

ON PEBBLE MOTION ON GRAPHS AND ABSTRACT MULTI-ROBOT PATH PLANNING

Pavel Surynek

presenting: **Daniel Toropila**

Charles University in Prague

Faculty of Mathematics and Physics

The Czech Republic

PEBBLE MOTION ON GRAPHS

- **Input:** An undirected graph $G=(V,E)$ and a set of pebbles $P=\{p_1,p_2,\dots,p_\mu\}$, where $\mu < |V|$
- **Rules:**
 - **each pebble** is placed in a vertex (at most one pebble in a vertex)
 - a **pebble can be moved into an unoccupied** vertex through an edge (no other pebble is allowed to enter same the vertex)
 - **initial positions** of pebbles...simple function $S_P^0: P \rightarrow V$
 - **goal positions** of pebbles... simple function $S_P^+: P \rightarrow V$
- **Task:** Find a **sequence of allowed moves** for pebbles such that all the pebbles are relocated from their initial positions to the given goal positions



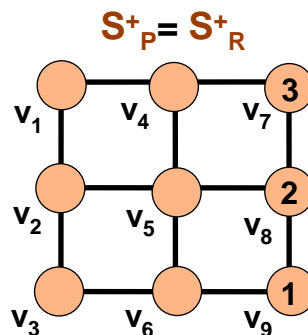
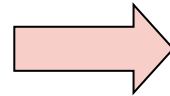
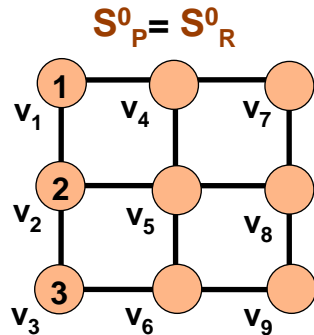
PATH PLANNING FOR MULTIPLE ROBOTS

- **Input:** An undirected graph $G=(V,E)$ and a set of robots $R=\{r_1,r_2,\dots,r_\mu\}$, where $\mu < |V|$
- **Rules:**
 - **each robot is placed in a vertex** (at most one robot in a vertex)
 - **a robot can move through an edge into an unoccupied vertex or into a vertex that is being left** (no other robot is allowed to enter the same vertex)
 - **initial positions** of robots ... simple function $S_R^0: R \rightarrow V$
 - **goal positions** of robots ... simple function $S_R^+: R \rightarrow V$
- **Task: Find a sequence of allowed moves** for robots such that all the robots reach their goal positions starting from the given initial positions
- A solution to the problem of pebble motions on a graph is also a solution to the corresponding multi-robot path planning problem



EXAMPLE OF PEBBLE MOTION ON GRAPHS AND MULTI-ROBOT PATH PLANNING

- Initial positions of pebbles/robots given by S_P^0 / S_R^0
- Goal positions of pebbles/robots given by S_P^+ / S_R^+



Solution of Pebble Motion Problem with $P=\{1,2,3\}$

length=7

M_1	$[v_1, v_4, v_7, v_8, v_9, v_9, v_9]$
M_2	$[v_2, v_2, v_1, v_4, v_7, v_8, v_8]$
M_3	$[v_3, v_3, v_3, v_2, v_1, v_4, v_7]$

Step: 1 2 3 4 5 6 7

Solution of Multi-robot Path Planning Problem with $R=\{1,2,3\}$

length=5

O_1	$[v_1, v_4, v_7, v_8, v_9]$
O_2	$[v_2, v_1, v_4, v_7, v_8]$
O_3	$[v_3, v_2, v_1, v_4, v_7]$

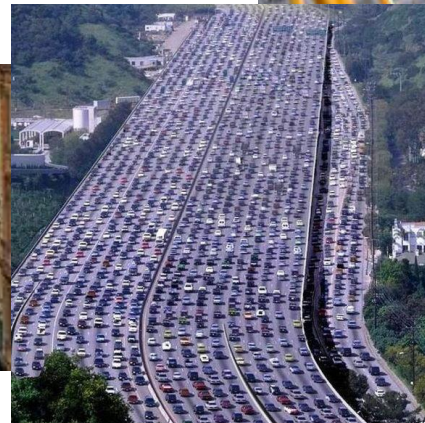
Step: 1 2 3 4 5

- M_p is a sequence of positions of the pebble p in all the discrete time steps
- O_r is a sequence of positions of the robot r in all the discrete time steps
- Notice the **parallelism** within the solutions (multi-robot path planning allows higher parallelism)
- Short solutions** are preferred (a decision problem whether there is a solution no longer than the given length is **NP-complete**)



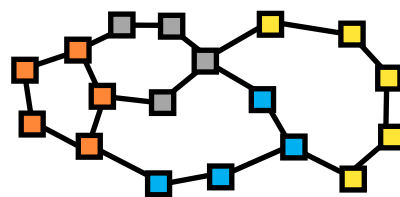
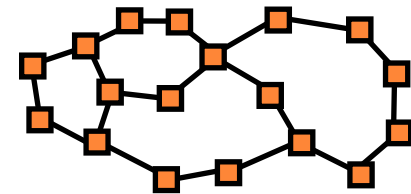
MOTIVATION FOR THE PROBLEM

- rearranging containers (robot = **container**)
- heavy traffic control (robot = **car**)
- data transfer planning (robot = **data packet**)
- lift transportation planning in future buildings (robot = **lift**)



A CASE WITH A BI-CONNECTED GRAPH

- Pebble motion problems with **bi-connected graphs** are most important for practice
- **Single unoccupied** vertex is supposed (represents **the most difficult case**)
- Almost **all the goal arrangements** of pebbles are reachable using allowed moves in bi-connected graphs
- An undirected graph $G=(V,E)$ is **bi-connected** if and only if $|V| \geq 3$ and $\forall v \in V$ the graph $G=(V-\{v\}, E')$ where $E'=\{\{x,y\} \in E \mid x,y \neq v\}$ is connected
- **Property:** Every bi-connected graph can be constructed from a cycle by consecutive adding of loops \rightarrow **loop decomposition**

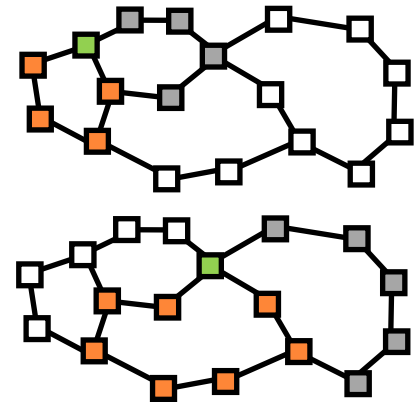


- Original cycle
- 1st loop
- 2nd loop
- 3rd loop



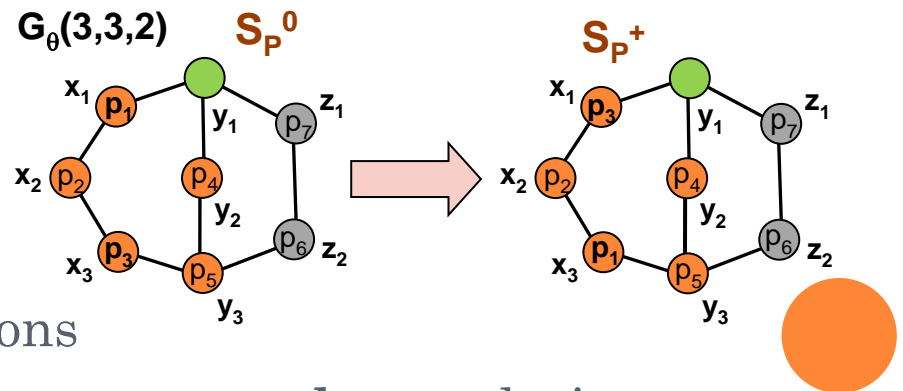
A NOTION OF θ -LIKE GRAPH

- A **θ -like graph** is the simplest non-trivial bi-connected graph for which the problem is solvable
- θ -like graph $G_\theta(\mathbf{a}, \mathbf{b}, \mathbf{c})$ is specified by three parameters \mathbf{a} , \mathbf{b} , \mathbf{c} where:
 - \mathbf{a} is the size of the left loop ... vertices $\{x_1, x_2, \dots, x_a\}$
 - \mathbf{b} is the size of the middle bone ... vertices $\{y_1, y_2, \dots, y_b\}$
 - \mathbf{c} is the size of the right loop ... vertices $\{z_1, z_2, \dots, z_c\}$
 - the vertex y_1 is preserved **unoccupied**



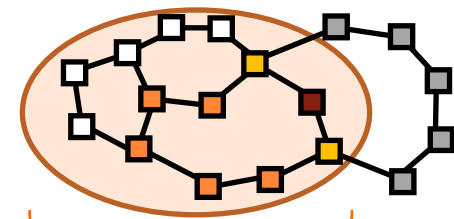
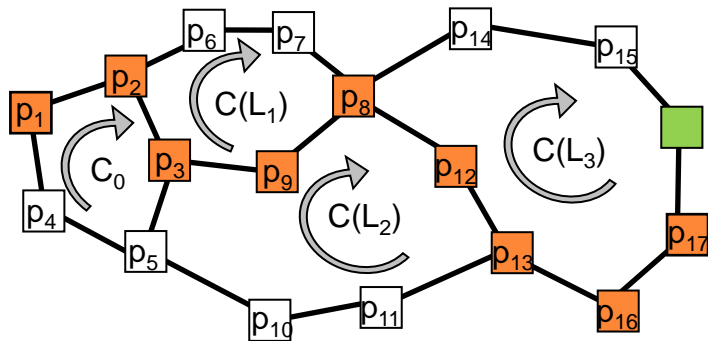
examples of θ -like sub-graphs in a bi-connected graph

- Can be solved using **3-transitivity** of bi-connected graphs (any three pebbles can be moved to any three vertices)
- 3-transitivity induces **3-cycles**, 3-cycles induce **even** permutations
- However, the use of 3-transitivity generates **long** solutions



ORIGINAL ALGORITHM BASED ON 3-TRANSITIVITY (MIT ALGORITHM)

- **Proposition:** Any two distinct vertices in a bi-connected graph are connected by **two vertex disjoint paths**.
- **θ -decomposition** = decomposition of the bi-connected graph into θ -like sub-graphs (sub-graphs are not disjoint)
 - constructed from **loop decomposition**
 - a **θ -like sub-graph** is constructed for each loop of the loop decomposition
 - the θ -like sub-graph **consists** of the **loop** and **two disjoint paths** between vertices where the loop is connected to the bi-connected remainder



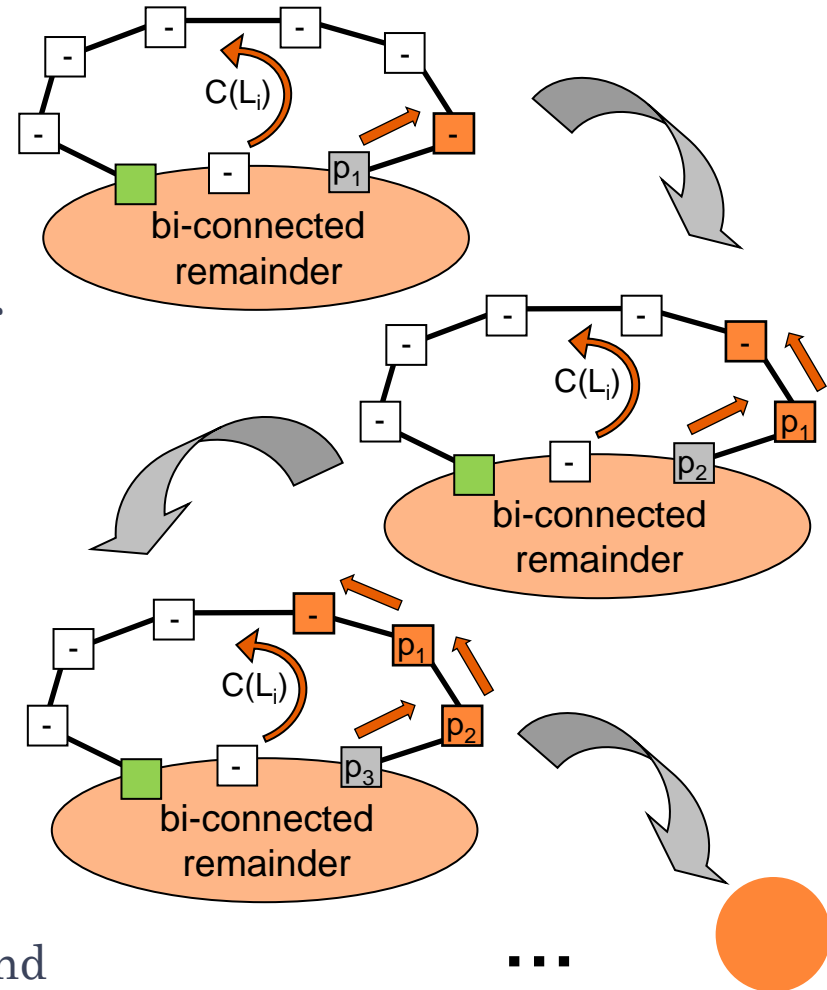
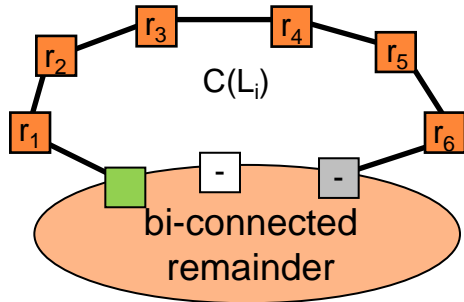
bi-connected remainder

- **MIT Algorithm:**
 - move a pebble into the θ -like sub-graph
 - use **3-transitivity** to place pebble to its final position within θ -like sub-graph

- loop
- connection vertices
- first path
- second path



A NEW ALGORITHM BASED ON LOOP DECOMPOSITION (BIBOX ALGORITHM)



- Pebbles are placed in stack like manner into the loop
 - The next pebble is moved into the **gray** connection vertex
 - Two cases
 - the pebble is somewhere in the loop → must be rotated out of the loop first
 - the pebble is outside the loop
 - Then the pebble is rotated into the loop (using **green** unoccupied vertex)
- The final rotation places all the pebbles to their goal positions
- **Not applicable** for the original cycle and the first loop (**original θ -like sub-graph** – 3-transitivity is used)

θ -LIKE GRAPHS AND PATTERN DATABASE WITH OPTIMAL SOLUTIONS

- Interpret arrangement of pebbles in a θ -like graph as a **permutation**
- **Proposition 1:**
 - Any **permutation** over μ elements can be obtained as a composition of at most $\mu-1$ **transpositions**.
- **Proposition 2:**
 - Any **even permutation** over μ elements can be obtained as a composition of at most $\mu-1$ **rotations along a triple (3-cycle)**.
- **Proposition 3:**
 - Rotation along a **triple** is always solvable in a θ -like graph; **transposition** is solvable, if the θ -like graph contains an **odd cycle**.
- **Transposition and 3-cycle rotation** case are good candidates to be stored in a pattern database (**optimal solutions** to these cases are stored)
 - **polynomial number** of records in the database ($O(|V|^5) + O(|V|^6)$)
 - they completely solve every pebble motion problem on a θ -like graph

APPLICATION OF PATTERN DATABASE WITHIN **MIT** AND **BIBOX** ALGORITHMS

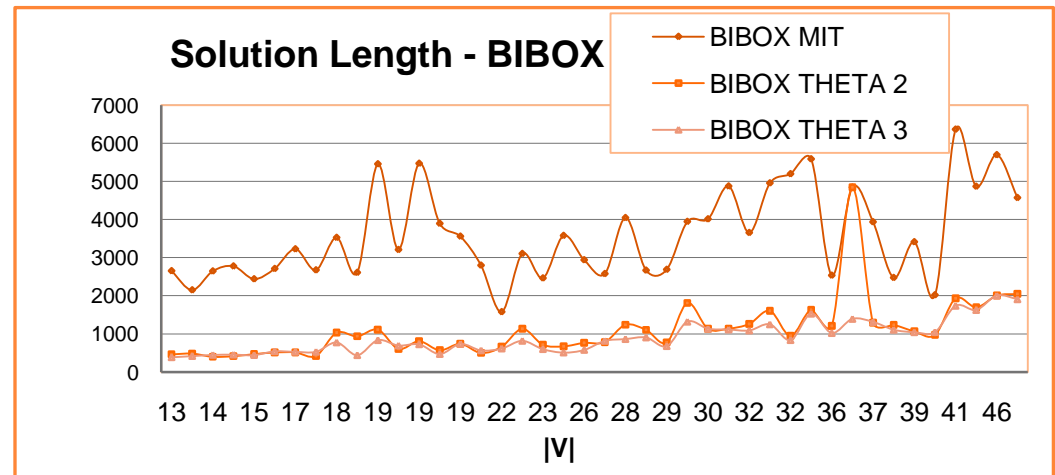
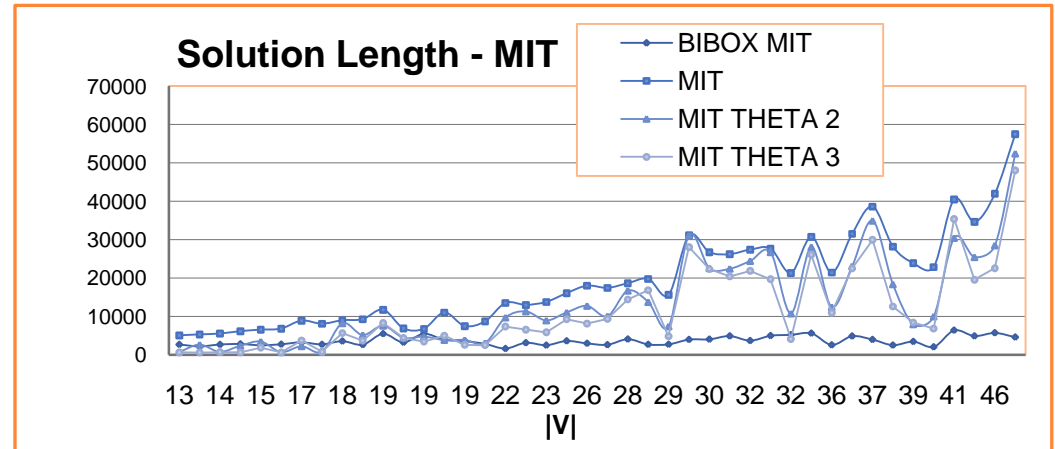
- Optimal macros (solutions) from the pattern database are used to compose a sub-optimal solution of a given situation
- **MIT Algorithm:**
 - Placing pebbles within the target θ -like sub-graph based on **3-transitivity** is replaced by **optimal macros** → **MIT- θ algorithm**
- **BIBOX Algorithm:**
 - Placing pebbles within the original θ -like sub-graph (**original cycle+first loop** of the loop decomposition) based on **3-transitivity** is replaced by **optimal macros** → **BIBOX- θ algorithm**
- All the variants of algorithms – that is **MIT**, **MIT- θ** , **BIBOX**, and **BIBOX- θ** :
 - have worst case time complexity of $O(|V|^3)$
 - generate solutions of the length $O(|V|^3)$
- Theoretically same, however MIT and MIT- θ have higher constants in the asymptotic time and solution length estimation



EXPERIMENTAL EVALUATION

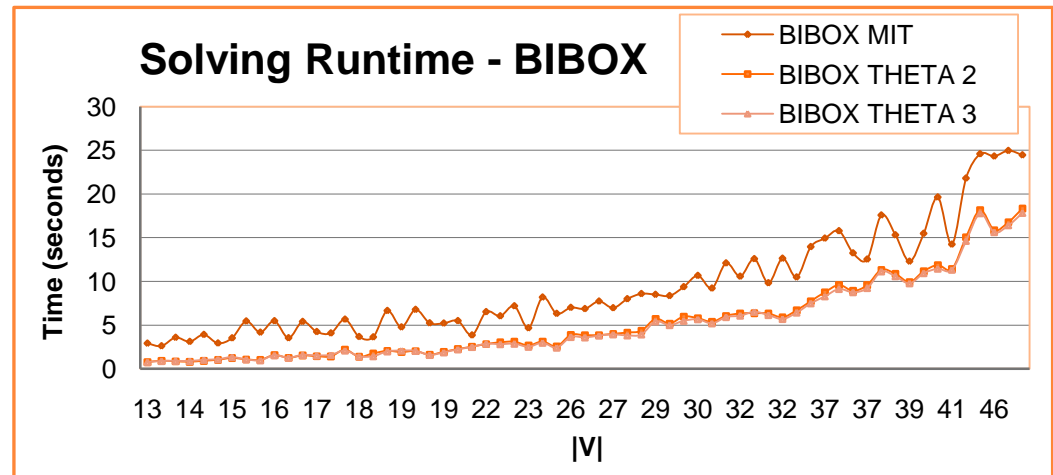
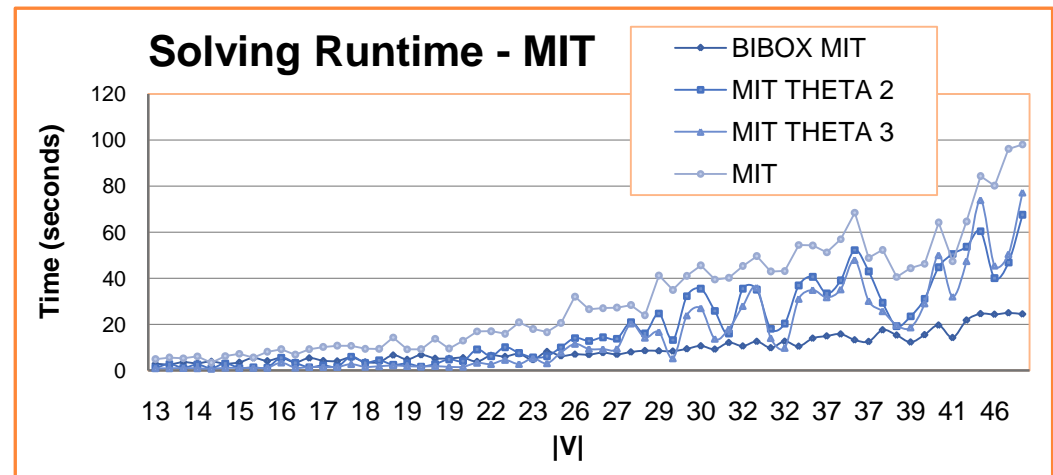
SOLUTION LENGTH

- All the algorithms implemented in C++
- Tests made with random bi-connected graphs
 - up-to 48 vertices
 - one unoccupied vertex
 - random arrangement of pebbles
- All the variants of the **BIBOX** algorithm generate order of magnitude **shorter** solution than the variants of **MIT** algorithm
- The application of macros **improves** all the algorithms significantly
- Preference of **transpositions** and **3-cycle rotations** were tested



- The same collection of problems
- Each problem solved **1000 times** to accumulate measurable time
- **Runtime** correlates with length of generated solutions
- All the variants of the **BIBOX** algorithm are faster than the variants of **MIT** algorithm
- The use of macros brings significant **speedup**
- There is almost **no difference** in runtime between the preference of **transpositions** and **3-cycle rotations**

EXPERIMENTAL EVALUATION RUNTIME



CONCLUSIONS AND REMARKS

- An application of **optimal macros** for composing a solution to a certain situations in the problem of **pebble motion on graphs** has been proposed
- Optimal macros were integrated into two existing algorithms (**MIT, BIBOX**) for pebble motion on graphs
- The resulting algorithms (**MIT- θ , BIBOX- θ**) proved to be better in terms of **length** of generated solutions and **runtime**
 - **BIBOX- θ** algorithm is **the best** of all the tested algorithms
- All the algorithms can be used for **multi-robot path planning**, however parallelism may be wasted
- Increasing parallelism:
 - define **dependence** between moves within generated sequential solutions
 - dependent move must be performed sequentially
 - for each move an **earliest execution time** is calculated using the **method of critical path**
- Future work: **optimize solutions** in a post-processing step

