Redundancy Elimination in Highly Parallel Solutions of Motion Coordination Problems

Pavel Surynek

CSP Seminar, Kobe University
October 26, 2011

Problem of motion on a graph

- **Abstraction** for tasks of motion of multiple (autonomous or passive) entities in a certain environment (real or virtual).
  - Entities have given an **initial** and a **goal** arrangement in the environment.
  - We need to **plan movements of entities in time**, so that entities reach the goal arrangement while **physical limitations are observed**.
- **Physical limitations** are:
  - Entities must **not collide with each other**.
  - Entities must **not collide with obstacles** in the environment.
- There are two basic **abstractions** of the task:
  - The problem of ***pebble motion on a graph***.
  - The problem of ***path-planning for multiple robots***.

Problem of **pebble motion on a graph** (1)

Wilson, 1974; Kornhauser et al., 1984

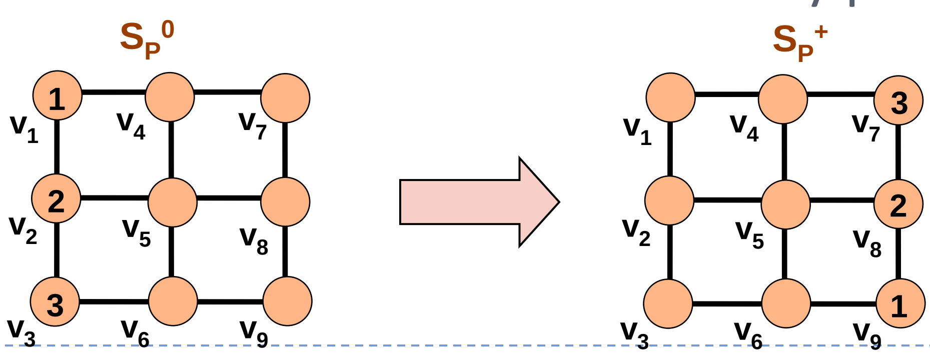
- A popular moving puzzle, that can be abstracted as the problem of pebble motion on a graph is known as **Lloyd's fifteen.**
  - Entities are represented by **pebbles** labeled by numbers.

- The environment is modeled as an **undirected graph** where **vertices represent locations** in the environment occupied by pebbles and **edges** enable pebbles to go to the **neighboring location**.
- **Formal definition** of the task of pebble motion on a graph:
  - It is a quadruple $\Pi = (G, P, S_P^0, S_P^+)$, where:
    - $G=(V,E)$ is an **undirected graph**,
    - $P = \{p_1, p_2, \ldots, p_\mu\}$, where $\mu < |V|$ is a **set of pebbles**,
    - $S_P^0: P \rightarrow V$ is a uniquely invertible function determining the **initial arrangement of pebbles** in vertices of G, and
    - $S_P^+: P \rightarrow V$ is a uniquely invertible function determining the **goal arrangement of pebbles** in vertices of G.

Problem of **pebble motion on a graph** (2)

Wilson, 1974; Kornhauser et al., 1984

- Time is discrete in the model. **Time steps** and their ordering is isomorphic to the structure of natural numbers.
- The **dynamicity** of the task is as follows:
  - A pebble occupying a vertex at time step *i* can move into a neighboring vertex (the move is finished at time step *i+1*) if the target vertex is **unoccupied** at time step *i* and **no other pebble** is moving simultaneously into the same target vertex
- For the given $\Pi = (G, P, S_P^0, S_P^+)$, we need to find:
  - A sequence of moves for every pebble such that dynamicity constraint is satisfied and every pebble reaches its goal vertex.



Solution of an instance of the problem of pebble motion on a graph with P={1,2,3}

makespan=7

$M_1=[\mathbf{v_1}, v_4, v_7, v_8, v_9, v_9, \mathbf{v_9}]$

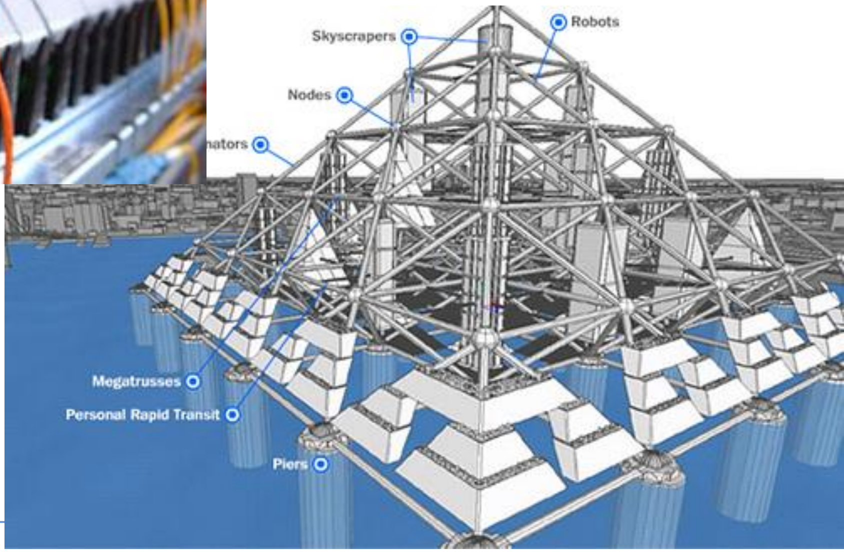
$M_2=[\mathbf{v_2}, v_2, v_1, v_4, v_7, v_8, \mathbf{v_8}]$

$M_3=[\mathbf{v_3}, v_3, v_3, v_2, v_1, v_4, \mathbf{v_7}]$

Time step: **1 2 3 4 5 6 7**

Is there any real-life **motivation**?

- Container rearrangement
  (entity = **container**)
- Heavy traffic
  (entity = **automobile** (in jam))
- Data transfer
  (entity = **data packet**)
- Generalized lifts
  (entity = **lift**)



Is the motion task **easy** or **hard**?

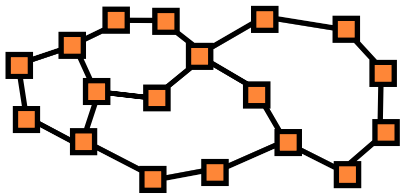
- **Basic** variant of the task is **easy to solve**:
  - There exists an algorithm with **worst case time complexity** of $O(|V|^3)$ that generates solutions of the **makespan** $O(|V|^3)$ for any instance of pebble motion on G=(V,E) (Kornhauser et al., 1984).

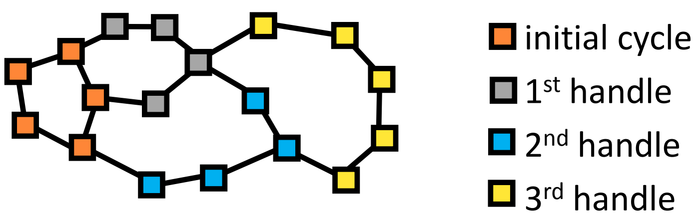
- If we want a **solution** that is **as short as possible** the complexity increases:
  - The optimization variant of the problem of pebble motion on a graph is **NP-hard** (Ratner a Warmuth, 1986).

- We focused on generating and improving **sub-optimal** solutions:
  - Restriction on **bi-connected graphs** – the task is almost always solvable.

The case with **bi-connected graph**

- Instances over bi-connected graph are practically most important.
  - Almost all the goal arrangements of pebbles are **reachable** from any initial arrangement.
- We allow only a **single unoccupied vertex** (this represents the most difficult case).
- An undirected graph $G=(V,E)$ is **bi-connected** 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.
- The **important property:** Every bi-connected graph can be constructed from a **cycle** by adding **handles**.
  → **handle decomposition**

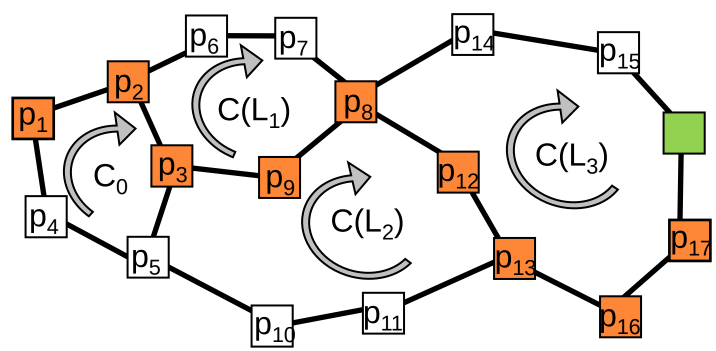




Algorithm **BIBOX-θ** (1)

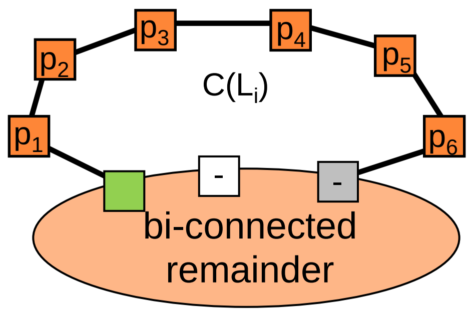
Surynek, 2009

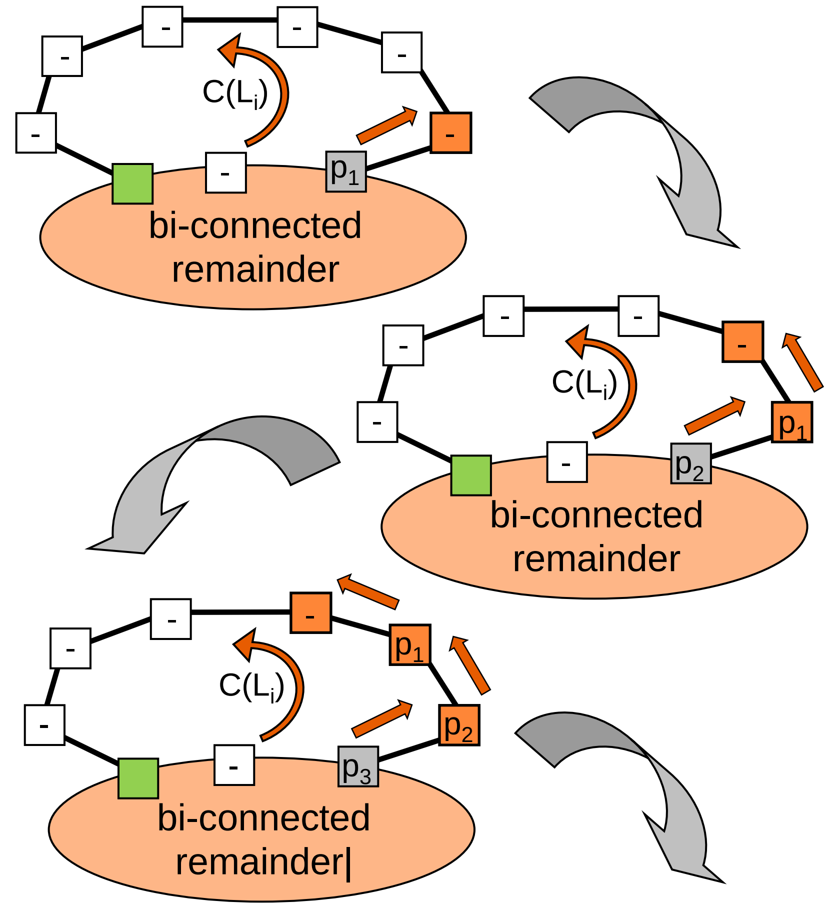
- Algorithm **BIBOX-θ** solves tasks of pebble motion on a graph.
  - The input graph is supposed to be **bi-connected**.
    - The algorithm is exploits handle decomposition of the input graph.
  - Just **one vertex** is supposed to be **unoccupied**.
    - If this is not the case, dummy pebbles are added to the graph. They are eventually filtered out of the final solution.
  - Algorithms produces a solution of any instance over G=(V,E) in the worst case time of $O(|V|^4)$, still practically better than (Kornhauser et al., 1984).
- The basic ability it to move a pebble into a selected vertex:
  - **Relocation** of the unoccupied vertex,
  - **rotations** along handles.



Algorithm **BIBOX-θ** (2)

- Using the ability of moving a selected pebble into a selected vertex more complex movements can be done:
  - Stacking pebble into a handle:



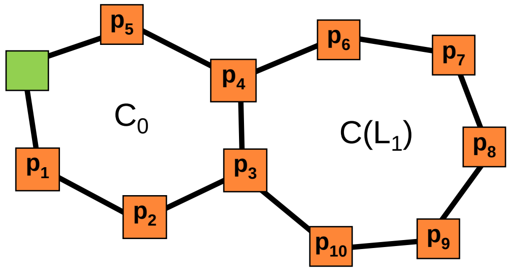


- The process of **stacking**
  - Consider the **last handle**
    - Move the pebble into the grey vertex.
    - A rotation of the handle is made using the **green** unoccupied vertex.
    - …

…

Algorithm **BIBOX-θ** (3)

- Initial cycle and the first handle (so called **θ-like graph**) represent a special case.
  - The process of stacking does not work here.

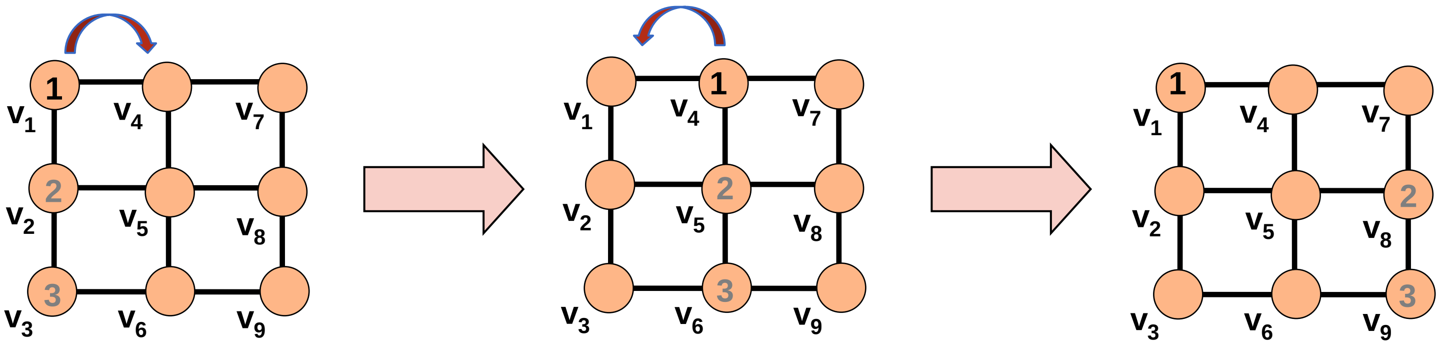


- The resulting (even) **permutation** of pebbles is composed of rotations along 3-cycles (without further details).
  - **Bottleneck** of the algorithm – known constructions of solutions to 3-cycle rotations use too many moves.
  - We exploit a **database** containing pre-computed optimal solutions to 3-cycle rotations instead (a form of pattern database)
  - The **overall sub-optimal solution** is composed of optimal solutions to 3-cycle rotations.
    - → **Sub-optimal** solution of relatively high quality.

The **major drawback** of the described process

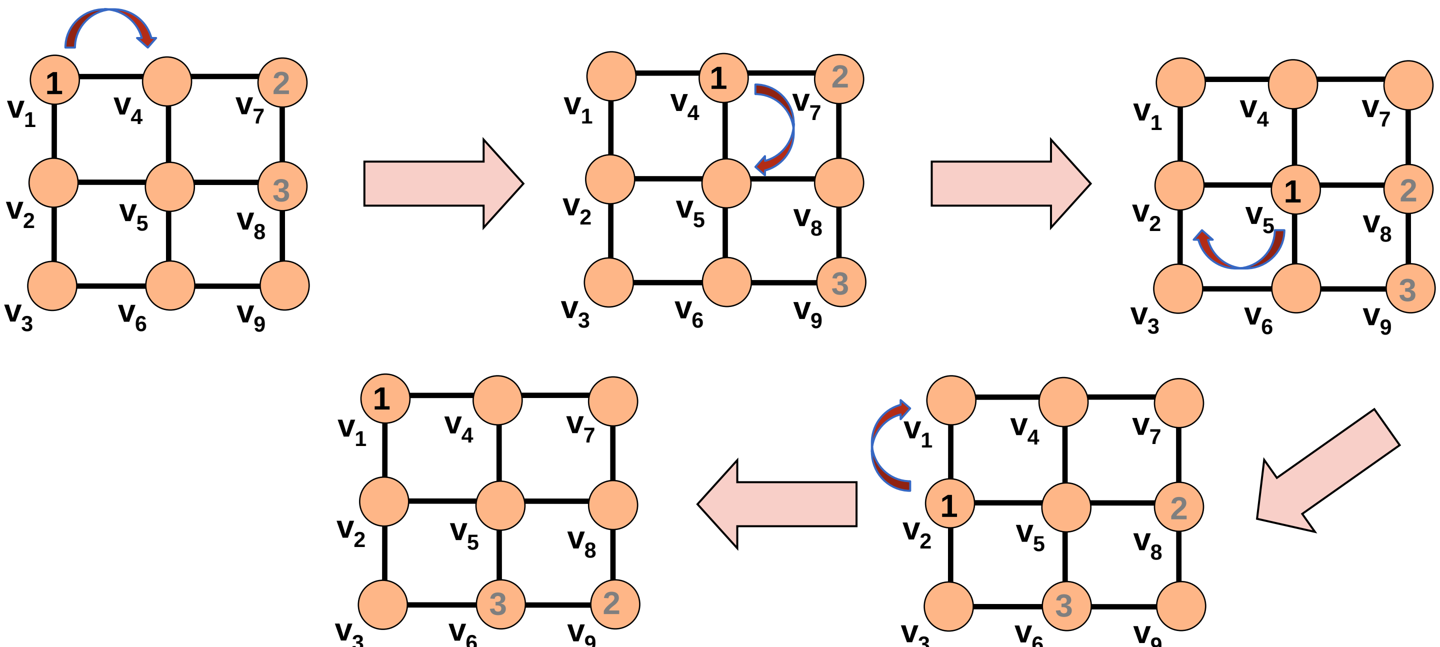
- If the initial graph is not fully occupied by pebbles at the beginning.
  - **Dummy pebbles are added**, modified instance is solved.
  - Movements of dummy pebbles are **filtered out** eventually.
- Several types of **redundancies** in generated solutions were discovered using visualization software **GraphRec** (Koupý, 2010):
  - **(i) Inverse moves**
    - A move that reverts the directly preceding move.
  - **(ii) Redundant moves**
    - A sequence of moves that relocates a pebble into the same vertex (notice possible interference).
  - **(iii) Long sequence of moves**
    - A sequence of moves that relocates a pebble into some vertex while there exists a shorter sequence doing the same (notice possible interference).

(i) **Inverse** moves



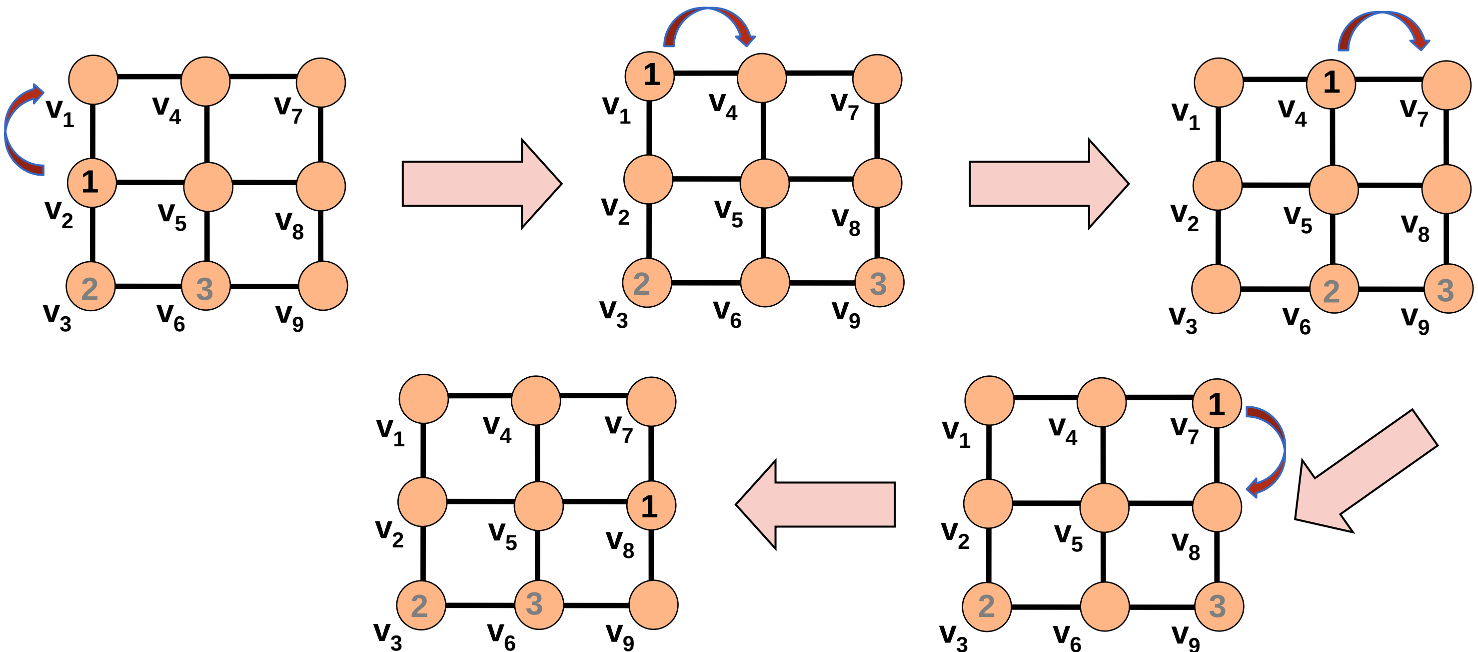
- **Pebble 1** has performed a pair of **inverse** moves.
  - Let us have a sequence of moves Φ
  - A simple algorithm can eliminate inverse moves from Φ in the worst case time of $O(|\Phi|^2)$
  - Removal of a single pair of inverse moves can result into occurrence of a new pair of inverse moves.

(ii) **Redundant** moves



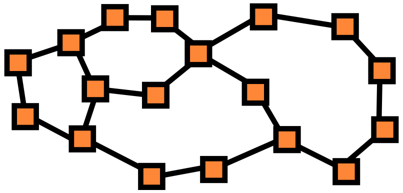
- **Pebble 1** has performed a sequence of redundant moves.
  - It has returned to the starting vertex without interfering with other pebbles.
  - A simple algorithm can eliminate redundant moves from Φ in the worst case time of $O(|\Phi|^4)$.
  - New redundant sequences can appear as well.

(iii) **Long sequence** of moves

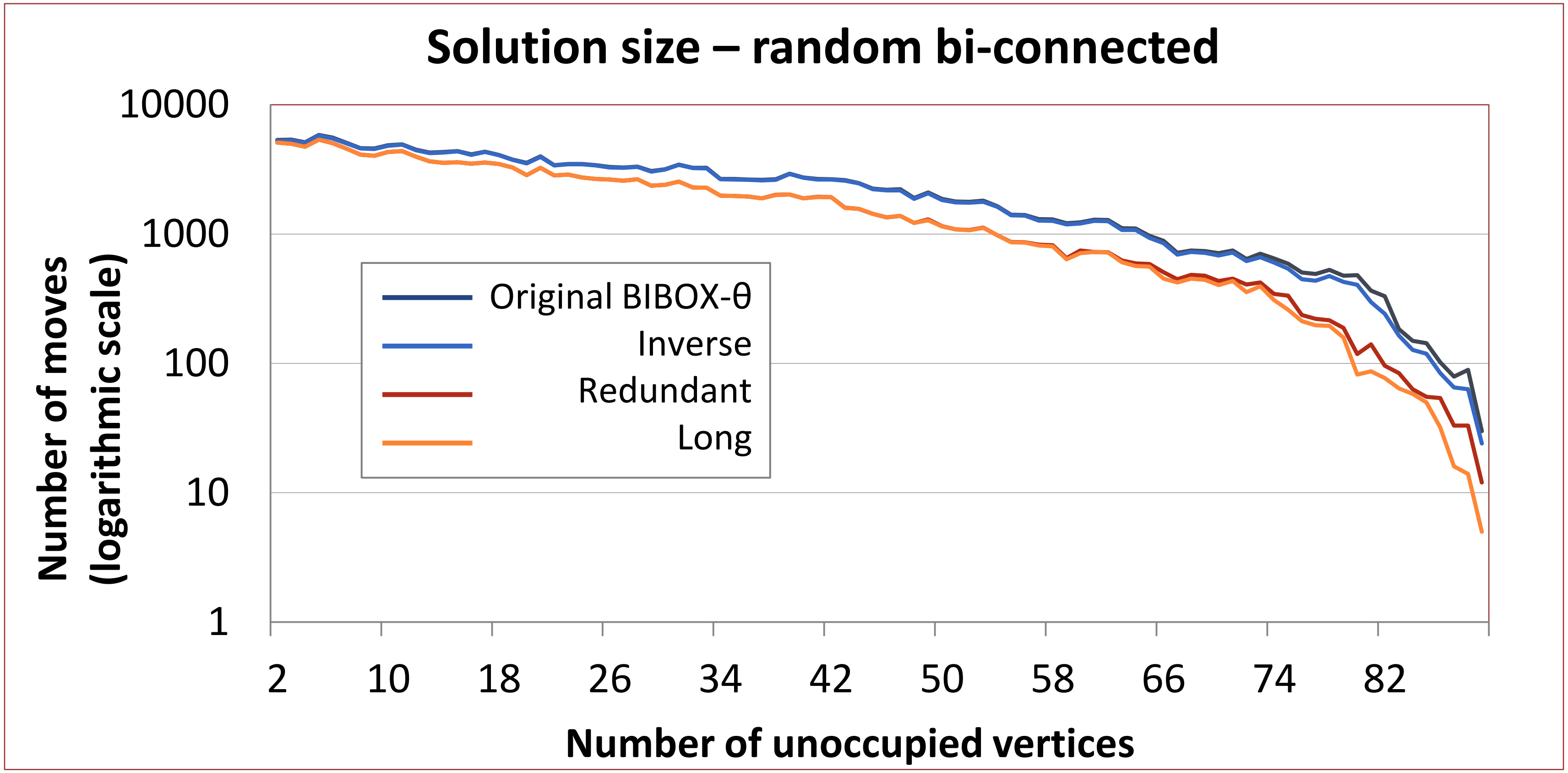


- **Pebble 1** has performed **long sequence** of moves**.**
  - It is possible to go along a shorter path without interfering with other pebbles.
  - A simple algorithm can eliminate long sequences from Φ in the worst case time of $O(|\Phi|^4+|\Phi|^3|V|^2)$.
  - Again, new long sequences of moves can appear.

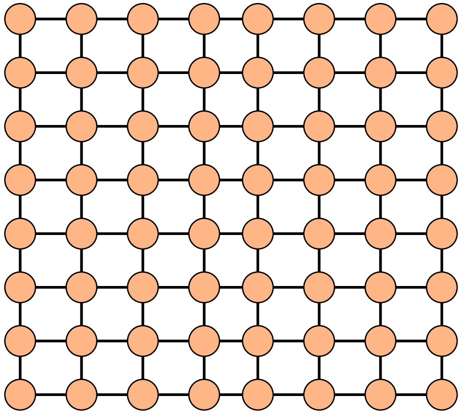
Experimental evaluation (1)



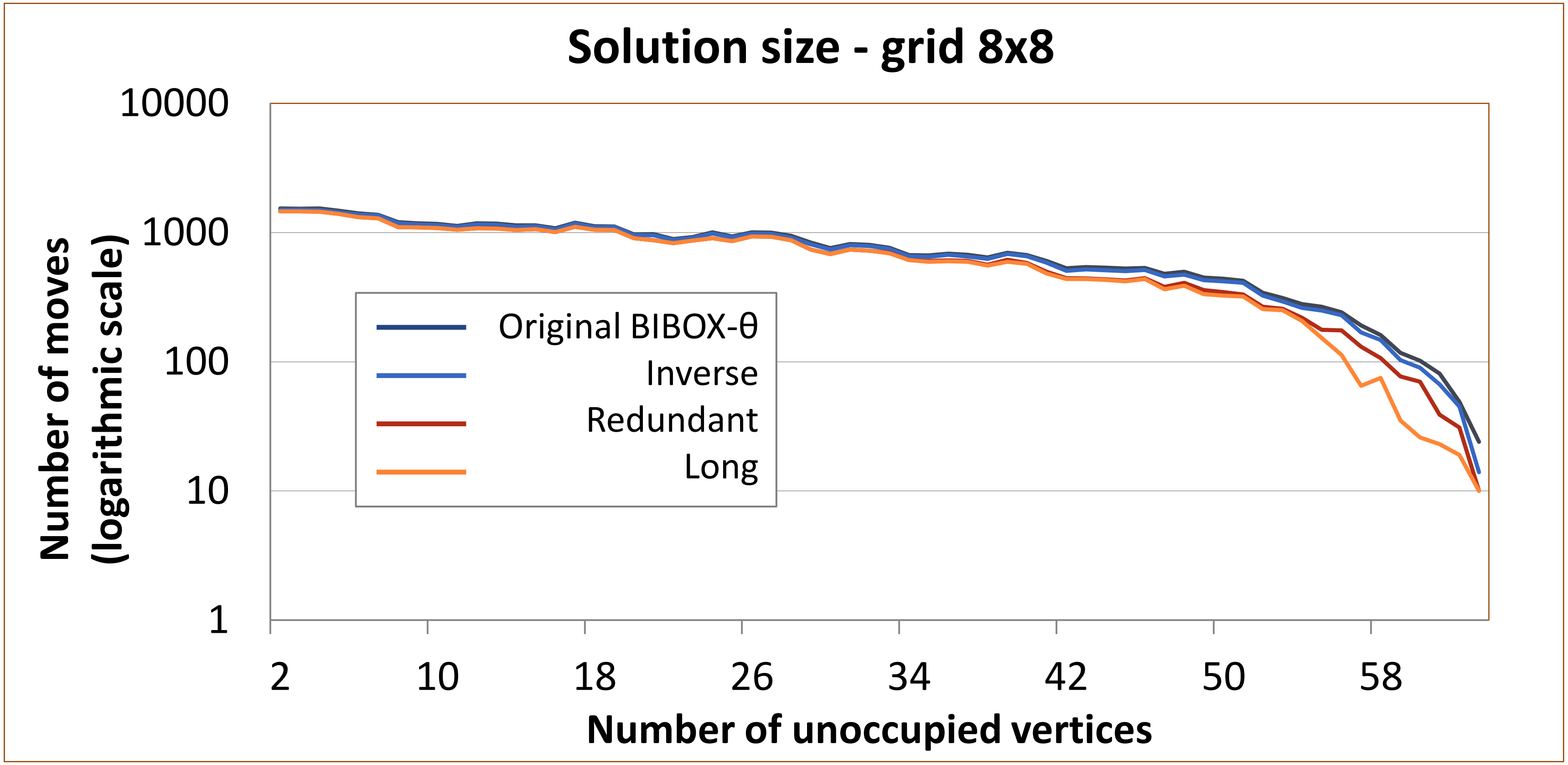
- **Random bi-connected** graph:
  - **Addition of handles of random lengths** to the currently constructed graph.
  - Initial and goal arrangement of pebbles are **random permutations**.



Experimental evaluation (2)



- **Grid 8x8**:
  - The initial and goal arrangement of pebble is a **random permutation** again.



Concluding remarks

- Visualization software **GraphRec** has been used to acquire knowledge about solutions of instances of pebble motion problem.
- Acquired knowledge has been used to **identify** redundancies and to develop algorithms to eliminate them.
- The experimental evaluation showed that the proposed elimination of redundancies can improve solutions significantly.
  - Especially if there are many unoccupied vertices