Report on Bubble Blast Game

Prakhar Asthana*

Indian Institute of Technology Ropar
prakharas@iitrpr.ac.in

25/12/2013

Abstract

This report is about a game which goes by name BubbleBlast. The report presents and discusses the empirical data of the vaious approaches to solve this game.

I. Introduction

Bubbleblast is also known as *SameGame* and *Clickomania*. It was first invented by *Kuniaki Moribe* under the name *Chain-Shot!* in 1985. It was created again under name *SameGame* by *Eiji Fukumoto* in 1992[2]. In this report, author discusses the various approaches to solve this game.

II. Description of Game

Initial Game setup and rules are as under

1. This game is played on a rectangular grid of size *mxn* where m denotes number of horizontal rows and n denotes number of vertical coloumns.
2. Initialy each cell of grid has a *bubble* of some *color*.
3. *Color* of a *bubble* is denoted by an integer in interval $[1, k]$, $1 <= k <= m * n$. Here, different integer value denotes different *color* and vice-versa . k is maximum number of *colors* possible.
4. We define a position (i, j) on grid as cell at intersection of i^{th} row and j^{th} coloumn. $0 <= i < m, 0 <= j < n$.
5. A position (i, j) is called *empty* if the cell present at that position does not have a bubble. Color of such a cell is denoted by integer value 0. Initially all the cells are $non - empty$.
6. If all the cells in a vertical coloumn are empty it is called an *empty coloumn*.
7. A *block* is defined as a **maximal** set of **2 or more** bubbles of **same** color such that it is possible to visit any bubble to any other bubble in block following 4- neighbourhood connectvity.
8. *Size* of a block is defined as no. of bubbles in that block.
9. A bubble is in *singleton* state when it does not belong to any block.
10. Deleting the block means removing all the bubbles belonging to that block from the grid.
11. A *blast* means deleting a block from the grid and after that making changes in

*Entry No:2011cs1027, B.tech 3rd Year, Computer Science and Engineering

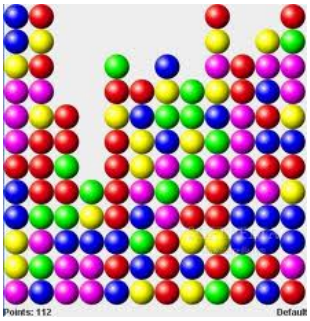
the grid following rules of *vertical* and *horizontal shift* **inorder** defined as under:

(a) Vertical Shift : When a bubble is deleted from the grid, all the bubbles above that bubble are shifted down and the topmost position is padded with an empty cell.

(b) Horizontal Shift : After performing *Vertical Shift*, if a coloumn becomes empty then the all the coloumns to the left of it are shifted right and the leftmost coloumn is padded with an empty coloumn.

12. Each blast generates a *score* which is defined as square of the size of the block deleted in that move.

13. Game ends when all the bubbles are blasted or there are no further blocks possible i.e. all remaining bubbles are in singleton state.



Our aim is to blast the bubbles in a way to obtain maximum total score.

III. Complexity

1. It has been proven that the problem of determining that whether a given grid can be completely emptied or not is NP-complete[3].

2. Further it is also proved the problem of finding the sequence of blasts which results in maximum total score **independent of its scoring function** is also NP-complete[2].

Further in the report the author discusses various approaches that can be followed to tackle a given grid.

IV. Various Approaches

1. Always pick a random non-empty non-singleton position and blast the bubble block at that position.

2. Always blast the **minimum** size block of the **least** frequent color if possible. Move to higher frequency colors inorder if all the bubble of less frequent colors are in singleton state or have already been blasted. This is known as *TabuColorRandom* strategy[2]. Any ties are broken arbitrarily.

3. Starting from least frequent color and moving to higher frequency colors inorder, pick the color with least frequency whose block is possible. If the color lies in bottom $1/3^{rd}$ of color array sorted by their frequency then blast the maximum sized block of that color. Otherwise, blast the minimum sized block. Any ties are broken arbitrarily.

4. Always blast the **maximum** size block of the **least** frequent color if possible. Move to higher frequency colors inorder if all the bubble of less frequent colors are in singleton state or have already been blasted. Any ties are broken arbitrarily.

5. Always blast the **maximum** size block of the **most** frequent color if possible. Move to lower frequency colors inorder if all the bubble of more frequent colors are

in singleton state or have already been blasted.Any ties are broken arbitrarily.

V. Results

All the approaches were tested over various inputs of various sizes and colors. Then the mean and standard deviation were calculated as under

Table 1: *Mean and standard deviation of total score*

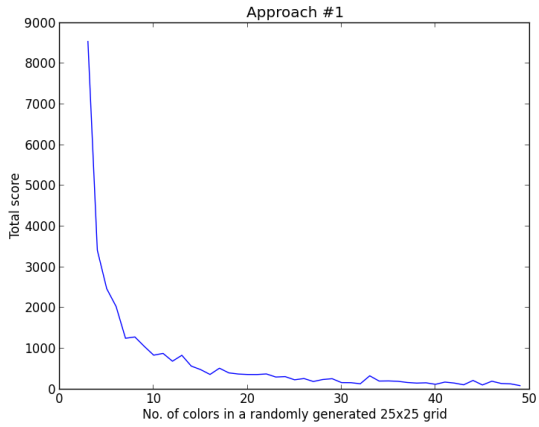
Approach	Mean	Standard Deviation
1	3554.235	359.254
2	20719.598	5491.497
3	20348.544	5658.565
4	9276.280	5667.464
5	3637.266	302.88

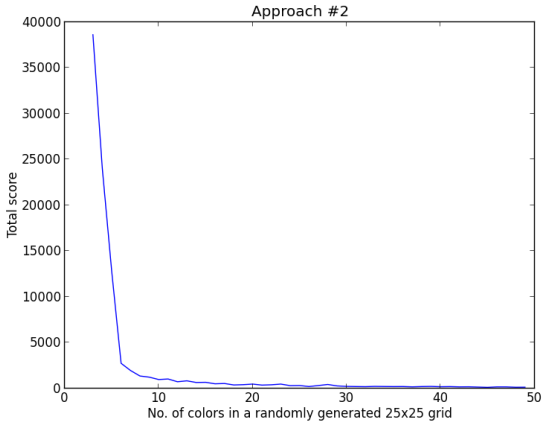
As clear from the emiprical data, approach #2 and #3 give almost similar results but they are stringingly better than all other approaches. Reason for this is that approach # 2, #3 (and #5 too) tries to build up block size of most frequent color and other more frequent colors over time. This results in an increase in *Average Block Size*(see next page). Since our score function is quadratic in nature, it resuls in great increase in score due to fact

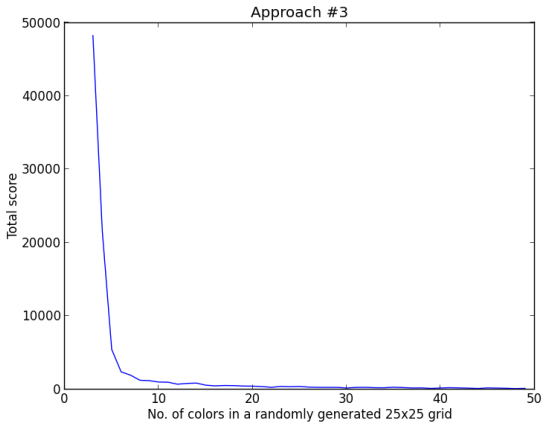
$$(a+b)^2 \geq a^2 + b^2 \tag{1}$$

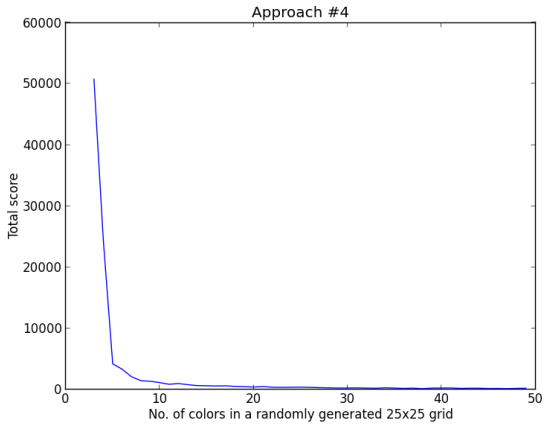
Thus keeping big blocks intact and blasting them later is better than blasting the block in form of various chunks as it gives higher score.

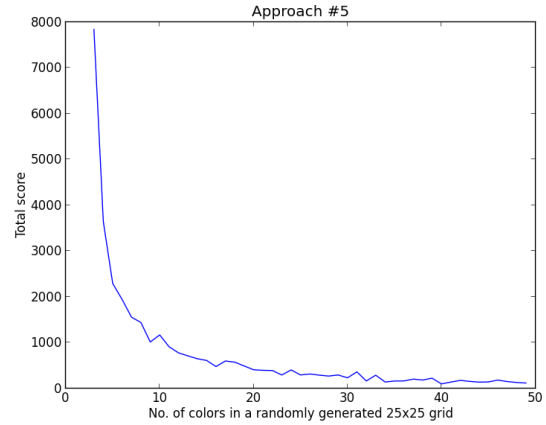
Figures below show the performance of all the approaches on a 25X25 grid with varying no. of colors. As clear from the figure too approach #2 and #3 give best results among all.











Score V number of colors ; Grid Size=25x25

Let us define *Average Block Size* for a given grid as under

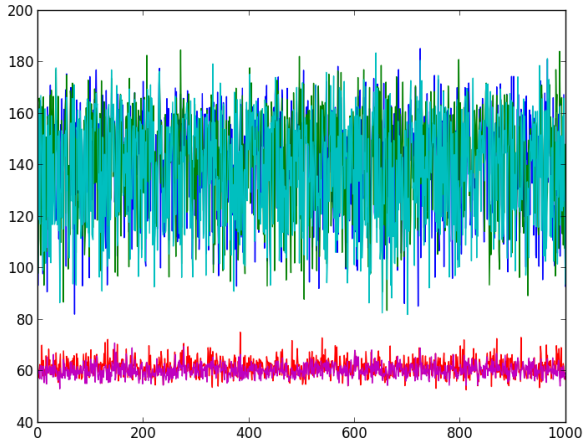
$$AverageBlockSize(ABS) = \sqrt{TotalScore} \quad (2)$$

Grids varying in size and number of colors were given as input to all the approaches and the mean and standard deviation of ABS is as under.

Table 2: *Mean and standard deviation of ABS*

Approach	Mean	Standard Deviation
1	61.414	3.304
2	142.92	19.972
3	141.139	20.689
4	137.250	20.943
5	60.258	2.495

As clear from the table, approach #2 and #3 have highest ABS and this is the reason they generate higher scores as compared to other approaches.
Figure below shows variation in ABS(y-axis) over 1000 different input grids for all 5 approaches



Red-Approach#1; Green-Approach#2; Blue-Approach#3; SeaBlue-Approach#4; Purple-Approach#5

As clear form the figure above approach #2, #3 and #4 has higher ABS values(100-160) while approach #1 and #5 have lower ABS values(60-70).

VI. Other Approaches

Classical approach of solving puzzles are A^* (removing the node from the front of the queue and storing all its children in the sorted order

of the *key* at the back of the queue till goal state is arrived) and IDA^* (iterative depth-first version of A^*). Here *key* is the value of evaluation function evaluated at that node.
But these methods fail work in our problem because it is not easy to find the evaluation function for Bubbleblast game.[2]

Other techniques are Beam search and Depth Budgeted search[1]. In Beam Search a *beam* of best B nodes is kept for each iteration. If in the process, a leaf node is reached it is checked whether it is better than the current best solution so far. Its complexity is $O(B * d)$ where d is the maximum depth reached.
In Depth Budgeted search each node is allowed to explore the tree below it. Initial budget to each node is assigned through Banker's Algorithm[1]. When the all the assigned budget gets used up, a greedy approach is followed to arrive at the solution.

So far the best known approach is **Monte Carlo with Roulette-Wheel Selection (MC-RWS)** , which performs the Monte Carlo simulation on each node and chooses the one which is evaluated higher by evaluation function as defined in [1]

VII. References

1. Takes, Frank W., and Walter A. Kosters. "Solving SameGame and its chessboard variant." Proceedings of the 21st Benelux Conference on Artificial Intelligence (BNAICâĂŹ09)(eds. T. Calders, K. Tuyls, and M. Pechenizkiy). 2009.

2. Schadd, Maarten PD, et al. "Addressing NP-complete puzzles with Monte-Carlo methods." Proceedings of the AISB 2008 Symposium on Logic and the Simulation of Interaction and Reasoning. Vol. 9. 2008.

3. Biedl, Therese C., et al. "The complexity of Clickomania." More games of no chance 42 (2002): 389.