DE GRUYTER OPEN

BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **15**, No 2

Sofia • 2015

Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.1515/cait-2015-0036

Robot Path Planning Based on Improved A* Algorithm

Jiansheng Peng, Yiyong Huang, Guan Luo

National Key Laboratory of Communication, UEST of China Chengdu 610054, China
Department of Physics and Mechanical and Electronic Engineering, Hechi University, Yizhou 546300, China
Email: sheng120410@163.com

Abstract: *Due to the characteristic that A* algorithm takes a long time when traversing an OPEN table and a CLOSED table, an improved method is proposed that is a new way of array storing in an OPEN table and a CLOSED table. Compared to the original A* algorithm, the way of array storing accesses the array elements by locating the number ranks each time you visit a specified element, which can be done by only one operation. The original A* algorithm requires the traverse of multiple nodes in order to find a specified element. The experimental results show that the comparison of the improved A* algorithm with the original A* algorithm shows that the operating efficiency is improved by more than 40%. Based on the improved A* algorithm the method preserves the advantages of the original A* algorithm, improving the operating efficiency of A* algorithm.*

Keywords: *A* algorithm, path planning, grid, robot.*

1. Introduction

A* algorithm is jointly proposed by P. E. H a r t, N. J. N i l s s o n and B. R a p h a e l 1968 [1]. A* algorithm is a compact and efficient algorithm. A* algorithm is a typical artificial intelligence algorithm of heuristic search. Compared to other artificial intelligence algorithms [2, 3] it has many advantages, such as shorter running time, high efficiency, easy implementation. Therefore, A* algorithm has been widely used in various fields [4]. The path planning aspects of A* algorithm application [5] have been connected with a lot of research achievements.

A* algorithm has become mature after decades of development. Currently, the way of improving A* algorithm uses two methods – lowering the algorithm running time and reducing the storage space. Usually by improving the traversal way, the algorithm running time is reduced, by changing the way the data is stored, the storage space is reduced. A* algorithm is a progressive global search algorithm, an algorithm from local start searching, through local speculation global search. Although A* algorithm is a global search algorithm, it does not traverse the full global one.

2. A* algorithm related introduction

2.1. Definition of a child node and a parent node

Definition 1. The child node is an extension of the parent node, the child node always points to a parent node, the child node has one and only one parent.

Definition 2. The parent node is a node that can extend, the parent node can extend the child nodes up to 8.

As shown in Fig. 1, the red grid is a parent node, the eight green grids are child nodes.

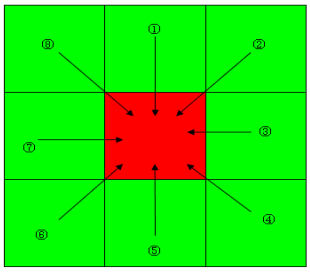

Fig. 1. A child node and a parent node definition diagram

2.2. Evaluation function

The core of A* algorithm is the evaluation function. A* algorithm selects the next expanded child node through an evaluation function. The evaluation function is used to get an efficient and lightweight A* algorithm. The expression of the evaluation function is

$$F(n) = G(n) + H(n), \tag{1}$$

where: $F(n)$ is the evaluation function of A* algorithm; $G(n)$ is the actual consideration from the start node to the current node n (that is the total distance of the optimal path from the starting point to node n); $H(n)$ is the estimated consideration from the current node n to the destination node (that is the estimated distance from node n to the target node).

The function $G(n)$ of $F(n)$ can actually be calculated, but $H(n)$ (the evaluation function) cannot calculate the actual values, only estimated values of the approximate estimation. Selecting different evaluation functions $H(n)$, obtained from different results of $F(n)$ values, the shortest path search and the time required to run the program also vary.

2.3. Using Manhattan distance as an evaluation function

Manhattan distance is proposed by Hermann Minkowski in the nineteenth century, its representation in the geometry of space is: the distance between two points in x direction, plus y direction.

Assuming that point P1 is the node n, point P2 is the end node m, the coordinates of point P1 are (x_1, y_1) and the coordinates of point P2 are (x_2, y_2). Manhattan distance from node n to node m is

(2) $$H_m(n) = | x_1 - x_2 | + | y_1 - y_2 | .$$

2.4. A* algorithmic process

A* algorithm needs to set up two tables: an OPEN table and a CLOSED table. The OPEN table saves all the nodes been generated, but not yet examined. The CLOSED table records the nodes that have been visited. The A* algorithm flow chart is shown in Fig. 2.

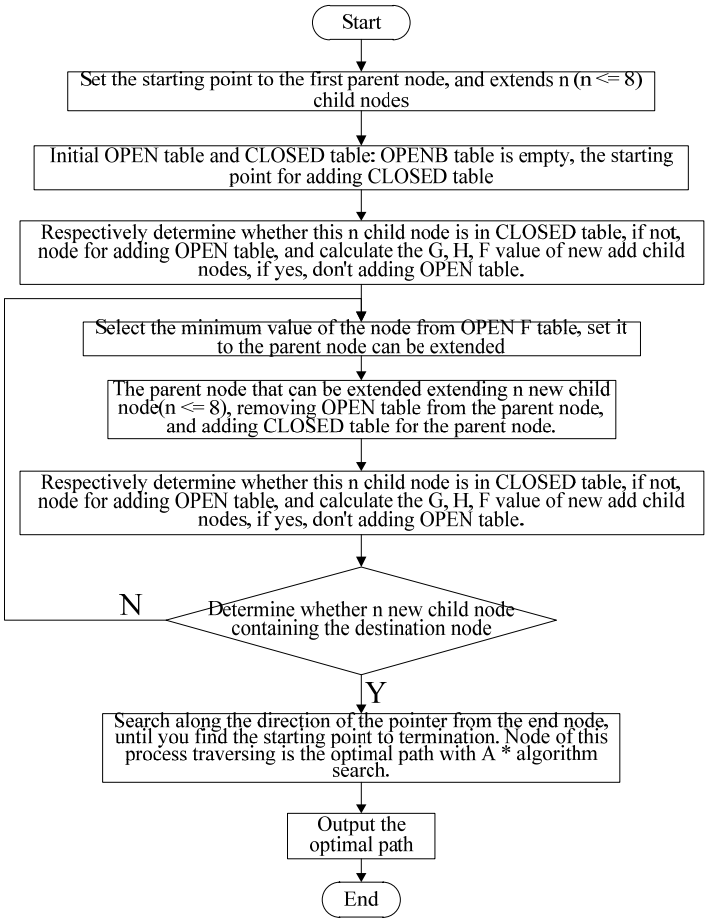

Fig. 2. A* algorithm flow chart

3. Improved A* algorithm

The OPEN table and the CLOSED table of A* algorithm store data in the form of a binary trees or list tables. Although the binary trees and the list tables have the advantages of easy inserting and deleting of operations, in order to find a node requires to traverse several times to determine whether the data is in a list table or a binary tree location. Every time of visiting an OPEN table and a CLOSED table needs to traverse multiple nodes in order to find the specified node. The array is able to achieve one operation for positioning the node. According to the advantage of the array, a data structure query_table(i, j) is proposed, an alternative of the improved method in an OPEN table and a CLOSED table lookup function. By accessing the structured data, the nodes can be found, and determine the status of the node. The states of the node are: a free state, an OPEN table state and a CLOSED table state. The improved A* algorithm still retains an OPEN table, the CLOSED table does not exist, and the structural data query_table(i, j) is used instead of the query functions of the OPEN table and CLOSED table.

3.1. Structured data query_table(i, j)

The structure of the data is in MATLAB language structural body data types. The structure of the data query_table(i, j) members is shown in Table 1.

Table 1. Description structure of the data group members

Structured data members	Description of the structure data member
query_table(i, j).F	recorded F value of (i, j) node (evaluation value)
query_table(i, j).G	record G value of (i, j) node (actual substituting value)
query_table(i, j).H	recorded H value of (i, j) node (estimation value)
query_table(i, j). Pointer	record coordinates of the parent node of the (i, j) node (record variables pointing to a parent node)
query_table(i, j). State	record the status of (i, j) node; state of query_table(i, j) has three values: 0 indicates a free state, 1 indicates the OPEN table state, 2 indicates the CLOSED table state

The variable of query_table(i, j).Pointer is used for recording the coordinates of the parent node of (i, j) node, e.g., query_table(2, 3).Pointer = (2, 2) shows a parent node with 2 rows and 3 columns nodes, equal to 2 rows and 2 columns nodes.

Initialization of the structured data members query_table(i, j).State and description of the functions:

1. Initialize the structure data member of query_table(i, j).State

$$\begin{cases} \text{query\_table}(i, j).\text{State} = 0, \\ 0 < i \le \text{line}, \\ 0 < j \le \text{row}. \end{cases} \tag{3}$$

2. Update the instructions of the structure data members query_table(i, j).State: suppose (3, 4) node is selected as an extending parent node, (4, 5) node is the optimal extension child node of the extension node, and the query_table(4, 5).State

is equal to 0, then the extension child node joins the OPEN table, and set query_table(4, 5).State is 1. Removing (3, 4) node from the OPEN table, while the re-set query_table(3, 4).State is 2.

3.2. Steps of the process of improving A* algorithm

Step 1. Initializing each member of the structure data query_table(i, j).State is

$$\begin{cases} \text{query\_table}(i, j).F = +\infty, \\ \text{query\_table}(i, j).G = 0, \\ \text{query\_table}(i, j).\text{Pointer} = (0, 0), \\ 0 < i \le \text{line}, \quad 0 < j \le \text{row}, \end{cases} \tag{4}$$

$$\text{query\_table}(i, j).\text{State} = \begin{cases} 0 & \text{freely accessible nodes}, \\ 2 & \text{barriers nodes}. \end{cases} \tag{5}$$

Step 2. Select the starting point (i_0, j_0) as the child node that the parent node is extending, and determine whether the new child node (i, j) of the query_table(i, j).State value is equal to 0 or not. If its value is equal to 0, then the new child node (i, j) of the query_table(i, j).State value is set to 1 again, and the node (i, j) adding an OPEN table, and calculate the new child node of the query_table (i, j).G, query_table(i, j).H, query_table(i, j).F value, set the new child node (i, j) of the query_table(i, j).Pointer value presents the position coordinates of the parent node, namely, query_table(i, j).Pointer = (i_0, j_0). The parent node (i_0, j_0) of query_table(i_0, j_0).State value is reset to 2, namely, query_table(i_0, j_0).State = 2.

Step 3. Selecting the minimum node (i_m, j_m) of query_table(i, j).F value from the OPEN table (where in), and set the node (i_m, j_m) as the extending parent node. Based on node (i_m, j_m) extending a new child node, according to these new child nodes (i_n, j_n) of the query_table(i_n, j_n).State the value to determine whether it can be added to an OPEN table or not. If query_table(i_n, j_n).State is equal to 0, the new child node can be added to an OPEN table, and set query_table(i_n, j_n).State = 1, update the new child node of query_table(i, j).F and query_table(i_n, j_n).Pointer. If query_table(i_n, j_n).State is equal to 1, calculate the new child node of query_table(i, j).F value, if the new calculated query_table(i, j).F value is smaller than the actual value, then update the new child node of query_table(i_n, j_n).F and query_table(i, j).Pointer. Query_table(i_n, j_n).State is equal to 2, then it cannot join in OPEN table.

Step 4. Remove the OPEN table from the extension parent node (i_m, j_m) and set query_table(i_m, j_m).State = 2.

Step 5. Determine whether the new child node has a target node (endpoint) or not, otherwise go back to Step 3 to continue the cycle, if yes, exit the loop.

Since the child node only points to a parent node, the parent node along the target node refers to finding the path of the end point to the starting point. Finding the path is the optimal path which the improved A* algorithm has searched for. A specific operating procedures flow chart of the improved A* algorithm is shown in Fig. 3.

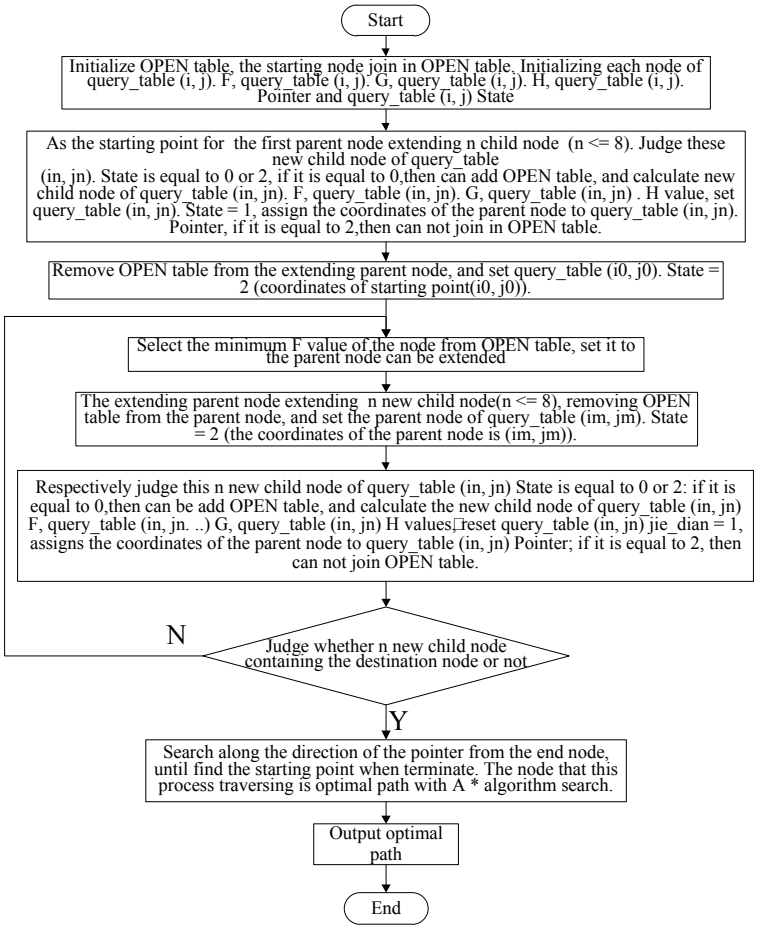

Fig. 3. Improved A * algorithm flow chart

4. Experimental simulation and performance analysis

Currently, the research advice of robot path planning is very mature. In [6] single robot path planning based on genetic algorithm is given, in [7] mobile robot path planning, based on particle swarm algorithm, in [8] – based on AFSA robot path planning, etc. Path planning is divided into global path planning and local path planning. The global path planning that is path planning under circumstances based on obstacles environment is completely known, while local path planning is the local path planning based on the local perceived obstacle environment of sensors. Single robot path planning is the global path planning based on obstacles environment. In the global path planning obstacles the environmental modelling method can be divided into: a grid method, can view method, topological hair method, free space method, neural network method, etc., [9, 10]. The grid method among all has more comprehensive reflect obstacles distribution and can only access the spatial distribution, the convenient precise movement of the robot. This paper uses the grid method for obstacle environmental modelling.

4.1. Path planning experiments

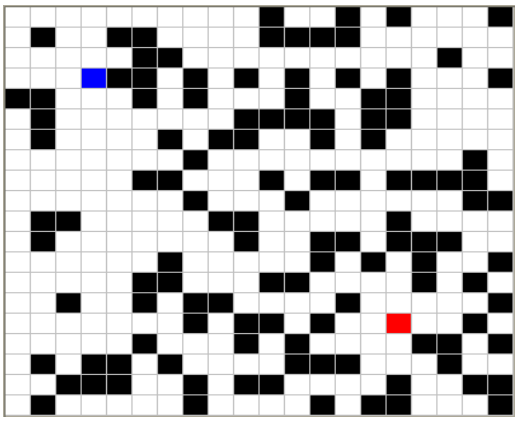

Fig. 4. Experiment on one grid diagram

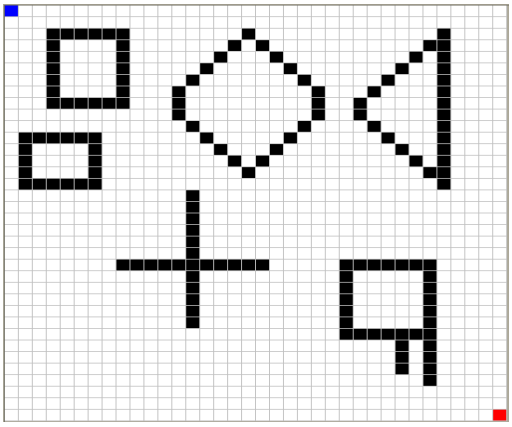

Fig. 5. Experiment on two grid diagram

Experiment one uses a grid map of size 20 × 20, the starting point of the grid position is (4, 4), the end point of the grid position is (16, 16), as it is shown in Fig. 4. Experiment two uses a grid map of size 36 × 36, the starting point of the grid position is (1, 1), the end point of the grid position is (36, 36), as it is shown in Fig. 5. In the grid diagram of experiment one and experiment two, each side of the grid is fixed at 1 cm. The black grid of the grid diagram is the barrier that cannot be passed through, the white grid shows no obstacle space that can pass through, the blue grid shows the end of robot motion paths, the red grid represents the starting path.

4.2. Experimental research

The simulation platform is MATLAB R2010A software and VC^{++} 6.0 software. Using MATLAB R2010A software development A* algorithm, the optimal path of the grid map is obtained, with VC^{++} 6.0 software drawing the optimal path diagram of A* algorithm searched for.

4.2.1. Experiment one

Figs 6 and 7 are the path optimization renderings based on an experiment with one grid diagram, where the path with the yellow grid in it is the optimal path from the starting point to the end point. The path composed of yellow grid is the optimal path with traditional A* algorithm searched in Fig. 6, the path composed of an yellow grid is the optimal path with improved A* algorithm searched in Fig. 7.

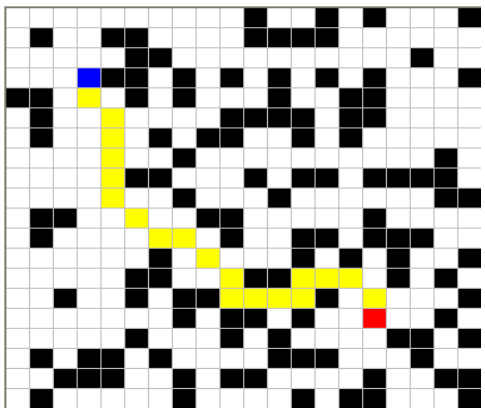

Fig. 6. The optimal path diagram with A* algorithm search

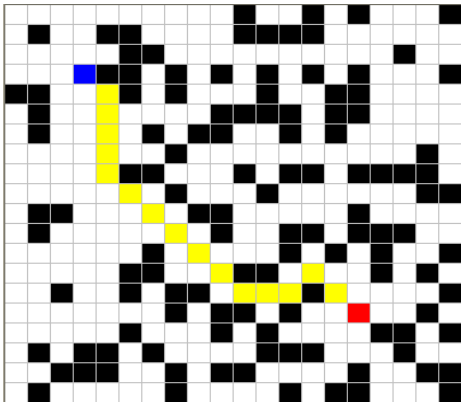

Fig.7. The optimal path diagram with improved A* algorithm search

Comparing Figs 6 and 7, it can be seen that the optimal path search with improved A* algorithm is shorter than with traditional A* algorithm search. Since A* algorithm and improved A* algorithms are run 20 times respectively, and then averaged, obtain the data shown in Table 2.

Table 2. Experiment 1 of the optimal path of two algorithms searches

Parameter	A* algorithm	Improved A* algorithm	The increasing percentage of performance
Optimal path length, cm	224.8528	201.4214	10.42%
Running time, s	0.0234	0.0078	66.66%

4.2.2. Experiment two

Figs 8 and 9 respectively are optimizing the structure of A* algorithm and optimizing the results of the improved A* algorithm.

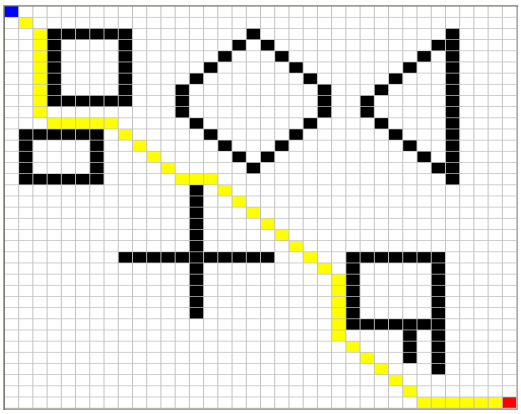

Fig. 8. The optimal path diagram with A* algorithm search

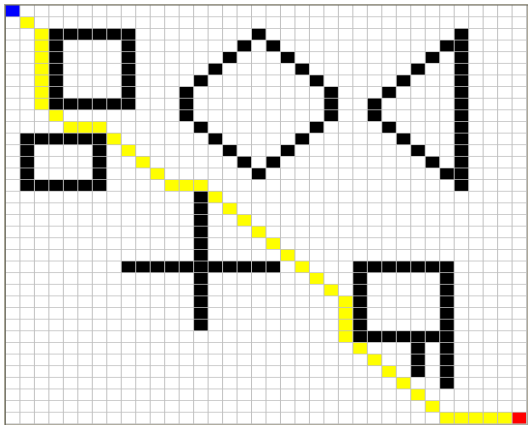

Fig. 9. The optimal path diagram with improved A* algorithm search

The following uses specific data to describe the optimal path difference of the two diagrams. Under the same conditions of running 20 times for averaging, the optimal length decreased by 3.11%, the running efficiency can be improved by 43.25% , as shown in Table 3 for the two algorithms of the optimal path contrast.

Table 3. Experiment 2˜ the optimal path of two algorithms searching

Parameter	A* algorithm	Improved A* algorithm	The increasing percentage of performance
Optimal path length, cm	565.2691	547.6955	3.11%
Running time, s	0.0289	0.0164	43.25%

From the experimental results of experiments one and two, it can be seen that the running time of the improved the A* algorithm is reduced more than 40%, while the shortest path is also with a slight decrease. The use of the structured data

members biao (i, j) instead of the query function of the OPEN table and CLOSED table, can efficiently reduce the operating frequency per query and improve the operating efficiency of the algorithm.

5. Conclusion

The paper provides structured data biao (i, j) instead of the improved method that looks up the OPEN table and the CLOSED table, so that the method can improve the operating efficiency of the algorithm theoretically. Through two different obstacles distributions, comparative experiments for A* algorithm and improved A* algorithm are executed, respectively running 20 times. The experimental data obtained show that the operating efficiency of the improved A* algorithm is increased by more than 40%, while the shortest path optimization is not obvious.

***Acknowledgments*:** The authors are highly thankful to Guangxi Natural Science Foundation (ID: 2013GXNSFBA019282), to the Research Program of Science at the Universities of Guang Xi Autonomous Region (ID: ZD2014112, 2013YB205). This research was financially supported by the project of outstanding young teachers' training in higher education institutions of Guangxi, Guangxi Colleges and Universities Key Laboratory Breeding Base of System Control and Information Processing.

References

1. Hart, P. E., N. J. Nilsson, B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths In Graphs. – IEEE Trans. Syst. Sci. and Cybernetics, Vol. **4**, 1968, No 2, pp. 100-107.
2. Wansen, W. Principles and Applications of Artificial Intelligence. Electronic Industry Publishing, 2000.
3. Zheng, K.-G. Katyn Zhuang Translation. Artificial Intelligence. Machinery Industry Publishing, 2000, p. 10.
4. Hart, P. E., N. J. Nilsson, B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. – IEEE Trans. Syst. Sci. and Cybernetics, Vol. **SSC-4**, 1968, No 2, pp. 100-107.
5. Li, L., Y. Tao, C. Langton. Current Situation and Future of Researching Move Robot Technology. – Robot, Vol. **24**, 2002, No 5, pp. 475-480.
6. Zhou, W. The Path Planning of Rescuing and Probing Robot in the Coal Mine and Trajectory Tracking Control Studying. Shanxi: Taiyuan University of Technology, 2011.
7. Xia, L. AFSA and Its Application. Guangxi, Guangxi University for Nationalities, 2009.
8. Yu, H., J. Wei, J. Li. Transformer Fault Diagnosis Based on Improved Artificial Fish Swarm Optimization Algorithm and BP Network. – In: Second International Conference on Industrial Mechatronics and Automation, 2010.
9. Lei, H. Researching Move Robot Path Planning Based on Neural Networks. Wuhan: Wuhan University of Technology, 2008.
10. Sehwartz, J. T., M. Sharir. A Survey of Motion Planning and Related Geometric Algorithms. – Artificial Intelligence, 1989, pp. 157-169.