Scientific Journals
Maritime University of Szczecin

Zeszyty Naukowe
Akademia Morska w Szczecinie

2012, 29(101) pp. 25–29

2012, 29(101) s. 25–29

A comparison between Dijkstra algorithm and simplified ant colony optimization in navigation

Analiza porównawcza algorytmu Dijkstry i uproszczonego algorytmu mrówkowego w nawigacji

Mariusz Dramski

Maritime University of Szczecin, Faculty of Navigation
Akademia Morska w Szczecinie, Wydział Nawigacyjny
70-500 Szczecin, ul. Wały Chrobrego 1–2, e-mail: m.dramski@am.szczecin.pl

Key words: shortest path routing, restricted area, navigation

Abstract
In this paper, two different shortest path routing algorithms in respect of basic navigation problems are discussed. First of them is a "state of art" in computer science – well known Dijkstra algorithm. The second one is a method based on artificial intelligence – simplified ant colony optimization proposed originally by Marco Dorigo. Author used both ways to find an optimal / suboptimal route for a ship in a restricted area. Results showed the advantages and disadvantages of both algorithms in simple static navigation situations.

Słowa kluczowe: poszukiwanie najkrótszej drogi, akwen ograniczony, nawigacja

Abstrakt
W artykule omówiono dwa różne algorytmy poszukiwania najkrótszej drogi w odniesieniu do zagadnień nawigacji. Pierwszym z nich jest algorytm Dijkstry, stanowiący podstawę rozwiązywania tego typu problemów. Drugi to metoda bazująca na sztucznej inteligencji – uproszczony algorytm mrówkowy, zaproponowany przez Marco Dorigo. Autor używał obu sposobów w celu uzyskania optymalnej, bądź suboptymalnej trasy dla statku na akwenie ograniczonym. Rezultaty badań pokazały korzyści i wady ze stosowania obu rozwiązań w prostych sytuacjach nawigacyjnych.

Introduction

Conducting ships safely from the point of departure to destination is the basic task of each navigator. Safety requires also taking into account movement of other ships (objects), restrictions of the area itself, dynamics of own ship etc.

An increasing number of devices supporting navigator's work leads to such excess of information that it hampers taking the right decisions. In this connection research is being done on decision support systems in which different methods find increasingly wider applications.

This article describes and compares two different approaches to solve the problem of navigation in a restricted area. In every area, it can design and place a graph which represents all the possible moves for the ship. The main task is to find the shortest possible and safe route from departure to destination using the graph. It can be done by applying different methods such Dijkstra, Bellman-Ford, or A* algorithms. There are also other alternative methods based on artificial intelligence like evolutionary algorithms [1] or ant colony optimization [2]. In this paper, Dijkstra and SACO (Simplified Ant Colony Optimization) algorithms are compared. The purpose of this comparison is to show the advantages and disadvantages of both approaches.

Dijkstra algorithm

Dijkstra algorithm proposed in [3] is one of the most popular solutions for the shortest path problem. It finds the path with the lowest cost between a start vertex and every other vertex in the graph.

This algorithm is often used in network routing protocols and many other problems where graphs can be applied.

The main steps of Dijkstra algorithm:

- let s be the start node, $w(i, j)$ is the weight of edge i, j;
- create a distance matrix d for all the vertices of graph, assuming $d(s) = 0$ and $d(v) = \infty$ if there is no edge;
- create a priority queue Q, where the priority is a distance from the start node s;
- repeat until Q is not empty:
  - remove from the queue the vertex u with the lowest priority,
  - for each neighbour v of the vertex u, $d(v) = \min (d(u) + w(u,v), d(v))$;
- the last row of matrix d is a vector containing the shortest distance values from s to all the vertices of graph.

SACO algorithm

The observation of nature offers inspiration for seeking new solutions. One of them is ant colony optimization which comes from the insects (ants) living in colonies. It has been proved that ant colonies have some inherent optimization capabilities [4].

SACO (Simplified Ant Colony Optimization) is an alternative method based on artificial intelligence. This is the simpliest solution with the use of artificial ants. The insects are in one of two modes:

- "ahead",
- "return".

In the ahead mode ants move from the nest (original point) to the point where food is located (destination). When an ant achieves its goal its mode changes into the "return" and the insect goes back along the same trail. Ants in the ahead mode randomly choose the next node in the graph $G = (N, A)$, where: N – set of all points in the graph, A – set of all branches in the graph, on condition that there exists a suitable branch. This random selection is determined by the amount of pheromone left on the path between nodes. In the SACO algorithm, apart from the actual path, its length is remembered, so that one can estimate how much pheromone an ant lays on a given path. In this way ants relatively quickly begin to travel along the shortest route. In this context a few principles should be applied:

- pheromone "updating" takes place only on the way back;
- created loops should be eliminated;
- quality of solution is analyzed on the basis of the pheromone amount left.

Each branch (i, j) in the graph $G = (N, A)$ is related to the variable:

$$\tau_{ij} = 1, \forall (i, j) \in A$$

When an ant k is found in the node i, the above variable can be used to calculate the probability of choosing the route to the node j:

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha}}{\sum_{i \in N_i^k} \tau_{ij}^{\alpha}} & \text{for} \quad j \in N_i^k \\ 0 & \text{for} \quad j \notin N_i^k \end{cases}$$

where:

- N_k^i – neighbourhood of the ant k when it is in the node i. This neighbourhood contains all other nodes directed to i, except the preceding node (this does not refer to a situation where the node i is an extreme node – there is no other way than the way back;
- α – parameter determined experimentally (value $\alpha = 2$) [1];
- τ_{ij} – will be called the artificial pheromone trace. This trace is read out and recorded by ants. The pheromone update can be done according to this formula:

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \Delta\tau_{ij} \quad \text{where} \quad \Delta\tau_{ij} = 1$$

The phenomenon of pheromone evaporation is not taken account in the SACO algorithm.

For the algorithm to work, it has to define the graph of all possible paths that ants may follow. The correct creation of the graph is extremely important as it will make up a basis for the proper representation of the examined area in the form of points of a grid. How fast the ant algorithm will operate depends on the number of these points and their actual distribution.

Comparison – infinite graph

The infinite graph is each graph which has infinite sets $V(G)$ and $E(G)$ (vertices and edges respectively). If every vertex in the graph has finite number of edges, the graph is called locally finite. This kind of graphs is very useful, for example for doing some tests.

Few fragments of an infinite graph of squares (locally finite) were taken under research for example (Fig. 2). Every vertex of this graph has 8 edges (sides of the squares ant it's diagonals – except external vertices). The bottom left node (number 1) of every fragment is a start node for shortest path routing algorithm and the top right one (number n) is the stop node. Then it's easy to observe that the

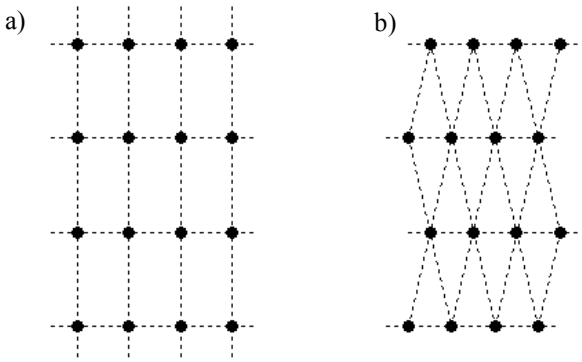

Fig. 1. Examples of infnite graphs; a) infinite graph of squares, b) infinite graph of triangles
Rys. 1. Przykłady wykresów nieskończonych; a) nieskończony wykres kwadratów, b) nieskończony wykres trójkątów

shortest path will always be a diagonal of the square. Two parameters were taken into a consideration:

– the average time of finding a solution;
– the length of found path.

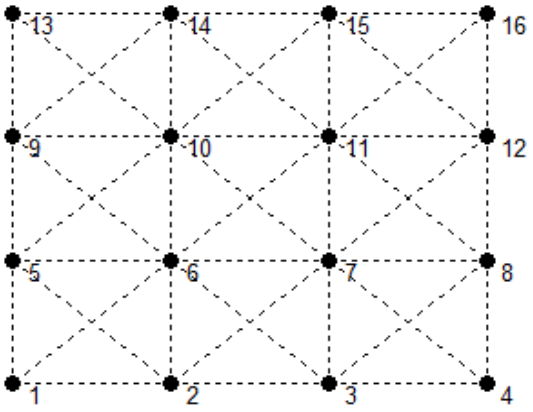

Fig. 2. 4×4 section of infinite graph
Rys. 2. Sekcja 4×4 wykresu nieskończonego

The results are illustrated in table 1 (results discussion is placed in paragraph comparison – summary of this paper).

Table 1. Experiment results
Tabela 1. Wyniki eksperymentu

n	Time SACO	Time Dijkstra	Length SACO	Length Dijkstra
4	1.059524	0.015662	1	1
9	1.745180	0.016641	4	2
16	1.282009	0.007564	5	3
25	1.859087	0.014378	7	4
36	33.795519	0.024027	12	5
49	2.633760	0.034885	12	6
64	27.584932	0.043464	14	7

n – the number of nodes in the fragment of infinite graph; Time SACO – average time of finding a solution for SACO algorithm; Time Dijkstra – average time of finding a solution for Dijkstra algorithm; Length SACO – the length of path for SACO algorithm; Length Dijkstra – the length of path for Dijkstra algorithm.

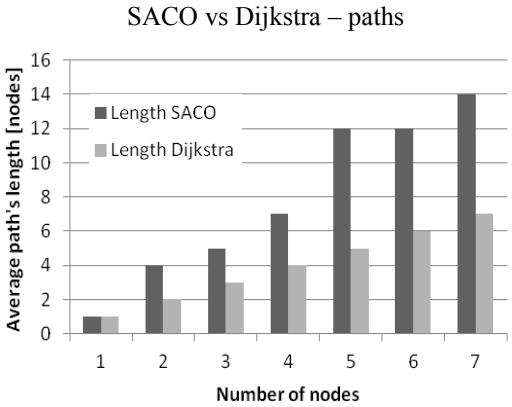

Fig. 3. The comparison of paths' length for each algorithm
Rys. 3. Porównanie długości tras dla każdego algorytmu

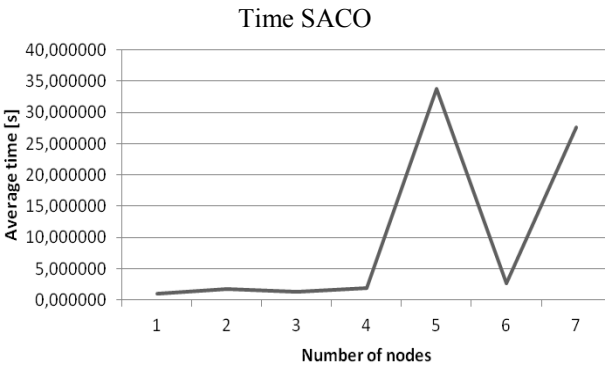

Fig. 4. Average time for SACO algorithm
Rys. 4. Średni czas dla algorytmu SACO

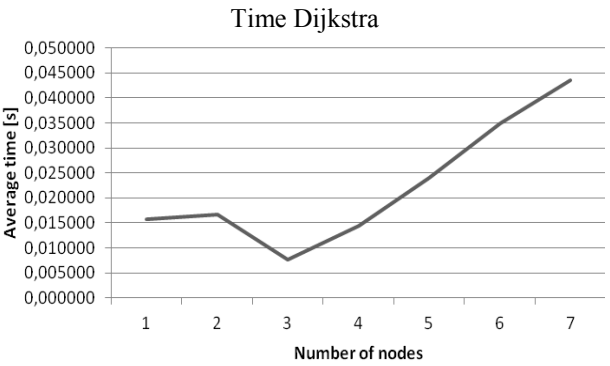

Fig. 5. Average time for Dijkstra algorithm
Rys. 5. Średni czas dla algorytmu Dijkstra

Comparison – an example area

An example restricted area was created. It's illustrated at figure 6. The graph of possible paths is a set of simple square graphs which also can be treated as fragments of an infinite graph. The purpose of this choice was simple – easy to see the shortest path even without doing research.

There are many ways to build a graph of paths. In [5] the use of quadtrees is proposed. Anyway, creating a graph still remains significant problem and requires further consideration.

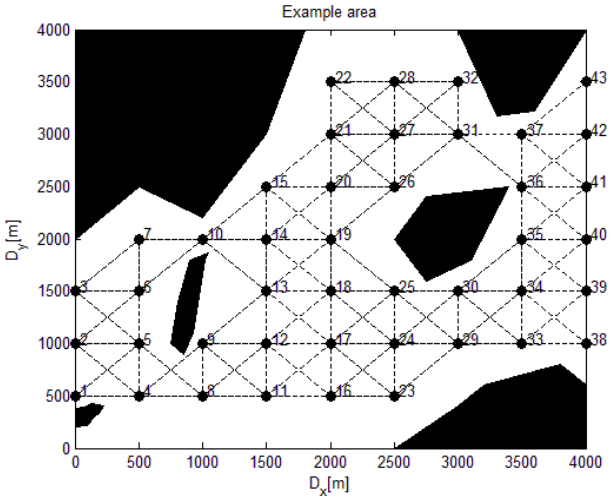

Fig. 6. Example area with a graph of possible paths
Rys. 6. Przykładowy obszar z wykresem możliwych tras

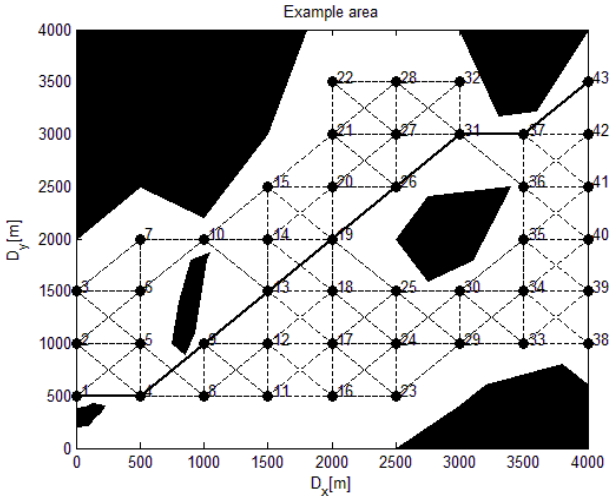

Fig. 7. Dijkstra algorithm result
Rys. 7. Wyniki algorytmu Dijkstra

In table 2 the results of experiment were illustrated. This data conducts to a conclusion that well known method of shortest path routing such Dijkstra algorithm is better then new, alternative methods (in this case SACO algorithm). Besides, it has to be told, that in the case of SACO algorithm only the best obtained route is presented. This method does not guarantee any optimality, so there is a problem with the repeatability too. Every single use of SACO algorithm can result other solution.

Table 2. Comparison between Dijkstra and SACO algorithms – experiments' results
Tabela 2. Porównanie algorytmów Dijkstra i SACO – wyniki eksperymentów

Algorithm	Route [nodes]	Route's length [nodes]	Route's length [m]	Time [s]
Dijkstra	1-4-9-13-19-26-31-37-43	9	5242.6	0.04331
SACO	1-4-9-13-14-20-27-31-37-43	10	5535.5	12.49039

Obtained paths can also be compared by analyzing figure 7 and figure 8. As it can be seen both routes are safe and possible, but Dijkstra algorithm let it be quite shorter in this case.

Comparison – summary

The discussion about these two algorithms can be based on informations presented in table 1. Few important features were taken into consideration. First of all, the most important question is if the algorithm can guarantee the optimality. Dijkstra algorithm is always able to find a shortest, optimal path (Fig. 7). SACO algorithm very often chooses other suboptimal paths (Fig. 8), so doesn't guarantee optimality. Besides, the alternative way requires more execution time.

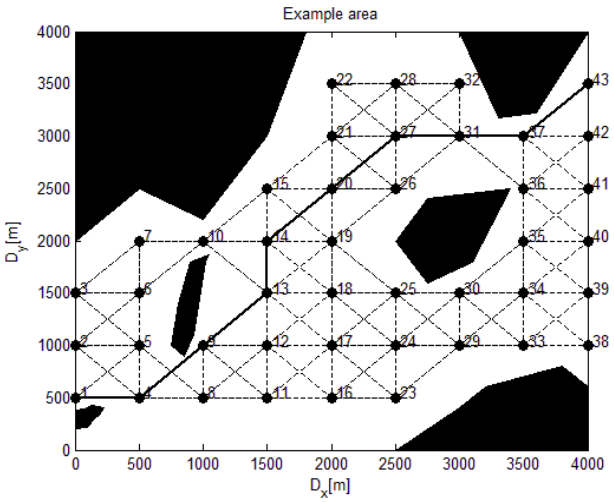

Fig. 8. SACO algorithm result
Rys. 8. Wyniki algorytmu SACO

Table 3. Comparison between Dijkstra and SACO algorithms
Tabela 3. Porównanie algorytmów Dijkstra i SACO

	Dijkstra	SACO
Guarantee optimality	Yes	No
Computational coplexibility	$O(n^2)$	$O(n^3)$
Time	Shorter	Longer
Distances	From source to all others	From source to destination
Numbering of nodes	Free	Almost free (source must be 1, destination must be *n*)
Route repeatibility	Yes	No

Conclusions

There is a lot of proposals of solving a problem of shortest path routing. One of the most popular ways to solve is Dijksta algorithm. Sometimes, if classical methods are not able to find a solution or have some difficulties, other alternative ones can

be used. SACO algorithm based on the theory of artificial ants is one of such ideas. It's able to find a suboptimal route. A comparison between these two algorithms was made. The research prove that Dijkstra algorithm is a better way to solve the problem of shortest path routing in graph. It's faster, guarantees optimality and has less computational complexilibity. It's necessary to tell that only static situations were investigated.

Besides the SACO algorithm is a worse way in shortest path routing, the whole theory of ant colony optimization can't be omitted in further research. Researchers make a lot of work on this topic and more and more literature can be found.

References

1. ŚMIERZCHALSKI R., MICHALEWICZ Z.: Modeling of ship trajectory in collision situations by an evolutionary algorithm. IEEE Transactions on Evolutionary Computation, Vol. 4, 2000, 227–241.
2. DORIGO M., STUTZLE T.: Ant Colony Optimization. MIT Press, 2004.
3. DIJKSTRA E.W.: A note on two problems in connexion with graphs. Numerische Mathematik, 1, 1959, 269–271.
4. DENEUBOURG J.L., ARON S., GOSS S., PASTEELS J.M.: The self-organizing exploratory pattern of the Argentine ant. Journal of Insect Behavior, 3, 1990, 159–168.
5. DRAMSKI M., MĄKA M.: The choice of ship's safe route in a restricted area with the use of quadtrees for a simplified ant algorithm. Marine Traffic Engeneering Conference MTE 2011, October 12–14, Świnoujście 2011.