Periodic Event Scheduling for Automated Production Systems

Christoph Helmberg, Tobias Hofmann*, David Wenzel

Chemnitz University of Technology
*tobias.hofmann@math.tu-chemnitz.de

Abstract. Consider optimizing a periodic schedule for an automated production plant as a last step of a more comprehensive design process. In our scenario, each robot's cyclic sequence of operations and trajectories between potential waiting points have already been fully specified. Further given are those precedences that fix sequence requirements on operations between different robots. It remains to determine the starting time for each operation or movement of each robot within a common cyclic time period so as to avoid collisions of robots that operate in the same space simultaneously. So the task is to find a conflict resolving schedule that minimizes this common periodic cycle time while observing all precedence relations and collision avoidance constraints.

The proposed cycle time minimization problem for robot coordination has, to the best of our knowledge, not been studied before. We develop an approach for solving it by employing binary search for determining the smallest feasible period time of an Iso-Periodic Event Scheduling Problem (IPESP). This is a variant of the Periodic Event Scheduling Problem where the objects that have to be scheduled need to obey exactly the same period time. The possibility to wait arbitrarily long at waiting points turns out to be essential to justify the use of binary search for identifying the minimum cycle time, thereby avoiding bilinear mixed integer formulations. Special properties of the given scenario admit bounds on the periodic tension variables of an integer programming formulation. While the IPESP subproblems remain *NP*-complete in general, these bounds allow to solve real-world instances sufficiently fast for the approach to be applicable in practice. Numerical experiments on real-world and randomly generated data are supplied to illustrate the potential and limitations of this approach.

Keywords. automated systems, network optimization, periodic event scheduling, mixed-integer programming

1 Introduction

We consider a specific scheduling task that is one of the last steps in an involved design system for industrial robot production plants. At this stage, most decisions have been fixed already. The number and placement of the robots within the plant are given. The robots will operate periodically, interwoven in a common cycle with a yet unknown period time. For each robot the sequence of its operations and the associated trajectories of its movements between potential waiting points are fully specified. Certain operations on shared objects have to fulfill precedence constraints in order to result in a meaningful production process, e.g., an object can only be removed after it has been worked on. These precedence requirements are also part of the input. We are, however, free to adapt the length of the period time and to prescribe the starting times of successive operations or movements of the individual robots within their common cycle. These decisions have to satisfy the precedence constraints and to ensure that no collisions occur. In particular, we are still free to decide which robot operates first in the case that a number of robots compete for the same space while performing their individual process steps. These decisions have a significant influence on the period time, and the aim is to find a periodic starting time schedule that minimizes the overall period time of the production plant.

The robot production lines under investigation. The concrete application scenario is detailed in [10]. Here, we only enlarge on a few distinctive features that shed more light on the scenario at hand. Each robot in our scenario executes a fixed sequence of operations and moves between potential waiting points along a predestined trajectory in space. We regard the waiting points as states and the motions between them as state transitions. The states are visited periodically in a loop. The only freedom robots have is to wait at each such state, e.g., for letting another robot pass in order to avoid a collision if their trajectories lead them through the same space. Because trajectories are given, a preprocessing step allows to determine which state transitions may be executed simultaneously by both robots and which must not overlap in time. Once this information is available, there is no need to consider the trajectories within the scheduling problem. Indeed, their data is not even known to us. We are only given collision avoidance constraints in a tabular form. For each pair of robots, a separate table specifies for each pair of state transitions whether they may be executed simultaneously or not. This clearly distinguishes our scenario from many other robotics applications, which may incorporate online path finding algorithms, and renders it a pure offline periodic scheduling problem. It also ensures that the resulting schedule can be executed deterministically without further intervention. Such schedules are the favored choice in large-scale production systems, such as in the automotive sector.

Scheduling problems in robotics that are somewhat related to the application at hand, appear in wafer fabrication. Differing from our situation, altering the sequence in which a robot approaches the states is an additional challenge. However, a simpler setup helps to compensate this. As described by [1] or [30], the interdependencies in the production process have a linear or tree-like topology, respectively. This allows the authors to derive tailored algorithms with polynomial running time. In contrast, our application admits more complex interactions between robots, resulting in a computationally hard problem for which we cannot hope for such an algorithm; see Theorem 12.

Surprisingly, we find a suitable framework for our scenario in the context of train timetabling, which may not seem very similar at first glance. However, train lines can be seen as a fixed periodic sequence of passages between stations, at which a train may stop. Likewise, a robot moves from one program point to the next and can wait there for some time. Furthermore, two trains must not use the same track at the same time. This corresponds to our collision avoidance constraints, which ensure that two robots do not enter a shared space simultaneously. There are two major differences in this analogy. First, the period time is given for a railway schedule, while we seek the minimum period time with a valid schedule for the robots to make production fast and effective. And second, train lines may differ in their period time, while in our scenario robots have to perform their sequence exactly once in each production cycle.

Mathematical Contributions to Periodic Scheduling Problems. In train timetabling as well as robot system scheduling, one is confronted with temporal precedences, collision avoidance constraints, and of course periodic processes. In the realm of train timetabling, several effective solution approaches are based on the Periodic Event Scheduling Problem (PESP) introduced by [26]. Besides its application to train networks, the PESP has also been proposed for the Periodic Job Shop Problem or traffic light scheduling; see [26] or [8], respectively. The rich modeling capabilities of the PESP in the context of train timetable generation are presented comprehensively by [15]. While the PESP proves to be well applicable, [25] and, using a different proof idea, [22] established its *NP*-completeness.

Our scientific and technological interest lies in transferring and adapting the modeling capabilities of the PESP to this new application field in the design of automated production lines. In order to sketch the main ideas we first recall the definition of the PESP. For this, let $\mathbb{N}$ denote the set of natural numbers starting from one, and use $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$. Furthermore, all times are regarded as multiples of some unsplittable time unit, whence parameters and variables are considered as integers.

Figure 1: Motion Points and Corresponding Graph Structure
The graph theoretic robot cycle naturally emerges from the programmed path.

Definition 1 (PESP)**.** Given a directed graph $G = (V, A)$, vectors $l, u \in \mathbb{N}^A$, as well as a period time $T \in \mathbb{N}$, the task of the Periodic Event Scheduling Problem is to find vectors $x \in \mathbb{N}_0^V$ and $p \in \mathbb{Z}^A$ such that for any $a = (\alpha, \beta) \in A$ the *time window constraint*

$$l_a \leq x_\beta - x_\alpha + T\, p_a \leq u_a$$

is satisfied.

Figure 1 illustrates the link between a real robot's trajectory and its individual cycle in the graph structure. The states the robots visit during their respective cyclic sequence form the set of nodes V with some of the arcs in $A \subset V \times V$ encoding the precedence relations within the individual sequences. Collision avoidance constraints and further precedence relations will be encoded by further arcs between two robot cycles, as described in Section 2. For each state $\alpha \in V$ the x_α variables will give the point in time within the period time T, at which the respective robot will start the transition from α to its successor. For each precedence relation $a = (\alpha, \beta)$ the time window constraint specifies the lower and upper bounds on the time difference in starting times x_α and x_β modulo the time period T. This modulo operation is implemented by suitably choosing $p_a \in \mathbb{Z}$. The range of possible values that these *offset variables* p_a can attain, has a decisive influence on the problem's properties. If these variables could be fixed in advance, the PESP could be solved in $\mathcal{O}(|V| \cdot |A|)$ time; see for example [14]. It turned out that one of the many integer programming formulations of the PESP, the so-called Cycle Periodicity Formulation introduced by [21], seems to be a good choice in order to reduce the number of integer variables in a PESP instance. In this formulation it is sufficient to require the appearing Cycle Periodicity Constraints only for $\nu = |A| - |V| + 1$ fundamental cycles of a connected PESP instance; see [21]. We adapt these aspects to our scenario in Section 3. [16] further showed that instead of fundamental cycle bases one can choose integral ones, which form a wider class of cycle bases. In order to reduce the domain of the offset variables, the Minimum Integral Cycle Basis Problem and related issues have been studied. An overview is given by [17] or [13]

Valid inequalities for integer programming formulations of the PESP and related polyhedral aspects were investigated by many authors. [25] established the cycle inequalities and [22] the change-cycle inequalities. [23] investigated node-disjoint chain inequalities, [20] studied flow and chain flow inequalities, whereas [18] established multi-circuit cuts. Typically, these inequalities are used in branch-and-cut implementations that incorporate different separation heuristics. As [5] or [3] present, these frameworks may also comprise satisfiability methods as those of [7] or local improvement heuristics like the modulo network simplex algorithm, which is described by [24]. Despite of these efforts, the PESP is considered to be a notoriously hard problem; see [4] or [3]. Moreover, the computational results on the usefulness of the different classes of valid inequalities show an ambivalent picture; see [28] or [18]. However, there are very recent works that promise further progress. For example, [2] provide a pseudo-polynomial time separation algorithm for cycle inequalities and [19] propose flip inequalities that generalize both, the cycle and change cycle inequalities.

A modeling alternative to the PESP might in principle be given by the Max-Plus-Algebra framework as it is described by [6] or [9] for train timetabling applications. However, the constraint structure could not cover all periodicity and precedence requirements of our setting. Indeed, as with the models for wafer fabrication, Max-Plus-Algebra approaches allow algorithms with polynomial running time, whereas Theorem 12 states that the problem at hand is *NP*-complete.

Outline of this research. In Section 2 we describe in detail how to formalize the given application as a scheduling problem and show in Proposition 7 that all occurring modeling aspects can be realized within the PESP framework. Eventually, this gives rise to a variant of the PESP that we call Iso-Periodic Event Scheduling Problem (IPESP). A decisive difference between the scenario at hand to other PESP applications is that our goal is to realize the minimum period time, which causes nonlinear time window constraints. We provide sufficient conditions in Proposition 8 that allow to solve this optimization problem by employing a binary search approach. Moreover, we show that relaxing these conditions might cause feasibility issues. In Section 3 we prove that in spite of the iso-periodicity constraints the IPESP remains *NP*-complete. Furthermore, we adapt the Cycle Periodicity Formulation of the PESP to our scenario and provide admissible a priori bounds on the appearing integer periodic tension variables. Finally, in Section 4, we present numerical results for real-world as well as generated instances, which demonstrate the practical usability of our approach.

2 Modeling Inter-Linked Robot Cycles

Let R be a finite set of robots. For each robot $r \in R$ we are given a state set

$$V_r = \{0, 1, \ldots, n_r - 1\} \quad \text{with} \quad 2 \leq n_r \in \mathbb{N}$$

that the robot has to cyclically run through. A robot with only one state would be mathematically trivial and practically useless. If we need to address two or more robots at the same time, we subscript the states with the respective robot index, i.e. $0_r, 1_r, \ldots$ or α_r, to circumvent ambiguity. The direct successor and direct predecessor of some state $\alpha \in V_r$ shall be abbreviated by

$$\alpha' := (\alpha + 1) \bmod n_r \quad \text{and} \quad {}'\alpha := (\alpha - 1) \bmod n_r.$$

We remark that in [10] the alternative numbering scheme from 1 to n_r has been chosen, as it simplifies the notation in preparation algorithms. The cyclic sequence of state transitions will be represented by the arc set

$$A_r := \{(0, 1), (1, 2), \ldots, (n_r - 1, 0)\}$$

of a directed simple closed walk on V_r. The state sets V_r are pairwise disjoint by definition. Likewise the transition sets A_r do not intersect. All states or transitions are collected in

$$V = \dot{\bigcup_{r \in R}} V_r \quad \text{or} \quad \mathring{A} = \dot{\bigcup_{r \in R}} A_r,$$

respectively. The task is to find a minimum period time $T \in \mathbb{N}$ as well as an assignment of starting times $x \in \{0, \ldots, T-1\}^V$ that satisfy the subsequent constraints, of which three types are distinguished. We opted for the commonly used start formulation, though the arrival time could be utilized equivalently.

Transition time windows. If $\alpha, \beta \in V$, denote by

$$\mathrm{d}(\alpha, \beta) := (x_\beta - x_\alpha) \bmod T$$

the duration in time units elapsing from starting in state α until starting in state β. Be aware that this starting in the two states may also address different robots. For each pair of cyclically successive states $a := (\alpha, \alpha') \in \mathring{A}$ (indeed it is an element of A_r for some robot $r \in R$), the duration $\mathrm{d}(a) := \mathrm{d}(\alpha, \alpha')$ has to respect given lower bounds $\mathring{l} \in \mathbb{N}^{\mathring{A}}$ and upper bounds $\mathring{u} \in \mathbb{N}^{\mathring{A}}$, i.e.,

$$\mathring{l}_a \leq \mathrm{d}(a) \leq \mathring{u}_a \quad \text{for } a \in \mathring{A}. \tag{1}$$

The minimum transition times are given. The traversed distances or motion speeds of the robots are only limited by technology and have no influence on the model's

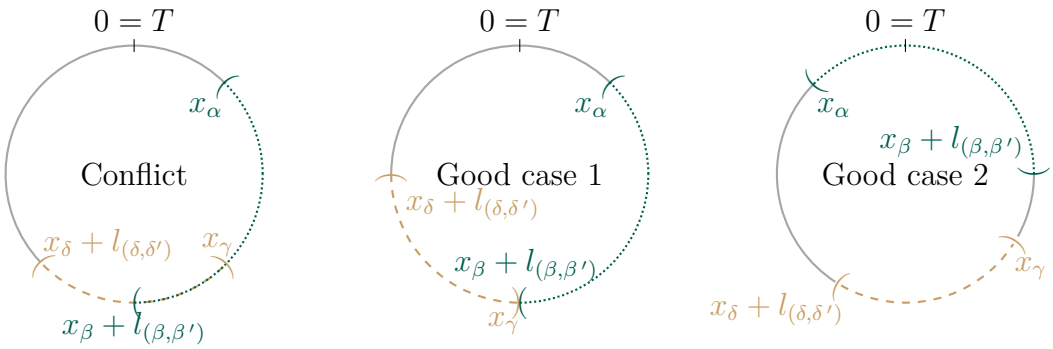

Figure 2: Clock Visualization
The two modulo intervals from (3) are put on the circle; see [26, p. 554]. Their intersection must be empty.

properties. Since actually all production steps are to be finished within the period time T, the upper bounds can always be assumed to range within $1 \leq \mathring{u}_a \leq T-1$ if they exist. Larger values otherwise yield inactive constraints.

Period homogeneity. Furthermore, all robots have to run through their states with the same period. Formally,

$$\sum_{a \in A_r} \mathrm{d}(a) = T \quad \text{for } r \in R. \tag{2}$$

This is different to train timetabling, where several cycle periods can be wanted. In a production line, a robot cannot proceed with a whole new run when others are still working on the old item. Of course, a robot's job may be to, e.g., attach two pieces to one item. However, then a cycle needs the two distinctive operations to be completed in one run.

Collision avoidance. Finally, the simultaneous execution of certain subsequences of state transitions may cause conflicts in pairs of robots, leading to collision avoidance constraints. For two robots $r, s \in R$, $r < s$, these restrictions are given by a subset

$$\mathcal{C}_{r,s} \subset \Big\{ \{(\beta,\gamma),(\delta,\alpha)\} \mid \alpha,\beta \in V_r,\ \gamma,\delta \in V_s \Big\},$$

where an element $\{(\beta,\gamma),(\delta,\alpha)\} \in \mathcal{C}_{r,s}$ encodes that any actions of robot r following state α up to state β' and any concurrent actions of robot s following state γ up to state δ' are forbidden. With the modulo-inspired interval notation for times i, j,

$$(i,j)\big|_T := \begin{cases} (i,j) & \text{if } i \leq j < T, \\ (i,T) \cup [0, j \bmod T) & \text{if } i < T \leq j, \\ (i,T) \cup [0,j) & \text{if } j < i < T, \end{cases}$$

the collision avoidance constraints are given by the expression

$$(x_\alpha, x_\beta + l_{(\beta,\beta')})\Big|_T \cap (x_\gamma, x_\delta + l_{(\delta,\delta')})\Big|_T = \emptyset \quad \text{for } \{(\beta,\gamma),(\delta,\alpha)\} \in \mathcal{C}_{r,s},\ r,s \in R. \tag{3}$$

Keep in mind that the second-listed state in these pairs is the one before the relevant border state, an arrival time like $x_\beta + l_{(\beta,\beta')}$ is specified in the intervals' right limit. Additional waiting time in a collision-free state β' should no longer block motion, so $x_{\beta'}$ would be too late. Again, all such robot pair sets are collected in $\mathcal{C} := \dot{\bigcup}_{r,s\in R} \mathcal{C}_{r,s}$.

One way to think of the above interval notation is the clock visualization, which already [26] made use of. Condition (3) then means that the time intervals in which certain operations are to be performed must not overlap, which is illustrated in Figure 2. Moreover, if in practice two points on the respective trajectories give rise to a collision, then this is also considered to happen close by, whence the intervals are open. As a consequence, condition (3) for example does not interdict the robots being in their border states α and γ at the same time, whereas moving from state α to state α' and beyond until state β' is not allowed when the other robot is moving from state γ to state δ'. After all, the set structures described can be summarized as follows.

Definition 2 (IPEC)**.** The tuple $(R, V, \mathring{A}, \mathcal{C}, \mathring{l}, \mathring{u})$ is the Iso-Periodic Event Configuration.

The term "Iso-Periodic" refers to the periodic structure of the operations that have to be performed as well as to the requirement that all robots in a production line have to work with the same frequency. In this notation, the scheduling problem reads as follows.

Definition 3 (CTMP)**.** For an IPEC, the Cycle Time Minimization Problem is

$$\begin{array}{ll} \text{minimize} & T \\ \text{subject to} & (1), (2), (3), \\ & T \in \mathbb{N},\ x \in \{0, \dots, T-1\}^V. \end{array}$$

Whereas this original problem formulation reveals the motivation of its constraints, there is a more developable and concise formulation, which is related to the PESP. The graph G in Definition 1 contains the structure of the set of constraints, and it is a helpful tool for visualizing instances of these problems. As in other applications, see [26] or [15], the set structures underlying the CTMP can also be interpreted graph-theoretically. When looking at the IPEC, $(V, \mathring{A})$ consists of several isolated simple closed walks, i.e. it is a disconnected graph. Our inequalities (1) obviously are of the same type as the time window constraints of the PESP. In contrast, the

constraints (3) do not fit into the preferable scheme. Consequently, the formulation should be made more tractable. A closer look unveils that $\mathcal{C}$ consists of pairs of edges between states of two different robots. These will be added to the IPEC graph, connecting the individual robot cycles. Hence, define

$$A := \mathring{A} \cup \bigcup_{\{c_1,c_2\}\in\mathcal{C}} \{c_1, c_2\}.$$

The lower and upper bounds have been defined for all edges in $\mathring{A}$. The new edges $c \in A \setminus \mathring{A}$ will get bounds too, coming from the collision avoidance constraints. The vectors $\mathring{l}, \mathring{u} \in \mathbb{N}^{\mathring{A}}$ are extended to vectors $l, u \in \mathbb{N}^A$ by defining the value for any edge $c \in \{c_1, c_2\} \in \mathcal{C}$ via

$$l_c := \mathring{l}_{(\beta,\beta')} \quad \text{and} \quad u_c := T \quad \text{for } c = (\beta, \gamma) \in A \setminus \mathring{A}. \tag{4}$$

The rationale behind this choice is that robot s may continue from state $\gamma = \gamma_s$ only if robot r has reached state $\beta' = \beta'_r$, i.e., in requiring (4), we guarantee

$$x_\gamma \geq x_\beta + \mathring{l}_{(\beta,\beta')},$$

linking the starting times of one robot to the ones of another. In order to reformulate the CTMP within the PESP framework it will be convenient to associate a particular circuit with each pair of collision constraint arcs. To build this, we add to the collision constraint arcs two directed paths, leading on the respective robot cycle from the head of one collision constraint arc to the tail of the other one. So, for each of the elements $\{(\beta_r, \gamma_s), (\delta_s, \alpha_r)\} \in \mathcal{C}$, we extract the directed paths

$$A^r_{\alpha,\beta} := \{(\alpha, \alpha'), \ldots, (\beta', \beta)\} \subset A_r,$$

$$A^s_{\gamma,\delta} := \{(\gamma, \gamma'), \ldots, (\delta', \delta)\} \subset A_s$$

and define the *blocking circuit* as the simple closed walk

$$D(\{(\beta_r, \gamma_s), (\delta_s, \alpha_r)\}) := \{(\beta_r, \gamma_s)\} \cup A^s_{\gamma,\delta} \cup \{(\delta_s, \alpha_r)\} \cup A^r_{\alpha,\beta}.$$

Remark 4. The definition of the robot cycle segments is a bit sloppy. Of course, the segment also could consist of only one edge ($\beta = \alpha'$) or even might be empty ($\beta = \alpha$). This advances technical details that often can be disregarded in return for easier notation. The definition of the circuit is correct in any case.

Also note that the edges of such a blocking circuit have all the same orientation and for practically relevant settings, we may assume $\alpha \leq \beta$ and $\gamma \leq \delta$ in any

collision edge pair. This restriction is not mandatory, but further simplifies certain explanations. In consequence, there is no blocking in all of the robots' first states, and the existence of a schedule is granted for T large enough by just letting the robots move one after the other. This assumption is justified by the industrial habit of a *home position* for security reasons. We do not need to prohibit collision segments extending over the modulus in our investigations, but be aware that then contradicting constraints could emerge, preventing the success of the search for a solution.

We finally remark that a blocking circuit can also originate from technological precedence relations rather than from the need to avoid physical collisions. The model for such interdependencies is the same.

The special knowledge on the collision edge pairs is now stored in the collective set of inter-linking circuits

$$\mathcal{D} := \dot{\bigcup_{C \in \mathcal{C}}} D(C),$$

which can be understood as practically motivated, additional information. In summary, the collision information has been modeled directly into the graph structure, while the problem characteristics are kept available in distinguishing certain substructures through the blocking circuit set.

Definition 5 (IPEN)**.** For an IPEC, the associated Iso-Periodic Event Network is the tuple $(R, V, A, \mathcal{D}, l, u)$.

The general structure of the resulting graphs is illustrated in Figure 3. Condition (4) for example requires that arc c_1 gets the same lower bound as arc a. The introduced circuit D for a collision edge pair $C = \{c_1, c_2\} \in \mathcal{C}$ is also highlighted at the sample network. Based on such data, we then regard the following problem.

Definition 6 (IPESP)**.** For an IPEN, the Iso-Periodic Event Scheduling Problem is

$$\begin{aligned}
\text{minimize} \quad & T \\
\text{subject to} \quad & l_a \le x_\beta - x_\alpha + T\, p_a \le u_a && \text{for } a = (\alpha, \beta) \in A, \\
& \sum_{a \in A_r} p_a = 1 && \text{for } r \in R, \\
& \sum_{d \in D} p_d = 1 && \text{for } D \in \mathcal{D}, \\
& T \in \mathbb{N},\ x \in \{0, \dots, T-1\}^V,\ p \in \{0,1\}^A.
\end{aligned}$$

Note that in contrast to the classical PESP, the term $T\, p_a$ is quadratic, since T is not constant here, but it is well suited to represent the given practical scenario. We will see in an instant that the time windows for robot transitions are directly

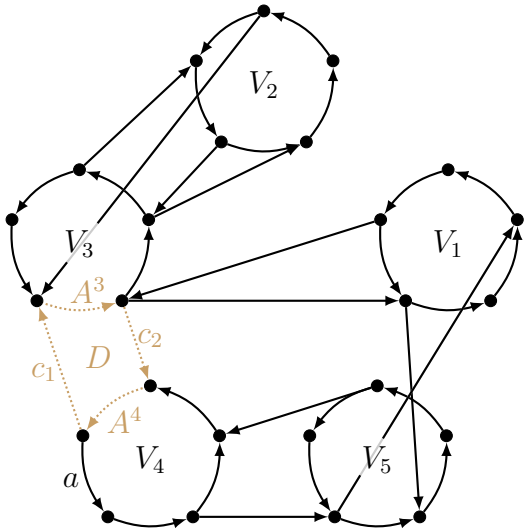

Figure 3: Graph of an IPEN

This illustration concerns a simple network of five robots. Here, robots 1 and 2 have no inter-links. In contrast, between robots 2 and 3, two collision edge pairs are added. For the pair of robots 3 and 4, the blocking circuit D is constructed from the collision edge pair $\{c_1, c_2\}$ by joining them with segments A^3 and A^4 of the robot cycles.

transferred from the CTMP to the IPESP. For this, recall our general assumption $u_a < T$. The period homogeneity is just rewritten, too. The collision avoidance is more challenging. Each pair of $\mathcal{C}$ gives a similar constraint for *blocking synchronization* and two new time windows for inter-linking switches.

Proposition 7. Solving the CTMP and the derived IPESP is equivalent.

Proof. Constraint (1) of the CTMP demands for $a = (\alpha, \beta) \in A_r$, $r \in R$, that

$$\begin{aligned} & l_a \le \mathrm{d}(a) \le u_a, \\ \Leftrightarrow\ & l_a \le (x_\beta - x_\alpha) \bmod T \le u_a, \\ \Leftrightarrow\ & l_a \le x_\beta - x_\alpha + T\,p_a \le u_a \quad \text{with a } p_a \in \{0,1\}. \end{aligned}$$

The last equivalence holds because $x \in \{0, \dots, T-1\}^V$ and hence $x_\beta - x_\alpha \in (-T+1, T-1)$. Since we assumed $u < T$ within the CTMP, the modulus T has to be added once, or there is nothing to correct. This argumentation covers all $a \in \mathring{A}$. The inter-linking arcs are included in a similar fashion later on.

Constraint (2) of the CTMP reads

$$T = \sum_{a \in A_r} \mathrm{d}(a) = \sum_{a=(\alpha,\beta) \in A_r} (x_\beta - x_\alpha + T\, p_a)$$
$$= \sum_{(\alpha,\beta) \in A_r} (x_\beta - x_\alpha) + T \sum_{a \in A_r} p_a = T \sum_{a \in A_r} p_a \quad \text{for } r \in R.$$

This is satisfied if and only if

$$\sum_{a \in A_r} p_a = 1 \quad \text{for } r \in R.$$

Notice that the latter means that exactly one edge of a robot cycle has an offset variable $p_a = 1$, while the others are stuck to $p_a = 0$.

Finally, constraint (3) of the CTMP requires that for $C = \{(\beta, \gamma), (\delta, \alpha)\} \in \mathcal{C}_{r,s}$, $r, s \in R$,

$$(x_\alpha, x_\beta + l_{(\beta,\beta')})\Big|_T \cap (x_\gamma, x_\delta + l_{(\delta,\delta')})\Big|_T = \emptyset.$$

The comparison with the IPESP notation needs a case differentiation that replaces the modulo intervals above.

Case 1: $x_\alpha > x_\beta + l_{(\beta,\beta')}$ and $x_\gamma > x_\delta + l_{(\delta,\delta')}$. This is impossible, because otherwise, falling back to the last two lines of the definition of the interval notation, 0 is contained in both intervals, violating condition (3). Likewise, this case is not possible in an IPESP formulation. Because of $x_\alpha > x_\beta$, the "date line" for robot r's cycle (at which the modulo time jump takes place) is somewhere on the path from α to β. That is, one of the arcs $a \in A^r_{\alpha,\beta}$ must have a non-vanishing offset variable $p_a = 1$. (Keep in mind that $A^r_{\alpha,\beta} \neq \emptyset$ due to $\alpha \neq \beta$.) Analogously, the other inequality implies the existence of an $b \in A^s_{\gamma,\delta}$ with $p_b = 1$. This would lead to the contradiction

$$1 = \sum_{d \in D(C)} p_d \geq p_a + p_b = 2.$$

Case 2: $x_\alpha \leq x_\beta + l_{(\beta,\beta')}$ and $x_\gamma \leq x_\delta + l_{(\delta,\delta')}$. Here, we have $x_\alpha < x_{\alpha'} < \ldots < x_\beta$, which is equivalent to $p_a = 0$ for all $a \in A^r_{\alpha,\beta}$, and $x_\gamma < x_{\gamma'} < \ldots < x_\delta$, which is equivalent to $p_a = 0$ for all $a \in A^s_{\gamma,\delta}$ (if there are some). While all starting times have values less than T due to the problem definition, the arrival times may exceed the period time. Similar to the left bound of the intervals, also the right bounds $x_\beta + l_{(\beta,\beta')}$ and $x_\delta + l_{(\delta,\delta')}$ have to be differentiated in accordance with the modulo interval definition.

Case 2a: Both do not exceed T. Referring to Figure 2 middle for an illustration, condition (3) then simply is

$$(x_\alpha, x_\beta + l_{(\beta,\beta')}) \cap (x_\gamma, x_\delta + l_{(\delta,\delta')}) = \emptyset,$$

which is equivalent to that

$$\text{either} \quad x_\beta + l_{(\beta,\beta')} \leq x_\gamma \quad \text{or} \quad x_\delta + l_{(\delta,\delta')} \leq x_\alpha.$$

These inequalities in fact are the additional IPESP time window constraints (for edges in $A \setminus \mathring{A}$) with zero offset. Reminding that exactly one of them is valid, there is one p equal to zero, while the reversed inequality enforces the other p to be one. In consequence,

$$\sum_{d \in D(C)} p_d = \sum_{c \in C} p_c = 1 \quad \text{and} \quad l_{(\beta,\gamma)} \leq x_\gamma - x_\beta + T\, p_{(\beta,\gamma)}, \quad l_{(\delta,\alpha)} \leq x_\alpha - x_\delta + T\, p_{(\delta,\alpha)}.$$

This is, in view of Definition 5, what was to be shown.

Case 2b: One is less than T, the other bigger or equal. If, without loss of generality, $x_\beta + l_{(\beta,\beta')} \geq T$, the first interval is split into two parts:

$$\Big[(x_\alpha, T) \cup [0, x_\beta + l_{(\beta,\beta')} \bmod T)\Big] \cap (x_\gamma, x_\delta + l_{(\delta,\delta')}) = \emptyset.$$

This is only realizable when

$$x_\beta + l_{(\beta,\beta')} - T \leq x_\gamma \quad \text{and} \quad x_\delta + l_{(\delta,\delta')} \leq x_\alpha,$$

so, we again have one edge of the collision pair with $p_{(\beta,\gamma)} = 1$ and one with $p_{(\delta,\alpha)} = 0$. The ordering of the intervals we encountered in Case 2a appears similarly, but is fixed automatically depending on which time jumps.

Case 2c: Both are at least T. As in Case 1, the intersection can only be non-empty; both intervals extend to T. But here, arcs on the robot cycles not necessarily exist, or their offset variables vanish anyway. The contradiction must hence be based on the collision edge pair. Considerations like in Case 2b yield $p_{(\beta,\gamma)} = 1$ and $p_{(\delta,\alpha)} = 1$ here.

Case 3: $x_\alpha > x_\beta + l_{(\beta,\beta')}$ and $x_\gamma \leq x_\delta + l_{(\delta,\delta')}$ or $x_\alpha \leq x_\beta + l_{(\beta,\beta')}$ and $x_\gamma > x_\delta + l_{(\delta,\delta')}$. Due to symmetry considerations, we restrict ourselves to the first situation. Analyzing the inequalities, we see that this case occurs exactly when $p_a = 1$ for some $a \in A^r_{\alpha,\beta} \neq \emptyset$ and $p_a = 0$ for all $a \in A^s_{\gamma,\delta}$. Again, we need to split with respect to the right bound of the interval. But this time, it can occur only for the second constraint.

Case 3a: $x_\delta + l_{(\delta,\delta')} < T$. This is also illustrated in the right image of Figure 2. Condition (3) then becomes

$$\Big[(x_\alpha, T) \cup [0, x_\beta + l_{(\beta,\beta')})\Big] \cap (x_\gamma, x_\delta + l_{(\delta,\delta')}) = \emptyset,$$

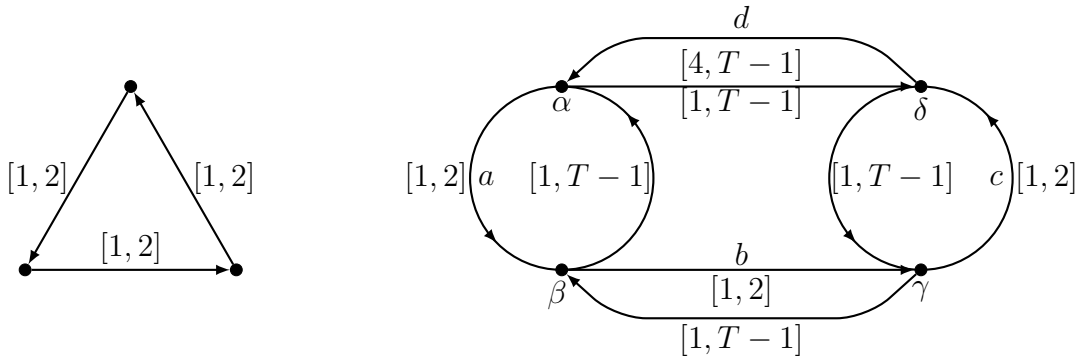

Figure 4: Illustrations to Problems with Constant Upper Bounds
The depicted graph structures demonstrate feasibility issues that arise when relaxing the condition of non-constant upper bounds in Proposition 8, as pointed out in Remark 10. In the pictures, lower and upper bounds are assigned to the arcs in form of time intervals.

looking almost like Case 2b. We easily see $p_{(\beta,\gamma)} = 0$ and $p_{(\delta,\alpha)} = 0$ in the inequalities here and get as desired

$$\sum_{d \in D(C)} p_d = \sum_{a \in A^r} p_a = 1 \quad \text{and} \quad l_{(\beta,\gamma)} \leq x_\gamma - x_\beta + T\, p_{(\beta,\gamma)}, \quad l_{(\delta,\alpha)} \leq x_\alpha - x_\delta + T\, p_{(\delta,\alpha)}.$$

Case 3b: $x_\delta + l_{(\delta,\delta')} \geq T$. Obviously, the intersection of the intervals would be non-empty. Recalling Case 2c, we obtain $p_{(\delta,\alpha)} = 1$, so that the blocking synchronization sum is two. In both formulations, this case was excluded. □

Note that in contrast to the classical PESP our formulation also contains the objective to minimize T. This causes nonlinearities $T\, p_a$ in the constraints. In the implementations, we accomplish this by solving the corresponding feasibility problem for a given T and employing binary search for reducing the period time. This approach is based on the following observation.

Proposition 8. Consider an IPESP instance having upper bounds $u_a = T - c_a$ with $c_a \in \mathbb{N}$ for each $a \in A$. A solution $x \in \{0, \ldots, T-1\}^V$, $p \in \{0,1\}^A$ of the corresponding feasibility problem for a fixed $T \in \mathbb{N}$ remains feasible for $u_a = T' - c_a$ with $T' = T + 1$.

Proof. Let $x \in \{0, \ldots, T-1\}^V$, $p \in \{0,1\}^A$ be a solution for a given $T \in \mathbb{N}$, as required. Then this solution also satisfies the time window constraints for $T' = T + 1$, because

$$l_a \leq x_\beta - x_\alpha + T\, p_a < x_\beta - x_\alpha + T' p_a \quad \text{and}$$
$$x_\beta - x_\alpha + T' p_a = x_\beta - x_\alpha + T\, p_a + p_a \leq T - c_a + p_a \leq T + 1 - c_a = T' - c_a.$$

In view of Definition 6, nothing else remains to be shown. □

Remark 9. In our application, all upper bounds can be set to $u_a = T - 1$. So any feasible solution of the IPESP for some T is also valid for any larger time. Thus there is a monotonous behavior when varying T, which allows to apply the search. Actually, for real-world data sets the bounds often can be reduced even more to $u_a = T - \sum_{a \in K} l_a$ where the index set $K \subset A$ describes the complement of some directed circuit containing a. Obviously, if a is part of an individual robot cycle, it must reserve enough time for the remaining state transitions.

Remark 10. Relaxing the condition of non-constant upper bounds in Proposition 8 might cause feasibility issues. Consider the two examples given in Figure 4. An IPESP having the triangle on the left as a subgraph may be feasible for $T = 6$, but cannot have a feasible solution for $T = 7$. Even worse, having non-constant upper bounds in every closed walk is still not sufficient. The occurrence of just a few constantly bounded arcs may cause feasibility issues, as illustrated by the instance on the right of Figure 4. For example, it is solvable for $T = 5$ with starting times $x_\alpha = 0$, $x_\beta = 2$, $x_\gamma = 4$ and $x_\delta = 1$, but it is infeasible for $T = 6$. Starting w.l.o.g. in state α at time $x_\alpha = 0$ implies $x_\beta \in [1, 2]$ and thus $x_\gamma \in [2, 4]$ and $x_\delta \in [3, 6) \cup \{0\}$ by the bounds of arcs a, b and c, respectively. Taking into account the constraint of arc d yields $x_\alpha \in [1, 5]$, contradicting $x_\alpha = 0$.

Remark 11. An alternative to the binary search approach presented here, is given by [27]. They treat T as a real variable in an interval $[0, U]$ with U chosen sufficiently large and handle the quadratic term $T\, p_a$ by linearizing the time window constraints

$$l_a \leq x_\beta - x_\alpha + T\, p_{(\alpha,\beta)} \leq u_a \quad \text{for } a = (\alpha, \beta) \in A.$$

As described by [29], a term $T\, p$ containing a binary variable p and a real variable T can be rewritten by introducing a new variable $z = T\, p$ subject to the constraints

$$0 \leq z \leq Up \quad \text{and} \quad T - U(1 - p) \leq z \leq T.$$

This allows to reformulate the associated cycle time minimization problem as a mixed-integer linear program that can be solved directly. However, in contrast to the binary search approach, the above linearization introduces a number of additional variables and constraints to the model. Furthermore, the focus in [27] is on the PESP and not on the frequently used CPF. This is probably due to the fact that the integer variables in the latter model are not binary and thus its constraints are more challenging to linearize. Nevertheless, comparing these approaches in both underlying applications (train timetabling and robot scheduling) is an interesting question for further research.

3 The IPESP and its Properties

This section clarifies the complexity status of the IPESP, establishes a Cycle Periodicity Formulation for it and closes with bounds for the integer offset variables in this formulation. It is well known that the original PESP is *NP*-complete; see [25]. But the IPESP involves more structural requirements on the underlying constraint graph than the classical PESP and has additional conditions on the integer offset variables. Thus, the IPESP represents a special case of the PESP and consequently does not automatically have to be *NP*-complete as well. However, the following statement clears this matter.

Theorem 12. The feasability version of the IPESP is *NP*-complete.

For the proof of this statement it is useful to recall the following definition.

Definition 13 (k-Coloring Problem)**.** For a simple graph $G = (V, E)$, the k-Coloring Problem asks for an assignment $f\colon V \to \{1, \dots, k\}$ satisfying

$$f(v_1) \neq f(v_2) \quad \text{for each } \{v_1, v_2\} \in E.$$

The k-Coloring Problem is one of Karp's [12] classical *NP*-complete problems. The idea of reducing the PESP to this problem can be found in [25], and it can also be adopted to establish the *NP*-completeness of the IPESP. Together with some transformation ideas, which are also illustrated in Figure 5, this is presented next.

Proof of Theorem 12. We show that for each given graph G one can construct an instance $(R, V, A, \mathcal{D}, l, u)$ of the IPESP that has a periodic solution in $\{0, \dots, T-1\}$ if and only if G has a valid coloring with T colors. The process is explained in the following. Corresponding to any node v_r in G, we create a simple closed 2-walk $V_r := \{v_r, v_r'\}$, $A_r := \{(v_r, v_r'), (v_r', v_r)\}$, which represents one robot cycle with minimum number of states. Now take the edges $\{v_r, v_s\}$ of G and set $\mathcal{C}_{r,s} := \{\{(v_r, v_s), (v_s, v_r)\}\}$. (The collision edge pair touches the robot cycles in only one point. There are no segments on the robot cycles.) The sets V, A, and $\mathcal{C}$ or $\mathcal{D}$ then can be built by collecting the elementary parts in union sets as described in Section 2. By choosing $l_a := 1$ as well as $u_a := T$ for each $a \in A$, this constitutes an IPEN.

Taking one of the feasible solutions $x \in \{0, \dots, T-1\}^V$ and $p \in \{0, 1\}^A$ of the associated IPESP for prescribed T, one in particular gets for each collision pair set $\mathcal{C}_{r,s}$,

$$1 \leq x_s - x_r + T\, p_{(r,s)}, \quad 1 \leq x_r - x_s + T\, p_{(s,r)}, \quad p_{(r,s)} + p_{(s,r)} = 1,$$

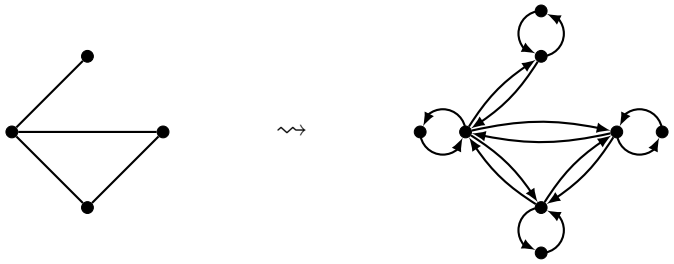

Figure 5: Construction Principles for Relating IPESP to the T-Coloring Problem
The nodes of the T-coloring graph (left) are replaced by 2-cycles and the edges by a pair of opposite edges.

where all the states v_r are directly identified with the robot number $r \in R$. Similarly looking constraints for arcs on the robot cycles exist too, as all A_r have the same two-edged shape. However, these are not important for the aim, and the v_r' are no longer of interest. Because of $x \in \{0, \ldots, T-1\}^V$, the collision constraints are equivalent to

$$x_s \neq x_r \bmod T \quad \text{for } \{v_r, v_s\} \in E.$$

This defines a T-coloring on G, since all neighbored nodes bear different discrete starting times that are in one-to-one correspondence with the available color palette.

Conversely, for a given T-Coloring of G with colors $\{0, \ldots, T-1\}$, one can assign to the variable $x_r = x_{v_r}$ the corresponding color value of v_r and set for the second robot states $x_{v_r'} := (x_r + 1) \bmod T$. Furthermore, allocate offset one to the reverse edge of every pair between two nodes by putting

$$p_{(\alpha,\beta)} := \begin{cases} 1 & \text{if } x_\alpha > x_\beta, \\ 0 & \text{if } x_\alpha < x_\beta \end{cases} \quad \text{for } (\alpha, \beta) \in A \quad (\text{i.e. in } A_r \text{ or } C \in \mathcal{C}_{r,s}).$$

Thanks to the construction, the assignment is uniquely determined. The chosen values satisfy $1 \leq x_\beta - x_\alpha + T\, p_{(\alpha,\beta)} \leq T$ for $a = (\alpha, \beta) \in A$. Moreover, it is $D(C) = C$ and thus, for $C = \{(v_r, v_s), (v_s, v_r)\} \in \mathcal{C}$, it follows

$$\sum_{d \in D(C)} p_d = p_{(r,s)} + p_{(s,r)} = 1.$$

Analogously, $\sum_{a \in A_r} p_a = 1$ is implied for $r \in R$. So, one has a valid solution for the instance $(R, V, A, \mathcal{D}, l, u)$. By this, the reduction is complete. □

After all, as with the PESP, we cannot hope for an efficient algorithm solving the IPESP in general, unless $P = NP$. Nevertheless, some techniques were developed

that are helpful to solve the PESP satisfactory in the light of the underlying application. We transfer these approaches to the IPESP. One of the ideas is based on considering tension variables instead of node potentials, which goes back to [21].

Definition 14 (Periodic tension)**.** Given a directed graph $G = (V, A)$, a node potential $x \in \{0, \dots, T-1\}^V$, and some $T \in \mathbb{N}$, a vector $y \in \mathbb{N}^A$ is called periodic tension if

$$\exists\, p \in \{0,1\}^A \;\forall\, a = (\alpha, \beta) \in A: \quad y_a = x_\beta - x_\alpha + T\, p_a.$$

With this, the time window inequalities can be interpreted as equalities—the usual Cycle Periodicity Constraints. The period homogeneity and the blocking synchronizations become constraints of the same (but even simpler) shape.

Lemma 15 (CPCs)**.** Let $(R, V, A, \mathcal{D}, l, u)$ be an IPEN and $x \in \{0, \dots, T-1\}^V$ with $p \in \{0,1\}^A$ be the solution of the corresponding IPESP. Then the related periodic tension $y \in \mathbb{N}^A$ satisfies the Cycle Periodicity Constraints

$$\sum_{b \in B_+} y_b - \sum_{b \in B_-} y_b = T\, q_B \quad \text{for a } q_B \in \mathbb{Z} \qquad \text{for simple closed walks } B \text{ in } G = (V, A),$$

$$\sum_{a \in A_r} y_a = T \qquad \text{for } r \in R,$$

$$\sum_{d \in D} y_d = T \qquad \text{for } D \in \mathcal{D}.$$

In this, B_+ denotes the set of forward arcs and B_- the set of backward arcs when choosing an arbitrary orientation for the simple closed walk B.

Note that by construction of the IPEN, we actually have $y \in \{1, \dots, T-1\}^A$.

Proof. Let B be an arbitrary simple closed walk in G. Then

$$\sum_{b \in B_+} y_b - \sum_{b \in B_-} y_b \;=\; \sum_{b=(\alpha,\beta) \in B_+} (x_\beta - x_\alpha + T\, p_b) - \sum_{b=(\alpha,\beta) \in B_-} (x_\beta - x_\alpha + T\, p_b)$$

$$= T \left(\sum_{b \in B_+} p_b - \sum_{b \in B_-} p_b \right) = T\, q_B \quad \text{by choosing } q_B := \sum_{b \in B_+} p_b - \sum_{b \in B_-} p_b \in \mathbb{Z}.$$

Recall that for $r \in R$ the cycle A_r is directed. So,

$$\sum_{a \in A_r} y_a \;=\; \sum_{a=(\alpha,\beta) \in A_r} (x_\beta - x_\alpha + T\, p_a) \;=\; T \sum_{a \in A_r} p_a \;=\; T,$$

and an analogous statement is obtained for every circuit $D \in \mathcal{D}$. □

These observations lead to an alternative formulation of the IPESP, because [21] showed that it suffices to require the CPCs on only some selected simple closed walks—a so-called fundamental cycle basis—in order to guarantee the validity in general. [16, 17] extended the result by proving that any integral cycle basis of the PESP constraint graph will serve the same purpose. Strictly speaking, the previous publications even investigated a broader setting with real-valued node potentials and integer offsets. Our additional restrictions do not harm the transformation procedure from the IPESP to the upcoming formulation in Definition 16 and back. In the end, since the dimension of the cycle space of a graph $G = (V, A)$ is $\nu = |A| - |V| + 1$, just ν Cycle Periodicity Constraints must be demanded in rewriting an IPESP equivalently.

Definition 16 (ICPF)**.** For an IPEN $(R, V, A, \mathcal{D}, l, u)$ let $\mathcal{B}$ be a set of simple closed walks in G so that $\mathcal{B} \cup \{A_r\}_{r \in R} \cup \mathcal{D}$ contains an integral cycle basis $\tilde{\mathcal{B}}$ of G. Then the corresponding IPESP Cycle Periodicity Formulation is

$$
\begin{aligned}
\text{minimize} \quad & T \\
\text{subject to} \quad & \sum_{b \in B_+} y_b - \sum_{b \in B_-} y_b = T\, q_B && \text{for } B \in \mathcal{B}, \\
& \sum_{a \in A_r} y_a = T && \text{for } r \in R, \\
& \sum_{d \in D} y_d = T && \text{for } D \in \mathcal{D}, \\
& l_a \le y_a \le u_a && \text{for } a \in A, \\
& T \in \mathbb{N},\ y \in \{1, \dots, T-1\}^A,\ q \in \mathbb{Z}^{\mathcal{B}}.
\end{aligned}
$$

An advantage of the ICPF over the previous IPESP formulation is its smaller number of integer offset variables. However, these cumulative variables are no longer binary. So it is reasonable to investigate bounds on the so-called cycle offset variables in this formulation. The following refinement of Odijk's [25] result provides cycle inequalities for the ICPF. For this, observe that each simple closed walk in an IPEN is composed of directed paths that are either part of some cycle A_r for an $r \in R$ or consist simply of an edge in some $C \in \mathcal{C}$. Since all these paths are directed, one can denote the set of forward directed arcs in B by

$$B_+ = P_1 \cup \dots \cup P_{m_+}$$

and the set of backward directed arcs by

$$B_- = Q_1 \cup \dots \cup Q_{m_-},$$

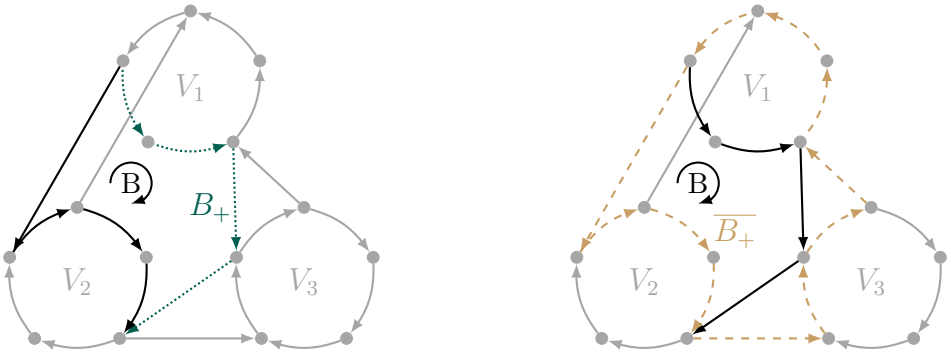

Figure 6: Complementary Structures
The simple closed walk B is split into B_+ (and B_-) depending on the direction of the arcs (left). The complement $\overline{B_+}$ is constructed by joining B_- with complementary walks of B_+'s components. For instance, P_1 and $\overline{P_1}$ are dotted, Q_1 and $\overline{Q_1}$ are dashed.

respectively. For such components P_k or Q_k, the complement in regard of the type is introduced as

$$\overline{P_k} := \begin{cases} A_r \setminus P_k & \text{if } P_k \subset A_r \text{ for an } r \in R, \\ D(C) \setminus P_k & \text{if } P_k \in C \text{ for an } C \in \mathcal{C}. \end{cases}$$

Finally, all these can be collected in the multisets

$$\overline{B_+} := B_- \cup \dot{\bigcup_{k=1}^{m_+}} \overline{P_k} \quad \text{and} \quad \overline{B_-} := B_+ \cup \dot{\bigcup_{k=1}^{m_-}} \overline{Q_k}.$$

The construction is illustrated in Figure 6. The complementary walks can be used in obtaining quite tight estimates.

Theorem 17. Regard the ICPF for an IPEN as stated in Definition 16. For any simple closed walk B in $\mathcal{B}$, the cycle offset variable $q_B \in \mathbb{Z}$ is bounded by

$$\left\lceil -m_- + \frac{1}{T} \sum_{a \in \overline{B_-}} l_a \right\rceil \leq q_B \leq \left\lfloor m_+ - \frac{1}{T} \sum_{a \in \overline{B_+}} l_a \right\rfloor .$$

Proof. By the cycle periodicity constraint for B, one gets

$$\begin{aligned} T\,q_B &= \sum_{b\in B_+} y_b - \sum_{b\in B_-} y_b = \sum_{k=1}^{m_+} \sum_{b\in P_k} y_b - \sum_{b\in B_-} y_b = \sum_{k=1}^{m_+} \left(T - \sum_{a\in\overline{P_k}} y_a \right) - \sum_{b\in B_-} y_b \\ &= m_+\,T - \sum_{k=1}^{m_+} \sum_{a\in\overline{P_k}} y_a - \sum_{b\in B_-} y_b = m_+\,T - \sum_{a\in\overline{B_+}} y_a \le m_+\,T - \sum_{a\in\overline{B_+}} l_a \quad \text{and} \\ T\,q_B &= \sum_{b\in B_+} y_b - \sum_{b\in B_-} y_b = \sum_{b\in B_+} y_b - \sum_{k=1}^{m_-} \sum_{b\in Q_k} y_b = \sum_{b\in B_+} y_b - \sum_{k=1}^{m_-} \left(T - \sum_{a\in\overline{Q_k}} y_a \right) \\ &= -m_-\,T + \sum_{k=1}^{m_-} \sum_{a\in\overline{Q_k}} y_a + \sum_{b\in B_+} y_b = -m_-\,T + \sum_{a\in\overline{B_-}} y_a \ge -m_-\,T + \sum_{a\in\overline{B_-}} l_a. \end{aligned}$$

The statement to be shown follows via dividing these inequalities by T and through the integrality of q_B. □

4 Computational Results

In order to study the practical usability of the PESP in the context of the described robot scheduling task, we implemented and compared the problem formulations from Definitions 6 and 16. Our implementations incorporate standard preprocessing procedures, which allow for example to eliminate nodes of degree one or two from the constraint graph. An overview of such techniques can be found in [14] or [5]. Of the several types of valid inequalities that have been proposed to bound PESP instance variables, we only employed the cycle type inequalities of Theorem 17. While including further types would strengthen the formulations, according to [28] or [18] most of them gave rise to ambivalent results concerning computation time. In view of the requirements of our industrial partners, the strength of inequalities is not the focus of this study. Rather, the computational results presented here are intended to illustrate the potential and limitations of the two conceptual approaches for realistic instance types and sizes.

For the analysis, we were provided with several instances originating from working industrial plants. The practical background is explained in [10] and the characteristics of the instances are summarized in Table 1. All of them have a comparable number of robots $|R|$. In typical applications, there are about seven to twelve robots commonly present in well-separable sealed production groups of higher complexity. However, one observes significant variations in the amount of inter-linking blocking constraints $|\mathcal{C}|$ ranging from 26 to 350. This is the result of different layouts of the facilities. Robots standing far away do not share much working space,

Instance	$\lvert R\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert V\rvert$	$\lvert A\rvert$	ν
r1	10	73	811	957	147
r2	11	26	1544	1596	53
r3	10	159	544	862	319
r4	11	197	641	1035	395
r5	11	197	620	1014	395
r6	11	222	930	1374	445
r7	9	187	448	822	375
r8	10	280	536	1096	561
r9	10	350	512	1212	701

Table 1: Real-World Instances

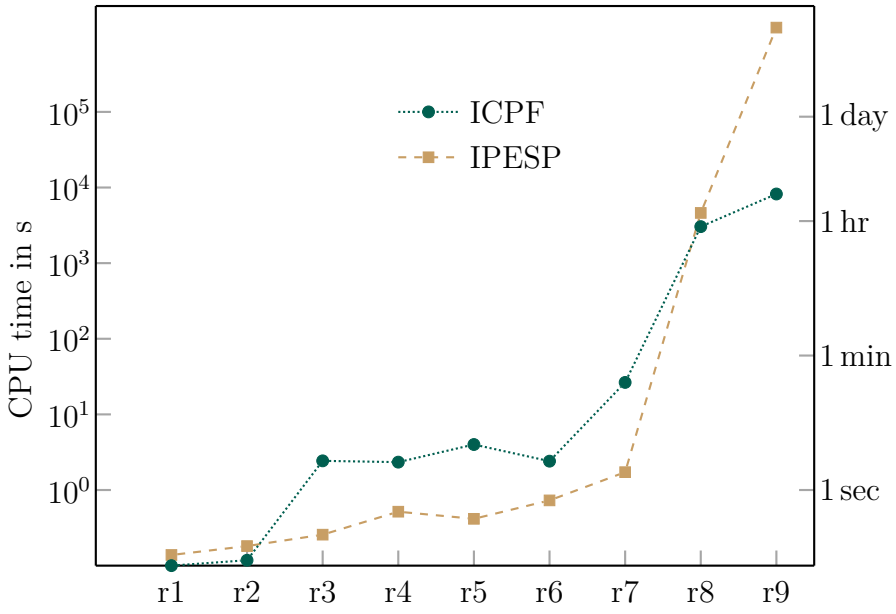

Figure 7: Computation Time of Real-World Instances

The times of the examples in IPESP and ICPF formulation were recorded on a 4×3400 MHz machine using Gurobi 7.5 for solving mixed-integer linear programs. Small instances (r1–r7) are treated in seconds, whereas more complex graphs (r8,r9) may take hours.

whereas a geometrically clustered positioning increases the potential for conflict situations. The actual sizes $|V|$ and $|A|$ result from practical factors like the robots' traces. But from the algorithmic perspective, the number of program points is less important, as successive path segments without any inter-links are merged within the preprocessing steps. The calculated periods T are in a range from 47 to 86 seconds. Similar cycle times are widespread in application, although also some processes exist that might consume several minutes.

Instance	$\lvert R\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert V\rvert$	$\lvert A\rvert$	ν
Gs1	5	200	125	525	401
Gs2	5	400	125	925	801
Gs3	5	600	125	1325	1201
Gm1	10	200	250	650	401
Gm2	10	400	250	1050	801
Gm3	10	600	250	1450	1201
Gl1	15	400	375	775	401
Gl2	15	800	375	1175	801
Gl3	15	1200	375	1575	1201

Table 2: Generated Instance Classes

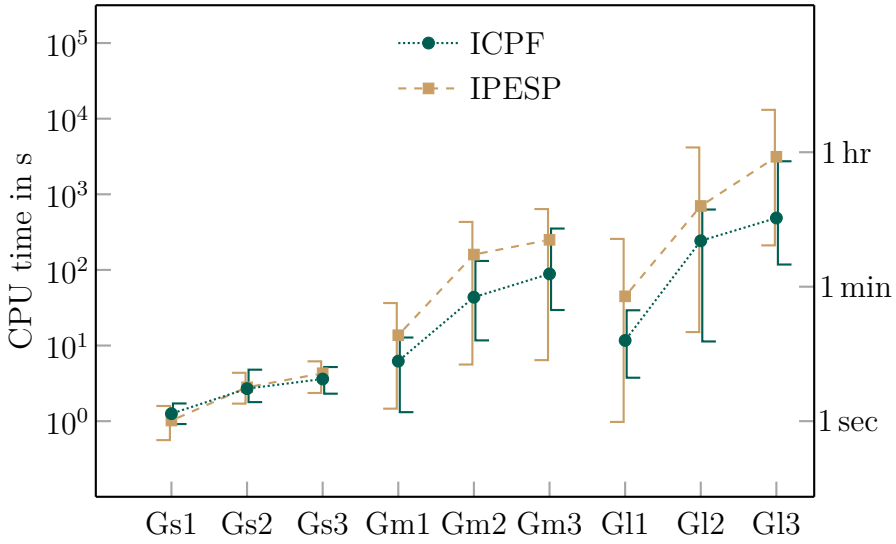

Figure 8: Computation Time of Generated Instances

For each of the instance classes ten random examples were generated and solved with the same hard- and software. The graph displays the mean as well as minimum and maximum of the computation time required for finding optimal solutions for each of the two problem formulations.

Table 1 also shows the respective cyclomatic number $\nu = |A| - |V| + 1$. This parameter is commonly assumed to correlate with the hardness of PESP instances; see [5]. According to the rating given in [19], one iteration of our binary search on instances r_1 and r_2 is easy, while r_3 to r_7 can be regarded as medium. Instances r_8 and r_9 are medium to hard, again to be solved several times. Indeed, the last two instances are the hardest real-world-instances that were available to us so far.

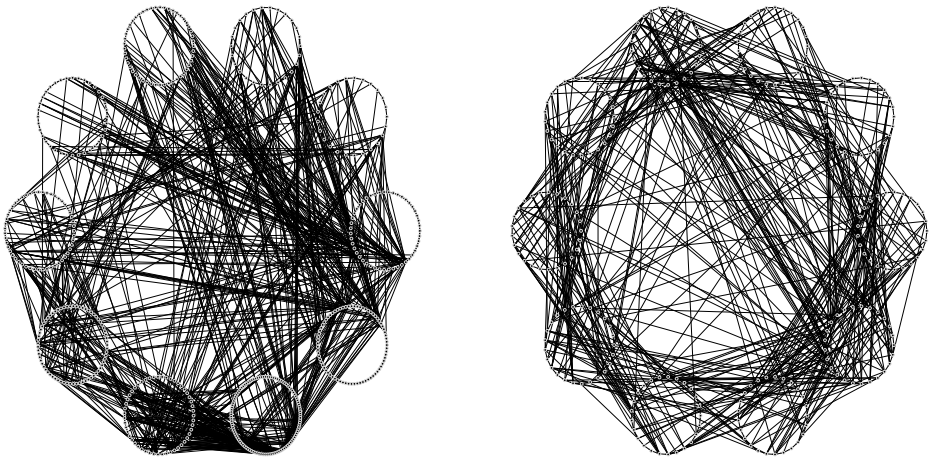

Figure 9: Real-World vs. Generated Instance

On the left is the graph of r8 from Table 1, which is an already demanding instance. Clearly, the many connections between the robot cycles induce a highly nonlinear structure unlike in other production models. The right image depicts one generated example of the moderate class Gm1 according to Table 2. Similar structures are visible in both; especially some heavier links to non-direct neighbors appear sporadically. In fact, the parameters matching r8 would likely put it into the class Gm2. This can also be seen by the darker image reflecting the larger number of connections.

The evaluation time for the nine examples is depicted in Figure 7. Apparently, about 200 blocking constraints are mastered with both formulations in very short time. In contrast, instance r8 with 280 added connection pairs was finished after some hours. The small costs for assembling the ICPF here are compensated by the reduced time for solving smaller-dimensional systems. An even more inter-linked graph as the one induced by r9 is only handled effectively as ICPF (consuming less than three hours), while the IPESP formulation did not find an end within two weeks. That is why the Cycle Periodicity Formulation (ICPF) is preferred for obtaining optimal solutions. Nevertheless, the first steps of the binary search in any formulation are processed quite fast. And similar to r8, after only three hours the solution range was already bounded to few seconds; more precisely, the time span then was restricted to ten percent of the optimal value.

In order to investigate the limits of the discussed approaches, we have set up an IPESP instance generator, which is available online along with the instances that are addressed in this section; see [11]. This generator, developed together with our industry partners, comprises parameters for the numbers of robots and their motion points, as well as the density and complexity of the desired instances. With the appropriate parameter configuration, realistic instances can be replicated (see Figure 9 for a comparison), and specific scenarios that have not yet been considered can be generated additionally.

Table 2 summarizes the randomized instance classes we used in our analysis. Each of them contains ten randomly generated samples for the specified parameters. The classes Gm1, Gm2, and Gm3 reflect the sizes of real-world instances. They are differentiated with respect to the amount of additional connectivity induced by the blocking constraints. The separation into little, average, and highly inter-linked instances was also introduced for smaller and larger numbers of robots, resulting in the classes Gs# and Gl# for numerical experiments. Clearly, the ICPF admits a narrower and lower range for the computation time. While for small instances this is of little importance, the r9-inspired medium classes substantiate the need for the reformulation. Furthermore, larger (generated) instances are undoubtedly tractable with a decent waiting time, too.

5 Conclusion and Outlook

We have introduced a special cycle time minimization problem that appears in a final stage of a more comprehensive design system for industrial robot production plants and showed in detail that this formulation is equivalent to an iso-frequency-variant of the PESP, which we call IPESP. This problem remains *NP*-complete, regardless of the restricted setting—in terms of synchronized individual periods and additional blocking relations compatible with the well-known cycle periodicity constraints. For the better computational treatment, reasonable bounds on the integer variables in the problem's cycle periodicity formulation could be identified. The study confirms the applicability of the model and implies that it should be usable even with larger-sized instances. The techniques established in other contexts motivated by different applications were successfully adapted to work also in the case of linked robot cycles with collision avoidance. Figures 7 and 8 illustrate that ICPF computations do not take more than three hours and are hence comparable to known train timetabling calculations. For the application at hand, this is an acceptable time.

From a computational perspective, it would be interesting to explore the linearization idea from Remark 11 as an alternative to the binary search approach on which our implementations are based on. The search interval containing the optimal solution is narrowed down suitably fast. Locating the exact value, however, can still be expensive, typically because almost feasible instances are hard to eliminate. In view of this and the recent article of [19], it may therefore be worth to consider adding their newly introduced flip inequalities, which generalize those inequalities that were reported most useful in practical computations. Additional separation strategies could be based on concepts developed in [2]. Because the bounds on the integer variables in the constraints representing the ICPF strongly depend on the

choice of the underlying cycle basis, a deeper understanding of integral cycle bases might help to further reduce the calculation time. The theoretical question on the complexity status of the minimum integral cycle basis problem is still open and an intriguing research topic by itself.

From a practical point of view, extending the model for sensitivity aspects is of interest, as this possibly allows to guidedly alter the plant if the cycle time aimed for is proven unreachable under the current design. For this and for safety as well as robustness reasons it would be helpful to identify among all solutions with optimal period time those for which distances between robots are maximized.

Acknowledgments

This research has been supported by the European Union project ERDF / SAB 100206299. We thank our collaboration partners Fraunhofer IWU and in particular Leadec Industrial Services for tests with real-world data as well as giving expertise in establishing models and analyzing results.

References

[1] Liping Bai, Naiqi Wu, Zhiwu Li, and MengChu Zhou. Optimal one-wafer cyclic scheduling and buffer space configuration for single-arm multicluster tools with linear topology. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46(10):1456–1467, 2016.

[2] Ralf Borndörfer, Heide Hoppmann, Marika Karbstein, and Niels Lindner. Separation of cycle inequalities in periodic timetabling. *Discrete Optimization*, 35:100552, 2020.

[3] Ralf Borndörfer, Niels Lindner, and Sarah Roth. A concurrent approach to the periodic event scheduling problem. *Journal of Rail Transport Planning & Management*, 15:100175, 2020.

[4] Matteo Fischetti and Andrea Lodi. Optimizing over the first Chvátal closure. *Mathematical Programming*, 110(1):3–20, 2007.

[5] Marc Goerigk and Christian Liebchen. An improved algorithm for the periodic timetabling problem. In *17th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[6] Rob M. P. Goverde. Railway timetable stability analysis using max-plus system theory. *Transportation Research Part B: Methodological*, 41(2):179–201, 2007.

[7] Peter Großmann, Steffen Hölldobler, Norbert Manthey, Karl Nachtigall, Jens Opitz, and Peter Steinke. Solving periodic event scheduling problems with SAT. In *International conference on industrial, engineering and other applications of applied intelligent systems*, pages 166–175. Springer, 2012.

[8] Refael Hassin. A flow algorithm for network synchronization. *Operations Research*, 44(4):570–579, 1996.

[9] Bernd Heidergott, Geert Jan Olsder, and Jacob Van Der Woude. *Max Plus at work: modeling and analysis of synchronized systems: a course on Max-Plus algebra and its applications*, volume 48. Princeton University Press, 2014.

[10] Tobias Hofmann and David Wenzel. How to minimize cycle times of robot manufacturing systems. *Optimization and Engineering*, pages 1–19, 2020.

[11] Tobias Hofmann and David Wenzel. *IPESP instance generator*. Chemnitz University of Technology, 2020.

[12] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[13] Telikepalli Kavitha, Christian Liebchen, Kurt Mehlhorn, Dimitrios Michail, Romeo Rizzi, Torsten Ueckerdt, and Katharina A. Zweig. Cycle bases in graphs characterization, algorithms, complexity, and applications. *Computer Science Review*, 3(4):199–243, 2009.

[14] Christian Liebchen. *Periodic timetable optimization in public transport*. dissertation.de, 2006.

[15] Christian Liebchen and Rolf H. Möhring. The modeling power of the periodic event scheduling problem: railway timetables—and beyond. In *Algorithmic methods for railway optimization*, pages 3–40. Springer, 2007.

[16] Christian Liebchen and L. Peeters. On cyclic timetabling and cycles in graphs. *Technical Report 761/2002, TU Berlin*, 2002.

[17] Christian Liebchen and L. Peeters. Integral cycle bases for cyclic timetabling. *Discrete Optimization*, 6(1):98–109, 2009.

[18] Christian Liebchen and Elmar Swarat. The second Chvátal closure can yield better railway timetables. In *8th Workshop on Algorithmic Approaches for*

Transportation Modeling, Optimization, and Systems (ATMOS'08). Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2008.

[19] Niels Lindner and Christian Liebchen. Determining all integer vertices of the PESP polytope by flipping arcs. Technical Report 20-19, ZIB, 2020.

[20] Thomas Lindner. *Train schedule optimization in public rail transport*. Technische Universität Braunschweig, 2000.

[21] Karl Nachtigall. *A branch and cut approach for periodic network programming*. Institut für Mathematik, 1994.

[22] Karl Nachtigall. Cutting planes for a polyhedron associated with a periodic network. *DLR Report*, 1996.

[23] Karl Nachtigall. Periodic network optimization and fixed interval timetables. *DLR Report*, 1998.

[24] Karl Nachtigall and Jens Opitz. A modulo network simplex method for solving periodic timetable optimisation problems. In *Operations Research Proceedings 2007*, pages 461–466. Springer, 2008.

[25] Michiel A. Odijk. *Construction of periodic timetables. Pt. 1. A cutting plane algorithm*. TU Delft, 1994.

[26] Paolo Serafini and Walter Ukovich. A mathematical model for periodic scheduling problems. *SIAM Journal on Discrete Mathematics*, 2(4):550–581, 1989.

[27] Daniel Sparing and Rob M. P. Goverde. A cycle time optimization model for generating stable periodic railway timetables. *Transportation Research Part B: Methodological*, 98:198–223, 2017.

[28] Jonas Christoffer Villumsen. *Construction of timetables based on periodic event scheduling*. Technical University of Denmark, 2006.

[29] Paul H. Williams. *Model building in mathematical programming*. John Wiley & Sons, 2013.

[30] Fajun Yang, Naiqi Wu, Yan Qiao, and Rong Su. Polynomial approach to optimal one-wafer cyclic scheduling of treelike hybrid multi-cluster tools via petri nets. *IEEE/CAA Journal of Automatica Sinica*, 5(1):270–280, 2017.