An Application of Pebble Motion on Graphs to Abstract Multi-robot Path Planning

Pavel Surynek
Charles University in Prague, Faculty of Mathematics and Physics
Department of Theoretical Computer Science and Mathematical Logic
Malostranské náměstí 2/25, 118 00 Praha 1, The Czech Republic
pavel.surynek@mff.cuni.cz

Abstract

An abstraction of a problem of rearranging group of mobile robots is addressed in this paper (the problem of multi-robot path planning). The robots are moving in an environment in which they must avoid obstacles and each other. An abstraction where the environment is modeled as an undirected graph is adopted throughout this work. A case when the graph modeling the environment is bi-connected is particularly studied. This paper puts into a relation the well known problems of moving pebbles on graphs (sliding box puzzles) with problems of multi-robot path planning. Theoretical results gained for problems of pebble motion on graphs are utilized for the development of algorithms for multi-robot path planning. As the optimization variant of both problems (a shortest solution is required) is known to be computationally hard (NP-hard), we concentrate on construction of sub-optimal solving procedures. However, the quality of solution is still an objective. Therefore a process of composition of a sub-optimal solution of the problem (a plan) of the pre-calculated optimal plans for the sub-problems (macros) is suggested. The plan composition using macros was integrated into two existing sub-optimal solving algorithms. In both cases, substantial improvements of the quality of resulting plans were achieved in comparison to the original algorithms. The no less important result is that one of the existing algorithms was generalized by integrating macros for larger class of problems of multi-robot path planning.

1. Introduction and Motivation

Consider a group of mobile robots moving in a certain environment where each robot needs to reach a given goal position. The robots must avoid obstacles and each other. An abstraction where the environment with obstacles is modeled as an undirected graph is adopted throughout this work. This abstraction of the task is widely known as a problem of *multi-robot path planning* [5]. It ranks among the most challenging problems of artificial intelligence [4] and it motivates efforts of theorists as well as practicians [5]. The main difficulty of the problem arises from the requirement on the optimality of solutions (shortest sequences of moves are required) and from complex interactions among robots (spatial-temporal motion of one robot is influenced by motion of the other robots) [3].

The primary motivations for the problem of multi-robot path planning are tasks of moving objects within a limited free space. These tasks include rearranging containers in storage yards, coordination of movements of a large group of automated agents, or optimization of dense traffic. However, this is not the only motivation. Many tasks from virtual spaces can be also viewed as problems of path planning for multiple robots. An example may be data transfer with limited buffers at communication nodes, a coordination of a group of agents in strategic computer games, or planning movements in mass scenes in computer-generated imagery.

In contrast to a *multi-agent* approach where each robot behaves as an autonomous individual, the whole group of robots is treated as a single entity in our approach. This allows producing solutions of higher quality (shorter solutions) since the centralized control can capture more global view of the situation. Moreover, the adopted graph abstraction allows using graph theoretical results to tackle the problem. In particular, a similarity between the formal definitions of the problem of *multi-robot path planning* as it is discussed in [5] and the problem of *pebble motion on a graph* [2] has been observed. As there is lot of theoretical results for pebble motion on graphs we tried to utilize these results in our algorithms.

Both studied problems are computationally difficult when the shortest possible solution is required (*NP-hard*) [3]. Therefore we concentrate on developing of sub-optimal methods. However, the quality of solutions is still

This work is supported by the Czech Science Foundation under the contracts number 201/07/0205, 201/09/P318 and by the Ministry of Education, Youth and Sports, Czech Republic under the contract number MSM 0021620838.

an objective. Our suggestion is to improve existing sub-optimal methods by utilization of pre-calculated optimal solutions of sub-problems (*macros*) [7] from which the overall solution is composed. A successful attempt to apply pre-calculated optimal macros within two existing state-of-the-art sub-optimal algorithms [2], [6] has been made. The resulting modified algorithms proved to be better in terms of the length of generated solutions (shorter solutions are preferred) as well as in terms of runtime. Moreover, the algorithm from [6] was generalized by integration of macros for a larger class of problems.

The main contributions of this paper consist in the following aspects: **(i)** the problem of multi-robot path planning and the problem of pebble motion on a graph are put into relation, **(ii)** two existing algorithms are improved by integration of macro utilization – one of the described algorithms becomes state-of-the-art for certain class of problems, and **(iii)** this algorithm was also extended so it is now applicable to more general class of problems.

2. Pebble Motion on Graphs and Multi-robot Path Planning

Consider an environment in which a group of mobile robots is moving. Each robot starts at the given initial position and needs to reach a given goal position. The problem being addressed in this paper consists in finding spatial-temporal path for each robot to reach its goal. The robots **must not collide** with each other and they must **avoid obstacles** in the environment.

A relatively strong abstraction is adopted in this paper. The environment with obstacles within that the robots are moving is modeled as an undirected graph. The vertices of the graph represent positions in the environment and the edges model an unblocked way from one position to another. 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*.

2.1. Formal Definitions of the Problems

The following two definitions formalizes a problem of *pebble motion on a graph* (also called a *pebble motion puzzle* or *sliding box puzzle*) [2], [11] and the related problem of *multi-robot path planning* [5] (both problems and their solutions are illustrated in figure 1).

Definition 1 *(problem of pebble motion on a graph)*. Let us have an undirected graph $G=(V,E)$. Next, let us have a set of pebbles $P=\{p_1,p_2,\ldots,p_\mu\}$ where $\mu<|V|$. The ***initial*** arrangement of the pebbles is defined by a simple function $S_P^0: P\rightarrow V$ (that is $S_P^0(p_i)\neq S_P^0(p_j)$ for $i,j=1,2,\ldots,\mu$ with $i\neq j$). The ***goal*** arrangement of the pebbles is defined by a simple function $S_P^+: P\rightarrow V$ (that is $S_P^+(p_i)\neq S_P^+(p_j)$ for $i,j=1,2,\ldots,\mu$ with $i\neq j$). The problem of ***pebble motion on a graph*** is a task to find a number ξ and a sequence of moves represented as a sequence of vertices $M_p=[m_1^p,m_2^p,\ldots,m_\xi^p]$ for every pebble $p\in P$ where $m_i^p\in V$ for $i=1,2,\ldots,\xi$, $m_1^p=S_P^0(p)$, $m_l^r=S_P^+(p)$, and either $\{m_i^p,m_{i+1}^p\}\in E$ or $m_i^p=m_{i+1}^p$ for $i=1,2,\ldots,\ \xi-1$. Furthermore, sequences of vertices $M_p=[m_1^p,\ m_2^p,\ldots,m_\xi^p]$ and $M_q=[m_1^q,m_2^q,\ldots,$ $m_\xi^q]$ for every two pebbles $p\in P$ and $q\in P$ such that $p\neq q$ must satisfy that $m_{i+1}^p\neq m_i^q$ for $i=1,2,\ldots,\ \xi-1$ (the target vertex must be unoccupied) and $m_i^p\neq m_i^q$ for $i=1,2,\ldots,\xi$ (no other pebble can simultaneously enter the same target vertex). □

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 for a moving pebble/robot must be freed in the previous time step is relaxed. A motion of a robot entering the target vertex that is simultaneously left by another robot is allowed in multi-robot path planning. The problem is formalized in the following definition.

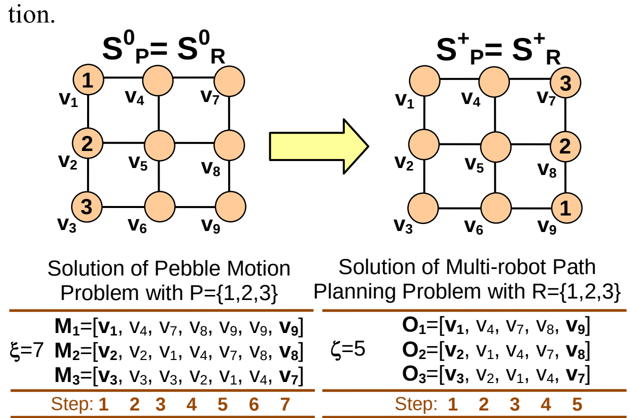


Figure 1. *An illustration of problems of **pebble motion on a graph** and **multi-robot path planning**.* The task is to move pebbles/robots from their initial positions specified by S_P^0/S_R^0 to the goal positions specified by S_P^+/S_R^+. A solution of length 7 is shown for the problem of pebble motion on a graph and a solution of length 5 is shown for the problem of multi-robot path planning. Notice the differences in parallelism between both solutions – multi-robot path planning allows the higher number of moves to be performed in parallel (in a single time step) thanks to weaker requirements on solutions.

Definition 2 *(problem of multi-robot path planning)*. Again, let us have an undirected graph $G=(V,E)$ but now instead of pebbles, a set of robots $R=\{r_1,r_2,\ldots,r_\mu\}$ where $\mu<|V|$ is given. The ***initial*** arrangement of the robots is defined by a simple function $S_R^0: R\rightarrow V$ (that is $S_R^0(r_i)\neq S_R^0(r_j)$ for $i,j=1,2,\ldots,\mu$ with $i\neq j$). The ***goal*** arrangement of the robots is defined by a simple function $S_R^+: R\rightarrow V$ (that is $S_R^+(r_i)\neq S_R^+(r_j)$ for $i,j=1,2,\ldots,\mu$ with $i\neq j$). The problem of ***multi-robot path planning*** is a task to find a number ζ and a sequence vertices $O_r=[o_1^r,o_2^r,\ldots,o_\zeta^r]$ for every robot $r\in R$ where $o_i^r\in V$ for $i=1,2,\ldots,\zeta$, $o_1^r=S_R^0(r)$, $o_k^r=S_R^+(r)$, and either $\{o_i^r,o_{i+1}^r\}\in E$ or $o_i^r=o_{i+1}^r$ for $i=1,2,\ldots,\zeta-1$. Furthermore, sequences of vertices $O_r=[o_1^r,o_2^r,\ldots,o_\zeta^r]$ and $O_s=[o_1^s,o_2^s,\ldots,o_\zeta^s]$ for every two robots $r\in R$ and $s\in R$ such that $r\neq s$ must satisfy that $o_i^r\neq o_i^s$ for $i=1,2,\ldots,\zeta$ (no two robots are simultaneously entering the same target vertex). □

2.2. Properties of the Defined Problems

Let us now summarize several basic properties of solutions of problems of pebble motion on graphs and multi-robot path planning.

Notice that a solution of the problem of pebble motion on a graph as well as a solution 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. Notice further that both problems intrinsically allow **parallel** movements of pebbles/robots. That is, more than one pebble/robot can move in a single time step. However, multi-robot path planning allows higher motion parallelism due to its weaker requirements (the target vertex is not required to be unoccupied in the previous time step before it is entered by another robot – see figure 1). To obtain a parallelism in the problem of pebble motion in a graph more than one unoccupied vertex is necessary. On the other hand, it is sufficient to have a single unoccupied vertex to obtain parallelism in the solution of multi-robot path planning (consider for example robots moving around a cycle).

It is not difficult to observe that a solution to an instance of the problem of pebble motion on a graph is also a solution to the corresponding multi-robot path planning problem. This fact is summarized in the following proposition.

Proposition 1 (pebble motion and multi-robot problem correspondence). Let us have a problem of pebble motion on a graph $G=(V,E)$, a set of pebbles $P=\{p_1, p_2, \ldots, p_\mu\}$, initial and goal positions of pebbles given by functions S_P^0, and S_P^+ respectively. The solution of this pebble motion problem $M=[M_{p_1}, M_{p_2}, \ldots, M_{p_\mu}]$ is also a solution of a problem of multi-robot path planning with the graph G, a set of robots $R=P$, and initial and goal positions of robots given by functions $S_R^0 = S_P^0$, and $S_R^+ = S_P^+$ respectively. ■

There is a variety of modifications of the defined problems. A natural additional requirement is to produce **shortest possible solutions** (that is, we require the numbers ξ or ζ respectively to be as small as possible). Unfortunately, this requirement makes the problem **intractable** (namely *NP*-hard; [3]) while without the requirement both problems are in the *P* class [2]. Nevertheless, we are usually concerned about the length of the solution in the real life situations. Taking into account the fact that existing fast sub-optimal methods [2], [7] generate too long solutions, we need some alternative sub-optimal solving method that would care about the quality of the generated solutions.

All the algorithms developed in the following sections are designed for the problem of pebble motion on a graph. Thanks to proposition 1, algorithms for pebble motion on a graph applies also for multi-robot path planning. The parallelism within the solution of the multi-robot path planning can be increased in a **post-processing step** using a *method of critical path* [4], [8].

3. A Special Case with Bi-connected Graph

A special case of the problem is addressed in this paper. A case where the graph modeling the environment is ***bi-connected*** and where there is only **one unoccupied** vertex is studied (that is, $\mu = |V|-1$). This class of problems is the most interesting since they are almost always solvable and allowing only one unoccupied vertex represents the most difficult setup.

3.1. Graph Theoretical Preliminaries

To preserve self-containment character of this paper, let us recall several graph theoretical notions [10] that represent foundations for algorithms presented further.

Definition 3 ***(graph connectivity).*** An undirected graph $G=(V,E)$ is *connected*, if $|V|\geq 2$ and for every pair of distinct vertices $u,v \in V$ there is a path connecting u and v consisting of edges from E. □

Definition 4 ***(graph bi-connectivity).*** An undirected graph $G=(V,E)$ is *bi-connected*, if $|V|\geq 3$ and the graph $G'=(V-\{v\}, E \cap \{\{u,w\} \mid u,w \in V \wedge u \neq v \wedge w \neq v\})$ is connected for every $v \in V$. □

Bi-connected graphs have an important well known property which we exploit further. Each bi-connected graph can be constructed starting with a cycle by a sequence of operations of *adding a loop (handle)* to the graph [9], [10]. Adding a loop which is a sequence of vertices $L=[u, x_1, x_2, ..., x_l, v]$ to an undirected graph $G=(V,E)$ where $u,v \in V$ with $u \neq v$ and $x_i \notin V$ for $i=1,2,\ldots,l$ (x_i are the new vertices) means to create a new graph $G'=(V',E')$; where $V'=V \cup \{x_1, x_2, \ldots, x_l\}$ and either $E'=E \cup \{\{u,v\}\}$ in the case when $l=0$ or $E'=E \cup \{\{u,x_1\},\{x_1,x_2\},\ldots,\{x_{l-1},x_l\}, \{x_l,v\}\}$ in the case when $l \geq 1$. As a preparation for the design of algorithms, the loop L is assigned a cycle $C(L)$ if the graph G is connected. The cycle $C(L)$ consists of vertices on a path between u and v in G followed by the vertices x_1, $x_2, \ldots, x_l$. Let us call the above construction sequence of a bi-connected graph a *loop decomposition*.

Lemma 1 (loop decomposition) **[9].** Any bi-connected graph $G=(V,E)$ can be obtained from a cycle by the **operation of adding a loop**. Moreover, the corresponding loop decomposition can be effectively found in the worst case time of $O(|V|+|E|)$ [9]. ■

3.2. Optimal Macros in Bi-connected Graphs

We are about to exploit a certain kind of **pattern database** [1] containing pre-calculated optimal solutions of sub-problems (macros). The structurally simplest almost always solvable sub-problem of the pebble motion problem consists of a so-called θ-like graph (see figure 2) where there is a single unoccupied vertex [7].

Definition 5 ***(θ-like graph).*** Let $A=\{x_1, x_2, \ldots, x_a\}$, $B=\{y_1, y_2, \ldots, y_b\}$, and $C=\{z_1, z_2, \ldots, z_c\}$ be a finite sets (of vertices) where $|A|\geq 1 \wedge |B|\geq 2 \wedge |C|\geq 1$. A *θ-like graph* $G_\theta(A,B,C)=(V_\theta, E_\theta)$ is an undirected graph where

$V_\theta = A \cup B \cup C$ and $E_\theta = \{\{x_1, x_2\}, \ldots, \{x_{a-1}, x_a\}, \{y_1, y_2\}, \ldots, \{y_{b-1}, y_b\}, \{z_1, z_2\}, \ldots, \{z_{c-1}, z_c\}, \{x_1, y_1\}, \{x_a, y_b\}, \{y_1, z_1\}, \{y_b, z_c\}\}$. □

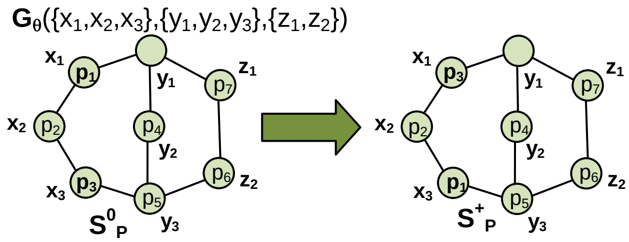


Figure 2: *An example of θ-like graph.* The task is to transpose pebbles p_1 and p_3.

The number of all the possible θ-like graphs grows polynomially with respect to the number of vertices (namely they are $O(|V_\theta|^3)$). However, the number of all the possible pebble motion problems on θ-like graphs grows exponentially with respect to the number of vertices (they are proportional to the number of permutations of the set of vertices). Hence, a restriction on the number of problems whose solution will be stored in the pattern database must be made. We need sub-problems from that a solution to the general problem can be composed. The following cases of problems satisfy both requirements. In the following text, we suppose (without loss of generality) that the unoccupied vertex in the initial and the goal arrangements of pebbles in θ-like graphs is the vertex y_1.

Definition 6 ***(transposition case).*** Let $G_\theta(A,B,C)$ be a θ-like graph and let $P = \{p_1, p_2, \ldots, p_\mu\}$ be a set of pebbles with $\mu = |V_\theta| - 1$. The pebble motion problem on a graph with the initial arrangement S_P^0 and the goal arrangement S_P^+ is called a *transposition case*, if there are pebbles $p, q \in P$ such that $p \neq q$ and $S_P^0(p) = S_P^+(q)$, $S_P^0(q) = S_P^+(p)$, and $(\forall r \in P)(r \neq p, q \Rightarrow S_P^0(r) = S_P^+(r))$ (see figure 2). □

Definition 7 ***(3-cycle rotation case).*** Let $G_\theta(A,B,C)$ be a θ-like graph and let $P = \{p_1, p_2, \ldots, p_\mu\}$ be a set of pebbles with $\mu = |V_\theta| - 1$. The pebble motion problem on a graph with the initial arrangement S_P^0 and the goal arrangement S_P^+ is called a *3-cycle rotation case*, if there are pebbles $p, q, s \in P$ such that p, q, s are pair-wise distinct and $S_P^0(p) = S_P^+(q)$, $S_P^0(q) = S_P^+(s)$, $S_P^0(s) = S_P^+(p)$, and $(\forall r \in P)(r \neq p, q, s \Rightarrow S_P^0(r) = S_P^+(r))$. □

Both, the number of transposition cases as well as the number of 3-cycle rotation cases, grows polynomially with respect to the number of vertices (they are $O(|V_\theta|^5)$ and $O(|V_\theta|^6)$ respectively). Thus it is realistic to store all the optimal solutions (macros) of the described cases up to the certain size of θ-like graphs in the pattern database.

The following two lemmas summarize usefulness of the transposition case and 3-cycle rotation case for solving the general problem.

Lemma 2 (solvability – transposition case) **[11].** A transposition case of the pebble motion problem on a θ-like graph $G_\theta(A,B,C)$ with $|A| \neq 2 \vee |B| \neq 3 \vee |C| \neq 2$ is solvable, if and only if G_θ contains a **cycle of** the **odd length**. A solution to **any problem** of pebble motion on a θ-like graph $G_\theta(A,B,C) = (V_\theta, E_\theta)$ can be composed of at most $|V_\theta| - 2$ solutions to **transposition cases** in the same graph. Moreover, a sequence of transposition cases whose solutions are necessary for producing the overall solution can be determined in the worst case time of $O(|V_\theta|)$. ■

The goal arrangement of robots S_P^+ in a θ-like graph $G_\theta(A,B,C) = (V_\theta, E_\theta)$ can be regarded as a permutation over $|V_\theta| - 1$ elements with respect to the initial arrangement S_P^0. S_P^+ represents an *even permutation* with respect to S_P^0, if it is reachable using the even number of solutions to transposition cases. Otherwise it represents an *odd permutation*.

Lemma 3 (solvability – 3-cycle case) **[2].** A 3-cycle rotation case of the problem of pebble motion on a θ-like graph $G_\theta(A,B,C)$ with $|A| \neq 2 \vee |B| \neq 3 \vee |C| \neq 2$ is **always solvable**. A solution to any pebble motion problem whose goal arrangement of pebbles S_P^+ represents an **even permutation** with respect to the initial arrangement S_P^0 in a θ-like graph $G_\theta(A,B,C) = (V_\theta, E_\theta)$ can be composed of at most $|V_\theta| - 2$ solutions to **3-cycle rotation case** in the same graph. Moreover, a sequence of 3-cycle rotation cases necessary for the task can be effectively determined in the worst case time of $O(|V_\theta|)$. ■

The exception of $G_\theta(A,B,C)$ with $|A| = 2 \wedge |B| = 3 \wedge |C| = 2$ can be solved separately. Due to small size of this exception, solutions to **all** the problems over this graph can be pre-calculated into the pattern database (that is, solutions for all permutations of pebbles can be stored).

At this point, we know how to solve the general pebble motion problem on a θ-like graph by composing its solution of macros for transposition and 3-cycle case. Let us now further generalize the approach for all the bi-connected graphs.

A covering of the given bi-connected graph with θ-like sub-graphs is the first step. That is, a set of θ-like graphs $\theta_1, \theta_2, \ldots, \theta_t$ such that $G = \bigcup_{i=1}^t \theta_i$ is needed. Let us call this covering a *θ-decomposition* of the bi-connected graph. If such θ-decomposition is available, then the remaining question is how to move robots to their target θ-like sub-graphs of the θ-decomposition. Goal positions of robots within θ-like sub-graphs can be then reached using macros from the database. The following lemmas justify the existence of θ-decomposition of the bi-connected graph.

Lemma 4 (two disjoint paths) **[10].** Let $G = (V, E)$ be a bi-connected graph and let $u, v \in V$ be two distinct vertices. There exist **two vertex disjoint paths** between u and v. Moreover, these two path can be effectively determined in the worst case time of $O(|V| + |E|)$. ■

Lemma 5 (θ-decomposition). Let $G = (V, E)$ be a bi-connected graph not being a single cycle. Then there **exists** a **θ-decomposition** $\theta_1, \theta_2, \ldots, \theta_t$ (θ_i is a θ-like graph for $i = 1, 2, \ldots, t$) such that $G = \bigcup_{i=1}^t \theta_i$. Moreover, the θ-decomposition of the graph can be effectively found in $O(|V| + |E|)$. ■

Proof. From lemma 1, we know that there exists a loop decomposition of the bi-connected graph G. Consider the last loop $L = [u, x_1, x_2, ..., x_l, v]$ of the loop decomposition. The graph G without the loop L is again a bi-connected graph, let us denote it G^-. Using lemma 4, there exist two vertex disjoint paths π, ψ connecting u and v in G^-. Now $G_\theta(\pi, L, \psi)$ is a θ-like graph. From the induction supposition there exists a θ-decomposition of G^-. Together with $G_\theta(\pi, L, \psi)$ it is a θ-decomposition of G. ■

4. Solving Algorithms for Bi-connected Case

Two algorithms for solving pebble motion problems on a bi-connected graph $G = (V, E)$ with a single unoccupied vertex ($\mu = |V| - 1$) are presented below. Both algorithms assume that a loop decomposition of the graph G was constructed. That is, we have a cycle C_0 and a sequence of loops $L_1, L_2, \ldots, L_t$ such that the graph G can be constructed from C_0 by adding loops $L_1, L_2, \ldots, L_t$ incrementally. If the graph G contains a cycle of odd length, C_0 is also supposed to be of odd length. Since the construction of the graph G starts with a cycle C_0 (which is a connected graph) $C(L_i)$ is defined for every $i = 1, 2, \ldots, t$. Specially, we define $C(C_0) = C_0$.

To reduce the complexity of the pseudo-code of algorithms we assume the unoccupied vertex of the goal situation S_P^+ to be in the cycle C_0 (that is, $(v \in V \wedge (\forall p \in P) S_P^+(p) \neq v) \Rightarrow v \in C_0$). Overcoming this assumption is discussed in the next section.

Except the functions S_P^0 and S_P^+ we further have a function $S_P : P \to V$ expressing current positions of pebbles. Next, we have functions $\Phi_P^0 : V \to P \cup \{\perp\}$, $\Phi_P^+ : V \to P \cup \{\perp\}$, and $\Phi_P : V \to P \cup \{\perp\}$ which are generalized inverses of S_P^0, S_P^+, and S_P respectively; the symbol $\perp$ stands for unoccupied vertex (that is, $(\forall p \in P) \Phi_P(S_P(p)) = p$; $\Phi_P(v) = \perp$ if $(\forall p \in P) S_P(p) \neq v$). Next, we assume that we have a sequence of potentially infinite sequences representing the solution of the problem $[M_{p_1}, M_{p_2}, \ldots, M_{p_\mu}]$.

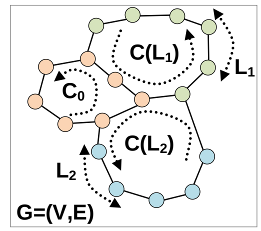
4.1. An Algorithm Based on θ-decomposition

In this section, we describe an improvement of the solving algorithm from [2] (called *MIT*). The new algorithm exploits θ-decomposition of the given bi-connected graph. The improvement consists in replacing the solving process of 3-cycle case that originally exploits 3-transitivity of θ-like sub-graphs by the use of macros. The resulting algorithm is called *MIT-θ* and it is formalized below using the pseudo-code as algorithm 1. The solving algorithm itself is represented by the function *MIT-θ*-Solve accompanied with several auxiliary functions. The next important procedure *θ-BOX*-Solve represents the solving process within θ-like graphs using pre-calculated optimal macros from the pattern database.

The solving algorithm proceeds inductively according to the pre-calculated loop decomposition $L_1, L_2, \ldots, L_t$ (lines 2-4 of *MIT-θ*-Solve). The pebbles are placed to their goal positions in loops starting with the last loop L_t and continuing to the original cycle with the loop (C_0, L_1 - original θ-like graph; lines 5-8 of *MIT-θ*-Solve). Having a loop L_c of the loop decomposition, a corresponding θ-like sub-graph is considered (lemma 5; lines 1-5 of SolveRegular-θ). All the pebbles whose goal positions are within the loop are placed. Two cases are distinguished. If the pebble to be placed is already within the θ-like sub-graph, then macro is used to place it to the right position (lines 14-19 of SolveRegular-θ). If the pebble is outside the θ-like sub-graph, then it must be first moved to into the θ-like sub-graph before the macro can be applied (lines 7-13 of SolveRegular-θ).

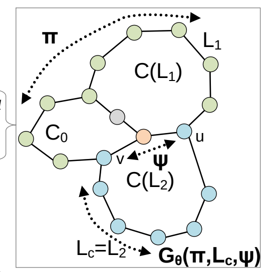
Algorithm 1. *The MIT-θ algorithm.* The algorithm solves a given pebble motion problem on a bi-connected graph modeling the environment with a single unoccupied vertex.

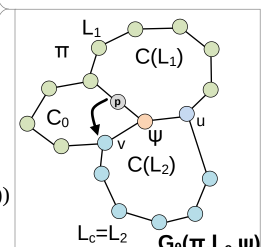
function *MIT-θ*-Solve (G, S_P^0, S_P^+) : **pair**
1: $\zeta \leftarrow 0$; $S_P \leftarrow S_P^0$
2: **for** $c = t, t-1, \ldots, 2$ **do**
3: | **if** $|L_c| > 2$ **then**
4: | | SolveRegular-$\theta(c)$
5: let $[u, x_1, x_2, \ldots, x_l, v] = L_1$
6: **let** π, ψ be two disjoint paths between
7: | u and v in C_0
8: *θ-BOX*-Solve $(G_\theta(\pi, L_1, \psi), S_P, S_P^+)$
9: **return** $(\zeta, [M_{r_1}, M_{r_2}, \ldots, M_{r_\mu}])$



Loop decomposition of a graph G

procedure SolveRegular-θ (c)
1: **let** $[u, x_1, x_2, \ldots, x_l, v] = L_c$
2: lock (L_c) ; unlock $(\{u, v\})$
3: **let** π, ψ be two disjoint paths between u
4: | and v not containing locked vertices
5: **let** $(V_\theta, E_\theta) = G_\theta(\pi, L_c, \psi)$
6: **for** $i = 1, 2, \ldots, l$ **do**
7: | **if** $S_P(\Phi_P^+(x_i)) \notin V_\theta$ **then**
8: | | lock (L_c) ; unlock $(\{u, v\})$
9: | | MovePebble $(\Phi_P^+(x_i), v)$
10: | | MoveUnoccupied (u)
11: | | unlock (L_c)
12: | | $S_\theta^+ \leftarrow S_P$; $S_\theta^+(\Phi_P^+(v)) = S_P^+(\Phi_P^+(v))$
13: | | *θ-BOX*-Solve $(G_\theta(\pi, L_1, \psi), S_P, S_\theta^+)$
14: | **else**
15: | | lock (L_c) ; unlock $(\{u, v\})$
16: | | MoveUnoccupied (u)
17: | | unlock (L_c)
18: | | $S_\theta^+ \leftarrow S_P$; $S_\theta^+(\Phi_P^+(x_i)) = S_P^+(\Phi_P^+(x_i))$
19: | | *θ-BOX*-Solve $(G_\theta(\pi, L_1, \psi), S_P, S_\theta^+)$
20: lock (L_c) ; unlock $(\{u, v\})$

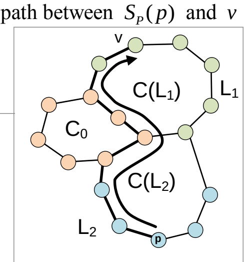




Pebble $p = \Phi_P^+(x_i)$ outside G_θ; move p to v

procedure MoveUnoccupied (v)
1: **let** $x \in V$ such that $\Phi_P(x) = \perp$ and x is not locked
2: **let** $[x = k_1, k_2, \ldots, k_j = u]$ be a shortest path between
3: | x and v in G not containing locked vertices
4: **for** $i = 1, 2, \ldots, j-1$ **do**
5: | SwapPebblesUnoccupied (k_{i+1}, k_i)

procedure MovePebble (p, v)
1: **let** $[S_P(p) = k_1, k_2, \ldots, k_j = v]$ be a shortest path between $S_P(p)$ and v
2: | in G not containing locked vertices
3: **for** $i = 1, 2, \ldots, j-1$ **do**
4: | lock $(\{k_i\})$
5: | MoveUnoccupied (k_{i+1})
6: | unlock $(\{k_i\})$
7: | SwapPebbleUnoccupied (k_i, k_{i+1})

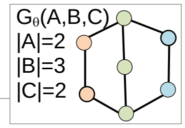


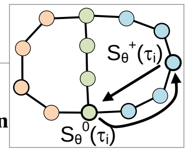
Pebble p is moved to v by rotating cycle $C(L_2)$, C_0, and $C(L_1)$

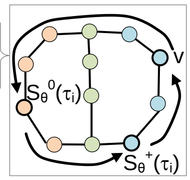
procedure SwapPebblesUnoccupied (u, v)
1: $S_P(\Phi_P(u)) = v$; $p = \Phi_P(u)$
2: $\Phi_P(u) = \perp$; $\Phi_P(v) = p$
3: **for** $i = 1, 2, \ldots, \mu$ **do**
4: | $m_\zeta^{p_i} = S_P(p_i)$
5: $\zeta \leftarrow \zeta + 1$

procedure θ-*BOX*-Solve $(G_\theta(A,B,C),S_\theta^0,S_\theta^+)$
1: **let** $(V_\theta,E_\theta)=G_\theta(A,B,C)$
2: **let** $\{\tau_1,\tau_2,\ldots,\tau_{|V_\theta|-1}\}=\{\tau \mid S_\theta^0(\tau)\in V_\theta\}$
3: **if** $|A|=2\wedge|B|=3\wedge|C|=2$ **then**
4: | ApplyMacro ($table^{232}[S_\theta^0,S_\theta^+]$)
5: **else**
6: | $S_\theta \leftarrow S_\theta^0$
7: | **if** G_θ contains an odd cycle **then**
8: | | **for** $i=1,2,\ldots,|V_\theta|-2$ **do**
9: | | | **if** $S_\theta(\tau_i)\neq S_\theta^+(\tau_i)$ **then**
10: | | | | ApplyMacro ($table_T^{G_\theta}[S_\theta(\tau_i),S_\theta^+(\tau_i)]$)
11: | **else** { G_θ does not contain any odd cycle}
12: | | **if** S_θ^+ gives an odd permutation w.r.t. S_θ **then**
13: | | | **fail** {the problem is unsolvable}
14: | | **else** { S_θ^+ gives an even permutation w.r.t. S_θ }
15: | | | **for** $i=1,2,\ldots,|V_\theta|-2$ **do**
16: | | | | **if** $S_\theta(\tau_i)\neq S_\theta^+(\tau_i)$ **then**
17: | | | | | **let** $v\neq S_\theta(\tau_i),S_\theta^+(\tau_1),S_\theta^+(\tau_2),\ldots,S_\theta^+(\tau_i)$
18: | | | | | ApplyMacro ($table_3^{G_\theta}[S_\theta(\tau_i),S_\theta^+(\tau_i),v]$)

procedure ApplyMacro (σ)
1: **let** $[(u_1,v_1),(u_2,v_2)\ldots,(u_k,v_k)]=\sigma$
2: **for** $i=1,2,\ldots,k$ **do**
3: | SwapPebblesUnoccupied (u_i,v_i)
4: | $S_\theta(\Phi_P(u))=v$







The original cycle C_0 with its loop L_1 is solved solely using macros (lines 5-8 of *MIT-θ*-Solve), since all the pebbles whose goal positions are within the original θ-like sub-graph are already there.

Without proof, let us summarize properties of the algorithm. The *MIT-θ* algorithm is **sound** and **complete**. The worst case time complexity is of $O(|V|^5)$.

4.2. An Algorithm Using Loop Decomposition

The second algorithm for solving pebble motion problems on bi-connected graphs is called *BIBOX-θ*. It is a modification of the algorithm from [6] (the original algorithm is called *BIBOX*) where the last phase of the algorithm placing the pebbles in the original cycle C_0 is replaced by solving process over the corresponding θ-like sub-graph based on macros. One of the main contributions of the new approach is that now only **one** unoccupied vertex is needed while the original version of the algorithm requires at least **two** unoccupied vertices.

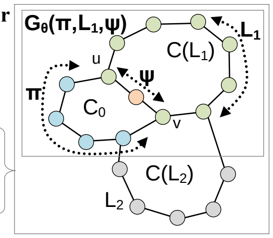
For easier expressing of the algorithm we have auxiliary functions $next/V(C,v)$, $prev/V(C,v)$ that return the next or the previous vertex in the given cycle with respect to a fixed orientation of the cycle. The solving algorithm itself is presented here using pseudo-code as algorithm 2.

The algorithm proceeds from the last loop to the first loop of the loop decomposition. This process is very similar to the corresponding process within the *MIT-θ* algorithm. The main difference rests in a way how the pebbles are placed within a loop. Within a loop, pebbles are placed to their goal positions in the stack manner (that is, a new pebble comes at the beginning of the loop and the loop is rotated - stack pushes). The last rotation of the loop places the pebbles to their destinations. When placing a pebble within the loop it is necessary to distinguish between the situation when the pebble is outside the loop (lines 3-8 of SolveRegularCycle) and the situation when the pebble is already within the current loop (lines 10-29 of SolveRegularCycle).

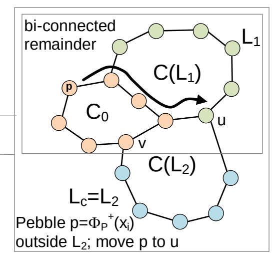
Again without proof, let us summarize properties of the algorithm. The *BIBOX-θ* algorithm is **sound** and **complete**. The worst case time complexity of the algorithm is $O(|V|^4)$.

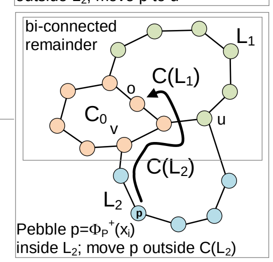
Algorithm 2. *The BIBOX-θ algorithm.* The algorithm solves a given pebble motion problem on a bi-connected graph modeling the environment with a single unoccupied vertex.

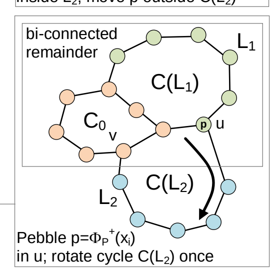
function *BIBOX-θ*-Solve (G,S_P^0,S_P^+) : **pair**
1: $\zeta\leftarrow 0$; $S_P\leftarrow S_P^0$
2: **for** $c=t,t-1,\ldots,2$ **do**
3: | **if** $|L_c|>2$ **then**
4: | | SolveRegularCycle (c)
5: **let** $[u,x_1,x_2,\ldots,x_l,v]=L_1$
6: **let** π,ψ be two disjoint paths between
7: | u and v in C_0
8: θ-*BOX*-Solve $(G_\theta(\pi,L_1,\psi),S_P,S_P^+)$
9: **return** $(\zeta,[M_{p_1},M_{p_2},\ldots,M_{p_\mu}])$



procedure SolveRegularCycle (c)
1: **let** $[u,x_1,x_2,\ldots,x_l,v]=L_c$
2: **for** $i=1,2,\ldots,l$ **do**
3: | **if** $S_P(\Phi_P^+(x_i))\notin L_c$ **then**
4: | | lock (L_c) ; unlock ($\{u,v\}$)
5: | | MovePebble ($\Phi_P^+(x_i),u$)
6: | | MoveUnoccupied (v)
7: | | unlock (L_c)
8: | | RotateCycle$^+$ ($C(L_c)$)
9: | **else**
10: | | lock (L_c) ; unlock ($\{u,v\}$)
11: | | MoveUnoccupied (u)
12: | | unlock (L_c)
13: | | $\rho\leftarrow 0$
14: | | **while** $S_P(\Phi_P^+(x_i)))\neq v$ **do**
15: | | | RotateCycle$^+$ ($C(L_c)$)
16: | | | $\rho\leftarrow\rho+1$
17: | | lock (L_c) ; unlock ($\{u,v\}$)
18: | | **let** $o\in V-(\bigcup_{i=c}^{k}L_i\cup C(L_c))$
19: | | MovePebble ($\Phi_P^+(x_i),o$)
20: | | lock ($\{o\}$)
21: | | MoveUnoccupied (u)
22: | | unlock (L_c)
23: | | **while** $\rho>0$ **do**
24: | | | RotateCycle$^-$ ($C(L_c)$)
25: | | | $\rho\leftarrow\rho-1$
26: | | unlock ($\{o\}$)
27: | | MovePebble ($\Phi_P^+(x_i),u$)
28: | | MoveUnoccupied (v)
29: | | RotateCycle$^+$ (L_c)
30: | lock (L_c) ; unlock ($\{u,v\}$)







procedure RotateCycle$^+$ (C)
1: **let** $x\in C$ such that $\Phi_P(x)=\perp$ and x is not locked
2: **for** $i=1,2,\ldots,|C|$ **do**
3: | SwapPebblesUnoccupied ($prev/V(C,x),x$)
4: | $x\leftarrow prev/V(C,x)$

4.3. Extensions and the Real Implementation

The presented pseudo-codes of the *MIT-θ* and the *BIBOX-θ* algorithms require a special assumption that the finally unoccupied vertex must be in the original cycle. To overcome this assumption we need to modify the required solution given by the function S_P^+ so that unoccupied vertex is moved to the original cycle along a path π. After solving the problem by the algorithm the unoccupied vertex is moved back along the path π which finishes the solution of the original unmodified problem.

When a pebble is moved from one vertex to another the shortest path between the original position and the target vertex is always used. Moreover, having more than one unoccupied vertex, the nearest unoccupied vertex to the place where it is needed is always used. Both heuristics reduces the number of moves in the solution.

If the required record is not in the pattern database, then the algorithm should switch to solving method based on 3-transitivity from [2].

4.4. Solving Multi-robot Path Planning Problems

Having a solving algorithm for the pebble motion problem on a graph, it is easy to solve the corresponding multi-robot path planning problem. We can just proclaim the solution of the pebble motion problem to be a solution of the corresponding multi-robot path planning problem. However, this may waste the potential parallelism.

The more sophisticated approach is to utilize the relaxed requirements on the solution in the multi-robot path planning problem to increase parallelism. The method of choice here is *critical path* [4]. An anti-symmetric relation of *dependence between motions* of pebbles can be defined. Two motions are *dependent* if one must precede the other in the solution (for example two motions of the same pebble are dependent).

More formally, a move $q: v_1 \rightarrow v_2$ with $v_1 \neq v_2$ (a robot q is moved from a vertex v_1 to a vertex v_2) generated at time step t_q is *trivially dependent* on a move $r: u_1 \rightarrow u_2$ with $u_1 \neq u_2$ generated at time t_r where $t_r < t_q$ if $r = q$ or $|\{u_1, u_2\} \cap \{v_1, v_2\}| \geq 1$ while $u_1 = v_2 \vee u_2 \neq v_1$ (the second constraint is in fact a negation of $u_1 \neq v_2 \wedge u_2 = v_1$). The relation of dependence between motions is the transitive closure of the relation of trivial dependence. Since the (anti-symmetric) relation of trivial dependence induces a directed acyclic graph on the set of moves generated by the solving algorithm it is easy to calculate the corresponding transitive closure. The method of critical path can be used in this case to calculate earliest time step for each move when it can be executed.

5. Experimental Evaluation

The presented algorithms - *MIT-θ* and *BIBOX-θ* for pebble motion on graph as well as its competitors - were implemented in C++ and an experimental evaluation was made. The experimental evaluation was made on a machine with Pentium 4 2.4 GHz with 512Mb of memory under Mandriva Linux 10.1. Source code and additional data for reproducing all the experiments are available at: http://ktiml.mff.cuni.cz/~surynek/research/ictai2009/. The comparison was concentrated on the length of solutions and on the solving runtime. The results are presented in figure 3 and 4.

The tests were made on random instances of problems of pebble motion on bi-connected graphs where the number of vertices ranged from 13 to 48. The number of loops of the loop decomposition ranged from 3 to 16. The length of loops of the decomposition had the random length with the uniform distribution in the interval of $1 \ldots 8$. All the problems had a single unoccupied vertex placed randomly (generally not in the original cycle).

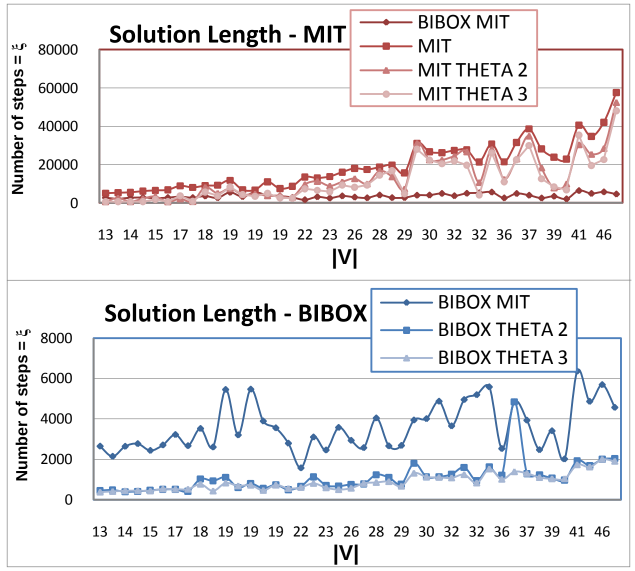


Figure 3. *Solution length comparison.* Six variants of solving algorithms are compared – *BIBOX* where the solving process for original cycle with loop is based on 3-transitivity (BIBOX MIT), the original *MIT* algorithm, *MIT-θ* where transposition cases are preferably used (MIT THETA 2), *MIT-θ* where 3-cycle rotations are preferably used (MIT THETA 3), *BIBOX-θ* where transposition cases are preferably used (BIBOX THETA 2), and *BIBOX-θ* where 3-cycle rotations are preferably used (BIBOX THETA 3).

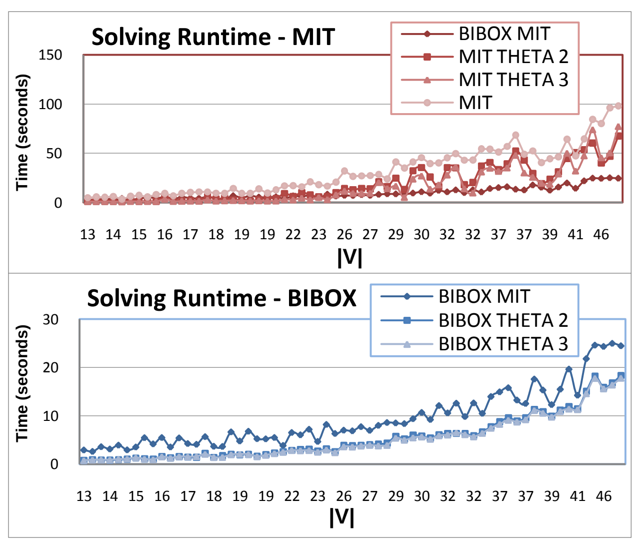


Figure 4. *Solving runtime comparison.* Six variants of solving algorithms are compared – see figure 3. Each problem was solved 1000 times to accumulate measurable time.

In all the tests, necessary optimal macros were found in the database (that is, the alternative method based on 3-transitivity was not used). The results show that replacement of the method based on 3-transitivity with optimal macros brings significant improvement in solution length and solving time of both algorithms. Moreover, the experiments show that all the variants of the *BIBOX* algorithm outperform the *MIT* algorithm significantly. It is also evident that the preference of 3-cycle rotation cases is slightly better than the preference of transposition case with respect to the solution length. However, notice that storing transposition cases in the pattern database is less space consuming.

Additional experiments were devoted to evaluation of parallelism reached by the method of critical path as described above. Generally, the comparison of lengths of solutions is relatively the same as in figures 3 and 4 (that is, the algorithm *BIBOX-θ* is the best again). However, the absolute lengths of solutions are shorter approximately by the factor corresponding to the diameter of the tested graphs (which was 4…10 in the above experiments; so the absolute values of lengths of solutions are about 4…10 lower). Space limitations do not allow us to present these experiments in the form of graphs.

6. Related Works and Conclusions

This work is significantly influenced by [5]. The author presents a solving method for the multi-robot path planning based on a decomposition of the environment into simpler sub-graphs that are easier to tackle. This approach has much in common with the approach presented above. However, deep theoretical results gained for pebble motion on graphs (sliding box puzzles) [2], [3], and [11] are ignored in [5], though they are so closely related to multi-robot path planning.

The major aim of this paper is to fill in the gap between theory and practical solving of problems of pebble motion on graph and multi-robot path planning. We have to emphasize that this paper intensively builds on existing works while we improve aspect regarding the optimality of solutions.

Let us further comment the related works. Graphical properties crucial for tackling the problem were identified in [11]. The solving methods for transposition and 3-cycle rotation cases were developed in [7]. The less general version of the *BIBOX* algorithm is presented in [6]. The variant of the algorithm in [6] requires at least **two** unoccupied vertices in a bi-connected graph. A comparison with domain-independent planners and scaling evaluation is also given in [6]. *LPG-td* and *SGPlan* were tested against the *BIBOX* algorithm; only extremely small pebble motion/multi-robot problems are solvable by domain-independent planners. These results render the domain-independent approach to be uncompetitive.

Our work can be summarized as follows. A successful application of optimal pre-calculated macros for solving problems of pebble motion and multi-robot path planning with bi-connected environments is presented in this paper. One existing algorithm (*MIT*) was improved by the integration of macros. Another algorithm (*BIBOX*) was improved and generalized – the new variant is called ***BIBOX-θ*** - so that is becomes the **state-of-the-art algorithm** (it is better than existing domain-dependent algorithms as well as domain independent planners) for solving the studied class of problem in terms of runtime and the quality of solutions.

For future work we plan to evaluate post-processing techniques from [8] which are designed for improving solutions in term length and parallelism.

7. References

[1] Felner, R. E. Korf, R. Meshulam, and R.C. Holte, “Compressed Pattern Databases”, *Journal of Artificial Intelligence Research* (JAIR), Volume 30, AAAI Press, 2007, pp. 213-247.

[2] 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), IEEE Press, 1984, pp. 241-250.

[3] 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), Morgan Kaufmann Publishers, 1986, pp. 168-172.

[4] S. Russell and P. Norvig, “Artificial Intelligence: A Modern Approach (second edition)”, Prentice Hall, 2003.

[5] M. R. K. Ryan, “Graph Decomposition for Efficient Multi-Robot Path Planning”, *Proceedings of the 20th International Joint Conference on Artificial Intelligence* (IJCAI 2007), IJCAI Conference, 2007, pp. 2003-2008.

[6] 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), IEEE Press, 2009, pp. 3613-3619.

[7] 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), AAAI Press, 2009.

[8] 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), CA, USA, USC, http://www.search-conference.org/index.php/Main/SOCS09, July 2009.

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

[10] D. B. West, “Introduction to Graph Theory, second edition”, Prentice-Hall, 2000.

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