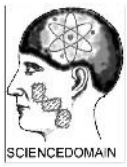
**British Journal of Mathematics & Computer Science**
**16(1): 1-18, 2016, Article no.BJMCS.24467**
***ISSN: 2231-0851***

**SCIENCEDOMAIN** *international*
www.sciencedomain.org




# A Comparative Study of Three Heuristic Functions Used to Solve the 8-Puzzle

**Anca-Elena Iordan**[1*]

[1]*Department of Electrical Engineering and Industrial Informatics, Engineering Faculty of Hunedoara, Polytechnic University Timisoara, Romania.*

***Author's contribution***

*The sole author designed, analyzed and interpreted and prepared the manuscript.*

***Article Information***

DOI: 10.9734/BJMCS/2016/24467
Editor(s):
(1) Junjie Chen, Department of Electrical Engineering, University of Texas at Arlington, USA.
Reviewers:
(1) D. G. Zhang, Tianjin University of Technology, China.
(2) R. Sabitha, Info Institute of Engineering, Anna University, India.
(3) Amarjeet Singh, Indian Institute of Technology, Roorkee, India.
Complete Peer review History: http://sciencedomain.org/review-history/14194

***Original Research Article***

***Received: 22nd January 2016***
***Accepted: 8th April 2016***
***Published: 15th April 2016***

## Abstract

This paper introduces a new heuristic function with high efficiency for an optimum solving of the 8-puzzle, this one being the double of the Chebyshev distance. The comparative study is realized among this new heuristic (Chebyshev heuristic), the Hamming heuristic and the Manhattan heuristic using A* algorithm implemented in Java. The Chebyshev heuristic function is more informed than Hamming and Manhattan heuristics. This paper also presents the necessary stages in object oriented development of an interactive software dedicated to simulate the A* search algorithm for 8-puzzle. The modeling of the software is achieved through specific UML diagrams representing the phases of analysis, design and implementation, the system thus being described in a clear and practical manner. In order to confirm the obtained theoretical results which show that Chebyshev heuristic is more efficient, two performance criteria were used: space complexity and time complexity. The space complexity was measured by the number of generated nodes from the search tree, by the number of the expanded nodes and by the effective branching factor. The time complexity was measured by the running time. From the experimental results obtained by using the Chebyshev heuristic, improvements were observed regarding space and time complexity of A* algorithm versus Hamming and Manhattan heuristics.

*Keywords: Heuristic functions; A* search algorithm; performance evaluation; java; UML.*

**Corresponding author: E-mail: anca.iordan@fih.upt.ro;*

# 1 Introduction

8-Puzzle represents a model problem which is very popular for measuring the performances of the heuristic search algorithms. The effort for improve search algorithms consists in determining of strong estimation heuristic functions which have to guide the search process to the most promising side of the search tree. Commonly used heuristics for this problem include counting the number of misplaced tiles (Hamming heuristic) and finding the sum of the Manhattan distances between each block and its position in the goal configuration (Manhattan heuristic).

From its inventing by Sam Loyd [1] till now, the solving of the 8-puzzle has been represented a continuous researching subject. There can be invented many algorithms that bring a game table from any start configuration to a given goal configuration. The difficulty appears when is desired to obtain in real time an optimal solution (as number of movements to find the solution).

8-puzzle [2] consists in 8 labeled tiles that can be moved using the only free available square space. The problem requires to be found the movements in order, starting from an initial configuration and reaching to a determined configuration. The actions represent the tiles movements, but an efficient approach from the point of view of solution search is considered that the free space is moving. In this way it result a simple problem formulation in which exist four possible actions: north movement, south movement, east movement and west movement of the free space.

# 2 Heuristic Functions

The heuristic algorithms take into consideration two pre-requisites: to find some distinction criteria which permit finding the solution with less resources that impose the uninformed algorithms and, on the other side, to determine the correct choices in order to reach the final state using the optimal path. So, the heuristic algorithms used information from the domain of the problem to be solved.

These information are usually used through an evaluation function [3] which has as argument a node of the search tree and determines a number as result, indicating the measure in which the respective node is indicated for expanding. A* search algorithm [4] combines Greedy algorithm with breadth-first search algorithm and the evaluation function has the following relation:

$$f(S) = g(S) + h(S), \tag{2.1}$$

where g(S) is the path cost from the start node to current node S, and h(S) is an estimation of the path cost from the current node to the goal node.

As it is shown in the followings, for this 8-puzzle problem there was used three heuristic functions: Hamming heuristic, Manhattan heuristic and Chebyshev heuristic. Hamming heuristic [5] is a simple heuristic function which is determined by the number of misplaced tiles.

The most used admissible and informed heuristic function recorded so far and specified to this puzzle problem is Manhattan heuristic [6]. This is calculated as follows:

$$h_M(S) = \sum \text{ManhattanDistance}(k), \text{ where } k \in \{1,2,3,\ldots,N\}. \tag{2.2}$$

ManhattanDistance(k) represents the distance of k number position (on horizontal and also on vertical axis) in S state towards its position in the goal state which is calculated with the following relation [7]:

$$\text{ManhattanDistance}(k) = |x_k - x_{kg}| + |y_k - y_{kg}|, \tag{2.3}$$

where $(x_k,y_k)$ represent the coordinates of k number position in current state and $(x_{kg},y_{kg})$ represent the coordinates of k number position in goal state.

Hereinafter the results obtained using Manhattan heuristic were improved by introducing a new heuristic function that is being calculated as the double of Chebyshev distance:

$$h_C(S) = 2 \cdot \sum \text{ChebyshevDistance(k)}, \text{ where } k \in \{1,2,3,\ldots,N\}. \quad (2.4)$$

ChebyshevDistance(k) [8] is the maximum between the horizontal distance and the vertical distance of the k number position in S state versus its position in goal state. Distance is calculated as follows:

$$\text{ChebyshevDistance(k)} = \max(|x_k - x_{kg}|, |y_k - y_{kg}|), \quad (2.5)$$

where $(x_k,y_k)$ and $(x_{kg},y_{kg})$ have the same signification as in the case of Manhattan distance.

Because:

$$a + b \leq 2 \cdot \max(a,b), \ \forall\ a, b \in \mathbb{R} \quad (2.6)$$

results that:

$$|x_k - x_{kg}| + |y_k - y_{kg}| \leq 2 \cdot \max(|x_k - x_{kg}|, |y_k - y_{kg}|). \quad (2.7)$$

Based on the definitions of two heuristic functions presented in relations (2.2) and (2.4) and using relation (2.7) it result the following relation between these heuristic functions for any intermediary state S:

$$h_C(S) \geq h_M(S). \quad (2.8)$$

The last relation demonstrates that the heuristic function $h_C$ is more informed then the heuristic function $h_M$, therefore the A* algorithm corresponding to function $h_C$ is dominating the A* algorithm corresponding to function $h_M$.

If it is considered the goal state presented in Fig. 1, the Hamming heuristic associated to an intermediary state is determined as in Fig. 2, the Manhattan heuristic associated to an intermediary state is determined as in Fig. 3 and the Chebyshev heuristic associated to the same intermediary state is determined as in Fig. 4.

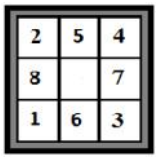



**Fig. 1. Goal state**

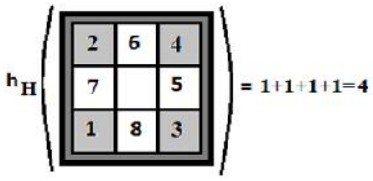



**Fig. 2. Determination of the Hamming heuristic corresponding to a current state**

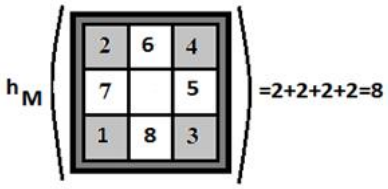



**Fig. 3. Determination of the Manhattan heuristic corresponding to a current state**

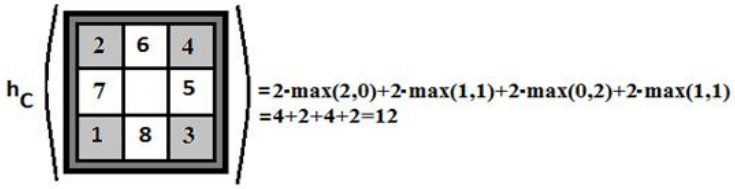



**Fig. 4. Determination of the Chebyshev heuristic corresponding to a current state**

# 3 Object Oriented Modeling of the Interactive Application

In order to observe how are updated the two lists which were used for implementing A* search algorithm (open list and closed list), but, also, to compare the obtained results by the use of the three heuristic functions that were described previously, there was implemented an interactive Java application [9], built as a game. For object oriented development of the application, Unified Modeling Language was used. The achievement of the UML diagrams was done using the ArgoUML software [10].

## 3.1 Analysis phase

By the instrumentality of UML [11], interactive application analysis consists in making use case diagram and activity diagrams. The software is exposing in a precise and concrete manner by representing the use cases [12]. Each case describes the interactions between user and interactive application. The use case represents a collection of possible scenarios that are referring to communication between software and external actors, characterized by some scopes.

Use case diagram is created in an iterative mode. First, there were identified the actors, starting from the formulated problem by identifying the roles played by different persons and external resources that are implicated in interactions. Identifying the uses cases and the relations between them was based on the analysing the responsibilities accomplished by every actor and also the global specification that are referring to the functional requisites.

The use case diagram representation is shown in Fig. 5. The diagram described defines the interactive application domain, allowing the visualization of the size and sphere of the action for the entire development process. This includes:

- One actor - the user who is external entity with which the Java application interact;
- Ten use cases that describe the functionalities of the interactive application;
- Relationships between user and use cases (association relationships) and relationships between use cases (dependency and generalization relationships).

## 3.2 Design phase

Object oriented methodologies introduce the representation of the static structure of interactive application using classes [13] and relations between them. Conceptual modeling allows identifying the most important

concepts for the interactive application. Inheritance was not used only as a generalization process, which is when derived classes are specializations of the base class. The class diagram is represented in Fig. 6 in order to be observed the connection mode between the classes and the interfaces that are used and also the composition and aggregation relationships between these class instances. The interactive application was built as flexible can be in order to permit adding of new algorithms for the resolution of 8-puzzle problem, using the possibilities offered by Java interfaces.

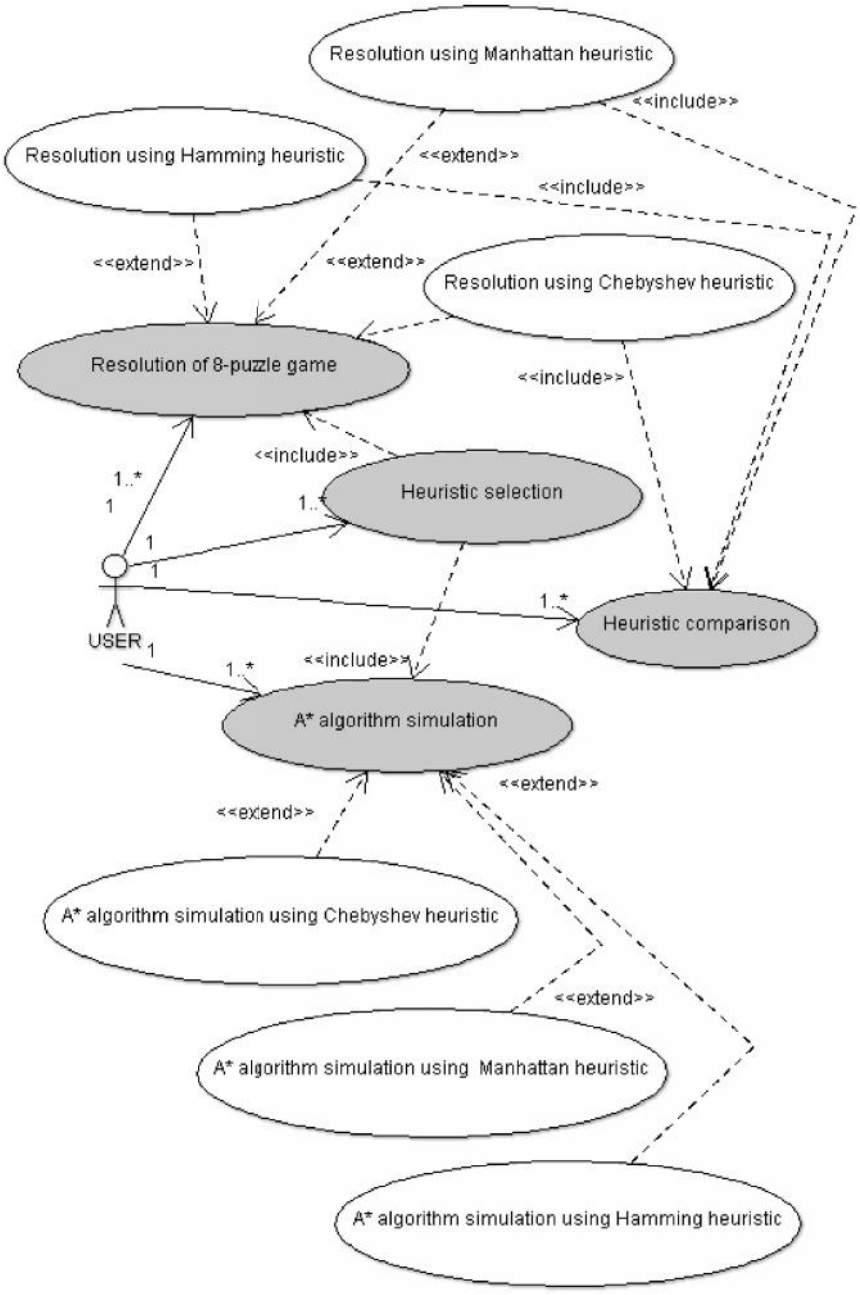



**Fig. 5. Use case diagram**

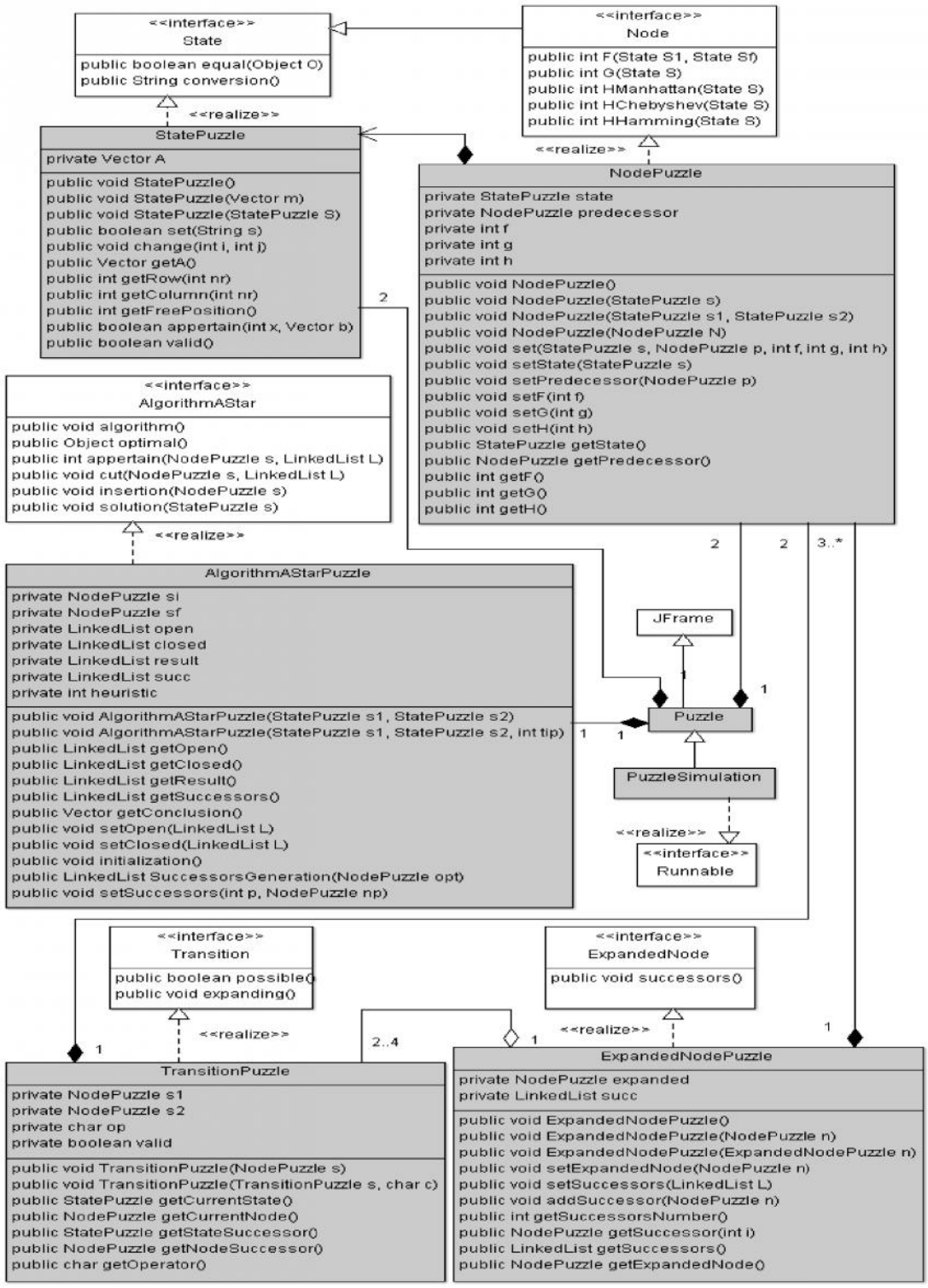



**Fig. 6. Class diagram**

In this way, for memorizing a game state configuration, there was implemented "StatePuzzle" class which implements "State" interface. For storing a node of the search tree, there was implemented "NodePuzzle" class which realizes "Node" interface and for memorizing an expanded node together with its successors, there was implemented "ExpandedNodePuzzle" class which realizes "ExpandedNode" interface.

A* search algorithm was implemented by using "AlgorithmAStarPuzzle" class which realizes "Algorithm-AStar" interface. An instance of this class is composed by two instances of "NodePuzzle" class, according to composition relationship which exists in the class diagram, but can also contain instances of "ExpandedNodePuzzle" class, as it can be observed from the aggregate relationship presented in the same diagram.

For realizing the two windows that will compose the graphical interface of the application, there were implemented the following classes: "Puzzle" for the main window and "PuzzleSimulation" for the simulation of A* algorithm that uses Hamming, Manhattan or Chebyshev heuristics. We can observe that the "PuzzleSimulation" class inherit attributes and methods of the "Puzzle" class, but implements the "Runnable" interface.

## 3.3 Implementation phase

Component diagram [14] is similar to package diagram, allowing visualization of how the interactive application is divided and the dependencies between modules. The diagram presented in Fig. 7 describes the collection of components that together provide application functionalities.

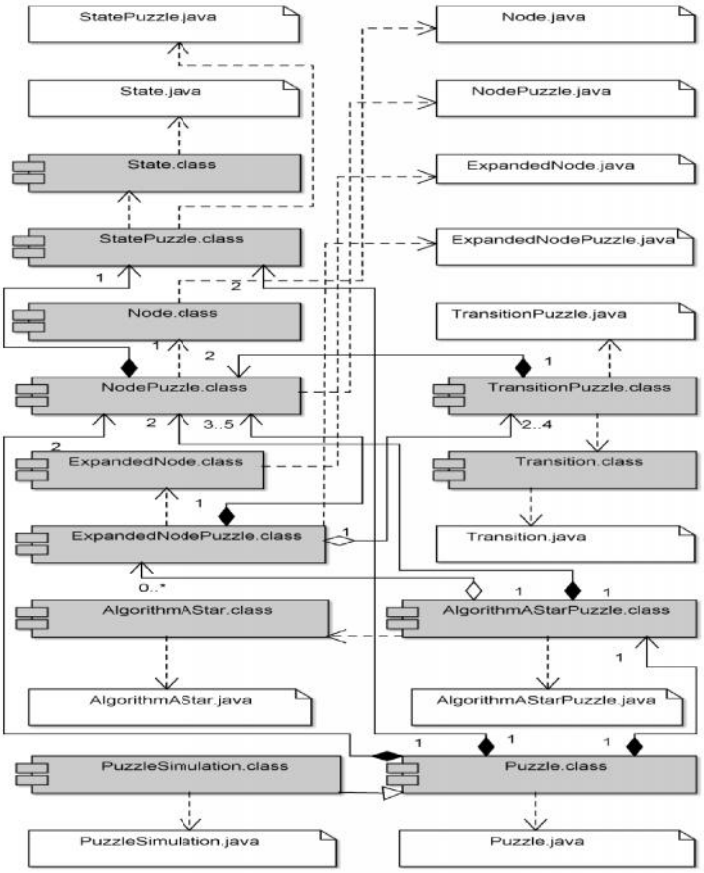



**Fig. 7. Component diagram**

The central component of the diagram is "Puzzle.class", a component obtained by transforming into executable code the "Puzzle.java" component by the Java compiler. There can be observed that this component interacts directly with the following components: "AlgorithmAStarPuzzle.class", "NodePuzzle.class", "PuzzleSimulation.class" and "StatePuzzle.class". The "AlgorithmAStarPuzzle.class" component interacts directly with "ExpandedNodePuzzle.class" component and "NodePuzzle.class" component. The "ExpandedNodePuzzle.class" component interacts directly with "TransitionPuzzle.class" component and "NodePuzzle.class" component.

# 4 Interactive Application Interface

A configuration on the game table which has to be solved can be introduced by the user or can be automatically generated, is being solved by using the implemented algorithm.

In Fig. 8 is presented the main graphical user interface of the application, a instance of "Puzzle" class which contains components that specify the start configuration, the goal configuration and an intermediary configuration of solution, lists for viewing the solutions obtained by using Hamming, Manhattan and Chebyshev heuristics and, also, a component for comparing the three variances of A* algorithm from the point of view of space and time complexity. In Fig. 8 are generated the solutions of a configuration which is introduced from the standard input by using the three heuristics. In this case, the optimal solution is being obtained in 22 movements.

The application also permits the representation of a 3x3 configuration corresponding to an intermediary state of solution, by selection of this one from the list.

# 5 Experimental Results

In order to confirm the obtained theoretical results which show that Chebyshev heuristic is more efficient, two performance criteria were used: space complexity [15] and time complexity [16]. The space complexity was measured by the generated nodes number from the search tree, by the number of the expanded nodes and by the effective branching factor [17]. The time complexity was measured by the running time.

I took, by convention, the configuration presented in Fig. 1 to be the goal state. There are 9! possible permutations on a 3 x 3 board, and every second permutation is solvable [18]. Hence, there is a total of 9!/2=181440 solvable problem instances.

For 8-puzzle game there was tested a set including 10000 random configurations. This was done by running of the interactive application presented in the previous paragraph. The average length of all optimal solution paths is f*=21,98, that means that almost 22 moves are needed to solve a given random configuration. Fig. 9 shows the distribution of the optimal solution path lengths for all tested configurations.

## 5.1 The graphical comparison of the results regarding the generated nodes number

For presenting the efficiency of the Chebyshev heuristic from the minimum generated nodes number point of view there are represented graphical the experimental values obtained for average value of solutions (move values belonging to the closed interval [8,20]), as well as for solutions of high values (move values belonging to the closed interval [21,30]).

The comparison of the average value of solutions, presented in Fig. 10, is realized for the three analyzed functions: Hamming heuristic, Manhattan heuristic and Chebyshev heuristic. The solutions of high value are compared for the two most efficient heuristics: Manhattan and Chebyshev heuristic, and the graphical representation it is shown in Fig. 11.

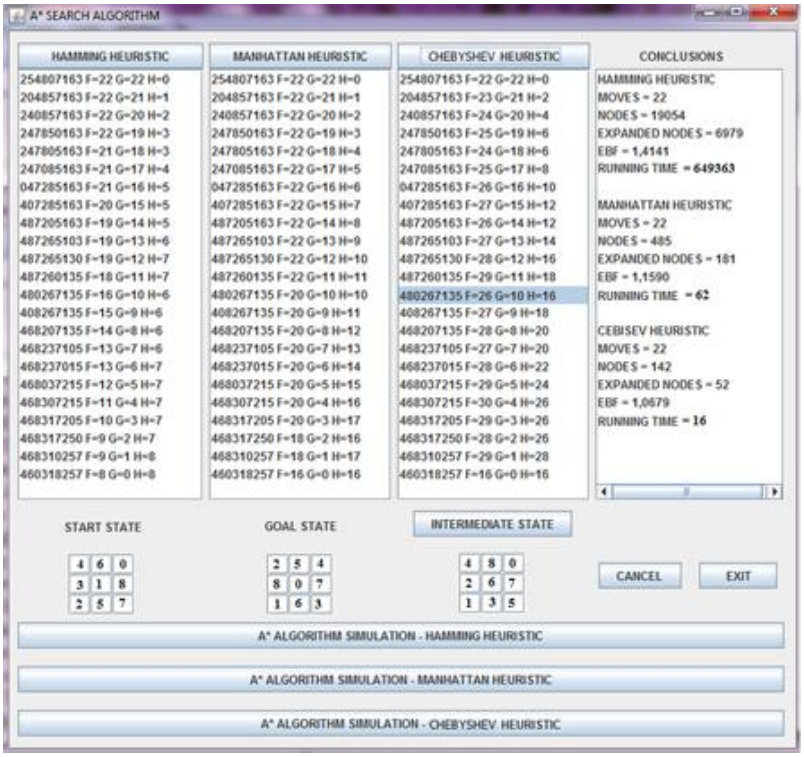



**Fig. 8. The graphical user interface of the application**

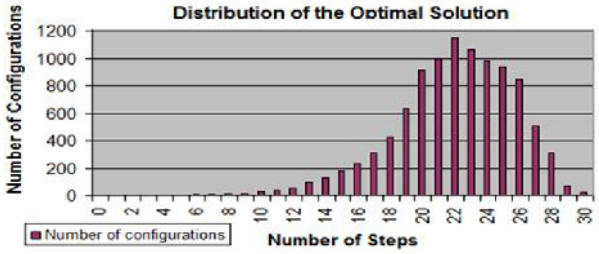



**Fig. 9. The distribution of the optimal solution**

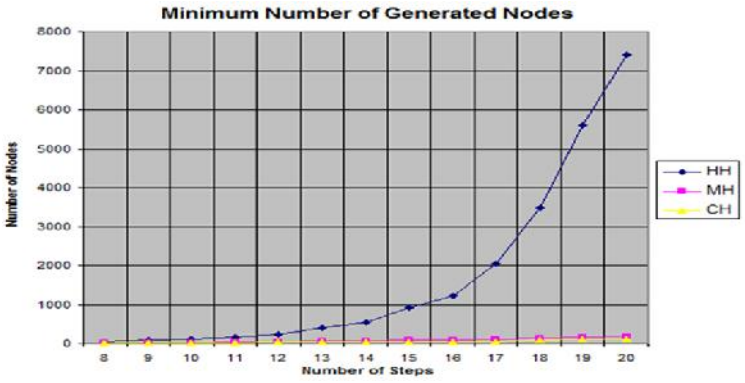



**Fig. 10. Minimum number of generated nodes for medium solutions**

For presenting the efficiency of Chebyshev heuristic from the point of view of the maximum generated nodes number the values obtained experimental of the medium dimension solutions and the grand dimension solutions are represented graphical.

The comparison of the results of medium dimension is presented in Fig. 12, the comparative study is made for the three functions: Hamming heuristic, Manhattan heuristic and Chebyshev heuristic. The grand dimension solutions are compared for the two most efficient heuristics: Manhattan and Chebyshev, and the graphic it is represented in Fig. 13.

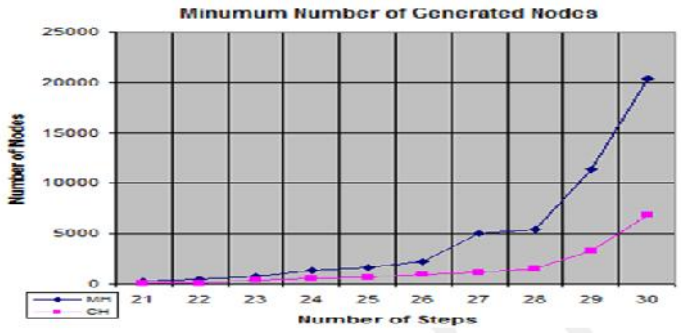



**Fig. 11. Minimum number of generated nodes for grand solutions**

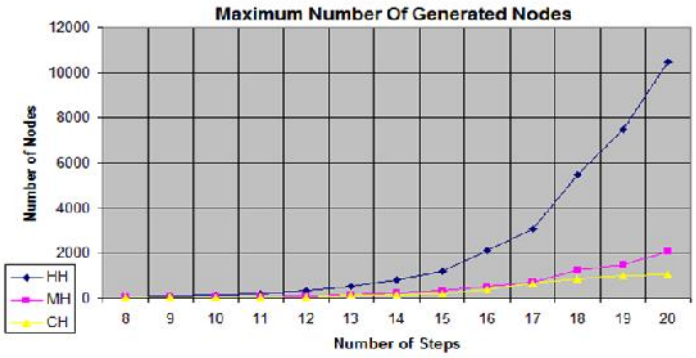



**Fig. 12. Maximum number of generated nodes for medium solutions**

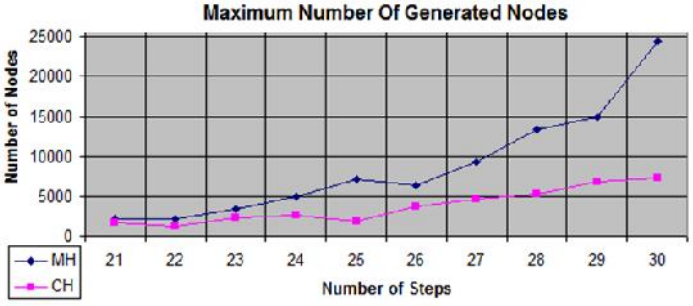



**Fig. 13. Maximum number of generated nodes for grand solutions**

For presenting the efficiency of Chebyshev heuristic from the point of view of the medium generated nodes number the values obtained experimental of the medium dimension solutions and the grand dimension solutions are represented graphical. The comparison of the results of medium dimension is presented in Fig. 14, and for the grand dimension is represented in Fig. 15.

## 5.2 The graphical comparison of the results regarding the expanded nodes number

With the purpose of showing the superiority of the Chebyshev heuristic from the minimum of the expanded nodes number point of view are presented graphical the experimental results. The examination of the medium dimension solutions, presented in Fig. 16, is referring to the three studied functions: Hamming, Manhattan and Chebyshev heuristics. The grand dimension solutions are presented for the two most efficient heuristics: Manhattan and Chebyshev, and the results are shown in Fig. 17.

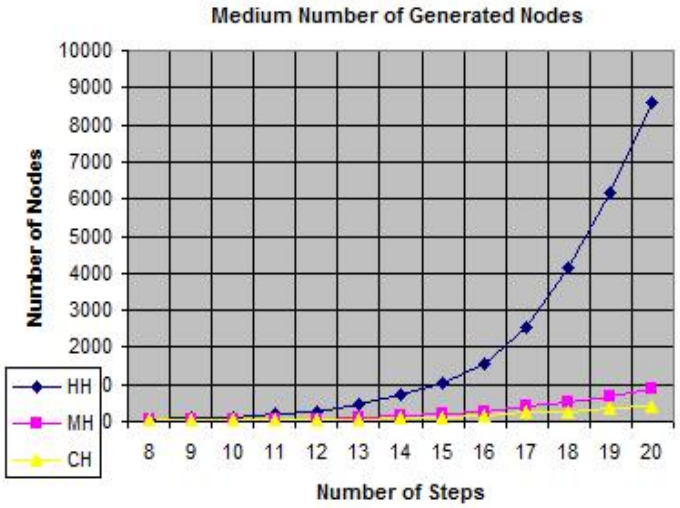



**Fig. 14. Medium number of generated nodes for medium solutions**

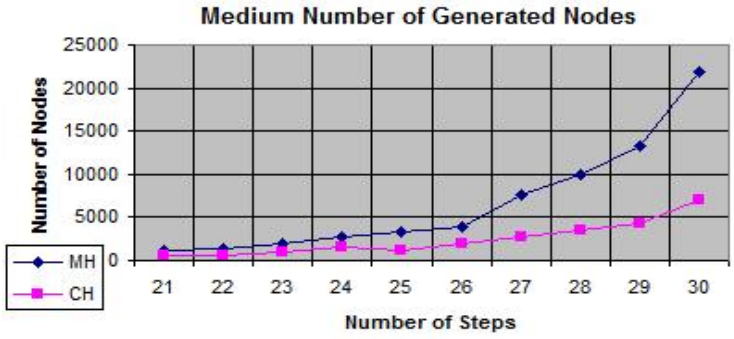



**Fig. 15. Medium number of generated nodes for grand solutions**

For illustrating the efficiently of Chebyshev heuristic from the maximum of the expanded nodes number point of view are presented graphically the experimental results. The examination of the medium dimension solutions, presented in Fig. 18, is achieved for the three studied heuristics: Hamming, Manhattan and Chebyshev. The graphic presented in Fig. 19 illustrates the grand dimension solutions for the two most efficient heuristics: Manhattan and Chebyshev.


**Fig. 16. Minimum number of expanded nodes for medium solutions**

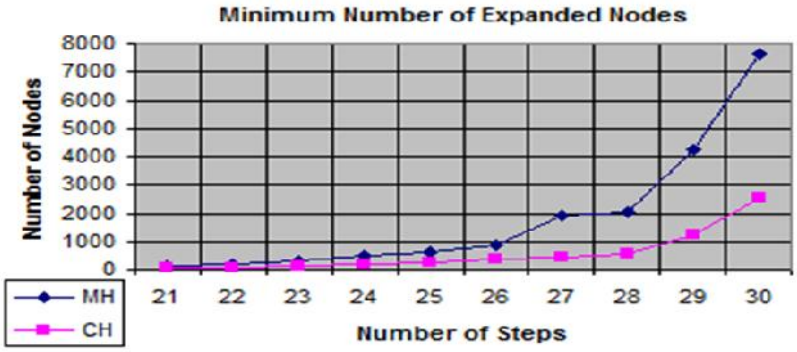



**Fig. 17. Minimum number of expanded nodes for grand solutions**

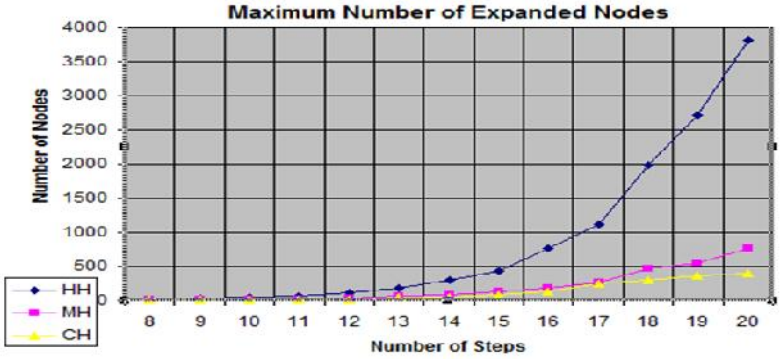



**Fig. 18. Maximum number of expanded nodes for medium solutions**

The values obtained from the realized experiments are represented graphical to illustrate the efficiency of the Chebyshev heuristic in terms of the medium expanded nodes number. In Fig. 20 are analyzed the medium dimension solutions, and the grand dimension solutions is illustrated in Fig. 21.

## 5.3 The graphical comparison of the results regarding the effective branching factor

For illustrating the efficiency of the Chebyshev heuristic from the minimum values corresponding to the effective branching factor point of view are presented graphical the experimental values. The analysis of the medium dimension results, presented in Fig. 22, is realized for the three examined functions: Manhattan, Hamming and Chebyshev heuristics. The grand dimension solutions are compared for the two most efficient heuristics: Manhattan and Chebyshev, and the graphic is illustrated in Fig. 23.

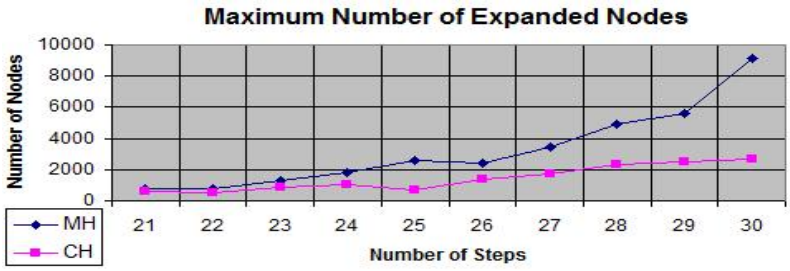



**Fig. 19. Maximum number of expanded nodes for grand solutions**

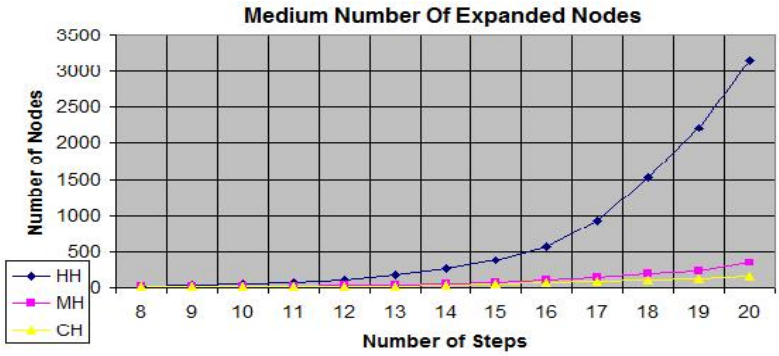



**Fig. 20. Medium number of expanded nodes for medium solutions**

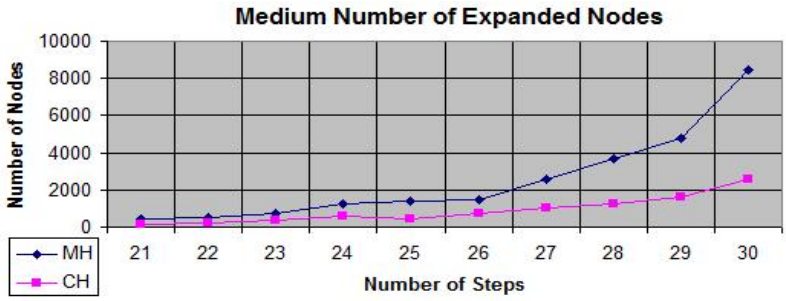



**Fig. 21. Medium number of expanded nodes for grand solutions**

For illustrating the efficiency of the Chebyshev heuristic from the maximum values corresponding to the effective branching factor point of view, the experimental values are represented graphical. Presented in Fig. 24, the analysis of the medium dimension solutions is achieved for the three examined functions: Manhattan, Hamming and Chebyshev heuristic. The grand dimension solutions are compared for the two most efficient heuristics: Manhattan and Chebyshev. The graphic is illustrated in Fig. 25.

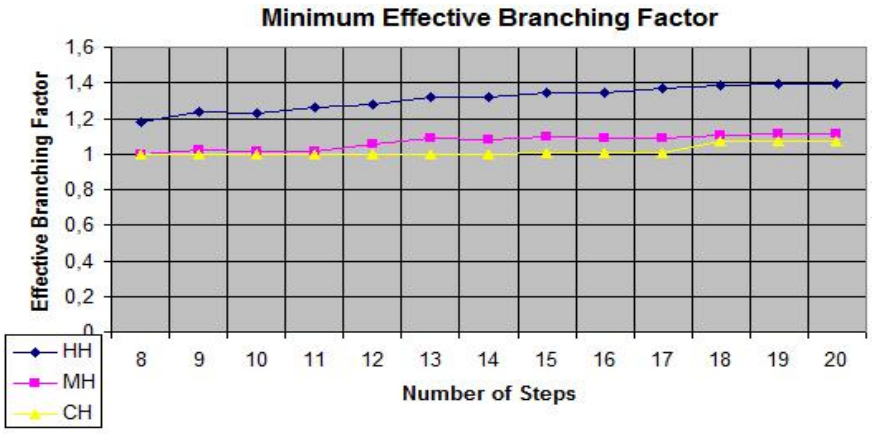



**Fig. 22. Minimum effective branching factor for medium solutions**

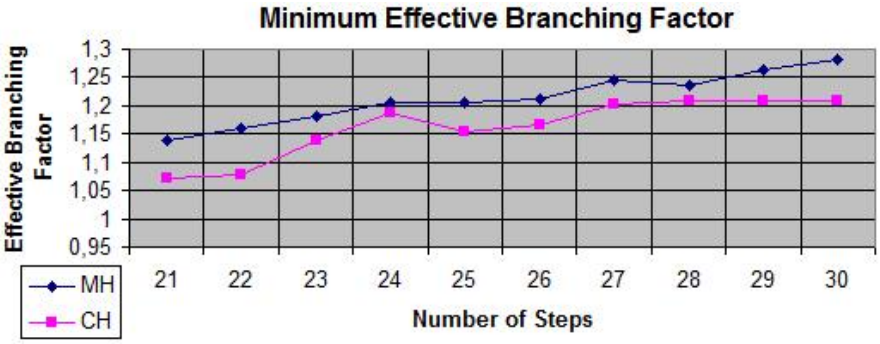



**Fig. 23. Minimum effective branching factor for grand solutions**

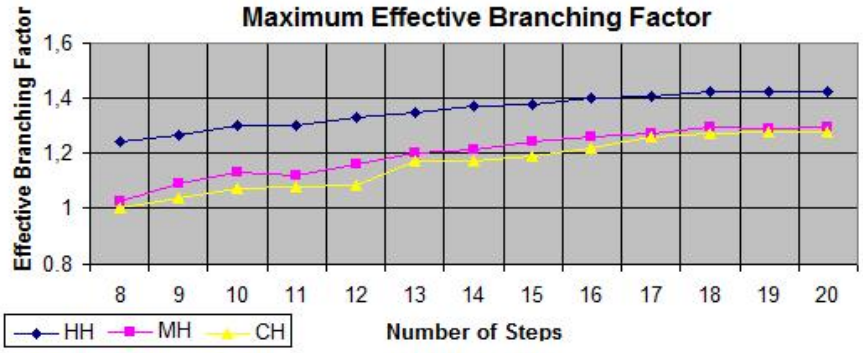



**Fig. 24. Maximum effective branching factor for medium solutions**

For illustrating the efficiency of the Chebyshev heuristic from the medium values corresponding to the effective branching factor point of view, the experimental values are presented graphical. The analysis of the medium dimension solutions, presented in Fig. 26, is realized for the three examined heuristics: Manhattan, Hamming and Chebyshev. The grand dimension solutions are compared for the two most efficient heuristics: Manhattan and Chebyshev. The graphic is illustrated in Fig. 27.

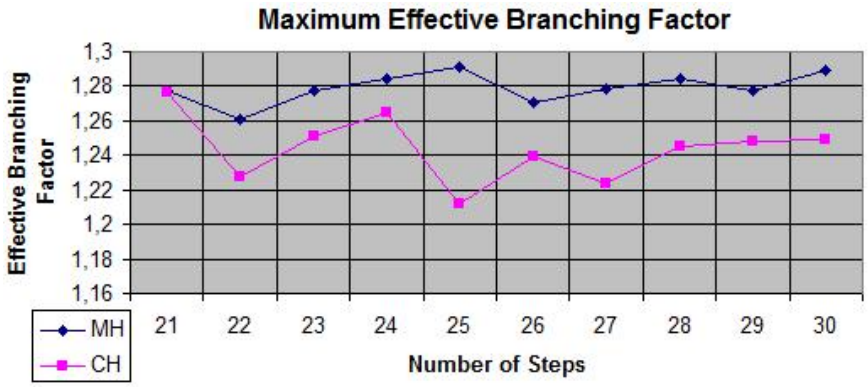



**Fig. 25. Maximum effective branching factor for grand solutions**

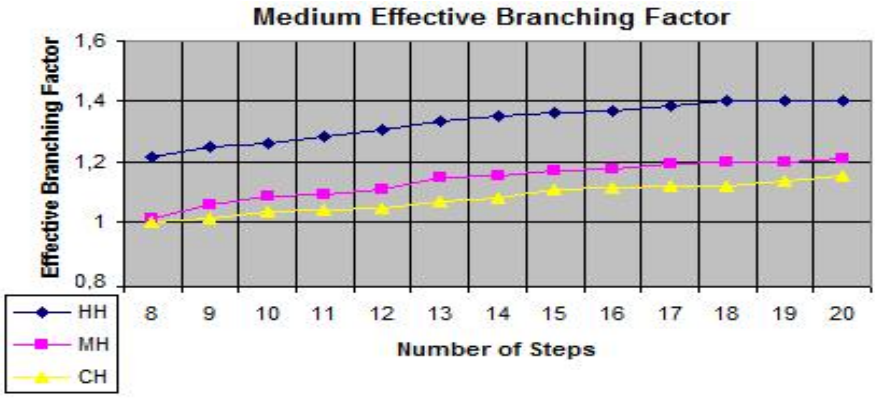



**Fig. 26. Medium effective branching factor for medium solutions**

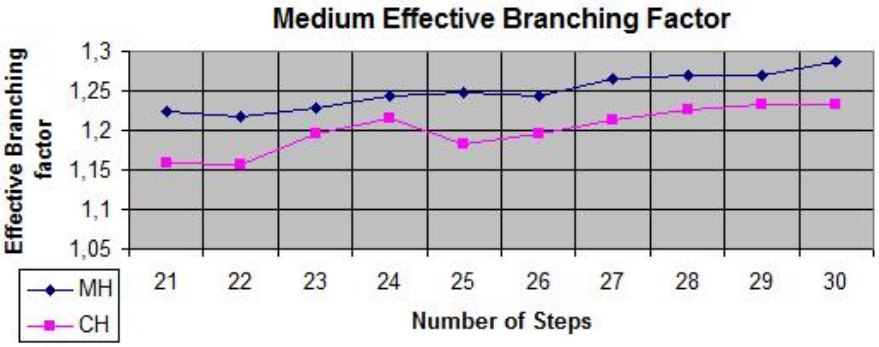



**Fig. 27. Medium effective branching factor for grand solutions**

## 5.4 Conclusions regarding the space complexity

Analyzing the results presented in the graphics represented in the Figs. 10 to 15, it can be observed that the number of steps made for obtaining the solution is the same for similar configurations, determining the optimal solutions for the three examined heuristics. But, investigating the generated nodes number in the search tree associated with the A* algorithm using the Chebyshev heuristic, it can be observed that this

number is strictly less than the number obtain by using the other two heuristics. By analyzing of the results presented in the graphics represented in the Figs. 16 to 21 referring to the expanded nodes number in the search tree, we reach to the same conclusion, that the expanded nodes number from the search tree associated to the A* algorithm using the Chebyshev heuristic is strictly less.

Another comparing criterion for the three heuristic searches is the effective branching factor [16]. Calculating approximately the effective branching factor for the three heuristics, using Newton method [8] for resolving the equation, there were obtained the values illustrated graphical in Figs. 22 to 27. The values of b* appropriate to the function $h_C$ are more appropriate to the value 1 than the values of b* appropriate to the functions $h_M$ and $h_H$, so the A* algorithm using $h_C$ heuristic drives to an optimal solution in a way that appears to be linear. According to these experimental values, results the superiority of Chebyshev heuristic from the Manhattan and Hamming heuristics. In this case we can say that $h_C$ heuristic dominates $h_M$ and $h_H$ heuristics, from the space complexity point of view.

## 5.5 The statistical investigation of the time complexity

In terms of time complexity it was compared the medium time of execution. The order of time complexity depends on the effective branching factor [4], its complexity is O((b*)m), where m represents the number of steps made for determining the optimal solutions. For illustrating the efficiency of Chebyshev heuristic from the time complexity point of view are presented graphical the experimental values in Figs. 28 and 29.

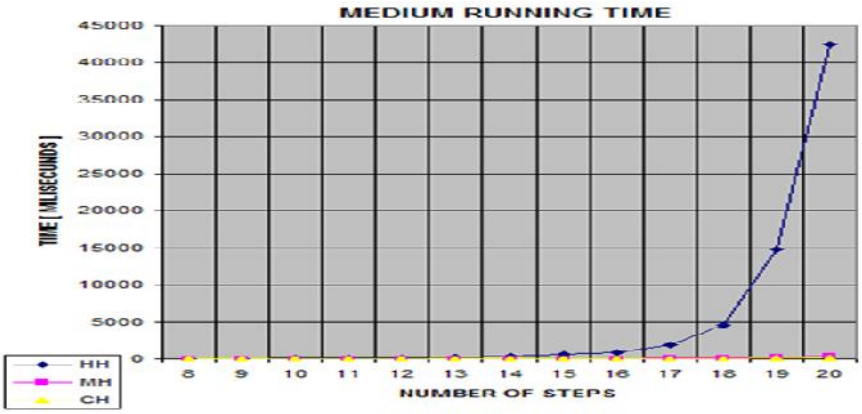



**Fig. 28. Medium running time for medium solutions**

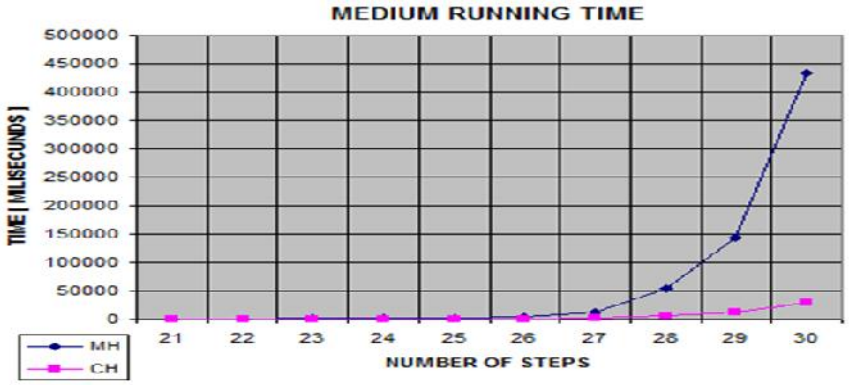



**Fig. 29. Medium running time for grand solutions**

## 6 Conclusions

In this paper was introduced the Chebyshev heuristic function as the double of the Chebyshev distance. Chebyshev heuristics is more informed than Manhattan and Hamming heuristics. From experimental results by using the Chebyshev heuristic, an improvement in space and time complexity of A* algorithm versus Hamming and Manhattan heuristics was observed. In three cases the generated solutions are obtained using an optimal number of movements.

## Competing Interests

Author has declared that no competing interests exist.

## References

[1] Pickard S. The puzzle king: Sam Loyd's chess problems and selected mathematical puzzles. Pickard & Son Pub; 1996.

[2] Kunkle D. Solving the 8 puzzle in a minimum number of move: An application of the A* algorithm. Introduction to Artificial Intelligence; 2001.

[3] Bonet B, Geffner H. Planning as heuristic search. Artificial Intelligence. Elsevier. 2001;5-33.

[4] Russel S, Norvig P., Artificial intelligence – A modern approach. Pearson Education. 2013.

[5] Rothlauf F. Design of modern heuristics – Principles and application. Springer-Verlag. Berlin; 2011.

[6] Edelkamp S, Schroedl S. Heuristic search: Theory and applications. Morgan Kaufmann Publishers; 2011.

[7] Bulitko V, Bjornsson Y, Sturtevant N, Lawrence R. Real-time heuristic search for Pathfinding in video games. Artificial Intelligence for Computer Games. Springer. 2011;1-30.

[8] Manetti M. Topology. Springer; 2015.

[9] Deitel P, Deitel H. Java SE 8 for programmers. Prentice Hall; 2014.

[10] Complak W, Wojciechowski A, Mishra A, Mishra D. Use cases and object modeling using ArgoUML. Lecture Notes in Computer Science. 2011;246-255.

[11] Dennis A, Wixom B, Roth R. Systems analysis and design. John Wiley & Sons Ltd; 2014.

[12] Craig L. Applying UML and patterns. Pearson Education; 2004.

[13] White F. Object-oriented software engineering: Practical software development using UML and java. McGraw-Hill Education; 2015

[14] Seidl M, Scholz M, Huemer C, Kappel G. UML@Classroom – An introduction to object oriented modeling. Springer; 2015.

[15] Hunt J. Guide to the unified process featuring UML, java and design patterns. Springer; 2014.

[16] Kratochv J. Combinatorial algorithms. Springer; 2015.

[17] Muscalagiu I, Vidal J, Cretu V, Popa H, Panoiu M. Experimental analysis of the effects of agent synchronization in asynchronous search algorithms. International Journal of Software Engineering and Knowledge Engineering. 2008;619-636.

[18] Korf R. Dept-first iterative-deepening: An optimal admissible tree search. Artificial Intelligence. 1985;97-109.



***Peer-review history:***
*The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)*
*http://sciencedomain.org/review-history/14194*