VERSITA
Information Technology and Management Science
doi: 10.2478/v10313-012-0004-9
2012 /15

The Analysis of Efficiency Dependence of the Shortest Path Finding Algorithms A* and HPA* on the Grid Size

Imants Zarembo[1], Oleg Uzhga-Rebrov[2], [1-2]*Rezekne Higher Education Institution*

***Abstract*– Many pathfinding problems in real-world applications require real-time computation of the shortest path between two points in a grid-based environment. It is not a trivial task. While some shortest path finding algorithms may perform admissibly in one condition, they may prove inadmissible in other conditions. HPA* pathfinding algorithm is faster and more memory efficient than the A*algorithm in relatively large two-dimensional grids, but this advantage may not apply to very small or very large grids. The paper deals with the efficiency of A* and HPA*in two-dimensional grids of different sizes. For the sake of completeness of the analysis, HPA* efficiency is measured taking into consideration the number of hierarchy levels and different cluster sizes. Both algorithms have been implemented and tests conducted. Experimental evidence is proposed to demonstrate the algorithm efficiency in various conditions.**

***Keywords*– A*, HPA*, shortest path problems**

I. Introduction

The finding of the shortest route between two points in the smallest amount of time is an important issue in various industries, starting from navigation systems, artificial intelligence and robotics and ending with video games. There are many different pathfinding algorithms and approaches intended for various needs. A* is a universal algorithm used in graph traversal and pathfinding. HPA* (*Hierarchical Pathfinding A**) is a version of A* that uses hierarchical approach to reduce the complexity of a problem. Reduced search effort using HPA*, compared to A*, results in shorter execution time even in the worst-case scenarios.

Within the framework of the present research, pathfinding algorithms A* and HPA* are implemented, and the analysis of their efficiency is conducted in the environment based on a two-dimensional grid. Such factors as cluster size, the number of hierarchy levels and grid size are taken into consideration while performing experiments.

The main reason for performing the efficiency analysis of pathfinding algorithms A* and HPA* is determined by the fact that there is still not enough information about HPA* because HPA* is a relatively new algorithm [7] and is considerable improvement over A*.

II. Methodology

Within the framework of the research, experiments have been conducted to find the shortest path, using A* and HPA*, as well as compare the elapsed time and the number of traversed nodes. Each experiment has been conducted 100 times to reduce the number of random errors.

To prepare a two-dimensional grid for HPA*, the initial grid was split into clusters. All grids used in experiments contained two types of nodes: blocked and passable. 20% of grid was randomly filled with blocked nodes. Path start and goal nodes were chosen randomly.

All experiments were conducted using a computer with CPU running at 2.8 GHz.

Manhattan distance was chosen as a heuristic function, because it is strictly grid-based distance:

$$H = |x_1 - x_2| + |y_1 - y_2|. \quad (1)$$

Algorithms were implemented assuming that pathfinding might occur only horizontally or vertically, with no diagonal movement.

III. Algorithm A*

A* is a pathfinding algorithm used for finding an optimal path between two points called nodes. A* uses best-first search to find the lowest cost path between start and goal nodes. The algorithm uses a heuristic function to determine the order in which to traverse nodes. The heuristic is sum of two functions:

- G — the exact cost of the path from the initial node to the current node;
- H — the admissible (not overestimated) cost of reaching the goal from the current node;
- $F = G + H$ — the cost to reach the goal if the current node is chosen as a next node in the path.

The estimated heuristic cost is considered admissible, if it does not overestimate the cost to reach the goal [10].

The selection of heuristic function plays an important role in ensuring the best performance of A*. Ideally H is equal to the cost necessary to reach the goal node. In this case, A* would always follow a perfect path and would not waste time traversing unnecessary nodes. If the overestimated value of H is chosen, the goal node is found faster, but at the expense of optimality. In some cases, it may lead to situations, where the algorithm fails to find a path at all, despite the fact, that the path exists. If the underestimated value of H is chosen, A* will always find the best possible path. The smaller H is chosen, the longer it will take for the algorithm to find a path. In the worst-case scenario, A* provides the same performance as Dijkstra's algorithm [9].

A* starts its work by creating two node lists — a closed list containing all traversed nodes and a list of nodes that are being

considered for inclusion in the path. Every node contains three values: *F*, *G* and *H*. In addition to these three values, every node needs to contain information about the node, which precedes it, to determine a path, by which this node can be reached.

IV. ALGORITHM HPA*

Hierarchical pathfinding A* was developed by Adi Botea and his colleagues in 2004. HPA* is a near-optimal pathfinding algorithm; it finds paths that are within 1% of optimal [7]. It is the combination of pathfinding and clustering algorithms, which works by creating an abstract graph on the basis of two dimensional grids. Main HPA* principle is based on dividing the problem into several smaller subtasks, and caching results for each segment of the path.

Clusterization, used in this algorithm, is relatively simple: a low resolution two-dimensional grid $s \times s$ is created, where s is a new grid size. New grid is placed directly above the initial grid. Each node in a new grid becomes a cluster, all initial grid nodes that are located under a corresponding cluster are considered to be members of this cluster. Each cluster is considered a separate graph. An abstract graph is then created to connect all separate graphs. To achieve that border nodes are found between neighbouring clusters, nodes that are on the outer side of the cluster are checked. If the node has a passable neighbour in an adjacent cluster, it is considered connected, and connection between two graphs representing clusters are added to an abstract graph. In cases, where there are many adjacent connections between two clusters, they are combined into one entrance. Entrances are then added to an abstract graph and connected. Abstract graph still lacks internal edges (paths between entrances inside one cluster). These edges are created by running A* algorithm through each node in a separate cluster. If A* finds a path, its cost becomes the costs of the found abstract edge, otherwise the edge is not added to the abstract graph. Inter-cluster edges inherit their costs from the initial graph edge cost. Finally, the abstract graph is ready for pathfinding using A*.

HPA* pathfinding consists of two stages called pre-processing and online search. During pre-processing, the start and goal nodes are inserted into the abstract graph, and inter-cluster edges are added. Then A* is used in the abstract graph to find the shortest route. During online search, the shortest route found in the abstract graph is refined to a full path in the initial graph using A*. To find a full path from the start to goal node, A* is used in each cluster on nodes that connect clusters. Finally, the partial results from each cluster are combined into a full path.

V. PATHFINDING RESULTS

Comparison of algorithm efficiency was determined by applying them to static two-dimensional grids of various sizes.. Grid sizes (64x64, 128x128, 256x256, 512x512 and 1024x1024) were chosen to assess the efficiency of pathfinding algorithm in small grids and in large grids.

Various cluster sizes (16, 32 and 64) were used with HPA* to determine cluster size impact on the efficiency of the algorithm. The smallest cluster size was 1/4 of the smallest grid size.

Experimental results have shown that under the given searching conditions, HPA* is faster than A* for each grid of various sizes. In smaller grids (64x64), difference in the elapsed time is small; advantage of HPA* increases when sizes of the grid grow larger. HPA* performance also increases adding more hierarchy levels from one to three. Again, the greatest performance advantage is achieved with large-size grids.

HPA* cluster size impact on overall performance, in the conducted experiments, is relatively small.

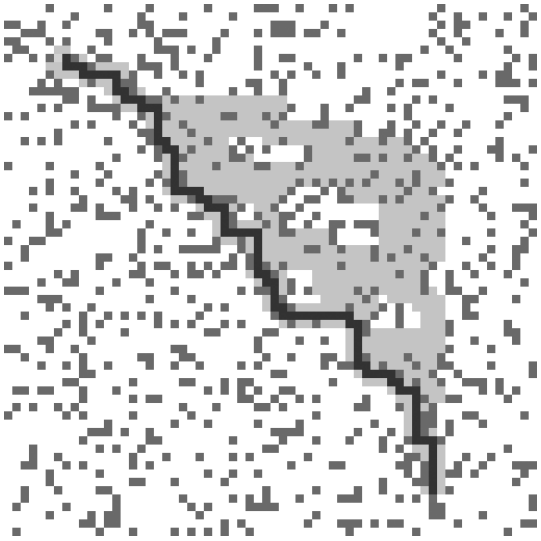

Fig. 1. A* pathfinding result in a 64x64 node grid

Figure 1 shows the end result of pathfinding in a 64x64 node two-dimensional grid using A* algorithm. The found path is depicted by a black line; light gray areas indicate the visited nodes.

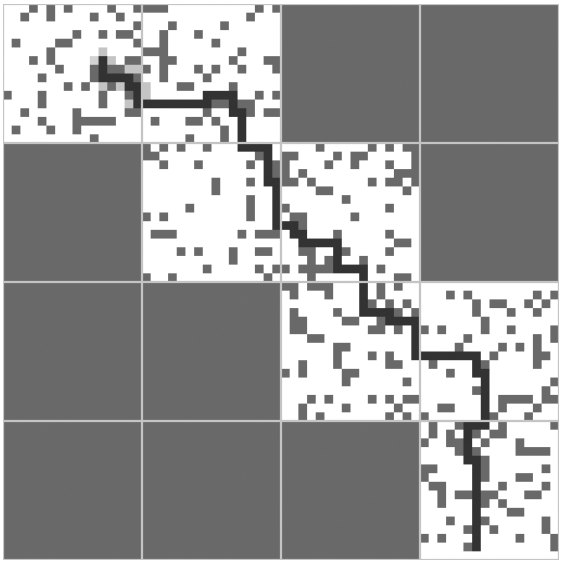

Fig. 2. HPA* pathfinding result in a 64x64 node grid

Figure 2 shows the end result of pathfinding in a 64x64 node two-dimensional grid using HPA* algorithm. The found

path is depicted by a black line; light gray areas indicate the visited nodes. Dark gray areas represent clusters that were disregarded for final search.

VI. Execution Time

Table I shows the execution time results for finding the shortest route. Since A* uses no abstraction, it contains zero levels of abstraction (L0). HPA* uses one to three levels of abstraction (L1 – L3). Only HPA* uses clusterization; therefore, A* does not have a cluster size associated with it.

TABLE I

Pathfinding Length, ms

Algorithm	Abstraction levels	Cluster size	Grid size				
			64	128	256	512	1024
A*	0	-	6	38	281	2940	36737
HPA*	1	16	6	14	58	434	4127
	2	16	6	20	52	216	1367
	3	16	6	20	98	204	809
	1	32	3	14	55	386	4401
	2	32	3	14	52	217	1276
	3	32	3	15	52	190	775
	1	64	2	10	48	287	4094
	2	64	2	10	50	201	1228
	3	64	2	10	51	205	756

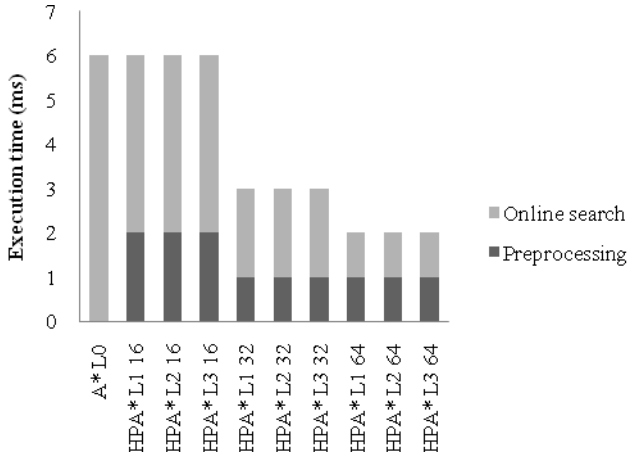

Fig. 3. Pathfinding results in a 64x64 node grid

Figure 3 graphically shows the algorithm execution times in a 64x64 node grid. A* and HPA* with one abstraction level find the path in 6 ms. Increasing the number of HPA* abstraction levels results in faster execution times; it takes less time to execute both online search and pre-processing phases. In a 64x64 grid, HPA* cluster size does not impact executions times.

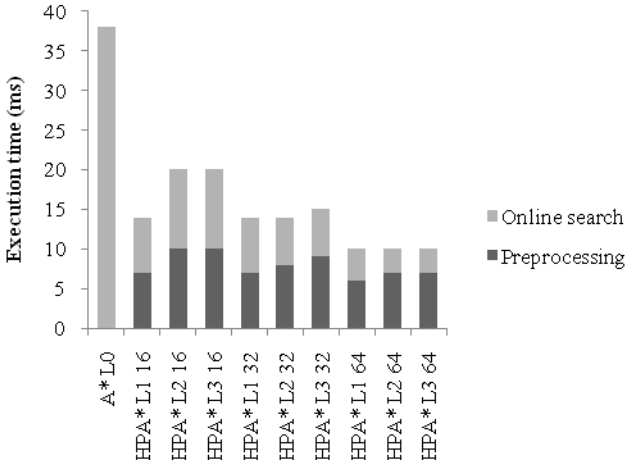

Fig. 4.Pathfinding results in a 128x128 node grid

Figure 4 graphically shows the algorithm execution times in a 128x128 node grid. For this grid size, HPA* is faster than A* in all cases. The diagram shows that in this particular case HPA* takes less time to find the shortest route, when using a larger cluster size.

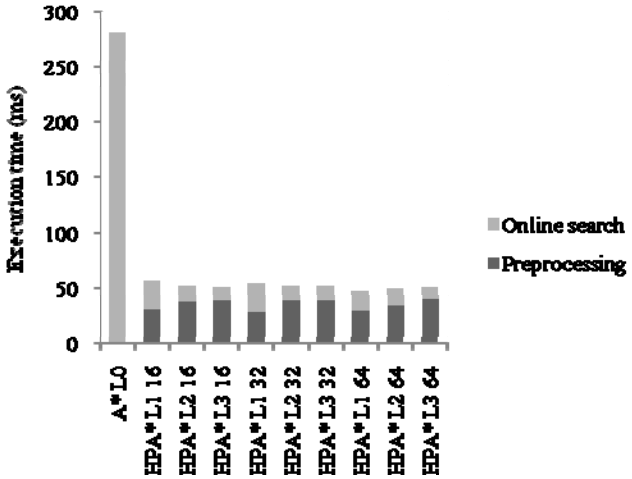

Fig. 5. Pathfinding results in a 256x256 node grid

Figure 5 graphically shows the algorithm execution times in a 256x256 node grid. HPA*, in all cases, finds the shortest route considerably faster than A*. The diagram shows that HPA* pre-processing phase takes more time than online search. In a 256x256 grid, the cluster size changes have no noticeable impact on overall performance.

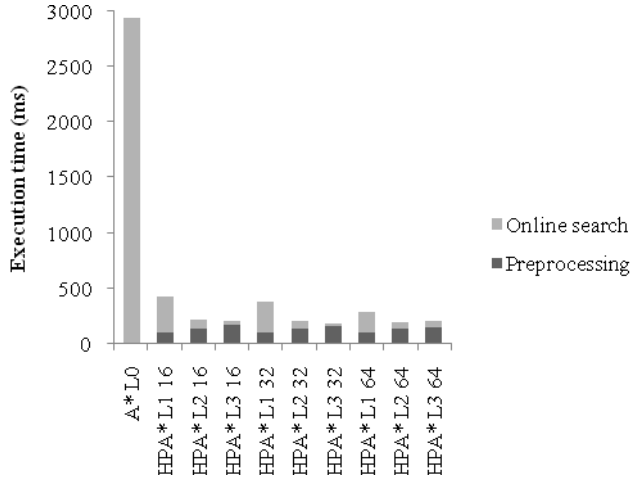

Fig. 6. Pathfinding results in a 512x512 node grid

Figure 6 graphically shows the algorithm execution times in a 512x512 node grid. HPA*, in all cases, finds the shortest route considerably faster than A*. The diagram shows that HPA* pre-processing phase length increases with the number of abstraction levels, but online search length decreases noticeably.

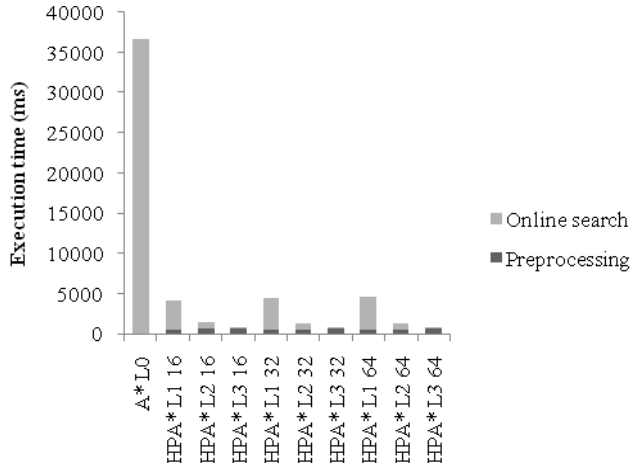

Fig. 7. Pathfinding results in a 1024x1024 node grid

Figure 7 graphically shows the algorithm execution times in a 1024x1024 node grid. HPA*, in all cases, finds the shortest route considerably faster than A*.The diagram shows that HPA* pre-processing phase length increases with the number of abstraction levels, but online search length decreases noticeably.

VII. The Number of Traversed Nodes

Table II shows the number of nodes traversed in the experiment. With the increase in size of two-dimensional grid, the number of traversed nodes increases considerably.

Experimental data shows that HPA* traverses fewer nodes than A* in all cases to find a path between the start and goal nodes. The number of traversed nodes decreases with the increased number of HPA* abstraction levels.

TABLE II
TRAVERSED NODES

Algorithm	Abstraction levels	Cluster size	Grid size				
			64	128	256	512	1024
A*	0	-	905	3575	8533	40333	104109
HPA*	1	16	492	1684	4678	16610	75085
	2	16	463	1046	3819	11842	38822
	3	16	425	1014	2513	7016	24245
	1	32	466	1343	4514	12863	63523
	2	32	454	1334	3551	10629	41491
	3	32	432	1062	2910	8319	24620
	1	64	349	1009	4371	15791	59221
	2	64	307	840	3709	10356	35844
	3	64	303	683	3473	9463	22007

When using larger cluster sizes, HPA* traverses fewer nodes than using smaller clusters. It can be explained by a larger number of inter-cluster entrance creation operations, when using smaller cluster sizes.

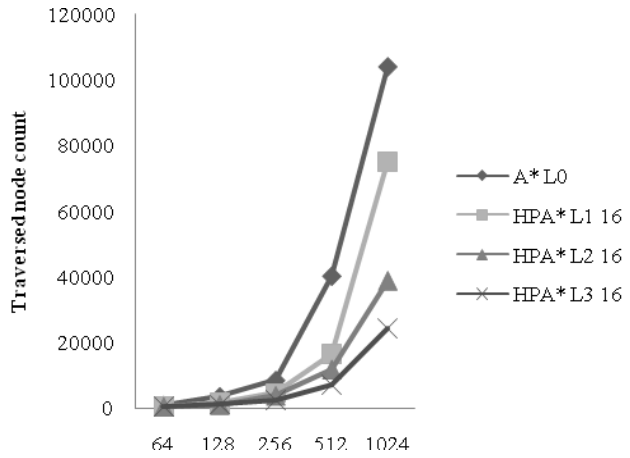

Fig. 8. The number of traversed nodes while using cluster size 16x16

Figure 8 graphically shows the number of traversed nodes for two-dimensional grids of various sizes using cluster size 16x16.

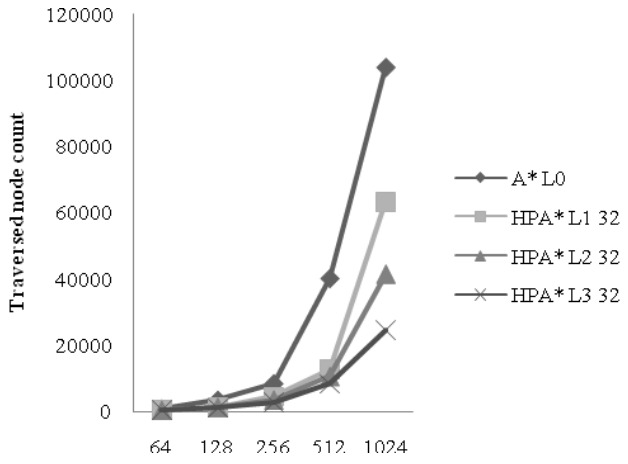

Fig. 9. The number of traversed nodes while using cluster size 32x32

Figure 9 graphically shows the number of traversed nodes for two-dimensional grids of various sizes using cluster size 32x32.

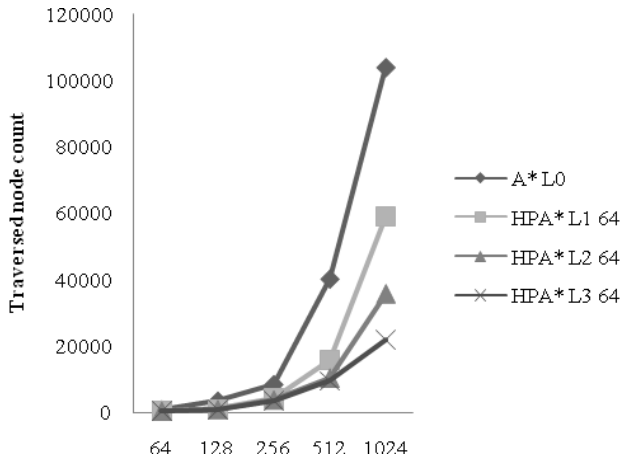

Fig. 10. The number of traversed nodes while using cluster size 64x64

Figure 10 graphically shows the number of traversed nodes for two-dimensional grids of various sizes using cluster size 64x64.

VIII. Conclusions

In this paper, A* and HPA* efficiency dependence on grid size has been analyzed: algorithm execution times and the number of traversed nodes have been measured.

HPA* performance has been the same only in a 64x64 node grid using cluster size 16; in all other cases HPA* execution has been faster than A*. Several abstraction levels with HPA* have produced faster execution times. This is especially noticeable with grids of larger sizes.

Increasing cluster size from 16 to 64 yields fewer traversed nodes during the pathfinding process. As a result, memory usage is reduced.

HPA* advantage over A* would increase even more in case the grid was static, and repeated pathfinding was provided. It would be enough to carry out pre-processing phase only once, and then use these results for every online search, while A* would have to re-search the grid from the beginning every time.

References

[1] Millington and J. Funge, *Artificial Intelligence for Games, Second Edition*. San Francisco: Morgan Kaufmann Publishers Inc., 2009.

[2] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots.* London: The MIT Press, 2004.

[3] R. C. Holte, M. B. Perez, R. M. Zimmer, A. J. MacDonald, *Hierarchical A*: Searching Abstraction Hierarchies Efficiently*: *AAAI 96 Proceedings of the thirteenth national conference on Artificial intelligence, vol. 1, pp. 530-535*, 1996.

[4] N. Sturtevant and M. Buro, *Partial Pathfinding Using Map Abstraction and Refinement: AAAI 05 Proceedings of the 20th national conference on Artificial intelligence, vol. 3, pp. 1392-1397*, 2005.

[5] M. R. Jansen, M. Buro. *HPA* Enhancements: Proceedings of the Third Artificial Intelligence and Interactive Digital Entertainment Conference*, June 6-8, 2007, Stanford, California, USA.

[6] Y. Bjornsson and K. Halldorsson. Improved Heuristics for Optimal Path-finding on Game Maps: Proceedings of the Second Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE'06), Marina Del Ray, California, USA, June 20-23, pp. 9–14, 2006.

[7] Botea, M. Müller, J. Schaeffer, "Near Optimal Hierarchical Path-Finding," in *Journal of Game Development*, vol. 1, pp. 7-28, 2004.

[8] Cui and H. Shi, "A*-based Pathfinding in Modern Computer Games," in *International Journal of Computer Science and Network Security*, vol. 11, pp. 125-130, 2011.

[9] E. W. Dijkstra., "A note on two problems in connexion with graphs," in *Numerische Mathematik*, vol. 1, pp. 269-271, 1959.

[10] P. E. Hart, N. J. Nilsson, B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," in *IEEE Transactions of Systems Science and Cybernetics*, vol. 4, pp. 100-107, 1968.

[11] R. C. Holte, T. Mkadmi, R. M. Zimmer, A. J. MacDonald, "Speeding up problem solving by abstraction: a graph oriented approach," in *Artificial Intelligence*, vol. 85, pp. 321–361, 1996.

Imant Zarembo graduated from Rezekne Higher Education Institution (Latvia), specialty "Programming Engineer" (bachelor's degree) in 2011. Currently he is a student at Rezekne Higher Education Institution, specialty "Computer Systems" (master study program). His main research interests include artificial intelligence and pathfinding algorithms.
E-mail: imants.zarembo@gmail.com

Oleg Uzhga-Rebrov is a Professor at the Faculty of Economics of Rezekne Higher Education Institution (Latvia). He received his Doctoral Degree in Information Systems from Riga Technical University in 1994. His research interests include different approaches to processing incomplete, uncertain and fuzzy information, in particular, fuzzy set theory, rough set theory as well as fuzzy classification and fuzzy clustering techniques and their applications in bioinformatics.
E-mail: rebrovs@tvnet.lv

Imants Zarembo, Oļegs Užga-Rebrovs. Īsākā ceļa meklēšanas algoritmu A* un HPA* efektivitātes atkarības no režģa izmēra analīze
Daudzas reālās pasaules ceļa meklēšanas problēmas prasa īsākā ceļa atrašanu starp diviem punktiem divdimensiju režģī reālajā laikā. Kamēr daži īsākā ceļa meklēšanas algoritmi var darboties pieņemami vienos apstākļos, citos tie var būt nepieņemami. A* ir universāls algoritms, ko izmanto gan ceļa meklēšanā, gan grafu apiešanā. HPA* ir relatīvi jauns algoritms, kas ir ātrāks un atmiņu efektīvāk izmantojošs nekā A* relatīvi lielos divdimensiju režģos, bet šī priekšrocība var

neattiekties uz ļoti maziem vai ļoti lieliem statiskiem divdimensiju režģiem. Rakstā tiek skaitļota A* un HPA* efektivitāte dažādu izmēru divdimensiju režģiem, ņemot vērā tādus parametrus kā dažādu hierarhijas līmeņu skaitu un dažādus klasteru izmērus. Algoritmi tiek realizēti un izpildīti eksperimentāli. Lai demonstrētu algoritmu efektivitāti dažādos apstākļos, tika realizēta eksperimentu sērija, kas arī parādīja HPA* priekšrocības salīdzinājumā ar A*.Izvēlēto izmēru divdimensiju režģos, kas ir aizpildīti ar nejaušā kartībā izvietotiem šķēršļiem, HPA* ātrdarbība vienāda ar A* ātrdarbību tikai 64x64 mezglu divdimensiju režģī, izmantojot klastera izmēru 16.Visos citos gadījumos HPA* izpildes laiks bija īsāks nekā A*. Vairāku hierarhijas līmeņu izmantošana samazināja HPA* ceļa meklēšanas ilgumu, kas īpaši labi bija redzams lielāko izmēru režģos. Pieaugot izmantojamo hierarhijas līmeņu skaitam, HPA* reālā laika ceļa meklēšanas fāzes ilgums samazinājās, kamēr priekšapstrādes laiks pieauga, rezultātā sniedzot īsākus summāros algoritma izpildes laikus. Klasteru izmēra palielināšana samazina ceļa meklēšanas laikā apieto mezglu skaitu visiem apskatītajiem divdimensiju režģu izmēriem, tādā veidā samazinot algoritma izpildei nepieciešamās atmiņas apjomu. HPA* algoritma priekšrocība pār A* būtu vēl lielāka situācijās, kad statiskā divdimensiju režģī ir nepieciešama atkārtota ceļa meklēšana. HPA* priekšapstrādes fāzi būtu pietiekami veikt tikai vienu reizi un visas vēlākās reālā laika meklēšanas balstīt uz priekšapstrādes rezultātiem. Kamēr A* katru reizi nāktos sākt meklēšanas procesu no jauna.

Имант Зарембо, Олег Ужга-Ребров. Анализ зависимости эффективности алгоритмов поиска кратчайшего пути A* и HPA* от размера двумерной решетки

В реальном мире многие задачи поиска кратчайшего пути требуют нахождения пути между двумя точками в реальном времени. В то время как некоторые алгоритмы поиска кратчайшего пути в одних обстоятельствах работают приемлемо, в других они неприемлемы. A* - универсальный алгоритм, который применяется как для нахождения кратчайшего пути, так и для обхождения графов. HPA* - относительно новый алгоритм, который быстрее и употребляет меньше памяти, чем A* в сравнительно больших статичных решетках, но это преимущество может не применяться к очень маленьким или очень большим решеткам. В статье определяется эффективность алгоритмов A* и HPA*, примененных к двумерным решеткам разных размеров, учитывая такие параметры, как количество уровней иерархии и размеры кластеров. Алгоритмы были реализованы и над ними проведены эксперименты. Чтобы продемонстрировать эффективность алгоритмов в разных обстоятельствах, была проведена серия экспериментов, которая показала преимущество алгоритма HPA* над A*. В использованных двумерных решетках, заполненных препятствиями случайным образом, производительность HPA* была равна A* только в решетке с размером 64x64 узлов при размере кластера 16. Во всех других случаях скорость выполнения HPA* была выше A*. Использование нескольких уровней иерархии в алгоритме HPA* снижало потребляемое время для поиска кратчайшего пути, что особенно хорошо было заметно в решетках большого размера. Увеличение размера кластеров снижает количество пройденных узлов, таким образом, снижая количество используемой памяти. С увеличением количества используемых уровней иерархии HPA* снизилось время выполнения фазы поиска пути в реальном времени и возросло время выполнения предварительной обработки, снижая в результате суммарное время выполнения алгоритма. Преимущество HPA* по сравнению с A* было бы еще больше в случае, когда в статичной решетке нужен многократный поиск пути. Достаточно было бы провести фазу предварительной обработки только один раз, и эти результаты в дальнейшем использовать для всех поисков пути в реальном времени. В то время как для A* понадобилось бы начинать поиски заново каждый раз.