/or workpiece can be further divided into a larger number of sides [5, 24, 36, 30], including also front and rear, and so on. This increases parallelism but complicates planning.

Open Stations

Station boundaries do not have to be fixed and absolute, which would be the case for a closed station [50]. An open station allows operators to cross station boundaries so that work can start early (before the station boundary) or finish late (after the station boundary). This margin is called a drift area [1]. Adjacent stations are then effectively able to borrow time from each other. This works particularly well with a driven line where time and distance are largely interchangeable. However, allowing operators to move between stations [4] achieves a similar effect in a stop-and-go line.

Overloads

Overloads occur when a station is unable to finish its work in time. For example, in a driven line with open stations, this would occur when the work extends even past the drift area. All strategies for dealing with overloads are costly. One option is to plan with a sufficient margin such that overloads are highly unlikely even when there are unexpected delays. But this creates a lot of unproductive idle time in the typical case. Another option is to employ a pool of utility workers [50], also called floaters [32], that can help out at a station that is about to fall behind. But this is just a different way to create idle time in the typical case, and these floaters must be competent and flexible enough to help out anywhere. Another option is to stop the line until the work is finished and production can commence as planned [46]. In this case, all other stations are idle and the profit of one finished product is lost each takt during the stoppage. Finally, the unfinished work can be accepted and later repaired by retrofitting missing parts [11]. This requires extra repair work, which can be more difficult and time-consuming and again requires more competent and flexible operators. Because all options are costly in one way or another, avoiding the possibility of overloads has a high priority.

Kitting

Depending on the line balance and product sequence, a station performs a certain set of tasks that typically involve parts to be mounted. Hence the operators at a station need a certain collection of parts to be available to perform their tasks. In line stocking [7], each station has containers of different types of parts, which are restocked when needed. The operators just take the parts they need for the task at hand from the station's local stock.

Kitting [7] is an alternative to line stocking where the parts that are needed at a particular station at a particular time (i.e. to assemble a particular model variant) are collected into kits and brought to the station. The kitting process typically involves manually picking parts from a kitting facade where an indicator at the relevant shelf or container lights up. The parts may be placed in a container or a on a kitting wagon, which is then delivered to the station.

3.1.2 Planning Problems

Assembly Line Balancing

Assembly line balancing (ALB) assigns tasks to stations. The process of assembling a product is broken down into the different tasks that need to be performed. These tasks then have to be assigned to the stations that make up the assembly line. Each task takes a certain amount of time to perform and is related to other tasks by precedence constraints – some tasks must be performed before/after certain other tasks. The stations have a limited time budget, i.e. an upper limit on the cycle time that is related to the takt time. If tasks are assigned such that the task time is evenly spread over all stations, the cycle time is minimized. However, with mixed-model lines, this is not straightforward.

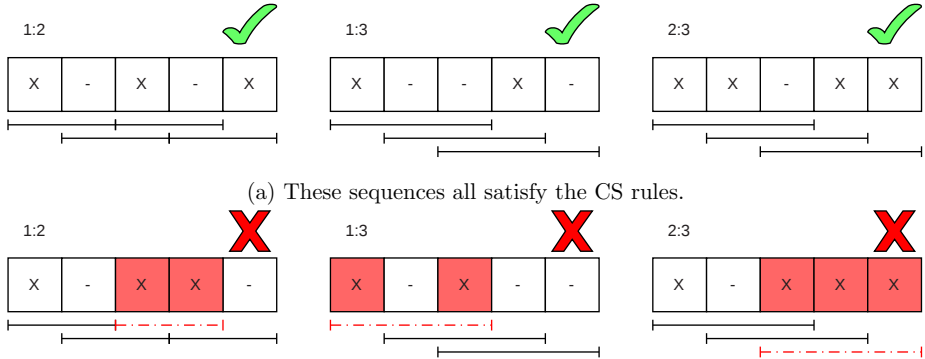
The variations between model variants adds a second dimension. Now, in addition to the variation from one station to the next, the variation within a station (i.e. from one variant to the next) must also be considered. To some extent, the relative frequencies of different models can be taken into account to create a weighted average. This way, the assembly line will be both vertically and horizontally balanced [10] for the expected product mix on average. However, each individual workpiece advancing along the line will have a particular set of tasks and total time at each station that is almost certainly different from the average. Therefore, assembly line balancing alone is typically not sufficient for dealing with the variation experienced on a mixed-model line. The specific sequence of products has to be chosen carefully with the given line balance in mind.

As a result of changes in consumer demand as well as changes to the production process, the line must occasionally be rebalanced [13]. This is typically done a few times per year, which means the ALB problem is solved months ahead of production and is therefore upstream of many other planning problems, including product sequencing. This makes ALB a long-term strategic problem.

Product Sequencing

Since mixed-model lines produce a variety of models that may require different tasks and different amounts of time at each station, the sequence of models to be produced matters. That is, given a set of customer orders, product sequencing is the problem of arranging them in a suitable sequence [50, 6]. Some models may require more total time than the average. The takt time could be set based on the maximum, but that would mean a lot of costly idle time in the typical case. Another option is to allow some models to slightly exceed the takt time. If some models exceed the average time, then some models must necessarily need less time than average. So, on average, the variation cancels out, and temporarily exceeding the takt time may be acceptable because a less demanding model can compensate. However, if several more demanding models occur consecutively in the sequence, one or more stations may increasingly fall behind until they can no longer finish their work in time; an overload. Even if the total time does not exceed the average, a model may require more work than average at a particular station. Again, placing multiple such models in a row in the sequence would overload that particular station.

One approach to product sequencing defines rules for which models or variants may be adjacent in the sequence. More generally, the rules can define how many models of some type X (or variants with some property X) may occur in any sequence of length N . This is expressed by rules of the form $H_X : N_X$, illustrated in figure 3.2. For example $H_X : N_X = 1 : 2$ would require a gap of at least 1 and thus prevent any two occurrences from being directly adjacent. A $1 : 3$ rule would create a gap of at least 2. This is more flexible than mandating a gap of a certain size, since it also allows a limited number of consecutive occurrences. For example, $2 : 3$ would allow at most two (possibly adjacent) out of three, effectively creating a gap of at least $1/2$ on average, since any run of 2 must be broken by a gap of at least 1. This approach is called car sequencing (CS) [38]. The goal is to find feasible sequences, i.e. sequences that do not violate any of the rules or, if no feasible sequences can be found, at least minimize the number of violations.

Figure 3.2: Illustrating $H : N$ CS rules.

Another approach, called level scheduling (LS) [35], focuses more on logistics. The idea is to sequence the products such that the demand for parts is smooth (or level) over time. This approach is useful when considering that feeder lines may need to produce sub-assemblies that are more or less demanding. That is, while the overload is not directly visible when considering the main line in isolation, the overload has instead been pushed to a feeder line, which faces exactly the same product sequencing problem. In fact, car sequencing rules may be defined precisely for this reason, i.e. because of effects that will impact the main line only indirectly because it cannot receive the parts (or rather sub-assemblies) it needs if the sequence overloads a feeder line.

Usually, the main reason for defining car sequencing rules is variation in work content [16], and to ensure that especially demanding variants do not occur too frequently and too close together. In the mixed-model sequencing (MMS) [50] approach, the time needed at stations is used directly to quantify the overload a sequence would yield, while taking potential drift areas into account. The goal is to find feasible sequences that do not cause overloads, or alternatively minimize the amount of overload.

The line balance determines at which stations different variants require more or less work. So the line balance affects which sequences are feasible or not, and so product sequencing is downstream of ALB and must be solved given the current line balance. The product sequence in turn determines when different parts are needed at different stations. It therefore affects logistics, and the sequence must be decided sufficiently far in advance to allow the necessary logistics planning. Product sequencing is therefore performed much more frequently than line balancing and at a shorter time scale, typically on the order of days or weeks before production. This makes product sequencing a short-term operational problem.

Kitting Layout

A kit has to be defined in terms of part types and counts but also the layout, i.e. the order and orientation in which parts are placed in the container or laid out on the wagon. Deciding a particular layout is important because fitting the parts may not be trivial, and some parts should or must be placed in a certain way. For example, placing a part on one of its sides may not be allowed because it would be unstable and fall over or because it would damage the part. The layout also matters because parts must be accessible in an ergonomically sensible manner and in the order they will be needed. The layout may even be mirrored depending on whether the operator is right- or left-handed. Developing a layout in advance ensures that these goals are met during kitting and that the layout will be consistent.

3.2 Motivating Challenges

In the automotive industry, there is often high product variability, and many variants are assembled on the same assembly line. This means that the same station will need varying parts, depending on the product variant currently being assembled. Therefore, there are also many variations of the same kit, possibly differing only in a single part. But manually creating layouts for all variants would be tedious and time-consuming. The variability also complicates assembly line balancing because there is not a single task DAG, and it creates the product sequencing problem because the variation in work content may make some sequences infeasible.

Despite more than 70 years of research, assembly line balancing is still done manually in many industries, such as the automotive industry. That is, the theory is not applied in practice. One challenge is to understand the reasons for this theory-practice gap. Industries that have not adopted techniques for automatic planning presumably have characteristics that make it particularly difficult to apply these techniques. These reasons should shed light on the obstacles to automated planning. Once they have been identified, the next challenge is to find ways of bridging the gap, which should help uncover the possibilities of automation.

The product sequencing problem is almost as old as assembly line balancing. In addition to a theory-practice gap, it also faces slightly different challenges, as it is a more dynamic, operational problem. The line balance is fixed and remains static for a much longer time, on the order of months, but the product sequence changes constantly (e.g. from day to day) depending on the current set of orders. The planned sequence can fail, e.g. due to logistics, and may need to be adjusted at short notice, possibly in near-real-time. These late, short-term changes have to be handled in a cost-effective way and without too much disruption to the current sequence and thereby the current logistics planning. Therefore, solving the sequencing problem and obtaining a suitable sequence is not sufficient. In addition, a fact often ignored in the academic literature is that the assembly line is never empty. Therefore, a sequence does not exist in isolation but rather extends the previous sequence, and the first products in the new sequence must be compatible with the last products in the previous sequence.

CHAPTER 4

Research Questions

The question investigated in the context of the micro-transaction use case is:

Research Question 1

How can a distributed blockchain be designed to achieve horizontal scalability through sharding?

There may be many ways to achieve high transaction throughput, including building a system with the absolute performance to handle the expected workload. However, if the system is not *scalable*, adapting it to cope with greater loads than anticipated may be difficult. A vertically scalable system can be equipped with more powerful hardware to increase its capacity, which has its limits and may not be the most cost-effective solution. In a horizontally scalable distributed system, both its capacity and resilience can be increased by adding more nodes and expanding the network.

One particular strategy for achieving horizontal scalability is sharding. Using sharding to achieve horizontal scalability raises several interrelated challenges. These include how shards are organized and transactions are routed, how cross-shard transactions can be handled while preserving global serializability, and how sharding affects the fault tolerance and security of the overall system. The work in this thesis addresses these challenges to different extents, with the main focus on the fundamental architecture and the cross-shard transaction protocol.

The question investigated in the context of assembly line planning is:

Research Question 2

What are the possibilities and challenges for automating planning problems related to assembly lines?

This question is investigated by considering three different but inherently related planning problems: kitting layout, assembly line balancing, and product sequencing. Kitting layout is an example of a problem where a substantial increase in automation seems feasible. Assembly line balancing provides the main case for investigating why automation is difficult in practice, as evidenced by the well-known theory-practice gap, and this problem is analyzed in great detail. Product sequencing is closely related to assembly line balancing and extends the investigation to shorter-term planning problems in a more dynamic operational setting.

5.1 Paper A – ScaleGraph

Title: *Dynamically Sharded Ledgers on a Distributed Hash Table*

Authors: Christoffer Fink, Olov Schelén, Ulf Bodin

Status: Accepted in *ACM Distributed Ledger Technology: Research and Practice* on 9th of December 2025.

Summary: This paper presents ScaleGraph, a DHT-inspired sharded blockchain system based on a one-transaction-per-block and one-shard-per-account architecture. It shows that a total ordering of transactions is unnecessarily strict, and that a partial order is sufficient. In particular, it is sufficient to serialize transactions on a per-account basis, i.e. by recording them in the ledger for each of the two affected accounts. This induces a partial order on all transactions. While the paper does not emphasize this point, this also makes the transactions globally serializable. Because each block includes one transaction, and that block is identically replicated in the blockchain for each account, each transaction is recorded as a single atomic event. The histories of two interacting accounts intersect at the shared transaction. Including a single transaction per block and defining a shard for each account also means that each block must only be agreed on by the two shards responsible for the two affected accounts, i.e. the sender and receiver. This means that pairs of shards can operate in parallel, thus allowing inter-shard parallelism to compensate for the intra-shard parallelism that is lost when each block contains only one transaction.

The paper extensively discusses the inherent trade-off between scalability and security. Scalability and throughput are improved by dividing nodes into smaller subsets that can process transactions in parallel. However, this also means that an attacker needs to control a smaller number of nodes in order to compromise at least one shard. Simulation experiments are used to explore the relationship between shard size, consensus protocol fault tolerance, and the probability that any shard is compromised. Because ScaleGraph

allows shards to overlap, the failure probability behaves very differently compared to typical sharded blockchains. Shards cannot be modeled as disjoint samples, and so the hypergeometric distribution cannot be used to approximate the failure probability based on the number of shards. However, the paper presents a preliminary model that approximates the failure probability using the hypergeometric distribution combined with an approximation of the effective number of shards, i.e. the number of disjoint subsets. The discussion includes an informal analysis of the expected scaling behavior and predicts two scaling regimes that are examined in more detail in Paper B. While the paper focuses heavily on how a DHT could be used as infrastructure and to define shards based on the DHT distance metric (specifically the XOR metric) between IDs, the core contribution is the one-transaction-per-block and one-shard-per-account architecture that is motivated by the relaxed event ordering.

Personal contribution: The original idea for ScaleGraph was conceived by Olov Schelén several years earlier. My contributions were to refine the concept and write the majority of the paper. I analyzed the scalability-security trade-off and identified constraints related to shard interaction and vote counting, realizing that a separate quorum is needed from each shard. The simulation experiments were designed and carried out by me, and I developed the preliminary mathematical model for approximating the shard compromise probability.

5.2 Paper B – Cross-Shard Commit Protocol

Title: *Cross-Shard Transactions with Inevitable Atomic Commit in Per-Account Sharded Ledgers*

Authors: Christoffer Fink

Status: To be submitted.

Summary: Paper A sketched a cross-shard consensus mechanism but did not fully resolve how transactions are agreed on and atomically committed in both ledgers. This paper fills that gap by proposing a new cross-shard atomic commit protocol that only relies on local, intra-shard consensus and certificate exchange. As part of the certificate exchange, information about the predecessor block in each shard is also exchanged. This allows both shards to implicitly agree on the whole block, not just the transaction. They can independently construct the block and replicate it identically in both account chains, linking it with the predecessor blocks of both chains. That is what makes each transaction an atomic event and preserves global serializability. Based on the assumption of eventual message delivery, the protocol guarantees that a valid transaction will be logged in both ledgers once the sender shard has decided to propose it. That is, atomic commit is inevitable, and the protocol does not require an abort path. A mathematical model is derived for the two scaling regimes predicted in Paper A. This model is supported by preliminary performance experiments that demonstrate the expected scaling behavior. In contrast to Paper A, this protocol is described in much more detail and has been formally verified. This paper also emphasizes the global serializability property that was

only implicit in Paper A and highlights the one-transaction-per-block and one-shard-per-account architecture as the core contribution of Paper A.

Personal contribution: I designed the protocol, wrote the TLA+ specification, implemented the protocol, derived the theoretical model of the two scaling regimes, ran the performance experiments, and wrote the paper.

5.3 Paper C – Layout Planning

Title: *Layout planning in assembly line kitting - a constraint programming approach*

Authors: Christoffer Fink, Wilma Krutrök, Olov Schelén, Ulf Bodin

Status: Published in the proceedings of the *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)* on 30th of November 2021.

Summary: This paper presents a proof-of-concept solution to a concrete, real-world problem posed by an industry partner, namely that of placing parts on kitting wagons for delivery to assembly line stations. A solver is implemented using a constraint solver and optimization framework, and the paper focuses on the potential to trade off development time and solver performance. A number of simple performance optimizations and their effects are presented. We show that, while a straightforward implementation can be developed with minimal effort and time investment, relatively small modifications can drastically shrink the search space and improve performance accordingly. The paper also touches on the concept of Decision Support Systems (DSS).

Personal contribution: Wilma Krutrök and I developed an early prototype of the layout planner. After she left the project, I made extensive changes to the implementation, designed and ran the experiments, and wrote the majority of the paper.

5.4 Paper D – Assembly Line Balancing

Title: *Why Decision Support Systems are needed for addressing the Theory-Practice Gap in Assembly Line Balancing*

Authors: Christoffer Fink, Ulf Bodin, Olov Schelén

Status: Published in *Journal of Manufacturing Systems* on 16th of February 2025.

Summary: This paper thoroughly examines the Assembly Line Balancing Problem and the Theory-Practice Gap. It attempts to explain this phenomenon by analyzing the real-world requirements of the automotive industry and contrasting them with the literature. As part of the explanation, three fundamental challenges are identified. These challenges suggest that complete automation is infeasible in practice, at least in some industries, which implies that some level of manual intervention by human experts is unavoidable. The paper then makes the case for DSS as a way to not only allow the minimally required intervention but to make the human-machine interaction as efficient and effective as possible. By enabling assisted – i.e. partially automated – planning

and supporting an interactive and iterative workflow, automation can be maximized. Furthermore, the solver built into the DSS, as a component that proposes solutions, has to be designed differently to be suited to that context. That is, certain features become possible or necessary. This means that integrating a solver into a DSS is not trivial and cannot be treated as an implementation detail. Therefore, the DSS concept plays a central role in addressing the Theory-Practice Gap in automotive assembly line balancing.

Personal contribution: It was my decision to approach the assembly line balancing problem from the perspective of understanding the theory-practice gap. I studied the literature, identified the three fundamental challenges and the solver requirements, and wrote the paper.

5.5 Paper E – Flexible Termination Criteria

Title: *Improving a SALBP-1 Solver by Tuning Flexible Termination Criteria with Simulated Solving*

Authors: Christoffer Fink, Olov Schelén, Ulf Bodin

Status: Published in the proceedings of the *2025 IEEE 8th International Conference on Industrial Cyber-Physical Systems (ICPS)* on 30th of July 2025.

Summary: The paper defines flexible termination criteria for optimization algorithms and examines their effect on a solver for the simple assembly line balancing problem (SALBP-1)¹. It shows that they can allow time to be used more efficiently when solving problems in batches and, presumably, also iteratively. A suggested time limit can be used as a user-supplied setting to indicate a desired time-quality trade-off. The actual (flexible) termination criteria are then derived from this time limit using parameters that have been chosen based on their effects on a set of problem instances used as training data. The paper also shows that, once a problem instance has been solved with a certain set of termination criteria, the effect of more restrictive criteria can be simulated, rather than requiring the solver to run again with the new criteria. This allows parameters controlling the termination criteria to be tuned efficiently. This paper essentially tests a hypothesis proposed in Paper D, which predicts that these kinds of flexible termination criteria are an important feature of a solver for a DSS. The results support that hypothesis, and the paper connects a technical solver feature to the interactive and iterative workflow associated with DSS-based planning.

Personal contribution: I implemented the original SALBP-1 solver and wrote the software that simulates the effects of different termination criteria to find the parameters for different scenarios and record the corresponding results. I also ran all the experiments and analyzed the data and visualized the results. Finally, I wrote the paper, including the diagrams and visualizations.

¹Specifically the type 1 problem that aims to minimize the number of stations for a given target cycle time.

5.6 Paper F – Product Sequencing

Title: *Work in Progress: Decision Support System for Rescheduling Blocked Orders*

Authors: Christoffer Fink, Olov Schelén, Ulf Bodin

Status: Published in the proceedings of the *2025 IEEE 8th International Conference on Industrial Cyber-Physical Systems (ICPS)* on 30th of July 2025.

Summary: This paper describes the early stages of work on a DSS for product sequencing and resequencing. It focuses on stating the problem and use case and identifying the assumptions and requirements. It also positions the work by identifying novel aspects relative to what has previously been done in the literature. The paper adds another assembly planning problem that has a different time horizon and is more dynamic than assembly line balancing. It also continues the DSS theme.

Personal contribution: I identified the academic framing of the problem, studied the related literature, positioned the work relative to prior research, formulated the assumptions and requirements, and wrote the paper.

CHAPTER 6

Research Methods

The papers included in this thesis address different kinds of questions and therefore rely on different kinds of methods. Together, they include both constructive and explanatory work. Some questions are about how a protocol, solver, or decision support component can be designed. Some are about what properties a design has, for example in terms of correctness, security, or scalability. Some are about how an implementation behaves empirically. Others are about why a theory-practice gap exists, what practical requirements look like, or what kind of decision support is needed. Because these are different kinds of questions, they call for different kinds of methods. This chapter therefore focuses on the concrete methods used across the thesis and on how they relate to the kinds of questions being investigated.

Conceptual analysis [21] and deductive reasoning are used extensively throughout the work. They are particularly important for clarifying problem structure, identifying relevant assumptions and constraints, and reasoning from these to architectural, algorithmic, or practical implications. They also recur across most of the thesis rather than being tied to any one specific kind of study.

This way of organizing the methods is consistent with the observation that different research contributions can arise through different combinations of theory, method, framing, and phenomenon rather than through one single formula [26]. It is also compatible with the view that a conceptual or theoretical contribution depends not only on what is proposed, but also on how and why it is argued to hold, and under what conditions [55]. More broadly, it reflects the methodological breadth of computing and information-systems research, where constructive, analytical, and empirical work often need to be combined rather than treated as mutually exclusive categories [20, 53].

6.1 Constructive, Analytical, and Evaluative Work

One recurring pattern in the thesis is the combination of constructive work with analytical or empirical methods for evaluating the design. This is clearest in the distributed-ledger papers, where the questions concern how a sharded ledger can be structured, how cross-

shard transactions can be handled, and what follows from those design choices in terms of correctness, security, and scalability.

Paper A approaches these questions through architecture design, analytical reasoning, mathematical modeling, and simulation-based experiments. It reasons about transaction ordering and what sort of ordering is actually necessary, analyzes security implications of sharding, and studies how shard failure probability depends on parameters such as shard size and Byzantine fraction. These latter questions motivate the use of mathematical modeling and Monte Carlo simulation. The simulation is used to empirically estimate the shard failure probability. These results can then be used to investigate to what extent standard approximations developed for more conventional sharding designs remain applicable to the proposed architecture.

Paper B continues the same research thread but places greater weight on correctness. It develops proof-style arguments for inevitable atomic commit and global serializability, formalizes the protocol in TLA+ [33], and uses model checking [57] to establish safety and liveness properties. It also derives mathematical scaling models and compares them to prototype-based experiments. The design work is therefore combined with formalization, analysis, and empirical validation more tightly in Paper B than in Paper A.

The same broader pattern also appears in the optimization-oriented papers, although the technical objects are different. Instead of distributed protocols, these papers develop and evaluate solver-based support for practical planning problems. Paper C centers on how the kitting layout problem can be formulated and solved in a way that supports an engineer-facing decision loop. This makes problem formulation and optimization modeling central: the problem must be represented in terms of planning variables, hard constraints, and soft preferences before anything can be computed. The resulting proof-of-concept layout planner is then implemented using an optimization framework, and several solver configurations are compared experimentally. These experiments are used both to assess the effect of specific optimizations, such as search-space reduction and improved initialization, and to evaluate the broader claim that a working solver can be developed with minimal time investment and then incrementally refined to achieve substantial performance improvements.

Paper E also combines constructive and evaluative work, but with a narrower focus. Here the question is how solver termination should be controlled in a decision-support context, in particular whether more flexible termination criteria can make better use of time than a rigid time limit when solving is iterative and interactive. That question leads naturally to comparative computational experiments, in which solver variants based on different termination criteria are evaluated against each other. At the same time, the paper introduces a distinction between productive and unproductive solving time, which gives the evaluation a more explicit conceptual framing and connects the comparison to the specific hypothesis about the importance of flexible termination criteria in the decision-support context.

These constructive and evaluative parts of the thesis fit well with broader accounts of computing research as methodologically plural and artifact-oriented [20, 31, 19]. They also align with established views of experimental validation in software and computing

research, where artifacts are not only proposed but examined through controlled comparisons, simulations, or other empirical means [58].

6.2 Synthesis, Requirements, and Conceptual Framing

Another recurring pattern is that some questions cannot be answered by implementing and evaluating a technical solution experimentally. If the question is why a theory-practice gap exists, or what kind of decision support is needed in a real industrial setting, then the first task is instead to understand the domain, the practical requirements, and the relevant literature.

Paper D is primarily analytical and synthetic. Its main question is why assembly line balancing research has had limited practical impact in industry. It analyzes the characteristics of real-world final assembly in order to understand the domain and its practical requirements. It also studies the line balancing literature and compares it to those requirements, identifying mismatches between the assumptions common in the literature and the conditions that hold in practice. This combines analysis of prior research with domain and requirements analysis. The result is not simply a summary of previous work, but an explanation of the theory-practice gap and an argument that, in some important settings, complete automation is unlikely to be feasible and assisted planning is therefore more appropriate. In the terminology of the review-method literature, this is closer to an integrative and theory-building form of synthesis than to a narrowly systematic review [48, 52, 17].

Paper F plays a related but more exploratory role. Its question is what a decision support system for sequencing and resequencing should look like in the specific case of blocked orders that must jump out of, and later jump back into, a mixed-model product sequence. The work is therefore about defining the problem space rather than presenting a finished solution to be evaluated. It identifies the relevant strands of sequencing research, clarifies what has and has not been addressed in the literature, formulates the assumptions that make a mixed-model-sequencing approach feasible, and outlines both short-term proof-of-concept goals and longer-term research directions. The paper thus contributes partly by framing the problem and delimiting a feasible direction for subsequent technical work.

For these two papers, the literature on conceptual work is particularly relevant. Conceptual papers can have explicit research designs of their own rather than being treated as loosely structured non-empirical essays [21]. In the present thesis, Papers D and F identify the practical problem structures that later technical work must address.

6.3 Relations Between Methods and Question Types

Another useful way to summarize the methods is to relate them to the kinds of questions the papers ask. When the question is how a technical solution can be devised for

a particular problem, the answer tends to involve constructive design work at the level of architecture, protocols, solvers, or system concepts. When the question is whether a design has a claimed property, the answer tends to involve proof-style reasoning, formal modeling, or model checking. When the question is how performance or security depends on parameters, the answer tends to involve mathematical modeling together with controlled computational experiments or simulation. When the question is how a practical industrial planning problem can be represented and solved, the answer tends to involve explicit optimization modeling, implementation, and comparative evaluation. When the question is why theory has transferred poorly into practice, and what implications a plausible explanation has, the answer tends to involve literature synthesis, requirements analysis, and conceptual argumentation.

6.4 Summary

This chapter has described the main method families used across the thesis and how they relate to the different questions addressed in the included papers. These include design of technical solutions, formal and semi-formal analytical reasoning, formal specification and model checking, mathematical modeling and analysis, computational experiments and simulation, optimization modeling, and literature-based synthesis together with domain and requirements analysis.

Table 6.1 summarizes these method families, the purposes they serve in this thesis, and the papers in which they are most clearly represented.

Table 6.1: Main method families used in the thesis.

Methods	Purpose	Papers
Design of technical solutions	Proposing a concrete technical solution to a defined problem, such as a ledger architecture, a cross-shard protocol, or a solver component.	A, B, C, E, F
Formal and semi-formal analytical reasoning	Showing what follows from assumptions and design choices, for example why a protocol preserves a property or why a certain problem framing is appropriate.	A, B, D
Formal specification and model checking	Mechanically checking safety and liveness properties of a protocol specification.	B
Mathematical modeling and analysis	Deriving relationships between parameters, such as failure probability or throughput, and identifying scaling regimes or useful approximations.	A, B
Computational experiments and simulation	Evaluating how a design behaves empirically, comparing alternatives, or validating theoretical expectations.	A, B, C, E
Problem formulation and optimization modeling	Translating a practical planning problem into variables, constraints, objectives, and solver representations.	C, E
Literature-based analytical synthesis	Synthesizing prior work in order to identify patterns, mismatches, and implications, rather than merely summarizing it.	D, F
Domain and requirements analysis	Characterizing the real-world problem that must be addressed, including practical constraints and assumptions.	D, F
Conceptual system design and research scoping	Defining a decision support concept, identifying necessary features, and outlining a direction for subsequent technical work.	D, F

Conclusions, Discussion, and Future Work

7.1 Conclusions

The first question was **how a distributed blockchain can be designed to achieve horizontal scalability through sharding**.

The architecture proposed for ScaleGraph and presented in paper A is based on relaxing the ordering of logged transactions from a total order to a partial order. This relaxation allows parallel execution of any number of transactions between disjoint pairs of accounts, limited by the number of pairs of accounts and thus the total number of accounts. Because it would be possible to associate multiple accounts with each client, possibly by transparently dividing a main account into multiple sub-accounts behind the scenes, the account limit can effectively be removed, allowing the potential parallelism to be arbitrarily high. However, the real limit on throughput is then the number of nodes and the per-node bandwidth. As paper B shows both theoretically and empirically, when the number of accounts is sufficiently large, the total throughput scales linearly with the number of nodes and with the per-node bandwidth. We therefore have strong evidence that the one-TX-per-block and one-shard-per-account architecture proposed in paper A together with the cross-shard commit protocol proposed in paper B can achieve a sharded distributed blockchain exhibiting not only horizontal but also vertical scalability.

Considering the two scaling regimes discussed in both papers (i.e. linear scaling as a function of the number of accounts when the number of accounts is small and as a function of the number of nodes when the number of accounts is large) is interesting from a theoretical perspective. However, in practice, it would be expected that the number of accounts and the number of nodes increase together. This would be the case in an open, permissionless system where the popularity of the system attracts more users (i.e. accounts) some fraction of which are interested in running validator nodes. It would also be the case in a closed, permissioned system where nodes are added specifically in order

to handle a growing user base, i.e. number of accounts. And, as the results presented in paper B show, total throughput scales linearly when increasing both accounts and nodes together in a fixed ratio. Furthermore, paper B also shows that this linear horizontal scaling can be achieved while preserving global serializability.

The second question concerned the **possibilities and challenges for automating planning problems related to assembly lines**.

This thesis has demonstrated one example of the possibility to increase automation in industrial planning problems by solving part of the kitting layout problem. The analysis of the assembly line balancing problem reveals some of the challenges and limitations to automation. This is an example of a planning problem that is infeasible to fully automate in practice. For such complex problems, the main challenges are modeling the problem in sufficient detail, gathering the necessary data, and defining the objective function. However, this only implies limits on *complete* automation, which does not mean that automation cannot be increased. Given the inevitable need for human intervention, the most viable option for increasing automation for such problems is *assisted planning*, i.e. *partial* automation. Focusing more on decision support systems, rather than standalone solvers, may therefore be a more promising approach to bridging the theory-practice gap.

The work on product sequencing included in the thesis (paper F) is preliminary and therefore does not support equally strong conclusions. However, the next section discusses insights based on subsequent and ongoing work that suggests challenges that parallel those found in assembly line balancing.

7.2 Discussion and Future Work

ScaleGraph was presented as a permissioned system. This makes it partially centralized, which is worth elaborating on. We assume that the system would likely be deployed by a consortium, which means that authority and control over the system is shared among multiple institutions or organizations, making the system partially decentralized. Another important point is that no central component is involved in validating transactions. It is only needed to assign an identity and grant permission to join the system. It should also be possible to adapt ScaleGraph to a fully open and permissionless setting if desired.

A legitimate question is why Byzantine-fault tolerance is required in a permissioned and partially centralized system. First, there is always the possibility that some nodes could be compromised by an attacker even if the authority that is nominally in control of it is honest. Second, malicious behavior is only a subset of Byzantine behavior, and Byzantine-fault tolerance also includes other kinds of misbehavior and malfunctions. For robustness more generally, rather than for security in particular, fault tolerance and redundancy therefore remain important. However, a permissioned system with controlled participation, where malicious or misconfigured nodes are rare, drastically reduces the required shard size and allows the system to be much more efficient.

One could also take a step back from the micro-transactions use case specifically and consider a widely adopted payment system more generally, such as a digital currency

issued by a central bank. Such an infrastructure would still have to support large transaction volumes, similar to traditional centralized payment systems, and it would have to be distributed and robust enough to continue working under extreme conditions, such as natural disasters or war.

Paper B experimentally evaluates the scalability of the cross-shard transaction commit protocol for ScaleGraph using a simulated network. A complete implementation of ScaleGraph still remains for future work. Future work also includes a direct performance comparison of such an implementation with existing sharded blockchain systems in realistic scenarios, i.e. using multiple nodes on a real network, and with representative workloads.

Turning from distributed ledgers to assembly line planning, the work on product sequencing presented in paper F is preliminary. This work is still ongoing but has progressed substantially and has revealed similar challenges compared to assembly line balancing. Modeling the situation in sufficient detail turns out to be difficult, mainly due to the lack of sufficiently detailed and reliable data that such a detailed model would require. Car sequencing (CS) rules are a way to coarse-grain these details and model limitations that cannot be directly and explicitly captured by the model. Just as we have seen with assembly line balancing, this results in manually taking known limitations not present in the model into account, both in the definition of the rules but also in the common practice of manually overriding the rules based on domain knowledge and experience.

Similarly to the task precedence graph in assembly line balancing, the most basic data needed for time-based mixed-model sequencing (MMS) are the times required by different model variants at each station. Just as the lack of precedence graphs limits the applicability of automated assembly line balancing, the lack of timing information limits the applicability of automated time-based product sequencing.

Another parallel is the need to make trade-offs in the case of late resequencing. In that case, there are three goals to balance: re-insert a removed product into the sequence as early as possible, minimize disruptions to the rest of the sequence, and produce the best¹ resulting sequence. Similarly to the difficulty in precisely specifying the objective function in assembly line balancing to capture trade-offs between different conflicting criteria, this trade-off is also difficult to specify precisely in advance and as a general rule, since it depends on timing, features of the products, and the possible outcomes, such as a line stoppage or an empty sequence slot. Therefore, the decision-support perspective is again needed so that multiple reasonable options can be presented to an expert decision-maker who can evaluate the options and choose an appropriate trade-off.

In addition to integrating optimization algorithms with decision support systems, a promising future direction to explore is solving the bootstrapping problem in terms of data and how to transition from rule-based CS to time-based MMS. That is, similarly to how past feasible line balances reveal information about the precedence graph, past feasible product sequences ought to reveal information about the time required by different models at each station. The CS rules themselves should also allow timing information to be at least approximately inferred. Assuming that issues caused by problematic se-

¹And here “best” is already itself defined in terms of multiple objectives to trade off.

quences have been recorded, this would mean that information about known infeasible sequences can also be extracted and used to refine rules and timing information. Finding the right way to combine limited timing information, existing CS rules, and past (feasible and/or infeasible) sequences may be a crucial step to start addressing the data availability problem.

Another likely path forward toward solving the data availability problem, both for assembly line balancing and product sequencing, is increased reliance on sensors and other forms of automatic data collection, as opposed to manual data entry.

The main motivation given for increasing automation in assembly line planning was efficiency in terms of finding better solutions compared to manual planning, and/or finding solutions faster. A less tangible benefit may be a reduction in implicit knowledge. Since planning is done manually, there is little incentive to explicitly represent the knowledge needed to carry out these planning tasks. Rejecting the feasibility of complete automation means that human planners cannot be replaced, and the goal is instead to empower them to work more efficiently and effectively. However, creating the tools to support this work would require a great deal of the necessary knowledge to be codified. On the one hand, doing so is one of the challenges to overcome. On the other hand, doing so would reduce the amount of implicit knowledge and make the organization less fragile; less reliant on a small number of individuals.

The work on assembly line balancing presented in this thesis was primarily conceptual and theoretical. It identified requirements on solvers that are to be incorporated into a DSS, and also outlined certain features the DSS itself should have. Paper E investigated the impact of one of these features directly, namely flexible termination criteria, and found strong support for their significance. Other proposed features should be similarly investigated, ideally by constructing and evaluating a DSS for assembly line balancing.

Paper D focused on the difficulties in adopting typical assembly line balancing theory in practical applications. For example, modeling the real-world processes and workflows is difficult because of their complexity relative to idealized and simplified models. While the paper suggests DSS as a way to cope with these difficulties by enabling effective manual intervention, an interesting question is to what extent practice could instead be adapted to theory. That is, would it be possible to obtain a net benefit by modifying current practices such that existing theory could be more easily applied? Perhaps the theory-practice gap could be bridged from both directions.

REFERENCES

- [1] Simon Altemeier, Marcel Helmdach, Achim Koberstein, and Wilhelm Dangelmaier. Reconfiguration of assembly lines under the influence of high product variety in the automotive industry—a decision support system. *International Journal of Production Research*, 48(21):6235–6256, 2010.
- [2] Adam Back et al. Hashcash—a denial of service counter-measure, 2002.
- [3] J. J. BARTHOLDI. Balancing two-sided assembly lines: a case study. *International Journal of Production Research*, 31(10):2447–2461, 1993.
- [4] Olga Battaïa, Xavier Delorme, Alexandre Dolgui, Johannes Hagemann, Anika Horlemann, Sergey Kovalev, and Sergey Malyutin. Workforce minimization for a mixed-model assembly line in the automotive industry. *International Journal of Production Economics*, 170:489–500, 2015. Current Research Issues in Production Economics.
- [5] Christian Becker and Armin Scholl. Balancing assembly lines with variable parallel workplaces: Problem definition and effective solution procedure. *European Journal of Operational Research*, 199(2):359–374, 2009.
- [6] Nils Boysen, Malte Fliedner, and Armin Scholl. Sequencing mixed-model assembly lines: Survey, classification and model critique. *European Journal of Operational Research*, 192(2):349–373, 2009.
- [7] Yavuz A Bozer and Leon F McGinnis. Kitting versus line stocking: A conceptual framework and a descriptive model. *International Journal of Production Economics*, 28(1):1–19, 1992.
- [8] Vitalik Buterin et al. Ethereum white paper. *GitHub repository*, 1:22–23, 2013.
- [9] Vitalik Buterin, Diego Hernandez, Thor Kamphofner, Khiem Pham, Zhi Qiao, Danny Ryan, Juhyeok Sin, Ying Wang, and Yan X Zhang. Combining ghost and casper. *arXiv preprint arXiv:2003.03052*, 2020.
- [10] F. Nava C. Merengo and A. Pozzetti. Balancing and sequencing manual mixed-model assembly lines. *International Journal of Production Research*, 37(12):2835–2860, 1999.

-
- [11] John C Carter and Fred N Silverman. A cost-effective approach to stochastic line balancing with off-line repairs. *Journal of Operations Management*, 4(2):145–157, 1984.
- [12] David Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology: Proceedings of Crypto 82*, pages 199–203. Springer, 1983.
- [13] Tolga Çimen, Adil Baykasoglu, and Sebnem Demirkol Akyol. A detailed review and analysis of assembly line rebalancing problems. *Assembly Automation*, 42(6):742–760, 2022.
- [14] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, et al. On scaling decentralized blockchains: (a position paper). In *International conference on financial cryptography and data security*, pages 106–125. Springer, 2016.
- [15] Sarada Prasad Gochhayat, Sachin Shetty, Ravi Mukkamala, Peter Foytik, Georges A Kamhoua, and Laurent Njilla. Measuring decentrality in blockchain based systems. *IEEE Access*, 8:178372–178390, 2020.
- [16] Uli Golle, Franz Rothlauf, and Nils Boysen. Car sequencing versus mixed-model sequencing: A computational study. *European Journal of Operational Research*, 237(1):50–61, 2014.
- [17] Maria J. Grant and Andrew Booth. A typology of reviews: an analysis of 14 review types and associated methodologies. *Health Information & Libraries Journal*, 26(2):91–108, 2009.
- [18] Abdelatif Hafid, Abdelhakim Senhaji Hafid, and Mustapha Samih. Scaling blockchains: A comprehensive survey. *IEEE access*, 8:125244–125262, 2020.
- [19] Alan R Hevner, Salvatore T March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS quarterly*, pages 75–105, 2004.
- [20] Hilary J Holz, Anne Applin, Bruria Haberman, Donald Joyce, Helen Purchase, and Catherine Reed. Research methods in computing: What are they, and how should we teach them? In *Working group reports on ITiCSE on Innovation and technology in computer science education*, pages 96–114. 2006.
- [21] Elina Jaakkola. Designing conceptual articles: four approaches. *AMS review*, 10(1):18–26, 2020.
- [22] Archana Jain, Chinmay Jain, and Karolina Krystyniak. Blockchain transaction fee and ethereum merge. *Finance Research Letters*, 58:104507, 2023.
- [23] Ernest Koenigsberg. Production lines and internal storage—a review. *Management Science*, 5(4):410–433, 1959.

- [24] Ibrahim Kucukkoc, Zixiang Li, Aslan D. Karaoglan, and David Z. Zhang. Balancing of mixed-model two-sided assembly lines with underground workstations: A mathematical model and ant colony optimization algorithm. *International Journal of Production Economics*, 205:228–243, 2018.
- [25] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.
- [26] Dorothy E Leidner. What’s in a contribution? *Journal of the Association for Information Systems*, 21(1):2, 2020.
- [27] Colin LeMahieu. Nano: A feeless distributed cryptocurrency network. *White paper*, 16:17, 2018.
- [28] Ximmeng Liu, Haomeng Xie, Zheng Yan, and Xueqin Liang. A survey on blockchain sharding. *ISA transactions*, 141:30–43, 2023.
- [29] Thiago Cantos Lopes, Adalberto Sato Michels, Celso Gustavo Stall Sikora, and Leandro Magatão. Balancing and cyclical scheduling of asynchronous mixed-model assembly lines with parallel stations. *Journal of Manufacturing Systems*, 50:193–200, 2019.
- [30] Thiago Cantos Lopes, Giuliano Vidal Pastre, Adalberto Sato Michels, and Leandro Magatão. Flexible multi-manned assembly line balancing problem: Model, heuristic procedure, and lower bounds for line length minimization. *Omega*, 95:102063, 2020.
- [31] Salvatore T March and Gerald F Smith. Design and natural science research on information technology. *Decision support systems*, 15(4):251–266, 1995.
- [32] Walter Mayrhofer, Lothar März, and Wilfried Sihm. Simulation-based optimization of personnel assignment planning in sequenced commercial vehicle assembly: A software tool for iterative improvement. *Journal of Manufacturing Systems*, 32(3):423–428, 2013. Assembly Technologies and Systems.
- [33] Stephan Merz. The specification language tla+. In *Logics of specification languages*, pages 401–451. Springer, 2007.
- [34] GJ Miltenburg and Jacob Wijngaard. The u-line line balancing problem. *Management science*, 40(10):1378–1388, 1994.
- [35] John Miltenburg. Level schedules for mixed-model assembly lines in just-in-time production systems. *Management Science*, 35(2):192–207, 1989.
- [36] Bahman Naderi, Ahmed Azab, and Katayoun Borooshan. A realistic multi-manned five-sided mixed-model assembly line balancing and scheduling problem with moving workers and limited workspace. *International Journal of Production Research*, 57(3):643–661, 2019.

- [37] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, 2008.
- [38] Bruce D Parrello, Waldo C Kabat, and Larry Vos. Job-shop scheduling using automated reasoning: A case study of the car-sequencing problem. *Journal of Automated reasoning*, 2:1–42, 1986.
- [39] Giuseppe Antonio Pierro and Roberto Tonelli. Can solana be the solution to the blockchain scalability problem? In *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 1219–1226. IEEE, 2022.
- [40] PETER PINTO, DAVID G. DANNENBRING, and BASHEER M. KHUMAWALA. A branch and bound algorithm for assembly line balancing with paralleling. *International Journal of Production Research*, 13(2):183–196, 1975.
- [41] Suporn Pongnumkul, Chaiyaphum Siripanpornchana, and Suttipong Thajchayapong. Performance analysis of private blockchain platforms in varying workloads. In *2017 26th international conference on computer communication and networks (ICCCN)*, pages 1–6. IEEE, 2017.
- [42] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments, 2016.
- [43] Serguei Popov. The tangle. *White paper*, 1(3):30, 2018.
- [44] Vishnu Prasad Ranganthan, Ram Dantu, Aditya Paul, Paula Mears, and Kirill Morozov. A decentralized marketplace application on the ethereum blockchain. In *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, pages 90–97. IEEE, 2018.
- [45] Victor Shoup. Proof of history: What is it good for, 2022.
- [46] Fred N Silverman and John C Carter. A cost-based methodology for stochastic line balancing with intermittent line stoppages. *Management Science*, 32(4):455–463, 1986.
- [47] Jakub Sliwinski, Quentin Kniep, Roger Wattenhofer, and Fabian Schaich. Halting the solana blockchain with epsilon stake. In *Proceedings of the 25th international conference on distributed computing and networking*, pages 45–54, 2024.
- [48] Hannah Snyder. Literature review as a research methodology: An overview and guidelines. *Journal of business research*, 104:333–339, 2019.
- [49] Horst Tempelmeier. Practical considerations in the optimization of flow production systems. *International Journal of Production Research*, 41(1):149–170, 2003.
- [50] Nick T. Thomopoulos. Line balancing-sequencing for mixed-model assembly. *Management Science*, 14(2):B59–B75, 1967.

-
- [51] Lorenzo Tiacci. Simultaneous balancing and buffer allocation decisions for the design of mixed-model assembly lines with parallel workstations and stochastic task times. *International Journal of Production Economics*, 162:201–215, 2015.
- [52] Richard J Torraco. Writing integrative literature reviews: Guidelines and examples. *Human resource development review*, 4(3):356–367, 2005.
- [53] Douglas R Vogel and James C Wetherbe. Mis research: a profile of leading journals and universities. *ACM SIGMIS Database: the DATABASE for Advances in Information Systems*, 16(1):3–14, 1984.
- [54] Harald Vranken. Sustainability of bitcoin and blockchains. *Current opinion in environmental sustainability*, 28:1–9, 2017.
- [55] David A Whetten. What constitutes a theoretical contribution? *Academy of management review*, 14(4):490–495, 1989.
- [56] Anatoly Yakovenko. Solana: A new architecture for a high performance blockchain v0.8.13. *Whitepaper*, 2018.
- [57] Yuan Yu, Panagiotis Manolios, and Leslie Lamport. Model checking tla+ specifications. In *Advanced research working conference on correct hardware design and verification methods*, pages 54–66. Springer, 1999.
- [58] Marvin V Zelkowitz and Dolores Wallace. Experimental validation in software engineering. *Information and Software Technology*, 39(11):735–743, 1997.

Part II

Department of Computer Science, Electrical and Space Engineering
Division of EISLAB

ISSN 1402-1544
ISBN 978-91-8142-040-1 (print)
ISBN 978-91-8142-040-8 (pdf)

Luleå University of Technology 2026