# On the Complexity of Optimal Parallel Cooperative Path-Finding

**Pavel Surynek**

*Charles University Prague*
*Faculty of Mathematics and Physics*
*Department of Theoretical Computer Science and Mathematical Logic*
*Malostranské náměstí 25, Praha, 118 00, Czech Republic*
pavel.surynek@mff.cuni.cz

***Abstract.*** A parallel version of the problem of cooperative path-finding (pCPF) is introduced in this paper. The task in CPF is to determine a spatio-temporal plan for each member of a group of agents. Each agent is given its initial location in the environment and its task is to reach the given goal location. Agents must avoid obstacles and must not collide with one another. The environment where agents are moving is modeled as an undirected graph. Agents are placed in vertices and they move along edges. At most one agent is placed in each vertex and at least one vertex remains unoccupied.

An agent can only move into a currently unoccupied vertex in the standard version of CPF. In the parallel version, an agent can also move into a vertex being currently vacated by another agent supposing the character of this movement is not cyclic.

The optimal pCPF where the task is to find the smallest possible solution of the makespan is particularly studied. The main contribution of this paper is the proof of *NP*-completeness of the decision version of the optimal pCPF. A reduction of propositional satisfiability (SAT) to the problem is used in the proof.

***Keywords*:** cooperative path-finding (CPF), parallelism, multi-agent, sliding puzzle, $(N^2-1)$-puzzle, N×N-puzzle, 15-puzzle, domain dependent planning, complexity, *NP*-completeness

## 1. Introduction and Motivation

This paper addresses the problem of *cooperative path-finding* (CPF) [17, 18, 22] and its parallel version. Consider a group of mobile agents that are moving in some environment (for example in the 2-dimensional plane with obstacles). Each agent in the group is given an initial and a goal location. The question of interest is to determine a sequence of moves for each agent such that all the agents reach their goal locations, supposing they started from the given initial ones, by following this sequence. Physical limitations must be observed: agents must **not collide** with one another and they must **avoid obstacles**.

The CPF problem is **motivated** by many practical tasks. Various problems regarding the navigation of a group of mobile agents can be formulated as CPF. However, the problem's pri-

Submitted to *Fundamenta Informaticae* on March 30, 2012. Reviews received on October 1, 2012. Revision submitted on January 21, 2013. Reviews of the revision received on April 17, 2014. Second revision submitted on June 7, 2014.

mary motivation is the need to solve tasks of relocating certain entities (autonomous or centrally controlled) within an environment with a limited free space. Hence, the problem is not limited to cases where agents are actually represented by mobile agents. Such real-life examples include rearranging of stored items in automated storages (an agent is represented by a movable pile with stored items – see Figure 1) or coordination of vehicles in dense traffic (agent = vehicle). Moreover, the reasoning about rearrangement/coordination tasks should not be limited to physical entities. An agent may be represented by a virtual entity or by a piece of commodity as well. Thus, many tasks such as the planning of a data transfer between communication nodes with limited storage capacity (agent = data packet), commodity transportation in the commodity transportation network (agent = certain amount of commodity), or even the motion planning of large groups of virtual agents in the computer-generated imagery can be expressed as an instance of CPF.

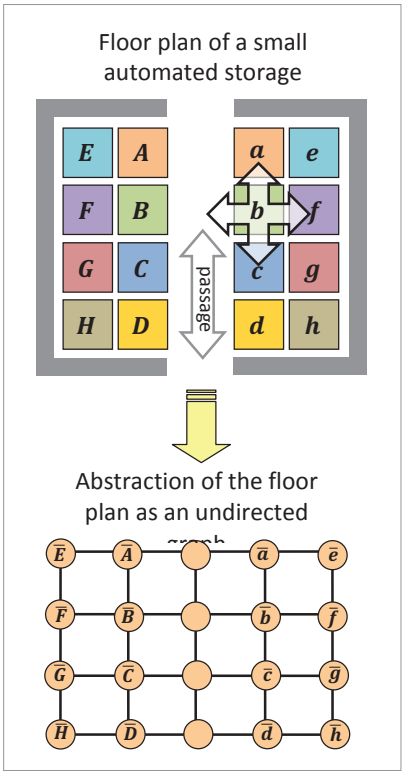



**Figure 1.** *An illustration of a real-scenario* ***modeling*** *of the environment by an undirected graph.* The scenario consists of a small automated storage with movable piles of stored items (labeled *A* to *H* and *a* to *h*). Each pile can be moved left/right/forward/backward. Items in piles are accessible from the passage – to access piles *E*-*H* or *e*-*h* the storage needs to be rearranged. The environment is modeled as a grid of the size 4 × 5, which is a bi-connected graph.

A parallel version of CPF (pCPF) is suggested in this paper and its computational complexity is studied. The standard CPF is usually formulated on an undirected graph that models the environment. The vertices of the graph represent locations and edges represent passable regions. Agents are placed in the graph's vertices and they are allowed to move into a neighboring vertex if it is currently **unoccupied.** The parallel version of CPF is more relaxed – an agent is also allowed to enter a vertex that is **simultaneously vacated** by another agent supposed that agents do not perform a cyclic movement (a cyclic movement includes a rotation along a cycle but also the swapping of a pair of agents along a single edge). In other words, there must exist an agent entering an unoccupied vertex, **leading** this simultaneous movement.

An abstract instance for a given specific real-life cooperative path-finding situation can be modeled in a variety of ways. For instance, it is necessary to sample locations in the original environment in order to make the abstract instance as precise as needed. Nevertheless, these issues fall outside the scope of this work.

The main contribution of this paper is the proof of *NP*-completeness of the optimal pCPF. This result was already noted in a short conference paper [26]. However, the paper was too short to cover the proof. In the present paper, the proof is presented with all the details, including the rigorous treatment and illustrations.

In the context of CPF, works on the problems of motion planning over graphs must be mentioned [13, 14, 15, 16, 37] since they are closely related. Namely, the so-called *pebble motion on graphs* (PMG), of which the most widely known representative is the *15-puzzle* [13, 15, 16, 37], in fact, represents the standard (non-parallel) CPF. Many theoretical results have been obtained for PMG – it is known that the problem can be

solved in a polynomial time (in $\mathcal{O}(|V|^3)$ for $G = (V, E)$ modeling the environment) with the solution consisting of a polynomial number of moves (again $\mathcal{O}(|V|^3)$ moves) [13, 37]. Moreover, the decision version of the optimal PMG (that is, a yes/no question if a solution of a given length/makespan exists) is known to be *NP*-complete [15, 16]. This result has been shown for a generalized variant of the 15-puzzle that is also known as the $(N^2 - 1)$-puzzle. Hence, the question naturally arises whether the situation changes in the case of pCPF. The present paper provides an answer.

The paper is organized as follows: a formal definition of PMG is recalled and a definition of pCPF is given in Section 2 (3). Some basic properties of both the problems and their correspondence are also discussed in this section. Section 3 (7) represents the core part of the paper – a description of several techniques for polynomial transformation of propositional satisfiability to pCPF. The last section – Section 4 (26) – contains an overview of related works, and the conclusion.

## 2. Pebble Motion on a Graph and Cooperative Path-finding

The problems of *pebble motion on a graph* (PMG) and *parallel cooperative path-finding* (pCPF) are formally defined in this section. As has been mentioned, non-parallel CPF and PMG are used to denote the same concept by many authors [13, 20, 37]. The PMG/CPF problem has been already studied in the literature and many theoretical results have been obtained for this problem. The *parallel* version of CPF represents a relaxation of PMG/CPF with respect to the dynamic character of the movements.

Consider an environment in which a group of mobile agents is moving. The agents are all identical (that is, they are all of the same size and have the same moving abilities). Each agent starts at a given initial location and it needs to reach a given goal location. Both problems consist in finding a spatial-temporal path for each agent so that it can reach its goal by following this path. Agents must **not collide** with one another and they must **avoid obstacles** in the environment.

An abstraction common in the literature related to PMG/CPF is adopted regarding the model of the environment [18, 20]. The environment containing obstacles, in which the agents are moving, is modeled as an **undirected graph**. Vertices of this graph represent locations in the environment and the edges model a passable way from one location to the neighboring location. The time is discrete – each agent is located in a vertex at each time step. The motion of an agent is an instantaneous event. If the agent is placed in a vertex at a given time step then the result of the motion is a situation where the agent is placed in the neighboring vertex at the following time step.

### 2.1. *Formal Definitions of Motion Problems*

The notion of *pebble motion on a graph* – PMG (also called the *pebble motion puzzle*, *sliding box puzzle*; special variants are known as the *15-puzzle* and $(N^2 - 1)$-*puzzle*) [13, 16, 37] and the related problems of *cooperative path-finding* – CPF (also known as *multi-agent path-finding*) [20, 25, 32] are described in the following definition.

**Definition 1** ***(pebble motion on a graph*– PMG).** Let $G = (V, E)$ be an undirected graph and let $P = \{\bar{p}_1, \bar{p}_2, \dots, \bar{p}_\mu\}$, where $\mu < |V|$, be a set of *pebbles*. The *initial arrangement* and the *goal arrangement* of pebbles in $G$ are defined by two uniquely invertible functions $S_P^0 : P \longrightarrow V$ (that is $S_P^0(p) \neq S_P^0(q)$ for every $p, q \in P$ with $p \neq q$) and $S_P^+ : P \longrightarrow V$, respectively. In the *pebble motion on a graph* (*PMG*) problem, the task is to find a number $\xi$ and a sequence of pebble arrangements $\mathcal{S}_P = [S_P^0, S_P^1, \dots, S_P^\xi]$ such that the following conditions hold (the sequence represents arrangements of pebbles at each time step – the time step is indicated by the upper index):

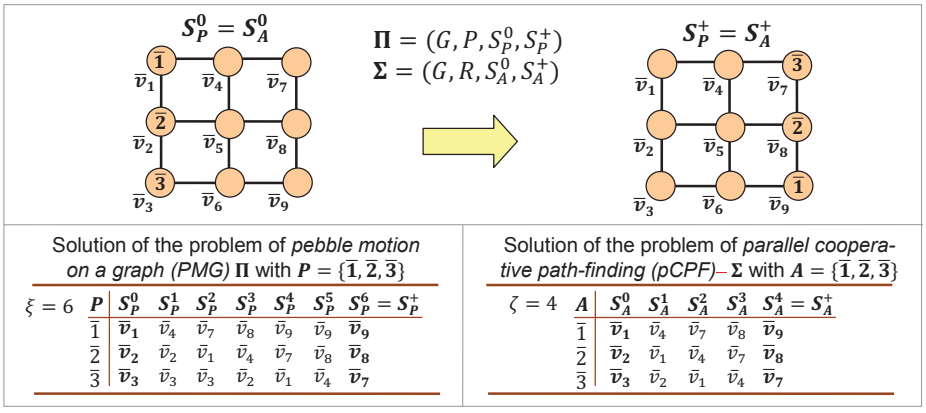
(i) $S_P^k : P \longrightarrow V$ is a uniquely invertible function for every $k = 1,2, \dots, \xi$;

(ii) $S_P^\xi = S_P^+$ (that is, all the pebbles eventually reach their destination vertices);

(iii) either $S_P^k(p) = S_P^{k+1}(p)$ or $\{S_P^k(p), S_P^{k+1}(p)\} \in E$ for every $p \in P$ and $k = 1,2, \dots, \xi - 1$ (that is, a pebble either stays in a vertex or moves along an edge);

(iv) if $S_P^k(p) \neq S_P^{k+1}(p)$ (that is, the pebble $p$ moves between time steps $k$ and $k+1$) then $S_P^k(q) \neq S_P^{k+1}(p)\ \forall q \in P$ with $q \neq p$ must hold for every $p \in P$ and $k = 1,2, \dots, \xi - 1$ (that is, a pebble can move into a currently unoccupied vertex only).

The instance of PMG is formally a quadruple $\Pi = (G, P, S_P^0, S_P^+)$. A solution to the instance $\Pi$ will be denoted as $\mathcal{S}_P(\Pi) = [S_P^0, S_P^1, \dots, S_P^\xi]$. □




| $P$ | $S_P^0$ | $S_P^1$ | $S_P^2$ | $S_P^3$ | $S_P^4$ | $S_P^5$ | $S_P^6 = S_P^+$ |
|---|---|---|---|---|---|---|---|
| $\bar{1}$ | $\bar{v}_1$ | $\bar{v}_4$ | $\bar{v}_7$ | $\bar{v}_8$ | $\bar{v}_9$ | $\bar{v}_9$ | $\bar{v}_9$ |
| $\bar{2}$ | $\bar{v}_2$ | $\bar{v}_2$ | $\bar{v}_1$ | $\bar{v}_4$ | $\bar{v}_7$ | $\bar{v}_8$ | $\bar{v}_8$ |
| $\bar{3}$ | $\bar{v}_3$ | $\bar{v}_3$ | $\bar{v}_3$ | $\bar{v}_2$ | $\bar{v}_1$ | $\bar{v}_4$ | $\bar{v}_7$ |

| $A$ | $S_A^0$ | $S_A^1$ | $S_A^2$ | $S_A^3$ | $S_A^4 = S_A^+$ |
|---|---|---|---|---|---|
| $\bar{1}$ | $\bar{v}_1$ | $\bar{v}_4$ | $\bar{v}_7$ | $\bar{v}_8$ | $\bar{v}_9$ |
| $\bar{2}$ | $\bar{v}_2$ | $\bar{v}_1$ | $\bar{v}_4$ | $\bar{v}_7$ | $\bar{v}_8$ |
| $\bar{3}$ | $\bar{v}_3$ | $\bar{v}_2$ | $\bar{v}_1$ | $\bar{v}_4$ | $\bar{v}_7$ |

**Figure 2.** *An illustration of the problem of* ***pebble motion on a graph (PMG)*** *and* ***parallel cooperative path-finding (pCPF)****.* Both problems are illustrated on the same graph with the same initial and goal locations. The task is to move pebbles/agents from their initial locations specified by $S_P^0/S_A^0$ to their goal locations specified by $S_P^+/S_A^+$. The solution of makespan 6 ($\xi = 6$) is shown for PMG and the solution of makespan 4 ($\zeta = 4$) is shown for pCPF. Note the differences in parallelism between these two solutions – pCPF allows a higher number of moves to be performed in parallel.

The **notation** in the form of a line above the symbol is used to distinguish a constant from a variable (for example, $p \in P$ is a variable while $\bar{p}_2$ is a constant; sometimes a constant parameterized by a variable or by an expression will be used – for example, $\bar{p}_i$ denotes a constant parameterized by the index $i \in \mathbb{N}$; parameterization by an expression will be clear from the context).

When speaking about a move at time step $k$, the time step of commencing the move is referred to (the move is performed between time steps $k$ and $k+1$).

A parallel version of CPF derives from a **relaxation** of PMG/CPF. The requirement that the target vertex of a pebble/agent must be vacated in the previous time step is relaxed. Thus, the move of an agent entering the target vertex, which is simultaneously vacated by another agent

and no other agent is trying to enter the same target vertex, is allowed in the parallel version of CPF. However, there must be some leading agent initiating such a chain of moves by moving into an unoccupied vertex (that is, agents can move like a train with the leading agent in front), which is not entered by any other agent at the same time step. These requirements rule out the rotation of agents along a cycle with no vacant position as well as the swapping of a pair of agents along an edge. The problem is formalized in the following definition.

**Definition 2** ***(parallel cooperative path-finding – pCPF).*** Again, let $G = (V, E)$ be an undirected graph. A set of *agents* $A = \{\bar{a}_1, \bar{a}_2, \dots, \bar{a}_\nu\}$, where $\nu < |V|$ is given instead of a set of pebbles. Similarly, the graph models the environment where the agents are moving. The *initial arrangement* and *the goal arrangement* of agents are defined by two uniquely invertible functions $S_A^0 : A \longrightarrow V$ (that is, $S_A^0(a) \neq S_A^0(b)$ for every $a, b \in A$ with $a \neq b$) and $S_A^+ : A \longrightarrow V$, respectively. The problem of *parallel cooperative path-finding* (*pCPF*) then consists in the need to solve the task of finding a number $\zeta$ and a sequence of agent arrangements $\mathcal{S}_A = [S_A^0, S_A^1, \dots, S_A^\zeta]$ for which the following conditions hold:

(i) $S_A^k : A \longrightarrow V$ is a valid arrangement for every $k = 1,2, \dots, \zeta$ (that is, uniquely invertible);

(ii) $S_A^\zeta = S_A^+$ (that is, all the agents eventually reach their destinations);

(iii) either $S_A^k(a) = S_A^{k+1}(a)$ or $\{S_A^k(a), S_A^{k+1}(a)\} \in E$ for every $a \in A$ and $k = 1,2, \dots, \zeta - 1$ (that is, an agent either stays in a vertex or moves into the neighboring vertex);

(iv) if $S_A^k(a) \neq S_A^{k+1}(a)$ (that is, the agent $a$ moves between time steps $k$ and $k + 1$) then there must exist a sequence of distinct agents $[a = b_0, b_1, \dots, b_\lambda]$ with $\lambda \in \mathbb{N}_0$ such that $S_A^k(c) \neq S_A^{k+1}(b_\lambda)$ $\forall c \in A$ with $c \neq b_\lambda$ ($b_\lambda$ moves to a vertex that is unoccupied at time step $k$; $b_\lambda$ is the *leading* agent in the chain of agents which includes the sequence as its part) and $S_A^{k+1}(b_i) = S_A^k(b_{i+1})$ for $i = 0,1, \dots, \lambda - 1$ (agents $a = b_0, b_1, \dots, b_{\lambda-1}$ follow the leader like a chain; they move all at once between time steps $k$ and $k + 1$).

The instance of pCPF is formally a quadruple $\Sigma = (G, A, S_A^0, S_A^+)$. The solution to the instance $\Sigma$ will be denoted as $\mathcal{S}_A(\Sigma) = [S_A^0, S_A^1, \dots, S_A^\zeta]$. □

The only conceptual difference between the definition of PMG/CPF and that of pCPF consists in point (iv). The remaining differences are attributable to different names of functions representing arrangements of agents.

The numbers $\xi$ and $\zeta$ are called the ***makespan*** of the solution of PMG/CPF and pCPF, respectively. The makespan needs to be distinguished from the ***size*** of the solution, which is the total number of moves performed by pebbles/agents. The makespan is typically smaller than the size of the solution. In case of the PMG/CPF with just a single unoccupied vertex, the makespan and the size of the solution are the same.

Examples of instances of PMG/CPF and pCPF and their solutions are shown in Figure 2.

## 2.2. *Known Properties of Motion Problems and Related Questions*

Note that a solution of an instance of PMG/CPF as well as a solution of an instance of pCPF allows a pebble/agent to **stay** in a vertex for more than a single time step. It is also possible that a pebble/agent **visits** the same vertex **several times** within the solution. Hence, a sequence of moves for a single pebble/agent does not necessarily form a simple path in the given input graph (if the trajectory of the agent is to be modeled by a simple path, a *time expanded graph* with a

copy of the input graph for every time step may need to be considered; a path in the time-expanded graph is always simple as it connects vertices in consecutive time steps only).

Note further that both these problems intrinsically allow parallel movements of pebbles/agents. That is, more than one pebble/agent can perform a move at a single time step. However, pCPF allows higher parallelism due to its weaker requirements on movements (the target vertex is required to be unoccupied only for the leading agent in the current time step – see Figure 2). More than one unoccupied vertex is necessary to obtain parallelism in PMG/CPF. On the other hand, it is sufficient to have a single unoccupied vertex to obtain parallelism in pCPF (consider, for example, agents moving along a cycle with one vacant position).

There is an easy way to prove a correspondence between the PMG/CPF and the pCPF solution, as summarized in the following proposition. It states that the solution of an instance of PMG/CPF can be used as a solution to the corresponding instance of pCPF, which has the same graph, the same set of agents, and the same initial and goal arrangements.

**Proposition 1** ***(problem correspondence).*** Let $\Pi = (G, P, S_P^0, S_P^+)$ be an instance of PMG/CPF and let $\mathcal{S}_P(\Pi) = [S_P^0, S_P^1, \dots, S_P^\xi]$ be its solution. Then $\mathcal{S}_A(\Sigma) = \mathcal{S}_P(\Pi)$ is a solution of the instance pCPF $\Sigma = (G, P, S_P^0, S_P^+)$ (that is, the instance of pCPF on the same graph has the set of agents represented by the set of pebbles and the initial/goal locations of agents are the same as those of the pebbles). ■

There is a variety of modifications of PMG/CPF and pCPF. A natural additional requirement is to produce solutions with the **shortest possible makespan** (that is, the numbers $\xi$ or $\zeta$, respectively, are required to be as small as possible). Unfortunately, this requirement makes the problem of PMG/CPF **intractable**. It was shown in [15, 16] that producing an optimal makespan solution to the special case of PMG/CPF known as $N \times N$*-puzzle* (or $(N^2 - 1)$*-puzzle*), which takes place on a 4-connected grid of the $N \times N$ size with one vacant position, is ***NP*-hard**. Hence, PMG/CPF on a general graph with an arbitrary set of pebbles/agents is $NP$-hard as well.

However, no similar conclusion regarding the complexity of the optimal pCPF can simply be drawn based on the above facts. The situation here is complicated due to the inherent parallelism, which may affect the makespan in an unforeseen way. The proof constructions used for the $N \times N$-puzzle in [15, 16] thus no longer apply for pCPF.

Observe further that difficult cases of the problem of PMG/CPF have a single unoccupied vertex. This fact may raise a question of how the situation changes when there are **more** than one **unoccupied vertices**. Intuition suggests that more unoccupied vertices may simplify the problem. Unfortunately, this is not the case. PMG/CPF on a general graph with the fixed number of unoccupied vertices is still $NP$-hard since multiple copies of the $N \times N$-puzzle from [15, 16] can be used to add as many unoccupied vertices as needed (note that the resulting graph may be disconnected).

The situation is much easier **without** the requirement of the **optimality** of the solution makespan; PMG/CPF is in the ***P* class**, as shown in [13, 37]. Due to Proposition 1, pCPF is also in the P class. It has also been shown in [13] that a solution of the size of $\mathcal{O}(|V|^3)$ can be generated for any solvable PMG/CPF instance $\Pi = (G = (V, E), P, S_P^0, S_P^+)$. Hence, it provides a polynomial upper bound on the size of the content of the oracle to guess the solution in a nondeterministic model. Thus, it is possible to conclude that the decision version of optimal

PMG/CPF is an ***NP*-complete** problem. The decision version here means the yes/no question as to whether there is a solution of Π of the makespan smaller than the given bound.

It seems that PMG/CPF and pCPF problems have already been resolved except for the case of the complexity of the optimal pCPF. However, there is another issue worth studying. Constructions proving the membership of the problem of PMG/CPF into the *P* class used in [13, 37] generate solutions that are too long for practical use. As the makespan of the solution is of great importance in practice, this property makes these methods unsuitable when dealing with a real-life motion problem abstracted as PMG/CPF or pCPF [23, 24, 25]. Hence, alternative solution methods that generate shorter, yet sub-optimal, solutions are also of interest [22, 23, 24, 25, 28].

## 3. The Intractability of Optimal Cooperative Path-Finding

The main result presented in this section is that the optimal pCPF is intractable. In particular, it is shown to be *NP*-hard and the corresponding decision version to be *NP*-complete. The proof technique was partially inspired by Even's et al. [6] proof of *NP*-completeness of the *two-commodity integral flow* problem [1]. Similarly as in [6], we reduce *propositional satisfiability* (SAT) [1, 8, 12] to optimal pCPF to show its *NP*-hardness. The reduction is quite complex and requires a thorough technical preparation, as elaborated in the following sections. On the other hand, proving the membership of optimal pCPF to *NP* is relatively easy.

### 3.1. *Overview of the Reduction of SAT to Optimal pCPF*

As the reduction of SAT to optimal pCPF is technically complicated, a brief overview is provided at this point to improve the readability of the technical description.

The top-level idea of the reduction is that the movement of agents will simulate valuation of the given propositional formula. Thus, we need to construct an instance of pCPF for the given propositional formula so that there are two options of going through a certain location in the graph that correspond to every propositional variable. If agents go through one of these two options, the corresponding propositional variable should be valuated accordingly as either $TRUE$ or $FALSE$. The question is how to construct the pCPF instance where movements of agents in any optimal solution simulate a valuation of the formula. Two fundamental properties are implicitly present when the propositional formula is valuated. Both of them need to be simulated explicitly when the formula is reduced to an instance of pCPF.

The first property is the so-called *propositional consistency*, which means that all the positive and all the negative occurrences of the same variable in the input formula have the same propositional value, respectively. The second property is the fact that all the *clauses* of the propositional formula in *CNF* [12] need to be satisfied in order to satisfy the entire formula. This characteristic will be called *clause satisfaction*.

The following section is devoted to techniques for controlling movements of agents over the graph in optimal solutions of pCPF. The so-called *vertex locking* mechanism is developed to force agents to move or prevent them from moving into some vertices of the graph. Movements of agents need to be controlled to allow a simulation of propositional consistency and clause satisfaction in pCPF eventually, which is elaborated in subsequent sections. The mechanism of the so-called *conjugation* is developed to keep a group of moving agents together in order to simulate propositional consistency properly – the group must not be divided between positive and negative optional pass ways, which simulates that all the occurrences of a given variable are

assigned the same truth-value. All the movement-control techniques are finally put together to simulate the identification of a satisfying valuation of the given propositional formula by finding an optimal makespan solution to the constructed instance of pCPF.

If not interested in the technical details of vertex locking, please skip to Section 3.3 (17).

## 3.2. *Vertex Locking Techniques for Controlling Agent Movements*

A technique to **prevent** agents **from entering** a given vertex at a given set of time steps will be shown in this section. This is a crucial skill used later to force agents to move in a required way in optimal solutions in order to simulate proper compliance with the propositional formula. For the sake of understanding the suggested concepts, the explanation is followed by a scheme where a simple vertex locking technique is gradually augmented to eventually obtain a technique that will actually be used in the reduction.

The vertex locking technique can be applied on an arbitrary instance of pCPF. The result of the application of the technique on the instance is that agents cannot enter a selected vertex at the selected time steps in any optimal solution (the shortest possible makespan of the solution is required). The augmentation of the problem consists in adding new vertices, edges, and agents into the instance. The selection of time steps at which the vertex will not be allowed entry by the original agents is modeled by an appropriate setting of the initial and goal locations of the newly added agents. The whole construction is formalized in the following lemma and its proof.

**Lemma 1** ***(vertex locking augmentation).*** Assume the following preconditions:

(a) Let $\Sigma = (G = (V, E), A, S_A^0, S_A^+)$ be an instance of pCPF and let $v \in V$ with $S_A^0(a) \neq v$ $\forall a \in A$ be the so-called *locked vertex*.

(b) Next, let $T = \{t_1, t_2, \dots, t_n\}$, where $t_i \in \mathbb{N}_0$ (*natural numbers* including 0) for $i = 1,2, \dots, n$ and $t_1 < t_2 < \cdots < t_n$, be a set of so called *lock time steps*.

Then there exists an instance of pCPF $\Sigma' = (G' = (V', E'), A', S_{A'}^0, S_{A'}^+)$ such that $\Sigma'|_V = \Sigma$ and it **never happens** that an agent $a \in A$ enters the vertex $v$ at any time step $t \in T$ in any **optimal** solution $\mathcal{S}_{A'}^*(\Sigma')$ (entering the vertex $v$ at time step $t$ means that an agent is located in $v$ at time step $t$). ■

The notation $\Sigma'|_V$ stands for a *restriction* of the pCPF problem on the set of vertices $V$. That is, if $\Sigma' = (G' = (V', E'), A', S_{A'}^0, S_{A'}^+)$ and $V \subseteq V'$, then $\Sigma'|_V = \left(G'|_V, A'|_V, S_{A'|_V}^0, S_{A'|_V}^+\right)$, where $G'|_V = (V, E' \cap \{\{u, v\} | u, v \in V\})$, $A'|_V = \{a \in A' | S_{A'}^0(a) \in V \wedge S_{A'}^+(a) \in V\}$, $S_{A'}^0|_V : A'|_V \longrightarrow V$ with $S_{A'}^0|_V(a) = S_{A'}^0(a)$ $\forall a \in A'|_V$, and $S_{A'}^+|_V : A'|_V \longrightarrow V$ with $S_{A'}^+|_V(a) = S_{A'}^+(a)$ $\forall a \in A'|_V$. In other words, each component of the description of the instance is naturally restricted on a smaller set of vertices. The lemma states that the augmented instance $\Sigma'$ after restriction on the original set of vertices is the same as the original instance $\Sigma$.

**Proof.** Let $\zeta^*$ be the makespan of any optimal solution of the pCPF instance $\Sigma$ (note that the number $\zeta^*$ is difficult to compute, as shown later; but assume that it is currently known).

An augmentation of the graph $G = (V, E)$ will be shown first. The set of vertices $V$ is extended with a set of new vertices $V_X = \{\bar{u}_{t_n}, \bar{u}_{t_n-1}, \dots, \bar{u}_1, \bar{w}_1, \bar{w}_2, \dots, \bar{w}_\lambda\}$, where $\lambda = \zeta^* + t_n - t_1$. The new vertices are connected around the locked vertex $v$ in the following way. A set of

edges $E_X = \{\{\bar{u}_{t_n}, \bar{u}_{t_n-1}\}, \{\bar{u}_{t_n-1}, \bar{u}_{t_n-2}\}, , \dots, \{\bar{u}_2, \bar{u}_1\}, \{\bar{u}_1, v\}, \{v, \bar{w}_1\}, \{\bar{w}_1, \bar{w}_2\}, \{\bar{w}_2, \bar{w}_3\}, \dots, \{\bar{w}_{\lambda-1}, \bar{w}_\lambda\}\}$ is added to the graph with the extended set of vertices. Thus, the augmented graph is $G' = (V' = V \cup V_X, E' = E \cup E_X)$.

The idea behind the construction of the augmented graph is that new agents are initially placed in new vertices $\bar{u}_t$ for $t \in T$ with $t \geq 1$ or a new agent is placed in $v$ if $0 \in T$. Then the newly added agents are forced to move straight ahead into the vertices $\bar{w}_1, \bar{w}_2, \dots, \bar{w}_\lambda$ through the vertex $v$. Agents are forced to move in this way as a result of the requirement on the optimality of the solution. That is if agents do not move in the above-suggested way, they cannot manage to reach their destinations in time. The motion of new agents through the vertex $v$ makes an obstruction in this vertex exactly at the selected time steps given by lock time steps $T$.

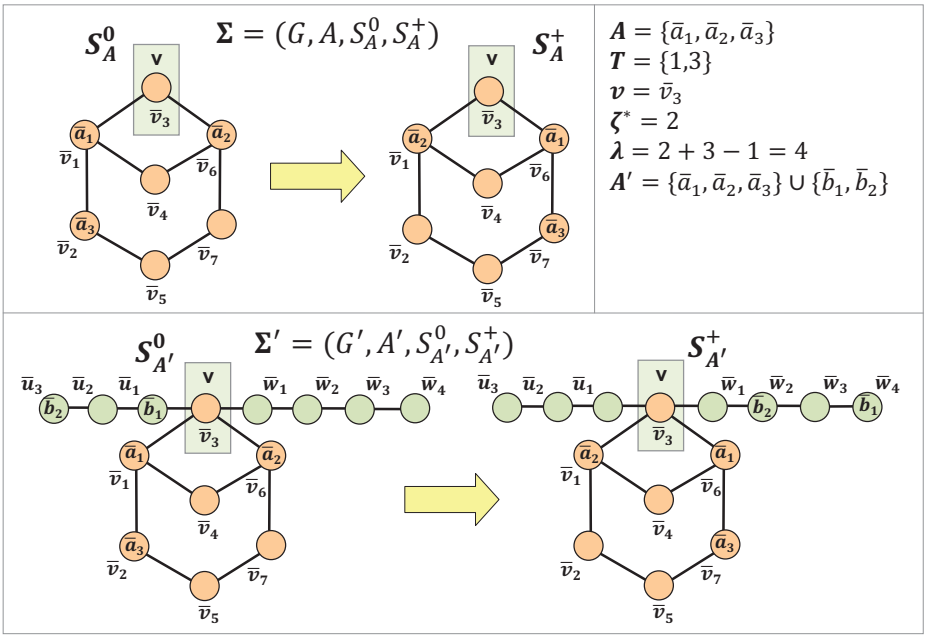



**Figure 3**. *An illustration of* ***vertex locking augmentation*** *in an instance of the pCPF problem.* Assume that we want to prevent the agents $\bar{a}_1$, $\bar{a}_2$, and $\bar{a}_3$ from entering the vertex $\bar{v}_3$ at time steps 1 and 3 in any optimal solution. The original instance $\Sigma$ with the set of agents $A = \{\bar{a}_1, \bar{a}_2, \bar{a}_3\}$ is shown in the upper part of the figure. The makespan of any optimal solution of $\Sigma$ is $\zeta^* = 2$. The augmented instance $\Sigma'$ is in the lower part of the figure. New vertices $\bar{u}_3$, $\bar{u}_2$, $\bar{u}_1$, $\bar{w}_1$, $\bar{w}_2$, $\bar{w}_3$, and $\bar{w}_4$ and new agents $\bar{b}_1$ and $\bar{b}_2$ were added. The makespan of any optimal solution of the augmented problem is $\lambda + t_1 = \zeta^* + t_n - t_1 + t_1 = \zeta^* + t_n = 2 + 3 = 5$ and it never happens that any of the original agents $\bar{a}_1$, $\bar{a}_2$, and $\bar{a}_3$ enters $\bar{v}_3$ at time step 1 or 3 in any optimal solution ($\bar{v}_3$ is occupied by $\bar{b}_1$ at time step 1 and by $\bar{b}_2$ at time step 3).

A formal description of the above idea follows. The set of agents is extended with a set of new agents $A_X = \{\bar{b}_1, \bar{b}_2, \dots, \bar{b}_n\}$; that is, $A' = A \cup A_X$. The initial and goal arrangements of new agents are spread around the locked vertex $v$ in the newly added vertices: $S_{A'}^0(\bar{b}_i) = \bar{u}_{t_i}$ if $t_i \neq 0$ and $S_{A'}^0(\bar{b}_i) = v$ if $t_i = 0$ for $i = 1,2,\dots,n$; $S_{A'}^+(\bar{b}_i) = \bar{w}_{\lambda+t_1-t_i}$ if $\lambda + t_1 - t_i \geq 1$ and $S_{A'}^+(\bar{b}_i) = v$ if $\lambda + t_1 - t_i = 0$ for $i = 1,2,\dots,n$. For the original agents, the initial and the goal arrangements remain the same; that is, $S_{A'}^0(a) = S_A^0(a)$ and $S_{A'}^+(a) = S_A^+(a)\ \forall a \in A$.

At this point, it is necessary to show that it never happens that an agent $a \in A$ enters the vertex $v$ at any time step $t \in T$ within the optimal solution $\mathcal{S}_{A'}^*(\Sigma')$. Any optimal solution of the

pCPF instance $\Sigma'$ has the makespan of $\lambda + t_1$. Moreover, any solution $\mathcal{S}^*_{A'}(\Sigma') = [S^0_{A'}, S^1_{A'}, \dots, S^{\lambda+t_1}_{A'}]$ of the optimal makespan of the instance $\Sigma'$ must satisfy that $S^0_{A'}(\bar{b}_i) = \bar{u}_{t_i},\ S^1_{A'}(\bar{b}_i) = \bar{u}_{t_i-1}, S^2_{A'}(\bar{b}_i) = \bar{u}_{t_i-2},\ \dots,\ S^{t_i-1}_{A'}(\bar{b}_i) = \bar{u}_1, S^{t_i}_{A'}(\bar{b}_i) = v,\ S^{t_i+1}_{A'}(\bar{b}_i) = \bar{w}_1,\ S^{t_i+2}_{A'}(\bar{b}_i) = \bar{w}_2, \dots, S^{\lambda+t_1}_{A'}(\bar{b}_i) = \bar{w}_{\lambda+t_1-t_i} = S^+_{A'}(\bar{b}_i)$ for $i = 1,2,\dots,n$. This is ensured by the fact that the shortest path from $S^0_{A'}(\bar{b}_i)$ to $S^+_{A'}(\bar{b}_i)$ in $G'$ has the length of $\lambda + t_1$ and it consists of vertices $[\bar{u}_{t_i}, \bar{u}_{t_i-1}, \bar{u}_{t_i-2}, \dots, \bar{u}_1, v, \bar{w}_1, \bar{w}_2, \dots, \bar{w}_{\lambda+t_1-t_i}]$. Hence, **no shorter** solution in terms of the makespan exists.

However, it remains to show that the original agents from $A$ manage to reach their destinations within the makespan of $\lambda + t_1$. This claim follows from the equality $\lambda = \zeta^* + t_n - t_1$, i.e. that for at least $\zeta^*$ time steps the vertex $v$ is not obstructed by any motion of newly added agents supposing they are moving straight towards their destinations. In any optimal solution of the original instance, it is sufficient to enter $v$ at most $\zeta^*$ times (note that none of the original agents need to occupy $v$ at the beginning). Thus, any optimal solution of the original instance can be simulated in the augmented instance while the moves of the original agents are stopped at the time steps during which $v$ is obstructed. Hence, the makespan of any optimal solution of $\Sigma'$ is exactly $\lambda + t_1$.

It has been shown that the vertex $v$ is obstructed at every time step $t \in T$ in any optimal solution. Hence, no original agent can enter $v$ at any time step $t \in T$. ■

The situation from Lemma 1 is illustrated in Figure 3. Note that it is not difficult to extend the construction from the proof of Lemma 1 on **multiple vertices** that will be **locked** at selected time steps (different sets of time steps for locking can be used for different vertices). Another useful property of the augmented problem is summarized in the following corollary.

**Corollary 1** ***(makespan preserving vertex locking).*** Assume preconditions (a) and (b) together with the following preconditions:

(c) There exists a solution $\mathcal{S}_A(\Sigma)$ of the instance $\Sigma = (G = (V, E), A, S^0_A, S^+_A)$ of the makespan $\zeta$, where $t_n \leq \zeta$.

(d) Let $v \in V$ be a locked vertex entered by an agent within $\mathcal{S}_A(\Sigma)$ at time steps $Y = \{y_1, y_2, \dots, y_m\}$ where $y_i \in \mathbb{N}_0$ for $i = 1,2,\dots,n$ and $y_1 < y_2 < \dots < y_m$ and it holds that $Y \cap T = \emptyset$.

Then there exists an instance $\Sigma' = (G' = (V', E'), A', S^0_{A'}, S^+_{A'})$ such that $\Sigma'|_V = \Sigma$ and it **never happens** that an agent $a \in A$ enters the vertex $v$ at any time step $t \in T$ within any **optimal** solution $\mathcal{S}^*_{A'}(\Sigma')$; moreover the makespan of any optimal solution of $\Sigma'$ is again $\zeta$. ■

**Proof.** The construction of $\Sigma'$ is almost the same as in the proof of Lemma 1, only the parameter $\lambda$ is now set to $\zeta - t_1$. Then, the construction ensures the makespan of $\zeta$ of any optimal solution of $\Sigma'$.

The makespan is **at least** $\zeta$ since the newly added agents must go along the newly added path towards its end, which cannot be carried out in a smaller makespan. On the other hand, there exists a solution of the makespan $\zeta$ of the augmented instance $\Sigma'$. The vertex $v$ needs to be occupied only at time steps $t_1, t_2, \dots, t_n$ by the newly added agents that do not interfere with the time steps at which the vertex $v$ is entered within the solution $\mathcal{S}_A(\Sigma)$ by the original agents (this

is due to $Y \cap T = \emptyset$). Altogether, the makespan of any optimal solution of an augmented instance $\Sigma'$ is $\zeta$. ■

Lemma 1 as well as Corollary 1 can be generalized for locking a given number of vertices of a selected subset of vertices $W \subseteq V$ at a selected set of time steps $T$. Nevertheless, only a special variant of this generalization, where just one vertex of $W$ is to be locked at selected time steps, will actually be used in further reasoning. To be more precise, at least one vertex in $W$ is required not to be occupied by an agent from the original set of agents at any time step $t \in T$. It can be regarded as a kind of **disjunctive** locking where the set considered in disjunction is $W$. An analogous extension to Corollary 1 that preserves makespan additionally assumes the existence of a solution of the original instance where at least one vertex of $W$ is unoccupied at any time step $t \in T$. These statements, which are merely a technical extension of Lemma 1 and Corollary 1, are formalized as Lemma 2 and Corollary 2.

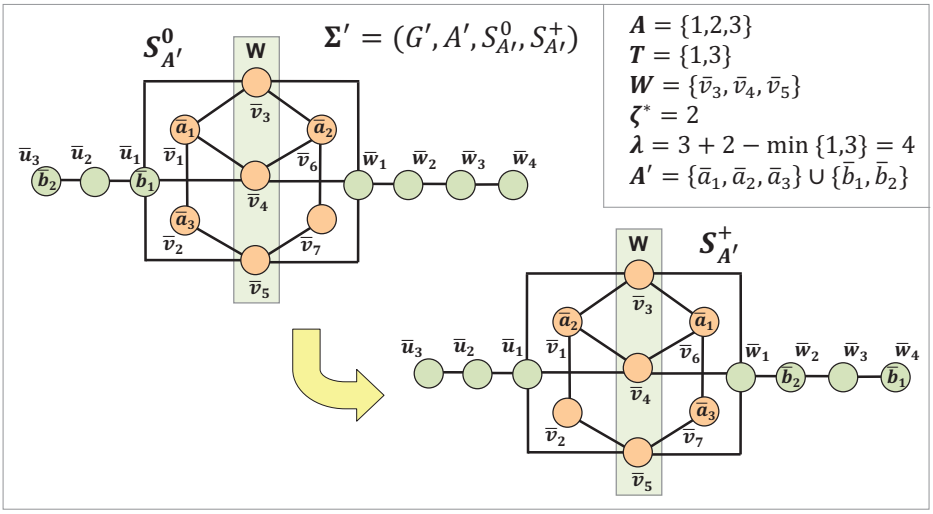



**Figure 4.** *An illustration of a vertex* ***set locking augmentation*** *in an instance of the pCPF problem.* Assume that we want at least one vertex of the set $W = \{\bar{v}_3, \bar{v}_4, \bar{v}_5\}$ not to be occupied by any of the original agents $\bar{a}_1$, $\bar{a}_2$, and $\bar{a}_3$ at time steps 1 and 3 in any optimal solution. The original instance $\Sigma$ with the set of agents $A = \{\bar{a}_1, \bar{a}_2, \bar{a}_3\}$ is taken from Figure 3. The augmentation is made by adding a new path consisting of vertices $\bar{u}_3, \bar{u}_2, \bar{u}_1, \bar{w}_1, \bar{w}_2, \bar{w}_3$, and $\bar{w}_4$ around the set $W$ and by adding new agents $\bar{b}_1$ and $\bar{b}_2$. The makespan of any optimal solution of the augmented instance $\Sigma'$ is $\lambda + 1 = t_n + \zeta^* = 3 + 2 = 5$. At least one vertex of $W$ is occupied by $\bar{b}_1$ at time step 1 and by $\bar{b}_2$ at time step 3 in any optimal solution. Hence, it never happens that all the vertices of $W$ are occupied by agents $\bar{a}_1, \bar{a}_2$, and $\bar{a}_3$ at time step 1 or 3 within the optimal solution.

**Lemma 2** ***(set locking augmentation).*** Let the following preconditions hold:

(aa) $\Sigma = (G = (V, E), A, S^0_A, S^+_A)$ be an instance of pCPF and $W \subseteq V$ with $S^0_A(a) \notin W$ $\forall a \in A$ be the so-called *set of locked vertices*.

(bb) Next, let $T = \{t_1, t_2, \dots, t_n\}$, where $t_i \in \mathbb{N}_0$ (*natural numbers* including 0) for $i = 1,2, \dots, n$, and $t_1 < t_2 < \cdots < t_n$ be a set of *lock time steps*.

Then there exists an instance of the problem of pCPF $\Sigma' = (G' = (V', E'), A', S^0_{A'}, S^+_{A'})$ such that $\Sigma'|_V = \Sigma$ and it **never happens** that all the vertices of the set $W$ are occupied by agents

from the set $A$ at any time step $t \in T$ within any **optimal** solution $\mathcal{S}_{A'}(\Sigma')$ (that is, at least one vertex from $W$ is not occupied by an agent from $A$ at any time step $t \in T$). ■

**Proof.** The instance Σ is augmented in a way that a new agent is forced to visit exactly one vertex of the set $W$ at each time step $t \in T$. The technique is almost the same as in the case of Lemma 1. A path of new vertices is added around the set of locked vertices. The path branches into all the vertices of $W$ at both connection points. Formally, the augmentation is as follows.

Let $\zeta^*$ be the makespan of any optimal solution of Σ. The set of vertices $V$ is extended with a set of new vertices $V_X = \{\bar{u}_{t_n}, \bar{u}_{t_n-1}, \dots, \bar{u}_1, \bar{w}_1, \bar{w}_2, \dots, \bar{w}_\lambda\}$, where $\lambda = \zeta^* + t_n - t_1$. A set of edges $E_X = \{\{\bar{u}_{t_n}, \bar{u}_{t_n-1}\}, \{\bar{u}_{t_n-1}, \bar{u}_{t_n-2}\}, \dots, \{\bar{u}_2, \bar{u}_1\}, \{\bar{w}_1, \bar{w}_2\}, \{\bar{w}_2, \bar{w}_3\}, \dots, \{\bar{w}_{\lambda-1}, \bar{w}_\lambda\}\} \cup \{\{\bar{u}_1, w\} \mid w \in W\} \cup \{\{w, \bar{w}_1\} \mid w \in W\}$ is added to the graph with the extended set of vertices. Thus, the augmented graph is $G' = (V' = V \cup V_X, E' = E \cup E_X)$.

The set of agents is extended with a set of new agents $A_X = \{\bar{b}_1, \bar{b}_2, \dots, \bar{b}_n\}$; that is, $A' = A \cup A_X$. The initial and goal arrangements of new agents are spread around a set of locked vertices in the newly added vertices as follows: $S^0_{A'}(\bar{b}_i) = \bar{u}_{t_i}$ if $t_i \neq 0$ and $S^0_{A'}(\bar{b}_i) = w$ for some $w \in W$ if $t_i = 0$ for $i = 1,2,\dots,n$; $S^+_{A'}(\bar{b}_i) = \bar{w}_{\lambda+t_1-t_i}$ if $\lambda + t_1 - t_i \geq 1$ and $S^+_{A'}(\bar{b}_i) = w$ for some $w \in W$ if $\lambda + t_1 - t_i = 0$ for $i = 1,2,\dots,n$. For the original agents, the initial and the goal arrangements remain the same; that is, $S^0_{A'}(a) = S^0_A(a)$ and $S^+_{A'}(a) = S^+_A(a)\ \forall a \in A$.

The makespan of any optimal solution of Σ′ is **at least** $\lambda + t_1$ since the shortest path from $S^0_{A'}(\bar{b}_i)$ to $S^+_{A'}(\bar{b}_i)$ in $G'$ has the length of $\lambda + t_1$ for any $i = 1,2,\dots,n$. On the other hand, since $\lambda = \zeta^* + t_n - t_1$, no vertex of $W$ is occupied by any new agent at least for $\zeta^*$ time steps supposing the new agents are moving straight towards their destinations. Together with the fact that in any optimal solution of the original instance Σ, it is sufficient to occupy $W$ for at most $\zeta^*$ time steps, the makespan of any optimal solution of Σ′ is exactly $\lambda + t_1$. ■

The construction of the augmentation from the proof of the above lemma is shown in Figure 4. Observe that the construction can easily be extended to include the locking of **multiple sets** of locked vertices while different lock time steps may be used for each locked set.

**Corollary 2** ***(makespan preserving set locking)*****.** Assume that the preconditions (aa) and (bb) hold; in addition, assume that the following preconditions hold as well:

(cc) There exists a solution $\mathcal{S}_A(\Sigma)$ of the instance $\Sigma = (G = (V, E), A, S^0_A, S^+_A)$ of the makespan $\zeta$, where $t_n \leq \zeta$.

(dd) There is at least one unoccupied vertex in the selected set $W \subseteq V$ at all the time steps within $\mathcal{S}_A(\Sigma)$ except time steps $Y = \{y_1, y_2, \dots, y_m\}$ with $y_i \in \mathbb{N}_0$ for $i = 1,2,\dots,n$ and $y_1 < y_2 < \dots < y_m$ and it holds that $Y \cap T = \emptyset$.

Then there exists an instance $\Sigma' = (G' = (V', E'), A', S^0_{A'}, S^+_{A'})$ such that $\Sigma'|_V = \Sigma$ and it **never happens** that all the vertices of $W$ are occupied by the original agents from the set $A$ at any time step $t \in T$ within any **optimal** solution $\mathcal{S}^*_{A'}(\Sigma')$; moreover the makespan of any optimal solution of Σ′ is again $\zeta$. ■

**Proof.** The construction of Σ′ is almost the same as in the proof of Corollary 1. The difference is that the parameter $\lambda$ is now set to $\zeta - t_1$. The construction then ensures that the makespan of any optimal solution of Σ′ is $\zeta$.

The makespan of any optimal solution is **at least** $\zeta$ since the newly added agents must go to the end of the newly added path. On the other hand, all the vertices of the set $W$ need to be occupied by the original agents within the solution $\mathcal{S}_A(\Sigma)$ only at time steps $y_1, y_2, \ldots, y_m$ that do not interfere with time steps $t_1, t_2, \ldots, t_n$ (since $Y \cap T = \emptyset$) at which the newly added agents need to occupy at least one vertex of $W$ (supposing they are going directly to their destinations along the newly added path). Hence, there exists a solution of the makespan $\zeta$ of the augmented instance $\Sigma'$. Altogether, any optimal solution of $\Sigma'$ has the makespan $\zeta$. ■

Observe that the original agents are allowed to enter newly added vertices in all the above augmentations. This may help the original agents to reach their destinations **faster** (the newly added vertices may be used as an additional "parking place" for agents). This behavior of agents is undesirable in the planned reduction where it is needed to isolate the vertex locking mechanism from the original instance. Hence, a slight adaptation of the vertex locking technique must be used.

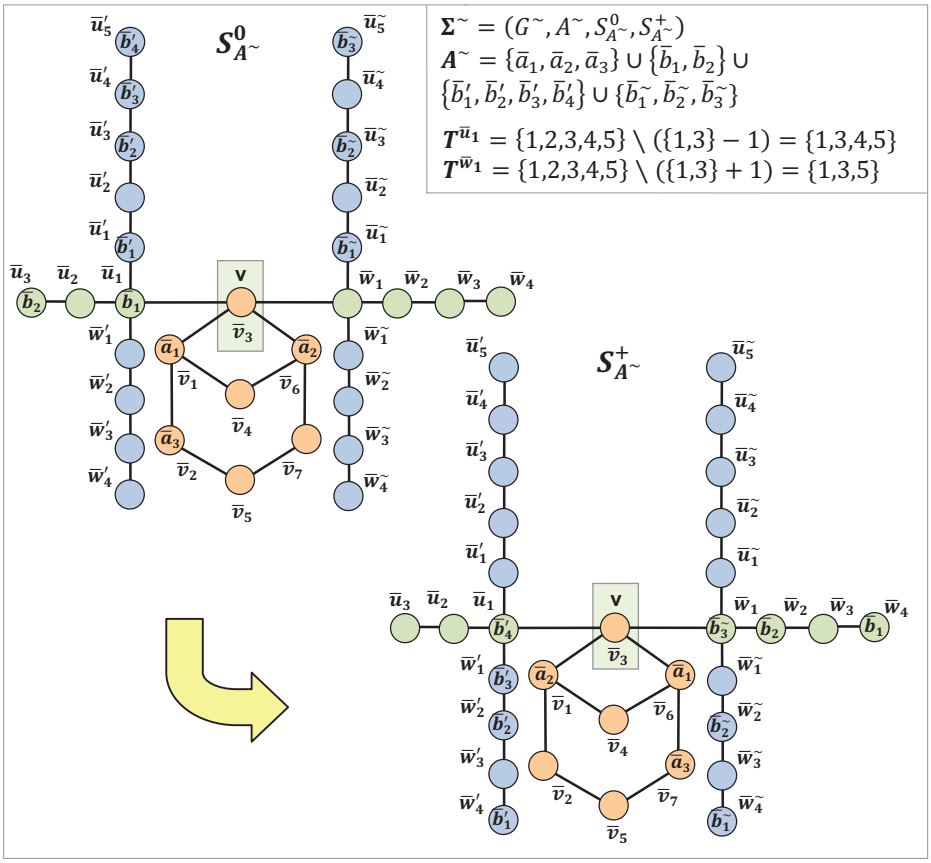



**Figure 5.** *An illustration of* ***two-stage vertex locking*** *in an instance of the pCPF problem.* Assume that we want to prevent agents $\bar{a}_1$, $\bar{a}_2$, and $\bar{a}_3$ from entering the vertex $\bar{v}_3$ at time steps 1 and 3. Additionally, no vertex added by the augmentation can be entered by the original agents $\bar{a}_1$, $\bar{a}_2$, and $\bar{a}_3$. These requirements are ensured by the two-stage locking mechanism. First, $\bar{v}_3$ is locked at time steps 1 and 3 using a path of new vertices $\bar{u}_3$, $\bar{u}_2$, $\bar{u}_1$, $\bar{w}_1$, $\bar{w}_2$, $\bar{w}_3$, and $\bar{w}_4$ (this stage corresponds to Figure 3). Then $\bar{u}_1$ and $\bar{w}_1$ are locked at time steps $T^{\bar{u}_1} = \{1,3,4,5\}$ and $T^{\bar{w}_1} = \{1,3,5\}$, respectively, by the same technique. The makespan of any optimal solution of $\Sigma^\sim$ is 5 (the same as of $\Sigma'$).

Some additional notations are needed to formally express the requirement that the newly added vertices are not to be used by the original agents. Let $\mathcal{S}^*_{A'}(\Sigma') = [S^0_{A'}, S^1_{A'}, \dots, S^\zeta_{A'}]$ be an optimal solution of the pCPF instance $\Sigma'$ over the graph $G' = (V', E')$ and let $V \subseteq V'$. Then the restriction of the solution $\mathcal{S}^*_{A'}(\Sigma')$ on the set of vertices $V$ is denoted as $\mathcal{S}^*_{A'}(\Sigma')|_V = [S^0_{A'}|_V, S^1_{A'}|_V, \dots, S^\zeta_{A'}|_V]$, where $S^i_{A'}|_V : A'|_V \longrightarrow V$ with $S^i_{A'}|_V(a) = S^i_{A'}(a)\ \ \forall a \in A'|_V$ for $i = 0,1,\dots,\zeta$. Next, let $Sol^*(\Sigma') = \{\mathcal{S}^*_{A'}(\Sigma') | \mathcal{S}^*_{A'}(\Sigma')$ be an optimal solution of $\Sigma'\}$, then $Sol^*(\Sigma')|_V = \{\mathcal{S}^*_{A'}(\Sigma')|_V |\ \mathcal{S}^*_{A'}(\Sigma') \in Sol^*(\Sigma')\}$, and let $Sol(\Sigma) = \{\mathcal{S}_A(\Sigma) |\ \mathcal{S}_A(\Sigma)$ be a solution (not necessarily optimal) of $\Sigma\}$. An augmentation $\Sigma'$ of the instance $\Sigma$, where added vertices are never used, can be expressed by the condition $Sol^*(\Sigma')|_V \subseteq Sol(\Sigma)$.

**Proposition 2** ***(two-stage vertex locking).*** Assume that the preconditions (a) and (b) hold. Then there exists an instance of pCPF $\Sigma^\sim = (G^\sim = (V^\sim, E^\sim), A^\sim, S^0_{A^\sim}, S^+_{A^\sim})$ such that $\Sigma^\sim|_V = \Sigma$, where it **never happens** that an agent $a \in A$ enters $v$ at any time step $t \in T$ within any **optimal** solution $\mathcal{S}_{A^\sim}(\Sigma^\sim)$ and $Sol^*(\Sigma^\sim)|_V \subseteq Sol(\Sigma)$ (that is, **original** agents cannot use any added vertex in any optimal solution). ■

Note that Proposition 2 (14) is almost the same as Lemma 1 except for the additionally required condition $Sol^*(\Sigma^\sim)|_V \subseteq Sol(\Sigma)$.

**Proof.** The basic construction will be adopted from the proof of Lemma 1; then some further augmentations will be made by successive applications of Corollary 1 to enforce the condition that $Sol^*(\Sigma^\sim)|_V \subseteq Sol(\Sigma)$.

Let $\zeta^*$ denote the makespan of optimal solutions of $\Sigma$. In the **first stage**, the graph $G$ is extended exactly as in the previous case. That is, a set of vertices $V_X = \{\bar{u}_{t_n}, \bar{u}_{t_n-1}, \dots, \bar{u}_1, \bar{w}_1, \bar{w}_2, \dots, \bar{w}_\lambda\}$, where $\lambda = \zeta^* + t_n - t_1$, and a set of edges $E_X = \{\{\bar{u}_{t_n}, \bar{u}_{t_n-1}\}, \{\bar{u}_{t_n-1}, \bar{u}_{t_n-2}\}, \dots, \{\bar{u}_2, \bar{u}_1\}, \{\bar{u}_1, v\}, \{v, \bar{w}_1\}, \{\bar{w}_1, \bar{w}_2\}, \{\bar{w}_2, \bar{w}_3\}, \dots, \{\bar{w}_{\lambda-1}, \bar{w}_\lambda\}\}$ are added to the graph; that is $G' = (V' = V \cup V_X, E' = E \cup E_X)$. The set of agents is extended with $A_X = \{\bar{b}_1, \bar{b}_2, \dots, \bar{b}_n\}$; that is, $A' = A \cup A_X$ and the initial and goal arrangements of the new agents are set as follows: $S^0_{A'}(\bar{b}_i) = \bar{u}_{t_i}$ if $t_i \neq 0$ and $S^0_{A'}(\bar{b}_i) = v$ if $t_i = 0$ for $i = 1,2,\dots,n$; $S^+_{A'}(\bar{b}_i) = \bar{w}_{\lambda+t_1-t_i}$ if $\lambda + t_1 - t_i \geq 1$ and $S^+_{A'}(\bar{b}_i) = v$ if $\lambda + t_1 - t_i = 0$ for $i = 1,2,\dots,n$. As it has been shown, this construction suffices for satisfying almost all the requirements except $Sol^*(\Sigma')|_V \subseteq Sol(\Sigma)$.

Now, it is necessary to prevent agents from $A$ from entering any of the added vertices $V_X$. Observe that locking vertices $\bar{u}_1$ and $\bar{w}_1$ suffices to fulfill this requirement since the newly added vertices form a path around $v$ and this is the only vertex through which the path is connected to the original graph (neighboring vertices of $v$ are $\bar{u}_1$ and $\bar{w}_1$). Vertices $\bar{u}_1$ and $\bar{w}_1$ need to be locked for all the time steps except for the time steps at which agents from the set $A_X$ go through them in the optimal solution – this represents the **second-stage** locking. More precisely, the vertex $\bar{u}_1$ needs to be locked at time steps from the set $T^{\bar{u}_1} = \{1,2,\dots,\lambda + t_1\} \setminus (\{t_1, t_2, \dots, t_n\} - 1)$ (where $\{t_1, t_2, \dots, t_n\} - 1 = \{t_1 - 1, t_2 - 1, \dots, t_n - 1\}$) and the vertex $\bar{w}_1$ needs to be locked at time steps from the set $T^{\bar{w}_1} = \{1,2,\dots,\lambda + t_1\} \setminus (\{t_1, t_2, \dots, t_n\} + 1)$. Note that $\max(T^{\bar{u}_1}) \leq \lambda + t_1$ as well as $\max(T^{\bar{w}_1}) \leq \lambda + t_1$. Moreover, the construction of sets $T^{\bar{u}_1}$ and $T^{\bar{w}_1}$ ensures that vertices $\bar{u}_1$ and $\bar{w}_1$, respectively, are locked at time steps at which they are not entered within some solution (which is known to be the optimal solution). Hence, Corollary

1 applies for $\Sigma'$, the locked vertex $\overline{u}_1$, and the set of lock time steps $T^{\overline{u}_1}$; that is, the optimal makespan is preserved. In other words, the vertex locking is **synchronized** with the vertex locking from the first stage. Subsequently, Corollary 1 is applied once more for the resulting instance, the locked vertex $\overline{w}_1$, and the set of lock time steps $T^{\overline{w}_1}$. Let $\Sigma^{\sim} = (G^{\sim} = (V^{\sim}, E^{\sim}), A^{\sim}, S^0_{A^{\sim}}, S^+_{A^{\sim}})$ denote the final instance, then $Sol^*(\Sigma^{\sim})|_V \subseteq Sol(\Sigma)$. ■

The construction from Proposition 2 is illustrated in Figure 5. In fact, it represents a further augmentation of the instance in Figure 3.

The important property is that the **size** of all the augmented instances of the problem is $\mathcal{O}(\max(\zeta^*, t_n) + |V| + |E|)$, where $\zeta^*$ is the optimal makespan (that is, asymptotically as many as $\max(\zeta^*, t_n)$ vertices and agents are added). Consequently, if an augmented instance needs to be kept small (with respect to $|V| + |E|$), the numbers $\zeta^*$ and $t_n$ must also be small.

**Corollary 3** ***(makespan preserving two-stage vertex locking).*** Assume that the preconditions (a), (b), (c), and (d) hold. Then there exists an instance $\Sigma^{\sim} = (G^{\sim} = (V^{\sim}, E^{\sim}), A^{\sim}, S^0_{A^{\sim}}, S^+_{A^{\sim}})$ such that $\Sigma^{\sim}|_V = \Sigma$ and it **never happens** that an agent $a \in A$ enters the locked vertex $v$ at any time step $t \in T$ within any **optimal** solution $\mathcal{S}^*_{A^{\sim}}(\Sigma^{\sim})$ and $Sol^*(\Sigma^{\sim})|_V \subseteq Sol(\Sigma)$ (that is, the **original** agents cannot use any added vertex in any optimal solution); moreover, the makespan of any optimal solution of $\Sigma^{\sim}$ is again $\zeta$. ■

**Proof.** The construction from the proof of Proposition 2 can be adopted with a minor change. In the first stage of the construction of $\Sigma^{\sim}$ where the construction from the proof of Lemma 1 has been applied, Corollary 1 is applied instead. This ensures that the intermediate instance after the first stage locking preserves the makespan of $\zeta$. The rest of the proof can be applied without any change. ■

Again, it is not difficult to generalize the construction for locking a subset of a certain size of a selected set of vertices at given time steps where the original agents can move only in the original vertices. These merely technical extensions of Proposition 2 and Corollary 3 are listed as Proposition 3 and Corollary 4.

**Proposition 3** ***(two-stage set locking).*** Assume that the preconditions (aa) and (bb) hold. Then there exists an instance of the problem of pCPF $\Sigma^{\sim} = (G^{\sim} = (V^{\sim}, E^{\sim}), A^{\sim}, S^0_{A^{\sim}}, S^+_{A^{\sim}})$ such that $\Sigma^{\sim}|_V = \Sigma$, where it **never happens** that all the vertices of $W$ are occupied by the original agents from $A$ at any time step $t \in T$ within any **optimal** solution $\mathcal{S}^*_{A^{\sim}}(\Sigma^{\sim})$ and $Sol^*(\Sigma^{\sim})|_V \subseteq Sol^*(\Sigma)$ (that is, **original** agents cannot use any added vertex in any optimal solution). ■

**Proof.** The proof will partially adopt the basic idea of the construction from the proof of Proposition 2. The vertex set locking will be done in two stages by means of successive applications of Corollary 1 to enforce the condition $Sol^*(\Sigma^{\sim})|_V \subseteq Sol^*(\Sigma)$.

Let $\zeta^*$ be the makespan of optimal solutions of the pCPF instance $\Sigma$. The first stage of the augmentation will be performed as in the case of Proposition 2. A set of vertices $V_X = \{\overline{u}_{t_n}, \overline{u}_{t_n-1}, \dots, \overline{u}_1, \overline{w}_1, \overline{w}_2, \dots, \overline{w}_\lambda\}$, where $\lambda = \zeta^* + t_n - t_1$, and a set of edges $E_X = \{\{\overline{u}_{t_n}, \overline{u}_{t_n-1}\}, \{\overline{u}_{t_n-1}, \overline{u}_{t_n-2}\}, \dots, \{\overline{u}_2, \overline{u}_1\}, \{\overline{w}_1, \overline{w}_2\}, \{\overline{w}_2, \overline{w}_3\}, \dots, \{\overline{w}_{\lambda-1}, \overline{w}_\lambda\}\} \cup \{\{\overline{u}_1, w\} \mid w \in W\} \cup \{\{w, \overline{w}_1\} \mid w \in W\}$ are added to the graph; that is $G' = (V' = V \cup V_X, E' = E \cup E_X)$. The

set of agents is extended with a set of new agents $A_X = \{\bar{b}_1, \bar{b}_2, \ldots, \bar{b}_n\}$; that is, $A' = A \cup A_X$ and the initial and goal arrangements of the new agents are set as follows: $S^0_{A'}(\bar{b}_i) = \bar{u}_{t_i}$ if $t_i \neq 0$ and $S^0_{A'}(\bar{b}_i) = w$ for some $w \in W$ if $t_i = 0$ for $i = 1,2,\ldots,n$; $S^+_{A'}(\bar{b}_i) = \bar{w}_{\lambda+t_1-t_i}$ if $\lambda + t_1 - t_i \geq 1$ and $S^+_{A'}(\bar{b}_i) = w$ for some $w \in W$ if $\lambda + t_1 - t_i = 0$ for $i = 1,2,\ldots,n$.

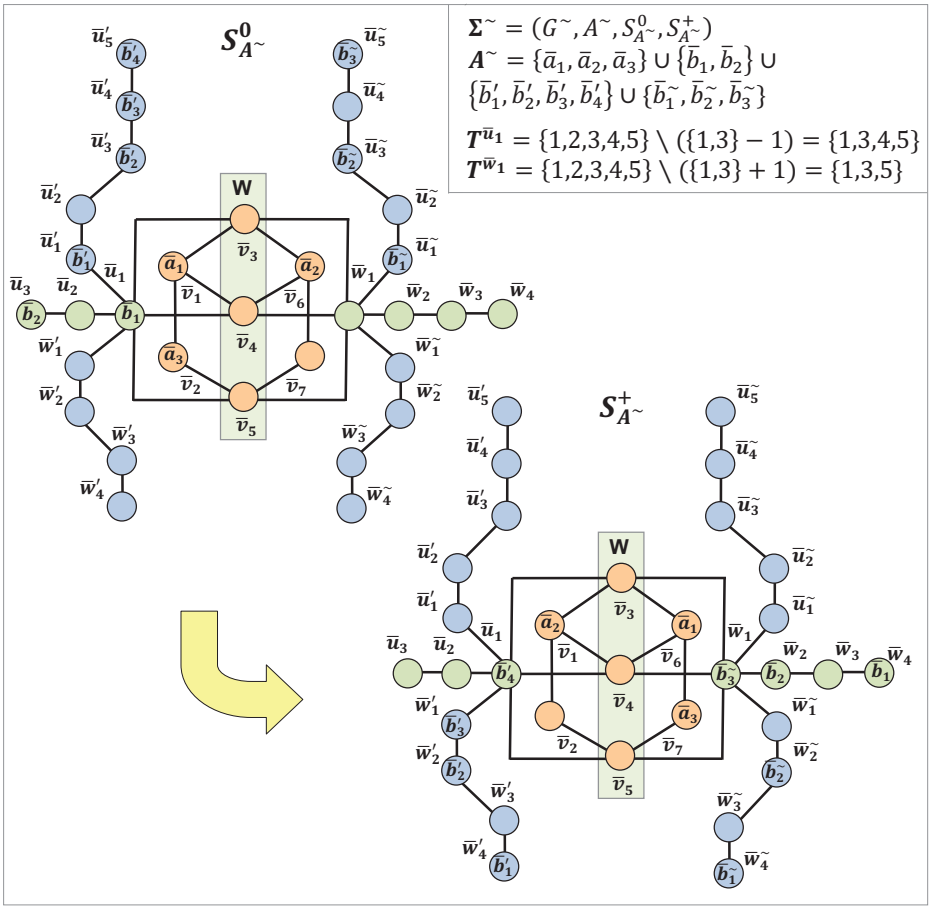



**Figure 6.** *An illustration of two-stage vertex set locking in an instance of the pCPF problem.* At least one vertex of the set $W = \{\bar{v}_3, \bar{v}_4, \bar{v}_5\}$ must not be occupied by any of the original agents $\bar{a}_1$, $\bar{a}_2$, and $\bar{a}_3$ at time steps 1 and 3. Additionally, no vertex added by the augmentation can be entered by any of the original agents. These requirements are ensured by the two-stage set locking mechanism. First, the set $W$ is locked at time steps 1 and 3 by adding a path of new vertices $\bar{u}_3$, $\bar{u}_2$, $\bar{u}_1$, $\bar{w}_1$, $\bar{w}_2$, $\bar{w}_3$, and $\bar{w}_4$ (this stage corresponds to Figure 4). Then $\bar{u}_1$ and $\bar{w}_1$ are locked at time steps $T^{\bar{u}_1} = \{1,3,4,5\}$ and $T^{\bar{w}_1} = \{1,3,5\}$, respectively, by the vertex locking technique. The makespan of any optimal solution of $\Sigma^\sim$ is 5 (the same as of $\Sigma'$ in Figure 4).

To prevent agents from $A$ from entering any of the added vertices $V_X$ **second-stage** vertex locking must be done. It is sufficient to **lock** the vertices $u_1$ and $w_1$ since these two vertices are the only connection points of the original graph with the newly added parts. The vertices $u_1$ and $w_1$ need to be locked for all the time steps except for the time steps at which agents of $A_X$ go through them in the optimal solution. More precisely, the vertex $\bar{u}_1$ needs to be locked for time

steps from the set $T^{\bar{u}_1} = \{1,2,\dots,\lambda + t_1\} \setminus (\{t_1, t_2, \dots, t_n\} - 1)$ and the vertex $\bar{w}_1$ needs to be locked for time steps from the set $T^{\bar{w}_1} = \{1,2,\dots,\lambda + t_1\} \setminus (\{t_1, t_2, \dots, t_n\} + 1)$.

Since $\max(T^{\bar{u}_1}) \leq \lambda + t_1$ (as well as $\max(T^{\bar{w}_1}) \leq \lambda + t_1$) and the vertex $\bar{u}_1$ is to be locked for the time steps at which it is not entered as part of some optimal solution, Corollary 1 applies for $\Sigma'$, the locked vertex $\bar{u}_1$, and the set of lock time steps $T^{\bar{u}_1}$. That is, the optimal makespan is preserved. Again, vertex locking is **synchronized** with vertex locking from the first stage. Corollary 1 is subsequently once more applied on the resulting instance with the locked vertex $\bar{w}_1$ and the set of lock time steps $T^{\bar{w}_1}$. Let $\Sigma^{\sim} = (G^{\sim} = (V^{\sim}, E^{\sim}), A^{\sim}, S^0_{A^{\sim}}, S^+_{A^{\sim}})$ denote the final instance, then $Sol^*(\Sigma^{\sim})|_V \subseteq Sol(\Sigma)$. ■

The construction of the two-stage vertex locking from the above proof is shown in Figure 6. As in the case of locking a single vertex, the **size** of all the augmented instances of the problem is $\mathcal{O}(\max(\zeta^*, t_n) + |V| + |E|)$, where $\zeta^*$ is the optimal makespan of $\Sigma$.

**Corollary 4** ***(makespan preserving two-stage set locking).*** Assume that the preconditions (aa), (bb), (cc), and (dd) hold. Then there exists an instance $\Sigma^{\sim} = (G^{\sim} = (V^{\sim}, E^{\sim}), A^{\sim}, S^0_{A^{\sim}}, S^+_{A^{\sim}})$ such that $\Sigma^{\sim}|_V = \Sigma$, and it **never happens** that all the vertices of $W$ are occupied by the original agents of $A$ at any time step $t \in T$ within any **optimal** solution $\mathcal{S}^*_{A^{\sim}}(\Sigma^{\sim})$; moreover, the makespan of any optimal solution of $\Sigma^{\sim}$ is again $\zeta$ and and $Sol^*(\Sigma^{\sim})|_V \subseteq Sol(\Sigma)$ (that is, the **original** agents cannot use any added vertex in any optimal solution). ■

**Proof.** The construction of $\Sigma^{\sim}$ from the proof of Proposition 3 can be adopted with a minor change. Instead of using the construction from the proof of Lemma 1 in the first stage, Corollary 1 is applied. This ensures that the intermediate instance after the first stage of locking preserves the makespan of $\zeta$. The rest of the proof can be applied without any change. ■

### 3.3. *Conjugation – Moving Agents Together to Simulate Propositional Consistency*

We will simulate a valuation of variables of the propositional formula by passing certain pass ways in the graph. There will be two pass ways for each variable – one representing a positive valuation and the other a negative valuation. Since we need to preserve the **propositional consistency** (the positive and negative literals of the same propositional variable should have **complementary** values), a group of agents for valuating a given variable must not be split between these two pass ways. All the agents must pass either the positive branch or the negative branch. Hence, we need some technique that would keep a group of agents together even though they can choose between two alternative pass ways. A technique that ensures such a behavior of agents will be called a ***conjugation technique***.

Let $A = \{\bar{c}_1, \bar{c}_2, \dots, \bar{c}_n\}$ be a set of agents to be conjugated. Formally, the **conjugation** means that there is an instance of the problem of path-finding for multiple agents $\Xi = (G = (V, E), A', S^0_{A'}, S^+_{A'})$, where $V = V^0 \cup V^{\mathcal{L}} \cup V^{\mathcal{R}} \cup V^+$; $V^0, V^{\mathcal{L}}, V^{\mathcal{R}}$, $V^+$ are pair-wise disjoint, $|V^{\mathcal{L}}| = |V^{\mathcal{R}}| = |A|$, $A \subseteq A'$, $S^0_{A'}(A) \subseteq V^0$ (the image of the set by $S^0_{A'}$ is defined naturally: $S^0_{A'}(A) = \{v | (\exists a \in A) S^0_{A'}(a) = v\}$), $S^+_{A'}(A) \subseteq V^+$, and there **exists** a time step $t$ such that within any optimal solution $\mathcal{S}^*_{A'}(\Xi) = [S^0_{A'}, S^1_{A'}, \dots, S^{\zeta}_{A'}]$ either $\{S^t_{A'}(\bar{c}_1), S^t_{A'}(\bar{c}_2), \dots, S^t_{A'}(\bar{c}_n)\} \subseteq V^{\mathcal{L}}$ or $\{S^t_{A'}(\bar{c}_1), S^t_{A'}(\bar{c}_2), \dots, S^t_{A'}(\bar{c}_n)\} \subseteq V^{\mathcal{R}}$ holds. That is, at time step $t$ the whole group of agents that

are conjugated appears either in $V^{\mathcal{L}}$ or $V^{\mathcal{R}}$. No other cases, in which some of the conjugated agents appear in $V^{\mathcal{L}}$ and some in $V^{\mathcal{R}}$, can appear in any optimal solution.

In order to rule out trivial cases of $\Xi$ a requirement that $\{S^*_{A'}(\Xi)|\ S^*_{A'}(\Xi) = [S^0_{A'}, \dots, S^{\zeta}_{A'}] \wedge \{S^t_{A'}(\bar{c}_1), \dots, S^t_{A'}(\bar{c}_n)\} \subseteq V^{\mathcal{L}}\} \neq \emptyset$ and $\{S^*_{A'}(\Xi)|S^*_{A'}(\Xi) = [S^0_{A'}, \dots, S^{\zeta}_{A'}] \wedge \{S^t_{A'}(\bar{c}_1), \dots, S^t_{A'}(\bar{c}_n)\} \subseteq V^{\mathcal{R}}\} \neq \emptyset$ should be taken into account. That is, agents $A$ must go through one of the two alternative pass ways represented by $V^{\mathcal{L}}$ and $V^{\mathcal{R}}$. The task is now to build such an instance of the pCPF problem.

The main idea of the construction is to order the agents of $A$ into a queue that starts with an additional agent called a ***leading agent***. There is a branching in the graph into $V^{\mathcal{L}}$ and $V^{\mathcal{R}}$, which are then joined together, and two leading agents are prepared. The destination for the leading agents is temporarily closed by the construction from Corollary 1. This prevents the leading agents from **escaping** before fulfilling their task. The destination for agents of $A$ is accessible from both the $V^{\mathcal{L}}$ and $V^{\mathcal{R}}$ branch **symmetrically**. The leading agents have no other choice than to lead the group of agents to their destinations. Finally, the leading agent has to go out of the way.

The crucial observation is that if the group of agents $A$ is split between both branches, then the leading agents inevitably block each other causing an obstruction which eliminates any chance to reach the destinations in time (that is, the solution cannot be finished as optimal). Hence, the agent must go into one of the branches of $V^{\mathcal{L}}$ or $V^{\mathcal{R}}$ together (they must conjugate). The formal description of the construction is set out below.

The graph $G = (V, E)$ consists of the following sets of **vertices**:

$V^0 = \{\bar{v}^0_1, \bar{v}^0_2, \dots, \bar{v}^0_n\}$
(called ***initial*** *vertices*),

$V^{\mathcal{L}} = \{\bar{v}^{\mathcal{L}}_1, \bar{v}^{\mathcal{L}}_2, \dots, \bar{v}^{\mathcal{L}}_n\}$
(called ***left*** *vertices*),

$V^{\mathcal{R}} = \{\bar{v}^{\mathcal{R}}_n, \bar{v}^{\mathcal{R}}_{n-1}, \dots, \bar{v}^{\mathcal{R}}_1\}$
(called ***right*** *vertices*),

$V^+ = V^+_{\mathcal{L}} \cup V^+_{\mathcal{R}} \cup V^+_G \cup V^+_Y$
(called ***destination*** *vertices*), with

$V^+_{\mathcal{L}} = \{\bar{v}^{\mathcal{L}}_+, \bar{v}^{\mathcal{L}}_{-2}, \bar{v}^{\mathcal{L}}_{-1}, \bar{v}^{\mathcal{L}}_0\}$
(called ***left part*** of destination vertices)

$V^+_{\mathcal{R}} = \{\bar{v}^{\mathcal{R}}_+, \bar{v}^{\mathcal{R}}_0, \bar{v}^{\mathcal{R}}_{-1}, \bar{v}^{\mathcal{R}}_{-2}\}$
(called ***right part*** of destination vertices)

$V^+_G = \{\bar{v}^+_1, \bar{v}^+_2, \dots, \bar{v}^+_n\}$
(called ***gate part*** of destination vertices) and

$V^+_Y = \{\bar{v}^+_{1,1}, \bar{v}^+_{1,2}, \dots, \bar{v}^+_{1,n}, \bar{v}^+_{2,1}, \bar{v}^+_{2,2}, \dots, \bar{v}^+_{2,n}, \dots, \dots, \bar{v}^+_{\vartheta,1}, \bar{v}^+_{\vartheta,2}, \dots, \bar{v}^+_{\vartheta,n}\}$
(called ***array part*** of destination vertices),

where $\vartheta$ is a parameter determining the length of a solution; it is required that $\vartheta \geq n + 4$. Note that $V^+_Y$ is in fact an array of $\vartheta$ rows of $n$ vertices within $V^+$. In total, the set of vertices is $V = V^0 \cup V^{\mathcal{L}} \cup V^{\mathcal{R}} \cup V^+$.

The **edges** of the graph are as follows:

$E^0 = \{\{\bar{v}^0_1, \bar{v}^{\mathcal{L}}_1\}, \{\bar{v}^0_2, \bar{v}^{\mathcal{L}}_2\}, \dots, \{\bar{v}^0_n, \bar{v}^{\mathcal{L}}_n\}\} \cup \{\{\bar{v}^0_1, \bar{v}^{\mathcal{R}}_n\}, \{\bar{v}^0_2, \bar{v}^{\mathcal{R}}_{n-1}\}, \dots, \{\bar{v}^0_n, \bar{v}^{\mathcal{R}}_1\}\}$
(edges for making a connection between the **initial** vertices and **left**/**right** vertices),

$E^- = \{\{\bar{v}^{\mathcal{L}}_{-2}, \bar{v}^{\mathcal{L}}_{-1}\}, \{\bar{v}^{\mathcal{L}}_{-1}, \bar{v}^{\mathcal{L}}_{0}\}, \{\bar{v}^{\mathcal{L}}_{0}, \bar{v}^{\mathcal{L}}_{1}\}\} \cup \{\{\bar{v}^{\mathcal{R}}_{-2}, \bar{v}^{\mathcal{R}}_{-1}\}, \{\bar{v}^{\mathcal{R}}_{-1}, \bar{v}^{\mathcal{R}}_{0}\}, \{\bar{v}^{\mathcal{R}}_{0}, \bar{v}^{\mathcal{R}}_{1}\}\}$
(edges for connecting the remaining **left/right** vertices),

$E^+ = \{\{\bar{v}^{\mathcal{L}}_{0}, \bar{v}^{+}_{\mathcal{L}}\}, \{\bar{v}^{+}_{L}, \bar{v}^{+}_{1}\}, \{\bar{v}^{+}_{1}, \bar{v}^{+}_{2}\}, \{\bar{v}^{+}_{2}, \bar{v}^{+}_{3}\}, \dots, \{\bar{v}^{+}_{n-1}, \bar{v}^{+}_{n}\}, \{\bar{v}^{+}_{n}, \bar{v}^{+}_{\mathcal{R}}\}, \{\bar{v}^{+}_{\mathcal{R}}, \bar{v}^{\mathcal{R}}_{0}\}\}$
(edges for connecting the **left/right** vertices to the **gate part** of the destination vertices),

$E^+_G = \{\{\bar{v}^{+}_{1}, \bar{v}^{+}_{1,1}\}, \{\bar{v}^{+}_{1}, \bar{v}^{+}_{1,2}\}, \dots, \{\bar{v}^{+}_{1}, \bar{v}^{+}_{1,n}\}\}$
(edges for connecting the **gate part** to the array part of the **destination** vertices),

$E^+_Y = \{\{\bar{v}^{+}_{1,1}, \bar{v}^{+}_{2,1}\}, \dots, \{\bar{v}^{+}_{1,n}, \bar{v}^{+}_{2,n}\}, \dots, \dots, \{\bar{v}^{+}_{\vartheta-1,1}, \bar{v}^{+}_{\vartheta,1}\}, \dots, \{\bar{v}^{+}_{\vartheta-1,n}, \bar{v}^{+}_{\vartheta,n}\}\}$
(edges for connecting rows of the **array part** of the destination vertices),

$E^+_\times = \{\{\bar{v}^{+}_{\vartheta-1,1}, \bar{v}^{+}_{\vartheta,n}\}, \{\bar{v}^{+}_{\vartheta-1,2}, \bar{v}^{+}_{\vartheta,n-1}\}, \dots, \{\bar{v}^{+}_{\vartheta-1,n}, \bar{v}^{+}_{\vartheta,1}\}\}$
(edges for connecting the **last row** of the **array part** in the **reversed** order);

in total, the set of edges of the graph $G$ is $E = E^0 \cup E^- \cup E^+ \cup E^+_G \cup E^+_Y \cup E^+_\times$.

The **set of agents** is extended with two leading agents $\bar{l}_{\mathcal{L}}$ and $\bar{l}_{\mathcal{R}}$ (the left and the right leading agent); that is, $A' = A \cup \{\bar{l}_{\mathcal{L}}, \bar{l}_{\mathcal{R}}\}$. The **initial arrangement** of agents is as follows: $S^0_{A'}(\bar{c}_i) = \bar{v}^0_i$ for $i = 1,2,\dots,n$; $S^0_{A'}(\bar{l}_{\mathcal{L}}) = \bar{v}^{\mathcal{L}}_0$ and $S^0_{A'}(\bar{l}_{\mathcal{R}}) = \bar{v}^{\mathcal{R}}_0$. That is, the original agents are placed into the initial vertices while the leading agents are placed in such a way that original agents can join either of them. The **goal arrangement** is: $S^+_{A'}(\bar{c}_i) = \bar{v}^+_{\vartheta,i}$ for $i = 1,2,\dots,n$; $S^+_{A'}(\bar{l}_{\mathcal{L}}) = \bar{v}^{\mathcal{L}}_{-2}$ and $S^+_{A'}(\bar{l}_{\mathcal{R}}) = \bar{v}^{\mathcal{R}}_{-2}$; that is, the original agents should finally reach the last row of the array part of the destination vertices and the leading agents should go out of the way.

The required conjugation of agents into the left and right vertices at a certain time step can be satisfied if the agents move as follows: all the agents $\bar{c}_1, \bar{c}_2, \dots, \bar{c}_n$ from the set of vertices $\bar{v}^0_1, \bar{v}^0_2, \dots, \bar{v}^0_n$ move into the set of vertices $\bar{v}^{\mathcal{L}}_1, \bar{v}^{\mathcal{L}}_2, \dots, \bar{v}^{\mathcal{L}}_n$ if the left branch is chosen, or into the set of vertices $\bar{v}^{\mathcal{R}}_1, \bar{v}^{\mathcal{R}}_2, \dots, \bar{v}^{\mathcal{R}}_n$ if the right branch is chosen.

Without loss of generality, suppose the left branch has been chosen. Agents $\bar{c}_1, \bar{c}_2, \dots, \bar{c}_n$, together with the leading agent $\bar{l}_{\mathcal{L}}$, then move into vertices $\bar{v}^+_1, \bar{v}^+_2, \dots, \bar{v}^+_n, \{\bar{v}^+_{\mathcal{R}}, \bar{v}^{\mathcal{R}}_0\}$. This is followed by the movement of agents $\bar{c}_1, \bar{c}_2, \dots, \bar{c}_n$ towards the last row of the array part of the destination vertices where their order is eventually **reversed** (if the right branch has been chosen, no reversing is necessary). Leading agents return to their goal locations in $\bar{v}^{\mathcal{L}}_{-2}$ and $\bar{v}^{\mathcal{R}}_{-2}$ at the same time. The described behavior of agents within the optimal solution is ensured by locking appropriate vertices at appropriate time steps. That is, the pCPF instance $\Xi$ is further extended with additional agents and vertices used for locking vertices, as shown in the proof of Corollary 1. However, for the sake of simplicity, the description below will be restricted to the original components of the problem $\Xi$.

Thus, the optimal solution for the **left branch** $\mathcal{S}^*_{\mathcal{L},A'}(\Xi) = [S^0_{\mathcal{L},A'}, S^1_{\mathcal{L},A'}, \dots, S^\zeta_{\mathcal{L},A'}]$ should satisfy that $S^0_{\mathcal{L},A'}(\bar{c}_i) = \bar{v}^0_i$, $S^1_{\mathcal{L},A'}(\bar{c}_i) = \bar{v}^{\mathcal{L}}_i$, $S^2_{\mathcal{L},A'}(\bar{c}_i) = \bar{v}^{\mathcal{L}}_{i-1}$, …, $S^i_{\mathcal{L},A'}(\bar{c}_i) = \bar{v}^{\mathcal{L}}_1$, $S^{i+1}_{\mathcal{L},A'}(\bar{c}_i) = \bar{v}^{\mathcal{L}}_0$, $S^{i+2}_{\mathcal{L},A'}(\bar{c}_i) = \bar{v}^+_{\mathcal{L}}$, $S^{i+3}_{\mathcal{L},A'}(\bar{c}_i) = \bar{v}^+_1$, $S^{i+4}_{\mathcal{L},A'}(\bar{c}_i) = \bar{v}^+_2$, …, $S^{n+2}_{\mathcal{L},A'}(\bar{c}_i) = \bar{v}^+_{n-i}$, $S^{n+3}_{\mathcal{L},A'}(\bar{c}_i) = \bar{v}^+_{n-i+1}$, $S^{n+4}_{\mathcal{L},A'}(\bar{c}_i) = \bar{v}^+_{1,n-i+1}$, $S^{n+5}_{\mathcal{L},A'}(\bar{c}_i) = \bar{v}^+_{2,n-i+1}$, …, $S^{n+\vartheta+2}_{\mathcal{L},A'}(\bar{c}_i) = \bar{v}^+_{\vartheta-1,n-i+1}$, and $S^{n+\vartheta+3}_{\mathcal{L},A'}(\bar{c}_i) = \bar{v}^+_{\vartheta,i} = S^+_{A'}(\bar{c}_i)$ for $i = 1,2,\dots,n$; $S^0_{\mathcal{L},A'}(\bar{l}_{\mathcal{L}}) = \bar{v}^{\mathcal{L}}_0$, $S^1_{\mathcal{L},A'}(\bar{l}_{\mathcal{L}}) \in \{\bar{v}^{\mathcal{L}}_0, \bar{v}^+_{\mathcal{L}}\}$, $S^2_{\mathcal{L},A'}(\bar{l}_{\mathcal{L}}) \in \{\bar{v}^+_{\mathcal{L}}, \bar{v}^+_1\}$, $S^3_{\mathcal{L},A'}(\bar{l}_{\mathcal{L}}) \in \{\bar{v}^+_1, \bar{v}^+_2\}$, …, $S^{n+1}_{\mathcal{L},A'}(\bar{l}_{\mathcal{L}}) \in \{\bar{v}^+_{n-1}, \bar{v}^+_n\}$, $S^{n+2}_{\mathcal{L},A'}(\bar{l}_{\mathcal{L}}) \in \{\bar{v}^+_n, \bar{v}^+_{\mathcal{R}}\}$, $S^{n+3}_{\mathcal{L},A'}(\bar{l}_{\mathcal{L}}) \in \{\bar{v}^+_{\mathcal{R}}, \bar{v}^{\mathcal{R}}_0\}$ (the left leading agent is going in front of the queue formed by the sequence of agents $\bar{c}_1, \bar{c}_2, \dots, \bar{c}_n$), there is no special requirement on $S^{n+4}_{\mathcal{L},A'}(\bar{l}_{\mathcal{L}})$, $S^{n+5}_{\mathcal{L},A'}(\bar{l}_{\mathcal{L}})$, …, $S^{n+\vartheta+2}_{\mathcal{L},A'}(\bar{l}_{\mathcal{L}})$, indeed $S^{n+\vartheta+3}_{\mathcal{L},A'}(\bar{l}_{\mathcal{L}}) = \bar{v}^{\mathcal{L}}_{-2} = S^+_{A'}(\bar{l}_{\mathcal{L}})$. Similarly, there is no special requirement on $S^i_{\mathcal{L},A'}(\bar{l}_{\mathcal{R}})$ for any

$i = 1,2,\dots,n$. The optimal solution for the **right branch** $\mathcal{S}^{*}_{\mathcal{R},A'}(\Xi) = [S^{0}_{\mathcal{R},A'}, S^{1}_{\mathcal{R},A'}, \dots, S^{\zeta}_{\mathcal{R},A'}]$ has almost the same form. The only difference is that the final reversal of the agents $\bar{c}_1, \bar{c}_2, \dots, \bar{c}_n$ to fit into the last row of the array part of the destination vertices is not performed. Observe that the time step at which the **conjugation** occurs is $t = 1$.

Now, the task is to show that the described behavior is feasible and no other behavior can occur within any optimal solution. In other words, any optimal solution of the problem has either the form of the solution for the **left branch** or the solution for the **right branch**.

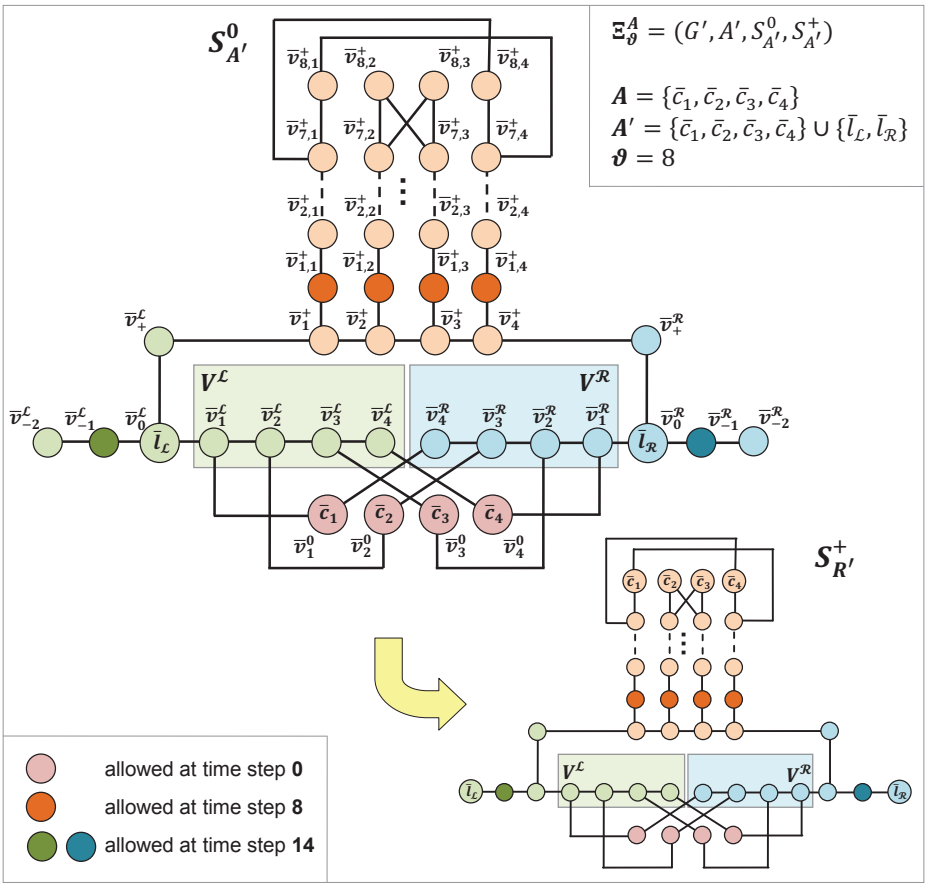



**Figure 7.** *A conjugation instance of the pCPF problem.* The conjugation instance $\Xi^{A}_{\vartheta}$ shown in the figure is constructed with respect to a set of agents $A = \{\bar{c}_1, \bar{c}_2, \bar{c}_3, \bar{c}_4\}$ and a parameter $\vartheta = 8$. The agents are restricted in their movements using vertex locking – namely, the initial vertices $\bar{v}^{0}_{1}$, $\bar{v}^{0}_{2}$, $\bar{v}^{0}_{3}$, and $\bar{v}^{0}_{4}$ can be entered only at time step 0; the vertices $\bar{v}^{+}_{1,1}$, $\bar{v}^{+}_{1,2}$, $\bar{v}^{+}_{1,3}$, and $\bar{v}^{+}_{1,4}$ can be entered only at time step 8; and the vertices $\bar{v}^{\mathcal{L}}_{-1}$ and $\bar{v}^{\mathcal{R}}_{-1}$ can be entered only at time step 14. These conditions enforce that the agents $\bar{c}_1$, $\bar{c}_2$, $\bar{c}_3$, and $\bar{c}_4$ are located either in vertices $\bar{v}^{\mathcal{L}}_{1}$, $\bar{v}^{\mathcal{L}}_{2}$, $\bar{v}^{\mathcal{L}}_{3}$, and $\bar{v}^{\mathcal{L}}_{4}$ or in vertices $\bar{v}^{\mathcal{R}}_{1}$, $\bar{v}^{\mathcal{R}}_{2}$, $\bar{v}^{\mathcal{R}}_{3}$, and $\bar{v}^{\mathcal{R}}_{4}$ at time step 1 in any optimal solution of $\Xi^{A}_{\vartheta}$.

The first row of the array part of the destination vertices, that is, vertices $\bar{v}^{+}_{1,1}, \bar{v}^{+}_{1,2}, \dots, \bar{v}^{+}_{1,n}$, is locked (closed for entering) for all the time steps except for time step $n + 4$. At this time step, all the agents $\bar{c}_1, \bar{c}_2, \dots, \bar{c}_n$ are entering the array part of the destination vertices. Then they continue towards their goal locations and hence vertices $\bar{v}^{+}_{1,1}, \bar{v}^{+}_{1,2}, \dots, \bar{v}^{+}_{1,n}$ can be locked again for

the remaining time steps. The vertices $\bar{v}_{-1}^{\mathcal{L}}$ and $\bar{v}_{-1}^{\mathcal{R}}$ are locked for all the time steps except the time step $n + \vartheta + 2$. Similarly, the initial vertices are locked for all the time steps except for time step 0.

At the time of opening the first row of the array part of the destination vertices (at time step $n + 3$), all the agents $\bar{c}_1, \bar{c}_2, \ldots, \bar{c}_n$ must reside in the vertices $\bar{v}_1^+, \bar{v}_2^+, \ldots, \bar{v}_n^+$ (eventually in the reverse order). Otherwise, they have no chance to reach their goal locations at all. Then, the fastest way to reach their goal locations starting from vertices $\bar{v}_1^+, \bar{v}_2^+, \ldots, \bar{v}_n^+$ is to exactly follow the shortest paths to the last row of the array part of the destination vertices (all these paths are of the same length). Since $\vartheta \geq n + 4$, which is enough time steps for the leading agents to reach their destination locations; the motion of agents $\bar{c}_1, \bar{c}_2, \ldots, \bar{c}_n$ within the array part of the destination vertices represents the **bottleneck**.

It remains to check the behavior of agents before time step $n + 3$. Since the initial vertices are only allowed to be occupied at time step 0, the agents $\bar{c}_1, \bar{c}_2, \ldots, \bar{c}_n$ must enter the left or the right vertices immediately at the next time step. Between time steps 1 and $n + 3$, it is **impossible to swap** agents in the currently accessible part of the graph since it consists of a single path. Hence, if the agents $\bar{c}_1, \bar{c}_2, \ldots, c_n$ split between the left and the right vertices, then they cannot be arranged into vertices $\bar{v}_1^+, \bar{v}_2^+, \ldots, \bar{v}_n^+$ in the required order, because they are **obstructed** by the leading agents $\bar{l}_{\mathcal{L}}$ and $\bar{l}_{\mathcal{R}}$.

The just described instance will be called a ***conjugation*** *instance of the pCPF problem.* Note that the instance is parameterized by a set of agents $A$ and an integer parameter $\vartheta \geq |A| + 4$. An instance of the problem corresponding to the given parameters will be denoted as $\Xi_\vartheta^A$. Note further that the makespan of any optimal solution of $\Xi_\vartheta^A$ is $|A| + \vartheta + 3 \geq 2|A| + 7$. It is easy to see that the **size** of $\Xi_\vartheta^A$ is $(3 + \vartheta)|A| + 8$, which is $\mathcal{O}(\vartheta|A|)$.

An example of a conjugation instance is shown in Figure 7. Although some edges of the conjugation instance intersect, it is just a matter of a graph drawing in a plane. There is actually no interference between agents traversing edges that intersect (notice further that pCPF may take place in high dimensional spaces that cannot be drawn in a plane without intersecting an edge).

### 3.4. *NP-completeness of pCPF*

All the ingredients have been prepared to prove that a *decision version* of the optimal pCPF is $NP$-complete. The **membership** in ***NP*** will be checked first. Subsequently, a **polynomial time reduction** of a *propositional satisfiability* instance (***SAT***) **[**1**]** to the instance of the decision version of the optimal pCPF will be constructed.

**Definition 3** ***(decision version of pCPF).*** A *decision version of the optimal pCPF* is a task designed to decide whether, for a given instance of pCPF $\Sigma$ and a number $\eta \in \mathbb{N}_0$, there exists a solution $\mathcal{S}_A(\Sigma)$ of the makespan at most $\eta$. A notation $\Sigma/\eta$ will be used for the decision instance. Next, let $pCPF_{OPT}$ denote the language of positive instances of this problem. □

It is not that easy to see that $pCPF_{OPT} \in NP$, since no upper bound on the size of the solution of $pCPF_{OPT}$ has so far been established. Hence, the standard technique of "guessing and checking" cannot be used immediately. Note that decision variants of several related *sliding piece problems* [10] such as the ***Sokoban*** *game* [4] and the ***Rush-hour*** *puzzle* [7] are proven to be $PSPACE$-complete [8, 9] but it is not known whether they are in $NP$. The reason is that the pol-

ynomial upper bound on the size of the solution has not yet been found. Fortunately, this is not the case of $pCPF_{OPT}$. It is possible to establish the polynomial upper bound on the size of the solution of $pCPF_{OPT}$ using results shown in [13].

**Lemma 3.** $pCPF_{OPT} \in NP$. ■

**Proof.** It has been shown in [13] that there exists a solution $\mathcal{S}_P(\Pi) = [S_P^0, S_P^1, \dots, S_P^\xi]$ for any solvable instance of the problem of PMG $\Pi = (G = (V, E), P, S_P^0, S_P^+)$ such that $\xi \in \mathcal{O}(|V|^3)$ ($\xi$ is regarded as a function of $\Pi$ here). Since the solution of an instance of PMG can be used as a solution of the corresponding pCPF instance (Proposition 1), it implies that there exists a solution $\mathcal{S}_A(\Sigma) = [S_A^0, S_A^1, \dots, S_A^\zeta]$ for any solvable instance of the problem of pCPF $\Sigma = (G = (V, E), A, S_A^0, S_A^+)$ such that $\zeta \in \mathcal{O}(|V|^3)$ ($\zeta$ is also regarded a function of $\Sigma$). An instance of $pCPF_{OPT}$ $\Sigma/\eta$ can be solved on a Turing machine with an oracle in polynomial time as follows. A solution of the size $\mathcal{O}(|V|^3)$ of $\Sigma$ is generated first by the oracle. Then, the generated solution is checked as to whether its makespan is at most $\eta$ and whether it satisfies Definition 2. This check can be carried out in polynomial time with respect to the size of $\Sigma/\eta$. ■

Propositional satisfiability is a decision problem where the question is whether a given propositional formula has a satisfying valuation or not. As it is usual, propositional formulas in a *conjunctive normal form* (*CNF*) [12] are considered. Let $SAT$ denote the language of satisfiable instances of propositional formulas in CNF as it is formalized in the following definition.

**Definition 4** ***(propositional satisfiability – SAT).*** A *propositional variable* is a variable that can be evaluated as either $TRUE$ or $FALSE$. A *literal* is a propositional variable or its negation. A *clause* is a disjunction of literals; that is, $\bigvee_{i=1}^{n} l_i$, where $n \in \mathbb{N}_0$ and $l_i$ is a literal for $i = 1,2,\dots,n$. A *propositional formula in CNF* is a conjunction of clauses; that is, $\bigwedge_{j=1}^{m} C_j$, where $m \in \mathbb{N}_0$ and $C_j$ is a clause for $i = 1,2,\dots,m$. Let $Var(F)$ denote the set of propositional variables of the CNF formula $F$, then the *valuation* of variables of $F$ is an assignment $e: Var(F) \longrightarrow \{FALSE, TRUE\}$. The valuation of variables is naturally extended from variables to literals, clauses, and the complete CNF formula. The *propositional satisfiability* problem (SAT) is a decision problem where the question is whether a given formula $F$ in CNF has a valuation of its variables so that $F$ evaluates to $TRUE$ under this valuation. □

If the CNF formula $F$ has a satisfying valuation, then $F$ is said to be *satisfiable* or, otherwise, it is said to be *unsatisfiable* (examples of several propositional formulae in CNF are shown in Figure 8). It is well known that $SAT$ is $NP$-complete. However, a slight technical adaptation of propositional satisfiability is necessary to carry out the required reduction to pCPF. A restriction on formulas in CNF where positive and negative *literals* of the same variable have the same number of occurrences in the formula will be made. Let the language of satisfiable formulas that comply with this restriction be denoted as $SAT_=$ (see Figure 8 again).

**Definition 5** ***(equality propositional satisfiability – SAT$_=$).*** Let $F$ be a propositional formula in CNF. Next, let $pos(x, F)$ with $x \in Var(F)$ denote the set of positive occurrences of $x$ in $F$ and, similarly, let $neg(x, F)$ denote a set of negative occurrences of $x$ in $F$. The *equality proposi-*

*tional satisfiability problem* ($SAT_=$) is a decision problem where the question is whether a given formula $F$ in CNF such that $|pos(x,F)| = |neg(x,F)|$ for every $x \in Var(F)$ is satisfiable or not. □

**Lemma 4.** $SAT_=$ is $NP$-complete. ■

**Proof.** With respect to the membership in $NP$, the restriction makes no change; thus $SAT_= \in NP$. Any instance of $SAT$ can be reduced to an instance of $SAT_=$ by adding *clauses* to balance the number of positive and negative literals of the same variable. The added clauses should preserve equisatisfiability of the resulting formula with the original one.

$\boldsymbol{F_1} = \underbrace{(\neg x_1 \vee \neg x_2)}_{C_1} \wedge \underbrace{x_1}_{C_2} \wedge \underbrace{x_2}_{C_3}$

**unsatisfiable**

$Var(F_1) = \{x_1, x_2\}$ $\quad pos(x_1, F_1) = \{C_2\}$ $\quad neg(x_1, F_1) = \{C_1\}$
$\quad pos(x_2, F_1) = \{C_3\}$ $\quad neg(x_2, F_1) = \{C_1\}$

$\boldsymbol{F_2} = \underbrace{(x_1 \vee \neg x_2 \vee x_3)}_{C_1} \wedge \underbrace{(\neg x_1 \vee \neg x_2 \vee \neg x_3)}_{C_2}$

**satisfiable** with $e(x_1) = TRUE, e(x_2) = FALSE, e(x_3) = TRUE$

$Var(F_2) = \{x_1, x_2, x_2\}$ $\quad pos(x_1, F_2) = \{C_1\}$ $\quad neg(x_1, F_2) = \{C_2\}$
$\quad pos(x_2, F_2) = \emptyset$ $\quad neg(x_2, F_2) = \{C_1, C_2\}$
$\quad pos(x_3, F_2) = \{C_1\}$ $\quad neg(x_3, F_2) = \{C_2\}$

$\boldsymbol{F_3} = \underbrace{(x_1 \vee \neg x_2 \vee x_3)}_{C_1} \wedge \underbrace{(\neg x_1 \vee \neg x_2 \vee \neg x_3)}_{C_2} \wedge \underbrace{(x_2 \vee x_2 \vee y_1 \vee \neg y_1)}_{C_3}$

**satisfiable** with $e(x_1) = TRUE, e(x_2) = FALSE, e(x_3) = TRUE, e(y_1) = TRUE$

$Var(F_3) = \{x_1, x_2, x_2, y_1\}$ $\quad pos(x_1, F_3) = \{C_1\}$ $\quad neg(x_1, F_3) = \{C_2\}$
$\quad pos(x_2, F_3) = \{C_3, C_3'\}$ $\quad neg(x_2, F_3) = \{C_1, C_2\}$
$\quad pos(x_3, F_3) = \{C_1\}$ $\quad neg(x_3, F_3) = \{C_2\}$
$\quad pos(y_1, F_3) = \{C_3\}$ $\quad neg(y_1, F_3) = \{C_3\}$

**Figure 8.** ***Examples*** *of propositional formulae in* ***CNF***. Three formulae $F_1, F_2$, and $F_3$ are shown. $F_1$ is unsatisfiable while the other two are satisfiable. Positive and negative occurrences of literals in the formulae are depicted. Note that the number of positive and negative occurrences of $x_2$ in $F_2$ is unbalanced; that is, $F_2 \notin SAT_=$ (satisfiable but syntactically incorrect). The result of the rebalancing of $F_2$ is $F_3 \in SAT_=$ (both satisfiable and syntactically correct).

Let $F$ be a formula in CNF and let $x$ be a variable with **unbalanced** positive and negative occurrences. Without loss of generality, let $|pos(x,F)| < |neg(x,F)|$. Then a clause $(\bigvee_{i=1}^{|neg(x,F)|-|pos(x,F)|} x) \vee y \vee \neg y$, where $y$ is a new variable, is added to $F$. Now $x$ as well as the newly added $y$ have the same number of positive and negative occurrences. Clearly, the resulting formula is equisatisfiable with $F$ since the newly added clause is always satisfied. The

described process should be done for all the unbalanced variables. The length of the resulting formula is at most twice that of $F$, thus the reduction can be done in polynomial time. ■

**Theorem 1.** $pCPF_{OPT}$ is $NP$-complete. ■

**Proof.** It remains to prove that $pCPF_{OPT}$ is $NP$-hard. A polynomial time reduction of $SAT_{=}$ to $pCPF_{OPT}$ will be used. Let $F_{=}$ be a formula in CNF, that is, $F_{=} = \bigwedge_{i=1}^{n}(\bigvee_{j}^{k_i} l_j^i)$, where $l_j^i$ is the $j$-th literal of the $i$-th clause; there are $n$ clauses, where the $i$-th clause has $k_i$ literals.

Assume further that that each variable has the same number of positive and negative occurrences in $F_{=}$. Let $Var(F_{=})$ denote the set of propositional variables of $F_{=}$. An instance $\Sigma = (G = (V, E), A, S_A^0, S_A^+)/\eta$ of the decision version of the optimal pCPF for $F_{=}$ will be constructed in the following way. Every occurrence of a literal in $F_{=}$ will be associated with a vertex. Thus, a set of vertices $V^{F_{=}} = \bigcup_{i=1}^{n} \bigcup_{j=1}^{k_i} \{\bar{l}_j^i\}$ is constructed ($\bar{l}_j^i$ is a symbol while $l_j^i$ is a variable standing for a literal); a vertex $\bar{l}_j^i$ corresponds to an occurrence of a literal $l_j^i$ in the $i$-th clause as the $j$-th disjunct. A **conjugation** instance of pCPF will be associated with each propositional variable of $F_{=}$ while left and right vertices of the conjugation graph will be one-to-one matched to vertices from $V^{F_{=}}$ that correspond to negative and positive occurrences of the variable, respectively. This is possible since there is the same number of positive and negative occurrences of each variable in $F_{=}$ (a conjugation graph has the same number of left and right vertices).

The idea is to prepare a group of agents of the size $|pos(x, F_{=})| = |neg(x, F_{=})|$ for each propositional variable $x \in Var(F_{=})$. This group of agents will be placed in the initial vertices of the conjugation subgraph corresponding to $x$. The construction of the conjugation subgraph will enforce that all the agents must go either into the vertices corresponding to positive literals or into the vertices corresponding to negative literals. If the movement of agents is interpreted in the way that literals corresponding to vertices of $V^{F_{=}}$ visited at time step 1 will be assigned the same propositional value, then the conjugation technique assures **propositional consistency** of the assignment. However, this is not enough to establish correspondence between an assignment satisfying $F_{=}$ and a solution of $\Sigma/\eta$. It is furthermore necessary to make agents simulate **clause satisfaction** by means of any solution whose makespan is at most $\eta$.

This can be done by enforcing agents either to visit at least one literal/vertex of each clause of $F_{=}$ (in cases when the visited literals/vertices are assigned the value $TRUE$) or leave at least one literal/vertex of each clause of $F_{=}$ unoccupied at time step 1 (in cases where the visited literals/vertices are assigned the value $FALSE$). Since the second option can easily be implemented through the vertex **set locking** mechanism (Proposition 3, Corollary 4), the value $FALSE$ will be used for literals corresponding to vertices visited at time step 1.

Nevertheless, some technical details such as the exact specification of $\eta$ need to be discussed. Equality between makespans of optimal solutions over the individual conjugation instances needs to be established.

Recall that a conjugation instance $\Xi_{\vartheta}^{A}$ is characterized by two parameters: $A$ – the set conjugated agents and $\vartheta$ – parameter affecting the makespan of the optimal solution of the instance. Let $\varrho = max_{x \in Var(F_{=})}\{|pos(x, F_{=})| = |neg(x, F_{=})|\}$. For a given $x \in Var(F_{=})$, the conjugation instance $\Xi_{\vartheta(x)}^{A(x)}$ will have the parameters $A(x) = \{\bar{c}_1^x, \bar{c}_2^x, \ldots, \bar{c}_{|pos(x,F_{=})|}^x\}$ and $\vartheta(x) = 2\varrho -$

$|pos(x, F_=)| + 4 \geq |A(x)| + 4$. Hence, the makespan of any optimal solution of the conjugation instance $\Xi^{A(x)}_{\vartheta(x)}$ is $|A(x)| + \vartheta(x) + 3 = 2\varrho + 7$.

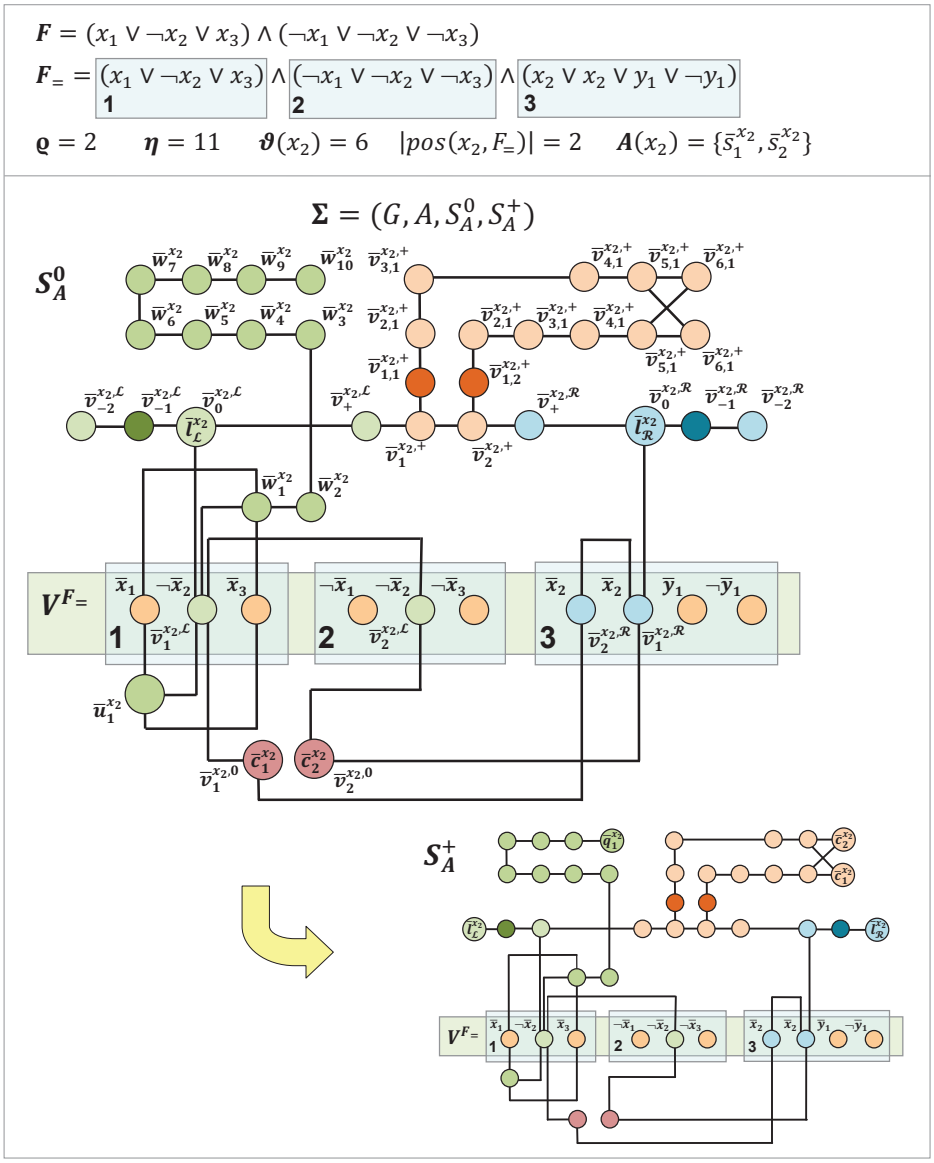



**Figure 9.** *A polynomial time* ***reduction*** *of a propositional formula to a decision instance of pCPF.* A formula $F$ is transformed to a formula $F_=$ in which each variable has the same number of positive and negative occurrences. Subsequently, an instance of the decision version of the problem of pCPF $\Sigma/\eta$ is constructed. The conjugation technique is used to simulate propositional consistency and the set locking technique is used to simulate clause satisfaction (the reduction of one variable using the conjugation technique and the reduction of one clause using the set locking technique are shown). There exists a solution of Σ of the makespan $\eta = 11$ if and only if the formula $F$ is satisfiable.

Left and right vertices of $\Xi_{\vartheta(x)}^{A(x)}$ are matched with vertices from $V^{F_=}$ as follows: $\{\bar{v}_1^{x,\mathcal{L}}, \bar{v}_2^{x,\mathcal{L}}, \dots, \bar{v}_{|A(x)|}^{x,\mathcal{L}}\} = \{\bar{l}_j^i | l_j^i = \neg x; i = 1,2,..n; j = 1,2,\dots,k_i\}$ and $\{\bar{v}_1^{x,\mathcal{R}}, \bar{v}_2^{x,\mathcal{R}}, \dots, \bar{v}_{|A(x)|}^{x,\mathcal{R}}\} = \{\bar{l}_j^i | l_j^i = x; i = 1,2,..n; j = 1,2,\dots,k_i\}$. Now, a crucial observation has to be made. It holds that $F_=$ is satisfiable if and only if there **exists a solution** of the currently constructed instance of the makespan of $2\varrho + 7$ such that at time step 1 at least one vertex from the set of vertices corresponding to each clause remains unoccupied.

Let $e: Var(x) \longrightarrow \{FALSE, TRUE\}$ be a satisfying valuation of $F_=$. If $e(x) = FALSE$, then agents $\bar{c}_1^x, \bar{c}_2^x, \dots, \bar{c}_{|pos(x,F_=)|}^x$ are placed in $\bar{v}_1^{x,\mathcal{R}}, \bar{v}_2^{x,\mathcal{R}}, \dots, \bar{v}_{|A(x)|}^{x,\mathcal{R}}$ at time step 1; if $e(x) = TRUE$ then they are placed in $\bar{v}_1^{x,\mathcal{L}}, \bar{v}_2^{x,\mathcal{L}}, \dots, \bar{v}_{|A(x)|}^{x,\mathcal{L}}$ at time step 1. The placement of agents at time steps other than 1 is straightforward. Since $e$ is the satisfying assignment, at least one vertex from the set of vertices corresponding to each clause remains unoccupied. On the other hand, Corollary 4 can be used to augment the instance to enforce that at least one vertex from the set of vertices that corresponds to literals of a clause is not occupied by agents from the set $\bigcup_{x \in Var(F_=)} A(x)$ within **any optimal solution** while the makespan of $2\varrho + 7$ remains preserved. That is, Corollary 4 is invoked with $W = \{\bar{l}_1^i, \bar{l}_2^i, \dots, \bar{l}_{k_i}^i\}$ that corresponds to satisfying the $i$th clause of $F_=$. Let $\Sigma$ denote the resulting instance. Any solution of the makespan of $2\varrho + 7$ of $\Sigma$ satisfies conditions at time step 1 and hence it induces a satisfying assignment of $F_=$.

The construction of $\Sigma$ requires **polynomial** time in the size of $F_=$; the size of $\Sigma$ is also polynomial (the size of each conjugation subgraph is polynomial in the size of $F_=$ and the number of conjugation subgraphs is bounded by the size of $F_=$). Now, if $\Sigma$ has a solution of the makespan $\eta = 2\varrho + 7$ then it is ensured that **conjugation** and **clause satisfaction** has been successfully simulated, thus a satisfying valuation of $F_=$ can easily be derived from this solution. Hence, $F_= \in SAT_=$ if and only if $\Sigma/\eta \in pCPF_{OPT}$. Together with Lemma 3 the claim that $pCPF_{OPT}$ is $NP$-complete has been obtained. ■

The reduction described in the proof is illustrated in Figure 9. The illustration shows the instantiation of the conjugation mechanism over a single variable. Figure 9 also represents the connection of the simulation of clause satisfaction to the conjugation mechanism.

## 4. Related Works and Conclusion

A **parallel** version of the cooperative path-finding problem (pCPF) is introduced in this paper. The new theoretical result shown in this paper is that the decision version of the optimal pCPF is ***NP*-complete**. The parameter, which is optimized, is the makespan, that is the maximum of arrival times to a destination over all the agents.

The **reduction** of propositional satisfiability to pCPF has been used for the proof of *NP*-hardness. Numerous techniques to simulate propositional consistency and clause satisfaction within pCPF were developed in this work. These techniques were inspired by works on multi-commodity flows [6]. We assume the existence of developed techniques generic enough to be used in different contexts. Vertex locking and conjugation techniques have recently been used in the proof of *NP*-hardness designed to check the existence of a winning strategy in the so-called *adversarial CPF* (ACPF) [29]. ACPF is CPF with multiple teams of agents that compete in reaching their goals.

The finding that optimal pCPF is *NP*-complete is rather negative. Fortunately, if the requirement on the shortest possible makespan of solutions is relaxed, the problem becomes **tractable**. In particular, it belongs to the *P* class. However, the situation is not as straightforward. Although algorithms developed for solving PMG/CPF [13, 37] can be used for solving pCPF, this practice is disadvantageous. Despite a promising theoretical makespan of $\mathcal{O}(|V|^3)$ of solutions generated by these algorithms, the makespan measured empirically is relatively high [28] due to a large constant in the estimation. Therefore, alternative solving sub-optimal algorithms for pCPF producing better solutions (so called *BIBOX* algorithms) and solution-improving techniques have been proposed [23, 24, 25, 28]. Recently, there has been a considerable development in sub-optimal algorithms for CPF represented by works [30].

An important **related work** is referred to in articles [30, 31, 32, 33]. Its authors study a version of pCPF similar to the one presented in this paper. The authors define a tractable class of this problem where graphs are restricted to grids and there is a relative abundance of unoccupied vertices.

Several attempts to find an optimal solution of the standard non-parallel CPF have been made. An algorithm based on A* has been presented in [21]. The algorithm is suitable for CPF instances with few agents and a plenty of free space in the graphs. An alternative approach to solving CPF optimally is to translate CPF to SAT, as has been suggested in [30]. Interestingly, SAT-based methods seem to be complementary to A* since they perform well on densely occupied instances.

An interesting question for future work is whether it is feasible to find a solution of a pCPF instance that is constantly worse than the optimum. Currently, it is an open question whether such an **approximation** algorithm exists. The answer to this question will therefore provide an estimate of how far from the optimum the solutions generated by algorithms for the non-optimization case of the problem are. Thus, an estimate of the makespan of the optimal solution of large instances would also be made available.

## Glossary

| | |
|---|---|
| PMG | pebble motion on a graph |
| $G = (V, E)$ | an undirected graph; $V$ denotes a set of vertices; $E$ denotes a set of edges |
| $P$ | a set of pebbles |
| $\mu$ | the number of pebbles |
| $\bar{p}_i$ | a pebble |
| $S_P^0 : P \longrightarrow V$ | the initial arrangement of pebbles |
| $S_P^+ : P \longrightarrow V$ | the goal arrangement of pebbles |
| $S_P^t : P \longrightarrow V$ | the arrangement of pebbles at time step $t$ |
| $\xi$ | the makespan of a solution of PMG and a sequence of pebble arrangements |
| $\mathcal{S}_P$ | a sequence of arrangements of pebbles forming a solution of PMG |
| $\Pi = (G, P, S_P^0, S_P^+)$ | an instance of PMG |
| $\mathcal{S}_P(\Pi)$ | a solution to the instance of PMG |
| CPF | cooperative path-finding |
| pCPF | parallel cooperative path-finding |
| $A$ | a set of agents |
| $\nu$ | the number of agents |

| | |
|---|---|
| $\bar{a}_i$ | an agent |
| $S_A^0: A \longrightarrow V$ | the initial arrangement of agents |
| $S_A^+: A \longrightarrow V$ | the goal arrangement of agents |
| $S_A^t: A \longrightarrow V$ | the arrangement of agents at time step $t$ |
| $\zeta$ | the makespan of a solution of pCPF |
| $\zeta^*$ | the makespan of an optimal solution of pCPF |
| $\mathcal{S}_A$ | a sequence of arrangements of agents forming a solution of pCPF |
| $\Sigma = (G, A, S_A^0, S_A^+)$ | an instance of pCPF |
| $\mathcal{S}_A(\Sigma)$ | a solution to the instance of pCPF |
| $Sol(\Sigma)$ | a set of solutions of a pCPF instance Σ |
| $Sol^*(\Sigma)$ | a set of makespan optimal solutions of a pCPF instance Σ |
| $\Xi_\vartheta^A$ | a conjugation instance of pCPF; $\vartheta$ is the length of a solution |
| $pCPF_{OPT}$ | a language consisting of pairs $\Sigma/\eta$ where Σ is a pCPF solvable by solution of the makespan at most $\eta$ |
| $SAT$ | a language consisting of satisfiable propositional formulas in CNF |
| $SAT_=$ | a subset of $SAT$ where each variable has the same number of positive and negative occurrences |
| $F$ | a propositional formula in CNF |
| $F_=$ | a propositional formula in CNF where each variable has the same number of its positive and negative occurrences |
| $pos(x, F)$ | a set of positive occurrences of a propositional variable $x$ in $F$ |
| $neg(x, F)$ | a set of negative occurrences of a propositional variable $x$ in $F$ |
| $O\|_V$ | the restriction of an object $O$ on a set of vertices $V$ |
| $V_X$ | newly added vertices |
| $E_X$ | newly added edges |
| $A_X$ | newly added agents |

## References


1. R. K. **Ahuja**, T. L. **Magnanti**, and J. B. **Orlin**. *Network flows: theory, algorithms, and applications.* Prentice Hall, 1993, ISBN 978-0136175490.
2. S. A. **Cook**. *The Complexity of Theorem Proving Procedures.* Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC 1971), pp. 151-158, ACM Press, 1971.
3. T. H. **Cormen**, C. E. **Leiserson**, R. L. **Rivest**, and C. **Stein**. *Introduction to Algorithms (Second edition)*. MIT Press and McGraw-Hill, 2001, ISBN 0-262-03293-7.
4. J. C. **Culberson**. *Sokoban is PSPACE-complete.* Technical Report TR 97-02, Department of Computing Science, University of Alberta, 1997, http://webdocs.cs.ualberta.ca/~joe/Preprints/Sokoban/ index.html [accessed April 2010].
5. J. D. **Dixon** and B. **Mortimer**. *Permutation Groups*. in Graduate Texts in Mathematics, Volume 163, Springer, 1996, ISBN 978-0-387-94599-6.
6. S. **Even**, A. **Itai**, A. **Shamir**. *On the Complexity of Timetable and Multicommodity Flow Problems.* SIAM Journal on Computing, Volume 5 (4), pp. 691-703, Society for Industrial and Applied Mathematics, 1976.

7. G. W. **Flake**, E. B. **Baum**. *Rush Hour is PSPACE-complete, or "Why you should generously tip parking lot attendants".* Theoretical Computer Science, Volume 270(1-2), pp. 895-911 Elsevier, 2002.
8. M. R. **Garey** and D. S. **Johnson**. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., 1979, ISBN: 978-0716710455.
9. R. A. **Hearn** and E. D. **Demaine**. *PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation.* Theoretical Computer Science, Volume 343(1-2), pp. 72-96, Elsevier, 2005.
10. E. **Hordern**. *Sliding Piece Puzzles.* Oxford University Press, 1986, ISBN: 978-0198532040.
11. M. **Ivanová**, P. **Surynek**. *Adversarial Cooperative Path-Finding: A First View.* The 27th AAAI Conference on Artificial Intelligence (AAAI 2013), Bellevue, WA, USA, late breaking track, technical report, AAAI Press, 2013
12. P. **Jackson**, D. **Sheridan**. *Clause Form Conversions for Propositional Circuits.* Theory and Applications of Satisfiability Testing, 7th International Conference (SAT 2004), Revised Selected Papers, pp. 183–198, Lecture Notes in Computer Science 3542, Springer 2005.
13. D. **Kornhauser**, G. L. **Miller**, and P. G. **Spirakis**. *Coordinating Pebble Motion on Graphs, the Diameter of Permutation Groups, and Applications*. Proceedings of the 25th Annual Symposium on Foundations of Computer Science (FOCS 1984), pp. 241-250, IEEE Press, 1984.
14. C. H. **Papadimitriou**, P. **Raghavan**, M. **Sudan**, and H. **Tamaki**. Motion Planning on a Graph. Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS 1994), pp. 511-520, IEEE Press, 1994.
15. D. **Ratner** and M. K. **Warmuth**. *Finding a Shortest Solution for the N×N Extension of the 15-PUZZLE Is Intractable*. Proceedings of the 5th National Conference on Artificial Intelligence (AAAI 1986), pp. 168-172, Morgan Kaufmann Publishers, 1986.
16. D. **Ratner** and M. K. **Warmuth**. *NxN Puzzle and Related Relocation Problems.* Journal of Symbolic Computation, Volume 10 (2), pp. 111-138, Elsevier, 1990.
17. M. R. K. **Ryan**. *Graph Decomlocation for Efficient Cooperative path-finding*. Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007), pp. 2003-2008, IJCAI Conference, 2007.
18. M. R. K. **Ryan**. *Exploiting Subgraph Structure in Cooperative path-finding*. Journal of Artificial Intelligence Research (JAIR), Volume 31, pp. 497-542, AAAI Press, 2008.
19. P. E. **Schupp** and R. C. **Lyndon**. *Combinatorial group theory*. Springer, 2001, ISBN 978-3-540-41158-1.
20. D. **Silver**. *Cooperative Pathfinding.* Proceedings of the 1st Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2005), pp. 117-122, AAAI Press.
21. T. **Standley**. *Finding Optimal Solutions to Cooperative Pathfinding Problems.* Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI 2010), pp. 173-178, AAAI Press, 2010.
22. P. **Surynek**. *A Novel Approach to Path-finding for Multiple Agents in Bi-connected Graphs*. Proceedings of the 2009 IEEE International Conference on Agentics and Automation (ICRA 2009), pp. 3613-3619, IEEE Press, 2009.
23. P. **Surynek**. *Towards Shorter Solutions for Problems of Path-finding for Multiple Agents in* θ*-like Environments*. Proceedings of the 22nd International FLAIRS Conference (FLAIRS 2009), pp. 207-212, AAAI Press, 2009.
24. P. **Surynek**. *Making Solutions of Cooperative path-finding Problems Shorter Using Weak Translocations and Critical Path Parallelism*. Proceedings of the 2009 International Symposium on Combinatorial Search (SoCS 2009), University of Southern California, 2009, http://www.search-conference.org/index.php/Main/SOCS09 [accessed July 2009].

25. P. **Surynek**. *An Application of Pebble Motion on Graphs to Abstract Cooperative path-finding.* Proceedings of the 21st International Conference on Tools with Artificial Intelligence (ICTAI 2009), pp. 151-158, IEEE Press, 2009.

26. P. **Surynek**. *An Optimization Variant of Cooperative path-finding is Intractable.* Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI 2010), pp. 1261-1263, AAAI Press, 2010.

27. P. **Surynek**. *Towards Optimal Cooperative Path Planning in Hard Setups through Satisfiability Solving*. Proceedings of 12th Pacific Rim International Conference on Artificial Intelligence (PRICAI 2012), pp. 564-576, LNCS 7458, Springer, 2012.

28. P. **Surynek**. *Solving Abstract Cooperative Path-Finding in Densely Populated Environments*. Computational Intelligence, Volume 30, Issue 2, pp. 402-450, Wiley, 2014.

29. R. E. **Tarjan**. *Depth-First Search and Linear Graph Algorithms.* SIAM Journal on Computing, Volume 1 (2), pp. 146-160, Society for Industrial and Applied Mathematics, 1972.

30. K. C. **Wang** and A. **Botea**. *Tractable Cooperative path-finding on Grid Maps*. Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009), pp. 1870-1875, IJCAI Conference, 2009.

31. K. C. **Wang**. *Bridging the Gap between Centralised and Decentralised Multi-Agent Pathfinding.* Proceedings of the 14th Annual AAAI/SIGART Doctoral Consortium (AAAI-DC 2009), pp. 23-24, AAAI Press, 2009.

32. K. C. **Wang** and A. **Botea**. *Fast and Memory-Efficient Multi-Agent Pathfinding.* Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling (ICAPS 2008), Australia, pp. 380-387, AAAI Press, 2008, ISBN 978-1-57735-386-7.

33. K. C. **Wang** and A. **Botea**. *Scalable Multi-Agent Pathfinding on Grid Maps with Tractability and Completeness Guarantees*. Proceedings of the European Conference on Artificial Intelligence (ECAI 2010), IOS Press, 2010.

34. D. B. **West**. *Introduction to Graph Theory.* Prentice Hall, 2000, ISBN: 978-0130144003.

35. J. **Westbrook**, R. E. **Tarjan**. *Maintaining bridge-connected and biconnected components on-line*. Algorithmica, Volume 7, Number 5&6, pp. 433–464, Springer, 1992.

36. B. de **Wilde**, A. ter **Mors**, C. **Witteveen**. *Push and rotate: cooperative multi-agent path planning*. Proceedings of International conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2013), pp. 87-94, IFAAMAS, 2013.

37. R. M. **Wilson**. *Graph Puzzles, Homotopy, and the Alternating Group.* Journal of Combinatorial Theory, Ser. B 16, pp. 86-96, Elsevier, 1974.