A Non-asymptotic Space Complexity of a Backtracking Algorithm for the N-queens Problem

Adrijan Bozinovski[1], Stevo Bozinovski[2]
[1]School of Computer Science and Information Technology
University American College Skopje
Macedonia
{bozinovski@uacs.edu.mk}

[2]South Carolina State University
Orangeburg, SC
USA
{sbozinovski@scsu.edu}

ABSTRACT: *In this research we have studied the space complexity function to offset the shortcoming in the application memory. This is applied particularly when solving the N-queens issue. We found that it is essential to design the required memory space and to monitor the space complexity function in some input domain. We have described the detailed space complexity which is given for when the condition the value of N is less than forty.*

Keywords: Space Complexity, Back-tracking Algorithm, Nqueens Problem

Received: 10 March 2020, Revised 19 June 2020, Accepted 24 June 2020

Copyright: With Authors

I. Introduction

Backtracking algorithms are one of the solutions of the constraint satisfaction problem [1]. Such an algorithm searches for a path in a tree, keeping in memory unexplored promising segments of paths. This raises a question of memory requirements, known as the space complexity problem, which is the focus of this paper. Using a backtracking algorithm for solving a problem is a standard approach in the theory of algorithms [2] and Artificial Intelligence [3], in cases when no information is given about the tree to be searched. Basically it uses a generate-and-test method, in which a tree expansion routine generates the next set of promising nodes, and a tester routine tests them for constraint satisfaction. The backtracking class of algorithms has been studied since the 1960's [4] and basic techniques are well known[5][6]. A backtracking algorithm has exponential worst-case time complexity, but there are studies showing that under certain conditions their average time complexity may not be exponential [7][8].

Space complexity of a backtracking algorithm has attracted less attention than time complexity. Newer theoretical reasoning has pointed out that space complexity is possibly polynomial [1]. That points out possibility of using such an algorithm in embedded applications where application memory is limited. Usually those applications require bestapproximation rather than worst-case asymptotic approximation of needed space.

A prominent example of a problem which is being solved using a backtracking algorithm is the N non-attacking queens, in short the N-queens problem The N-queens problem is stated as: given a NxN chessboard, arrange N queens such that they do not attack each other . The problem has long history, starting in 1850 as a chess problem [9][10].

The N-queens problem is a benchmark problem and is used to test various types of algorithms. Among many approaches and applications let us mention integer programming [11], multitasking programming [12], linear congruence equations [13], dynamic programming [14], genetic algorithms [15], neural networks [16] and deductive database [17]. Currently, fast search algorithms in polynomial time are known for large N [18][19].

Bactracking can be considered as an operation defined over tree objects. Other operation are search, traversing, etc, many of the described in literature (e.g., [20]). Recently new operations on trees such as the Binary Roll Tree [21-24] have been proposed.

The solutions of the N-queens problem produce a particular pattern on a NxN matrix. An example of an N-queens pattern for N = 26 is given in Fig. 1 [25].

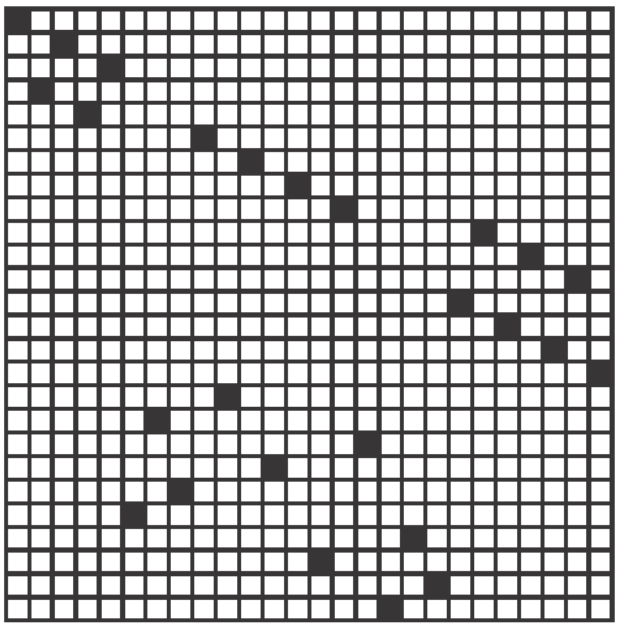

Figure 1. An N-queens pattern for N = 26

There are two variants of the problem: existence variant, in which only one solution is sufficient, and optimization variant, in which all feasible solutions need to be found and the optimal one to be chosen according to some optimality criterion. The numbers of solutions of the optimization variant for a given N are known, and for the first 10 values of N theyare 1, 0, 0, 2, 10, 4, 40, 92, 352, 724 respectively. The numbers grow exponentially and for N = 23 the number of possible solutions is over 24 trillion. Since the space complexity is the focus of this paper, the existence variant of the problem will be explored.

1.1. Scope of the Paper

This paper assumes an embedded system which has limited memory space for an application that runs a backtracking algorithm solving an N-queens type of problem. Since the memory is limited, we explore the space complexity for $N < N_{limit}$, in our case $N_{limit} = 40$. N_{limit} will depend on particular capacity of a embedded system.

2. Methodology

Firstly we provide a terminology that is we use in our research and we believe useful for both educational and research purposes.

A current path in a tree is a string of nodes passed up to the current node;

A promising path is the one whose last node is promising; *A* node is promising if it can produce a next feasible node;

A feasible node is one that satisfies some constraints; Given *N*, the promising path of length *k* is a string *e*[1]*e*[2]...*e*[*k*] where *e*[*k*] is a feasible node;

The promising path can be extended (toward a possible solution) or shortened (backtracked to a previous feasible path);

If $k = N$, a solution path of the problem has been found. Our approach is to look for polynomial bounds for a limited *N*, rather than finding asymptotic complexity for an unlimited *N*.

In a space complexity analysis the crucial issue is what should be kept in memory. For a backtracking algorithm it is the set of all promising paths. A promising path should be kept in memory either until it is explored or until a solution is found. A promising path in a search tree of the N-queens problem can be represented in various ways, and we used a string representation. For keeping in memory all the promising paths we used a stack data structure. The stack data structure is chosen because it implicitly resolves the issue of which promising path will be explored next – it is the one on the top of the stack.

The Java program written for this research is based on the following pseudocode:

```
Non-recursive backtrack algorithm:
        input N
        expand the parent node
        obtain the first N successor nodes
        stack them in reverse order
        repeat
                pop a node from the stack
                expand and test the expanded node
                        if(feasible AND solution_length = N)
                                then solution found, print it
                        if(feasible) push it into stack
    until empty stack
```

The solution is represented as a string which grows in length (solution-length) as the procedure progresses. At each step, a new element is appended from a set of possible expansions. The expansion set is tested for feasibility, and, if a node is not feasible, its subtree is pruned and the next element of the expansion set is considered. The length of the final solution is known to be *N*, and, once *N* is achieved, a test for a possible solution gives a solution of the problem.

3. Results

In our experimental research, we defined a space complexity function *SpaceQns*(*N*), as the length of the stack upon finding a solution to the N-queens problem. The values of *SpaceQns(N)* for $N < 40$, alongside the values of the functions N^2, $2N^2 2$, and $3N^2$, are shown on Figure 2.

More detailed observation gives the following description of space complexity of the *N*-queens problem, as represented by the *SpaceQns*(*N*) function:

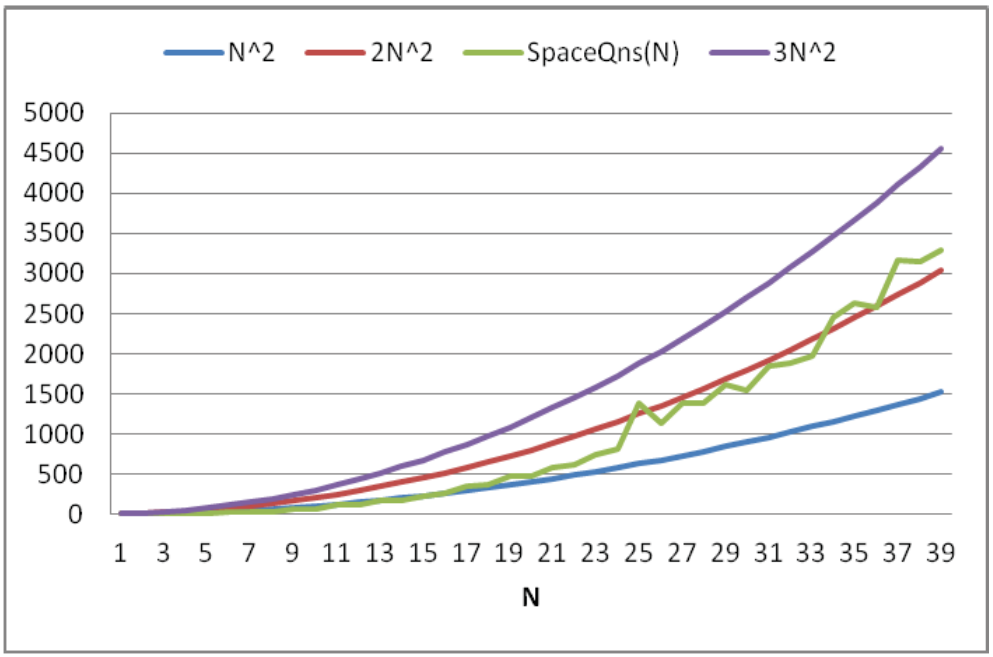

Figure 2. Space complexity function SpaceQns(N) in range $N < 40$

$$\text{SpaceQns(N)} \approx \begin{cases} \geq N^2, & N > 17, \text{ i.e. } \Omega(\text{SpaceQns(N)}) = N^2 \\ \leq 3N^2, & 4<N<N_3, \text{ i.e. } O(\text{SpaceQns(N)}) = 3N^2 \end{cases}$$

where $N_3 = min\{N|SpaceQns(N) \geq 3N^2\}$ is the point when $SpaceQns(N)$ passes the $3N^2$ bound. The value N_3 is larger than $N < 40$ and is outside our research scope, limited to $s < 40$. Within the experimental range of $1<N<40$, it can be seen that $©(SpaceQns(N)) = N^2$ (since $SpaceQns(N) > N^2$ for the entire range) and $O(SpaceQns(N)) = N^2$ (since $SpaceQns(N) < 3N^2$ for the entire range), which then leads to the conclusion that $\Theta(\text{SpaceQns(N)}) = N^2$ (as per [27]), i.e., the space complexity of the backtracking algorithm for solving the N-queens problem is tightly quadratic. On Figure 2 it can be seen that the $SpaceQns(N)$ function can be approximated with the function $2N^2$.

4. Conclusion

The paper provides a definition of terms used in backtrack programming and uses those terms in an experimental investigation on space complexity of a backtracking algorithm for solving N-queens problems. The scope of solution is considered for $N < 40$, which is a non-asymptotic case, a case oriented toward an application of an embedded system.

The experimental investigation shows that in that scope the space complexity is a quadratic function, more specifically it can be approximated with the function $2N^2$. A more detailed observation in various segments of $N < 40$ is also provided. A precise knowledge for space complexity function (in terms of $2N^2$ rather than $O(N^2)$) is especially needed in embedded applications, where memory management is part of application design.

References

[1] van Beek, P. (2006). *Backtracking Search Algorithms*. *In* Rossi, F. F., van Beck, P., Walsh, T. (eds.) Handbook of Constraint Programming, Elsevier, pp. 85-134, 2006

[2] Brassard, G., Bratley, P. (1996). Fundamentals of Algorithmics. Prentice Hall, 1996.

[3] Rich, E. (1983). *Artificial Intelligence*, McGraw-Hi11, 1983.

[4] Golomb, S., Baumert, L. (1965). Backtrack programming. *Journal of the ACM,* 12, p516-524.

[5] Fillmore, J., Williamson, S. (1974). On backtracking: *A combinatorial description of the algorithm*. *SIAM Journal of Computing,* 3, p 41-55.

[6] Bitner, J., Reingold, E. (1975). Backtrack programming techniques. *Communications of the ACM*, 18, p 651-656.

[7] Nicol, D. (1986). Expected performance of m-solution backtracking. *SIAM Journal on Computation,* 17(1), p 114-127, 1988.

[8] Stone, H., Sipala, P. (1986). The average complexity of depth-first search with backtracking and cutoff. *IBM Journal of Research and Development*, 30(3), pp. 242-258, 1986.

[9] Nauck, F. (1850). Schach. Illustrierter Zeitung 361:352, 1850.

[10] Pauls, E. (1874). Das Maximalproblem der Damen auf dem Schachbrete. *Deutsche Schachzeitung, Bd*. 29, p 129-134, 257-267.

[11] Foulds, L., Johnson, D. (1984). An application of graph theory and integer programming: Chessboard nonattacking puzzles. *Mathematics Magazine,* 57(3), p 95-104.

[12] Clapp, R., Mudge, T., Vol, R. (1986). Solutions to the N-queens problem using tasking in Ada. *SIGPLAN Notices,* 21, p 99-110.

[13] Erbas, C., Tanik, M., Aliyaziciogly, Z. (1992). Linear congruence equations for the solution of the N-queens problem. *Information Processing Letters,* 41, p 301-306.

[14] Rivin, L., Zabih, R. (1992). A dynamic programming solution to the n-queens problem, *Information Processing Letters,* 41, p 253-256.

[15] Crawford, K. (1992). Solving the n-queens problem using genetic algorithms. *In*: *Proceedings ACM/SIGAPP Symposium on Applied Computing*, Kanzas City, p 1039-1047.

[16] Shagrir, O., A neural net with self-inhibiting units for the N-queens problem. *International Journal of Neural Systems,* 3, p 249-252.

[17] Han, J., Liu, L., Lu, T. (1998). Evaluation of declarative n-queens recursion: A deductive database approach. *Information Sciences,* 105, p 69-100.

[18] Sosic, R., Gu, R. (1991). Fast search algorithms for the N-queens problem. *IEEE Transactions on Systems*, *Man, and Cybernetics,* 21(6), p 1572-1576.

[19] Sosic, R. (1994). A parallel search algorithm for the N-queens problem. *In*: *Proceedings Parallel Computing and Transputer Conference*, *Wollongong*, p 162-172.

[20] Knuth, D. (1997). *The art of Computer Programming*. Addison Wesley, 1997.

[21] Bo2inovski, A., Ackovska, N. (2012). The Binary Tree Roll Operation: Definition Explanation and Algorithm, *International Journal of Computer Applications*, 46 (8), p 40-47.

[22] BoNnovslci, A., Taney, G., Stojdevska, B., Patovski, V., Ackovska, N. (2016). Time complexity analysis of the binary tree roll algorithm, *Journal of Information Technology and Applications*, 6 (2), p 53-62.

[23] Bo2inovski, A., Taney, G., Stojtevska, B., Palovski, V., Ackovska, N. (2017). Space complexity analysis of the binary tree roll algorithm, *Journal of Information Technology and Applications*, 7 (1), p 9-19.

[24] Taney, G., Bcdinovski, A. (2017). A linear time algorithm for rolling binary trees, *In*: Proceedings of the 17th *IEEE International Conference on Smart Technologies IEEE EUROCON* 2017, Ohrid, Macedonia, p 255-260.

[25] Bozinovski, A., Bozinovski, S. (2004). N-queens pattern generation: An insight into space complexity of a backtracking algorithm, *In*: Proceedings of the 3rd *International Symposium on Information and Communication Technologies*. LasVegas, Nevada, USA, p 281-286.

[26] The On-Line Encyclopedia of Integer Sequences, published electronically at https://oeis.org, 2010, Sequence A000170.

[27] Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. (2009). Introduction to Algorithms. Third Edition. The MIT Press.