A Route Selection Problem Applied to Auto-Piloted Aircraft Tugs

A Route Selection Problem Applied to Auto-Piloted Aircraft Tugs

Giuseppe Sirigu
Politecnico di Torino
Department of Mechanical
and Aerospace Engineering
Corso Duca degli Abruzzi 24
10129, Torino, ITALY
giuseppe.sirigu@polito.it

Mario Cassaro
ONERA
Dpartement traitement de
l'information et systmes
Toulouse, France
Mario.Cassaro@onera.fr

Manuela Battipede, Piero Gili
Politecnico di Torino
Department of Mechanical
and Aerospace Engineering
C.so Duca degli Abruzzi 24
10129, Torino, ITALY
manuela.battipede@polito.it
piero.gili@polito.it

Abstract: The antithetical needs of increasing the air traffic, reducing the air pollutant and noise emissions, jointly with the prominent problem of airport congestion spur to radically innovate the entire ground operation system and airport management. In this scenario, an alternative autonomous system for engine-off taxiing (dispatch towing) attracts the interest of the civil aviation world. Even though structural and regulatory limitations undermine the employment of the already existing push-back tractors to this purpose, they remain the main candidates to accomplish the mission. New technologies are already under investigation to optimize towbarless tractor joints, so as to respond to the structure safety requirements. However, regulation limitations will soon be an issue. In this paper, a software solution for a route selection problem in a discretized airport environment is presented, in the believe that a full-authority control system, including tractors' selection logic, path planning and mission event sequencing algorithms will possibly meet the regulation requirements. Four different algorithms are implemented and compared: two Hopfield-type neural networks and two algorithms based on graph theory. They compute the shortest path, accounting for restricted airport areas, preferential directions and dynamic obstacles. The computed route checkpoints serve as a reference for the tractor autopilot. Two different missions are analyzed, concerning the towing of departing and arriving aircraft respectively. A single mission consists of three different events, called phases: Phase 1 goes from the actual tractor position (eventually the parking zone) to the selected aircraft (parked or just landed); Phase 2 is the actual engine-off taxi towing; Phase 3 is the tractor return to its own parking zone. Both missions are simulated and results are reported and compared.

Key–Words: Route selection, Airport Taxiing, Neural Network, Hopfield, Dijkstra, A*

1 Introduction

The current vivid interest of the civil aviation in the development and implementation of innovative airport management and ground operation systems is prompted by the antithetical needs of increasing the air traffic, reducing the air pollutant and noise emissions. In particular, airport congestion is globally recognized as one of the most prominent problem areas of the next future and the straightforward solution of expanding the airfields is not that efficient and not always doable. In fact, the higher complexity of air terminals deriving from the addition of runways and taxiways will penalize the overall system efficiency by increasing the human workload and error risk, resulting in potentially hazardous situations. For example, Hilburn in [4] and the ICAO regulations [10] demonstrate the high level of risk derived by the Head Down Operations (HDOs) during taxi phases. In addition, the increasing aircraft number, jointly with the longer taxiing, will significantly contribute to an increase in fuel burn and emissions, which is in contrast with the stringent environmental regulations. Therefore, the concept of autonomous engines-off taxiing, using towing vehicles, has been recently investigated and recognized as a feasible and effective solution to the aforementioned issues. It consists in employing push-back tractors to tow the aircraft from gate to runway for departure and from runway to gate for arrival, while keeping engines off during ground movements. To make the new concept operative, some structural and regulatory issues need to be solved. Structurally, nose landing gears (NLGs) are not designed to be towed for long distances; this feature can lead to fatigue failures due to transversal loads. Towbarless systems decrease NLG loads, while ensuring an effective connection between the aircraft and the tractor. Thus, they can be used to tow the aircraft along the entire taxiing path. Indeed, aeronautical industries and academics are currently investigating different solutions,

ranging from a semi-robotic towbarless tractor [15] to a completely autonomous system [14]. In the Taxibot solution, developed by an industrial consortium, the thrust required for taxing is provided by a diesel engine tug, while the direction control is managed by the aircraft pilot through the aircraft steering system [15]. This control setup allows to avoid regulations' limitations since the pilot stays in charge of the ground maneuvering. Even if the explained solution seems to be of ease implementation in short time and resolve the air-noise pollution and fuel consumption issues, it has a major drawback that is the increased hazardous conditions deriving from the higher complexity level of operations, to be performed by pilots and ground operators. On the other hand, an autonomous external taxiing system, with a manned tugs that tows and drives utterly the aircraft through the taxiways, will reduce the pilot workload and will optimize the airport ground movements. In this configuration, all the pre-flight procedures (*e.g.* aircraft system functionality checks, Flight Management System setup and engine warm up) might be performed by the pilot during the semi-autonomous taxi, without loss of safety [14]. The drawback consists of a regulatory limitation, especially in terms of responsibility allocation, deriving from the lack of Pilot in Control (PIC) situation.

To this purpose, another concept of airport ground operation system has been presented by our research team [5]. It consists in a semi-automatic system, activated by the control tower, in which autonomous autopiloted tractors are capable of accomplishing towing missions between points, selected by the tower operators. Albeit a full-authority control system for tug selection and route conflict avoidance will be required for the final application, at the project status, the route selection problem is the major objective. The route selection problem is thoroughly addressed in literature and it has been solve by several techniques such as genetic algorithms [6], heuristic methods [24] [3] [25] and tabu search algorithms [16]. Neural networks have been also used for vehicle and computer network routing. In particular, Hopfield-type Neural Networks (HNNs) [11] [12] [13] have been extensively used in the past to these purposes, as they minimize an energy function, which can be properly defined to perform a shortest path search [18] [20] or can be used in a non-static environment [27]. Lately, among all the other solutions, the graph theory and the Dijkstra's algorithm became established methodologies, because of their fastness and robustness, to solve routing selection problems in any kind of environment [8] [17] [2]. In addition, a modified version of the Dijkstra algorithm, which uses heuristic functions to define a preferential search direction, called $A*$ [22] [26] [7], allows for further computational time reduction with respect to the standard Dijkstra's algorithm, particularly noticeable for large dimensions of the searching domain.

In this paper, a software solution for the push-back tractor route selection problem in a discretized airport environment (apron) is presented. Four different algorithms are implemented: two Hopfield-type neural networks and two algorithms based on the graph theory. Their objective is the shortest path, accounting for restricted airport areas, preferential directions and dynamic obstacles. The computed route checkpoints serve as reference for the tractor autopilot. Two different missions are analyzed, concerning the towing of departing and arriving aircraft respectively. A single mission consists of three different events, called phases: Phase 1 goes from the actual tractor position (eventually the parking zone) to the selected aircraft (parked or just landed); Phase 2 is the actual engine-off taxi towing; Phase 3 is the tractor return to its own parking zone. The paper is organized as follows: Section 2 contains the problem formulation referred to our test case airport; in Section 3 the mathematical formulation of the proposed algorithms is presented and their application justified. Section 4 describes the algorithms' implementation and the results obtained for the selected test case. Pertinent conclusions are reported in the closing section.

2 Problem Formulation

A single mission of the tug is discretized in three different phases which have different initial and final checkpoints, whether an aircraft departure or arrival mission is being performed.

- Phase 1: reaching the targeted aircraft (A/C parking slot for departure mission, runways for arrival mission);
- Phase 2: perform engine-off taxi towing (apron to runways for departure mission, vice versa for arrival mission);
- Phase 3: returning to the tug parking area (runways to parking for departure mission, apron to parking for arrival mission).

As already mentioned, the problem to be solved for each phase is a shortest path route selection problem in a non-static environment with constraints. The considered constraints are the apron ground vehicles permitted ways, the one-way only routes and the different obstacles that can obstruct the way. Therefore, the trivial straight-line trajectory between initial and ending checkpoints of each phase is not always a solution of the problem. Fig. 1, which represents our test case,

shows the actual taxiways and ground vehicle permitted areas in the *Sandro Pertini* airport of Turin.

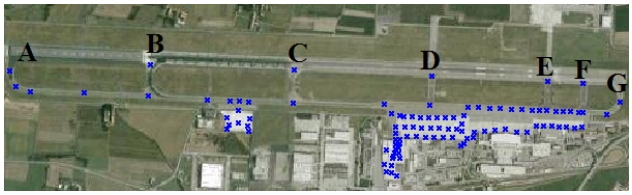

Figure 1: Turin Airport "Sandro Pertini", satellite picture with directions for taxi and ground movement. Credits: Pictures 2015 DigitalGlobe, Map data ©2015 Google

For the proposed application, the spatial domain is given by the airport runways, taxiways, ground vehicles routes, and aircraft parking lots. Route selection algorithms require a domain discretization in checkpoints and arcs with directions and possible obstacles. For the proposed application, this operation is automatically performed by a tool coded on purpose that discretizes differently depending on the path optimization algorithm to be used. Fig. 2 represents one of the possible *Sandro Pertini* airport discretization, with very few points for sake of clarity of the figure. The blue stars represent the checkpoints in the permitted airport area, while the black link are the permitted ways between checkpoints. Once the spatial domain is discretized, the proposed algorithm should compute the shortest path, for each phase separately and a preselected tug-aircraft combination. The mission (departure or arrival), tug and aircraft definition is considered at the moment as an external input, which can be provided from a human operator or a scheduling software in the future.

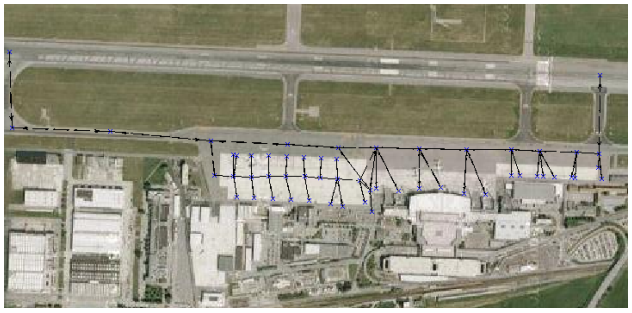

Figure 2: "Sandro Pertini" airport checkpoint discretization. Credits: Pictures 2015 DigitalGlobe, Map data ©2015 Google

3 Proposed Algorithms

The proposed algorithm consists of two main parts: the first one loads the airport geographical data (longitude and latitude) and discretizes the domain, based on the apron features; the second solves the route selection problem computing the shortest path for each mission phase. The computation on the domain is performed only once at the beginning, while the route selection algorithm is run three times per mission (arrival or departure). This section gives an overview of the mathematical formulations of the different methodologies employed and compared to solve the route selection problem under analysis.

3.1 Neural Network

Artificial neural networks are paradigms of a non-analytic way of solving problems, which are applied in different fields of science and engineering because of their versatility [9]. This system, inspired by biological nervous systems, is able to learn and make intelligent decisions without a precisely defined algorithm or a complete set of input data. Although the work of individual neurons can be quite slow, the overall neural network is very fast, which is due to the large neuronal connections and parallel processing [9], [23]. Among all types of neural networks, particular attention is given here to the recursive neural networks [18], which are characterized by a feedback signal that allows the net to use the neuron output value as an input, to obtain a faster and more robust response. Typically, this is the time delay of the signal [9]. Hopfield's neural networks are representative of this type of networks [11], [12], and its structure is shown in Fig. 3.

One of the main features of the Hopfield neural network is the possibility of a simple hardware implementation, as suggested by Fig. 4. Hopfield and Tank proposed their neural network structure [12] capable of solving different complex problems by the minimization of an energy function that has to be properly defined. This approach was demonstrated on the well-known and computationally very complex Traveling Salesman Problem (TSP) with 30 nodes [12]. Since then, many researchers have used similar models for solving a large variety of combinatorial optimization problems.

3.1.1 Hopfield Neural Network Implementation

Each neuron is realized as an operational amplifier with an increasing sigmoid function, relating the output V_i and the input U_i of the *i-th* neuron. In this way, the network acquires the characteristics of nonlinearity. Output values are called to range from 0 to 1. The

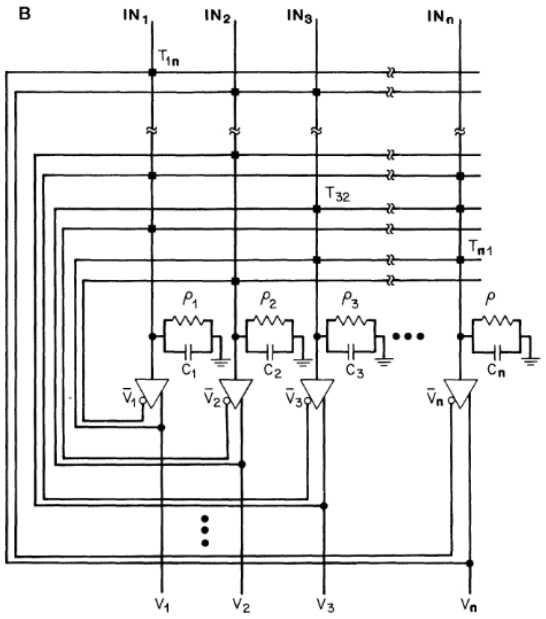

Figure 3: Hopfield neural network electric circuit model [13].

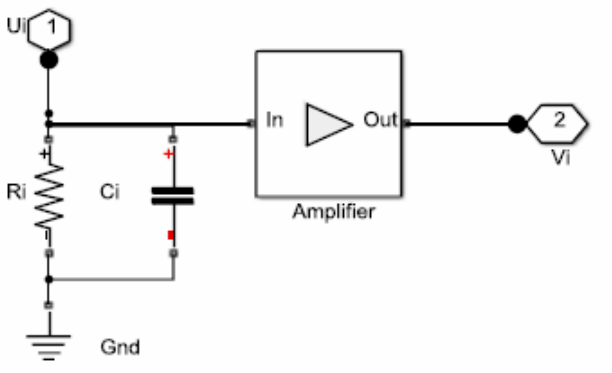

Figure 4: Electronic circuit model of Hopfield neuron.

activation function for each neuron is given as [12], [18]

$$V_{xi} = g_{xi}\left(U_{xi}\right) = \frac{1}{1 + e^{-\lambda_{xi} \cdot U_{xi}}} \tag{1}$$

where λ is a constant that determines the declination of the characteristics. In accordance with the rule of recursive networks, the output signal of the *i-th* neuron is fed back as input for the other neurons of the input layer, through resistive connections. This connectivity is defined with the synaptic weight matrix $T = [T_{ij}]$. In addition to receiving signals from the output neurons, each of the input neurons operates with the additional electrical signal *(bias current)* I_i. This is to adjust the polarization of the input neurons [12]. The input signals' dynamics is defined by equation 2,

$$\frac{dU_i}{dt} = -\frac{U_i}{\tau} + \sum_{j=1}^{N} T_{ij} V_j + I_i \tag{2}$$

where τ is the time constant. Each neuron has its own entrance U_i, the output signal V_i and the polarization signal I_i, which defines the checkpoint activation level. Output feedbacks V_i and the external inputs of each neuron pass through the resistance R, $i \neq j$ (called synapses), and iteratively provide a change of state of the network. At the end of the process, the network converges to a stable state. A capacitor C_i is connected in parallel to the resistance, and its capacity affects the neuron state as shown in the following relation [12].

$$C_i \frac{dU_i}{dt} = -\frac{U_i}{R_i} + \sum_{j=1}^{N} T_{ij} V_j + I_i \tag{3}$$

The capacitor C_i acts at the input of the nonlinear differential amplifier, whose output signals are function of V_i and $-V_i$ of these cells, according to Eq. (1) [11]. If the steepness of the sigmoid function is sufficiently large (for instance $\lambda_i > 100$), the stability of the network, in the Lyapunov sense, may be verified by observing the energy function, *E*, describing the state of the network [12]

$$E = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} T_{ij} \cdot V_j V_i - \sum_{i=1}^{N} V_i I_i \tag{4}$$

For the large reinforcement of an operational amplifier, the minimum energy, at a given N dimensional space, is allocated in 2^N distinct states, associated with an N-dimensional hypercube with sides $V \in \{0, 1\}$. Then the dynamics of the *i-th* neuron, according to equation (2), can be expressed as

$$\frac{dU_i}{dt} = -\frac{U_i}{\tau} - \frac{\partial E}{\partial V_i} \tag{5}$$

Relation (5) defines the change of the input signal and the neuron energy at every iteration. It can be demonstrated that this network architecture converges to stable states [21]. Such a network is used as a basic network structure for solving optimization problems. Starting from the original Hopfield-Tanks architecture, several modifications have been implemented for performance improvement purposes, as for example the model proposed by Ali and Kamoun [18]. In their model, the network has $n\left(n-1\right)$ neurons, where n denotes the dimension of the input square matrix, by neglecting the self-recursive signals that

are the diagonal elements (i, i) in matrix T. During the iteration process, the stable neuron states define the shortest path between the source (s) and the destination (d) points. A suitable energy function is of the form

$$\begin{aligned} E = & \frac{\mu_1}{2} \sum_X \sum_{\substack{i \neq X \\ (X,i) \neq (d,s)}} C_{Xi} V_{Xi} + \\ & + \frac{\mu_2}{2} \sum_X \sum_{\substack{i \neq X \\ (X,i) \neq (d,s)}} \rho_{Xi} V_{Xi} + \\ & + \frac{\mu_3}{2} \sum_X \left(\sum_{i \neq X} V_{Xi} - \sum_{i \neq X} V_{iX} \right)^2 + \\ & + \frac{\mu_4}{2} \sum_X \sum_{i \neq X} V_{Xi} \left(1 - V_{Xi}\right) + \\ & + \frac{\mu_5}{2} \left(1 - V_d s\right) \end{aligned} \tag{6}$$

Coefficients C_{Xi} are the link costs from neuron X to router i and the terms ρ_{Xi} describe the connection between routers: the value is 1 if the routers are not connected, and 0 for connected routers. The term μ_1 minimizes the total cost; μ_2 prevents nonexistent links from being included in the chosen path; μ_3 is zero for every router in the valid path (the number of incoming links is equal to the number of outgoing links); μ_4 forces the state of the neural network to converge to one of the stable statescorners of the hypercube defined by $V \in 0, 1$. The state V_i is close to 1 for router belonging to the valid path, otherwise the state is close to 0. The term μ_5 is zero when the output V_{ds} is equal to 1. This term is introduced to ensure that the source and the destination routers belong to the solution (the shortest path). The main contribution of the Ali and Kamoun's approach [21] consist in keeping the synaptic conductance constant (7), while the link costs and the information about the connection between nodes were associated to the bias currents I_i:

$$T_{Xi,Yi} = \mu_4 \delta_{XY} \delta_{ij} - \mu_3 \left(\delta_{XY} + \delta_{ij} - \delta_{jX} - \delta_{iY}\right) \tag{7}$$

$$I_{Xi} = \begin{cases} \frac{\mu_5}{2} - \frac{\mu_4}{2} & (X, i) = (d, s) \\ -\frac{\mu_1}{2} C_{Xi} - \frac{\mu_2}{2} \rho_{Xi} - \frac{\mu_4}{2} & otherwise \end{cases} \tag{8}$$

where $\delta_{ii} = 1$,$\delta_{ij} = 0$, for $i \neq j$. In this way, the neural network algorithm becomes very attractive for real time processing, since bias currents may be easily controlled via external circuitry, following the changes in actual traffic through the network. Several researches demonstrate that HNN is very effective for shortest or optimal path problem solution. On the other hand, the main drawbacks is the net possible instability, typically caused by the noise amplification in recursive algorithms, and the not guaranteed optimal solution convergence. However, as Hopfield demonstrated, if the obtained solution is not optimal, it will be in a group of solutions that are very "close to" the optimal.

3.1.2 Modified Hopfield Neural Network.

An Hopfield-type neural network has been proposed by Zhong et al. [27] for real-time collision-free robot path planning in a non-static environment. This approach is based on the propagation of the target activity through the local connectivity of neurons, which is formulated using harmonic functions. The main advantage in using harmonic functions lies in the local minima removal. In their application, Zhong et al. used the Laplace operator discretization applying a finite difference scheme on the grid reported in Fig. 5 [27].

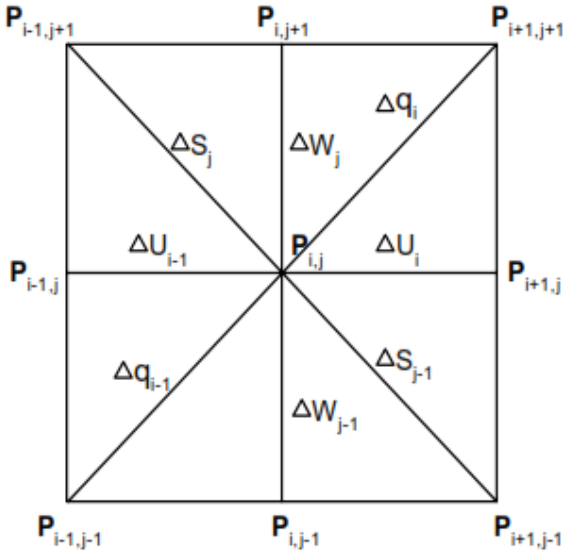

Figure 5: Local connectivity grid [27].

The dynamics of the neural network is given by Eq. (9)

$$\frac{dU_i}{dt} = -\frac{U_i}{R_i} + \sum_{e(j) \in N_r(i)} A_{ij} U_i + I_i \tag{9}$$

where $N_r\left(i\right)$ represents the neighborhood of radius r of the neuron $e\left(i\right)$ and A_{ij} is the jth element

of the local connectivity matrix of the neuron $e(i)$. The initial neurons activity can be modulated by using the bias current I_i, which assumes the value E if the *ith* neuron is the target, otherwise it is equal to 0, as expressed by Eq. (10):

$$I_i = \begin{cases} E & if\ there\ is\ a\ target \\ 0 & otherwise \end{cases} \tag{10}$$

Using the grid reported in Fig. 5, the local connectivity for each neuron assumes the shape reported in Eq. (11). The global stability of this neural network has been proved using the Lyapunov's method, even though the local connectivity is asymmetric [27].

$$A = \begin{pmatrix} A_1, & A_2, & A_3 \end{pmatrix} \tag{11}$$

where,

$$A_1 = \begin{pmatrix} \frac{2h}{\Delta s_j(\Delta s_{j-1}+\Delta s_j)} \\ \frac{2h}{\Delta u_{i-1}(\Delta u_{i-1}+\Delta u_i)} \\ \frac{2h}{\Delta q_{i-1}(\Delta q_{i-1}+\Delta q_i)} \end{pmatrix} \tag{12}$$

$$A_2 = \begin{pmatrix} \frac{2h}{\Delta w_j(\Delta w_{j-1}+\Delta w_j)} \\ 0 \\ \frac{2h}{\Delta w_{j-1}(\Delta w_{j-1}+\Delta w_j)} \end{pmatrix} \tag{13}$$

$$A_3 = \begin{pmatrix} \frac{2h}{\Delta q_i(\Delta q_{i-1}+\Delta q_i)} \\ \frac{2h}{\Delta u_i(\Delta u_{i-1}+\Delta u_i)} \\ \frac{2h}{\Delta s_{j-1}(\Delta s_{j-1}+\Delta s_j)} \end{pmatrix} \tag{14}$$

3.2 Graph Theory

Graph representation is very effective in describing real-world situations. Graphs are schemes consisting in points (nodes) and links (edges) connecting a pairs of nodes [1]. The basic graph form is composed of unweighted and undirected edges. However, some real applications could be more effectively represented by means of a weighted graph. The main idea is to associate a weight, or cost, to each edge and the total cost of a path in the graph is defined as the sum of the costs of each selected edge [19]. This approach allows for easy representation of optimization problems aiming to find a graph subset with minimum or maximum weight, as the shortest route selection problem. Furthermore, by specifying edges directions, it is possible to reproduce an asymmetrical domain. For the application under investigation, the relative distance between two points is used as edge weight and the edge direction is used to simulate one-ways only routes.

An algorithm based on graph theory was presented by Dijkstra in 1959 [8]. It is able to find the shortest route from one starting point within the spatial domain to all the vertices of the graph. A modified version of the Dijkstra's algorithm, which uses an heuristic function to define a preferential search direction, was proposed in 1968 and is called *A** [22]. These two algorithms do not require any parameter setting, leading to an easier implementation process with respect to the two neural networks implemented.

3.2.1 Dijkstra's Algorithm.

Consider a graph $G(V, E, w)$, where V is the set of nodes, E is the set of edges and $w(uv)$ is the weight of the edge from u tu v, a source node s and a target t. S is a subset of V such that $s \in S$ and denote $V \setminus S$ as $\bar{S}$. If $P = s...\bar{s}\bar{t}$ is the shortest path from s to $\bar{S}$, then, $\bar{s} \in S$ and the finite sequence $s...\bar{s}$ of P is the shortest path from s to $\bar{s}$. Thus, the distance from s to $\bar{t}$ is expressed as

$$d(s, \bar{t}) = d(s, \bar{s}) + w(\bar{s}\bar{t}) \tag{15}$$

Therefore, starting from the subset containing the only source $S_0 = \{s\}$, it is possible to construct an increasing sequence of subsets $S_0, S_1, ..., S_{n-1}$ of V such that, at a certain step i, the shortest paths from s to all the nodes in S_i are calculated. The Dijkstra's algorithm is schematically outlined as follows, reporting the different steps to obtain the shortest path from a generic source node to all the domain vertices.
Consider a domain with n points; first of all, it is necessary to define some variables:

- $l(v)$ is the length of the shortest path from the source node s to a generic node v;
- S is the set of nodes v for which the shortest path from s to v has been found (closed set);
- X is the set of nodes v for which a path has been found, but may be not the shortest path (open set);
- $E(v)$ is the set of neighbors of the generic node v.
- $parent$ is a $n \times 1$ vector, containing the label of the node that precedes v in the shortest path to v.

The Dijkstra's algorithm can be divided in four steps:

1. Set $S = \emptyset$, $X = \{s\}$, $l(s) = 0$, $l(v) = \infty$ for $v \neq s$;
2. Find $v \in X : l(v) = \min_{u \in X} l(u)$, add it to S and remove it from X;
3. $\forall u \in E(v)$

(a) If $u \in X \wedge l(v)+w(v,u) < l(u)$, set $l(u) = l(v) + w(v,u)$ and $parent(u) = v$;

(b) If $u \notin S \wedge u \notin X, \quad X = \{X, u\}$, set $l(u) = l(v) + w(v,u)$ and $parent(u) = v$;

4. If $X \neq \emptyset$ repeat from Step 2.

When the algorithm terminates, the shortest paths from s to all the other points are computed, and thus also the shortest path from the source s to the target t. To save computational time, it is possible to terminate the process when the node t is added to S. It can be proved by means of Lemma 1 in [22] that Dijkstra's algorithm always finds the optimal (shortest) path from s to v.

3.2.2 A* Algorithm

As mentioned, the *A** algorithm is an improvement of the Dijkstra's algorithm obtained by adding a heuristic function to fasten the solution [22]. The introduced evaluation function is

$$f(v) = l(v) + h(v) \tag{16}$$

where, $h(v)$ is the distance from v to t. Thus, $f(v)$ represents the distance from s to t, passing through v. $h(v)$ can be estimated by means of an heuristic function $\hat{h}(v)$, for example the Euclidean distance. However, the optimal heuristic function is problem dependent and so forth cannot be defined uniquely. Conversely to the Dijkstra's algorithm, thus, the subsequent check-point of the path is chosen using the following relation:

$$\hat{f}(v) = \min_{u \in X} \hat{f}(u) \tag{17}$$

Hart et al. [22] prove that the *A** algorithm finds the optimal solution as long as the distance from v to t is underestimated, which means that the relation $\hat{h}(v) \leq h(v)$ must be satisfied. The *A** algorithm steps are the same of the Dijkstra's one, where the function $l(v)$ is substituted by $f(v)$.

4 Implementation and Results

The proposed algorithms have been implemented for test and comparison purposes in Matlab® environment without using any existing toolbox. The code, which is optimized for airport environment, allows for an automatic discretization of the apron area of certain airports belonging to an available database.

4.1 Airport data loading and domain discretization

The first operation required to apply the several algorithms available is to define the spatial domain in terms of geographical coordinate system and properly discretize it. An airport official website can provide those information, as aircraft parking/docking, taxiways and runways locations through the aeronautical information publication (AIP). It must be pointed out that the spatial domain discretization differs whether neural network or graph theory based algorithms are being used. When the HNN based algorithm is employed, the whole set of available checkpoint coordinates are normalized to 1, indexed and stored in a 2D matrix. The maximum dimension of the airport domain and the geographical distances between checkpoints are stored in a second matrix for the actual path length computation. An extra matrix, called *connectivity matrix* contains the information about the available connections and directions between nodes (Fig. 2). As already explained in section 3, the generic element *ij* of the *connectivity matrix* is 0, if nodes *i* and *j* are connected, 1 otherwise.

As the modified HNN requires high spatial accuracy, a dense grid is needed, which implies an high number of neurons, because of the correspondence of nodes, checkpoints and neurons for this particular neural network mathematical model. For the modified HNN each neuron is connected with other 4 neurons, so a 4×4 *local connectivity matrix* is defined, instead of having a global matrix. In the proposed application, the four connected neurons are arranged in a + configuration, and the diagonal motion between nodes is so forth not allowed. In this case, the obstacle node indexes are stored in a different matrix and used to properly modify the *local connectivity matrix* during the simulation of the network dynamics.

As far as the graph theory based algorithm are concerned, the domain discretization coincides with the standard HNN application with a different concept of the *connectivity matrix*. The generic element *ij* is exactly the distance between the *i* and *j* nodes if connected, otherwise is 0.

A graphical user interface (GUI) has been developed, always in Matlab® environment, to make the preprocessing more user friendly and to easily access different airport database. The GUI showing the discretization of the *Sandro Pertini* airport for the graph algorithms is reported in Appendix A. The whole set of matrices represents the airport database named with its ICAO code.

4.2 Shortest path computation

The core of the implemented code is the shortest path computation and, as far as the HNN based algorithms are concerned, the time evolution of the state of the neural network is simulated by numerically solving Eq. (5). The net simulation corresponds to solve a system of $n\,(n-1)$ nonlinear differential equations, where the variables are the neuron output voltages V_{Xi}. The fourth order Runge-Kutta method has been chosen for the numerical solution in time domain, as it is sufficiently accurate and of easy implementation. Accordingly, the simulation consists in observing and updating the neuron output voltages at incremental discrete time steps $\delta t = 10^{-5} sec.$ The time constant τ of each neuron is set to 1 and, for simplicity, $\lambda_{Xi} = \lambda$ and $g_{Xi} = g$ are assumed to be independent of the subscript (X, i). Of critical importance is the network initial condition, which is defined by the neurons initial input voltages U_{Xi}. Albeit, a null voltage condition for the entire domain will guarantee to not preconditioning the final solution, the symmetric network topology imposes the insertion of some initial random noise $-0.0002 \leq \delta U_{Xi} \leq +0.0002$ to drive the net to one of the possible multiple shortest paths. The simulation is stopped when the system reaches a stable final state. This is assumed to occur when all neuron output voltages do not change by more than a threshold value of $\Delta V_{th} = 10^{-5}$ between two time steps. When the steady state of the net is reached, each neuron is defined either On if $V_{Xi} \geq 0.5$ or Off if $V_{Xi} \leq 0.5$. The described process is referred to a single mission phase and then is repeated three times per mission. The shortest path computed as the sum of the three phases is finally displayed on the airport satellite map.

Unlike the standard HNN, its modified version always leads to an optimal solution resulting in the shortest path from the starting node to the target. From the source position, the activation values of the neighbor nodes, belonging to the local connectivity matrix, is evaluated at each discrete time step. The highest activity value identifies the next neuron of the path. If none of the neighbor nodes has higher activity value than the actual neuron, the path does not evolve. This mechanism allows to simulate unsteady environments with, for example, moving obstacles or changing targets. The simulation terminates when the vehicle reaches the target. However, for sake of clarity, the dynamics of the moving obstacle must be slower than the net evolution time in order to be captured. Also in this case, the described process is referred to a single mission phase and then is repeated three times per mission. The shortest path computed as the sum of the three phases is finally displayed on the airport satellite map.

As far as the two graph theory based algorithms are concerned, after the airport database initialization, iterations start by adding the source node s to the set of the found shortest paths S. The algorithm prosecutes by repeating iteratively step 2 and 3, described in the previous section, until the target is added to the closed set S. As S contains the shortest path from the source to the different nodes, the shortest path from s to t must be reversely reconstructed: starting from the target node, the point stored in the corresponding $parent$ vector row is added to the path; the operation is repeated until the source node is reached.

4.3 Test case results using standard HNN

To validate the algorithms' functioning and to compare their performance, a test case application for the Turin airport *Sandro Pertini* is implemented (Fig. 1). A network of 102 checkpoints is created corresponding to the real airport parking decks and taxiways. The global *connectivity matrix*, defined using Eq. (7), respects the airport regulations. Value 1 is given to the matrix elements representing unconnected nodes, while the zeros are existing connections. As previously discussed, the matrix is not symmetric because some taxiways are one way only.

For the simulated test case an appropriate parameters' setup is found by the general rules reported in [18]:

$$\mu_1 = 100; \mu_2 = 3500; \mu_3 = 2500; \mu_4 = 100; \mu_5 = 3500.$$

In particular, the suboptimal parameter setup is a compromise between obtaining legitimate tours (μ_1 small) and heavily weighting the distances between nodes (μ_1 large). Furthermore, undersized or oversized values of λ (gain) result in fuzzy V_{Xi} distribution, which drives the simulation to a non-optimal solution. In the reported test case, this parameter is set to $\lambda = 50$. Results of the simulations are reported from Fig. 6 to Fig. 11, where two different tug missions are analyzed. The first is an aircraft departure mission where the tug initial position is in its deposit area and the airplane is located in a generic parking lot in the apron. The second is an aircraft arrival mission where the tug is randomly located in the apron, as its initial position, and an appropriate parking lot is chosen for airplane. For sake of clarity, each figure shows separately one mission phase, the optimized paths are depicted in the figures with different colors and the initial and target points are indicated respectively with a magenta and black cross. It is worth pointing out that the HNN parameters are kept constant during the entire simulation test case.

Figure 6: Standard HNN. Departure mission, Phase 1: from tractor deposit to departing aircraft parking lot. Credits: Pictures 2015 DigitalGlobe, Map data ©2015 Google.

Figure 7: Standard HNN. Departure mission, Phase 2: from departing aircraft parking lot to runway threshold. Credits: Pictures 2015 DigitalGlobe, Map data ©2015 Google.

Figure 8: Standard HNN. Departure mission, Phase 3: from runway threshold to tractor deposit. Credits: Pictures 2015 DigitalGlobe, Map data ©2015 Google.

4.4 Test case results using modified HNN

The same test case is solved with the modified HNN algorithms. For this application, a 700×700 neurons net has been created and dynamic obstacles are used to prevent convergence through unpermitted direction. This is a programming expedient to overtake the limitation of using a local connectivity matrix to speed up the solution.

Figure 9: Standard HNN. Arrival Mission: Phase 1, from tractor deposit to aircraft landing waiting position. Credits: Pictures 2015 DigitalGlobe, Map data ©2015 Google.

Figure 10: Standard HNN. Arrival Mission: Phase 2, from aircraft landing waiting position to parking lot. Credits: Pictures 2015 DigitalGlobe, Map data ©2015 Google.

Figure 11: Standard HNN. Arrival Mission: Phase 3, from aircraft selected parking lot to tractor deposit. Credits: Pictures 2015 DigitalGlobe, Map data ©2015 Google.

For this test case, the parameters are set as follows: the bias current for the target node is $E = 100$, while the time step is $dt = 10^{-2}$. Results are shown in figures from Fig. 13 to Fig. 18. The same missions considered for the previous algorithm are analyzed and, for consistency, results are shown in the same order as before.

It is worth pointing out the effectiveness of the virtual dynamic obstacle, as the resulting path is al-

Figure 12: "Sandro Pertini" airport scheme for modified Hopfield neural network application. Credits: Pictures 2015 DigitalGlobe, Map data ©2015 Google.

Figure 13: Modified HNN. Departure mission, Phase 1: from tractor deposit to departing aircraft parking lot. Credits: Pictures 2015 DigitalGlobe, Map data ©2015 Google.

Figure 14: Modified HNN. Departure mission, Phase 2: from departing aircraft parking lot to runway threshold. Credits: Pictures 2015 DigitalGlobe, Map data ©2015 Google.

ways the shortest path, Fig. 15 and Fig. 18. However, because of the higher number of checkpoints in the airport discretization, as shown in Fig. 12, the computational time increases with respect to the standard Hopfield Neural Network.

4.5 Test case results using graph theory based algorithms

For these test cases, the same spatial domain discretization used for the standard HNN is employed (Fig. 1). For consistency purpose, the same path optimizations are performed by means of the Dijkstra's and the $A*$ methods. In this cases, the optimal paths computed by the two algorithms is identical. Results are reported from Fig. 19 to Fig. 24.

Figure 15: Modified HNN. Departure mission, Phase 3: from runway threshold to tractor deposit. Credits: Pictures 2015 DigitalGlobe, Map data ©2015 Google.

Figure 16: Modified HNN. Arrival Mission: Phase 1, from moving tractor to aircraft landing waiting position. Credits: Pictures 2015 DigitalGlobe, Map data ©2015 Google.

Figure 17: Modified HNN. Arrival Mission: Phase 2, from aircraft landing waiting position to parking lot. Credits: Pictures 2015 DigitalGlobe, Map data ©2015 Google.

4.6 Results comparison

The result analysis suggests that the final optimized solutions, computed by the four implemented algorithms, slightly differ, as shown in Fig. 15 and Fig. 21. This might be caused by the different domain discretization employed. However, the length of the paths are very similar as shown in Table 1 and Table

Figure 18: Modified HNN. Arrival Mission: Phase 3, from aircraft selected parking lot to tractor deposit. Credits: Pictures 2015 DigitalGlobe, Map data ©2015 Google.1

Figure 19: Algorithm 3. Departure mission, Phase 1: from tractor deposit to departing aircraft parking lot. Credits: Pictures 2015 DigitalGlobe, Map data ©2015 Google.

Figure 20: Algorithm 3. Departure mission, Phase 2: from departing aircraft parking lot to runway threshold. Credits: Pictures 2015 DigitalGlobe, Map data ©2015 Google.

2, for the two missions respectively.
A substantial difference between the implemented algorithms arises when considering the computational time required to generate the optimal shortest path.

Figure 21: Algorithm 3. Departure mission, Phase 3: from runway threshold to tractor deposit. Credits: Pictures 2015 DigitalGlobe, Map data ©2015 Google.

Figure 22: Algorithm 3. Arrival Mission: Phase 1, from moving tractor to aircraft landing waiting position. Credits: Pictures 2015 DigitalGlobe, Map data ©2015 Google.

Figure 23: Algorithm 3. Arrival Mission: Phase 2, from aircraft landing waiting position to parking lot. Credits: Pictures 2015 DigitalGlobe, Map data ©2015 Google.

The code was executed on a Windows 8.1 Pro platform supported by an Intel Core i7-4810MQ CPU and 8 GB RAM. Table 3 reports the required computational time for each algorithm for both test cases. It is clear that the computational time required by the graph theory-based algorithms is some order of magnitude smaller than the one required by the two neural networks. In particular, the modified HNN is the slowest algorithm, because of the huge amount of nodes (neurons) required for the proposed application, especially during the first phase of the computation, when the target activation must propagate to the initial tug position so that it starts virtually moving.

Table 1: Optimal trajectories length comparison for the Departure mission.

Algorithm	**Phase1** [m]	**Phase2** [m]	**Phase3** [m]	**Total length** [m]
HNN	1097.20	1442.50	1342.60	3882.30
mHNN	1071.90	1407.50	1376.10	3855.50
Dijkstra	1097.20	1442.50	1342.60	3882.30
A*	1097.20	1442.50	1342.60	3882.30

Table 2: Optimal trajectories length comparison for the Arrival Mission.

Algorithm	**Phase1** [m]	**Phase2** [m]	**Phase3** [m]	**Total length** [m]
HNN	1106.00	1619.50	455.38	3180.88
mHNN	1126.40	1570.00	526.05	3222.45
Dijkstra	1106.00	1619.50	455.38	3180.88
A*	1106.00	1619.50	455.38	3180.88

Figure 24: Algorithm 3. Arrival Mission: Phase 3, from aircraft selected parking lot to tractor deposit. Credits: Pictures 2015 DigitalGlobe, Map data ©2015 Google.

Table 3: Computational time comparison for both missions.

Algorithm	***Departure*** [s]	***Arrival*** [s]
HNN	705.01	767.83
mHNN	1839.66	1272.93
Dijkstra	0.01	0.01
A*	0.01	0.01

5 Conclusion

This paper presents a software solution for a route selection problem applied to airport environment for the path planning of innovative and automatic aircraft tugs, responding to the need of engine-off taxi in the next airport generation. Four different algorithms, based on artificial intelligence and graph theory, have been implemented and their performance compared. The developed software works with any airport and is able to properly discretize the spatial domain (*e.g.* taxiways, runways and aircraft parking lots) in function of the employed algorithm. To solve the problem, an entire engine-off towing operation is called mission and is divided into three sub-operations, called mission phases. There are two types of mission, depending whether the aircraft to be towed is arriving or departing, whereas the phases sequence is the same and in the following order: reach the targeted aircraft, perform the engine-off taxi towing and return to the tug parking area. In detail, a standard Hopfield neural network and a modified version, which considers a local connectivity of neurons, based on harmonic functions, a *Dijkstra* and a *A** algorithms have been employed. The algorithms are conceptually different, with their own pros and cons. In particular, the standard HNN and the two graph theory based algorithms minimize the entire path in once, while the modified HNN works locally, in a small neighborhood of checkpoints, allowing for dynamic obstacles avoidance (i.e. moving tugs, airplanes and foreign objects along the path).

A unique test case, referred to the real data of the *Sandro Pertini* airport of Turin, has been performed for each algorithm for comparison purposes. The results demonstrate that all the algorithms converge to a unique shortest path, exception done for the modified HNN, which result in a slightly longer route. In addition, the two graph theory based algorithms are several order of magnitude more efficient, requiring an almost negligible computational time, making them more attractive for a possible real-time application.

A Airport Discretization Tool

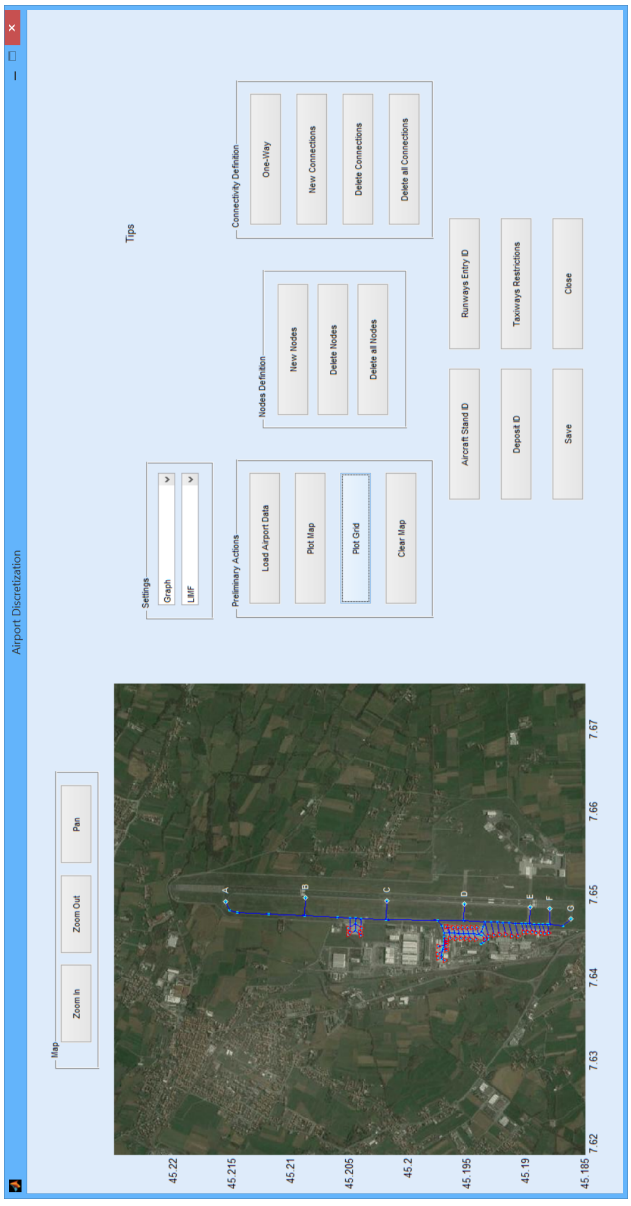

Figure 25: Airport Discretization Graphical User Interface. Credits: Pictures 2015 DigitalGlobe, Map data ©2015 Google.

Acknowledgements: Authors would like to thank Prof. Giovanni Squillero, from the Politecnico di Torino, for his advising during the implementation of the presented route selection algorithms.

References:

[1] Bondy A.J. and Murty U.S.R. *Graph Theory with Applications*.

[2] Goldberg A.V. and Tarjan R.E. Expected performance of dijkstra's shortest path algorithm. Technical report, NEC RESEARCH INSTITUTE REPORT, 1996.

[3] Bonet B. and Geffner H. Planning as heuristic search. *Artificial Intelligence*, 129(12):5 – 33, 2001.

[4] Hilburn B. Head down time in aerodrome operations: a scope study. 2004.

[5] M. Battipede, A. Della Corte, M. Vazzola, and D. Tancredi. Innovative airplane ground handling system for green operations. In 27^{th} *International Congress Of The Aeronautical Sciences ICAS*, 2010.

[6] Baker B.M. and Ayechew M.A. A genetic algorithm for the vehicle routing problem. *Computers and Operations Research*, 30(5):787–800, 2003.

[7] Lesire C. An iterative a* algorithm for planning of airport ground movements. In *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, pages 413–418, 2010.

[8] Dijkstra E.W. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[9] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 1998.

[10] ICAO. Manual on the prevention of runway incursions. Technical Report Doc 9870 AN/463, International Civil Aviation Organization, 2007.

[11] Hopfield J.J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

[12] Hopfield J.J. and Tank D.W. Neural computation of decisions in optimization problems. *Biological Cybernetics*, 52(3):141–152, 1985.

[13] Hopfield J.J. and Tank D.W. Computing with neural circuits: a model. *Science*, 233(4764):625–633, 1986.

[14] J.Y. Kim, Aktay K., Aktay K., and Ropp T.D. Ants-automated nextgen taxi system. FAA Design Competition 2009-2010, 2010.

[15] Lufthansa. Innovative taxibot now used in real flight operations, 2015.

[16] Gendreau M., Guertin F., J.Y. Potvin, and Taillard . Parallel tabu search for real-time vehicle routing and dispatching. *Transportation Science*, 33(4):381–390, 1999.

[17] Xu M.H., Liu Y.Q., Huang Q.L., Zhang Y.X., and Luan G.F. An improved dijkstras shortest path algorithm for sparse network. *Applied Mathematics and Computation*, 185(1):247 – 254, 2007.

[18] Ali M.K.M and Kamoun F. Neural networks for shortest path computation and routing in computer networks. *Neural Networks, IEEE Transactions on*, 4(6):941–954, Nov 1993.

[19] Christofides N. *Graph Theory: An Algorithmic Approach*. Computer Science and Applied Mathematics.

[20] Kojic N., Reljin I., and Reljin B. Route selection problem based on hopfield neural network. *Radioengineering*, 22(4):1182–1193, 2013.

[21] Wasserman P.D. *Advanced Methods in Neural Computing*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1993.

[22] Hart P.E., Nilsson N.J., and Raphael B. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, July 1968.

[23] S.A. Rahman, M.S. Ansari, A.A. Moinuddin, and et al. Solution of linear programming problems using a neural network with non-linear feedback. *Radioengineering*, 21(4):1171, 2012.

[24] Khebbache-Hadji S., Prins C., Yalaoui A., and Reghioui M. Heuristics and memetic algorithm for the two-dimensional loading capacitated vehicle routing problem with time windows. *Central European Journal of Operations Research*, 21(2):307–336, 2013.

[25] Mitrovi-Mini S., Krishnamurti R., and Laporte G. Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, 38(8):669 – 685, 2004.

[26] Zeng W. and Church R.L. Finding shortest paths on real road networks: The case for a*. *Int. J. Geogr. Inf. Sci.*, 23(4):531–543, April 2009.

[27] Zhong Y., Shirinzadeh B., and Tian Y. A new neural network for robot path planning. In *Advanced Intelligent Mechatronics, 2008. AIM 2008. IEEE/ASME International Conference on*, pages 1361–1366, July 2008.