Efficient Solver for Dynamic Puzzle Games: Baba Is Y'all

Yuhang Lin, Xianqing Zeng

***Abstract*—The demand for intelligent solvers for puzzle games has overgrown with the increase in depth and complexity of these games. Most traditional puzzles with fixed rules that do not change during the playing process already have many high-performance algorithms to find solutions. However, puzzles with dynamic rule systems just occur in recent years and have only a few algorithms to obtain a solution. Because of the unique dynamic mechanisms, we choose the game *Baba Is You* as the research object, and attempt to develop an efficient and high-performance agent for it.**

***Index Terms*—Baba Is You, Agent, Dynamic Rule, Algorithm Design.**

I. Introduction

WITH the increase in depth and complexity of puzzle games, answers to these puzzles require more intelligence and better problem-solving ability. More variables and limitations in consideration make it hard to calculate and obtain correct answers manually and may take more time, like a few hours or even a few days. Thus the solver agent arouses people's attention. It can help people find answers quickly and help puzzle designers better understand players' behaviors during the game process, which contributes to improvements to game design and makes it more fascinating and challenging.

A. Puzzle Category

According to mechanisms, puzzles divide into two categories: puzzles with static and dynamic rules.

Puzzles with static rules mean that their domain knowledge stays consistent from level to level and will not change at any time during the playing process. One of the most classic examples is *Sokoban* because of its discrete and straightforward action, consistent rule system, and varying size and length flexibility to solutions.

Puzzles with dynamic rules are games where players' actions affect mechanic space. These changes may cause temporary or permanent effects on the rule system. Thus rules do not have to be initially made at the start of the game and can be manipulated at any point while the player attempts to solve the level. One example of this type is *Monument Valley* as shown in Fig.1 and Fig.2, which depends on optical illusion and transition between unreal structures to solve problems.

B. Algorithms for puzzle games

Because of the research and study of static rules over a long history, the solution space of traditional games has been comprehensively explored, and many efficient and high-performance algorithms have already existed. This type of agent consists of tree search algorithm, reinforcement learning, evolutionary algorithms, etc. [1]

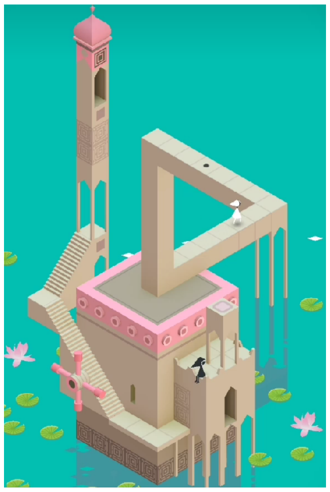


Fig. 1: Before Rotating

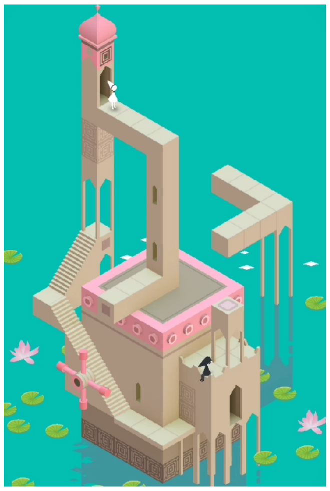


Fig. 2: After Rotating

However, since dynamic rule games have appeared in recent years, they have not been studied thoroughly. Some AI competitions exist for adaptive agents, like *Hanabi* (a multi-agent competitive strategy game) [2] and a 2D arcade-style fighting game [3]. However, the experience in such a field is still inadequate.

To allow innovative and efficient agents to emerge from this type of puzzle and provide insight into the design process and evaluation of these agents in an adaptive game environment, we choose one single game with dynamic rules as a research object and attempt to develop an intelligent agent for this game: *Baba Is Y'all*.

II. Background

Baba Is You is a puzzle game developed by Arvi Teikari originally for the 2017 Nordic Game Jam and then expanded to a full game with more mechanics. It has a similar game style like *Sokoban*, a player can control a character in the map to push blocks into a particular space to beat the level. However, some blocks may contain words and can form rules in the game. These rules can be created or broken at any time, and much of the game involves manipulating the rules in a certain order or orientation to allow the puzzle to be solved [4]. Fig.3 shows an example.

Besides the dynamic rule system, the most significant reason for choosing this game is that there has already been some flat research about this game, which offers some experience to study.

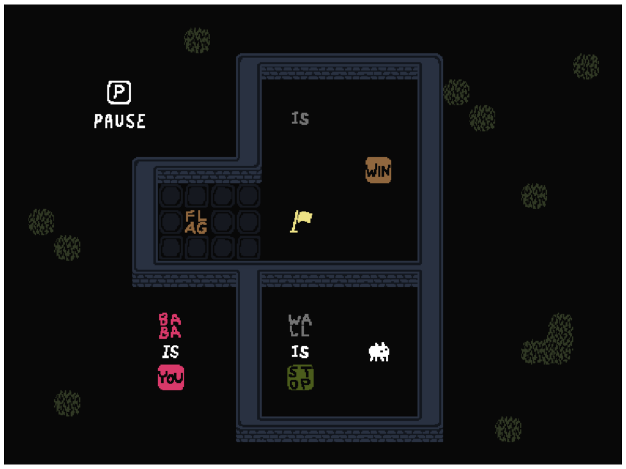


Fig. 3: A sample level in *Baba Is You*. Players have to break the active rule 'WALL-IS-STOP' and push the word blocks on the other side together, and create the rule 'FLAG-IS-WIN' to solve the level. [4]

A. *Baba is Y'all framework*

A level editor website called *Baba Is Y'all* allows players to make their levels, publish levels online, and rate levels. The high-performance levels in the website are good sources of test and debug material. We can also create our test maps to cover some extreme cases.

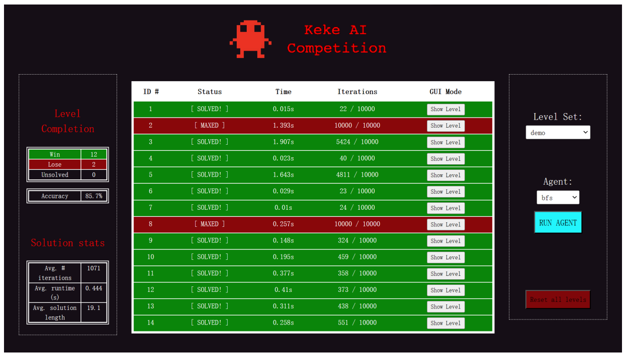
The integral version of *Baba Is You* contains hundreds of blocks and rules, which makes it hard to get started. This framework is the most original version instead of the latest version of the game. It only contains 11 object classes and 21 keyword classes and can form only nine formats of rules, which is a small subset of the full version of the game. Less complicated rules and core mechanisms make it the ideal material for algorithm design. [5]

B. *Keke AI Competition*

In IEEE Conference on Games 2022, a competition called "Keke AI Competition" adopts *Baba Is Y'all* framework and sets up an essential environment for programming.

The organizers reveal an open-source project containing a local website to evaluate the game [6]. The tests can fail or succeed, making it easy to verify the basic functionality of the agent and enter the improvement phase quickly. Some basic properties will also be shown on the website, like correctness rate, average time per solution, and the average length of solutions, all of which are significant figures for comparison and further improvement. One screenshot of the local evaluator is shown in Fig.4.

The project also provides a detailed GitHub wiki with a walk-through video to illustrate general ideas and variables in the environment, which saves time in reading and understanding code. Lastly, the organizers offer some valuable baseline agents in the project, which are the only agents for *Baba Is You* online and will be explained in IV.



Level Completion

Win	12
Lose	2
Unsolved	0

Accuracy	85.7%

Solution stats

Avg. # iterations	1071
Avg. runtime (s)	0.444
Avg. solution length	19.1

ID #	Status	Time	Iterations	GUI Mode
1	[SOLVED!]	0.015s	22 / 10000	Show Level
2	[MAXED]	1.393s	10000 / 10000	Show Level
3	[SOLVED!]	1.907s	5424 / 10000	Show Level
4	[SOLVED!]	0.023s	40 / 10000	Show Level
5	[SOLVED!]	1.643s	4811 / 10000	Show Level
6	[SOLVED!]	0.029s	23 / 10000	Show Level
7	[SOLVED!]	0.01s	24 / 10000	Show Level
8	[MAXED]	0.257s	10000 / 10000	Show Level
9	[SOLVED!]	0.148s	324 / 10000	Show Level
10	[SOLVED!]	0.195s	459 / 10000	Show Level
11	[SOLVED!]	0.377s	358 / 10000	Show Level
12	[SOLVED!]	0.41s	373 / 10000	Show Level
13	[SOLVED!]	0.311s	438 / 10000	Show Level
14	[SOLVED!]	0.258s	551 / 10000	Show Level

Level Set: demo

Agent: bfs

Fig. 4: The evaluator screen of the offline Keke AI Competition offline interface

TABLE I: Objects and Keywords

Object	**Keyword**
BABA	YOU
BONE	WIN
FLAG	STOP
WALL	MOVE
GRAN	PUSH
LAVA	SINK
ROCK	KILL
FLOOR	HOT
KEKE	MELT
GOOP	IS
LOVE	

III. Rules for *Baba Is Y'all*

Each level of the game initially contains particular objects and keywords, and players must push them to achieve winning conditions.

A. *Object and Keyword*

"Object" refers to the former word in the format "X-IS-Y"; the object can only be placed at the former for validation. Putting an object in the last place will not form a good rule.

"Keyword" refers to the latter word in the format "X-IS-Y", and the keyword can only be placed at the latter for validation. Similarly, putting the keyword in the former place will not form a good rule. Players can push all keywords under any rules.

Table.I shows all objects and keywords.

B. *Rule formats*

Rules for the level are defined by statements readable as English text and read from up, down, and left-right. So "X-IS-YOU" would be interpreted as a valid rule, but not "YOU-IS-X". As such, all rules must take one of these three formats [1] [7]:

- **X-IS-(KEYWORD)** where KEYWORD belongs to a word in the keyword class that manipulates the property of the game object class X (i.e. 'WIN', 'YOU', 'MOVE', etc.)
- **X-IS-Y** ($X \neq Y$) a transformative rule that changes all instances of sprites identified as X to the sprite Y.

- **X-IS-X** a reflexive rule that prevents any transformations occurring on the word block X. This differs from the previous rule X-IS-Y, as Y must differ from X for a transformation to occur. If a transformative rule is created, the X word block will not transform into Y with the reflexive tule active.

C. Detail Meaning

Table.II shows all detail keywords in the rules.

TABLE II: Objects and Keywords

Rule Type	Definition
X	In this table refers to any object class
X-IS-X	objects of class X can't be changed to another class
X-IS-Y	objects of class X will transform to class Y, unless "X-IS-X" exists
X-IS-PUSH	X can be pushed
X-IS-MOVE	X will move towards its facing direction each action, even if SPACE is pressed
X-IS-STOP	X will prevent the player from passing through it
X-IS-KILL	X will kill the player on contact
X-IS-SINK	X will erase any object contacting it, and will disappear with the contacted object
X-IS-HOT	X can only erase Y with "Y-IS-MELT", and will disappear with Y
X-IS-MELT	X can only erase Y with "Y-IS-HOT", and will disappear with Y
X-IS-YOU	All objects of X class are controlled by the player
X-IS-WIN	Player can win by touching X

D. Winning and Dead End

The only winning condition is any player contacts any winning object.

Since players must move to push and interact with environment, the game will enter a dead end if none of "X-IS-YOU" exists. Players cannot control anything on the map and must restart the level.

IV. Algorithms for *Baba Is Y'all*

Because of the timeliness and specificity of *Baba Is You*, there is almost no present algorithm discussing methods and ideas about its solution. The only well-performed algorithm is the best-first tree search solver in the game module of *Baba Is Y'all* framework, along with traditional search algorithms: BFS, DFS, and default agent - provided by organizers of "Keke AI Competition".

A. Best-First Tree Search

This algorithm is first developed by *Baba Is Y'all* framework, which is used to evaluate the quality of a user-made level.

The algorithm uses a heuristic function based on a weighted average of the Manhattan distance to the centroid distance for three groups: keyword objects, objects associated with the 'WIN' rule, and objects associated with the 'PUSH' rule. These three categories are chosen because of their critical importance in solving process: winning objects are required to complete the level, keyword objects are responsible for rules manipulation, and pushable objects can directly or indirectly affect the layout of a level map and accessibility to reach winning objects.

The following equation represents the heuristic function:

$$h = \frac{n + w + p}{3}$$

The meaning of each variable is shown in Table.III.

TABLE III: Variables in *Baba Is Y'all*

Variable	Definition
h	The final heuristic value
n	The minimum Manhattan distance from any player object to the nearest winnable object
w	The minimum Manhattan distance from any player object to the nearest word block
p	The minimum Manhattan distance from any player object to the nearest pushable object

B. Default Agent

The default agent in Keke AI Competition is almost the same as the above best-first tree search. The only difference is that the default agent takes the average of all possible Manhattan distances, instead of the minimal nearest Manhattan distance. The pseudocode of the primary process is presented in Algorithm 1.

TABLE IV: Variables of default agent in *Keke AI Competition*

Variable	Definition
h	The final heuristic value
n	The average Manhattan distance from any player object to any winnable object
w	The average Manhattan distance from any player object to any word block
p	The average Manhattan distance from any player object to any pushable object

C. BFS / DFS Agent

As two of the most classic graph search algorithms, once the game board is represented as a graph, it can be explored using either BFS (breadth-first search) or DFS (depth-first search) and find a solution. BFS is typically better suited for finding the shortest path to a solution, while DFS is better for finding any solution with generally more calculation resources.

However, it is worth noting that the search space for "Baba is You" can be quite large, especially for more complex levels, so using heuristics or other optimizations may be necessary to find a solution in a reasonable amount of time.

Algorithm 1 Default Agent

```
Input: $init\_state$ - initial game state;
Output: $solution$ - action list $[a_0, ..., a_n]$ to win the game;
  function ITERSOLVE($init\_state$)
      $curnode \leftarrow queue.shift()$
      $[score, children] \leftarrow getChildren(curnode)$
      for $child$ in $children$ do
          if $child$ is win then
              return $child.actions$
          end if
      end for
      $queue.push(children)$
      $queue.sort()$
  end function
```

D. Experiments

To better understand the performance and efficiency of each agent, all agents are run on the official full level sets with a constraint of 10,000 iterations and 10.0s. The experiment results are shown in Table.V.

TABLE V: Evaluation for distinct agents' performance in *Keke AI Competition* platform

Agents	**Accuracy**	**Avg. iterations**	**Avg. runtime (s)**	**Avg. solution length**
Default	60.9%	1899	4.243	14.2
BFS	59.2%	4562	3.694	23.9
DFS	52.2%	1839	5.192	376.3
Random	64.1%	4138	0.752	50.0

V. Conclusion and Future Improvement

The demand intelligent agents arises as the puzzle's complexity keeps increasing. Although many algorithms perform well in traditional puzzles with static rules, dynamic rule puzzles remain to be explored. Future tests on one of the last games, , will conduct based on the programming environment provided by Keke AI Competition.

In the future, other methods will be tested and attempt to exceed the best score in the Keke AI Competition, like Monte Carlo Tree Search. Properties, including the correctness and average time cost, will be compared between various algorithms to show the advantage of our algorithms.

References

[1] G. N. Yannakakis and J. Togelius, *Artificial intelligence and games*. Springer, 2018, vol. 2.

[2] R. Canaan, H. Shen, R. Torrado, J. Togelius, A. Nealen, and S. Menzel, "Evolving agents for the hanabi 2018 cig competition," in *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2018, pp. 1–8.

[3] F. Lu, K. Yamamoto, L. H. Nomura, S. Mizuno, Y. Lee, and R. Thawonmas, "Fighting game artificial intelligence competition platform," in *2013 IEEE 2nd Global Conference on Consumer Electronics (GCCE)*. IEEE, 2013, pp. 320–323.

[4] J. T. M Charity, Sarah Chen, "Keke AI competition," http://keke-ai-competition.com/, 2022.

[5] M. Charity, A. Khalifa, and J. Togelius, "Baba is y'all," http://equius.gil.engineering.nyu.edu/, 2022.

[6] J. T. M Charity, Sarah Chen, "Keke AI competition," https://github.com/MasterMilkX/KekeCompetition, 2022.

[7] M. Charity, A. Khalifa, and J. Togelius, "Baba is y'all: Collaborative mixed-initiative level design," in *2020 IEEE Conference on Games (CoG)*. IEEE, 2020, pp. 542–549.