Joint Movement of Pebbles in Solving the (N^2-1)-Puzzle and its Applications in Cooperative Path-Finding

(JAAMAS Extended Abstract)

Pavel Surynek
National Institute of Advanced Industrial
Science and Technology (AIST)
2-3-26, Aomi, Koto-ku, Tokyo 135-0064, Japan
pavel.surynek@aist.go.jp

Petr Michalík
Accenture Central Europe B.V.Consulting
Jiráskovo náměstí 1981/6
120 00, Prague, Czechia
petr.michalik@accenture.com

ABSTRACT

Moving pebbles jointly in formations called snakes has been integrated into the Parberry's algorithm for solving the (N^2-1)-puzzle sub-optimally in the on-line mode. Using snakes consisting of 2 pebbles that are relocated jointly towards their goal positions after their opportunistic formation yields a measurable reduction of the total number of pebble movements at low extra computational cost. As the N×N-puzzle represents a special case of the cooperative path finding problem (CPF) we also transferred the concept of snake-like movements into the context of two rule-based sub-optimal algorithms for CPF – BIBOX and PUSH-and-SWAP (PUSH-and-ROTATE). The evaluation indicates significant benefit from employing snakes within the BIBOX algorithm and also increasing benefit in PUSH-and-SWAP being applied on bi-connected graphs with growing size of ears.

Author Keywords: Cooperative Path-Finding, Multi-agent Path Finding, Pebble Movement, PUSH-and-SWAP, BIBOX.

1. INTRODUCTION AND BACKGROUND

The task in the *(N^2-1)-puzzle* [3] is to move N^2-1 distinguishable pebbles on a board with N×N positions so that each pebble eventually reaches its unique goal position. Each position is either occupied by a pebble or is empty. A pebble can be moved in a discrete step to a vacant position in its neighborhood.

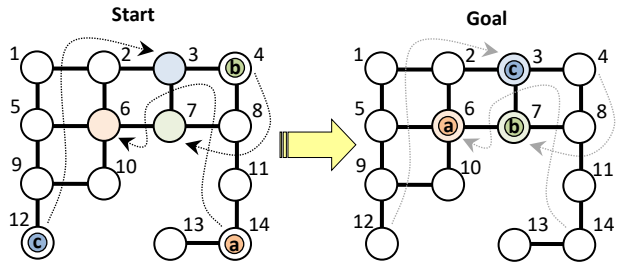

Figure 1. Example of CPF with 3 agents: a, b, c.

Cooperative path-finding (CPF) [4] represents a generalization of the (N^2-1)-puzzle. Instead of the board we have an undirected graph with vertices representing positions and edges defining the neighborhood into which movements can be done.

Appears in: *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017),*
S. Das, E. Durfee, K. Larson, M. Winikoff (eds.)
May 8–12, 2017, São Paulo, Brazil.

The role of pebbles is played by distinguishable agents in the context of CPF. Each vertex is occupied by at most one agent at a time. Again, each agent has its unique goal vertex and the task is to move agents so that they eventually reach their goals (see Figure 1).

Many algorithms have been developed for solving the relocation puzzles and CPF – see [6] for an overview. Here we focus on those that operate in the on-line mode and generate sub-optimal solutions in terms of the total number of moves. The contribution of this work is a technique for moving pebbles/agents jointly like snakes to reduce the total number of moves while still preserving the on-line character of target algorithms.

2. SNAKE-BASED RELOCATION

Parberry suggested an on-line sub-optimal algorithm for the (N^2-1)-puzzle that arranges pebbles one by one in rows of the board [2]. Movement of a pebble is enabled by vacating a vertex in front of it in the direction along a path towards the goal. Special cases are treated by a number of rules.

2.1 Snakes in the (N^2-1)-puzzle

Our improvement opportunistically groups pebbles that would be moved consecutively into pairs and relocate them towards their goals jointly like a *snake* if a preliminary check shows that it is less move consuming. One relocation of a pair of pebbles often consumes considerably less moves for vacating a vertex in front of the snake than if the two were relocated separately – Figure 2.

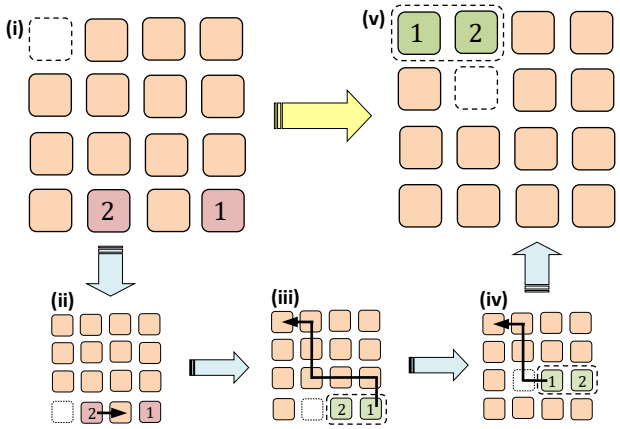

Figure 2. Snake formation of pebbles 1 and 2 and their joint relocation towards the left-top corner in the (4^2-1)-puzzle.

2.2 Snakes in Rule-based CPF Solving

The snake-based agent relocation has been also integrated into two sub-optimal rule-based algorithms for CPF – BIBOX [5] and PUSH-and-SWAP [1], [8].

BIBOX assumes bi-connected underlying graph G and at least two unoccupied vertices. It employs an *ear decomposition* [7] of the graph to arrange agents to their goals inductively. Any bi-connected graph G can be composed by adding ears $L_1,\ldots,L_n$ – each consisting of fresh vertices making a path – to a currently constructed graph by connecting entrance/endpoint of the ear somewhere in the existing graph; a cycle is taken at the beginning of the construction. Starting with the last ear L_n and arranging agents to their goal positions within L_n the task of path-finding for agents then reduces on the smaller sub-graph $G \setminus L_n$ with no need to move agents in L_n any more.

The process of arranging agents into an ear is done in a similar way to pushing items into a stack (see Figure 3). An agent arrives at the entrance of the ear (vertex v) and then all agents in the ear are pushed one position towards the endpoint (vertex u). This is the moment where the snake based reasoning can take place. Instead of moving agents towards the ear entrance one by one it is always checked if it is less motion consuming to move a pair of agents whose goal are consecutive in the ear jointly like a snake of size 2.

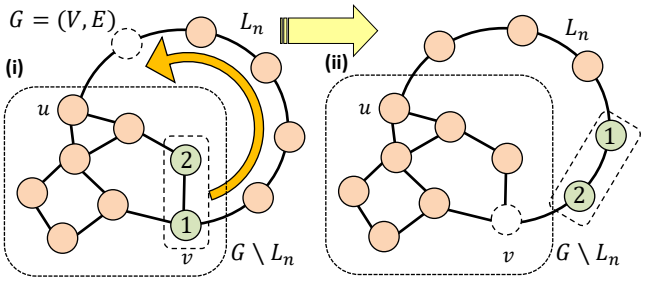

Figure 3. Agents 1 and 2 enter ear L_n like a snake jointly within a sub-procedure of the BIBOX algorithm.

A similar approach is used within our modification of the PUSH-and-SWAP algorithm. The algorithm is applicable on arbitrary underlying graph with at least two unoccupied vertices. Agents are moved towards their goals one by one while the placement of the next agent does not hurt already placed agents which however may be temporarily moved out of their goal positions. This is a substantial difference from BIBOX where agents once finished within an ear move never more.

An agent is moved towards its goal along a path connecting its current position with the goal. If on this path, the agent encounters another agent it is either moved out of the way in case this is possible or a more complex operation of swapping agents is initiated (an agent already placed in its goal cannot be simply moved out of the way). The swapping operation relocates the pair of agents needed to be swapped toward a vertex with enough neighbors where the ordering of agents is changed using fixed rules. Then all moves preceding the change of the ordering of agents are executed in the reverse order which leaves all other agents as if they were untouched except the swapped pair.

The snake based reasoning can be reflected inside the PUSH-and-SWAP algorithm by simply trying to move a pair of consecutive agents together if it saves moves with respect to their separate relocation. This modification leads to a more complicated set of rules for making the swap over a triple of agents instead of a pair [6].

3. EXPERIMENTAL EVALUATION

Our experimental evaluation fully shown in [6] revealed that snakes bring a consistent reduction of the total number of moves of approximately 8% to 9% in Parberry's algorithm and up to 50% in the BIBOX algorithm. This is observable in Figure 4: Parberry's algorithm and BIBOX are compared with their snake-improved versions. Forty randomly generated (50^2-1)-puzzles and instances over biconnected graph consisting of 90 vertices with initial cycle of size 7 and average size of ear 6 with varying number of unoccupied vertices were used.

PUSH-and-SWAP on the other hand turned out to be resistant against the snake based reasoning. Improvements were observed on biconnected graphs with growing size of ears but were unstable.

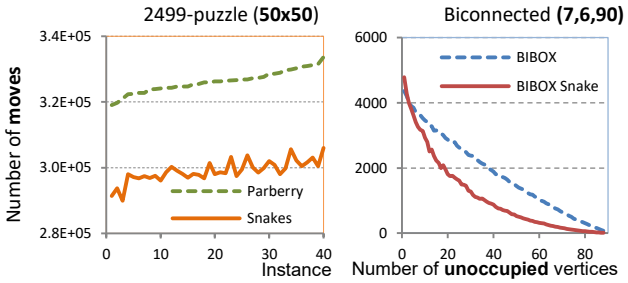

Figure 4. Improvement obtained by using snakes of size 2 in the algorithm of Parberry (left) and BIBOX (right).

4. CONCLUSIONS

The snake-based relocation of pebbles/agents has been shown to be a very flexible technique for reduction of the total number of moves generated by on-line algorithms for the (N^2-1)-puzzle and CPFs. We manage to integrate it into three different algorithms. As an open question, we consider extending the lengths of snakes from 2 to more pebble/agents which offers new challenges.

5. ACKNOWLEDGEMENT

This paper is supported by a project commissioned by the New Energy and Industrial Technology Development Organization Japan (NEDO).

6. REFERENCES

[1] Luna, R., Bekris, K. E. 2011. *Push and Swap: Fast Cooperative Path-Finding with Completeness Guarantees.* Proceedings IJCAI 2011, pp. 294-300, IJCAI.

[2] Parberry, I. 1995. *A real-time algorithm for the (n^2-1)-puzzle.* Information Processing Letters, Volume 56 (1), pp. 23-28, Elsevier.

[3] Ratner, D., Warmuth, M. K. 1990. *NxN Puzzle and Related Relocation Problems.* Journal of Symbolic Computation, Volume 10 (2), pp. 111-138, Elsevier.

[4] Silver, D. 2005. *Cooperative Pathfinding.* Proceedings of the AIIDE 2005, pp. 117-122, AAAI Press.

[5] Surynek, P. 2014. *Solving Abstract Cooperative Path-Finding in Densely Populated Environments.* Computational Intelligence, Volume 30 (2), pp. 402-450, Wiley.

[6] Surynek, P., Michalík, P. 2016. *The Joint Movement of Pebbles in Solving the (N^2-1)-Puzzle Suboptimally and its Applications in Rule-Based Cooperative Path-Finding.* Autonomous Agents and Multi-Agent Systems, IFAAMAS/Springer, in press.

[7] Tarjan, R. E. 1972. *Depth-First Search and Linear Graph Algorithms.* SIAM Journal on Computing, Volume 1 (2), pp. 146-160, SIAM.

[8] de Wilde, B., ter Mors, A., Witteveen, C. 2014. *Push and Rotate: a Complete Multi-robot Pathfinding Algorithm.* Journal of Artificial Intelligence Research (JAIR), Volume 51, pp. 443-492, AAAI Press.