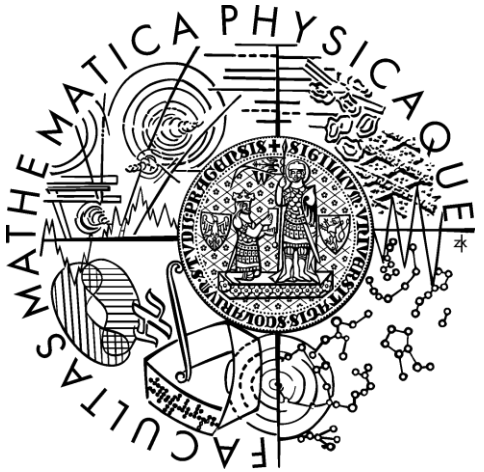


An Application of Pebble Motion on Graphs to Abstract Multi-robot Path Planning

Pavel Surynek
Charles University in Prague
Faculty of Mathematics and Physics
Czech Republic

Pebble Motion on Graphs

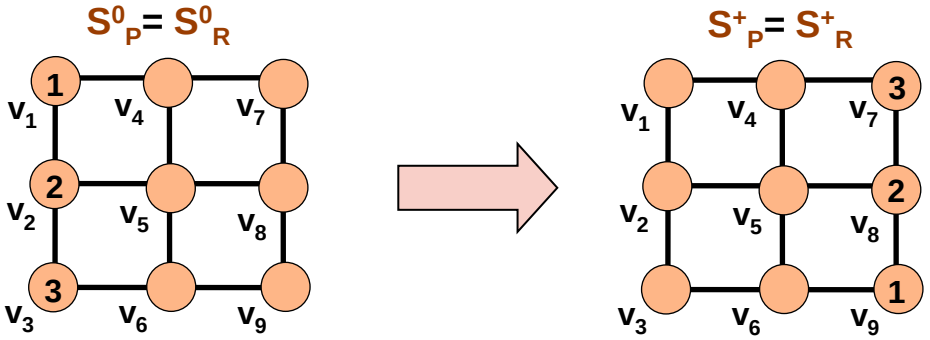
- **Input:** An undirected graph **G=(V,E)** and a set of pebbles **$P=\{p_1,p_2,...,p_\mu\}$**, where **$\mu<|V|$**
- **Rules:**
  - **each pebble** is placed **in a vertex** (at most one pebble in a vertex)
  - a **pebble can be moved into an unoccupied** vertex through an edge (no other pebble is 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 **$R=\{r_1,r_2,...,r_\mu\}$**, 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

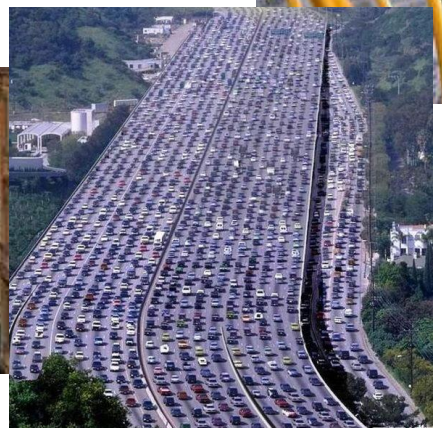
O_1=[**v_1**, v_4, v_7, v_8, **v_9**]
O_2=[**v_2**, v_1, v_4, v_7, **v_8**]
O_3=[**v_3**, v_2, v_1, v_4, **v_7**]

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**)

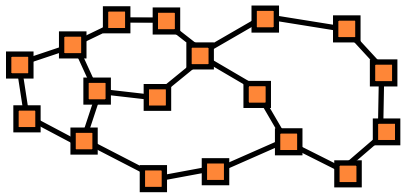
Motivation for the Problem

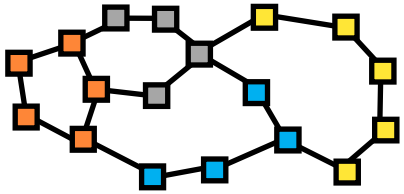
- rearranging containers (robot = **container**)
- heavy traffic control (robot = **car**)
- data transfer planning (robot = **data packet**)
- lift transportation planning in future buildings (robot = **lift**)



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}∈E | x,y ≠ v} is connected
- **Property:** Every bi-connected graph can be constructed from a cycle by consecutive adding of loops → **loop decomposition**

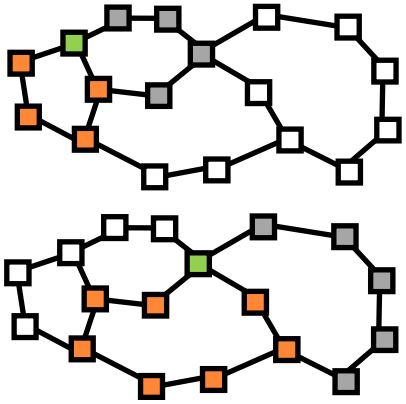




Original cycle
1st loop
2nd loop
3rd loop

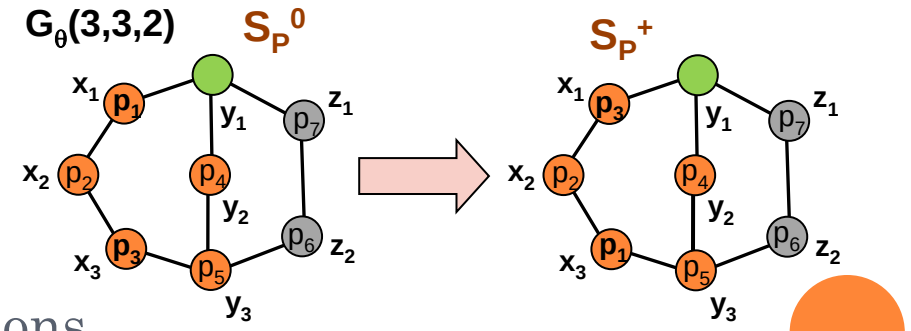
A Notion of θ-like Graph

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



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

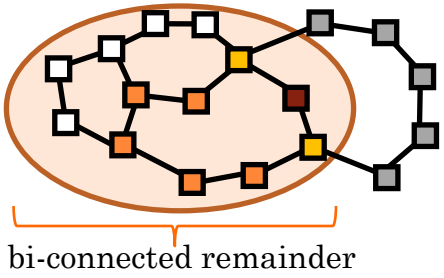
- Can be solved using **3-transitivity** of bi-connected graphs (any three pebbles can be moved to any three vertices)
- 3-transitivity induces **3-cycles**, 3-cycles induce **even** permutations

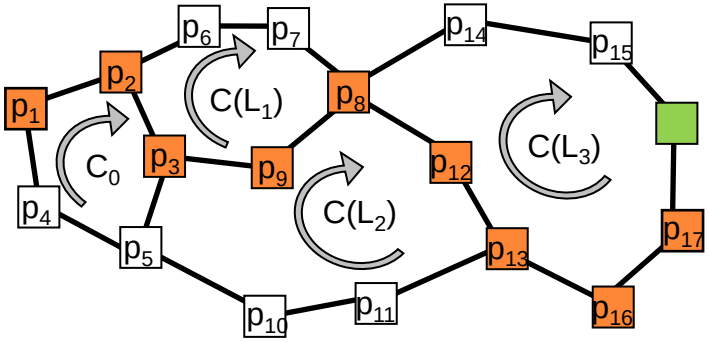


- However, the use of 3-transitivity generates **long** solutions

Original Algorithm Based on 3-transitivity (**MIT** Algorithm)

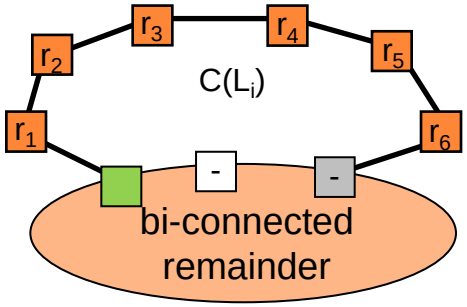
- **Proposition:** Any two distinct vertices in a bi-connected graph are connected by **two vertex disjoint paths**.
- **θ-decomposition =** decomposition of the bi-connected graph into **θ-like** sub-graphs (sub-graphs are not disjoint)
  - constructed from **loop decomposition**
  - a **θ-like sub-graph** 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

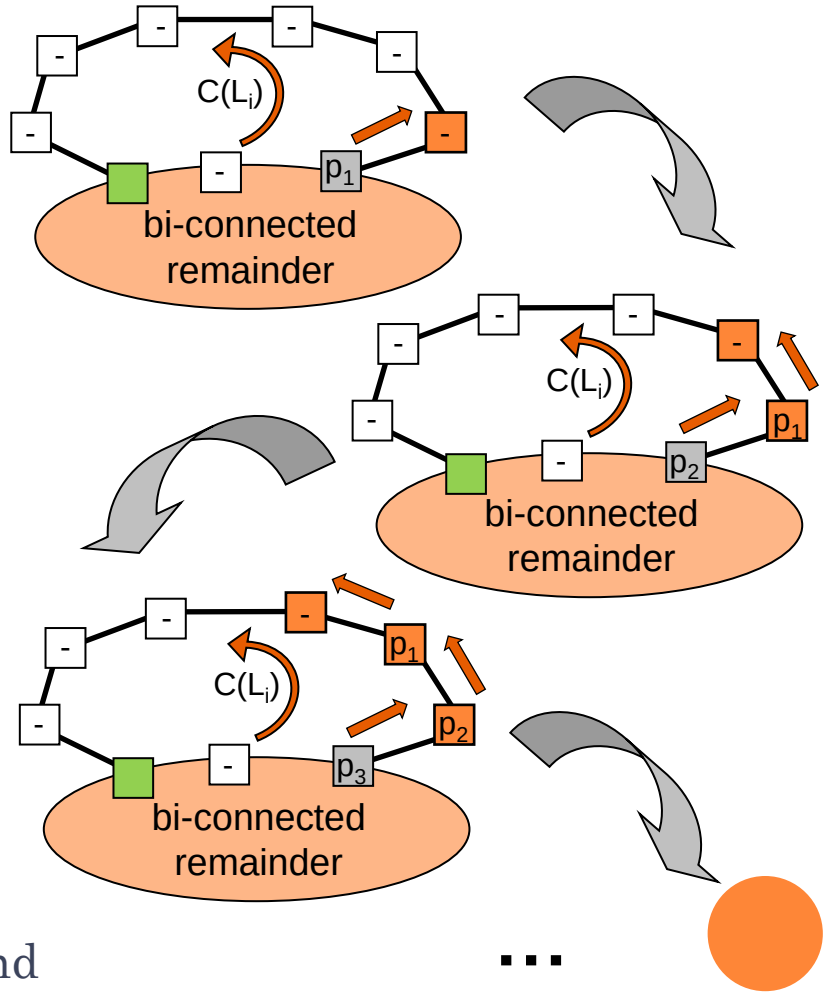




- **MIT Algorithm:**
  - move a pebble into the θ-like sub-graph
  - use **3-transitivity** to place pebble to its final position within θ-like sub-graph

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

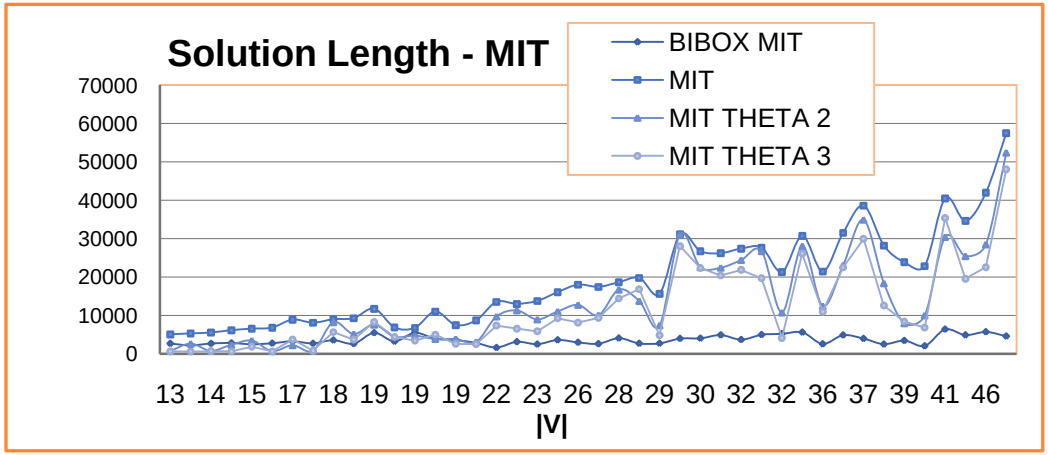
- Interpret arrangement of pebbles in a **θ-like graph** as a **permutation**
- **Proposition 1:**
  - **Any permutation** over **μ** elements can be obtained as a composition of at most **μ-1 transpositions**.
- **Proposition 2:**
  - Any **even permutation** over **μ** elements can be obtained as a composition of at most **μ-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

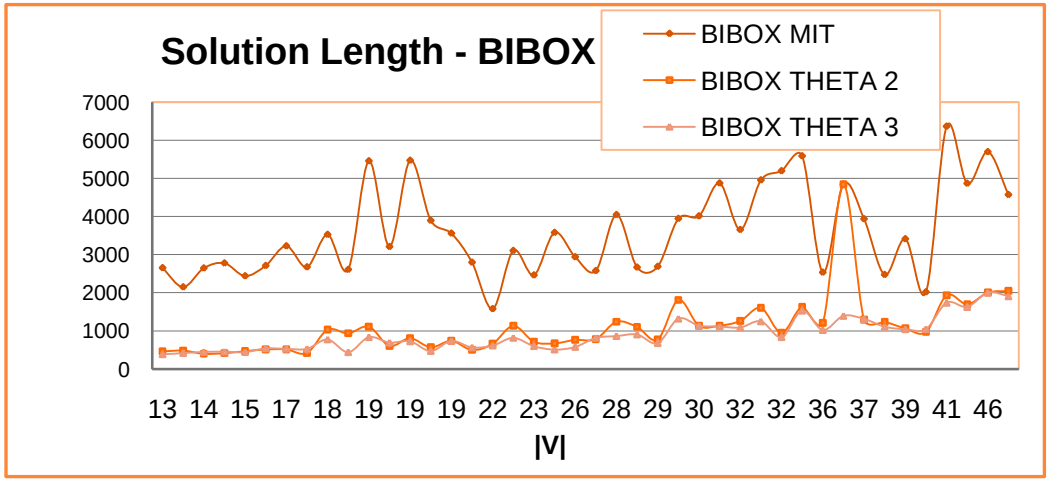
Application of Pattern Database within **MIT** and **BIBOX** Algorithms

- 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 $\mathbf{O(|V|^4)}$
  - generate solutions of the length $\mathbf{O(|V|^4)}$
- Theoretically same, however MIT and MIT-θ have higher constants in the asymptotic time and solution length estimation

Experimental Evaluation Solution Length

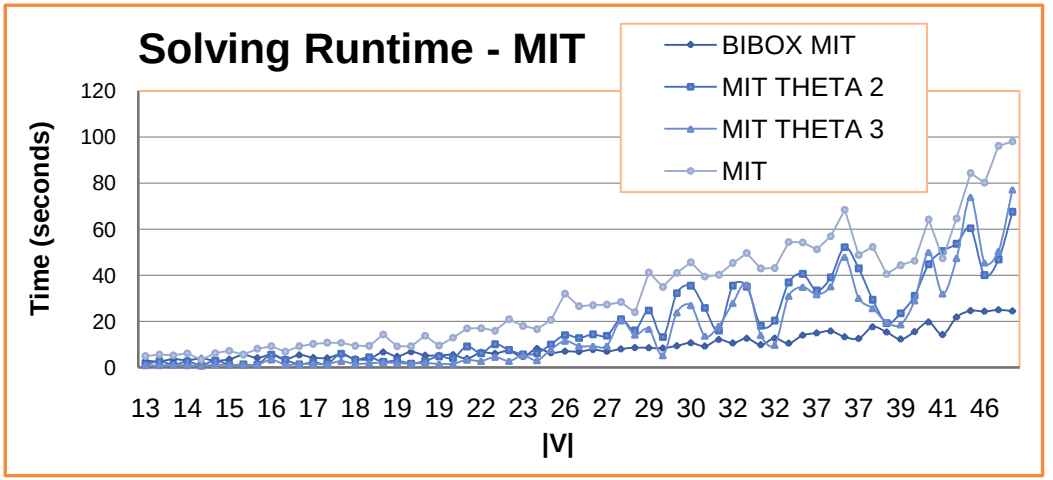
- All the algorithms implemented in **C++**
- Tests made with random bi-connected graphs
  - up-to 48 vertices
  - one unoccupied vertex
  - random arrangement of pebbles
- All the variants of the **BIBOX** algorithm generate order of magnitude **shorter** solution than the variants of **MIT** algorithm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

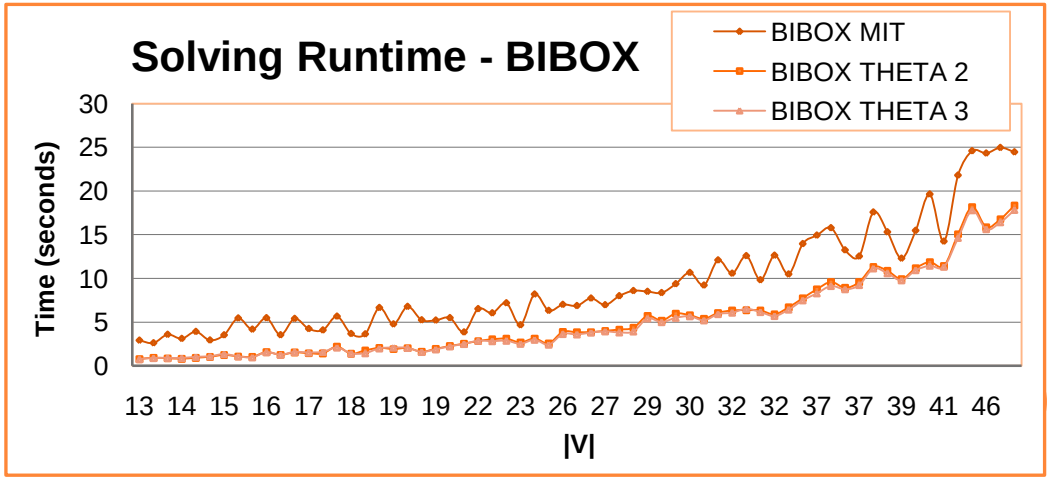




Experimental Evaluation Runtime

- 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**





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