UDC 004.942:004.42

ESTIMATION OF EFFICIENCY OF PATHFINDING ALGORITHMS IN DISCRETE LABYRINTH BASED ON COLLECTED STATISTICS DATA AND SOFTWARE SIMULATION

Krasnov E.S., Bagaev D.V.

Kovrov State Technological Academy named after V.A. Degtyarev

This article continues the previous study of the authors on development the strategy of analysis of the most common algorithms for pathfinding in discrete labyrinths using the statistical data collector software. The recent experimental data are reviewed and summarized. The common structure of the experiment, its stages, data collecting methods and data processing strategies are described. The 8-directional wave algorithm was shown to be the best one among the most popular algorithms. It is fast enough, consumes little memory and provides a short and smooth route in the discrete labyrinth.

KEY WORDS: discrete labyrinth, pathfinding, algorithms, statistical data, software.

ОЦЕНКА ЭФФЕКТИВНОСТИ АЛГОРИТМОВ ПОИСКА ПУТИ В ДИСКРЕТНОМ ЛАБИРИНТЕ, ОСНОВАНЫХ НА СБОРЕ СТАТИСТИЧЕСКОЙ ИНФОРМАЦИИ И КОМПЬЮТЕРНЫХ РАСЧЕТАХ

Краснов Е.С., Багаев Д.В.

Эта статья продолжает предыдущее исследование авторов по разработке стратегии анализа наиболее распространенных алгоритмов поиска пути в дискретных лабиринтах с помощью программного обеспечения для накопления статистических данных. Приведен обзор и анализ результатов последних экспериментов. Описаны общая структура эксперимента, его этапы, методы сбора и обработки данных. Показано, что волновой алгоритм работы в восьми направлениях является лучших среди самых популярных алгоритмов. Он достаточно быстрый, использует немного памяти, и строит короткий и гладкий путь в дискретном лабиринте.

КЛЮЧЕВЫЕ СЛОВА: дискретный лабиринт, поиск пути, алгоритмы, статистические данные, программное обеспечение.

ОЦІНКА ЕФЕКТИВНОСТІ АЛГОРИТМІВ ПОЗШУКУ ШЛЯХУ В ДИСКРЕТНОМУ ЛАБІРИНТІ, ЗАСНОВАНИХ НА ЗБОРІ СТАТИСТИЧНОЇ ІНФОРМАЦІЇ ТА КОМПЮТЕРНИХ РОЗРАХУНКАХ

Краснов Є.С., Багаєв Д.В.

Ця стаття продовжує попереднє дослідження авторів з розробки стратегії аналізу найбільш поширених алгоритмів пошуку шляху в дискретних лабіринтах за допомогою програмного забезпечення для накопичення статистичних даних. Наведено огляд і аналіз результатів останніх експериментів. Описані загальна структура експерименту, його етапи, методи збору та обробки даних. Показано, що хвильовий алгоритм роботи у восьми напрямках є кращим серед найпопулярніших алгоритмів. Він досить швидкий, використовує небагато пам'яті, і будує короткий і гладкий шлях в дискретному лабіринті.

КЛЮЧОВІ СЛОВА: дискретний лабіринт, пошук шляху, алгоритми, статистичні дані, програмне забезпечення.

1. Introduction. Statistical data are collected via the special simulation software described in [1]. The usability and interface of the software has been improved. The software, used in the experiment, is meant to be run on Microsoft Windows platform. It is a sort of «sandbox» for creating two-dimensional discrete labyrinths (also maps or mazes; for more detailed description see [1]) and manipulating with them. Also, it can perform different pathfinding algorithms on the created map (see [1] for the list of used algorithms), collecting the required statistical data. The main features of the software are the following:

- creation of two-dimensional discrete labyrinths via simple built-in graphics editor, similar to Microsoft Paintbrush, allowing to draw the obstacles and saving them to/loading them from file;
- running selected algorithm on the drawn map with displaying the result route, obtained via this algorithm;
- sequential running of all built-in algorithms with collecting the statistical data (see [1] for the list of the collecting values) and its saving for further analysis;

- viewing and analyses of the statistical data and exporting for further processing in the third party software.

The modified and improved interface of the simulation software, that is displaying the middle-sized map (filled for 42% with obstacles) and a result route, built with A-Star (A*) algorithm is presented in fig.1.

Let's introduce a notion: test (session) – it is single run of all of the built-in pathfinding algorithms on the current map (labyrinth), with measuring all the required values and saving the collected data. The test (session) could be run by pressing the «Run all algorithms» button.

To obtain a reliable time value of the algorithm run, the software needs to repeat the run of every algorithm multiple times (from 1 to 500 times; it is a user-defined value). It enables to obtain the average value of the time value.

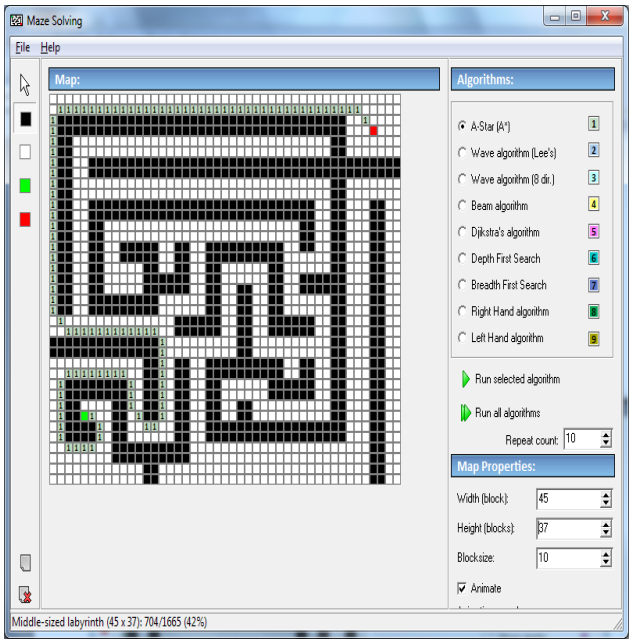

Fig. 1. Software graphic interface with middle-sized map example.

Running of all algorithms was initially programmed to take the special order, for avoiding influence of the processor's cache memory mechanism (it could give erroneous values). Such a thing could happen if the algorithms are run in the following order: 1, 1, 1, 1, … 2, 2, 2, 2, … 3, 3, 3, 3, 3… and so on. To avoid this, the next order was used: 1–2–3–4…, 1–2–3–4… and so on, where 1–9 are numbers of the algorithms. Nevertheless, the numerical experiment showed, that it doesn't actually happen, and moreover, the selected order makes impossible to measure the time for some algorithms (due to their extremely high speed). For example, our measurements of single run of the classic wave algorithm always resulted in zero milliseconds. So, the order of algorithms' running was restored, and the full time of full repeat cycle for every algorithm was measured. Later it was divided at by the number of the repetitions.

After finishing the session, the collected data were saved for further analyses (single algorithms runs are ignore, see explanation below). The viewing of the collected data and its partial analysis can be done via statis form window, displayed in fig.2.

The main features of the stats module:

- collecting and keeping the statistical data;
- calculation of the normalized and non-normalized characteristic values (see [1]) and the F value (estimation of the algorithm, also described in [1]), calculated with user-specified weighting coefficients k_T, k_L, k_M, k_H, k_A;
- selection of the partial data by the following criterions:
  - session's date;
  - map's filename;
  - width, height and type of the map (by size);
  - map's occupancy;
- data export (selected test or the whole amount) to the HTML-file (for more comfortable information representation, its further editing/formatting and transfer into third party software).

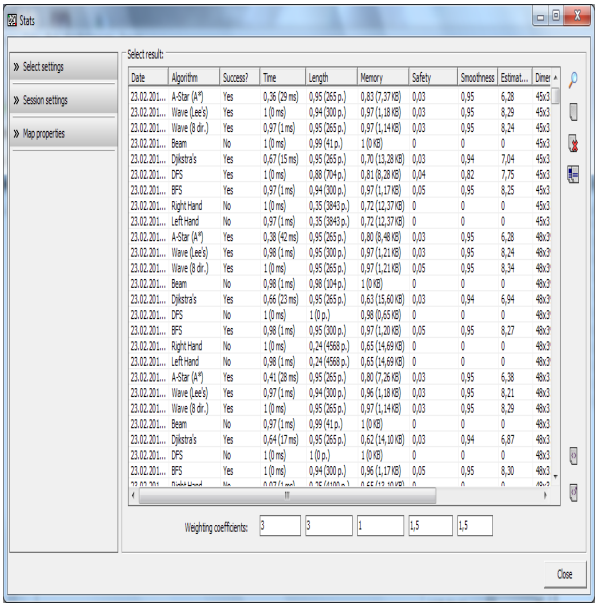

Fig. 2. Stats viewing window.

Selection of the partial data by the described criteria is designed to filter the whole test; so, for example, it's impossible to view the results for A* algorithm only. These constraints are created to avoid the improper characteristic values calculation (they have sense only as relative values, not as absolute ones). With some test data being deleted the software will calculate the wrong values, what is unacceptable and the tests can be deleted or filtered as a whole only.

2. Description of methods of experiments. The only improvement of the numerical methods is the following: the points of the map, close to map's edge are considered as hazardous during calculating the *H* value.

For collecting all required data, a special method has been invented.

The main map characteristics are its size and occupancy by the obstacles. As one can see in [1], there is the following classification of maps made (three occupancy-types):
– poorly-filled (less than 10% of coverage by obstacles);
– medium-filled (more than 10% and less than 40% of coverage by obstacles);
– strong-filled (more than 50% coverage by obstacles).

and four types by size:
– small (about 10x10 blocks);
– medium (from 40x40 up to 60x60 blocks);
– large (about 100x100 blocks);
– huge (about 200x200 blocks or more).

In that way it is necessary to run at least 12 groups of tests (3 x 4), which enables to make a conclusion about algorithms' efficiency in each group. The number of tests (sessions) in each group is about 100. Every test is run on the map with a little modified size, occupancy and structure.

After running all the tests, the average value of *F* (see [1]) is calculated for every group, which allows making a number-supported conclusion about each algorithm's efficiency.

The F value can be calculated by the following equation:

$$F_{alg} = k_T \cdot T_N + k_L \cdot L_N + k_M \cdot M_N + k_H \cdot H_N + k_A \cdot A_N, \quad (1)$$

which is described in details in [1].

The following weighting coefficients have been used during the experiments:

$$k_T = 3,\ k_L = 3,\ k_M = 1,\ k_H = 1.5,\ k_A = 1.5,$$

which limit the *F* value to the range from 0 to 10 (see [1]). Time and route-length characteristics values T_N and L_N are the most important, while the M_N value have the lowest weighting coefficient, since it is the least significant.

Also, the additional batch of tests was run to answer the question, posed in [1] – which algorithm is better: the right-hand algorithm or the left-hand one? Since both algorithms can find a route only if the end-point is close to the wall, the test were run with such a condition, taken into the account (unlike to the main series of tests, in which it wasn't mentioned for avoiding the improper high *F* values for these algorithms).

3. Experimental results. In this section the main results of the stat data analyses are given (without listing all the processed data due to its huge size and unobviousness). All calculations have been made in Microsoft Excel ®.

The results are given as a table 1 which has 12 sections (one for every group of tests), divided into a smaller blocks, that represent average F values for each algorithm. The «best» algorithms are highlighted. Note, the following designations are made:

- A* – A-Star algorithm;
- Wave – Wave algorithm (Lee's);
- Wave8 – Wave algorithm (8 directional);
- Beam – Beam algorithm;
- D – Djikstra's algorithm;
- DFS – Depth First Search;
- BFS – Breadth First Search;
- R – right-hand algorithm;
- L – left-hand algorithm.

Table 1. Results of the primary series of tests.

	Poorly-filled		Medium-filled		Strong-filled	
Small	A*	6,82	A*	6,41	A*	6,05
	Wave	8,16	Wave	8,01	Wave	7,53
	Wave8	**8,27**	**Wave8**	**8,16**	**Wave8**	**7,72**
	Beam	1,53	Beam	0,97	Beam	0,53
	D	6,54	D	6,46	D	6,44
	DFS	4,32	DFS	4,16	DFS	3,05
	BFS	8,17	BFS	8,02	BFS	7,57
	R	5,62	R	3,71	R	7,2
	L	5,78	L	3,72	L	7,24
Medium	A*	6,88	A*	6,71	A*	6,5
	Wave	**8,68**	Wave	7,71	Wave	7,67
	Wave8	8,59	**Wave8**	**7,86**	**Wave8**	**7,84**
	Beam	2,34	Beam	0,32	Beam	0
	D	7,35	D	5,99	D	6,05
	DFS	8,12	DFS	0,92	DFS	0,56
	BFS	8,41	BFS	7,7	BFS	7,68
	R	6,2	R	5,69	R	5,55
	L	6,44	L	5,81	L	5,49
Large	A*	6,22	A*	6,18	A*	6,06
	Wave	**8,14**	**Wave**	**8,06**	Wave	7,98
	Wave8	8,13	**Wave8**	**8,06**	**Wave8**	**8,05**
	Beam	3,65	Beam	0,11	Beam	0
	D	6,69	D	6,58	D	6,33
	DFS	1,23	DFS	0,42	DFS	0
	BFS	8,04	BFS	8,02	BFS	7,97
	R	5,65	R	5,8	R	5,78
	L	5,74	L	5,76	L	5,81
Huge	A*	6,51	A*	6,31	A*	6,2
	Wave	8,04	Wave	8,02	Wave	7,98
	Wave8	**8,57**	**Wave8**	**8,15**	**Wave8**	**8,14**
	Beam	1,03	Beam	0,05	Beam	0
	D	7,04	D	6,58	D	6,48
	DFS	0	DFS	0	DFS	0
	BFS	8,31	BFS	8,11	BFS	8,09
	R	5,41	R	5,43	R	5,45
	L	5,50	L	5,49	L	5,7

The results of the secondary series of experiments are given in table 2.

Table 2. Results of the secondary series of tests.

	Poorly-filled		Medium-filled		Strong-filled	
Small	R	7,26	R	7,81	R	7,31
	L	7,84	L	7,83	L	7,26
Medium	R	7,35	R	7,69	R	7,64
	L	7,22	L	7,81	L	7,42
Large	R	7,46	R	7,62	R	7,88
	L	7,71	L	7,56	L	7,73
Huge	R	7,33	R	7,4	R	7,68
	L	7,44	L	7,32	L	7,59

Note: the following designations are made:

- R – right-hand algorithm;
- L – left-hand algorithm;

4. Conclusions. According to the experimental results, the following alignment takes place:

1. A* and Djikstra's algorithms have the lowest performance and require more memory than other algorithms, but the route provided by them is often the shortest.
2. Classic Wave algorithm (Lee's) is faster than the 8-directional one (about 30–60% faster), but as for their great speed it doesn't make any real difference. At the other hand, the route, provided by the 8-directional Wave algorithm is much smoother with lower number of turns. Also mostly it is as short as the A* and Djikstra's ones.
3. Beam algorithm has extremely high fail rate, so it can be used only for a few situations and studying examples, with the simple maps only. Besides, it has the great performance rate. Using it with real problems is mostly inappropriate.
4. Depth-first search cannot reach the end-point in big labyrinths due to overflowing the stack in recursion. It is faster even than both Wave algorithms, but it can be used for large maps.
5. Breadth-first search is comparable to the 8-directional Wave algorithm, but provide a little bit longer routes. In that way, it is better than the classic Wave algorithm.
6. The right/left hand algorithms are the fastest (in the case of success), but they have about 50% of failing rate. The secondary series of tests cleared that these algorithms are equal in common.

So the 8-directional wave algorithm seems to be the best one. It is fast enough, consumes little memory and provides a short and smooth route, which is medium-safe.

Thus the main questions, posed in [1], have been answered. The results are statistically reliable and there's only one «winner».

REFERENCES

1. Krasnov E.S. Strategy of analyzing most common algorithms for path finding in discrete labyrinth using software statistic data collector. *East-West Design & Test Symposium (EWDTS 2012)*. – 2012. – P. 34–41.