Hierarchical Tabu Programming for Finding the Underwater Vehicle Trajectory

Jerzy Balicki †,

Naval University of Gdynia, ul. Smidowicza 69, Gdynia, Poland

Summary
Trajectory of an autonomous underwater vehicle can be obtained by using *tabu programming* that is a new paradigm of artificial intelligence. Three crucial criteria can be used for an assessment of an underwater vehicle trajectory quality: a total length, a smoothness, and a measure of safety. Finally, results of some numerical experiments have been presented.

Key words:
Tabu programming, remote operating vehicle, efficient solutions, multi-criterion optimization.

1. Introduction

Path of an underwater vehicle can be determined by tabu programming that is capable to solve some multiobjective optimization problems [3]. Tabu programming is a new paradigm of artificial intelligence that can be applied for computer decision aid. Similarly to the genetic programming, tabu programming solves different problems by using a general solver that is based on a tabu algorithm.

For evaluation of a vehicle trajectory, three main criteria can be used: a total length of a path, a measure of safety, and a smoothness of a trajectory [2]. An algorithm implemented in the computer board of an underwater vehicle is supposed to find a path between two specified locations in a three dimensional space, which is collision-free and satisfies optimization criteria [20].

Remotely operated vehicles are usually designed for underwater observation in hostile environment [21]. The foremost attributes of these vehicles are their power capability and their compactness. The vehicle consists of two divisions. The upper part ensures the vehicle positive buoyancy and houses the sonar head. The lower part consists of a watertight frame made of welded pressure-resistant tubular stainless steel. The underwater vehicle is equipped with four three-phase asynchronous thruster motors with propellers. There is a surface control unit with its power cable. It is possible to extend unit's capabilities by using the board computer to find the trajectory of the vehicle.

Manuscript received November 16, 2007
Manuscript revised November 25, 2007

Tabu search can be treated as a general combinatorial optimization technique for using in zero-one programming, non-convex non-linear programming, and general mixed integer optimization [9]. This technique is basically used to continuous functions by selection a discrete encoding of the problem [17]. In a tabu search, special areas are forbidden during the seeking in a space of all possible combinations [4].

Tabu programming paradigm is implemented as a tabu search algorithm operated on the computer program that produces the current solution. A solution is generated as the program function and then tabu search procedures are applied for finding Pareto-suboptimal solutions. Tabu programming is a relatively new paradigm of an artificial intelligence [3].

In this paper, tabu programming for solving multiobjective optimization problems of finding trajectory of underwater vehicle has been considered. Tabu search algorithm has been extended by using a computer program instead of a mathematical variable. Finally, results of some numerical experiments have been presented.

1. Vehicle trajectory evaluation

Trajectory between point (x_1,y_1,z_1) and (x_M,y_M,z_M), where M is the number of turn points, can be represented as the x vector, as below [2]:

$$x=(M,x_1,y_1,z_1,\ldots,x_m,y_m,z_m,\ldots,x_M,y_M,z_M) \quad (1)$$

The point (x_m,y_m,z_m) for $m\in\{1,\ldots M\}$ is feasible, if both the segment from (x_{m1},y_{m-1},z_{m-1}) to (x_m,y_m,z_m) and the segment from (x_m,y_m,z_m) to $(x_{m+1},y_{m+1},z_{m+1})$ do not cut forbidden areas for the vehicle. Some basic constraints for coordinates of trajectory points in the given water areas can be formulated, as follows:

$$\begin{aligned} & M_{\min} \le M \le M_{\max}, \\ & X^{\min} \le x_m \le X^{\max}, \quad m = \overline{1, M}, \\ & Y^{\min} \le y_m \le Y^{\max}, \quad m = \overline{1, M}, \\ & 0 \le z_m \le Z_m^{\max}, \qquad m = \overline{1, M}, \end{aligned} \tag{2}$$

where

$X^{\min}, X^{\max}$ – area constraints for the coordinate x_m,

$Y^{\min}, Y^{\max}$ – area constraints for the coordinate y_m,

$Z_m^{\max}$ – the maximal depth of water at (x_m, y_m).

The length of the trajectory x can be presented, as bellow:

$$L(x) = \sum_{m=1}^{M-1} d(p_m, p_{m+1}), \tag{3}$$

where $d(p_m,p_{m+1})$ is a distance between two neighboring points $p_m=(x_m,y_m,z_m)$ and $p_{m+1}=(x_{m+1},y_{m+1},z_{m+1})$.

The distance between two adjacent path points can be calculated, as below:

$$d(p_m,p_{m+1})=\sqrt{(x_m-x_{m+1})^2+(y_m-y_{m+1})^2+(z_m-z_{m+1})^2}. \tag{4}$$

2. Optimization problem

A safety measure, where regions of non-moving obstacles $\Omega_1,...,\Omega_k,...,\Omega_K$ in the water are known, can be defined, as follows [2]:

$$P(x) = \max_{m=\overline{1,M-1}} b(p_m, p_{m+1}), \tag{5}$$

where $b(p_m,p_{m+1})$ - the penalty for the segment (p_m, p_{m+1}), if the segment cuts the forbidden area.

The penalty value $b(p_m , p_{m+1})$ is calculated, as follows:

$$b(p_m,p_{m+1})=\begin{cases} d_{\min}-r(p_m,p_{m+1}), & \text{if } r(p_m,p_{m+1}) \ge d_{\min}, \\ e^{\beta(d_{\min}-r_m(p_m,p_{m+1}))}-1, & \text{otherwise}, \end{cases} \tag{6}$$

where

$r(p_m, p_{m.+1})$ – the sampled smallest distance from the line segment $(p_m, p_{m.+1})$ to the nearest forbidden area,

$d_{\min}$ – a minimal safe distance from the vehicle to another object,

β – a positive penalty coefficient.

If $r(p_m, p_{m.+1})$ is non-smaller than $d_{\min}$, then $b(p_m, p_{m.+1})$ is negative. When $r(p_m, p_{m.+1})$ is smaller than $d_{\min}$, then $b(p_m, p_{m.+1})$ is positive and it grows exponentially due to the subtraction. The function P is a maximum of penalties for all segments it is supposed to be minimized to obtain a trajectory as safe as possible.

The criterion S is supposed to maintain a smooth trajectory to avoid sudden turns of direction, as follows [2]:

$$S(x) = \max_{m=\overline{2,M-1}} \frac{\alpha_m}{\min\{d(p_{m-1}, p_m), d(p_m, p_{m+1})\}}, \tag{7}$$

where

$\alpha_{m.}$ is the angle between the extension of the line segment (p_{m-1}, p_m) and the line segment $(p_m, p_{m.+1})$ on a plane determined by above segments.

For the same distances the trajectory is smoother, if the maximum angle for it is smaller. We assume, that $\alpha_m \in [0,\pi]$. If the minimum length from $d(p_{m-1}, p_m)$ and $d(p_m, p_{m.+1})$ is longer, then there are less points p_m, where the direction of trajectory is changed.

The smoothness of trajectory can be related to the sum of all trajectory curvatures at points, as bellow:

$$\hat{S}(x) = \sum_{m=2}^{M-1} \frac{\alpha_m}{\min\{d(p_{m-1}, p_m), d(p_m, p_{m+1})\}}. \tag{8}$$

What is more, the minimization of root for sum-squared function can be considered, as below:

$$\tilde{S}(x) = \sqrt{\sum_{m=2}^{M-1} \frac{\alpha_m}{\min\{d(p_{m-1}, p_m), d(p_m, p_{m+1})\}}}. \tag{9}$$

We order partial criteria from the most important criterion to the least important criterion. In such a way, a hierarchical solution can be found. In a multicriteria navigation of the underwater vehicle, the safety criterion seems to be the most important. Let the multicriteria optimization problem be considered for finding trajectory of the underwater vehicle as the Pareto-optimal solution:

$$(X, F, R), \tag{10}$$

where

X – the set of admissible trajectories,

F – the vector criterion,

R – the relation for finding *hierarchical* solutions [1].

Because of the variable number of points in trajectory, a set of all trajectories (admissible or non-admissible) consists of vectors with no more than $3M_{\max}$ coordinates. It

can be denotes as $X = 2^T$, where $T = \mathsf{R}^{3M_{\max}}$ and R is a set of real numbers.

The set X of feasible trajectories is defined, as follows:

$$\begin{aligned} X = \{ & x \in \mathsf{X} \mid x = (x_1, y_1, z_1, \ldots, x_m, y_m, z_m, \ldots, x_M, y_M, z_M), \\ & M_{\min} \le M \le M_{\max}, \\ & X^{\min} \le x_m \le X^{\max}, \ m = \overline{1, M}, \\ & Y^{\min} \le y_m \le Y^{\max}, \ m = \overline{1, M}, \\ & 0 \le z_m \le Z_m^{\max}, \qquad m = \overline{1, M} \} \end{aligned} \tag{11}$$

Obstacles are respected by increasing the safety criterion of the trajectory. The vector criterion $F: X \to \mathsf{R}^3$ has three partial criteria, as follows:

$$F(x) = [L(x), P(x), S(x)], \ x \in \mathsf{X} \ , \tag{12}$$

where $L(x), P(x), S(x)$ are calculated according to (2), (5) and (7).

3. Tabu programming

Tabu programming is based on tabu search rules. The tabu search starts by moving to a local minimum. The tabu approach avoids entrainment in cycles by forbidding moves which lead to points in the solution space previously visited [12]. To avoid a path already investigated a point with poor quality can be accepted [10]. This insures new regions of a solution space will be explored in with the goal of avoiding local minima and finding the global minimum.

To avoid repeating the steps recent moves are recorded in some tabu lists [18]. That lists are form the tabu search memory. The memory can vary as the search proceeds [5]. At the beginning, the target is testing the solution space, during a 'diversification' [11]. As candidate locations are identified the algorithm is more focused to find local optimal solutions in an 'intensification' process. The tabu method operates with the size, variability, and adaptability of the tabu memory to a solved problem [7].

Special areas are forbidden during the seeking in a space of all possible combinations. From that neighborhood $N(x^{\text{now}})$ of the current solution, we can choose the next solution x^{next} to a search trajectory [14]. The accepted alternative is supposed to have the best value of an objective function among the current neighborhood. In the tabu search algorithm based on the short-term memory, a basic neighborhood of a current solution may be reduced to a considered neighborhood $\boldsymbol{K}(x^{\text{now}})$ because of the maintaining a selective history of the states encountered during the exploration [19]. Some solutions, which were visited during the given last term, are excluded from the basic neighborhood according to the tabu classification of movements [6]. If any solutions performs aspiration criterion, then it can be included to the considered neighborhood, only [16].

Tabu programming is the tabu search algorithm that operates on the dedicated population of computer program [3]. Computer programs are constructed from the basic program that produces the current solution. The basic program is modeled as a tree (Fig. 1).

That tree is equivalent to the parse tree that most compilers construct internally to represent the given computer program. A tree can be changed to create the neighborhood $N(x^{\text{now}})$ of the current program. We can remove a sub-tree with the randomly chosen node from the parent tree. Next, the randomly selected node as a terminal is required to be inserted. A functional node is an elementary procedure randomly selected from the primary defined set of functions [15]:

$$\mathsf{F} = \{ f_1, \ldots, f_n, \ldots, f_N \} \tag{13}$$

In the studied problem, we define set of functions, as bellow:

$$\mathsf{F} = \{ IF\_OBSTACLE, MOVE, IF\_END, +, -, *, / \} \tag{14}$$

The procedure *IF_OBSTACLE* takes two arguments. If the obstacle is recognized ahead the underwater vehicle, the first argument is performed. In the other case, the second argument is executed.

The function *MOVE* requires three arguments. It causes the movement along the given direction with the velocity equals the first argument during assumed time Δt. The time Δt is the value that is equal to the division a limited time by $M_{\max}$. The direction of the movement is changed according to the second and third arguments. The second argument is the angle of changing this direction up if it is positive or down if it is negative. Similarly, the third argument represents an angle of changing the direction to the left if it is positive or – to the right if it is negative.

The procedure *IF_END* ends the journey of the underwater vehicle if it is in the destination region or the expedition is continued if it is not there.

Furthermore, each procedure is supposed to be capable to allow any value and data type that may possible be assumed by any terminal selected from the following terminal set:

$$\mathsf{T} = \{a_1,...,a_m,...,a_M\} \tag{15}$$

For finding the trajectory of the underwater vehicle, the set of arguments consists of the real numbers generated from the interval (-1; 1).

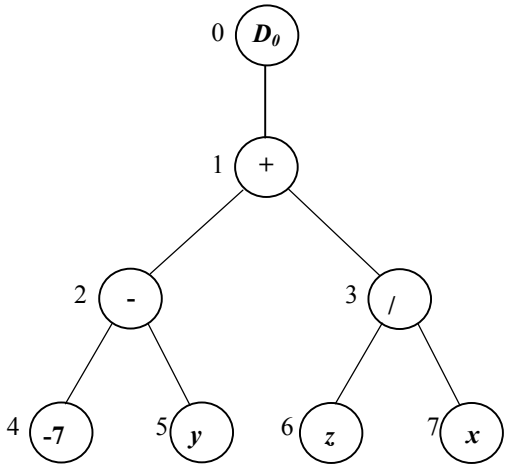

Fig. 1. The program tree that can be modified by tabu algorithm

Another sort of movements is related to removing the randomly chosen terminal node and then adding a sub-tree with the functional node as a root. That sub-tree can be constructed from the random number of nodes.

If the node is the root of the reducing sub-tree for the current program, it can be protected against choosing it to be that root in a reducing operation until the next λ_1 movements is performed. However, that node can be selected to be the root for adding the sub-tree. Similarly, if the node is the root of the adding tree, it can be protected against choosing him to be that root in a adding operation until the next λ_2 movements is performed.

We can implement that by introducing the assignment vector of the node names to the node numbers. We consider a dummy node D_0 (Fig. 1) as the number 0, for the formal reason. The node index $l = \overline{1, L_{\max}}$, where $L_{\max}$ represents the assumed maximal number of nodes in the tree. Numbers are assigned from the dummy node to lower layers and from the left to the right at the current layer. The assignment vector of the node names to the node numbers for the tree from the Figure 1 can be represented, as below:

$$\omega = (D_0,+,-,/,-7,y,z,x) \tag{14}$$

Moreover, the vector of function f and argument assignment can be defined, as follows:

$$\psi = (f,f,f,f,a,a,a,a) \tag{15}$$

The vector of the argument number can be determined, as below:

$$\chi = (1,2,2,2,0,0,0,0) \tag{16}$$

Now, we can introduce the matrix of reducing node memory $M^- = [m_{nm}]_{L_{\max} \times L_{\max}}$, where m_{nm} represents the number of steps that can be missed after reduction the function f_m (with the parent f_n) as a root of the chosen sub-tree. After exchanging that root, $m_{nm} = \lambda_1$.

Similarly, we can define the matrix of adding node memory $M^+ = [\tilde{m}_{nm}]_{L_{\max} \times L_{\max}}$, where $\tilde{m}_{nm}$ represents the number of steps that can be missed after adding the function f_m (with the parent f_n) as a root of the created sub-tree. After exchanging that root, $\tilde{m}_{nm} = \lambda_2$.

Parameters λ_1 and λ_2 are usually equal to λ, but we can adjust their values to tune the tabu programming for the solved problem. On the other hand, the length of the short-term memory λ is supposed to be no greater than $L_{\max}$. After λ movements, the selected node may be chosen for operation once again.

Tabu programming rules can be implemented as an algorithm ATP (Fig. 2) that can be used for optimisation. ATP can be used for solving an optimization problem with one criterion, as follows:

$$F_{\min} = \min_{x \in X} F(x) \tag{17}$$

where

F – criterion of the problem,

X – set of admissible trajectories of vehicle.

The selection function W is constructed from the criterion F and functions describing constraints [8]. Usually, the penalty function can be applied [13].

If some admissible solutions are in the neighborhood of the current solutions produced by the modified programs, then the *hierarchical* solutions are determined.

A tabu algorithm has been written in the Matlab language. Our initial numerical experiments confirm that feasible, sub-optimal in Pareto sense, trajectories can be found by tabu programming. A paradigm of tabu programming gives opportunity to solve the several problems.

1. Initial procedure k**:=0**

(A) Generation of the program that produces x^{now}

(B) $x^{best} := x^{now}$, $x^{bis} := x^{now}$

(C) $F_{min} := F(x^{now})$

(D) Initialization of restriction matrixes M^+, M^-

(E) Setting λ_1, λ_2

2. Solution selection and stop criterion $k := k+1$

(A) Finding a set of tree candidates $K(M^+, M^-, x^{now})$ from the neighborhood $N(x^{now})$

(B) Selection of the next solution $x^{next} \in K$ (M^+, M^-, x^{now}) with the minimal value of the selection function W among solutions taken from K

(C) **Aspiration condition.** If all solutions from the neighbourhood are tabu-active and $F_{min} \geq F(x^{now})$, then $x^{best} := x^{now}$, $F_{min} := F(x^{now})$

(D) **Re-linking of search trajectory.** If x^{next} was not changed during main iteration, then crossover procedure for parents x^{best}, x^{bis} is performed. A child with the smaller value of F is x^{next}, and another one is x^{bis}

(E) If $k = 0.4\, K_{max}$, then $\lambda_1 := 4\lambda_1$, $\lambda_2 := 4\lambda_2$

(F) If $k = K_{max}$ or maximal time of calculation is exceeded, then STOP.

3. Up-dating

(A) $x^{now} := x^{next}$

(B) If $F(x^{now}) < F_{min}$, then $x^{bis} := x^{best}$ and go to 1(B)

(C) After reduction the procedure f_m (with the parent f_n) as a root of the chosen sub-tree $M^- := M^- - 1$, $m_{nm} = \lambda_1$.

(D) After adding the procedure f_m (with the parent f_n) as a root of the created sub-tree $M^+ := M^+ - 1$, $\tilde{m}_{nm} = \lambda_2$.

(E) go to 2

Fig. 2. An algorithm ATP of tabu search programming

If the trajectory x is admissible, then the selection function value is estimated, as below:

$$f(x) = r_{\max} - r(x) + P_{\max} + 1, \qquad (18)$$

where $r(x)$ denotes the rank of an admissible solution in the neighbourhood of the current solution,

Let the Pareto points $\{P_1, P_2, ..., P_U\}$ be given for any instance of the optimal trajectory problem. If the AMT finds the efficient point (A_{u1}, P_{u2}) for the smoothness P_{u2}, this point is associated to the uth Pareto result (P_{u1}, P_{u2}) with the same value of the smoothness.

The distance between points (A_{u1}, P_{u2}) and (P_{u1}, P_{u2}) is calculated according to an expression $|P_{u1} - A_{u1}|$. If the point (A_{u1}, P_{u2}) is not discovered by the algorithm, we assume the distance is $|P_{u1} - A_{u1}^{\min}|$, where $A_{u1}^{\min}$ is the minimal length of the trajectory for the instance of problem.

The level of convergence to the Pareto front is calculated, as follows:

$$S = \sum_{u=1}^{U} |P_{u1} - A_{u1}|. \qquad (19)$$

An average level $\overline{S}$ is calculated for several runs of the ATP. That tabu programming ATM gives better outcomes than the genetic programming AMEA/GP (Fig. 3). After 300 selections, an average level of Pareto set obtaining is 1.2% for the ATM, 3.6% for the AMEA/GP. 40 initial trajectories were prepared, and each algorithm starts 40 times from these points.

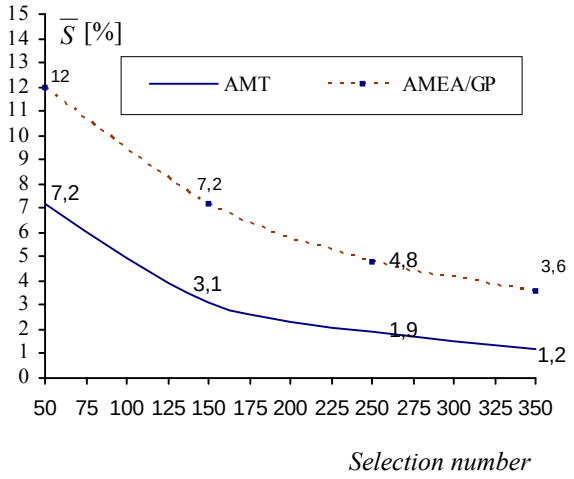

Fig. 3. Convergence of results for the AMT and the AMEA/GP

An average level of convergence to the Pareto set, an maximal level, and the average number of optimal solutions become worse, when the number of internal points of trajectory, size of the water space, and number of obstacles increase. An average level is 23.7% for the AMT versus 37.9% for the AMEA/GP, if the instance includes 200 internal points, and also 12 obstacles.

4. Concluding remarks

Tabu programming is a new paradigm of artificial intelligence that can be used for finding solution to several problems. Tabu programming can be applied for control an underwater vehicle. A computer program as a tree is a subject of tabu operators such as selection from neighborhood, short-term memory and re-linking of the search trajectory. A tabu programming has been applied for operating on the computer procedures written in the Matlab language.

Our initial numerical experiments confirm that feasible, sub-optimal in Pareto sense, task assignments can be found by tabu programming. A paradigm of tabu programming gives opportunity to solve this problem for changeable environment.

Our future works will focus on testing the other sets of procedures and terminals to find the Pareto-optimal solutions for distinguish criteria and constraints. Moreover, we will concern on a development the combination between tabu search and evolutionary algorithms for finding efficient solutions.

References

[1] A. Ameljańczyk, *Multicriteria Optimization*, WAT, Warsaw 1986.

[2] J. Balicki: *Finding Trajectory of Underwater Vehicle by Multi-Criterion Genetic Programming*, Proceedings of the 9th IEEE International Conference on *Methods and Models in Automation and Robotics*, (Ed. R. Kaszyński), Vol. 1, Międzyzdroje, Poland, 25-28 August 2005, pp. 237-242.

[3] J. Balicki, *Tabu Programming for Multiobjective Optimization Problems,* International Journal of Computer Science and Network Security, VOL.7 No.10, October 2007, pp. 863-870.

[4] R. Battiti, *Reactive search: Toward self-tuning heuristics*, in V. J. Rayward-Smith, editor, *Modern Heuristic Search Methods*, John Wiley and Sons Ltd, 1996, pp. 61-83.

[5] R. Battiti, G. Tecchiolli, *Simulated annealing and tabu search in the long run: a comparison on qap tasks*, Computer Math. Applic., Vol. 28, No. 6, 1994, pp. 1-8.

[6] T. G. Crainic, M. Toulouse and M. Gendreau, *Toward a Taxonomy of Parallel Tabu Search Heuristics*, INFORMS Journal on Computing, Vol. 9, No. 1, 1997, pp. 61-72.

[7] M. Dell'Amico, M. Trubian, *Applying Tabu Search to the Job-Shop Scheduling Problem*, Annals of Operations Research, Vol. 41, 1993, pp. 231-252.

[8] U. Faigle, W. Kern, *Some Convergence Results for Probabilistic Tabu Search*, *ORSA Journal on Computing,* Vol. 4, No. 1, 1992, pp. 32-38.

[9] F. Glover, *Tabu Search — Part I*, *ORSA Journal on Computing*, Vol. 1, No. 3, 1989, pp. 190-206.

[10] F. Glover, *Tabu Search — Part II*, ORSA Journal on Computing, Vol. 2, No. 1, 1990, pp. 4-32.

[11] F. Glover, *Tabu Search: A Tutorial*, *Interfaces*, Vol. 20, No. 4, 1990, pp. 74-94.

[12] F. Glover, M. Laguna, *Tabu Search*, Kluwer Academic Publishers, Boston 1997

[13] M. P. Hansen, *Tabu Search for Multicriteria Optimisation: MOTS*. Proceedings of the Multi Criteria Decision Making, Cape Town, South Africa, 1997

[14] A. Hertz, *Finding a Feasible Course Schedule Using Tabu Search*, Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science, Vol. 35, 1992.

[15] J.R. Koza, *Genetic programming.* The MIT Press, Cambridge 1992.

[16] A. Lokketangen, A. K. Jornsten and S. Storoy, *Tabu Search within a Pivot and Complement Framework*, International Transactions in Operations Research, Vol. 1, No. 3, 1994, pp. 305-316.

[17] C. Rego, *A Subpath Ejection Method for the Vehicle Routing Problem*, Management Science, Vol. 44, No. 10, 1998, pp. 1447-1459.

[18] J. Węglarz, *Recent Advances in Project Scheduling.* Kluwer Academic Publishers, Dordrecht 1998.

[19] A. M. Widmer, *The Job-shop Scheduling with Tooling Constraints: A Tabu Search Approach*, J. Opt. Res. S, Vol. 42, 1991, pp. 75-82

[20] J. Xiao, Z. Michalewicz, L. Zhang, K. Trojanowski: *Adaptive Evolutionary Planner/Navigator for Mobile Robots.* IEEE Transactions on Evolutionary Computation, Vol. 1, No. 1, April 1997, pp. 18-28.

[21] C.K. Yap, *Algorithm motion planning.* In the Advances in Robotics, vol. I: Algorithmic and Geometric Aspects of Robotics, J.T. Schwartz and C.K. Yap, Eds., Hillsdale, NJ: Lawrence Erlbaum, 1987, pp. 95-14.

Jerzy Balicki received the M.S. and Ph.D. degrees in Computer Science from Warsaw University of Technology in 1982 and 1987, respectively. During 1982-1997, he stayed in Computer Center of Maritime High School of Gdynia to study management systems, mobile systems, and decision support systems. Then, he achieved habilitation from Technical University of Poznan in 2001. He was admitted as a university professor at Naval University of Gdynia in 2002.