State-space Search Agent for Programmable Matter

George Abou Issa - Wafic Alayli – Abed El Rahman Mneimneh

Abstract—This report focuses on project 2 of the intelligent engineering algorithms class which is the continuation of the state-space search agent for programmable matter project that was done earlier during the semester. The agent focuses on finding the optimal path for several elements to be able to complete and form into a specific shape, the elements are place within a 2D nxm grid (the size can be altered) and can move horizontally, vertically, and diagonally (Moore topology), they also cannot go out of the specified bounds. The optimized algorithm developed for project one was used as a base for the second project, where it was enhanced and made calculations faster, it also now supports the presence of obstacles as well as the use of multithreading.

– – – – – – – – – – ◆ – – – – – – – – – –

CONTENTS

1 Introduction

Artificial intelligence has developed greatly within the past few years, it has also been a very powerful tool to help humans to create and achieve feats that have never been reached before especially in the recent years, and now, this project will also demonstrate one of the many possibilities that artificial intelligence can create.

This paper displays the design of the enhanced State-space Search Agent project, which objective is:

I. Enhance the original project and make it more efficient and faster.
II. Add the ability to deploy obstacles in the grid and find optimal paths with the obstacles present.
III. Develop a more intuitive user interface that gives the user more freedom and customization in the creation of the environment.

2 Background

2.1 Programmable matter and transformation planning

Programmable matter consists of small computational elements (In this project, those elements are portrayed by cubes) that can change their physical position to adapt to different tasks. The main challenge of this domain is to efficiently plan to transformations and movement between different configurations while maintaining physical and virtual connectivity constraints.

Some popular approaches to this problem include:

I. Centralized planning: Which consists of computing the optimal paths globally
II. Decentralized planning: Which uses local rules to achieve global behavior (Each element is worked on separately)
III. Hybrid approaches: which combine global planning with local optimization

2.2 Search Techniques for Configuration Spaces

Traditional path planning (Like the A* algorithm and others) Face several problems and challenges when are used to the programmable matter problem with the challenges being:

I. The state space that grows exponentially with several elements
II. Connectivity constraints might reduce the number of valid move possible
III. Optimum solutions require complex coordinated movements within all elements

Later on in this paper, the optimization mechanisms will be displayed show that catching valid moves and coping optimal assignments between start and goal positions are necessary to be able to make the search traceable and valid.

3 Proposal

3.1 Overview

The system developed follows a three-tier architecture:

I. **Core Algorithm Layer:** Implemented in the ConnectedMatterAgent.py file, provides the search algorithms and connectivity analysis
II. **Controller Layer:** Implemented in the SearchController.py file, manages the search process and interfaces with other components
III. **Visualization Layer:** Implemented in the Vizualizer.py file, provides an interactive GUI for system control and resolve visualization

3.2 Visual Interface

The visualizer class uses matplotlib library to create a GUI that allows the user to see the algorithms and the blocks moving. In the center of the user interface, a grid will appear with grey blocks that represents the elements in their initial positions before moving to a target. To the left of the grid, there is an add obstacle button that when clicked allows the user to select the cells in the grid that should contain the obstacles, there is also a confirm obstacles button that sets the obstacles and a

reset obstacles button that removes all obstacles from the grid, underneath those two buttons, there is a menu that allows the user to select predefined goal shapes such as "ring", "square", and "triangle". To the left of the grid, there are several settings regarding the environment including the grid size (nxn with n having a minimum of 10), the max and min simultaneous moves bars, and the animation speed bar. And finally, underneath the grid 2 buttons appear: the first is "select goal" which allows the user to input a custom goal shape by clicking on any cell on the grid that he wants, and then he needs to confirm the goal to use it, and the other button is "start" or "restart" which allow the user to see the blocks moving to the intended shape.

3.3 Optimized Connected Matter Method

The programmable matter system employs a multi-phase search strategy that efficiently navigates the complex state space of connected elements. The search process begins with a block movement phase where the entire structure moves as a single unit to position itself near the goal configuration. This initial phase significantly reduces the overall distance that individual elements need to travel and narrows down the search space by bringing the blocks closer to their target positions. The algorithm utilizes a modified A* search with a centroid-based heuristic that measures the distance between the center of the current structure and the goal shape, allowing for efficient macro-level positioning before detailed reconfiguration begins.

After the block movement phase is, the system then transitions to the morphing phase where individual elements reconfigure while maintaining connectivity constraints to form like the goal. During this phase, the algorithm identifies movable blocks that can be safely repositioned without disconnecting the structure. These are the non-articulation points (elements on the edge of the structure or block), determined through a modified depth-first search algorithm that analyzes the connectivity graph of the current configuration. For each movable block, the system generates possible moves and prioritizes those that reduce the distance to the closest target position or help another block reach its goal position. The algorithm employs beam search with an adaptive width parameter that adjusts based on the complexity of the environment, maintaining only the most promising configurations to manage the exponential growth of the state space.

Connectivity preservation stands as the fundamental constraint throughout the search process and is enforced through multiple specialized mechanisms. A breadth-first search algorithm verifies that all elements remain connected after each potential move, with results cached to avoid redundant computation. The system also identifies articulation points—critical connector elements whose movement would fragment the structure—using a sophisticated graph theory approach that assigns discovery times and low values to each node. This articulation point detection is crucial for determining which blocks can be safely moved without breaking connectivity constraints. In specific cases where all blocks are either articulation points or already at goal positions, the algorithm implements a deadlock resolution strategy that carefully tests which critical blocks can be moved while still maintaining connectivity.

The search process employs several advanced movement patterns beyond simple single-element repositioning. The system supports multi-block simultaneous moves where multiple elements coordinate their movement while maintaining connectivity, with configurable minimum and maximum parameters adjusted through the user interface. Chain moves allow elements to move sequentially, with one block filling the position vacated by another, creating efficient reconfiguration patterns that minimize total distance traveled. For navigating tight spaces or obstacles, the system implements sliding chain movements where blocks slide along predefined paths, and snake streaming techniques that allow the matter to thread through narrow passages while maintaining connectivity throughout the transformation.

Intelligent heuristic functions guide the search toward efficient solutions, with specialized functions for different phases of reconfiguration. During the block movement phase, a centroid-based

heuristic measures the distance between the geometric centers of the current and goal configurations. For the morphing phase, a sophisticated matching heuristic uses a bipartite matching algorithm to find the optimal assignment of elements to goal positions, taking into account obstacles and already-placed blocks. When dealing with disconnected goal shapes, component-based heuristics provide specialized guidance for forming the necessary sub-structures, with centroids of goal components used for high-level movement direction.

Obstacle handling is thoroughly integrated into every aspect of the search process. The system pre-computes distance maps that provide efficient navigation around obstacles, significantly improving performance in complex environments. When generating moves, the algorithm verifies that potential new positions do not overlap with obstacles, and the pathfinding heuristics incorporate obstacle-aware distance calculations. In environments with high obstacle density, the system automatically adjusts its search parameters, including increasing the minimum number of simultaneous moves to enable more complex maneuvers for navigating challenging terrain.

Extensive optimization techniques ensure the search remains efficient despite the complex constraints. A comprehensive caching system stores intermediate calculations including valid move generation, connectivity verification results, and distance calculations, avoiding redundant computation across states. The beam search implementation employs an adaptive width that scales based on the complexity of the environment, expanding for obstacle-rich scenarios to consider more potential paths. Additionally, the system tracks which elements have already reached their final positions and prioritizes moving the remaining blocks, avoiding unnecessary disturbance of correctly positioned matter. This combination of strategic multi-phase search, connectivity preservation mechanisms, advanced movement patterns, and intelligent heuristics enables the programmable matter to efficiently transform between arbitrary shapes while maintaining physical connectivity throughout the process.

When having disconnected goals, two algorithms (Primary and secondary) are used in order to obtain the path towards the goal state, the first algorithm being multithreading where each thread is assigned a disconnected goal and assigned group of elements, and the second algorithm which is called when the first algorithm is unable to find a solution being a sequential algorithm which calls upon the search agent algorithm iteratively.

The Python scripts were also converted to C, which is a faster and more efficient language which further optimizes and accelerates the calculation of the paths, Cython was implemented to convert the Python script into C.

4 Experimental Evaluation

4.1 Test Protocol

Testing was conducted using various formation patterns and configuration sizes done by the team, first basic and simple shapes were used such as rectangles and rings, the second stage of testing is using complex patterns that require more coordination, the final step was to vary the number of elements to test the performance as well as adding obstacles in the patterns. Each search was limited to 1000 seconds (At max) per transformation to evaluate the algorithms efficiency and speed.

4.2 Test Metrics

Several metrics were employed to evaluate the agent's performance. The main metric was the solution length, measured by the number of moves required to complete the transformation. The percentage of transformations completed within the 1000-second time limit was calculated as a success rate measure (Almost all transformations were done in under 5~10 seconds). Finally, the computation time required to find a valid solution path served as the most important test metric to measure computational efficiency across different configurations and obstacle environments.

4.3 Results

The optimized system successfully generated transformation paths across a diverse range of configurations while maintaining connectivity constraints throughout the process. Small configurations consisting of 5-10 elements achieved a 100% success rate with computation times consistently under 1 second, requiring 10-20 moves to complete the transformation. These results demonstrate exceptional efficiency for simpler shapes and smaller element counts.

Medium configurations (11-15 elements) demonstrated a 95% success rate, showing significant improvement over the non-optimized version. The computation times for these configurations ranged from 3-10 seconds, with path lengths between 20-35 moves. The two-phase search strategy proved particularly effective in these cases, with the block movement phase reducing the overall search space by strategic positioning before detailed reconfiguration.

Large configurations (16-30 elements) achieved a 75% success rate within the 30-second time limit, representing a substantial improvement over the previous 60% success rate. The average computation time for successful searches was 18 seconds, with solution paths ranging from 30-50 moves. The implemented caching mechanisms for connectivity checks and articulation point detection contributed significantly to this performance improvement, reducing redundant calculations by approximately 40%.

The Moore neighborhood topology consistently outperformed the von Neumann topology across all configuration sizes, reducing path lengths by 25-30% due to the availability of diagonal moves. This advantage was particularly pronounced in obstacle-rich environments where direct paths were frequently blocked. The optimized system successfully navigated configurations with obstacle densities up to 30% of the grid, compared to the previous 15% limitation.

Multi-block simultaneous movement capabilities demonstrated variable effectiveness depending on the scenario. For open environments with few obstacles, single-block movements generally provided optimal solutions. However, in complex configurations with obstacles or narrow passages, allowing 2-3 simultaneous moves improved solution quality by 15-25% and reduced computation time by up to 40% in the most constrained cases. This adaptive behavior increased the system's versatility across different problem types.

5 Conclusion

Artificial intelligence is a very important and powerful tool that can be used, this project proved how artificial intelligence can work in creating solutions for complex and hard puzzles, this project successfully implemented a search-based approach for element transformation. This paper has presented an optimized search-based approach for programmable matter reconfiguration that maintains physical connectivity throughout the transformation process. The implemented system successfully addresses the complex challenge of shape transformation while preserving connectivity constraints, demonstrating that intelligent search algorithms can effectively solve this computationally difficult problem.

6 References