AARMS Vol. 12, No. 2 (2013) 289–301. **10.32565/aarms.2013.2.10**

The AD–AHP planner: Incremental path planning for robots incorporating decision theory

KOMLÓSI István[1]

Path planning for mobile robots in dynamic environments with many objectives to regard at once is challenging and often impossible. However recent outdoor robot- ic applications encounter many situations when such an approach is needed. This paper presents a path planning method for mobile robots that incorporates deci- sion theory to guide the search. The method is able to handle an arbitrary number of objectives at the same time and also enables the incorporation of human logic into the planning process. All parts of the algorithm suit real–time implementation.

Introduction

Recent developments in field robotics and autonomous vehicles have raised the demand for advanced and intelligent navigation systems capable of planning safe and dynamically feasable trajectories while minimizing objectives like path length, energy cost and trip time. In a dynamic environment the relevance of the objectives and their relation to each other can also change rapidly, i.e. if the robot has limited time to reach its goal destination minimizing the total trip time has higher priority over energy consumption until changes in the environ- ment like a sudden blockade on its previously planned path forces the robot to choose a more energy efficient yet more time consuming path.

In the case of unmanned ground vehicles, when passing through rugged terrain the robot can be exposed to risk. Consider a planetary rover with limited battery charge and limited time to reach its destination. The robot has to navigate through a canyon where the solar cells cannot recharge the robot's battery efficiently. At the same time the robot is exposed to the risk of turning over on the rocky surface or getting trapped due to falling rocks. The robot's navigation system has to be capable of planning an energy efficient but safe path avoiding dangerous areas while incorporating the risk of getting trapped within the canyon. In the case of the Mars rovers such an operation could result in losing the robot.

Robots applied in military operations can also be exposed to significant risk. Consider a military robot that has a rescue mission. It has limited time to reach its destination and also a limited battery charge that it must not consume before it reaches the goal state. Between the robot's current state and the goal destination lies a mine field. The robot's navigation system has to be capable of planning a safe path while continously evaluating the risk of damage when passing through the mine field.

Robots often have to navigate in dynamic and cluttered environments with imperfect in-
formation on the terrain. Efficient replanning of the path is needed since a previously planned

1 National University of Public Service, Budapest, Hungary

path can be invalidated by obstacles like falling rocks, or due to imperfect map information when new sensory information received show previously low risk areas to be high risk areas.

Optimizing many objectives at once is challenging. The robot's path planning problem results in minimizing a dynamically changing multi–cost function. Minimizing such a cost function can be time consuming and computionally cumbersome. Also the robot has to per- form this operation frequently when replanning is needed. Instead of finding the minimum of such a cost, function decision theory has to be incorporated into the path planning process.

We propose a modified version of the widely used D* algorithm, where determining the
focus of the search is performed using the Analytic Hierarchy Process (AHP) method.

This paper is organized as follows. In the next section the AHP method is discussed fol- lowed by a short overview on incremental planners for robots. After that we introduce the AD–AHP method that incorporates decision making into the incremental planner. After the simulationresults the conclusion ends the study.

The Analytic Hieararchy Process

The Analytic Hierarchy Process introduced in the relevant documents (Saaty, 1980), (Saaty,
1988) is an efficient method for multi–criteria decision making. Given a set of alternatives and a set of criteria AHP selects the best alternative. The AHP method has been used in machine learning for artifitial intelligence applications (Kong, Liu, 2005), also it proved to be useful in military engineering for selecting optimal military equipment (Gyarmati, Kende, Felházi,
2009), (Gyarmati, 2006), (Kavas, 2009). The study (Fallahi, Leung, Chandana, 2009) intro- duces a method where the Fuzzy–AHP was used along with the Ant Colony Optimization path planner for selecting an appropriate UAV for a mission. Another application has shown AHP to be useful in disaster management (Mali, Rao, Mantha, 2013) where finding the best root in an emergency case was carried out by Dijkstra's algorithm on a graph where the arc costs were determined by AHP. The study (Vaidyaa, Kumarb, 2006) gives a general overview on recent AHP applications.

As given in the decision matrix below, the decision system has *M* alternatives and *N* cri- teria. The entries of the decision matrix are the performance values of the alternatives within each criterion. To determine the best alternative these entries are weighed by the weighing coefficients that express the importance of a criterion.

$$\begin{array}{cccccc}
 & & & Criterion & & \\
 & C_1 & C_2 & C_3 & \dots & C_N \\
Alt. & W_1 & W_2 & W_3 & \dots & W_N \\
A_1 & a_{11} & a_{12} & a_{13} & \dots & a_{1N} \\
A_2 & a_{21} & a_{22} & a_{23} & \dots & a_{2N} \\
A_3 & a_{31} & a_{32} & a_{33} & & a_{3N} \\
 & & & & \dots & \\
\dots & \dots & \dots & \dots & \dots & \dots \\
A_M & a_{M1} & a_{M2} & a_{M3} & \dots & a_{MN}
\end{array}$$

Analytic Hierarchy Process performs pairwise comparison within each criterion. A judge- ment matrix has to be established for each criterion (not to be confused with the formally mentioned decision matrix)

$$\begin{array}{cccc} C_i & A_1 & A_2 & A_3 \\ A_1 & a_{11} & a_{12} & a_{13} \\ A_2 & 1/a_{12} & a_{22} & a_{23} \\ A_3 & 1/a_{31} & 1/a_{32} & a_{33} \end{array}$$

where $a_{ij}=1/a_{ji}$ are values from the relative scale of importance according to Saaty's book (1980). When $a_{ik}a_{kj}=a_{ij}$ is satisfied for all entries in the matrix then the judgement matrix is consistent. Judgement matrices are often inconsistent since human decisions cannot always assure consistent rankings. The W_i weighing coefficients can be derived from calculating the priority vector according to the judgement matrix for the criteria.

AHP works as follows. One has to determine the priority vector that is the right principal eigenvector of the judgement matrix for each C_i criterion. It has been shown in article (Saaty, 2003) along with other thoughts on the priority vector that it must be the right principal ei- genvector of the judgement matrix that satisfies

$$A\omega=\lambda_{\max}\omega$$

where $\lambda_{\max}$ is the maximal eigenvalue and ω is the eigenvector. This is especially import- ant in the case of inconsistent matrices. One way to obtain the priority vector is to approxi- mate it by taking the geometric means of each row and then normalizing them by their sum so that the entires of the priority vector can be approximated by

$$p_i = \frac{\sqrt[n]{\prod_{j=1}^{n} a_{ij}}}{\sum_i \sqrt[n]{\prod_{j=1}^{n} a_{ij}}}$$

Next the consistency index *CI* has to be estimated so that it can be derived from the formula

$$CI = \frac{\lambda_{\max} - n}{n-1}$$

The consistency ratio *CR* is to be calculated by dividing the consistency index by the Random Consistency Index that is often given in tables (Saaty, 1980) as a function of *n*. The consistency ratio refers to the consistency property of the judgement matrix. It has been shown in (Saaty, 1980) that if the consistency ratio is greater than 0.1 it is preferable to rees- tablish the judgement matrix.

The entries of the decision matrix are the entries of the priority vectors for each criterion. The final priorities can be calculated by taking the scalar product of each row according to the alternatives and the weight vector that is

$$A^i_{AHP} = \sum_{j=1}^{n} a_{ij} w_j \qquad for \quad i = 1,2,3\ldots \quad M$$

where A^i_{AHP} the final priority of the *i*th alternative.

Incremental planning

Robot path planning in dynamic environments has been in the focus of robotics research for a long time (Latombe, 1991). Many methods have proven to be efficient and useful. The study (LaValle, 2006) gives a general overview on planning algorithms. The popular poten- tial field's method (Hwang, Ahuja, 1992) has proven to be efficient for simple path planning tasks. Since it suffers from the problem of getting trapped in local minima, different versions of this method were developed to overcome this difficulty (Fox, Burgard, Thrun, 1997), also the Probabilistic Road Map method has been widely introduced (Kavraki, Svestka, Latombe, Overmars, 1996). The likewise popular Rapidly Exploring Random Tree (RRT) method in- troduced in (LaValle, 2006) also has many variants (Ferguson, Kalra, Stentz, 2006), (Fergu- son, Stentz, 2006a), some of them are capable of planning in high dimensional search spaces (Ferguson, Stentz, 2007). Soft computing methods like genetic algorithms can be incorporat- ed in path planning, also neural networks and fuzzy–systems have been widely used for this task (Li, Yang, Seto, 2009). The (Yanduo, Kun, 2009) study presents a path planning method, using Liquid State Machines, that is a member of the family of recurrent neural networks, and shows promising results (Maass, Natschlaeger, Markram, 2002).

When detailed information on the terrain is available in the form of a map for the robot's navigation system, map based path planners prove to be useful. Among map based planners grid based path planning methods were frequently addressed since incremental planners like A* (Hart, Nilsson, Raphael, 1968) are capable of solving the planning problem. The main idea of grid based path planning is dividing the map into cells where each cell has an associ- ated traversal cost. When a cell is occupied by an obstacle the cell's traversal cost is set to a high value (or to infinity). A graph is built where the graph nodes correspond to the corner or the center of the grid cells while the edge costs correspond to the traversal cost of the cells. The objective of minimizing the overall path cost, which is the sum of the traversal costs of the cells the robot passes through, can be performed by finding the shortest path in the cor- responding graph.

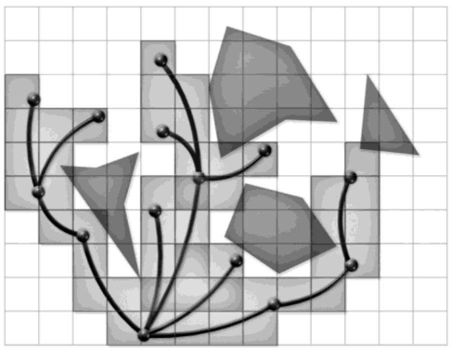


Figure 1. The concept of grid based planning[2]

Incremental planners keep a record of the path cost of every visited node. At the same time a heuristic cost is estimatated from the dedicated node to the goal or to the start state depending on the search direction. Each visited node is inserted into a priority queue called an open list with a key value that is the sum of the path cost and the heuristic cost estimate. When the heuristic does not overestimate the real path cost to the goal the resulting path will be optimal. The open list thus contains the potential nodes that the search can be continued from and the search is focused towards the node with minimal key value on the open list. The heuristic cost estimate focuses the direction of the search and vastly influences the optimality of the solution. Finding efficient heuristics in a multi–cost environment is challenging and is often impossible in high dimensional search spaces.

Every time a node is expanded, i.e. its neighbours are visited, it is taken off the open list. Incremental planners also maintain a list for backpointers. When a previously visited node is revisited by expanding a node from the open list and the node's cost value decreases due to the expansion the state is updated, that is, the cost value is set to the decreased value and the node's backpointer is set to the expanded node. This way the algorithm can keep a record of the best path obtained so far.

The D* algorithm (Stentz, 1994) is the dynamic version of A*. It has been proven to be at least two orders of magnitude more efficient than planning from scratch with A* when chang- es in the arc costs occur. The main idea was to regard only those affected states that could potentially improve or invalidate the previously planned path. A one step lookahead of the path cost is used for this reason to identify inconsistent states where arc costs have changed. When the grid resolution of the map was insufficient but the process had to operate with fixed memory capacity, the Multi–Resolution Field D* (Ferguson, Stentz, 2006b) algorithm was developed, which is now currently used as the primary path planner along with TEMPEST onboard the Mars rovers Spirit and Opportunity (Carsten, Rankin, Ferguson, Stentz, 2007).

2 http://www.robodesign.ro/marius/my-projects/illustrations/motion-planning (downloaded: ca. 2013)

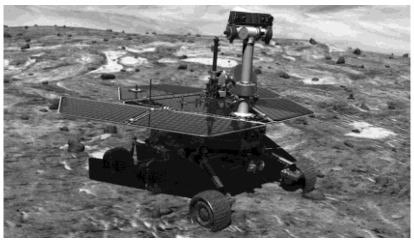


Figure 2. Mars rover[3]

In a large state space, as mentioned above, efficient heuristics are needed to focus the search. Another way to deal with the large number of states is to define dominance relations. The DD* algorithm (Korsah, Stentz, Dias, 2007) efficiently prunes the search space by using dominance relations, i.e. when a node is dominated by another by the mean of dominance, the search will be aborted from the dominated node. In a rapidly changing environment non– dominated states can turn dominated and vice–versa.

The most recently introduced member of the D* family is the Anytime Dynamic A* (AD*) planner (Likhachev, Ferguson, Gordon, Stentz, Thrun, 2005), which is capable of planning in rapidly changing environments by continously improving previous solutions through refin- ing its heuristic cost estimate. Since it is an anytime algorithm it runs until elaboration time allows and within the given time frame it produces a path with a suboptimality bound.

We will take the AD* planner as a basis of the AD–AHP algorithm that is introduced in the next section.

The AD–AHP path planner

The AHP improved AD* algorithm works as follows. Instead of determining a single cum- mulative cost function that sums up all costs, i.e. path length, time, energy, risk, etc. with given weights and instead of defining dominance relations a priority queue is maintained for each cost function where nodes are inserted with their corresponding key values when visited. The key values similarly to the basic operation are the sum of the given cost (i.e. energy cost, time cost) and the heuristic cost estimate. The cost of risk can be expressed as a probability value so the key value of a node on the “risk open list” must be calculated using an appropriate method in probability theory. It is assumed that the ranking of the objectives is done by another process that monitors the robot’s vicinity constantly and decides the best ranking of the objectives in any situation. Some neural network approchaes like the Learning Vector Quantization (LVQ) method (Kohonen, 1995) could be used for this reason.

First the start state is inserted into each open list and the neighbouring nodes of the start state are visited. For each visited node a cost value, a one step lookahead and a heuristic cost estimate is calculated according to all cost functions. The visited nodes are then inserted into the corresponding priority queues.

The next step is to determine which node to select for expansion. Given the actual ranking of the criteria, i.e. the ranking of the different costs and given a finite set of alternatives, i.e. graph nodes with minimun key values in each priority queue by using AHP, the best node to

3 http://www.universetoday.com/50930/mars-rover-pictures/ (downloaded: ca. 2013)

continue the search from can be determined. The AHP procedure has to be performed every time before taking a node off the open lists. It is wise not to take only the nodes with the best key values but to take nodes from a finite horizon, say the four or five best nodes in order to gain a greater set of alternatives. Of course, this could result in extra computational expense.

A quintessential question of the method is when to update a node's cost value if it is re- visited, since an improvement in energy cost could result in an increase in the time cost and vice–versa. The main idea is to update states according to the weight vector of the criteria. Let
us denote the actual path length of the current node with d_0, its energy cost with E_0, time–cost with t_0, and the cost or probability of risk with r_0. Assume that after updating the state one gets d_1, E_1, t_1, r_1, for the same values. The change of the values can be expressed in proportions of each other in a vector. This vector is then multiplied by the weight vector of the criteria that is assumed to be normalized to unity. Given the weighing coefficients w ,w ,w,w, when

$d \quad E \quad t \quad r$

$$\begin{pmatrix} w_d & w_E & w_t & w_r \end{pmatrix} \begin{pmatrix} d_1 \blacksquare d_0 \\ E_1 \blacksquare E_0 \\ t_1 \blacksquare t_0 \\ r_1 \blacksquare r_0 \end{pmatrix} < \begin{pmatrix} w_d & w_E & w_t & w_r \end{pmatrix} \begin{pmatrix} d_0 \blacksquare d_1 \\ E_0 \blacksquare E_1 \\ t_0 \blacksquare t_1 \\ r_0 \blacksquare r_1 \end{pmatrix}$$

is satisfied, the state can be updated, i.e. the cost values are set and the node's backpointer is set to the currently expanded node. It is assumed that all values are nonnegative, and zero entries in the above equation should be discarded and left out of the comparison.

The algorithm operates on as AD*, after reaching the goal state the heuristics are im- proved and the search restarts. It has to be mentioned that if a dramatic change in the envi- ronment or a dramatic reordering of the objectives occurs, it is wise to replan from scratch instead of trying to improve already existing solutions.

The main advantage of this approach is that human logic can be incorporated into the path planning process and thus an arbitrary number and arbitrary kinds of objectives can be handled at once. The robot can often find itself in situations when ranking the objectives by importance requires human experties. If the robot's navigation system is capable of learning, with human aid, it is possible for the robot to build up and contniuosly update a database of the rankings and apply the previously mentioned LVQ or some other method to determine the proper ranking when it finds itself in unfamiliar situations.

The formal description of the AD–AHP can be found in the Appendix. The main steps are of AD* (Likhachev, Ferguson, Gordon, Stentz, Thrun, 2005), the relevant modifications are where the AHP was incorporated into the algorithm.

Simulation Results

Simulations were carried out with four objectives, for cell traversal costs the cost values according to the different objectives were set randomly. Given was a start state, a goal state and three obstacles. The robot was regarded a point robot. Fig. 3. – Fig. 6. show the result of the path planning when each objective was regarded independently. On each picture the cells with a darker shade have higher cost value. On Fig. 7. and Fig. 8. the results can be seen when the planner had to regard all objectives at once first with weight vector and then with weight vector .

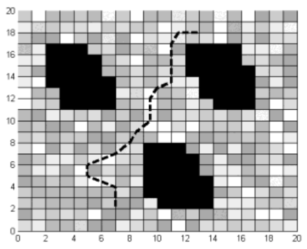
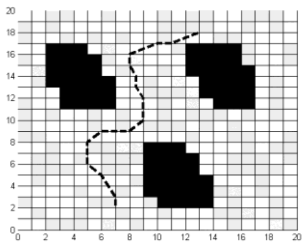
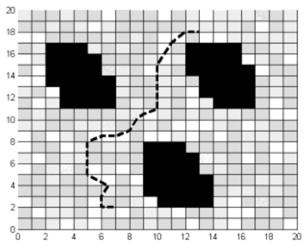


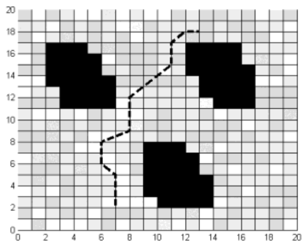
Figure 3. Planning results with the first objective Figure 4. Planning results with



the second objective Figure 5. Planning results with the third objective Figure 6.



Planning results with the fourth objective



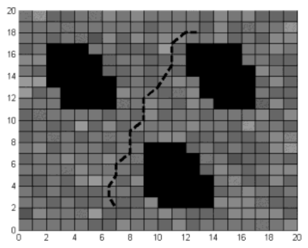


Figure 7. Planning results with all objectives with the first weight vector

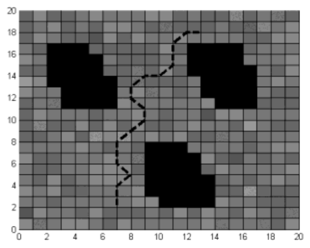


Figure 8. Planning results with all objectives with the second weight vector

Conclusions

We have introduced a novel incremental planner that is capable of planning collision free trajectories for robots in a dynamic multi–cost environment while incorporating decision the- ory. Instead of using a single cost function and state dominance, Analytic Hierarchy Process was used along with AD* to determine the focus of the search. The advantage of the algo- rithm is that very different objectives can be handled at the same time with the incorporation of human logic. All parts of the algorithm suit real–time implementation.

References

CARSTEN, J., RANKIN, A., FERGUSON, D., STENTZ, A. (T.) (2007): *Global Path Planning on–board the Mars Exploration Rovers*, IEEE Aerospace Conference https://doi.org/10.1109/AERO.2007.352683

FALLAHI, K., LEUNG, H., CHANDANA, S. (2009): *An Integrated ACO–AHP Approach for Resource Management Optimization,* Proceedings of the 2009 IEEE International Conference on Systems, Man and Cybernetics USA: San Antonio, October, pp. 4335–4340. https://doi.org/10.1109/ICSMC.2009.5346794

FERGUSON, D., KALRA, N., STENTZ, A. (T.) (2006): *Replanning with RRTs,* Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 1243–1248. FERGUSON, D., STENTZ, A. (T.) (2006a): *Anytime RRTs,* Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS ‘06), pp. 5369–5375 FERGUSON, D., STENTZ, A. (T.) (2006b): *Multi–resolution Field D**, Proceedings of the International Conference on Intelligent Autonomous Systems (IAS)

FERGUSON, D., STENTZ, A. (T.) (2007): *Anytime, Dynamic Planning in High Dimensional Search Spaces,* IEEE International Conference on Robotics and Automation (ICRA)

FOX, D., BURGARD, W., THRUN, S. (1997): The dynamic window approach to collision avoidance, In. *Robotics & Automation*, IEEE, Vol. 4 Issue 1, pp. 23–33. https://doi.org/10.1109/100.580977

GYARMATI, J. (2006): Haditechnikai eszközök összehasonlítása közbeszerzési eljárás során, In. *Hadmérnök* Vol. 2. pp. 68–93. ISSN 1788–1919

GYARMATI, J., KENDE, Gy., FELHÁZI, S. (2009): Choosing the Optimal Mortar for an Infantry Batallion's Mortar Battery with Analytic Hierarchy Process using Multivariate Satistics, In. *Decision Support Methodologies for Acquisition of Military Equipment*, Brussels,
2009. 10. 22–2009. 10. 23, NATO RTO, pp. 1–12., ISBN: 978–92–837–0101–9

HART, P. E., NILSSON, N. J., RAPHAEL, B. (1968): A Formal Basis for the Heuristic Determination of Minimum Cost Paths, In. *Systems Science and Cybernetics*, IEEE Transactions on SSC4 Vol. 4 Issue 2, pp. 100–107

HWANG, Y. K., AHUJA, N. (1992): A potential field approach to path planning, In. *Robotics and Automation*, IEEE Transactions on, Vol. 8 Issue 1, pp. 23–32. https://doi.org/10.1109/70.127236

KAVAS L. (2009): *Harcászati repülőgép kiválasztásának módszere gazdasági – hatékonysági mutatók alapján kis létszámú haderő légierejének korszerűsítésére*, PhD–tézisek, Szolnok, pp.
40–45.

KAVRAKI, L. E., SVESTKA, P., LATOMBE, J. C., OVERMARS, M. H. (1996): Probabilistic roadmaps for path planning in high–dimensional configuration spaces, In. *Robotics & Automation*, IEEE Transactions on, vol. 12 Issue 4 pp. 566–580.

KOHONEN, T. (1995): Learning vector quantization, In. *The Handbook of Brain Theory and Neural Networks*, Cambridge: MIT Press, pp. 537–540.

KONG, F., LIU, H. (2005): *An Improvement on Saaty's AHP, Artificial Intelligence Applications and Innovations,* The International Federation for Information Processing Volume 187, pp. 301–312. KORSAH, G. A., STENTZ, A (T.), DIAS, M. B. (2007): *DD* Lite: Efficient Incremental Search with State Dominance*, Tech. Report CMU–RI–TR–07–12, Robotics Institute, Carnegie Mellon University

LATOMBE, J. C. (1991): *Robot Motion Planning,* Boston: Kluwer Academic Publishers

LaVALLE, S. M. (2006): *Planning algorithms,* Cambridge: University Press, 2006

LI, H., YANG, S. X., SETO, M. L. (2009): Neural–Network–Based Path Planning for a Multirobot System with Moving Obstacles, In. *Systems, Man and Cybernetics*, IEEE Transactions on, Part C: *Applications and Reviews*, Vol. 39 Issue 4, pp. 410–419. https://doi.org/10.1109/TSMCC.2009.2020789

LIKHACHEV, M., FERGUSON, D., GORDON, G., STENTZ, A. (T.), THRUN, S. (2005): *Anytime Dynamic A*: An Anytime, Replanning Algorithm*, Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS) MAASS, W., NATSCHLAEGER, T., MARKRAM, H. (2002): Real–time computing without stable states: A new framework for neural computation based on perturbations, In. *Neural Computation* Vol. 14 Issue 11 pp. 2531–2560.

MALI, V., RAO, M., MANTHA, S. S. (2013): AHP Driven GIS Based Emergency Routing in Disaster Management, Advances in Computing, In. *Communication, and Control Communications in Computer and Information Science* Volume 361, pp. 237–248. https://doi.org/10.1007/978-3-642-36321-4_22

SAATY, T. L. (1980): *The analytic hierarchy process*, New York: McGraw Hill

SAATY, T. L. (1988): What is the Analytic Hierarchy Process? In. *Mathematical Models for Decision Support*, NATO ASI Series Vol. 48, pp 109–121. https://doi.org/10.1007/978-3-642-83555-1_5

SAATY, T. L. (2003): Decision–making with the AHP: Why is the principal eigenvector necessary, In. *European Journal of Operational Research*, Elsevier, Vol. 145 Issue 1, pp. 85–91. https://doi.org/10.1016/S0377-2217(02)00227-8

STENTZ, A. (T.) (1994): *Optimal and Efficient Path Planning for Partially–Known Environments*, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '94), pp. 3310–3317.

VAIDYAA, O. S., KUMARB, S. (2006): Analytic hierarchy process: An overview of applications, In. *European Journal of Operational Research* Vol. 169 Issue 1, pp. 1–29. https://doi.org/10.1016/j.ejor.2004.04.028

YANDUO, Z., KUN, W. (2009): The Application of Liquid State Machines in Robot Path Planning, In. *Journal of Computers*, Vol. 4 Issue 11, p. 1182.

Appendix

Notation

s	state
g(s)	cost vector of state *S*
rhs(s)	one step lookahead cost vector of state *S*
s'	neighbour state of *S*
h(s,s$_{start}$)	heuristic cost estimate vector from state *s* to the starting state
Pred(s)	predecessor states of *S Succ(s)* successor states of
S bptr(s)	backpointer of *S*
OPEN	priority queue for states to expand *CLOSED* queue for states
removed from *OPEN INCONS*	queue for inconsistent states
ε	suboptimality bound
./	element by element division

Main ()

```
01. $g(s_{start}) = rsh(s_{start}) = \infty; g(s_{goal}) = \infty;$
02. $rhs(s_{start}) = 0; \varepsilon = \varepsilon_0;$
03. $CLOSED = INCONS = \emptyset$
04. for each criterion $OPEN_i = \emptyset$
05. $OPEN = \bigcup_{i}^{N} OPEN_i$
06. calculate weights for all criteria and store them in $W$
07. insert $s_{goal}$ into every $OPEN$ with $key_i(s_{goal})$
08. ComputeorImprovePath();
09. publish current $\varepsilon$ suboptimal solution
10. forever
11. if changes in the weights are detected
12. update weight vector $W$
13. if changes in edge costs are detected
14. for all directed edges $(u,v)$ with changed edge cost
15. update edge costs according to each criteria
16. UpdateState(u);
17. if significant edge cost changes were observed
18. increase $\varepsilon$ or replan from scratch
19. else if $\varepsilon > 1$
20. decrease $\varepsilon$
21. if significant changes in the weights were observed
22. replan from scratch
23. Move states from $INCONS$ to all $OPEN$;
24. $CLOSED = \emptyset$
```

```
25.  ComputeorImprovePath();
26.  publish current $\varepsilon$ – suboptimal solution;
27.  if $\varepsilon = 1$
28.  wait for changes in edge cost and in weights
```

ComputeorImprovePath ()

```
01.  while (key( $s$ = SelectMinState($OPEN$,$W$)) < key($s_{start}$) OR $g(s_{start}) \neq rsh(s_{start})$ )
02.  remove $s$ from $OPEN$
03.  if $W^T(g(s)./ rsh(s)) > W^T(rhs(s)./ g(s))$
04.  $g(s) = rhs(s)$;
05.  $CLOSED = CLOSED \cup \{s\}$
06.  for all $s' \in Pred(s)$ UpdateState($s'$);
07.  else
08.  $g(s) = \infty$ ;
09.  for all $s' \in Pred(s) \cup \{s\}$ UpdateState($s'$);
```

UpdateState (s)

```
01.  if $s$ was not visited before
02.  $g(s) = \infty$ ;
03.  if ($s \neq s_{goal}$)
04.  $s'$ = SelectMinState($Succ(s)$,$W$) ;
05.  $rhs(s) = g(s') + c(s,s')$ ;
06.  $bptr(s') = s$;
07.  if ( $s \in OPEN$ ) remove $s$ from $OPEN$
08.  if ($g(s) \neq rhs(s)$)
09.  if $s \notin CLOSED$
10.  insert $s$ into $OPEN$ with key( $s$ );
11.  else
12.  insert $s$ into $INCONS$ ;
```

key (s)

```
01.  if $W^T(g(s)./ rsh(s)) > W^T(rhs(s)./ g(s))$
02.  return $[rhs(s) + \varepsilon \cdot h(s_{start},s); rhs(s)]$;
03.  else
04.  return $[g(s) + h(s_{start},s); g(s)]$;
```

SelectMinState (*OPEN*, *W*)

```
01.  for all $OPEN_i \in OPEN$
02.  select $s_i$ with minimum key on $OPEN$ for all criteria
03.  establish judgement matrices with $s_i$ as alternatives with according cost
04.  calculate priority vectors
05.  establish decision matrix
06.  select best alternative according to $W$
07.  return best alternative
```

```
**SelectMinState** ($Succ(s)$, $W$)
01.    for all criteria
02.    establish judgement matrices with $s' \in Succ(s)$ as alternatives
03.    calculate priority vectors
04.    establish decision matrix
05.    select best alternative according to $W$
06.    return best alternative
```