# Abstract Path Planning for Multiple Robots: A Theoretical Study

**Pavel Surynek**

*Charles University in 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.* An abstraction of the problem of multi-robot path planning is introduced in this paper. The basic task is to determine spatial-temporal plan for each robot of a group of robots where each robot is given its initial position in the environment and it needs to go to the given goal position. Robots must avoid obstacles and must not collide with each other. The abstraction adopted in this work models the environment within that robots are moving as an undirected graph. Robots are placed in vertices of the graph; at most one robot is placed in each vertex and at most one vertex remains unoccupied .The move is allowed into the unoccupied vertex or into the vertex being vacated by an allowed move supposed that no other robot is entering the same target vertex. The relation of multi-robot path planning to the problem of pebble motion on a graph (which the most widely known representative is 15-puzzle) is discussed.

The optimization variant of the abstract multi-robot path planning is particularly studied. The task is to find a solution of the makespan as small as possible in the optimization variant. The main contribution of the paper is the proof of the *NP*-completeness of the decision version of the optimization variant of multi-robot path planning. The reduction of Boolean satisfiability to multi-robot path planning is used in the proof.

*Keywords*: multi-robot, path planning, multi-agent, coordination, sliding puzzle, ($n^2$-1)-puzzle, 15-puzzle, domain dependent planning, complexity, *NP*-completeness

## 1. Introduction and Motivation

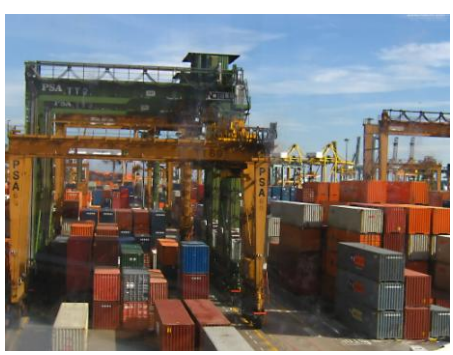


**Figure 1.** *An illustration of shipping container rearranging.* This problem can be formulated as path planning for multiple robots where robots are represented by containers.

This paper is devoted to a problem of *path planning for multiple robots* [14, 15, 18]. Consider a group of mobile robots that are moving in some environment (for example in the 2-dimensional plane with obstacles). Each robot of the group is given an initial and a goal position in the environment. The question of interest is how to determine a sequence of motions for each robot of the group such that all the robots reach their goal positions supposed they started from the given initial ones by following this sequence. Physical limitations must be respected by robots: robots must **not collide** with each other and they must **avoid obstacles** in the environ-

ment during their movements.

The problem of multi-robot path planning is **motivated** by many practical tasks. Various problems of navigating a group of mobile robots can be formulated as multi-robot path planning. However, the primary motivations for the problem are tasks of moving certain entities within an environment with a limited free space. Hence, the formulation of the problem is not restricted to the case where robots are actually represented by mobile robots. Such real-life examples include rearranging of shipping containers in warehouses (a robot is represented by a shipping container - see Figure 1) or coordination of vehicles in dense traffic (robot = vehicle). Moreover, the reasoning about these rearrangement/coordination tasks should not be limited to physical entities only. A robot may be represented by a virtual entity or by a piece of commodity as well. Thus, many tasks such as planning of data transfer between communication nodes with limited storage capacity (robot = data packet), commodity transportation in the commodity transportation network (robot = 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 the problem of multi-robot path planning.

The **primary aim of this paper** is to study theoretical aspects of the problem, namely computational complexity of the abstract formulation of the problem. The abstraction consists in modeling the environment, where robots are moving, as an undirected graph. Vertices of the graph represent locations within the environment and edges represent possibility of going from one location to the neighboring location. Robots are placed in vertices of the graph and they are allowed to move into neighboring vertex if it is **unoccupied or currently being vacated** by an allowed move supposed that no other robot is entering the same target vertex.

There is variety of ways how to create an abstract instance of a given specific real-life multi-robot path planning instance. It is necessary to make decisions how to sample locations in the original environment in order to make the abstract instance to model the real-life situation as precisely as needed. Nevertheless, these issues are out of scope of this work. Here, the principal question about the complexity of solving instances of the abstract formulation itself is studied.

The main contribution of this paper is the proof of ***NP*-completeness** [6] of the optimization variant of multi-robot path planning. This result has been already previewed in the short conference paper [22]. However, only the sketch of the proof fitted into the short paper. This paper should be regarded as the full version of the proof where all the details are rigorously treated and illustrated. The technical report from that this paper is coming out is available as [23].

In the context of multi-robot path planning, works on problems of motion planning over graphs must be mentioned [11, 12, 13, 32] since they are closely related. Namely, works on so called problems of *pebble motion on graphs* (which the most widely known representative is the *15-puzzle*) [11, 13, 32] represents almost the same problem as multi-robot path planning. The difference lays in the condition on the dynamicity in the problem - moves are allowed into **currently unoccupied** vertices only (and no other pebble is

entering the same target vertex) in the problem of pebble motion on a graph. Many theoretical results are known for pebble motion on a graph - it is known that the problem can be solved in polynomial time (in $\mathcal{O}(|V|^3)$ for $G = (V, E)$ modeling the environment) with solution consisting of polynomial number of moves (again $\mathcal{O}(|V|^3)$ moves) [11, 32]. Moreover, it is known that the decision version of the optimization variant of pebble motion on a graph is *NP*-complete [13] (this has been actually shown for generalized variant of the 15-puzzle). Hence, a natural question how situation will change in the case of multi-robot path planning arose and this paper gives the answer.

The **secondary aim of this paper** is to clarify terminology, since many papers actually use the term multi-robot path planning for pebble motion on a graph, which evokes an impression that these problems are different. This aspect is discussed in details along with definitions of problems.

The organization of the paper is as follows: the formal definition of the problem of pebble motion on a graph is recalled and the definition of the abstraction of multi-robot path planning is given in **Section 3**. Some basic properties of problems and their correspondence is discussed in this section too. **Section 3** represents the core of the paper - several techniques for polynomial transformation of Boolean satisfiability to the problem of multi-robot path planning are described in this section. The last section - **Section 4** - is devoted to related works and to concluding remarks.

## 2. Pebble Motion on a Graph and Multi-robot Path Planning

Problems of *pebble motion on a graph* and *multi-robot path planning* are formally defined in this section. A relation of both problems is discussed and their basic theoretical properties are summarized.

The primary problem studied in this paper is the problem of multi-robot path planning. It is almost the same problem as the problem of pebble motion on a graph [11, 32]. The problem of pebble motion on a graph has been already studied in the literature and many theoretical results are known for this problem. The problem of multi-robot path planning represents a relaxation of pebble motion with respect to the dynamicity.

Consider an environment in which a group of mobile robots is moving. The robots are all identical (that is, they are all of the same size and have the same moving abilities). Each robot starts at a given initial position and it needs to reach a given goal position. The problem being addressed consists in finding a spatial-temporal path for each robot so that it can reach its goal by following this path. Robots must **not collide** with each other and they must **avoid obstacles** in the environment.

A relatively strong abstraction is adopted in this work. The environment with obstacles within that the robots are moving is modeled as an **undirected graph**. The vertices of this graph represent locations in the environment and the edges model an unblocked way from one location to the neighboring location. The time is discrete in this abstraction; it is an infinite linearly ordered set isomorphic to the set of natural numbers where each element is called a *time step* (time steps are numbered starting with 0). At each time

step, each robot is located in a vertex. A motion of a robot is an instantaneous event. That is, if the robot is placed in a vertex at a given time step then the result of the motion is the situation where the robot is placed in the same or in the neighboring vertex at the following time step.

The problem of pebble motion on a graph works with pebbles instead of robots. Moreover, a condition on the allowed motions of pebbles is more restrictive than in the case of robots in multi-robot path planning.

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

The following two definitions formalize a problem of *pebble motion on a graph* (also called a *pebble motion puzzle*, *sliding box puzzle*; special variants are known as the *15-puzzle* and $(n^2-1)$*-puzzle*) [1, 21] and the related problem of *multi-robot path planning* [14, 15 18]. Both problems and their solutions are illustrated in Figure 2.

**Definition 1** ***(problem of pebble motion on a graph).*** Let $G = (V, E)$ be an undirected graph. Next, let $P = \{\bar{p}_1, \bar{p}_2, \ldots, \bar{p}_\mu\}$ where $\mu < |V|$ be a set of pebbles. The graph models an environment in which the pebbles are moving. An ***initial arrangement*** of the pebbles is defined by a uniquely invertible function $S_P^0 : P \longrightarrow V$ (that is, $S_P^0(p) \neq S_P^0(q)$ for every $p, q \in P \in$ with $p \neq q$). A ***goal arrangement*** of the pebbles is defined by another uniquely invertible function $S_P^+ : P \longrightarrow V$ (that is, $S_P^+(p) \neq S_P^+(q)$ for every $p, q \in P$ with $p \neq q$). A problem of ***pebble motion on a graph*** is the task to find a number $\xi$ and a sequence $\mathcal{S}_P = [S_P^0, S_P^1, \ldots, S_P^\xi]$ where $S_P^k : P \longrightarrow V$ is a uniquely invertible function for every $k = 1,2, \ldots, \xi$. Additionally, the following conditions must hold for the sequence $\mathcal{S}_P$:

(i) $S_P^\xi = S_P^+$; that is, all the pebble reaches their destination vertices.

(ii) 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 $k = 1,2, \ldots, \xi - 1$; that is, a pebble can either stay in a vertex or move into the neighboring vertex between each two successive time steps.

(iii) 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 $S_P^k(q) \neq S_P^{k+1}(p)\ \forall q \in P$ such that $q \neq p$; must hold for every $p \in P$ and $k = 1,2, \ldots, \xi - 1$; that is, a pebble can move into an unoccupied neighboring vertex only. This condition together with unique invertibility of functions forming $\mathcal{S}_P$ implies that no two pebbles can enter the same target vertex at the same time step.

The instance of the problem of pebble motion on a graph is formally a quadruple $\Pi = (G, P, S_P^0, S_P^+)$. Sometimes, the solution of the problem $\Pi$ will be denoted as $\mathcal{S}_P(\Pi) = [S_P^0, S_P^1, \ldots, S_P^\xi]$. □

The **notation** with a stripe 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

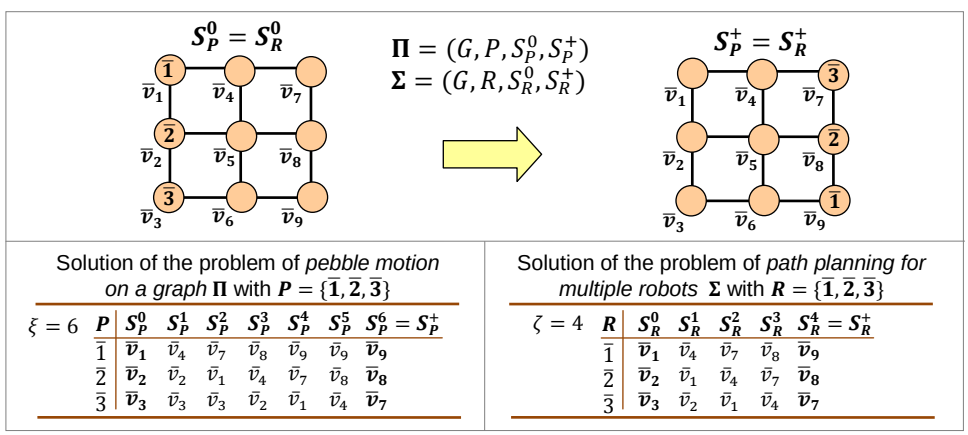
constant parameterized by an index $i \in \mathbb{N}$; the parameterization by an expression will be clear from the context).

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

As it has been mentioned, the term multi-robot path planning has been already used in literature for pebble motion on a graph in fact. In the work titled "*Exploiting Subgraph Structure in Multi-Robot Path Planning*" [15] the dynamicity of the problem is described as follows:

> *"Further, we shall assume that the map is constructed so that collisions only occur when one robot is entering a vertex v at the same time as another robot is occupying, entering or leaving this vertex."*

In other words, a robot can enter a vertex if and only if it is unoccupied at the time of commencing the move and no other robot is entering the same target vertex, which is exactly the definition of the dynamicity in the problem of pebble motion on a graph.




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

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

**Figure 2.** *An illustration of problems of* ***pebble motion on a graph*** *and* ***multi-robot path planning***. Both problems are illustrated on the same graph with the same initial and goal positions. The task is to move pebbles/robots from their initial positions specified by $S_P^0/S_R^0$ to the goal positions specified by $S_P^+/S_R^+$. A solution of the makespan 6 ($\xi = 6$) is shown for the problem of pebble motion on a graph and a solution of the makespan 4 ($\zeta = 4$) is shown for the problem of multi-robot path planning. Notice the differences in parallelism between both solutions – multi-robot path planning allows a higher number of moves to be performed in parallel thanks to weaker requirements on solutions.

An alternative supposedly more reasonable definition of multi-robot path planning is adopted in this work. A problem of multi-robot path planning is a **relaxation** of the problem of pebble motion on a graph. The condition that the target vertex of a pebble/robot must be vacated in the previous time step is relaxed. Thus, the motion of a robot entering

the target vertex, that is simultaneously vacated by another robot and no other robot is trying to enter the same target vertex, is allowed in multi-robot path planning. However, there must be some leading robot initiating such chain of moves by moving into an unoccupied vertex (that is, robots can move like a train with the leading robot in front) that is not entered by another robot at the same time step. The problem is formalized in the following definition.

**Definition 2** ***(problem of multi-robot path planning)*.** Again, let $G = (V, E)$ be an undirected graph. Now a set of robots $R = \{\bar{r}_1, \bar{r}_2, \ldots, \bar{r}_\nu\}$ where $\nu < |V|$ is given instead of the set of pebbles. Similarly, the graph models an environment in which the robots are moving. The ***initial arrangement*** of the robots is defined by a uniquely invertible function $S_R^0: R \longrightarrow V$ (that is, $S_R^0(r) \neq S_R^0(s)$ for every $r, s \in R$ with $r \neq s$). The ***goal arrangement*** of the robots is defined by another uniquely invertible function $S_R^+: R \longrightarrow V$ (that is, $S_R^+(r) \neq S_R^+(s)$ for every $r, s \in R$ with $r \neq s$). A problem of ***multi-robot path planning*** is the task to find a number $\zeta$ and a sequence $\mathcal{S}_R = [S_R^0, S_R^1, \ldots, S_R^\zeta]$ where $S_R^k: R \longrightarrow V$ is a uniquely invertible function for every $k = 1,2, \ldots, \zeta$. The following conditions must hold for the sequence $\mathcal{S}_R$:

(i) $S_R^\zeta = S_R^+$; that is, all the robots reaches their destination vertices.

(ii) Either $S_R^k(r) = S_P^{k+1}(r)$ or $\{S_R^k(r), S_R^{k+1}(r)\} \in E$ for every $r \in R$ and $k = 1,2, \ldots, \zeta - 1$; that is, a robot can either stay in a vertex or move to the neighboring vertex at each time step.

(iii) If $S_R^k(r) \neq S_R^{k+1}(r)$ (that is, the robot $r$ moves between time steps $k$ and $k+1$) and $S_R^k(s) \neq S_R^{k+1}(r)\ \forall s \in R$ such that $s \neq r$ (that is, no other robot $s$ occupies the target vertex at time step $k$), then the move of $r$ at the time step $k$ is called to be ***allowed*** (that is, the robot $r$ moves into an unoccupied neighboring vertex – a ***leading*** robot). If $S_R^k(r) \neq S_R^{k+1}(r)$ and there is $s \in R$ such that $s \neq r \wedge S_R^k(s) = S_R^{k+1}(r) \wedge S_R^k(s) \neq S_R^{k+1}(s)$ (that is, the robot $r$ moves into a vertex that is being left by the robot $s$) and the move of $s$ at the time step $k$ is allowed, then the move of $r$ at the time step $k$ is also ***allowed***. All the moves of robots at all the time steps **must be allowed**. Analogically, this condition together with the requirement on unique invertibility of functions forming $\mathcal{S}_R$ implies that no two robots can enter the same target vertex at the same time step.

The instance of the problem of multi-robot path planning is formally a quadruple $\Sigma = (G, R, S_R^0, S_R^+)$. The solution of the problem $\Sigma$ will be sometimes denoted as $\mathcal{S}_R(\Sigma) = [S_R^0, S_R^1, \ldots, S_R^\zeta]$. □

The numbers $\xi$ and $\zeta$ are called ***makespan*** of the solution of pebble motion on a graph and multi-robot path planning respectively. The makespan need to be distinguished from the ***size*** of the solution, which is the total number of moves performed by pebbles/robots. The makespan is typically less than the size of the solution. In case of the

pebble motion on a graph with just single unoccupied vertex, the makespan and the size of the solution are the same.

### 2.2. *Known Properties of Motion Problems and Open Question*

Several basic properties of solutions of problems of pebble motion on graphs and multi-robot path planning are summarized in this section.

Notice that a solution of an instance of the problem of pebble motion on a graph as well as a solution of an instance of the problem of multi-robot path planning allows a pebble/robot to **stay** in a vertex for more than a single time step. It is also possible that a pebble/robot **visits** the same vertex **several times** within the solution. Hence, a sequence of moves for a single pebble/robot does not necessarily form a simple path in the given graph.

Notice further that both problems intrinsically allow parallel movements of pebbles/robots. That is, more than one pebble/robot can perform a move at a single time step. However, multi-robot path planning allows higher motion parallelism due to its weaker requirements on robot movements (the target vertex is required to be unoccupied only for the leading robot in the previous time step – see Figure 2). More than one unoccupied vertex is necessary to obtain parallelism in the problem of pebble motion on a graph. On the other hand, it is sufficient to have single unoccupied vertex to obtain parallelism within the solution of an instance of the multi-robot path planning problem (consider for example robots moving around a cycle).

**Proposition 1** ***(problem correspondence).*** Let $\Pi = (G, P, S_P^0, S_P^+)$ be an instance of the problem of pebble motion on a graph and let $\mathcal{S}_P(\Pi) = [S_P^0, S_P^1, \dots, S_P^\xi]$ be its solution. Then $\mathcal{S}_R(\Sigma) = \mathcal{S}_P(\Pi)$ is a solution of an instance of the problem of path planning for multiple robots $\Sigma = (G, P, S_P^0, S_P^+)$. In other words, the instance of the multi-robot path planning problem consists of the same graph, the set of robots is represented by the set of pebbles, and the initial/goal positions of robots are the same as in the case of pebbles. Then the solution of the instance of the pebble motion problem can be used as a solution of the corresponding instance of the multi-robot path planning problem. ■

**Proof.** The proof of the statement is straightforward using Definition 1 and Definition 2. The condition on sequence of moves required by Definition 2 needs to be checked for $\mathcal{S}_P(\Pi)$. Conditions (i) and (ii) of Definition 2 are trivially satisfied. Condition (iii) is also satisfied since it holds that if $S_P^k(p) \neq S_P^{k+1}(p)$ then $S_P^k(q) \neq S_P^{k+1}(p)\ \forall q \in P$ such that $q \neq p$ is true for every $p \in P$ and $k = 1,2, \dots, \xi - 1$. In other words, all the moves within $\mathcal{S}_P(\Pi)$ are **allowed**. ■

There is a variety of modifications of the defined problems. 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 re-

quirement makes the problem of pebble motion on a graph **intractable**. It is shown in [13] that the optimization variant of a special case of the problem of pebble motion on a graph is ***NP*-hard [6]**. The restriction forming the special case adopted in [13] works with a graph that can be embedded in plane as a square grid and there is a single unoccupied vertex - this case is known as $N \times N$ *puzzle* (also known as $(n^2 - 1)$-*puzzle*). Hence, the general optimization variant of the problem of pebble motion on a graph is also $NP$-hard.

A restriction of both types of problems on *bi-connected graphs* [30, 31] (for the precise definitions see Section 0) represents important subclass with respect to the existence of a solution. Hence, it is a reasonable question what is the complexity of these classes of problems. Since the grid graph forming the mentioned $N \times N$ puzzle is **bi-connected** as well**,** the immediate answer is that the optimization variant of the problem of pebble motion on a bi-connected graph with a single unoccupied vertex is again $NP$-hard.

However, it is not simply possible to make any similar statement about the complexity of the optimization variant of multi-robot path planning based on the above facts. The situation there is complicated by the inherent parallelism, which can reduce the makespan of the solution significantly. Constructions used for the $N \times N$ puzzle in [13] thus no longer apply for this case.

Observe further that difficult cases of the problem of pebble motion on a graph have a single unoccupied vertex. This fact may raise a question how the situation is changed when there are **more** than one **unoccupied vertices**. The intuition prompts that more unoccupied vertices may simplify the problem. Unfortunately, it is not the case. The pebble motion problem 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 [13] can be used to add as many unoccupied vertices as needed (the resulting graph may be disconnected).

**Without** the requirement on the **optimality** of the makespan of solutions the situation is much easier; the problem of pebble motion on a graph is in the ***P* class** as it is shown in [11, 32]. Due to Proposition 1, the problem of path planning for multiple robots is also in the $P$ class. In fact, this result concerns the decision version of the problem, which is the question whether there exists a solution for a given instance. Fortunately, it has been shown in [11] that a solution of the size of $\mathcal{O}(|V|^3)$ can be generated for any solvable $\Pi = (G = (V, E), P, S_P^0, S_P^+)$. Hence, it provides a polynomial upper bound on size of the content of the oracle to guess in non-deterministic model [8]. Thus, it is possible to conclude that decision version of optimization variant of pebble motion on a graph is an *NP*-complete problem. By the decision version here, it is meant the yes/no question whether there is a solution of $\Pi$ of the makespan smaller than the given bound.

Thus, it seems that pebble motion on a graph and multi-robot path planning problems have been already resolved except the case of the complexity of the optimization variant of multi-robot path planning. However there is another issue worth studying. Constructions proving the membership of the problem of pebble motion on a graph into the $P$ class used in [11, 32] generate solutions that are too long for practical use. As the makespan of the solution is of great importance in practice, this fact makes these methods unsuitable

when dealing with some real life motion problem abstracted as the problem of pebble motion on a graph or multi-robot path planning [19, 20, 21]. Hence, alternative solving methods for the problem of multi-robot path planning are of interest [18, 19, 20, 21, 24].

## 3. The Intractability of the Optimization Variant of Multi-robot Path Planning

Several **complexity results** for the optimization variant of studied problems will be shown in this section. The main result is that the optimization variant is **intractable** for **multi-robot** path planning. Namely, it is *NP*-complete. The first sketch of the proof of this result has been presented in [22]. A rigorous version of the proof is presented in the following paragraphs. All the details that are missing in [22] are supplied here.

### 3.1. *Reduction Techniques*

A reduction of Boolean satisfiability [1, 6, 10] to the problem of multi-robot path planning will be used to prove the *NP*-completeness of the problem. The problem of Boolean satisfiability exhibits some characteristics that need to be simulated in multi-robot path planning. The first characteristic is so called *Boolean consistency*, which means that all positive and all the negative occurrences of the same variable in the input formula have the same Boolean value respectively. The second characteristic is the fact that all the *clauses* of the Boolean formula in *CNF* [10] need to be satisfied in order to satisfy the formula as the whole. This characteristic will be called *clause satisfaction*. Description of techniques how to simulate Boolean consistency and clause satisfaction in multi-robot path planning is provided in following paragraphs.

First, a technique how to **prevent** robots **from entering** a given vertex at a given set of time steps will be shown. This is the crucial skill used later. The technique works with an arbitrary instance of the problem of path planning for multiple robots. An augmentation of the given instance of the problem can be made so that robots are prevented from entering a selected vertex at 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 robots into the instance. The selection of time steps at that the vertex will not be allowed to entering by the original robots is modeled by an appropriate setting of the initial and goal positions of the newly added robots. The whole construction is formalized in the following proposition and its proof.

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

(a) Let $\Sigma = (G = (V, E), R, S_R^0, S_R^+)$ be an instance of multi-robot path planning and let $v \in V$ with $S_R^0(r) \neq v \; \forall r \in R$ be a so called *locked vertex*.

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

Then there exists an instance of the problem of multi-robot path planning $\Sigma' = (G' = (V', E'), R', S_{R'}^0, S_{R'}^+)$ such that $\Sigma'|_V = \Sigma$ and it **never happens** that a robot $r \in R$ enters

the vertex $v$ at any time step $t \in T$ within any **optimal** solution $\mathcal{S}^*_{R'}(\Sigma')$ (entering the vertex $v$ at the time step $t$ means that a robot is located in $v$ at time step $t$). ■

The notation $\Sigma'|_V$ stands for a *restriction* of the multi-robot path planning problem on the set of vertices $V$. That is, if $\Sigma' = (G' = (V', E'), R', S^0_{R'}, S^+_{R'})$ and $V \subseteq V'$, then $\Sigma'|_V = (G'|_V, R'|_V, S^0_{R'|_V}, S^+_{R'|_V})$ where $G'|_V = (V, E' \cap \{\{u, v\} | u, v \in V\})$, $R'|_V = \{r \in R' | S^0_{R'}(r) \in V \wedge S^+_{R'}(r) \in V\}$, $S^0_{R'}|_V : R'|_V \rightarrow V$ with $S^0_{R'}|_V(r) = S^0_{R'}(r)\ \forall r \in R'|_V$, and $S^+_{R'}|_V : R'|_V \rightarrow V$ with $S^+_{R'}|_V(r) = S^+_{R'}(r)\ \forall r \in R'|_V$. In other words, each component of the description of the instance is naturally restricted on the smaller set of vertices.

**Proof.** Let $\zeta^*$ be the makespan of any optimal solution of the multi-robot path planning instance $\Sigma$ (the number $\zeta^*$ is however difficult to compute as it is shown later).

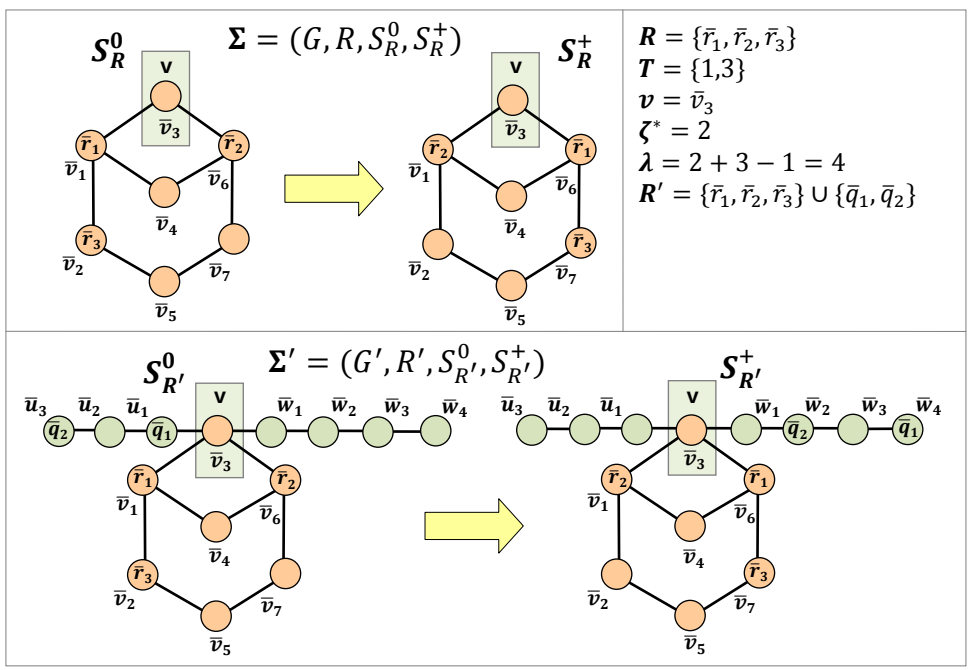



**Figure 3**. *An illustration* ***vertex locking augmentation*** *in instance of multi-robot path planning problem.* Robots $\bar{r}_1$, $\bar{r}_2$, and $\bar{r}_3$ are needed to be prevented from entering the vertex $\bar{v}_3$ at time steps 1 and 3 in any optimal solution. The original instance $\boldsymbol{\Sigma}$ with a set of robots $R = \{\bar{r}_1, \bar{r}_2, \bar{r}_3\}$ is shown in the upper part of the figure. The makespan of any optimal solution of $\Sigma$ is $\zeta^* = 2$. The augmented instance $\boldsymbol{\Sigma}'$ is in the lower path 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 robots $\bar{q}_1$ and $\bar{q}_2$ were added. The makespan of any optimal solution of the augmented problem is $\lambda + t_1 = \zeta^* + t_n - t_1 + t_1 = \zeta^* + t_n = 2 + 3 = 5$.

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 $V_X = \{\bar{u}_{t_n}, \bar{u}_{t_n - 1}, \ldots, \bar{u}_1, \bar{w}_1, \bar{w}_2, \ldots, \bar{w}_\lambda\}$ 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 robots are initially placed in new vertices $\bar{u}_t$ for $t \in T$ with $t \geq 1$ or a new robot is placed in $v$ if $0 \in T$. Then the newly added robots are forced to move straight ahead into the vertices $\bar{w}_1, \bar{w}_2, \dots, \bar{w}_\lambda$ through the vertex $v$. Making robots to move in this way is imposed by the condition on the optimality of the solution (otherwise, the robots cannot manage to reach their destinations on time). The motion of new robots through the vertex $v$ makes an obstruction in this vertex exactly at selected time steps given by $T$.

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

At this point, it is necessary to show that it really never happens that a robot $r \in R$ enters the vertex $v$ at any time step $t \in T$ within the optimal solution $\mathcal{S}^*_R(\Sigma')$. Any optimal solution of the multi-robot path planning instance $\Sigma'$ has the makespan of $\lambda + 1$. Moreover, any solution $\mathcal{S}^*_{R'}(\Sigma') = [S^0_{R'}, S^1_{R'}, \dots, S^\lambda_{R'}]$ of the optimal makespan of the instance $\Sigma'$, must satisfy that $S^0_{R'}(\bar{q}_i) = \bar{u}_{t_i}$, $S^1_{R'}(\bar{q}_i) = \bar{u}_{t_i-1}$, $S^2_{R'}(\bar{q}_i) = \bar{u}_{t_i-2}, \dots$, $S^{t_i-1}_{R'}(\bar{q}_i) = \bar{u}_1$, $S^{t_i}_{R'}(\bar{q}_i) = v$, $S^{t_i+1}_{R'}(\bar{q}_i) = \bar{w}_1$, $S^{t_i+2}_{R'}(\bar{q}_i) = \bar{w}_2, \dots, S^\lambda_{R'}(\bar{q}_i) = \bar{w}_{\lambda+t_1-t_i} = S^+_{R'}(\bar{q}_i)$ for $i = 1,2, \dots, n$. This is ensured by the fact that the shortest path from $S^0_{R'}(\bar{q}_i)$ to $S^+_{R'}(\bar{q}_i)$ in $G'$ has the length $\lambda + t_1$ 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 makespan exists.

However, it remains to show that the original robots from $R$ manage to reach their destinations within the makespan of $\lambda + t_1$. This claim flows from the equality $\lambda = \zeta^* + t_n - t_1$, that is at least for $\zeta^*$ time steps the vertex $v$ is not obstructed by any motion of newly added robots supposed 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 (notice that no of the original robots need to occupy $v$ at the beginning). Thus, any optimal solution of the original instance can be simulated in the augmented instance while movements of original robots are stopped at time steps when $v$ 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 robot can enter $v$ at any time step $t \in T$. ■

The situation from Lemma 1 is illustrated in Figure 3. Notice, 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 $\mathcal{S}_R(\Sigma)$ of the instance $\Sigma = (G = (V, E), R, S_R^0, S_R^+)$ of the makespan $\zeta$ where $t_n \leq \zeta$.

(d) Let the locked vertex $v \in V$ is entered by a robot within $\mathcal{S}_R(\Sigma)$ at time steps $E = \{e_1, e_2, \ldots, e_m\}$ where $e_i \in \mathbb{N}_0$ for $i = 1,2, \ldots, n$ and $e_1 < e_2 < \cdots < e_m$ and it holds that $E \cap T = \emptyset$.

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

The makespan is **at least** $\zeta$ since the newly added robots must go along the newly added path towards its end which cannot be carried out in any 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 $t_1, t_2, \ldots, t_n$ by the newly added robots that do not interfere with time steps at which the vertex $v$ is entered within the solution $\mathcal{S}_R(\Sigma)$ by the original robots (this is due to $E \cap T = \emptyset$). Altogether, the makespan of any optimal solution of the 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 the selected subset of vertices $W$ is to be locked at selected time steps, will be actually used in further reasoning. To be more precise, at least one vertex in $W$ is required not to be occupied by a robot from the original set of robots at any time step $t \in T$. An extension analogical 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.

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

(aa) $\Sigma = (G = (V, E), R, S_R^0, S_R^+)$ is an instance of multi-robot path planning and $W \subseteq V$ with $S_R^0(r) \notin W \ \forall r \in R$ be a 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 multi-robot path planning $\Sigma' = (G' = (V', E'), R', S^0_{R'}, S^+_{R'})$ such that $\Sigma'|_V = \Sigma$ and it **never happens** that all the vertices of the set $W$ are occupied by robots from the set $R$ at any time step $t \in T$ within any **optimal** solution $\mathcal{S}_{R'}(\Sigma')$ (that is, at least one vertex from $W$ is not occupied by a robot from $R$ at any time step $t \in T$). ■

**Proof.** The instance Σ is augmented in a way that a new robot 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 the multi-robot path planning instance 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 $E_X = \{\{\bar{u}_{t_n}, \bar{u}_{t_n-1}\}, \{\bar{u}_{t_n-1}, \bar{u}_{t_n-2}\}, \dots, \{\bar{u}_2, \bar{u}_1\}, \{\bar{w}_1, \bar{w}_2\}, \{\bar{w}_2, \bar{w}_3\}, \dots, \{\bar{w}_{\lambda-1}, \bar{w}_\lambda\}\} \cup \{\{\bar{u}_1, w\} | w \in W\} \cup \{\{w, \bar{w}_1\} | w \in W\}$ is added to the graph with the extended set of vertices. Thus, the augmented graph is $G' = (V' = V \cup V_X, E' = E \cup E_X)$.

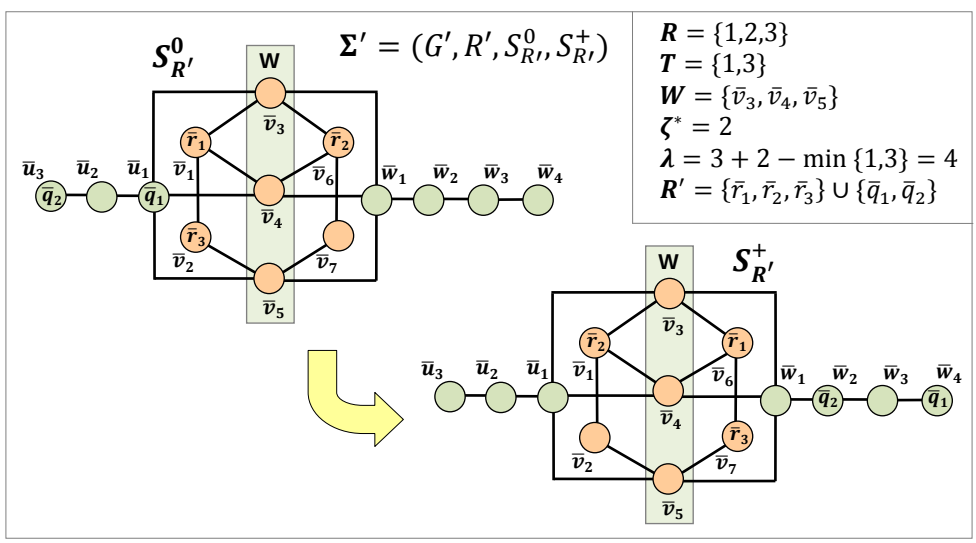



**Figure 4.** An illustration of the vertex set locking augmentation in an instance of a multi-robot path planning problem. At least one vertex of the set $\boldsymbol{W} = \{\bar{v}_3, \bar{v}_4, \bar{v}_5\}$ must not be occupied by any of the original robots $\bar{r}_1$, $\bar{r}_2$, and $\bar{r}_3$ at time steps 1 and 3 in any optimal solution. The original instance Σ with the set of robots $\boldsymbol{R} = \{\bar{r}_1, \bar{r}_2, \bar{r}_3\}$ 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 robots $\bar{q}_1$ and $\bar{q}_2$. The makespan of any optimal solution of the augmented instance $\boldsymbol{\Sigma'}$ is $\lambda + 1 = t_n + \zeta^* = 3 + 2 = 5$.

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

The makespan of any optimal solution of $\Sigma'$ is **at least** $\lambda + t_1$ since the shortest path from $S^0_{R'}(\bar{q}_i)$ to $S^+_{R'}(\bar{q}_i)$ in $G'$ has the length of $\lambda + t_1$ for any $i = 1,2,\ldots,n$. On the other hand, since $\lambda = \zeta^* + t_n - t_1$, no vertex of $W$ is occupied by any new robot at least for $\zeta^*$ time steps supposed the new robots are moving straight towards their destinations. Together with the fact that in any optimal solution of the original instance $\Sigma$ it is sufficient to occupy $W$ for at most $\zeta^*$ time steps, the makespan of any optimal solution of $\Sigma'$ 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 be easily extended for locking **multiple sets** of locked vertices while for each locked set a different lock time steps may be used.

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

(cc) There exists a solution $\mathcal{S}_R(\Sigma)$ of the instance $\Sigma = (G = (V, E), R, S^0_R, S^+_R)$ of the ma 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 $\mathcal{S}_R(\Sigma)$ except time steps $E = \{e_1, e_2, \ldots, e_m\}$ with $e_i \in \mathbb{N}_0$ for $i = 1,2,\ldots,n$ and $e_1 < e_2 < \cdots < e_m$ and it holds that $E \cap T = \emptyset$.

Then there exists an instance $\Sigma' = (G' = (V', E'), R', S^0_{R'}, S^+_{R'})$ such that $\Sigma'|_V = \Sigma$ and it **never happens** that all the vertices of $W$ are occupied by the original robots from the set $R$ at any time step $t \in T$ within any **optimal** solution $\mathcal{S}^*_{R'}(\Sigma')$; 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 Corollary 1. The difference is that the parameter $\lambda$ is now set to $\zeta - t_1$. The construction then ensures that the makespan any optimal solution of $\Sigma'$ is $\zeta$.

The makespan of any optimal solution is **at least** $\zeta$ since the newly added robots must go to the end of the newly added path. On the other hand, all the vertices of the set $W$ needs to be occupied by the original robots within the solution $\mathcal{S}_R(\Sigma)$ only at time steps $e_1, e_2, \ldots, e_m$ that does not interfere with time steps $t_1, t_2, \ldots, t_n$ (since $E \cap T = \emptyset$) at which the newly added robots need to occupy at least one vertex of $W$ (supposed 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 original robots are allowed to enter the newly added vertices in all the above augmentations. This may help the original robots to reach their destinations **faster** (the newly added vertices may be used as additional "parking place" for robots). If this behavior needs to be ruled out, a slight adaptation of the technique must be used.

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

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

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

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

Let $\zeta^*$ denotes be the makespan of optimal solutions of the multi-robot path planning instance $\Sigma$. In the **first stage**, the graph $G$ is extended exactly as in the previous case. That is, the 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 the set of edges $E_X = \{\{\bar{u}_{t_n}, \bar{u}_{t_n-1}\}, \{\bar{u}_{t_n-1}, \bar{u}_{t_n-2}\}, \dots, \{\bar{u}_2, \bar{u}_1\}, \{\overline{u_1}, v\}, \{v, \bar{w}_1\}$ $\{\bar{w}_1, \bar{w}_2\}, \{\bar{w}_2, \bar{w}_3\}, \dots, \{\bar{w}_{\lambda-1}, \bar{w}_\lambda\}\}$ are added to the graph; that is $G' = (V' = V \cup V_X, E' = E \cup E_X)$. The set of robots is extended with $R_X = \{\bar{q}_1, \bar{q}_2, \dots, \bar{q}_n\}$; that is, $R' = R \cup R_X$ and the initial and goal arrangements of new robots are posed again in the same way: $S^0_{R'}(\bar{q}_i) = \bar{u}_{t_i}$ if $t_i \neq 0$ and $S^0_{R'}(\bar{q}_i) = v$ if $t_i = 0$ for $i = 1,2, \dots, n$; $S^+_{R'}(\bar{q}_i) = \bar{w}_{\lambda+t_1-t_i}$ if $\lambda + t_1 - t_i \geq 1$ and $S^+_{R'}(\bar{q}_i) = v$ 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 $Sol^*(\Sigma')|_V \subseteq Sol(\Sigma)$.

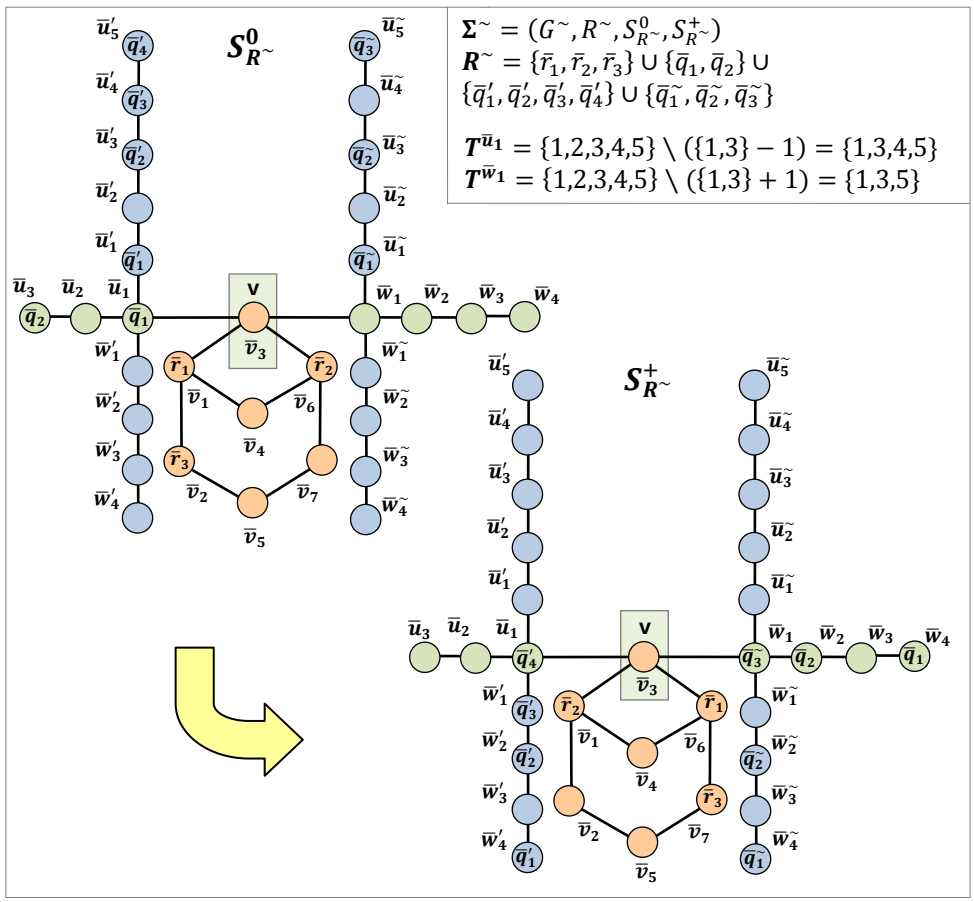



**Figure 5.** *An illustration of* ***two-stage vertex locking*** *in an instance of multi-robot path planning problem.* Robots $\bar{r}_1$, $\bar{r}_2$, and $\bar{r}_3$ are needed to be prevented from entering a vertex $\bar{v}_3$ at time steps 1 and 3. Additionally no vertex added by the augmentation can be entered by the original robots $\bar{r}_1$, $\bar{r}_2$, and $\bar{r}_3$. These requirements are ensured by two stage locking. 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 $\boldsymbol{T}^{\bar{u}_1} = \{1,3,4,5\}$ and $\boldsymbol{T}^{\bar{w}_1} = \{1,3,5\}$ respectively by the same technique. The makespan of any optimal solution of $\boldsymbol{\Sigma}^{\sim}$ is 5 (the same as of $\boldsymbol{\Sigma}'$).

Now, it is necessary to prevent original robots from the set $R$ from entering any of the added vertices $V_X$. Observe that it is sufficient to lock vertices $\bar{u}_1$ and $\bar{w}_1$ to fulfill this requirement since the newly added vertices forms 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 time

steps at which robots from the set $R_X$ go through them in an optimal solution – this will be the **second stage** locking. More precisely, the vertex $\bar{u}_1$ needs to be locked for time steps from the set $T^{\bar{u}_1} = \{1,2,\dots,\lambda + t_1\} \setminus (\{t_1, t_2, \dots, t_n\} - 1)$ (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 for time steps from the set $T^{\bar{w}_1} = \{1,2,\dots,\lambda + t_1\} \setminus (\{t_1, t_2, \dots, t_n\} + 1)$. Notice, 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 will be locked for time steps at which they are not entered within some solution (which is known to be an optimal solution). Hence, Corollary 1 applies for $\Sigma'$, the locked vertex $\bar{u}_1$, and the set of lock time steps $T^{\bar{u}_1}$; that is, optimal makespan is preserved. In other words, the just made vertex locking is **synchronized** with the vertex locking from the first stage. Then Corollary 1 is applied once more for the resulting instance, the locked vertex $\bar{w}_1$, and the set of lock time steps $T^{\bar{w}_1}$. Let $\Sigma^{\sim} = (G^{\sim} = (V^{\sim}, E^{\sim}), R^{\sim}, S^0_{R^{\sim}}, S^+_{R^{\sim}})$ denotes the final instance, then $Sol^*(\Sigma^{\sim})|_V \subseteq Sol(\Sigma)$. ■

The construction from Proposition 2 is illustrated in Figure 5. It is a further augmentation of the instance from Figure 3 in fact.

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 robots are added). Consequently, if an augmented instance is needed to be kept small (with respect to $|V| + |E|$), the numbers $\zeta^*$ and $t_n$ must be small as well.

**Corollary 3** ***(makespan preserving two-stage vertex locking).*** Assume preconditions (a), (b), (c) and (dd. Then there exists an instance $\Sigma^{\sim} = (G^{\sim} = (V^{\sim}, E^{\sim}), R^{\sim}, S^0_{R^{\sim}}, S^+_{R^{\sim}})$ such that $\Sigma^{\sim}|_V = \Sigma$ and it **never happens** that a robot $r \in R$ enters the locked vertex $v$ at any time step $t \in T$ within any **optimal** solution $\mathcal{S}^*_{R^{\sim}}(\Sigma^{\sim})$ and $Sol^*(\Sigma^{\sim})|_V \subseteq Sol(\Sigma)$ (that is, **original** robots 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 certain size of a selected set of vertices at given time steps where the original robots 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 preconditions (aa) and (bb) hold. Then there exists an instance of the problem of multi-robot path planning $\Sigma^{\sim} = (G^{\sim} = (V^{\sim}, E^{\sim}), R^{\sim}, S^0_{R^{\sim}}, S^+_{R^{\sim}})$ such that $\Sigma^{\sim}|_V = \Sigma$ where it **never happens** that all the vertices of $W$ are occupied by the original robots from the set $R$ at any time step $t \in T$ within any **optimal** solution $\mathcal{S}^*_{R^{\sim}}(\Sigma^{\sim})$ and $Sol^*(\Sigma^{\sim})|_V \subseteq Sol^*(\Sigma)$ (that is, **original** robots 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 a successive applications of Corollary 1 to enforce the condition $Sol^*(\Sigma^{\sim})|_V \subseteq Sol^*(\Sigma)$.

Let $\zeta^*$ denotes be the makespan of optimal solutions of the multi-robot path planning instance Σ. The first stage of the augmentation will be done as in the case of of Proposition 2. A set of vertices $V_X = \{\bar{u}_{t_n}, \bar{u}_{t_n-1}, \ldots, \bar{u}_1, \bar{w}_1, \bar{w}_2, \ldots, \bar{w}_\lambda\}$ where $\lambda = \zeta^* + t_n - t_1$ and a set of edges $E_X = \{\{\bar{u}_{t_n}, \bar{u}_{t_n-1}\}, \{\bar{u}_{t_n-1}, \bar{u}_{t_n-2}\}, \ldots, \{\bar{u}_2, \bar{u}_1\}, \{\bar{w}_1, \bar{w}_2\}, \{\bar{w}_2, \bar{w}_3\}, \ldots, \{\bar{w}_{\lambda-1}, \bar{w}_\lambda\}\} \cup \{\{\bar{u}_1, w\} \mid w \in W\} \cup \{\{w, \bar{w}_1\} \mid w \in W\}$ are added to the graph; that is $G' = (V' = V \cup V_X, E' = E \cup E_X)$. The set of robots is extended with a set of new robots $R_X = \{\bar{q}_1, \bar{q}_2, \ldots, \bar{q}_n\}$; that is, $R' = R \cup R_X$ and the initial and goal arrangements of new robots are: $S^0_{R'}(\bar{q}_i) = \bar{u}_{t_i}$ if $t_i \neq 0$ and $S^0_{R'}(\bar{q}_i) = w$ for some $w \in W$ if $t_i = 0$ for $i = 1,2,\ldots,n$; $S^+_{R'}(\bar{q}_i) = \bar{w}_{\lambda+t_1-t_i}$ if $\lambda + t_1 - t_i \geq 1$ and $S^+_{R'}(\bar{q}_i) = w$ for some $w \in W$ if $\lambda + t_1 - t_i = 0$ for $i = 1,2,\ldots,n$.

To prevent original robots from the set $R$ from entering any of the added vertices $V_X$ **second stage** vertex locking must be done. It is sufficient to **lock** vertices $u_1$ and $w_1$ since these two vertices are the only connection points of the original graph with the newly added parts. Vertices $u_1$ and $w_1$ need to be locked for all the time steps except time steps at which robots from the set $R_X$ go through them in an optimal solution. More precisely, the vertex $\bar{u}_1$ needs to be locked for time steps from the set $T^{\bar{u}_1} = \{1,2,\ldots,\lambda + t_1\} \setminus (\{t_1, t_2, \ldots, t_n\} - 1)$ and the vertex $\bar{w}_1$ needs to be locked for time steps from the set $T^{\bar{w}_1} = \{1,2,\ldots,\lambda + t_1\} \setminus (\{t_1, t_2, \ldots, t_n\} + 1)$. Since $\max(T^{\bar{u}_1}) \leq \lambda + t_1$ (as well as $\max(T^{\bar{w}_1}) \leq \lambda + t_1$) and vertex $\bar{u}_1$ is to be locked for time steps at which it is not entered within some (known to be 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, optimal makespan is preserved. In other words, the just made vertex locking is **synchronized** with the vertex locking from the first stage. Then Corollary 1 is applied once more on the resulting instance, the locked vertex $\bar{w}_1$, and the set of lock time steps $T^{\bar{w}_1}$. Let $\Sigma^{\sim} = (G^{\sim} = (V^{\sim}, E^{\sim}), R^{\sim}, S^0_{R^{\sim}}, S^+_{R^{\sim}})$ denotes 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 the original instance.

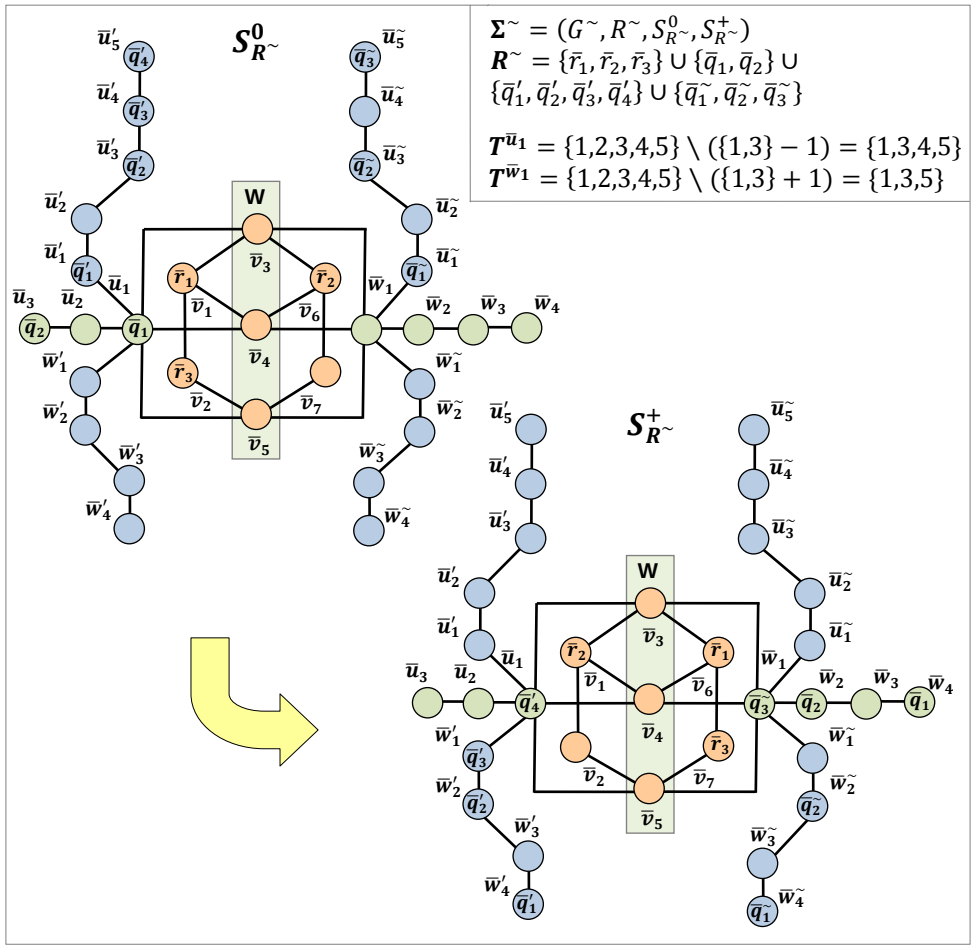



**Figure 6.** An illustration of two-stage vertex set locking in an instance of multi-robot path planning problem. At least one vertex of the set $\boldsymbol{W} = \{\bar{v}_3, \bar{v}_4, \bar{v}_5\}$ must not be occupied by any of the original robots $\bar{r}_1$, $\bar{r}_2$, and $\bar{r}_3$ at time steps 1 and 3. Additionally, no vertex added by the augmentation can be entered by any original of the robots. These requirements are ensured by two stage set locking. 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 vertex locking technique. The makespan of any optimal solution of $\boldsymbol{\Sigma}^{\sim}$ is 5 (the same as of $\boldsymbol{\Sigma}'$ from Figure 4).

**Corollary 4** ***(makespan preserving two-stage set locking).*** Assume that preconditions (aa), (bb), (cc), and (dd) hold. Then there exists an instance $\Sigma^{\sim} = (G^{\sim} = (V^{\sim}, E^{\sim}), R^{\sim}, S^0_{R^{\sim}}, S^+_{R^{\sim}})$ such that $\Sigma^{\sim}|_V = \Sigma$ and it **never happens** that all the vertices of $W$ are occupied by the original robots from the set $R$ at any time step $t \in T$ within any **optimal** solution $\mathcal{S}^*_{R^{\sim}}(\Sigma^{\sim})$; moreover the makespan of any optimal solution of

$\Sigma^{\sim}$ is again $\zeta$ and and $Sol^*(\Sigma^{\sim})|_V \subseteq Sol(\Sigma)$ (that is, **original** robots 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 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. ■

Consider a group of robots that has to go between two parts of the graph that are connected by disjoint branches. The aim is to force all the robots of the group to go through just one of these connecting branches. That is, the behavior when the part of the group of robots goes through one branch and the rest of the group goes through the other branch is unwanted. This behavior will be later used to simulate ***Boolean consistency*** of a valuation of some Boolean literals (positive and negative literals of the same Boolean variable should have **complementary** values). The required behavior of robots will be enforced by a so called ***conjugation technique*** which will be described below.

Let $R = \{\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_n\}$ be a set of robots that are to be conjugated. Formally the **conjugation** means that there is an instance of the problem of path planning for multiple robots $\Xi = (G = (V, E), R', S^0_{R'}, S^+_{R'})$, 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, $|V^{\mathcal{L}}| = |V^{\mathcal{R}}| = |R|$, $R \subseteq R'$, $S^0_{R'}(R) \subseteq V^0$ (image of the set by $S^0_{R'}$ is defined naturally: $S^0_{R'}(R) = \{v | (\exists r \in R) S^0_{R'}(r) = v\}$), $S^+_{R'}(R) \subseteq V^+$, and there exists a time step $t$ such that within any optimal solution $\mathcal{S}^*_{R'}(\Xi) = [S^0_{R'}, S^1_{R'}, \ldots, S^{\zeta}_{R'}]$ either $\{S^t_{R'}(\bar{s}_1), S^t_{R'}(\bar{s}_2), \ldots, S^t_{R'}(\bar{s}_n)\} \subseteq V^{\mathcal{L}}$ or $\{S^t_{R'}(\bar{s}_1), S^t_{R'}(\bar{s}_2), \ldots, S^t_{R'}(\bar{s}_n)\} \subseteq V^{\mathcal{R}}$. To rule out trivial cases of $\Xi$ a requirement that $\{\mathcal{S}^*_{R'}(\Xi) | \mathcal{S}^*_{R'}(\Xi) = [S^0_{R'}, \ldots, S^{\zeta}_{R'}] \wedge \{S^t_{R'}(\bar{s}_1), \ldots, S^t_{R'}(\bar{s}_n)\} \subseteq V^{\mathcal{L}}\} \neq \emptyset$ and $\{\mathcal{S}^*_{R'}(\Xi) | \mathcal{S}^*_{R'}(\Xi) = [S^0_{R'}, \ldots, S^{\zeta}_{R'}] \wedge \{S^t_{R'}(\bar{s}_1), \ldots, S^t_{R'}(\bar{s}_n)\} \subseteq V^{R}\} \neq \emptyset$ should be taken into account. The task is now to build such an instance of the multi-robot path planning problem.

The main idea of the construction is to order the robots from the set $R$ into the queue that starts with an additional robot called a ***leading robot***. There is a branching in the graph and two leading robots prepared. The destination for the leading robots is temporarily closed by the construction from Corollary 1. This prevents the leading robots from **escaping** before fulfilling their task. The destination for the robots from the set $R$ is accessible from both branches **symmetrically**. The leading robots have no other choice than to lead the group of robots to their destination. However, finally the leading robot has to go out of the way. If the group of robots from the set $R$ is split between both branches, then the leading robots inevitably block each other and then there is no chance to reach destinations in time. Hence, the robot must go into one of the branches together (they must conjugate). Below is the formal description of the construction.

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

$V^0 = \{\bar{v}^0_1, \bar{v}^0_2, \ldots, \bar{v}^0_n\}$

(called ***initial*** *vertices*),

$V^{\mathcal{L}} = \{\bar{v}_1^{\mathcal{L}}, \bar{v}_2^{\mathcal{L}}, \dots, \bar{v}_n^{\mathcal{L}}\}$
(called ***left*** *vertices*),
$V^{\mathcal{R}} = \{\bar{v}_n^{\mathcal{R}}, \bar{v}_{n-1}^{\mathcal{R}}, \dots, \bar{v}_1^{\mathcal{R}}\}$
(called ***right*** *vertices*),
$V^{+} = V_{\mathcal{L}}^{+} \cup V_{\mathcal{R}}^{+} \cup V_G^{+} \cup V_A^{+}$
(called ***destination*** *vertices*), with
$V_{\mathcal{L}}^{+} = \{\bar{v}_{+}^{\mathcal{L}}, \bar{v}_{-2}^{\mathcal{L}}, \bar{v}_{-1}^{\mathcal{L}}, \bar{v}_0^{\mathcal{L}}\}$
(called ***left part*** of destination vertices)
$V_{\mathcal{R}}^{+} = \{\bar{v}_{+}^{\mathcal{R}}, \bar{v}_0^{\mathcal{R}}, \bar{v}_{-1}^{\mathcal{R}}, \bar{v}_{-2}^{\mathcal{R}}\}$
(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
$V_A^{+} = \{\bar{v}_{1,1}^{+}, \bar{v}_{1,2}^{+}, \dots, \bar{v}_{1,n}^{+}, \bar{v}_{2,1}^{+}, \bar{v}_{2,2}^{+}, \dots, \bar{v}_{2,n}^{+}, \dots, \dots, \bar{v}_{\vartheta,1}^{+}, \bar{v}_{\vartheta,2}^{+}, \dots, \bar{v}_{\vartheta,n}^{+}\}$
(called ***array part*** of destination vertices),

where $\vartheta$ is a parameter determining the length of a solution; it is required that $\vartheta \geq n + 4$. Notice that $V_A^{+}$ 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:

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

$E^{-} = \{\{\bar{v}_{-2}^{\mathcal{L}}, \bar{v}_{-1}^{\mathcal{L}}\}, \{\bar{v}_{-1}^{\mathcal{L}}, \bar{v}_0^{\mathcal{L}}\}, \{\bar{v}_0^{\mathcal{L}}, \bar{v}_1^{\mathcal{L}}\}\} \cup \{\{\bar{v}_{-2}^{\mathcal{R}}, \bar{v}_{-1}^{\mathcal{R}}\}, \{\bar{v}_{-1}^{\mathcal{R}}, \bar{v}_0^{\mathcal{R}}\}, \{\bar{v}_0^{\mathcal{R}}, \bar{v}_1^{\mathcal{R}}\}\}$
(edges for connecting the remaining **left/right** vertices),

$E^{+} = \{\{\bar{v}_0^{\mathcal{L}}, \bar{v}_{\mathcal{L}}^{+}\}, \{\bar{v}_{\mathcal{L}}^{+}, \bar{v}_1^{+}\}, \{\bar{v}_1^{+}, \bar{v}_2^{+}\}, \{\bar{v}_2^{+}, \bar{v}_3^{+}\}, \dots, \{\bar{v}_{n-1}^{+}, \bar{v}_n^{+}\}, \{\bar{v}_n^{+}, \bar{v}_{\mathcal{R}}^{+}\}, \{\bar{v}_{\mathcal{R}}^{+}, \bar{v}_0^{\mathcal{R}}\}\}$
(edges for connecting **left/right** vertices to the **gate part** of 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 **destination** vertices),

$E_A^{+} = \{\{\bar{v}_{1,1}^{+}, \bar{v}_{2,1}^{+}\}, \dots, \{\bar{v}_{1,n}^{+}, \bar{v}_{2,n}^{+}\}, \dots, \dots, \{\bar{v}_{\vartheta-1,1}^{+}, \bar{v}_{\vartheta,1}^{+}\}, \dots, \{\bar{v}_{\vartheta-1,n}^{+}, \bar{v}_{\vartheta,n}^{+}\}\}$
(edges for connecting rows of the **array part** of 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 $E = E^0 \cup E^{-} \cup E^{+} \cup E_G^{+} \cup E_A^{+} \cup E_{\times}^{+}$.

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

The required conjugation of robots into the left and right vertices at a certain time step can be satisfied if the robots move in a way that first all the robots $\bar{s}_1, \bar{s}_2, \dots, \bar{s}_n$ from the

set of vertices $\bar{v}_1^0, \bar{v}_2^0, \ldots, \bar{v}_n^0$ move into the set of vertices $\bar{v}_1^{\mathcal{L}}, \bar{v}_2^{\mathcal{L}}, \ldots, \bar{v}_n^{\mathcal{L}}$ if the left branch is chosen (or into the set of vertices $\bar{v}_1^{\mathcal{R}}, \bar{v}_2^{\mathcal{R}}, \ldots, \bar{v}_n^{\mathcal{R}}$ if the right branch is chosen). Without loss of generality, suppose the left branch has been chosen. Then the robots $\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_n$ together with the leading robot $\bar{l}_{\mathcal{L}}$ moves into vertices $\bar{v}_1^+, \bar{v}_2^+, \ldots, \bar{v}_n^+, \{\bar{v}_{\mathcal{R}}^+, \bar{v}_0^{\mathcal{R}}\}$. And finally, the robots $\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_n$ move 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) and the leading robots return to their goal positions in $\bar{v}_{-2}^{\mathcal{L}}$ and $\bar{v}_{-2}^{\mathcal{R}}$. The described behavior of robots within the optimal solution is ensured by locking proper vertices at proper time steps. That is, the multi-robot path planning instance $\Xi$ is further extended with additional robots and vertices used for locking vertices as it is shown in the proof of Corollary 1. However, for sake of simplicity the description below will be restricted on the original components of the problem $\Xi$.

Thus, the optimal solution for the **left branch** $\mathcal{S}^*_{\mathcal{L},R'}(\Xi) = [S^0_{\mathcal{L},R'}, S^1_{\mathcal{L},R'}, \ldots, S^{\zeta}_{\mathcal{L},R'}]$ should satisfy that $S^0_{\mathcal{L},R'}(\bar{s}_i) = \bar{v}_i^0$, $S^1_{\mathcal{L},R'}(\bar{s}_i) = \bar{v}_i^{\mathcal{L}}$,$S^2_{\mathcal{L},R'}(\bar{s}_i) = \bar{v}_{i-1}^{\mathcal{L}}$,…, $S^i_{\mathcal{L},R'}(\bar{s}_i) = \bar{v}_1^{\mathcal{L}}$, $S^{i+1}_{\mathcal{L},R'}(\bar{s}_i) = \bar{v}_0^{\mathcal{L}}$,$S^{i+2}_{\mathcal{L},R'}(\bar{s}_i) = \bar{v}_{\mathcal{L}}^+$,$S^{i+3}_{\mathcal{L},R'}(s_i) = \bar{v}_1^+$,$S^{i+4}_{\mathcal{L},R'}(\bar{s}_i) = \bar{v}_2^+$,…,$S^{n+2}_{\mathcal{L},R'}(\bar{s}_i) = \bar{v}_{n-i}^+$,$S^{n+3}_{\mathcal{L},R'}(\bar{s}_i) = \bar{v}_{n-i+1}^+$, $S^{n+4}_{\mathcal{L},R'}(\bar{s}_i) = \bar{v}_{1,n-i+1}^+$,$S^{n+5}_{\mathcal{L},R'}(\bar{s}_i) = \bar{v}_{2,n-i+1}^+$,…,$S^{n+\vartheta+2}_{\mathcal{L},R'}(\bar{s}_i) = \bar{v}_{\vartheta-1,n-i+1}^+$,and $S^{n+\vartheta+3}_{\mathcal{L},R'}(\bar{s}_i) = \bar{v}_{\vartheta,i}^+ = S^+_{R'}(\bar{s}_i)$ for $i = 1,2,\ldots,n$;$S^0_{\mathcal{L},R'}(\bar{l}_{\mathcal{L}}) = \bar{v}_0^{\mathcal{L}}$, $S^1_{\mathcal{L},R'}(\bar{l}_{\mathcal{L}}) \in \{\bar{v}_0^{\mathcal{L}}, \bar{v}_{\mathcal{L}}^+\}$,$S^2_{\mathcal{L},R'}(\bar{l}_{\mathcal{L}}) \in \{\bar{v}_{\mathcal{L}}^+, \bar{v}_1^+\}$, $S^3_{\mathcal{L},R'}(\bar{l}_{\mathcal{L}}) \in \{\bar{v}_1^+, \bar{v}_2^+\}$,…, $S^{n+1}_{\mathcal{L},R'}(\bar{l}_{\mathcal{L}}) \in \{\bar{v}_{n-1}^+, \bar{v}_n^+\}$,$S^{n+2}_{\mathcal{L},R'}(\bar{l}_{\mathcal{L}}) \in \{\bar{v}_n^+, \bar{v}_{\mathcal{R}}^+\}$,$S^{n+3}_{\mathcal{L},R'}(\bar{l}_{\mathcal{L}}) \in \{\bar{v}_{\mathcal{R}}^+, \bar{v}_0^{\mathcal{R}}\}$ (the left leading robot is going in front of the queue formed by the sequence of robots $\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_n$), there is no special requirement on $S^{n+4}_{\mathcal{L},R'}(\bar{l}_{\mathcal{L}})$, $S^{n+5}_{\mathcal{L},R'}(\bar{l}_{\mathcal{L}})$, …, $S^{n+\vartheta+2}_{\mathcal{L},R'}(\bar{l}_{\mathcal{L}})$, indeed $S^{n+\vartheta+3}_{\mathcal{L},R'}(\bar{l}_{\mathcal{L}}) = \bar{v}_{-2}^{\mathcal{L}} = S^+_{R'}(\bar{l}_{\mathcal{L}})$. Similarly, there is no special requirement on $S^i_{\mathcal{L},R'}(\bar{l}_{\mathcal{R}})$ for any $i = 1,2,\ldots,n$. The optimal solution for the **right branch** $\mathcal{S}^*_{\mathcal{R},R'}(\Xi) = [S^0_{\mathcal{R},R'}, S^1_{\mathcal{R},R'}, \ldots, S^{\zeta}_{\mathcal{R},R'}]$ has almost the same form. The only difference is that the final reversal of the robots $\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_n$ to fit the last row of the array part of destination vertices is not performed. Observe, that the time step at which **conjugation** occurs is $t = 1$.

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

The first row of array part of destination vertices, that is, vertices $\bar{v}_{1,1}^+, \bar{v}_{1,2}^+, \ldots, \bar{v}_{1,n}^+$, is locked (closed for entering) for all the time steps except the time step $n + 4$. At this time step all the robots $\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_n$ are entering the array part of destination vertices. Then they continue towards their goal positions and hence vertices $\bar{v}_{1,1}^+, \bar{v}_{1,2}^+, \ldots, \bar{v}_{1,n}^+$ 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 the time step 0.

At the time of opening the first row of the array part of destination vertices (at the time step $n + 3$), all the robots $\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_n$ must reside in the vertices $\bar{v}_1^+, \bar{v}_2^+, \ldots, \bar{v}_n^+$ (eventually in the reversed order). Otherwise, they have no chance to reach their goal positions at all. Then, the fastest way to reach their goal positions starting from vertices $\bar{v}_1^+, \bar{v}_2^+, \ldots, \bar{v}_n^+$ is exact following shortest paths to the last row of the array part of destination vertices (all these paths are of the same length). Since $\vartheta \geq n + 4$ which is enough

time steps for the leading robots to reach their destination positions; the motion of robots $\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_n$ within the array part of destination vertices represents the **bottleneck**.

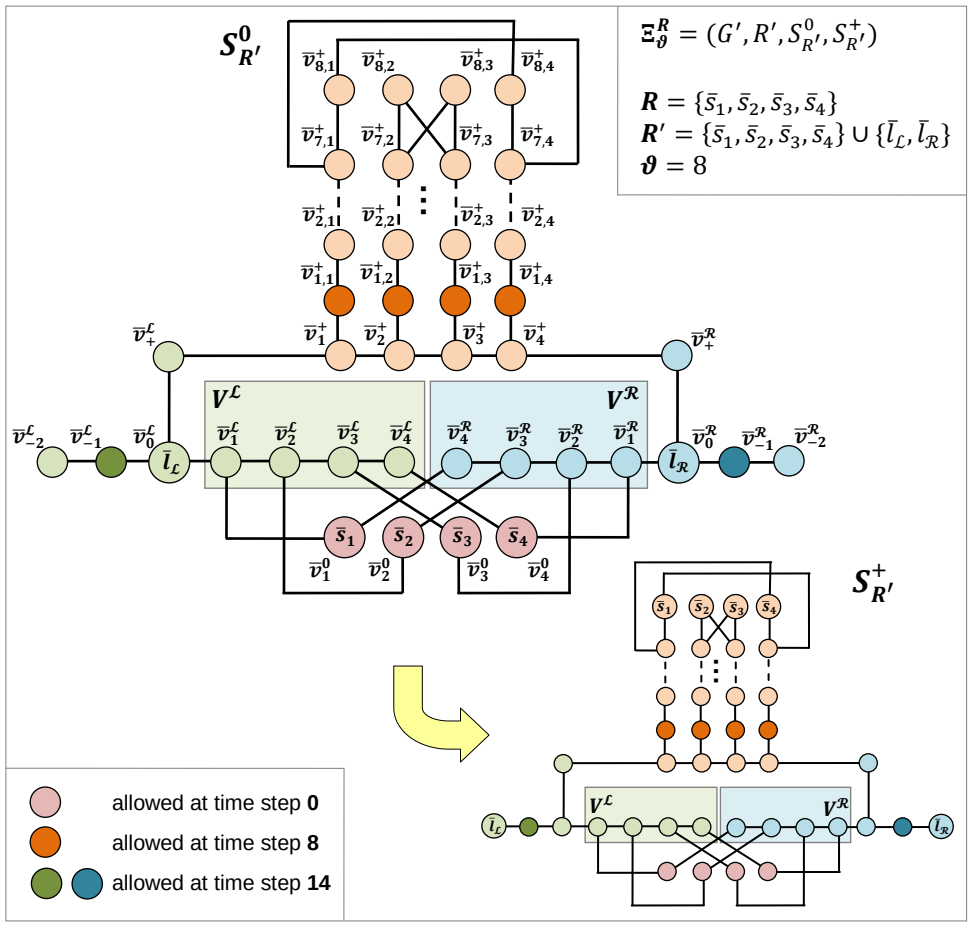



**Figure 7.** *A **conjugation instance** of the multi-robot path planning problem.* A conjugation instance $\Xi^R_\vartheta$ shown in figure is constructed with respect to a set of robots $\boldsymbol{R} = \{\bar{s}_1, \bar{s}_2, \bar{s}_3, \bar{s}_4\}$ and a parameter $\vartheta = 8$. The robots are 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 robots $\bar{s}_1$, $\bar{s}_2$, $\bar{s}_3$, and $\bar{s}_4$ 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 $\Xi^R_\vartheta$.

It remains to check the behavior of robots before the time step $n + 3$. Since the initial vertices are allowed to be occupied only at time step 0, the robots $\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_n$ must enter the left or the right vertices immediately at the next time step. Between time steps 1 and $n + 3$ it is **not possible to swap** robots in the currently accessible part of the graph since

it consists of a single path. Hence, if the robots $\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_n$ 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 robots $\bar{l}_{\mathcal{L}}$ and $\bar{l}_{\mathcal{R}}$.

The just described instance will be called a ***conjugation*** *instance of multi-robot path planning problem.* Notice, that the instance is parameterized by a set of robots $R$ and an integer parameter $\vartheta \geq |R| + 4$. An instance of the problem corresponding to the given parameters will be denoted as $\Xi_\vartheta^R$. Notice further, that the makespan of any optimal solution of $\Xi_\vartheta^R$ is $|R| + \vartheta + 3 \geq 2|R| + 7$.

It is easy to see that the **size** of the conjugation instance is $(3 + \vartheta)|R| + 8$ which is $\mathcal{O}(\vartheta|R|)$. An instance of the conjugation instance of multi-robot path planning problem is shown in Figure 7.

All the ingredients are now prepared to prove that a *decision version* of the optimization variant of multi-robot path planning is $NP$-complete [6]. The **membership** into ***NP*** will be checked first. Then a **polynomial time reduction** of a *Boolean satisfiability* instance (***SAT***) **[**1**]** to the instance of the decision version of the optimization variant of multi-robot path planning will be constructed.

**Definition 3** ***(decision version).*** A *decision version of the optimization variant of multi-robot path planning* is a task to decide for a given instance of multi-robot path planning $\Sigma$ and a number $\eta \in \mathbb{N}_0$ whether there exists a solution $\mathcal{S}_R(\Sigma)$ of the makespan no longer than $\eta$. A notation $\Sigma/\eta$ will be used for the decision instance. Next, let $MRPP_{OPT}$ denotes the language of positive instances of this problem. □

It is not that easy to see that $MRPP_{OPT} \in NP$, since no upper bound on the size of the solution of $MRPP_{OPT}$ has been established so far. Hence, the standard technique of "guessing and checking" cannot be used immediately. Notice that, decision variants of several related *sliding piece problems* [9] such as ***Sokoban*** *game* [3] and ***Rush-hour*** *puzzle* [5] are proven to be $PSPACE$-complete [6, 7] but it is not known whether they are in $NP$. The reason it that the polynomial upper bound on the size of the solution has not been found so far. Fortunately, this is not the case of $MRPP_{OPT}$. It is possible to establish the polynomial upper bound on the size of the solution of $MRPP_{OPT}$ using results shown in [11].

**Lemma 3.** $MRPP_{OPT} \in NP$. ■

**Proof.** It has been shown in [11] that there exists a solution $\mathcal{S}_P(\Pi) = [S_P^0, S_P^1, \ldots, S_P^\xi]$ for any solvable instance of the problem of pebble motion on a graph $\Pi = (G = (V, E), P, S_P^0, S_P^+)$ such that $\xi \in \mathcal{O}(|V|^3)$ ($\xi$ is regarded as a function of $\Pi$ here). Since the solution of an instance of pebble motion on a graph can be used as a solution of the corresponding multi-robot path planning instance (Proposition 1) it implies that there exists a solution $\mathcal{S}_R(\Sigma) = [S_R^0, S_R^1, \ldots, S_R^\zeta]$ for any solvable instance of the problem of multi-robot path planning $\Sigma = (G = (V, E), R, S_R^0, S_R^+)$ such that $\zeta \in \mathcal{O}(|V|^3)$ ($\zeta$ is regarded as a func-

tion of Σ as well). An instance of $MRPP_{OPT}$ $\Sigma/\eta$ can be solved on a Turing machine with oracle [8] in polynomial time as follows. A solution of the size $\mathcal{O}(|V|^3)$ of Σ is generated first by the oracle. Then, the generated solution is checked 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 $\Sigma/\eta$. ■

As it is usual, Boolean formulas in *conjunctive normal form* (*CNF*) [10] will be used in further reasoning. Let $SAT$ denotes the language of satisfiable instances of Boolean formulas in CNF. It is well known that $SAT$ is $NP$-complete. However, a slight technical adaptation of Boolean satisfiability is necessary to carry out the required reduction. 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 will be denoted as $SAT_{=}$.

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

**Proof.** With respect to membership into $NP$, 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. Let $F$ is a formula in CNF and let $x$ be a variable with **unbalanced** positive and negative occurrences. Let $pos(x, F)$ denotes a set of positive occurrences of $x$ in $F$ and similarly let $neg(x, F)$ denotes a set of negative occurrences of $x$ in $F$. 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 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 of $F$, thus the reduction can be done in polynomial time. ■

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

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

Assume further that that each variable has the same number of positive and negative occurrences in $F_{=}$. Let $Var(F_{=})$ denotes the set of Boolean variables of $F_{=}$. An instance $\Sigma = (G = (V, E), R, S_R^0, S_R^+)/\eta$ of the decision version of the optimization variant of multi-robot path planning 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 $i$th clause as $j$th disjunct. A **conjugation** instance of multi-robot path planning will be associated with each Boolean 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 $F_=$ (conjugation graph has also the same number of left and right vertices).

The idea is to prepare a group of robots of the size $|pos(x, F_=)| = |neg(x, F_=)|$ for each Boolean variable $x \in Var(F_=)$. This group of robots 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 robots must go either into the vertices corresponding to positive literals or into the vertices corresponding to negative literals. If the movement of robots in interpreted in the way that literals corresponding to vertices of $V^{F_=}$ visited at time step 1 will be assigned the same Boolean value, then the conjugation technique assures **Boolean 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 robots to simulate **clause satisfaction** by any solution which makespan is at most $\eta$. This can be done by enforcing robots either to visit at least one literal/vertex of each clause of $F_=$ (in the case when 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 the case when visited literals/vertices are assigned the value $FALSE$). Since the second option can be easily 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 exact specification of $\eta$ need to be discussed. The equality between makespans of optimal solutions over the individual conjugation instances needs to be established. Recall that a conjugation instance $\Xi_\vartheta^R$ is characterized by two parameters $R$ (the set conjugated robots) 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 $\Xi_{\vartheta(x)}^{R(x)}$ will have the parameters $R(x) = \{\bar{s}_1^x, \bar{s}_2^x, \ldots, \bar{s}_{|pos(x,F_=)|}^x\}$ and $\vartheta(x) = 2\varrho - |pos(x, F_=)| + 4 \geq |R(x)| + 4$. Hence, the makespan of any optimal solution of the conjugation instance $\Xi_{\vartheta(x)}^{R(x)}$ is $|R(x)| + \vartheta(x) + 3 = 2\varrho + 7$.

Matching of left and right vertices of $\Xi_{\vartheta(x)}^{R(x)}$ to vertices from $V^{F_=}$ is as follows: $\{\bar{v}_1^{x,\mathcal{L}}, \bar{v}_2^{x,\mathcal{L}}, \ldots, \bar{v}_{|R(x)|}^{x,\mathcal{L}}\} = \{\bar{l}_j^i | l_j^i = \neg x; i = 1,2,..n; j = 1,2,\ldots,k_i\}$ and $\{\bar{v}_1^{x,\mathcal{R}}, \bar{v}_2^{x,\mathcal{R}}, \ldots, \bar{v}_{|R(x)|}^{x,\mathcal{R}}\} = \{\bar{l}_j^i | l_j^i = x; i = 1,2,..n; j = 1,2,\ldots,k_i\}$. Now, the 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 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.

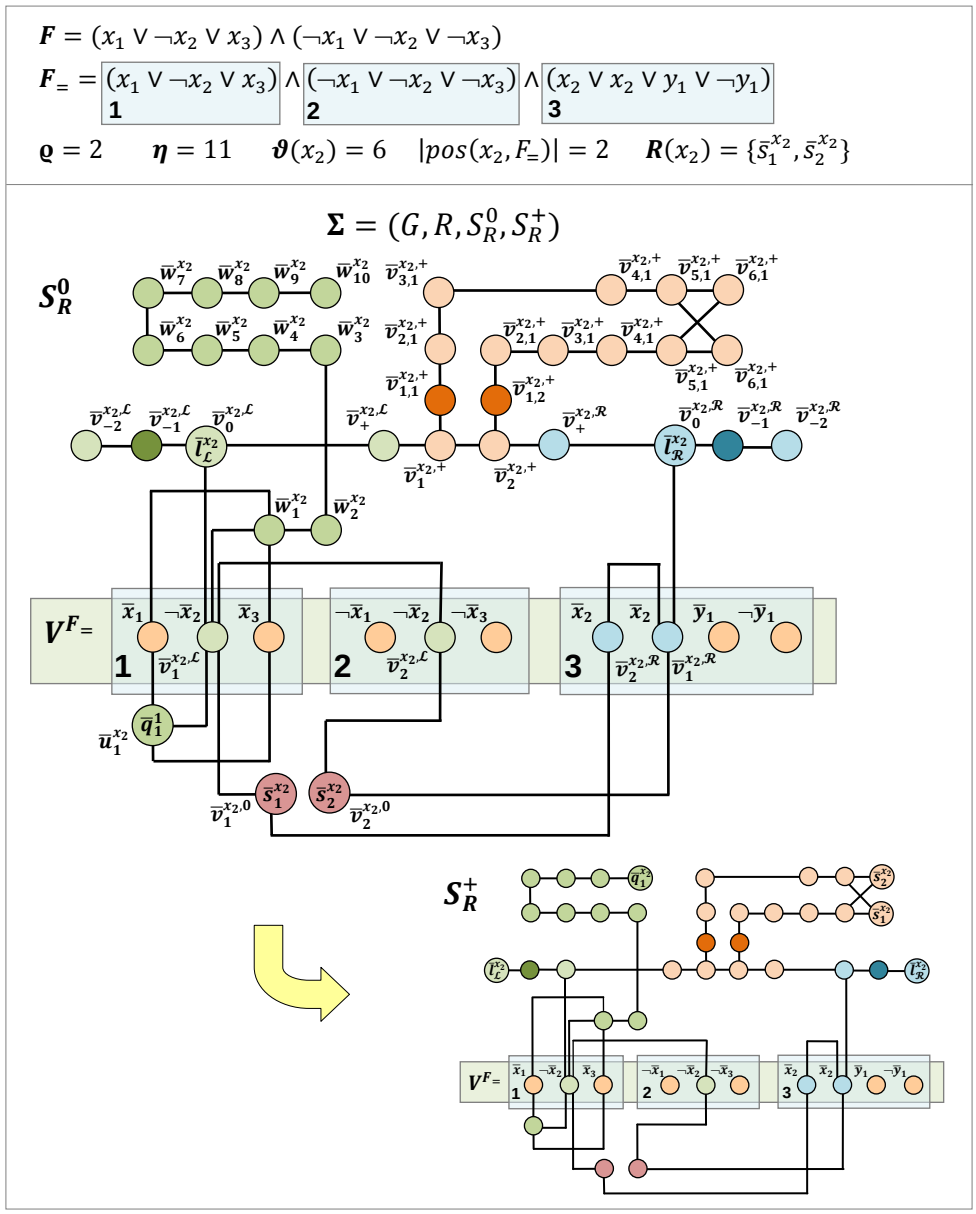



**Figure 8.** *A **polynomial time reduction** of a Boolean formula to a decision instance of multi-robot path planning.* A formula $F$ is transformed to a formula $F_=$ in which each variable has the same number of positive and negative occurrences. Then an instance of the decision version of the problem of multi-robot path planning $\Sigma/\eta$ is constructed. The conjugation technique is used to simulate **Boolean 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 $\Sigma$ of the makespan $\eta = 11$ if and only if the formula $F$ is satisfiable.

Let $e: Var(x) \longrightarrow \{FALSE, TRUE\}$ be a satisfying valuation of $F_{=}$. If $e(x) = FALSE$, then robots $\bar{r}_1^x, \bar{r}_2^x, \ldots, \bar{r}_{|pos(x,F_{=})|}^x$ are placed in $\bar{v}_1^{x,\mathcal{R}}, \bar{v}_2^{x,\mathcal{R}}, \ldots, \bar{v}_{|R(x)|}^{x,\mathcal{R}}$ at time step 1; if $e(x) = TRUE$ then they are placed in $\bar{v}_1^{x,\mathcal{L}}, \bar{v}_2^{x,\mathcal{L}}, \ldots, \bar{v}_{|R(x)|}^{x,\mathcal{L}}$ at time step 1. The placement of robots 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 at least one vertex from the set of vertices that corresponds to literals of a clause is not occupied by robots from the set $\bigcup_{x \in Var(F_{=})} R(x)$ within **any optimal solution** while the makespan of $2\varrho + 7$ remains preserved. That is, Corollary 4 is invoked with $W = \{\bar{l}_1^i, \bar{l}_2^i, \ldots, \bar{l}_{k_i}^i\}$ that corresponds to satisfying the $i$th clause of $F_{=}$. Let $\Sigma$ denotes 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 size of $F_{=}$; the size of $\Sigma$ is also polynomial (the size of each conjugation subgraph is polynomial in 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 be easily derived from this solution. Hence, $F_{=} \in SAT_{=}$ if and only if $\Sigma/\eta \in MRPP_{OPT}$. Together with Lemma 3 the claim that $MRPP_{OPT}$ is $NP$-complete has been obtained. ■

### 3.2. *The Case with Bi-connected Graphs*

The interesting open question is the complexity of the optimization variant of multi-robot path planning when only **bi-connected graphs** [30, 31] are took into account. As this is an important restricted case of the problem with respect to the existence of a solution, it is useful to know its complexity.

With some **exceptions** it can be briefly stated the following. An instance of the problem (pebble motion or multi-robot) over a bi-connected graph with single unoccupied vertex is **always solvable** if the goal arrangement of pebbles/robots differs from the initial one as an *even permutation* (it is supposed that an unoccupied vertex has the same position in the initial and the goal arrangement) [11, 32]. If the goal arrangement differs from the initial one as an odd permutation, then there must be an odd cycle in the input graph to be **able to construct a solution**, otherwise there is no solution of the instance [11, 32]. The exceptional cases are formed by special bi-connected graphs: the first one is a simple cycle and the second one is a bi-connected graph consisting of one handle connected to a cycle. If there are at least **two unoccupied vertices** in the input bi-connected graph other than a simple cycle, then the instance is always solvable [18, 21].

Unfortunately, the decision version of the optimization variant of the problem on bi-connected graphs remains $NP$-complete since all the augmentations of instances used in the proof of the $NP$-completeness can be further augmented to preserve bi-connectivity.

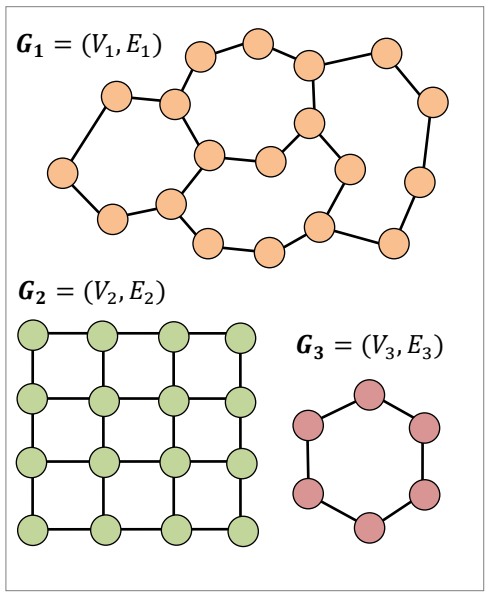



**Figure 9.** *Examples of* ***bi-connected graphs****.* Three bi-connected graphs $G_1$, $G_2$, and $G_3$. $G_1$ and $G_2$ are non-trivial bi-connected graphs. $G_3$ is a trivial one.

**Definition 4** ***(connected graph).*** An undirected graph $G = (V, E)$ is ***connected*** if $|V| \geq 2$ for any two vertices $u, v \in V$ such that $u \neq v$ there is an undirected path consisting of edges from $E$ connecting $u$ and $v$. □

**Definition 5 (*bi-connected graph, non-trivial).*** An undirected graph $G = (V, E)$ is ***bi-connected*** if $|V| \geq 3$ and the graph $G' = (V', E')$, where $V' = V \setminus \{v\}$ and $E' = \{\{u, w\} | u, w \in V \wedge u \neq v \wedge w \neq v\}$, is connected for every $v \in V$. A bi-connected graph not isomorphic to a cycle will be called ***non-trivial*** bi-connected graph. □

Observe that, if a graph is bi-connected, then every two distinct vertices are connected by at least two *vertex disjoint paths* (equivalently, there is a cycle containing both vertices; only internal vertices of paths are considered when speaking about vertex disjoint paths - vertex disjoint paths can intersect at their *start points* and *endpoints*). If a graph is not bi-connected then it is either disconnected or there exists a vertex which removal partitions the graph into at least two connected components – this vertex is called an *articulation point*. Several examples of bi-connected graphs are shown in Figure 9.

**Definition 6** ***(decision version - bi-connected).*** The optimization variant of multi-robot path planning can be naturally restricted on bi-connected graphs. A language of positive instances of the decision version of the optimization variant of multi-robot path planning on bi-connected graphs will be denoted as $MRPP_{OPT}^{bi}$. □

**Lemma 5.** $MRPP_{OPT}^{bi} \in NP$. ■

**Sketch of proof.** The proof of Lemma 3 applies here as well since additional requirement of **bi-connectivity** of the graph has does not affect the size of the input instance. Thus, due to [11] there exists a polynomial sized solution with respect to the size of the input of any solvable instance restricted on bi-connected graphs. The makespan of such a solution can be checked against a given bound in polynomial time as well. ■

**Theorem 2.** $MRPP^{bi}_{OPT}$ is $NP$-complete. ■

**Proof.** It remains to show that $MRPP^{bi}_{OPT}$ is $NP$-hard. Fortunately, this can be carried out by reducing $SAT_{=}$ to $MRPP^{bi}_{OPT}$ as it was done in above sections with additional augmentations to preserve bi-connectivity.

The **vertex locking** augmentation (Lemma 1) can be modified to preserve bi-connectivity. That is, if the graph $G$ of a multi-robot path planning instance $\Sigma = (G = (V, E), R, S^0_R, S^+_R)$ is bi-connected, then the graph $G'$ of the augmented instance $\Sigma' = (G' = (V', E'), R', S^0_{R'}, S^+_{R'})$, where a selected vertex $v \in V$ is locked for selected time steps, can be made bi-connected as well. The following text refers to the symbols defined in the proof of Lemma 1.

Observe that vertices added within the vertex locking augmentation are connected by a single path only. Thus, the second vertex disjoint path should be added while the vertex locking mechanism must be preserved. This can be done by connecting vertices $\bar{u}_{t_n}$ and $\bar{w}_\lambda$ that were added in the vertex locking augmentation with a new path. This ensures that all the vertices in the graph are connected by at least two disjoint paths. The removal of any of the added vertices does not disconnect the graph. However, the removal of $v$ can disconnect the newly added cycle around $v$ from the rest of the graph. Thus, the cycle needs to be connected to the rest of the graph by some additional edge to preserve bi-connectivity.

The new path forming the cycle must be constructed long enough to make robots added in the augmentation (that is, robots $R_X = \{\bar{q}_1, \bar{q}_2, \ldots, \bar{q}_n\}$) to move in the same way as in the original augmentation in any optimal solution. Moreover, the connection edge for preserving bi-connectivity must be far enough so that newly added robots cannot use it. The added robots consume $\lambda + t_1 = \zeta^* + t_n$ time steps before they reach their destinations (they go along the shortest possible path). Thus, it must take more than $\lambda + t_1$ time steps to get into destinations if the new alternative path is chosen. Moreover, it must take more than $\lambda + t_1$ time steps before the bi-connectivity preserving edge can be reached.

Let $\bar{z}_1, \bar{z}_2, \ldots, \bar{z}_{\lambda+t_1}, \bar{z}^c, \bar{z}_{\lambda+t_1+1}, \bar{z}_{\lambda+t_1+2}, \ldots, \bar{z}_{2(\lambda+t_1)}$ be new vertices making the new alternative path. The new vertices are connected into the augmented graph using the following edges: $\{\bar{u}_{t_n}, \bar{z}_1\}$, $\{\bar{z}_1, \bar{z}_2\}$, …, $\{\bar{z}_{\lambda+t_1}, \bar{z}^c\}$, $\{\bar{z}^c, \bar{z}_{\lambda+t_1+1}\}$, $\{\bar{z}_{\lambda+t_1+1}, \bar{z}_{\lambda+t_1+2}\}$, …, $\{\bar{z}_{2(\lambda+t_1)-1}, \bar{z}_{2(\lambda+t_1)}\}$, $\{\bar{z}_{2(\lambda+t_1)}, \bar{w}_\lambda\}$, and $\{\bar{z}^c, u\}$, where $u \in V$ such that $u \neq v$. The edge $\{\bar{z}^c, u\}$ is used to preserve bi-connectivity.

Now observe that the robot $\bar{q}_i$ must travel at least along $2(\lambda + t_1) + 2$ if it chooses the alternative path. This however cannot happen in an optimal solution since its makespan should be $\lambda + t_1$. Similarly, the robot $\bar{q}_i$ must travel at least along $\lambda + t_1 + 1$ edges if it tries to used the bi-connectivity preserving edge. Again, this cannot happen in an optimal solution since its makespan should be $\lambda + t_1$. It holds, that $2(\lambda + t_1)$ is $\mathcal{O}(\max(\zeta^*, t_n))$ since $2(\lambda + t_1) = 2(\zeta^* + t_n) \leq 4 * \max(\zeta^*, t_n)$; thus the size of the augmented instance is again $\mathcal{O}(\max(\zeta^*, t_n) + |V| + |E|)$ as in the original construction.

The second construction where bi-connectivity is needed to be preserved is the **conjugation instance**. The only vertices that are not connected by two vertex disjoint paths

within the conjugation instance are: $\bar{v}^{\mathcal{L}}_{-2}$, $\bar{v}^{\mathcal{L}}_{-1}$, $\bar{v}^{\mathcal{R}}_{-1}$, and $\bar{v}^{\mathcal{R}}_{-2}$. This can be treated by adding an edge $\{\bar{v}^{\mathcal{L}}_{-2}, \bar{v}^{\mathcal{R}}_{-2}\}$ which makes the graph of the conjugation instance bi-connected. Since vertices $\bar{v}^{\mathcal{L}}_{-2}$, $\bar{v}^{\mathcal{L}}_{-1}$, $\bar{v}^{\mathcal{R}}_{-1}$, and $\bar{v}^{\mathcal{R}}_{-2}$ are accessible at last two time steps only within any optimal solution, the newly added edge cannot affect the way how robots must move within the optimal solution. The size of the modified conjugation instance is still $\mathcal{O}(\vartheta|R|)$.

Now, it is possible to construct a decision version of the optimization variant of multi-robot path planning problem instance $\Sigma^{bi}/\eta$ with the **bi-connected** graph for a formula $F_{=}$ in CNF where each variable has the same number of positive and negative occurrences such that $F_{=} \in SAT_{=}$ if and only if $\Sigma^{bi}/\eta \in MRPP^{bi}_{OPT}$. The construction from the proof of Theorem 1 can be adopted with the modification that all the vertex locking augmentations and conjugation instances will preserve bi-connectivity as it is described above. The size of resulting instance $\Sigma^{bi}$ is polynomial in the size of $F_{=}$. Together with Lemma 5 the claim that $MRPP^{bi}_{OPT}$ is $NP$-complete has been obtained. ■

### 3.3. ***Summary on Complexity Results***

It has been shown in the above section or in existent works that all the interesting cases of the optimization variants of multi-robot path planning as well as pebble motion on a graph are $NP$-complete. Thus, unless it holds that $P = NP$ which is considered to be unlikely, optimization variants of these problems can be regarded as **intractable**. On the other hand, if the requirement on the optimal makespan is relaxed, all the studied cases fall into the $P$ class.

To provide a complete figure about studied cases of problems of motion on a graph at one place, all the complexity results from above sections and from existent works are summarized in Table 1.

**Table 1.** *A summary of* ***complexity results*** *of problems of* ***motion on a graph***. Complexity results of several interesting cases of problems of pebble motion on a graph and multi-robot path planning are summarized in the table. References to detailed proofs from the above sections or references to existent works that discuss the listed results are provided.

| **Pebble motion on a graph** | | ***Standard*** *variant* / ***Reference*** | | ***Optimization*** *variant* / ***Reference*** | |
|---|---|---|---|---|---|
| **Pebble motion** on a graph | **Arbitrary** graph | ***P* class** | [11, 32] + 2.2 | ***NP*-complete** | [13] + 2.2 |
| | **Bi-connected** graph | ***P* class** | [11, 32] + 2.2 | ***NP*-complete** | [13] |
| **Multi-robot** path planning | **Arbitrary** graph | ***P* class** | [11, 32] + 2.2 | ***NP*-complete** | 3 |
| | **Bi-connected** graph | ***P* class** | [11, 18,32] | ***NP*-complete** | 3 |

## 4. Related Works and Conclusion

An **abstraction** of the problem of multi-robot path planning is introduced in this paper. The relation to well known problems of pebble motion on a graph is studied. The new

result shown in this paper is that the decision version of the optimization variant of the problem of multi-robot path planning is ***NP*-complete**. The parameter with respect to which the optimization is made is the makespan. The same complexity result has been obtained for the class of instances restricted on bi-connected graphs only.

The **reduction** of Boolean satisfiability to multi-robot path planning has been used for the proof of *NP*-completeness. Numerous techniques how to simulate Boolean consistency and clause satisfaction within multi-robot path planning were developed in this work. These techniques are supposedly generic enough to be used in different context and hence considered to be interesting itself.

The fact that optimization variant of multi-robot path planning is *NP*-complete is quite negative result. Fortunately, if the requirement on the shortest possible makespan of solutions relaxed, the problem becomes **tractable**. Namely, it belongs to the *P* class. However, the situation is not that straightforward. Although algorithms developed for solving pebble motion problems [11, 32] can be used for solving multi-robot path planning as well, this practice is disadvantageous. Solutions generated by these algorithms have promising theoretical makespan of $\mathcal{O}(|V|^3)$. However, constants in the asymptotic estimation are too high. The experimental evaluation made within different work has shown that the makespan of these solutions measured empirically is relatively high as well [24]. Hence, these algorithms are unsuitable for practical use. This issue has been addressed in details in [19, 20, 21, 24] where alternative solving algorithm for multi-robot path planning problem producing better solutions (so called *BIBOX* algorithms) and solution improving techniques were proposed.

The important **related work** is represented by articles [26, 27, 28, 29]. Authors study so called *multi-agent path planning* which is similar to the notion of multi-robot path planning with some further relaxations (for example a swap of agents along an edge seems to be allowed). The number of moves is the optimized parameter. Authors define the tractable class of this optimization problem where graphs are restricted on grids and there is a relative abundance of unoccupied vertices. The theoretical relation of multi-agent path planning and multi-robot path planning is an interesting question for future work particularly in the light of presented complexity results.

In [17] it is claimed that it is possible to solve a similar problem to multi-robot path planning – so called *cooperative path-planning problems* - **optimally**. However, it is little bit exaggerated claim since author actually shown that instances with up to only 60 movable units in the environment with lot of free space can be solved by the search with several heuristic improvements in reasonable runtime.

An interesting question for future work is whether it is tractable to find a solution of multi-robot path planning instance which is constant time worse that the optimum. Currently, it is an open question whether such **approximation** solving algorithm can exist. The answer to this question will consequently provide the estimation how far from the optimum are solutions generated by algorithms for the standard case of the problem (not the optimal one) referred above. Consequently, the estimation what is the makespan of the optimal solution of large instances would be also available.

**Acknowledgments**

This work is supported by The Czech Science Foundation (Grantová agentura České republiky - GAČR) under the contract number 201/09/P318 and by The Ministry of Education, Youth and Sports, Czech Republic (Ministerstvo školství, mládeže a tělovýchovy ČR – MŠMT ČR) under the contract number MSM 0021620838.

**References**

1. 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.
2. 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.
3. 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 [April 2010].
4. J. D. **Dixon** and B. **Mortimer**. *Permutation Groups*. in Graduate Texts in Mathematics, Volume 163, Springer, 1996, ISBN 978-0-387-94599-6.
5. 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.
6. 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.
7. 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.
8. J. E. **Hopcroft**, R. **Motwani**, J. D. **Ullman**. *Introduction to Automata Theory, Languages, and Computation.* Addison Wesley, 2000, ISBN: 978-0201441246.
9. E. **Hordern**. *Sliding Piece Puzzles.* Oxford University Press, 1986, ISBN: 978-0198532040.
10. P. **Jackson**, D. **Sheridan**. *Clause Form Conversions for Boolean 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.
11. 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.
12. 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.
13. 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.
14. M. R. K. **Ryan**. *Graph Decomposition for Efficient Multi-Robot Path Planning*. Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007), pp. 2003-2008, IJCAI Conference, 2007.
15. M. R. K. **Ryan**. *Exploiting Subgraph Structure in Multi-Robot Path Planning*. Journal of Artificial Intelligence Research (JAIR), Volume 31, pp. 497-542, AAAI Press, 2008.
16. P. E. **Schupp** and R. C. **Lyndon**. *Combinatorial group theory*. Springer, 2001, ISBN 978-3-540-41158-1.

17. 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.
18. P. **Surynek**. *A Novel Approach to Path Planning for Multiple Robots in Bi-connected Graphs*. Proceedings of the 2009 IEEE International Conference on Robotics and Automation (ICRA 2009), pp. 3613-3619, IEEE Press, 2009.
19. P. **Surynek**. *Towards Shorter Solutions for Problems of Path Planning for Multiple Robots in θ-like Environments*. Proceedings of the 22nd International FLAIRS Conference (FLAIRS 2009), pp. 207-212, AAAI Press, 2009.
20. P. **Surynek**. *Making Solutions of Multi-robot Path Planning Problems Shorter Using Weak Transpositions 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 [July 2009].
21. P. **Surynek**. *An Application of Pebble Motion on Graphs to Abstract Multi-robot Path Planning.* Proceedings of the 21st International Conference on Tools with Artificial Intelligence (ICTAI 2009), pp. 151-158, IEEE Press, 2009.
22. P. **Surynek**. *An Optimization Variant of Multi-Robot Path Planning is Intractable.* Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI 2010), pp. 1261-1263, AAAI Press, 2010.
23. P. **Surynek**. *Abstract Path Planning for Multiple Robots: A Theoretical Study.* Technical Report, http://ktiml.mff.cuni.cz/~surynek/index.html.php?select=publications, Charles University in Prague, Czech Republic.
24. P. **Surynek**. *Abstract Path Planning for Multiple Robots: An Empirical Study.* Technical Report, http://ktiml.mff.cuni.cz/~surynek/index.html.php?select=publications, Charles University in Prague, Czech Republic.
25. 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.
26. K. C. **Wang** and A. **Botea**. *Tractable Multi-Agent Path Planning on Grid Maps*. Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009), pp. 1870-1875, IJCAI Conference, 2009.
27. 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.
28. 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.
29. 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.
30. D. B. **West**. *Introduction to Graph Theory.* Prentice Hall, 2000, ISBN: 978-0130144003.
31. J. **Westbrook**, R. E. **Tarjan**. *Maintaining bridge-connected and biconnected components online*. Algorithmica, Volume 7, Number 5&6, pp. 433–464, Springer, 1992.
32. R. M. **Wilson**. *Graph Puzzles, Homotopy, and the Alternating Group*. Journal of Combinatorial Theory, Ser. B 16, pp. 86-96, Elsevier, 1974.