

Keywords: feasibility-aware optimisation, computational reuse, synchronised systems, declarative optimisation, metaheuristics, synchronisation-aware architectures

Grzegorz BOCEWICZ ^{1*}, Czesław SMUTNICKI ², Zbigniew BANASZAK ¹

¹ Koszalin University of Technology, Poland, grzegorz.bocewicz@tu.koszalin.pl, zbigniew.banaszak@tu.koszalin.pl

² Wrocław University of Science and Technology, Poland, czeslaw.smutnicki@pwr.edu.pl

* Corresponding author: grzegorz.bocewicz@tu.koszalin.pl

Beyond classical optimisation: Toward feasibility-aware computational architectures for synchronised systems

Abstract

Contemporary logistics, manufacturing, transportation, and cyber-physical systems are increasingly evolving into large-scale synchronised environments in which multiple interdependent resources operate under tightly coupled temporal, structural, and operational constraints. In such settings, the feasibility and correctness of individual decisions are determined not only by local optimisation criteria but also by the ability to preserve synchronisation consistency across distributed processes and dynamically changing system states. As system complexity grows, conventional optimisation approaches often encounter limitations in scalability, computational efficiency, and operational feasibility. Motivated by these challenges, this paper presents a conceptual perspective on the future development of feasibility-aware computational architectures for synchronised multi-resource systems. The proposed vision advocates a multi-layer architectural framework integrating structural feasibility assessment, declarative optimisation models, scalable metaheuristic search mechanisms, and computational reuse strategies. Such an integration aims to support efficient exploration of large decision spaces while preserving synchronisation coherence and operational admissibility. The paper further identifies several open research challenges, including scalability-oriented synchronisation management, feasibility-preserving search processes, systematic exploitation of reusable computational knowledge, and the development of unified benchmarking methodologies for synchronised optimisation environments. Finally, a research agenda is outlined for the design of uncertainty-aware optimisation architectures that provide scalable and robust decision support in increasingly complex and distributed operational systems. The proposed architecture should be understood as a conceptual reference framework and research agenda rather than as a completed and experimentally validated computational system. The STTS example provides an illustrative demonstration of localised feasibility verification and computational reuse, whereas systematic implementation, comparative benchmarking, and scalability validation remain subjects of future research.

1. INTRODUCTION

In recent years, optimisation research has increasingly focused on more sophisticated mathematical models, richer constraint sets, multiple objective functions, and advanced solution methodologies. While these developments have significantly expanded the methodological landscape, they also raise an important question: do increasingly complex optimisation techniques remain suitable for practical deployment in large-scale engineering systems? This paper argues that future progress may depend not only on more powerful search algorithms but also on computational architectures explicitly designed to preserve and exploit structural feasibility and synchronisation knowledge.

1.1. From optimisation-centric toward feasibility-aware systems

Traditional optimisation methods have primarily focused on improving objective-function values within an assumed feasible solution space. In many classical applications, feasibility verification remains relatively localised and computationally inexpensive, allowing optimisation algorithms to concentrate on efficient exploration and solution improvement.

Contemporary synchronised systems challenge this assumption. Transportation-service networks, collaborative robotic systems, cyber-physical production environments, and healthcare logistics platforms operate through tightly coupled temporal and resource dependencies. In such environments, locally feasible decisions may become globally infeasible when synchronisation relationships between the usage of multiple resources are violated. As a result, feasibility can no longer be regarded solely as a preliminary verification step. Instead, it emerges as a structural property of the entire system, determined by temporal continuity, resource compatibility, and synchronisation consistency across interacting processes.

This observation motivates a transition from optimisation-centric reasoning to feasibility-aware computational architectures, in which admissibility assessment becomes an explicit computational layer that guides optimisation and limits exploration to synchronisation-consistent regions of the solution space.

1.2. Synchronisation as a foundation of structural feasibility

The increasing importance of feasibility-aware optimisation is closely related to the growing role of synchronisation in modern engineering systems. Synchronisation has long been recognised as a fundamental mechanism in scheduling theory, constraint programming, manufacturing systems, and transportation science, where operational success depends on maintaining consistent temporal and resource relationships among multiple interacting entities rather than on optimising isolated decisions. This perspective is reflected in classical scheduling theory (Pinedo, 2022) and declarative optimisation and constraint programming (Hooker, 2014; Laborie, 2003; Rossi et al., 2006). Foundational studies on synchronised routing and cyber-physical production systems include (Drexel, 2012; Monostori, 2014). More recent research has extended synchronisation-intensive optimisation to vehicle–crew planning, home healthcare, workforce routing, delivery–installation systems, and other multi-resource environments (Ammann et al., 2023; Coelho & Mancini, 2026; de Aguiar et al., 2023; Erdogan & M’Hallah, 2023; Lucci et al., 2026; Wang et al., 2025).

Representative examples arise across a broad spectrum of engineering domains. In industrial robotics and automated manufacturing, synchronisation coordinates multiple manipulators, machines, conveyors, and buffers executing tightly coupled operations while avoiding conflicts over shared workspaces. Cyclic production systems similarly rely on synchronisation to maintain common takt times and balanced material flows. In Industry 4.0 and cyber-physical production systems, geographically distributed facilities, digital twins, and autonomous controllers must preserve temporal and logical consistency despite decentralised decision making (Lee et al., 2015; Monostori, 2014; Relich et al., 2022).

Comparable principles govern transportation and service operations. Home healthcare routing, technician scheduling, workforce routing with shared vehicles, synchronized delivery-and-installation systems, and truck-drone coordination all require simultaneous coordination of multiple resource classes, including vehicles, personnel, equipment, and service activities (Ammann et al., 2023; Brailsford et al., 2022; Bredström & Rönnqvist, 2008; Drexel, 2012; Erdogan & M’Hallah, 2023; Frifita et al., 2017; Lei et al., 2026; Li et al., 2023; Lucci et al., 2026; Nasir & Kuo, 2020; Purnamasari & Santoso, 2018). Similar synchronisation requirements arise in autonomous mobile robot fleets and AGV systems, where collision-free movement, deadlock prevention, and exclusive occupancy of guideway segments are essential for safe operation.

Despite their diversity, these synchronisation mechanisms can be interpreted through two complementary dimensions. Temporal synchronisation specifies when activities may occur through precedence relations, simultaneous execution requirements, deadlines, and time windows. Resource synchronisation specifies which entities may simultaneously access limited resources through capacity restrictions, mutual exclusion, continuity constraints, and exclusive allocations. Representative synchronisation mechanisms observed in selected engineering domains, together with their implications for structural feasibility, are summarised in Table 1. In practice, these dimensions are inseparable: a schedule that satisfies all temporal constraints may remain infeasible due to resource conflicts, while conflict-free resource allocations may violate essential timing relationships. Such interactions have been extensively investigated within constraint programming and integrated scheduling frameworks (Hooker, 2014; Laborie, 2003; Rossi et al., 2006).

Consequently, feasibility should increasingly be regarded as a structural property emerging from synchronisation relationships among distributed processes rather than as a simple conjunction of local constraints. The principal question is therefore no longer whether individual activities are executable in isolation, but whether all interacting operations can coexist within a jointly admissible system configuration. The Service Team Transport Scheduling (STTS) problem provides an illustrative example of this broader perspective. Unlike conventional synchronised routing formulations that primarily coordinate selected

resources, STTS simultaneously integrates transportation vehicles, autonomous service teams, boarding operations, service execution, retrieval operations, and time-window constraints under dedicated or flexible dispatch strategies. In this setting, mission feasibility depends on preserving synchronisation across the complete deployment–service–retrieval cycle rather than merely satisfying routing or scheduling constraints. As a consequence, synchronisation becomes not only an execution mechanism but also the principal determinant of structural feasibility.

A complementary perspective on synchronisation emerges from systems exposed to collisions, blocking, and deadlocks. While synchronised routing and service scheduling studies usually emphasise coordinated execution of distributed tasks, particularly in home healthcare, workforce routing, vehicle–crew planning, flight crew assignment, and delivery–installation systems (Arikan et al., 2016; Bredström & Rönnqvist, 2008; de Aguiar et al., 2023; Drex1, 2012; Erdogan & M’Hallah, 2023; Lucci et al., 2026; Wang et al., 2025), collision- and deadlock-prone systems focus primarily on safe access to shared resources and the preservation of operational liveness. This research perspective is particularly relevant for automated manufacturing systems, AGV and AMR fleets, robotic workcells, railway traffic, automated warehouses, and cyber-physical production systems, where simultaneous occupation of the same spatial or resource segment may immediately destroy executability (Banaszak & Krogh, 1990; Čáp et al., 2016; Fanti, 2001; Wu & Zhou, 2005).

Tab. 1. Representative synchronisation mechanisms and their implications for structural feasibility

Application domain	Representative synchronisation mechanism	Structural feasibility implication	Representative references
Industrial robotics and automated manufacturing	Coordination of robots, machines and conveyors	Simultaneous satisfaction of precedence and shared-workspace constraints	(Čáp et al., 2015; Monostori, 2014; Pinedo, 2022)
Constraint programming and scheduling	Temporal/resource propagation	Joint satisfaction of precedence and resource constraints	(Ammann et al., 2023; Bocewicz, 2023; Hooker, 2014; Laborie, 2003; Rossi et al., 2006)
Home healthcare and workforce routing	Synchronisation of caregivers, vehicles and services	Feasible joint execution requires the coordinated arrival of heterogeneous resources	(Brailsford et al., 2022; Bredström & Rönnqvist, 2008; de Aguiar et al., 2023; Frifita et al., 2017; Nasir & Kuo, 2020; Wang et al., 2025)
Vehicle routing with synchronisation	Coordination of multiple interacting vehicles or crews	Feasibility depends on preserving synchronisation relationships across routes	(Drex1, 2012; Purnamasari & Santoso, 2018; Sharon et al., 2015; Wagner & Choset, 2015)
Delivery–installation and vehicle–crew planning	Coupled transport and service execution	Transportation and service activities must remain jointly admissible	(Arikan et al., 2016; Erdogan & M’Hallah, 2023; Lei et al., 2026; Lucci et al., 2026)
Cyber-physical and Industry 4.0 systems	Synchronisation of distributed autonomous components	Structural consistency becomes a prerequisite for scalable optimisation	(Lee et al., 2015; Monostori, 2014)
Service Team Transport Scheduling (STTS)	Integrated synchronisation of vehicles, service teams, TD–TW–TB operations	Structural feasibility emerges from maintaining synchronisation across transportation and service layers	(Radzki, Bocewicz & Banaszak, 2026; Radzki, Bocewicz, Jasiulewicz-Kaczmarek, et al., 2026; Radzki, Bocewicz, Nielsen, et al., 2026; Radzki et al., 2025)

In these systems, synchronisation is not limited to temporal coordination. It also acts as a mechanism of mutual exclusion, conflict avoidance, and liveness preservation. Constraint programming and resource propagation frameworks provide a natural formal background for this interpretation, since they explicitly combine temporal reasoning with resource-capacity and mutual-exclusion constraints (Hooker, 2014;

Laborie, 2003; Rossi et al., 2006). Similarly, cyber-physical production research emphasises that distributed autonomous components must preserve logical, temporal, and resource consistency in order to maintain safe and dependable operation (Lee et al., 2015; Monostori, 2014).

From this paper's perspective, these two research streams should not be treated as separate. Service-oriented synchronisation addresses the coordinated execution of activities by vehicles, crews, technicians, caregivers, or service teams (Radzki, Bocewicz, Jasiulewicz-Kaczmarek, et al., 2026; Radzki, Bocewicz, Nielsen, et al., 2026; Radzki et al., 2025), whereas safety-oriented synchronisation addresses collision avoidance, deadlock prevention, and liveness maintenance (Bocewicz, 2023; Čáp et al., 2016; Fanti, 2001; Wu & Zhou, 2005). Both streams ultimately concern the same structural issue: the ability of multiple interacting processes to coexist within a jointly executable configuration.

The Service Team Transport Scheduling (STTS) problem provides a useful bridge between these perspectives. On the one hand, it belongs to the family of synchronised routing and workforce/service scheduling problems, because vehicles, service teams, boarding operations, service execution, and retrieval operations must be coordinated under time-window constraints. On the other hand, its graph-based feasibility assessment reflects a safety-oriented logic, since conflicting operations are represented explicitly and structurally inadmissible configurations can be detected before optimisation begins (Radzki, Bocewicz, & Banaszak, 2026; Radzki, Bocewicz, Jasiulewicz-Kaczmarek, et al., 2026; Radzki, Bocewicz, Nielsen, et al., 2026; Radzki et al., 2025). Thus, STTS illustrates how synchronisation may evolve from a local scheduling constraint into a structural feasibility supervision mechanism. Table 2 summarises this positioning.

Viewed from this perspective, STTS links two previously weakly connected research streams: synchronisation for coordinated service execution and synchronisation for collision-free, deadlock-free system operation. This unifying viewpoint supports the paper's central claim: that structural feasibility should be treated as a first-class computational object and as a foundation for feasibility-aware computational architectures.

Tab. 2. Positioning of synchronisation in service-oriented and safety-oriented systems

Research stream	Main synchronisation objective	Representative systems	Feasibility interpretation	Representative references
Service-oriented synchronization	Coordinate vehicles, crews, technicians, caregivers, or service teams	Home healthcare, workforce routing, vehicle–crew scheduling, delivery–installation systems	Feasibility depends on the coordinated execution of distributed services	(Bredström & Rönnqvist, 2008; Drex1, 2012; Erdogan & M'Hallah, 2023; Lucci et al., 2026)
Safety-oriented synchronization	Prevent collisions, blocking, deadlocks, and unsafe resource occupation	AGV/AMR fleets, robotic cells, automated warehouses, railway and CPS systems	Feasibility depends on conflict-free and live system evolution	(Hooker, 2014; Laborie, 2003; Lee et al., 2015; Monostori, 2014; Rossi et al., 2006)
STTS as a bridging case	Coordinate service execution while excluding structurally inadmissible configurations	Mobile service teams transported by dedicated or flexible vehicles	Feasibility becomes a graph-supported structural property of the whole mission	(Radzki, Bocewicz, & Banaszak, 2026; Radzki, Bocewicz, Jasiulewicz-Kaczmarek, et al., 2026; Radzki, Bocewicz, Nielsen, et al., 2026; Radzki et al., 2025)

1.3. Scalability crisis in synchronised optimisation

The growing scale of synchronised systems exposes a fundamental limitation of contemporary optimisation approaches. While modern algorithms are increasingly effective at generating candidate solutions, maintaining synchronisation consistency often becomes the dominant computational challenge. In strongly coupled environments, even minor local modifications may propagate through multiple operational layers. A small delay in one transportation activity, for example, may affect service schedules, resource allocations, and temporal continuity constraints across the entire system. Similar cascading synchronisation effects have been

reported in vehicle-routing problems with synchronisation constraints, home healthcare scheduling, and cyber-physical production systems (Bredström & Rönnqvist, 2008; Drex1, 2012; Lee et al., 2015; Monostori, 2014). Consequently, a substantial portion of computational effort is frequently devoted to repeated feasibility verification and synchronisation management rather than to optimisation itself. The fragmentation of existing optimisation paradigms reinforces this phenomenon. Declarative approaches provide strong reasoning capabilities through constraint propagation but may encounter scalability limitations in highly interconnected environments (Hooker, 2014; Laborie, 2003; Rossi et al., 2006). Metaheuristics offer efficient large-scale exploration yet often rely on repeated evaluation or reconstruction of solution structures after local modifications (Nowicki & Smutnicki, 1996, 1998, 2005; Rudy et al., 2020; Rudy & Żelazny, 2015; Taillard, 1990). Likewise, graph-based synchronisation and conflict representations have proved effective in identifying admissibility conditions and preventing collisions or deadlocks (Banaszak & Krogh, 1990; Čáp et al., 2015; Čáp et al., 2016; Sharon et al., 2015; Wagner & Choset, 2015; Wu & Zhou, 2005), but are rarely integrated into broader optimisation frameworks supporting computational reuse.

Accordingly, future scalability improvements may derive less from increasingly sophisticated search procedures and more from computational architectures that can preserve, propagate, and reuse synchronisation knowledge throughout the optimisation process. Such a perspective is consistent with recent developments in synchronisation-intensive routing and scheduling, where maintaining structural consistency is becoming as important as optimising objective-function values (Drex1, 2012; Lee et al., 2015; Monostori, 2014; Radzki, Bocewicz, & Banaszak, 2026; Radzki, Bocewicz, Jasiulewicz-Kaczmarek, et al., 2026; Radzki, Bocewicz, Nielsen, et al., 2026; Radzki et al., 2025).

1.4. Distributed manufacturing

Contemporary manufacturing increasingly relies on distributed, multi-factory, and geographically dispersed production systems, in which cooperation among production sites, suppliers, transportation resources, inventories, assembly operations, and customer-oriented delivery processes must be coordinated across multiple organizational and spatial levels (Behnamian & Fatemi Ghomi, 2016; Brzozowska et al., 2023; Chen et al., 2015; Cochran et al., 2001; Fu et al., 2021; Gkiotsalitis et al., 2023; Lohmer & Lasch, 2021; Pour et al., 2019; Saeidlou et al., 2019; Schnetzler et al., 2007). Such systems extend classical manufacturing coordination problems by introducing strong interdependencies among material flows, production schedules, storage decisions, personnel availability, and supply-chain constraints (Fu et al., 2021; Lohmer & Lasch, 2021; Pour et al., 2019; Saeidlou et al., 2019). The need to maintain production continuity amid these interdependencies reflects a long-standing principle in Just-in-Time manufacturing, where temporal coordination and resource availability are essential to avoid disruption. From a modelling perspective, distributed manufacturing systems may be represented either by a single integrated optimisation model or by a decomposed decision structure. A monolithic formulation jointly captures resources, materials, production schedules, transportation links, inventory levels, and synchronisation constraints, but it often leads to models that are computationally too large for practical use (Behnamian & Fatemi Ghomi, 2016; Fu et al., 2021; Lohmer & Lasch, 2021; Saeidlou et al., 2019). Decomposition is therefore attractive because it partitions the overall problem into smaller submodels coordinated through shared variables and interrelated constraints. In two-level or multi-level formulations, lower-level subproblems represent local production, transport, or service decisions, whereas an upper-level coordination mechanism preserves consistency among interacting subsystems (Cochran et al., 2001; Fu et al., 2021; Lohmer & Lasch, 2021; Saeidlou et al., 2019). However, decomposition alone does not eliminate the feasibility problem. Local decisions that are admissible within individual submodels may become globally infeasible when synchronisation requirements between production sites, transportation resources, inventories, assembly operations, or service activities are violated. This observation directly supports the central thesis of the present paper: in synchronisation-intensive distributed manufacturing, scalability depends not only on decomposition and optimisation algorithms but also on architectural mechanisms that can supervise structural feasibility, propagate synchronisation consequences, and reuse previously established admissibility information. Consequently, distributed manufacturing provides a compelling application domain illustrating why future optimisation systems may benefit from architectures equipped with explicit mechanisms for structural-feasibility supervision, synchronisation management, and computational reuse.

1.5. Research hypothesis

The increasing complexity of synchronised multi-resource systems suggests that future scalability may depend less on increasingly sophisticated search algorithms and more on computational architectures that explicitly manage feasibility and synchronisation knowledge. Unlike classical optimisation environments, synchronised systems exhibit strong interdependencies among resources, operations, and temporal processes. Consequently, local modifications may propagate across multiple operational layers, making feasibility verification and synchronisation management major contributors to computational cost.

This paper hypothesises that scalable decision support in such environments requires architectures equipped with explicit structural-feasibility supervision integrating several complementary capabilities.

First, structural feasibility reasoning should identify synchronisation-consistent regions of the solution space before extensive optimisation begins. For example, two transportation tasks may be individually feasible but become jointly infeasible when they require the same vehicle at overlapping times.

Second, synchronisation-aware propagation should transmit the consequences of local changes only through affected synchronisation relationships. A delay of one vehicle, for instance, should trigger verification only of dependent transportation and service operations rather than the entire schedule. Third, localised incremental recomputation should update only those feasibility structures directly affected by a modification. When a single customer assignment changes, recomputation should be limited to the neighbouring transportation and service activities instead of reconstructing the complete solution. Fourth, reusable admissibility structures should preserve previously verified information on synchronisation and feasibility. Conflict graphs, admissibility networks, or validated route fragments can then be incrementally updated rather than repeatedly reconstructed from scratch. Finally, scalable optimisation and exploration processes should exploit the above mechanisms to navigate large solution spaces while avoiding structurally inadmissible regions and unnecessary recomputation.

The central premise of this work is that the combination of these capabilities may transform scalability from a purely algorithmic issue into an architectural property of synchronised decision-support systems. Rather than treating feasibility as a secondary verification step, future optimisation environments may increasingly rely on dedicated feasibility-supervision layers to guide admissible exploration and support computational reuse throughout the optimisation process (Laborie, 2003). In other words, the future scalability of synchronised optimisation systems is primarily an architectural problem rather than an algorithmic one; i.e., scalability is achieved not only through better search algorithms but also through the architectural integration of feasibility supervision, synchronisation management, and computational reuse. Throughout this paper, the term feasibility-aware computational architecture denotes a reference architectural pattern rather than a specific optimisation algorithm or software framework. Its purpose is to organise the interaction between structural-feasibility supervision, synchronisation management, optimisation, and computational reuse in large-scale synchronised systems, thereby guiding exploration toward structurally admissible regions of the solution space.

The objective of this paper is not to introduce a new optimisation algorithm, but to formulate a conceptual research agenda that advocates feasibility-aware computational architectures as a unifying paradigm for scalable synchronised optimisation.

2. FUTURE RESEARCH DIRECTIONS

The discussion presented in the preceding sections suggests that the next generation of synchronised optimisation systems will require a gradual transition from algorithm-centred approaches toward computational architectures equipped with explicit mechanisms for structural-feasibility supervision. Although substantial progress has been achieved in graph-based reasoning, declarative optimisation (Hooker, 2014; Radzki, Bocewicz, Nielsen, et al., 2026), scalable metaheuristics, and computational acceleration, these research streams have largely evolved in isolation. Consequently, contemporary optimisation systems often perform repeated feasibility verification, synchronisation propagation, and solution reconstruction despite the significant structural overlap between neighbouring solution states. This observation motivates a unified research agenda in which structural-feasibility reasoning, synchronisation-aware optimisation, and computational reuse operate as complementary architectural capabilities rather than independent algorithmic

components. The following subsections outline several research directions that may contribute to the development of such feasibility-aware computational architectures.

Recent years have witnessed a growing interest in synchronised systems involving multiple interacting resource classes whose activities must remain coordinated in both space and time (Drexler, 2012; Monostori, 2014). Representative examples include home healthcare routing and scheduling systems (Coelho & Mancini, 2026; Yin et al., 2026), synchronized vehicle–crew planning (Ammann et al., 2023; Lucci et al., 2026), workforce scheduling with vehicle sharing and drop-off/pick-up policies (Bredström & Rönnqvist, 2008; Öztürkoğlu & Özsakallı, 2023), delivery-and-installation systems (Erdoğan & M’Hallah, 2023), truck–drone coordination frameworks (Lei et al., 2026; Xu & Carlsson, 2026), and worker–vehicle synchronization problems (Radzki, Bocewicz, Nielsen, et al., 2026; Radzki et al., 2025). Despite their diversity, these application domains share a common computational characteristic: maintaining synchronisation consistency among heterogeneous resources is often more challenging than optimising the objective function itself. As synchronisation density, resource heterogeneity, and operational uncertainty continue to increase, feasibility-preservation itself becomes an increasingly important computational challenge rather than merely a verification step performed after optimisation.

2.1. Structural feasibility reasoning

One of the most important future research directions concerns the development of explicit structural feasibility layers that can supervise admissibility before large-scale optimisation begins. Structural feasibility denotes the existence of a synchronisation-consistent configuration of operations and resources satisfying temporal, continuity, resource, and coordination constraints. Building on classical constraint-satisfaction and constraint-programming foundations (Hooker, 2014; Laborie, 2003; Rossi et al., 2006), the perspective adopted here places particular emphasis on the joint executability of the complete synchronised system. Graph-based representations appear particularly useful because they make conflict, temporal dependency, and synchronisation structures explicit and can therefore support admissibility assessment and structural feasibility supervision (Banaszak & Krogh, 1990; Čáp et al., 2016; Radzki et al., 2025; Sharon et al., 2015).

Future research should therefore investigate how structural feasibility reasoning can evolve from a static verification procedure into a continuously operating supervisory mechanism capable of maintaining synchronisation consistency in dynamic and uncertainty-affected environments.

2.2. Declarative and constraint-based optimisation

Constraint Programming (Rossi et al., 2006) and related declarative paradigms provide powerful mechanisms for representing synchronisation relationships, resource continuity constraints, and temporal dependencies. Their main strength lies in their ability to reason about system admissibility through propagation. However, synchronised environments are often characterised by strong coupling, meaning that small local modifications can have consequences across multiple interconnected operational layers. A ten-minute delay in one operation may alter vehicle assignments, invalidate subsequent synchronisation relationships, and trigger modifications throughout the system. Future declarative architectures should therefore support dynamic structural modifications while avoiding repeated global propagation. One promising direction involves integrating declarative reasoning with graph-based synchronisation models that limit reasoning activities to structurally affected regions only. Such an approach would transform declarative optimisation from a purely exact optimisation methodology into a synchronisation-aware reasoning engine embedded within a larger feasibility-preserving architecture.

2.3. Scalable metaheuristic exploration

Metaheuristics, including Tabu Search and related neighbourhood-based approaches, remain important exploration mechanisms for large scheduling and synchronised optimisation problems (Erdoğan & M’Hallah, 2023; Nowicki & Smutnicki, 1996, 1998, 2005; Rudy et al., 2020; Smutnicki et al., 2026). However, future research should increasingly focus on reducing the computational overhead of feasibility verification and synchronisation maintenance, rather than solely on designing new search operators.

A particularly promising direction concerns synchronisation-aware propagation, which can be viewed as an intelligent mechanism for transmitting the effects of local changes via synchronisation relationships. For example, if a vehicle is delayed, the system should automatically identify only those service operations,

transportation tasks, and resource allocations directly affected by that delay, rather than re-evaluating the entire solution. Closely related is the concept of graph-supported propagation, in which dependencies are explicitly represented within conflict graphs, admissibility graphs, or synchronisation networks. Instead of performing global verification, changes propagate only along affected graph connections. This idea provides a natural mechanism for improving scalability in large synchronised environments. Another important research direction concerns localised incremental recomputation. Intuitively, a system should update only those solution fragments directly affected by a modification. If a single customer assignment changes, recomputation should be restricted to neighbouring transportation and service activities rather than reconstructing the complete operational schedule.

Taken together, synchronisation-aware propagation and localised incremental recomputation suggest a shift from optimisation frameworks based on repeated global reconstruction toward architectures capable of maintaining feasibility through continuous local updates and computational reuse.

2.4. Research gap

The growing body of literature on synchronised routing and scheduling demonstrates substantial progress in exact optimisation, decomposition approaches, declarative modelling, matheuristics, and scalable metaheuristics (Ammann et al., 2023; Bredström & Rönnqvist, 2008; Coelho & Mancini, 2026; Erdogan & M’Hallah, 2023; Lucci et al., 2026; Öztürkoğlu & Özsakallı, 2023). Existing studies successfully address synchronisation among vehicles, workers, drivers, technicians, drones, and installation teams. Nevertheless, most of these approaches remain optimisation-centric and treat feasibility primarily as a property of a generated solution rather than as an explicit computational object that can be represented, propagated, preserved, and reused during exploration. Consequently, comparatively little attention has been devoted to structural admissibility reasoning, reusable feasibility representations, synchronisation-aware computational reuse, or feasibility-preserving exploration strategies. In most existing approaches, feasibility verification is reconstructed from scratch after every significant modification to the solution state. As synchronised systems become increasingly large, heterogeneous, and uncertainty-affected, such repeated recomputation may itself become one of the dominant scalability bottlenecks.

From this perspective, the STTS problem (Radzki, Bocewicz, Nielsen, et al., 2026; Radzki et al., 2025) may be viewed as an early step toward feasibility-aware synchronised systems. Unlike classical vehicle routing models, workforce scheduling frameworks, synchronised crew-planning approaches, or truck–drone routing systems (Ammann et al., 2023; Bredström & Rönnqvist, 2008; Lucci et al., 2026; Xu & Carlsson, 2026), STTS explicitly separates transportation resources and service teams while introducing synchronisation through boarding, service execution, and retrieval operations. Dedicated and flexible dispatch policies naturally generate multiple synchronisation structures whose feasibility can be investigated independently of subsequent optimisation procedures. As discussed in Section 1.2, STTS exemplifies the broader transition from synchronisation-oriented scheduling to structural feasibility supervision, in which admissibility is determined by the joint consistency of interacting temporal and resource relationships rather than by isolated routing decisions.

More importantly, the graph-based representations associated with STTS (Radzki et al., 2025) suggest that feasibility may be assessed before optimisation begins rather than after candidate solutions have been generated. This perspective extends earlier synchronised routing models by emphasising structural feasibility supervision as a computational layer operating alongside optimisation. While related studies investigate synchronisation among vehicles, workers, drivers, drones, or installation teams (Erdogan & M’Hallah, 2023; Lei et al., 2026; Lucci et al., 2026; Öztürkoğlu & Özsakallı, 2023), none simultaneously combine autonomous service teams, dedicated and flexible dispatch strategies, explicit boarding/disembarking operations, dual resource-layer representations, and graph-based feasibility certification prior to optimisation. Consequently, contemporary research still lacks:

- unified architectures for structural feasibility supervision and optimisation,
- explicit integration of structural feasibility supervision and scalable exploration,
- synchronisation-aware propagation and recomputation mechanisms,
- reusable admissibility layers supporting computational reuse,
- unified benchmarking methodologies for synchronisation-aware optimisation systems.

The conceptual positioning of STTS within the broader landscape of synchronised systems is shown in Figure 1, while the identified research gaps and corresponding future research directions are summarised in Table 3.

Table 3 summarises the principal research gaps identified in synchronised routing and scheduling research and highlights the directions through which feasibility-aware computational architectures may emerge as a distinct research paradigm. Addressing these challenges may require a transition from optimisation-centric frameworks to feasibility-aware computational architectures, in which feasibility supervision, synchronisation management, optimisation, and computational reuse are integrated architectural components rather than independent algorithmic modules. Such a transition may ultimately contribute to the emergence of a new research direction centred on feasibility-aware synchronised systems, in which the primary challenge is no longer solely how to search larger solution spaces, but how to preserve, propagate, and reuse feasibility knowledge throughout the exploration process.

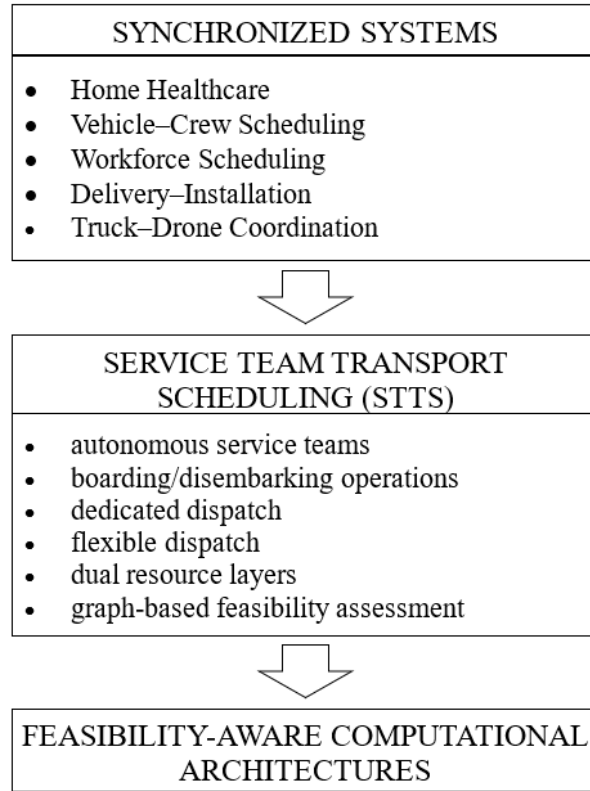


Fig. 1. Positioning of STTS within the broader landscape of synchronised systems and its role as a stepping stone toward architectures integrating explicit structural-feasibility reasoning.

Tab. 3. Research gaps and future research directions

Research Area	Current Limitation	Proposed Direction
Feasibility reasoning	Mostly verification-oriented	Structural feasibility architectures
Metaheuristics	Repeated global recomputation	Incremental synchronization-aware reuse
Declarative optimization	Scalability limitations	Hybrid feasibility-aware architectures
Benchmarking	Runtime-focused evaluation	Feasibility/scalability/reuse benchmarking
CPS synchronization	Weak integration with optimization	Unified synchronization-aware frameworks
Dynamic environments	Expensive global updates	Localized admissibility propagation

3. TOWARD ARCHITECTURES FOR STRUCTURAL FEASIBILITY SUPERVISION

Sections 1 and 2 established why synchronisation transforms feasibility from a local constraint-verification issue into a system-level structural property. Section 3, therefore, focuses on the architectural consequences of this observation rather than repeating the underlying motivation. Figure 2 summarises the proposed transition toward a multi-layer feasibility-aware architecture, while Table 4 positions this architecture relative to classical optimisation, declarative approaches, and metaheuristics.

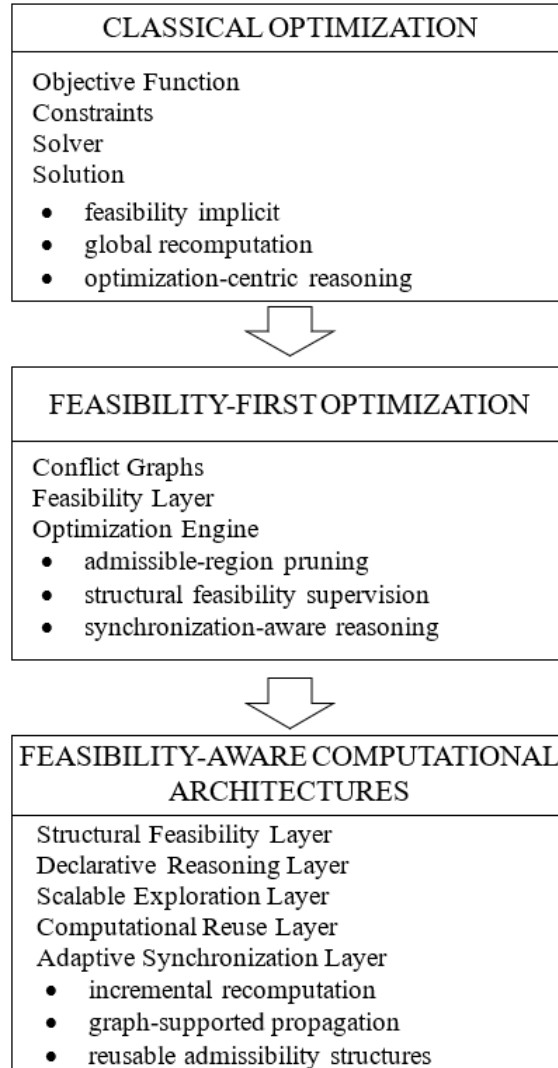


Fig. 2. Conceptual multi-layer transition from optimisation-centric paradigms toward feasibility-aware and synchronisation-oriented computational architectures

The originality claimed in this paper does not reside in constraint propagation, decomposition, dynamic graph updating, incremental evaluation, warm starts, neighbourhood evaluation, or local-search acceleration considered individually. These mechanisms are established within their respective research areas. The proposed contribution instead lies in organising such mechanisms around an explicit structural-feasibility supervision layer and treating previously established feasibility information as a persistent, reusable computational object. In this perspective, conflict structures, admissibility relations, synchronisation dependencies, and partial feasibility states are maintained across successive solution states rather than being regarded as transient by-products of individual algorithmic procedures.

3.1. Limitations of current optimisation paradigms

One of the principal limitations of contemporary optimisation paradigms lies in treating feasibility as an implicit property of the solution space rather than as an explicit computational object. In classical optimisation environments, feasibility is often verified locally through a collection of predefined constraints. Once a candidate solution satisfies these constraints, optimisation focuses on improving the objective function values. This paradigm has proven highly successful for many scheduling, routing, and resource-allocation problems. However, synchronised systems introduce a fundamentally different situation. Here, feasibility depends on what may be termed synchronisation consistency, namely the preservation of temporal and structural coherence among multiple interacting resources. Individual operations may be feasible when considered separately, while becoming infeasible once synchronisation relationships are taken into account. For example, two transportation tasks may be individually feasible but jointly infeasible if they compete for the same vehicle or violate synchronisation continuity.

Consequently, feasibility becomes a global structural property that emerges from interactions among distributed processes rather than from a collection of isolated local conditions. Existing optimisation paradigms only partially address this phenomenon, frequently discovering infeasibility after substantial computational effort has already been invested in exploring inadmissible solution regions.

Table 4 illustrates that feasibility-aware architectures should not be interpreted as a replacement for existing optimisation paradigms. Rather, they provide an integrative framework in which declarative reasoning, scalable exploration, synchronisation management, and computational reuse are coordinated through explicit feasibility supervision.

Tab. 4. Comparison of optimisation paradigms

Feature	Classical Optimization	Declarative Optimization	Metaheuristics	Feasibility-Aware Architectures
Feasibility handling	Implicit	Explicit	Often repair-based	Structural layer
Scalability	Moderate	Limited	High	High
Synchronization reasoning	Weak	Strong	Moderate	Strong
Incremental recomputation	Rare	Limited	Partial	Core mechanism
Computational reuse	Weak	Weak	Moderate	Explicit architectural layer
Dynamic adaptability	Limited	Moderate	High	High
Structural admissibility	Secondary	Strong	Often implicit	Primary concept

3.2. Repeated global recomputation

A second limitation concerns the widespread reliance on repeated reconstruction of complete solution structures following local modifications. Many optimisation procedures employ local search operators that alter only a small fragment of the current solution. Nevertheless, due to strong synchronisation dependencies, even minor modifications frequently trigger global updates affecting multiple resources and operational layers. This phenomenon is particularly pronounced in environments characterised by strong synchronisation coupling, where local changes propagate throughout the system. A delay affecting one transportation task, for instance, may influence subsequent vehicle assignments, service schedules, and resource continuity relationships across a large portion of the operational network.

The resulting computational burden often originates not from exploration itself but from repeated feasibility verification and synchronisation maintenance. As system size increases, the cost of such recomputation may eventually dominate the overall optimisation process.

Future architectures should therefore support feasibility-preserving recomputation, where only those synchronisation relationships directly affected by a modification are updated while previously verified admissibility conditions remain intact. This principle transforms feasibility maintenance from repeated global reconstruction into localised incremental supervision supported by reusable computational structures.

3.3. Fragmentation of computational paradigms

Another important challenge concerns the fragmentation of contemporary optimisation methodologies. Declarative optimisation, graph-based reasoning, metaheuristic exploration, and computational acceleration have evolved largely as independent research streams. Each paradigm contributes valuable capabilities, yet

interactions among them remain surprisingly limited. Declarative approaches provide rigorous reasoning and propagation mechanisms. Graph-based methods offer efficient conflict representation and admissibility analysis. Metaheuristics enable large-scale exploration. Acceleration techniques reduce computational overhead. However, these components are rarely integrated into a common architectural framework.

As a consequence, computational effort is frequently duplicated. Similar feasibility relationships may be verified repeatedly by different algorithmic layers, while structural information generated during one phase of optimisation is often discarded rather than reused. Existing acceleration methods typically operate as isolated algorithmic enhancements rather than as components of a broader synchronisation-aware architecture. The challenge, therefore, lies not merely in improving individual algorithms but in creating computational ecosystems that coordinate multiple reasoning and optimisation paradigms within a common feasibility-oriented framework.

3.4. Conceptual transition toward multi-layer architectures

The limitations discussed above suggest a conceptual transition from monolithic optimisation systems toward explicitly layered computational architectures.

In the proposed multi-layer architecture shown in Figure 2, feasibility becomes the primary organising principle of the optimisation process. Rather than serving as a secondary verification stage, it actively supervises exploration and restricts optimisation activities to structurally admissible regions of the solution space. The first layer concerns structural feasibility reasoning. Its purpose is to identify admissible exploration regions through conflict graphs, admissibility graphs, and synchronisation networks. A synchronisation network may be understood as a graph representing temporal, resource, and coordination dependencies among operations. Instead of reasoning about isolated tasks, the system reasons about an interconnected synchronisation structure governing admissibility throughout the operational environment. The second layer corresponds to declarative optimisation, which provides a rigorous representation of synchronisation constraints, temporal consistency requirements, and resource-continuity relationships. Constraint propagation mechanisms ensure formal consistency within the admissible regions identified by the feasibility layer.

The third layer concerns scalable exploration, typically realised through metaheuristic search procedures. Rather than indiscriminately generating candidate solutions, exploration is guided by admissibility information produced by preceding layers. This admissibility-guided exploration reduces the likelihood of spending computational effort on structurally infeasible neighbourhoods.

Finally, all layers are supported by computational reuse mechanisms that preserve the synchronisation knowledge generated during optimisation. Viewed collectively, these layers define a reference computational architecture rather than a specific optimisation algorithm, emphasising coordination among complementary reasoning mechanisms instead of replacing existing optimisation techniques.

3.5. Computational reuse and incremental evaluation

The ability to preserve and reuse synchronisation knowledge may constitute one of the most important scalability enablers for future optimisation architectures. Traditional optimisation systems repeatedly reconstruct feasibility information after each modification. In contrast, feasibility-aware architectures seek to preserve reusable intermediate structures representing synchronisation consistency and admissibility relationships. Representative reusable structures include conflict graphs, admissibility networks, synchronisation paths, partial feasibility states, and incremental evaluation records. Collectively, these artefacts form what may be termed reusable admissibility structures, namely preserved representations of previously verified synchronisation knowledge that can be incrementally updated during exploration. A closely related concept is localised incremental recomputation. Intuitively, when a local modification occurs, only the affected synchronisation neighbourhood should be updated. For example, changing a single transportation assignment should trigger updates only for neighbouring vehicles, service teams, and temporally connected operations, rather than the entire operational schedule.

This idea naturally leads to graph-supported propagation, in which admissibility changes are transmitted via explicit dependency structures rather than requiring repeated global verification. In such an approach, synchronisation information flows through conflict graphs and synchronisation networks, much like disturbances propagate through physical systems.

From this perspective, scalability becomes not merely a consequence of more efficient search algorithms but an emergent property of the computational architecture itself. Future optimisation systems may therefore derive their performance primarily from the ability to preserve, update, and exploit synchronisation knowledge throughout the optimisation process rather than from repeatedly reconstructing admissibility information from scratch.

4. SYNCHRONISATION-AWARE COMPUTATIONAL REUSE

Computational efficiency has long been regarded as one of the principal evaluation criteria for practical optimisation systems. While practitioners emphasise robustness, scalability, and responsiveness, researchers continue to develop increasingly sophisticated optimisation methodologies and acceleration techniques. Bridging these perspectives requires architectures that improve computational performance not only through faster search algorithms but also through the systematic reuse of previously established feasibility and synchronisation knowledge.

4.1. Local updating of feasibility structures

In synchronised systems, local modifications of a solution very often affect only a limited fragment of the global schedule structure. A change in resource assignment, a shift of a single operation, or a modification in the order of selected tasks does not necessarily invalidate the entire solution, but only a local region connected through temporal, resource-related, or synchronisation relationships. Nevertheless, classical optimisation procedures often perform a complete re-verification of the entire schedule after each local modification. This leads to substantial computational overhead, especially in large-scale systems characterised by a high number of operations, resources, and synchronisation relationships. Similar ideas have appeared in accelerator-based scheduling and metaheuristic research, where localised updates and reusable computational structures have been shown to substantially reduce evaluation costs (Bożejko & Bocewicz, 2020; Nowicki & Smutnicki, 2005; Smutnicki et al., 2026; Taillard, 1990). However, these mechanisms are typically introduced as isolated algorithmic enhancements rather than as components of a broader computational architecture. Future feasibility-preserving systems may therefore benefit from explicit reuse layers that can maintain and update synchronisation-related knowledge throughout the optimisation process. For this reason, an important direction in the development of optimisation architectures is the transition from global solution recomputation toward local updating of feasibility structures. This means that after a local modification has been performed, the entire solution is not reconstructed. Instead, only the fragment of the structure that may have been affected is identified. Then, solely the related conflict relations, temporal dependencies, and synchronisation conditions are updated. The remaining, unchanged fragments of the solution are preserved and reused in subsequent iterations of the optimisation process.

Such an approach can be referred to as synchronisation-aware computational reuse. Its essence is to preserve intermediate computational structures describing solution feasibility, such as conflict graphs, synchronisation graphs, precedence relations, forward/backward temporal structures, or partial schedule-evaluation states. These structures are not treated as a one-time result of feasibility verification, but as a form of computational memory that can be updated and reused while evaluating subsequent neighbouring solutions. In this sense, the proposed perspective extends classical incremental neighbourhood-evaluation concepts known from metaheuristics (Bożejko & Bocewicz, 2020; Nowicki & Smutnicki, 1996, 1998, 2005; Rudy et al., 2020; Rudy & Żelazny, 2015) toward structures that explicitly preserve feasibility.

Particular importance is attached here to mechanisms for propagating admissibility. If a local change affects a selected operation, resource, or synchronisation relation, the consequences of this change should be propagated only through those elements of the structure that are directly or indirectly related to it. In graph-based representations, this means analysing modified vertices, incident edges, dependency components, and possible propagation paths. Instead of asking whether the entire schedule is still feasible, one may ask a more local question: which admissibility relationships could have been violated by this particular change? This approach is consistent with graph-based feasibility reasoning, in which graph structures represent conflicts, temporal dependencies, and synchronisation conditions.

Forward- and backwards-evaluation structures may also support computational reuse. Earliest and latest feasible execution times, critical-path fragments, and synchronisation dependencies can often be updated incrementally rather than recomputed from scratch. Such mechanisms provide an additional source of

scalability, particularly in large synchronised systems where local modifications affect only limited portions of the overall structure (Nowicki & Smutnicki, 2005; Smutnicki et al., 2026; Taillard, 1993).

Future computational architectures may further extend these ideas through adaptive reuse mechanisms that identify which synchronisation structures, admissibility relationships, and evaluation components should be preserved across successive search iterations. Such capabilities may significantly reduce computational overhead while improving responsiveness in dynamic and uncertainty-affected environments.

From the perspective of synchronised systems, the key issue is therefore not only which search strategy is used, but also how the consequences of subsequent local modifications are evaluated. Scalability may increasingly depend on an optimisation architecture's ability to preserve, update, and reuse feasibility structures, rather than on ever more complex search mechanisms alone. In this sense, computational reuse becomes not only an acceleration technique but also a fundamental component of next-generation optimisation architectures.

4.2. Example: Local feasibility verification in the STTS problem

One of the most important issues discussed in Section 4.1 is the possibility of locally updating feasibility structures after performing a modification intended to improve a given solution. This mechanism is particularly important in synchronised systems, where even a small change in resource assignment may lead to a violation of temporal or synchronisation relationships. To illustrate this idea, an example based on the STTS problem is considered. The STTS case should therefore be interpreted as an illustrative demonstration of the proposed selective-feasibility mechanism rather than as experimental evidence of computational acceleration or scalability. Its purpose is to demonstrate how an affected feasibility region can be identified and updated while retaining unchanged admissibility information.

Let us consider the STTS problem, defined in (Radzki, Bocewicz, Nielsen, et al., 2026; Radzki et al., 2025), which concerns the planning of transportation for mobile teams in order to serve a given set of customers. In the considered example, the distribution network consists of the depot P_0 and six customers $P_1, \dots, P_6$, i.e., $\mathcal{P} = \{P_0, P_1, \dots, P_6\}$ – see Fig. 3a). A fleet of two vehicles $\mathcal{V} = \{V_1, V_2\}$, used to transport service teams, is available at the depot. Service operations are performed by three mobile service teams $\mathcal{S} = \{S_1, S_2, S_3\}$. Each customer expects the service to be performed within the time-window configuration $\mathcal{W}$, shown in Fig. 3b). In STTS problems, the objective is to determine a feasible service mission MS , comprising the assignment of service teams to customers and the routes of their delivery and pickup, while minimizing the adopted evaluation criterion f and satisfying all temporal, resource-related, continuity, and synchronization constraints. In the illustrative example, the criterion is defined as $f(MS) = \max\{T_j(MS): V_j \in \mathcal{V}\}$, where $T_j(MS)$ denotes the working time of vehicle V_j in mission MS .

The search for optimal solutions in the STTS problem is conditioned by the prior identification of feasible solutions. This means that before the actual solution-improvement procedure begins, it is necessary to determine whether, for a given network $\mathcal{P}$, vehicle fleet $\mathcal{V}$, set of service teams $\mathcal{S}$, and time-window configuration $\mathcal{W}$, there exists at least one mission that can be executed. Indeed, the available resources, i.e., the number of vehicles and/or service teams, do not always ensure timely service for all customers. For this purpose, graph structures representing conflict relations and spatiotemporal dependencies resulting from the time-window configuration are analysed. In particular, for the transportation layer, a graph $G^{\mathcal{V}}$ is constructed, whose vertices correspond to service-team disembarking and boarding operations, while edges represent conflicts that make it impossible to assign two operations to the same vehicle. An example of such a graph is shown in Fig. 4. If graph $G^{\mathcal{V}}$ can be properly colored using no more colours than the number of available vehicles, then the basic feasibility condition for the transportation layer is satisfied.

Together with analogous conditions for service teams and the continuity conditions for their flows, this enables the determination of at least one feasible mission. Such a solution may then serve as the starting point for further optimisation. The existence of a non-empty feasible solution space enables the use of the Tabu Search heuristic to search for solutions that are better with respect to the adopted criterion.

As shown in Fig. 4, the procedure starts from an initial feasible solution MS_0 . This solution is then modified by generating neighbouring solutions. A neighbouring solution is obtained through an elementary local modification, understood as a change of the current feasible solution in one selected decision dimension.

In the case considered, it corresponds to assigning one available vehicle to one transportation operation. In general, search strategies may also consider multi-dimensional neighbourhoods, in which several assignments or

several decision dimensions are modified simultaneously. Such neighbourhoods may require a broader, but still localised, feasibility update. In the example analysed below, constructing a neighbour therefore consists of changing the assignment of one of the available vehicles to a selected transportation operation, which corresponds to changing the colour of a selected vertex in graph G^V .

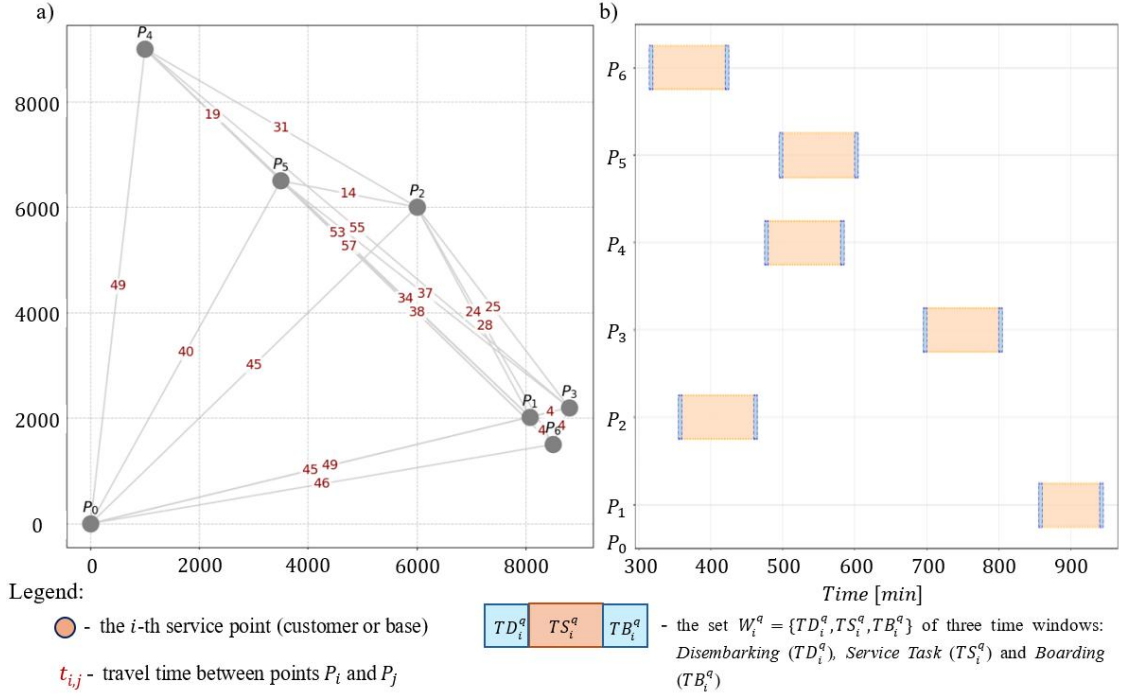


Fig. 3. The considered distribution network a) and within the time-window configuration b)

However, not every such change leads to a solution that remains feasible. Fig. 4 presents two illustrative cases. The first neighbourhood solution, denoted as MS_1 , represents a feasible mission because the change in vehicle assignment does not violate local conflict relations or synchronisation conditions. The second neighbourhood solution, denoted as MS_2 , leads to an infeasible mission because the colour change creates a conflict in graph G^V or violates the continuity conditions of a service-team flow.

In a classical approach, evaluating each generated neighbour would require re-verification of the entire schedule, including all transportation, service, and synchronisation operations. In the considered approach, this is not necessary. Since the modification concerns only a selected vehicle assignment, it is sufficient to verify the consistency of the graph G^V only in the part that may have changed. In practice, this means checking the modified vertex, its incident edges, and the related synchronisation relationships. The unchanged fragments of the solution structure are preserved and reused. As a result, infeasible solutions such as MS_2 can be quickly rejected without full schedule recomputation, whereas feasible solutions such as MS_1 can be passed to further evaluation and potentially accepted by the Tabu Search mechanism.

As a result, subsequent iterations of the heuristic yield an improved solution that preserves feasibility and enhances the value of the adopted criterion. Fig. 4 illustrates such a case MS_A : after a series of local modifications and incremental feasibility checks, a mission with a more favourable temporal structure is obtained. The objective-function value for this solution is $f_A = f(MS_A) = 525$ [u.t.], which corresponds to an improvement of approximately 17% relative to the initial solution MS_0 ($f_0 = f(MS_0) = 633$ [u.t.]).

The procedure illustrated in Fig. 4 corresponds to the idea of synchronisation-aware computational reuse. Its essence is that, after a neighbouring solution has been generated, the entire schedule is not reconstructed, and all constraints are not checked from scratch. Instead, the previously determined feasibility structure is reused, and only the fragment of graph G^V together with the related synchronisation relationships that may have been affected by the performed modification is verified. This approach can significantly accelerate the search within the feasible solution space, because expensive global verification is replaced by local propagation of changes. The example shows that in synchronised systems, not only the search strategy itself is important but also the way in which feasibility structures are updated locally. Rather than repeatedly reconstructing and

verifying the entire solution, a feasibility-aware architecture exploits previously computed and unchanged fragments of graph G^V , thereby reducing the computational cost of subsequent iterations.

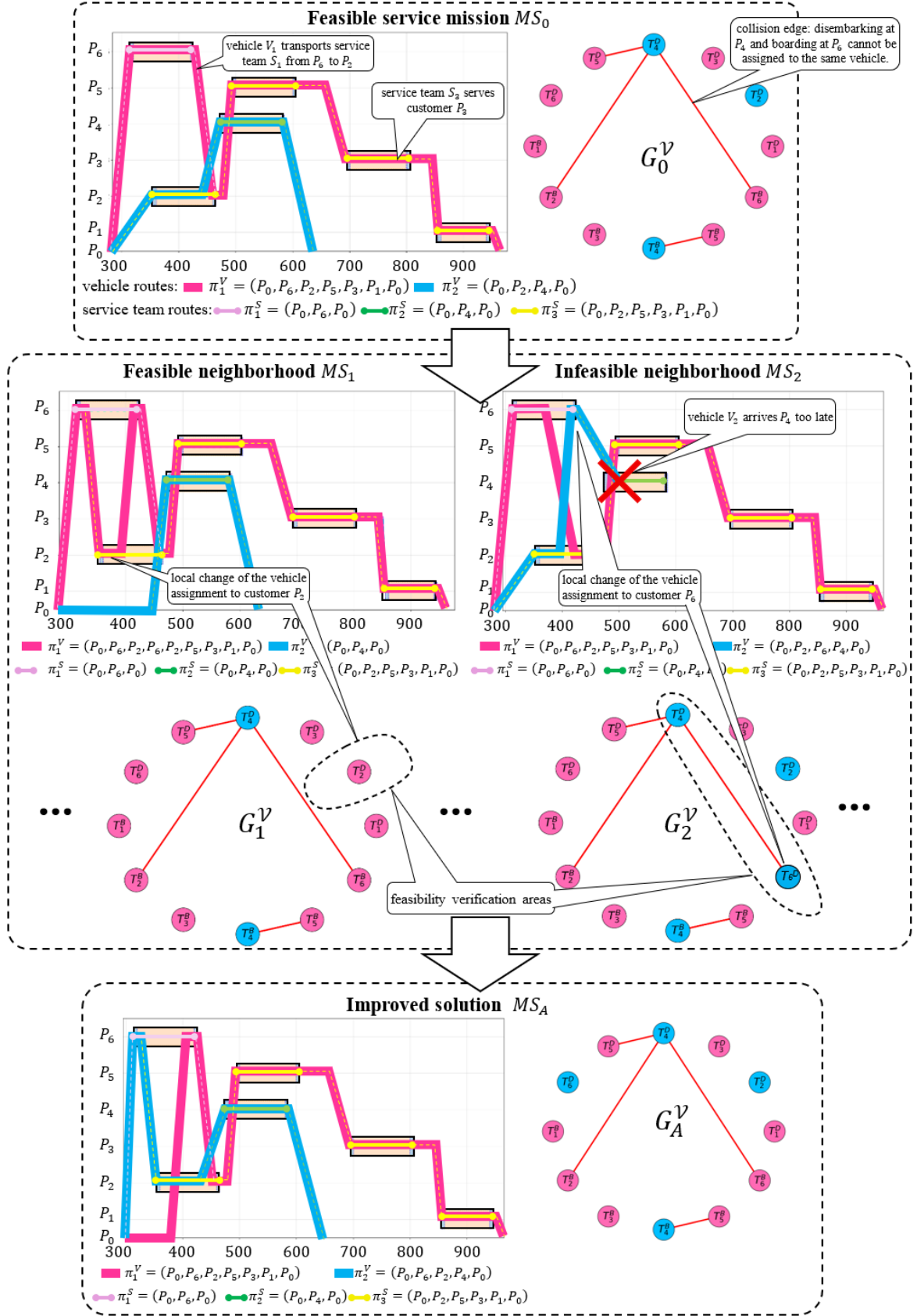


Fig. 4. Local feasibility verification of neighboring STTS missions based on incremental updating of graph G^V

Taken together, the concepts presented in this section illustrate that computational reuse should not be viewed merely as an implementation-level acceleration technique but as a fundamental architectural mechanism supporting scalable feasibility supervision in synchronised systems.

5. FUTURE DIRECTIONS TOWARD FEASIBILITY-AWARE COMPUTATIONAL ARCHITECTURES

The concepts developed throughout this paper suggest a broader shift in the design of optimisation systems operating in synchronised multi-resource environments. Rather than treating feasibility as a secondary verification step, future computational architectures may place structural-feasibility supervision at the core of exploration, propagation, and optimisation processes.

One important research direction concerns the relationship between scalability and computational reuse. Existing optimisation studies predominantly evaluate algorithms in terms of solution quality and execution time, whereas the ability to preserve and reuse previously computed structural knowledge remains largely unexplored. In strongly synchronised systems, local modifications often affect only limited fragments of the global synchronisation structure, motivating localised recomputation mechanisms that update only those portions of feasibility and admissibility structures directly influenced by a modification.

A closely related direction involves synchronisation-aware admissibility propagation. Instead of repeatedly verifying all synchronisation constraints after each neighbourhood transition, future architectures may exploit graph-supported propagation mechanisms that incrementally update admissibility information via conflict networks and synchronisation dependencies. Such mechanisms could substantially reduce computational overhead while maintaining structural consistency throughout the search process.

Future studies should also investigate feasibility-preserving exploration strategies that restrict optimisation to structurally admissible regions, thereby reducing the need for costly repair procedures. Equally important is the development of synchronisation-aware benchmarking environments that evaluate not only optimisation quality and runtime but also computational reuse efficiency, feasibility preservation, synchronisation overhead, and scalability as synchronisation density increases.

Particularly promising opportunities emerge in uncertainty-aware synchronised systems affected by stochastic travel times, failures, dynamically arriving tasks, and changing resource availability. The integration of machine-learning techniques capable of predicting patterns of admissibility propagation, selecting adaptive recomputation strategies, or identifying promising exploration regions represents another natural extension of the proposed framework.

Overall, these directions point toward self-adaptive architectures with explicit feasibility supervision that combine exact reasoning, declarative modelling, scalable exploration, and synchronisation-aware computational reuse within unified real-time decision-support systems.

6. CONCLUSIONS

Optimisation has dominated decision-support research for decades, with scalability traditionally viewed as a consequence of increasingly efficient search procedures. In strongly synchronised systems, however, the fundamental challenge may no longer be how to search larger solution spaces, but how to preserve, propagate, and reuse feasibility knowledge throughout the optimisation process. This observation motivates the conceptual perspective developed in this paper.

From this standpoint, feasibility should not be regarded merely as a prerequisite for optimisation or as a constraint-verification stage executed after candidate solutions have been generated. Instead, it can be viewed as a computational resource whose preservation and reuse support scalable synchronisation management and decision making.

Accordingly, the paper advocates a transition from optimisation-centric reasoning toward feasibility-aware computational architectures in which structural feasibility reasoning, declarative modelling, scalable exploration, synchronisation-aware propagation, and computational reuse operate as mutually reinforcing components of a unified framework. In such architectures, localised recomputation and reusable admissibility structures become as important as the search strategies themselves. The novelty of the proposed perspective, therefore, lies primarily in the architectural integration of established and prospective mechanisms around persistent structural-feasibility supervision, rather than in introducing each of these computational mechanisms individually.

The analysis also highlights current limitations, including repeated global recomputation, insufficient integration between declarative and heuristic methods, limited exploitation of reusable structural knowledge, and the lack of benchmarking methodologies assessing synchronisation-aware computational efficiency. Addressing these challenges may contribute to the emergence of a new generation of decision-support systems explicitly designed to preserve feasibility, exploit computational reuse, and incrementally maintain synchronisation consistency in complex cyber-physical and service-oriented environments.

Viewed from this perspective, synchronisation should no longer be regarded merely as an execution constraint but as the structural mechanism through which feasibility emerges in complex engineering systems. Consequently, future optimisation architectures may derive their scalability not only from increasingly efficient search procedures but also from their ability to preserve, supervise, propagate, and reuse synchronisation-consistent feasibility structures. This architectural viewpoint may ultimately serve as the foundation for a new generation of decision-support systems for large-scale cyber-physical, manufacturing, logistics, and service environments.

Conflicts of Interest

The authors declare no conflict of interest.

REFERENCES

- Ammann, P., Kolisch, R., & Schiffer, M. (2023). Driver routing and scheduling with synchronization constraints. *Transportation Research Part B: Methodological*, 174, Article 102760. <https://doi.org/10.1016/j.trb.2023.05.009>
- Anıkan, U., Gürel, S., & Aktürk, M. S. (2016). Integrated aircraft and passenger recovery with cruise time controllability. *Annals of Operations Research*, 236(2), 295–317. <https://doi.org/10.1007/s10479-013-1424-2>
- Banaszak, Z. A., & Krogh, B. H. (1990). Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows. *IEEE Transactions on Robotics and Automation*, 6(6), 724–734. <https://doi.org/10.1109/70.63273>
- Behnamian, J., & Fatemi Ghomi, S. M. T. (2016). A survey of multi-factory scheduling. *Journal of Intelligent Manufacturing*, 27(1), 231–249. <https://doi.org/10.1007/s10845-014-0890-y>
- Bocewicz, G. (2023). *Declarative models of concurrent cyclic processes*. Springer.
- Bożejko, W., & Bocewicz, G. (Eds.). (2020). *Modelling and performance analysis of cyclic systems*. Springer. <https://doi.org/10.1007/978-3-030-27652-2>
- Brailsford, S., Kozłowski, R., Haczyński, J., Haczyński, H., Pottie, K., Mirabnejad, M., Mohammadi, H., Mirzabaghi, M., Aghsami, A., Jolai, F., & Yazdani, M. (2022). Home health care problem with synchronization visits and considering samples transferring time: A case study in Tehran, Iran. *International Journal of Environmental Research and Public Health*, 19(22), Article 15036. <https://doi.org/10.3390/ijerph192215036>
- Bredström, D., & Rönnqvist, M. (2008). Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. *European Journal of Operational Research*, 191(1), 19–31. <https://doi.org/10.1016/j.ejor.2007.07.033>
- Brzozowska, J., Pizoń, J., Baytikenova, G., Gola, A., Zakimova, A., & Piotrowska, K. (2023). Data engineering in the CRISP-DM process production data – case study. *Applied Computer Science*, 19(3), 83–95. <https://doi.org/10.35784/acs-2023-26>
- Čáp, M., Gregoire, J., & Frazzoli, E. (2016). Provably safe and deadlock-free execution of multi-robot plans under delaying disturbances. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 5113–5118). IEEE. <https://doi.org/10.1109/IROS.2016.7759750>
- Čáp, M., Novak, P., Kleiner, A., & Selecký, M. (2015). Prioritized planning algorithms for trajectory coordination of multiple mobile robots. *IEEE Transactions on Automation Science and Engineering*, 12(3), 835–849. <https://doi.org/10.1109/TASE.2015.2445780>
- Chen, J., Huang, G. Q., Luo, H., & Wang, J. (2015). Synchronisation of production scheduling and shipment in an assembly flowshop. *International Journal of Production Research*, 53(9), 2787–2802. <https://doi.org/10.1080/00207543.2014.994075>
- Cochran, D. S., Arinez, J. F., Duda, J. W., & Linck, J. (2001). A decomposition approach for manufacturing system design. *Journal of Manufacturing Systems*, 20(6), 371–389. [https://doi.org/10.1016/S0278-6125\(01\)80058-3](https://doi.org/10.1016/S0278-6125(01)80058-3)
- Coelho, L. C., & Mancini, S. (2026). The collect-and-process vehicle routing problem with completion time minimization: An application in home healthcare. *European Journal of Operational Research*, 330(2), 520–535. <https://doi.org/10.1016/j.ejor.2026.05.034>
- de Aguiar, A. R. P., Ramos, T. R. P., & Gomes, M. I. (2023). Home care routing and scheduling problem with teams' synchronization. *Socio-Economic Planning Sciences*, 86, Article 101503. <https://doi.org/10.1016/j.seps.2022.101503>
- Drexl, M. (2012). Synchronization in vehicle routing—A survey of VRPs with multiple synchronization constraints. *Transportation Science*, 46(3), 297–316. <https://doi.org/10.1287/trsc.1110.0400>
- Erdogan, M. S., & M'Hallah, R. (2023). Synchronizing delivery and installation with vehicle sharing: A hybrid adaptive large neighborhood search. *Computers & Industrial Engineering*, 185, Article 109676. <https://doi.org/10.1016/j.cie.2023.109676>
- Fanti, M. P. (2001). Event control for deadlock avoidance in automated guided vehicle systems. In *2001 European Control Conference (ECC)* (pp. 1217–1222). IEEE. <https://doi.org/10.23919/ECC.2001.7076082>
- Frifita, S., Masmoudi, M., & Euch, J. (2017). General variable neighborhood search for home healthcare routing and scheduling problem with time windows and synchronized visits. *Electronic Notes in Discrete Mathematics*, 58, 63–70. <https://doi.org/10.1016/j.endm.2017.03.009>

- Fu, Y., Hou, Y., Wang, Z., Wu, X., Gao, K., & Wang, L. (2021). Distributed scheduling problems in intelligent manufacturing systems. *Tsinghua Science and Technology*, 26(5), 625–645. <https://doi.org/10.26599/TST.2021.9010009>
- Gkiotsalitis, K., Cats, O., & Liu, T. (2023). A review of public transport transfer synchronisation at the real-time control phase. *Transport Reviews*, 43(1), 88–107. <https://doi.org/10.1080/01441647.2022.2035014>
- Hooker, J. N. (2014). *Integrated methods for optimization* (2nd ed.). Springer. <https://doi.org/10.1007/978-1-4614-1899-3>
- Laborie, P. (2003). Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results. *Artificial Intelligence*, 143(2), 151–188. [https://doi.org/10.1016/S0004-3702\(02\)00362-4](https://doi.org/10.1016/S0004-3702(02)00362-4)
- Lee, J., Bagheri, B., & Kao, H. A. (2015). A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems. *Manufacturing Letters*, 3, 18–23. <https://doi.org/10.1016/j.mfglet.2014.12.001>
- Lei, H., Zhen, L., Shi, H., Pan, Y., & Zhu, C. (2026). Capacitated team orienteering with split delivery-based coordinated routing of truck-drone and drone-drone. *Expert Systems with Applications*, 329, Article 132974. <https://doi.org/10.1016/j.eswa.2026.132974>
- Li, Y., Xu, X., & Wang, F. (2023). Research on home health care scheduling considering synchronous access of caregivers and vehicles. *Sustainability*, 15(7), Article 6132. <https://doi.org/10.3390/su15076132>
- Lohmer, J., & Lasch, R. (2021). Production planning and scheduling in multi-factory production networks: A systematic literature review. *International Journal of Production Research*, 59(7), 2028–2054. <https://doi.org/10.1080/00207543.2020.1797207>
- Lucci, M., Severin, D., & Zabala, P. (2026). Exact resolution of a simultaneous vehicle routing and crew scheduling problem in long-haul transport. *International Transactions in Operational Research*, 33(5), 3158–3186. <https://doi.org/10.1111/itor.70014>
- Monostori, L. (2014). Cyber-physical production systems: Roots, expectations and R&D challenges. *Procedia CIRP*, 17, 9–13. <https://doi.org/10.1016/j.procir.2014.03.115>
- Nasir, J. A., & Kuo, Y. H. (2020). A decision support framework for home health care transportation with simultaneous multi-vehicle routing and staff scheduling synchronization. *Decision Support Systems*, 138, Article 113361. <https://doi.org/10.1016/j.dss.2020.113361>
- Nowicki, E., & Smutnicki, C. (1996). A fast tabu search algorithm for the permutation flow-shop problem. *European Journal of Operational Research*, 91(1), 160–175. [https://doi.org/10.1016/0377-2217\(95\)00037-2](https://doi.org/10.1016/0377-2217(95)00037-2)
- Nowicki, E., & Smutnicki, C. (1998). The flow shop with parallel machines: A tabu search approach. *European Journal of Operational Research*, 106(2–3), 226–253. [https://doi.org/10.1016/S0377-2217\(97\)00260-9](https://doi.org/10.1016/S0377-2217(97)00260-9)
- Nowicki, E., & Smutnicki, C. (2005). An advanced tabu search algorithm for the job shop problem. *Journal of Scheduling*, 8(2), 145–159. <https://doi.org/10.1007/s10951-005-6364-5>
- Öztürkoglu, Ö., & Özşakallı, G. (2023). A generic workforce scheduling and routing problem with the vehicle sharing and drop-off and pick-up policies. arXiv. <https://doi.org/10.48550/arXiv.2303.02229>
- Pinedo, M. L. (2022). *Scheduling: Theory, algorithms, and systems* (6th ed.). Springer. <https://doi.org/10.1007/978-3-031-05921-6>
- Pour, S. M., Marjani Rasmussen, K., Drake, J. H., & Burke, E. K. (2019). A constructive framework for the preventive signalling maintenance crew scheduling problem in the Danish railway system. *Journal of the Operational Research Society*, 70(11), 1965–1982. <https://doi.org/10.1080/01605682.2018.1507423>
- Purnamasari, C. D., & Santoso, A. (2018). Vehicle Routing Problem (VRP) for courier service: A review. *MATEC Web of Conferences*, 204, Article 07007. <https://doi.org/10.1051/mateconf/201820407007>
- Radzki, G., Bocewicz, G., & Banaszak, Z. (2026). Feasibility analysis of service mission planning under travel time uncertainty. In *30th Conference on Automation – Innovations and Future Perspectives (Automation 2026)*.
- Radzki, G., Bocewicz, G., Jasiulewicz-Kaczmarek, M., Nielsen, I., & Banaszak, Z. (2026). Enhancing the feasibility of service missions under travel time uncertainty. In *Proceedings of the IFAC World Congress*.
- Radzki, G., Bocewicz, G., Nielsen, I., Rudy, J., & Banaszak, Z. (2026). Multimodal processes driven distribution of customers' ordered services. *Journal of Industrial and Production Engineering*. <https://doi.org/10.1080/21681015.2025.2583199>
- Radzki, G., Bocewicz, G., Rudy, J., Idzikowski, R., & Banaszak, Z. (2025). Assessing feasibility in service teams transport scheduling with dedicated and flexible dispatch approaches. *Applied Sciences*, 15(23), Article 12727. <https://doi.org/10.3390/app152312727>
- Relich, M., Nielsen, I., & Gola, A. (2022). Reducing the total product cost at the product design stage. *Applied Sciences*, 12(4), Article 1921. <https://doi.org/10.3390/app12041921>
- Rossi, F., van Beek, P., & Walsh, T. (Eds.). (2006). *Handbook of constraint programming*. Elsevier.
- Rudy, J., & Żelazny, D. (2015). GACO: A parallel evolutionary approach to multi-objective scheduling. In *Parallel Processing and Applied Mathematics (PPAM 2015)* (pp. 307–320). Springer. https://doi.org/10.1007/978-3-319-15934-8_21
- Rudy, J., Pempera, J., & Smutnicki, C. (2020). Improving the TSAB algorithm through parallel computing. *Archives of Control Sciences*, 30(3), 411–435. <https://doi.org/10.24425/acs.2020.134672>
- Saeidlou, S., Saadat, M., Amini Sharifi, E., & Jules, G. D. (2019). Agent-based distributed manufacturing scheduling: An ontological approach. *Cogent Engineering*, 6(1), Article 1565630. <https://doi.org/10.1080/23311916.2019.1565630>
- Schnetzler, M. J., Sennheiser, A., & Schönsleben, P. (2007). A decomposition-based approach for the development of a supply chain strategy. *International Journal of Production Economics*, 105(1), 21–42. <https://doi.org/10.1016/j.ijpe.2006.02.004>
- Sharon, G., Stern, R., Felner, A., & Sturtevant, N. R. (2015). Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219, 40–66. <https://doi.org/10.1016/j.artint.2014.11.006>
- Smutnicki, C., Banaszak, Z., & Bocewicz, G. (2026). Accelerators for metaheuristics. In *EUROCAST 2026 Extended Abstracts* (pp. 68–69).
- Taillard, E. (1990). Some efficient heuristic methods for the flow shop sequencing problem. *European Journal of Operational Research*, 47(1), 65–74. [https://doi.org/10.1016/0377-2217\(90\)90090-x](https://doi.org/10.1016/0377-2217(90)90090-x)
- Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2), 278–285. [https://doi.org/10.1016/0377-2217\(93\)90182-m](https://doi.org/10.1016/0377-2217(93)90182-m)
- Wagner, G., & Choset, H. (2015). Subdimensional expansion for multirobot path planning. *Artificial Intelligence*, 219, 1–24. <https://doi.org/10.1016/j.artint.2014.11.001>

- Wang, Y., Hu, W., Hao, J. K., & Feng, J. (2025). Collaborative scheduling and routing of home healthcare service across multiple communities. *OR Spectrum*, 47(2), 525–563. <https://doi.org/10.1007/s00291-024-00798-4>
- Wu, N., & Zhou, M. C. (2005). Modeling and deadlock avoidance of automated manufacturing systems with multiple automated guided vehicles. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 35(6), 1193–1202. <https://doi.org/10.1109/TSMCB.2005.850141>
- Xu, K., & Carlsson, J. G. (2026). *A unified solution framework for truck-and-drone routing problems*. arXiv. <https://doi.org/10.48550/arXiv.2602.20310>
- Yin, Y., Liu, X., Qing, L., Qiu, H., & Wei, X. (2026). An exact solution approach for the multi-period EV home health care routing and scheduling problem with uncertain travel times and service durations. *Omega*, 144, Article 103587. <https://doi.org/10.1016/j.omega.2026.103587>