Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 345049, 23 pages
http://dx.doi.org/10.1155/2015/345049

Research Article

A Universal Concept for Robust Solving of Shortest Path Problems in Dynamically Reconfigurable Graphs

Jean Chamberlain Chedjou and Kyandoghere Kyamakya

Institute of Smart Systems Technologies, Transportation Informatics Group (TIG), Universität Klagenfurt, Klagenfurt, Austria

Correspondence should be addressed to Kyandoghere Kyamakya; kyandoghere.kyamakya@aau.at

Received 27 May 2015; Accepted 4 November 2015

Academic Editor: John D. Clayton

This paper develops a flexible analytical concept for robust shortest path detection in dynamically reconfigurable graphs. The concept is expressed by a mathematical model representing the shortest path problem solver. The proposed mathematical model is characterized by three fundamental parameters expressing (a) the graph topology (through the "incidence matrix"), (b) the edge weights (with dynamic external weights' setting capability), and (c) the dynamic reconfigurability through external input(s) of the source-destination nodes pair. In order to demonstrate the universality of the developed concept, a general algorithm is proposed to determine the three fundamental parameters (of the mathematical model developed) for all types of graphs regardless of their topology, magnitude, and size. It is demonstrated that the main advantage of the developed concept is that arc costs, the origin-destination pair setting, and the graph topology are dynamically provided by external commands, which are inputs of the shortest path solver model. This enables high flexibility and full reconfigurability of the developed concept, without any retraining need. To validate the concept developed, benchmarking is performed leading to a comparison of its performance with the performances of two well-known concepts based on neural networks.

1. Introduction

The shortest path problem (SPP) is one of the classical combinatorial optimization problems [1–3]. This problem relates to finding the minimum cost path in a predefined source-destination node pair in a weighted graph network [3]. The interest devoted to shortest path finding can be explained by the various potential applications of the SPP in science and engineering. Indeed, the SPP is essentially involved in many use cases, in, for example, the following: vehicle routing in transportation systems [4], traffic routing in communication networks [5], path planning in robotic systems [6, 7] and scheduling [8], and video image analysis [9]. Further applications of SP are found in electronics (e.g., for VLSI physical design) [10], medical imagery (e.g., for virtual endoscopy) [11], and image processing (e.g., for energy minimization in vision) [12], just to name a few.

In intelligent transportation systems, the SPP can be considered as a subproblem for many broader problems such as route guidance [4], vehicle dispatching [13], real-time traffic information sensing [4], and production systems planning. These specific problems require real-time (e.g., ultrafast) processing in order to achieve results in extreme short time deadlines, examples of results being the efficient schedules and identification of new routes (or paths) in transportation networks.

In computer science SP algorithms can be used in the automatic search of directions between physical locations (e.g., driving directions on web mapping websites like in the American free online web mapping service (MapQuest) and also in the desktop web mapping service (Google Maps)) [14]. SP can also be used in applications like network management (e.g., finding the most vital node of a shortest path) [15] and graph structure in the web [16].

Thus, there is an explicit necessity of enriching the state of the art by developing new and efficient (i.e., extremely fast) shortest path problem (SPP) solver concepts. The efficiency in this context relates to some key performance metrics: (a) fast computing, (b) low memory consumption, (c) robustness, (d) accuracy, and (f) dynamic/runtime reconfigurability.

In the aforementioned applications, the shortest path detection problem is an important problem to be addressed. This problem has been studied extensively in the fields of computer science, operations research, and transportation engineering [4–9]. The well-known polynomial-time algorithms for solving shortest path problems include Bellman's dynamic programming algorithm [17], Dijkstra's algorithm [18], and Bellman-Ford successive algorithm [19, 20]. However, polynomial-time algorithms are suitable only for nonnegative costs of edges and for cases of additive linear path cost models. Thus, these algorithms cannot solve the SPP in the presence of negative costs [21] or in the presence of nonlinear/nonadditive path cost models [22]. Further, polynomial algorithms may appear to be too time consuming (slow) specifically when dealing with real-time applications in real traffic networks. These algorithms can also fail to provide the correct solution in certain cases such as when multiple shortest paths of equal total cost but with differences in the number of hops (i.e., number of involved edges) exist in a given graph.

A dynamic algorithm was developed as an extension of polynomial Dijkstra's and Bellman-Ford algorithms in order to analyze shortest path tree problems [23]. Despite the fact that the dynamic algorithm can easily adapt to changes in the graph topology, it is however very time consuming and the convergence is an issue specifically for huge size problems [23]. Further methods have been developed to address SP problems while assuming integer values of the costs of edges such as the scaling technique [24], the integer matrix multiplication technique [25], the fast integer sorting technique [26], and the component hierarchy technique [27]. These methods are not valid for noninteger values of edge weights and therefore are not applicable to real-life or technical systems.

A first basic mathematical approach for SPP solving consists of modeling the shortest path detection problem into a linear programming constrained problem, which is further solved using Dantzig's simplex method [28]. This method is prone to failures and the accuracy is poor [29]. Several interesting contributions for SPP solving presented in the related literature do involve selected artificial intelligence concepts. Genetic algorithm (GA) [29] and particle swarm optimization (PSO) [30] can solve both deterministic and stochastic shortest path problems. In both cases, the solutions converge towards the optimal paths. However, GA is computationally very expensive compared to PSO [30] while the PSO also needs additional heuristics. Likewise, evolutionary algorithms (EA) are prone to invalid paths detection [30]. Heuristic search algorithms (e.g., A^*) [21] are developed to make use of additional knowledge in order to reduce the search efforts; however, heuristic search algorithms depend on the quality of the heuristic function used. This dependency affects the accuracy of results. Further, when the search area increases, more computation effort is required [21]. Hence, the solutions provided by heuristic algorithms are possibly less accurate and cost inefficient.

The concept of cellular automata (CA) has been intensively used to address SP problems in complex graphs [31–33]. Despite the potential significant improvement of the computing speed due to the parallel nature of the CA concept (i.e., CA is suitable for parallel computing), the reconfigurability is however an issue due to the strong dependence of the CA concept on initial conditions/states [34].

Another interesting SPP solver concept does involve artificial neural networks (ANN). The ANN SPP solver has been given tremendous attention due to its capability of performing parallel computing as well as its easy hardware implementability [35]. However, the basic ANN approach is prone to limitations such as lack of adaptability to dynamic graph topological changes and poor accuracy of results [35–37]. The Hopfield neural networks were developed based on linear programming to provide approximate solutions (to SPP) faster than the aforementioned algorithmic solutions [35, 38–40]. Mehmet Ali and Kamoun [36] proposed a variation of Hopfield neural networks as a new concept that can adapt to external varying conditions. This method fails however to converge towards valid solutions. Further, the computing performance degrades with increasing magnitude and size of the graph. To address these last mentioned limitations, Park and Choi [37] proposed a concept that is capable of handling graphs with huge sizes. This last-named concept is however prone to convergence issues [41]. The dependent variable Hopfield neural network (DVHNN) [41] was also proposed as a new neural network concept capable of addressing the inherent limitations of the previously mentioned classical concepts involving neural networks. Specifically, it was demonstrated that the DVHNN can efficiently tackle issues related to accuracy, convergence, and reliability when dealing with SPP [41]. Despite these strong points of the DVHNN, the method cannot efficiently handle real-time SPP detection problems in reconfigurable graphs since new training or retraining (of the neural network) is needed for each new source-destination node-pair setting. Furthermore, the DVHNN does not consider negative cost of edges and the accuracy degrades with the increasing costs of edges (higher values of weight values). This is justified by the assumption in [41] which considered only small values of edge costs.

In view of the above underlined limitations which are inherent to most neural network based concepts, a new neural network approach (called dynamic neural network (DNN)) was introduced and some interesting related works do demonstrate the effectiveness and efficiency of the DNN approach (see [42] and the references therein). However, despite the very good features of the DNN for SPP solving, some crucial limitations are underscored in [42], which are specifically related to the convergence failure observed under specific parameter settings. Another interesting issue worth mentioning is the route to convergence. This issue, which is extremely sensitive to changes in both initial values and the parameter settings of the dynamic mathematical model, is characterized by the duration of the transient phase and the computing time needed to reach convergence.

The key/main objective of this paper is to contribute to the enrichment of the SPP related state of the art by proposing a new approach expressed in form of ordinary differential

equations. These equations do in essence represent a novel NAOP SPP solver model. This new form of NAOP can/does efficiently overcome the limitations of the abovementioned DVHNN and DNN concepts in [41] and [42], respectively. For proof of concept and for stress-testing purposes of the novel NAOP SPP solver approach developed, extensive benchmarking is conducted whereby its performance is compared with that of the competing concepts presented in [41, 42]. The performance metrics involved in the benchmarking are the convergence under high values of edge costs, the convergence in the presence of negative edge-cost values, the convergence potential under various parameter settings, the transient phase cancellation/duration, and the necessary computing time until convergence.

The rest of the paper is organized as follows. Section 2 describes at an abstract level the new proposed NAOP SPP solver concept for dynamically reconfigurable network graphs. A synoptic representation of this concept is proposed and a description of the key parameters as well as the complete system model is presented. Section 3 presents the general methodology for finding shortest paths in graphs through NAOP. The BDMM application to the shortest path problem is then addressed. The resulting coupled ODE equations are derived. The comprehensive benchmarking of the novel NAOP SPP solver is presented in Section 4 whereby a series of selected examples published in [41] (i.e., DVHNN concept) and [42] (i.e., DNN concept) is systematically considered. SPP solving is carried out using the NAOP SPP simulator on the one hand and the concepts in [41, 42] on the other hand. Some graph scenarios addressed and published in [41, 42] are considered and systematic benchmarking of the new NAOP solver concept with the concepts in [41, 42] is performed. The performance metrics used for the diverse comparisons are hereby mainly the simulation duration until convergence (computational efficiency) and the robustness (convergence). Regarding reconfigurability, it is only valid for the novel NAOP based SPP solver; the other concepts cannot support it. The benchmarking results obtained are used to underscore both the effectiveness and the efficiency of the novel NAOP SPP solver concept in complex and dynamically reconfigurable network graphs while considering even negative as well as high values of edge costs. Lastly, Section 5 presents a series of concluding remarks. A summary of the core contributions of this paper is presented. Further, selected interesting open research questions (under investigation in some of our ongoing subsequent works) are listed in an outlook.

2. General Methodology Based on Nonlinear Adaptive Optimization (NAOP) for Modeling Shortest Path Problems (SPP)

In some of our recent contributions/papers (see [43, 44]), the NAOP concept has been successfully used for solving differential equations (ODEs and/or PDEs). We now want to demonstrate that the NAOP concept can be efficiently used as a general and robust framework for modeling shortest path problems (SPP) even in dynamically reconfigurable graphs.

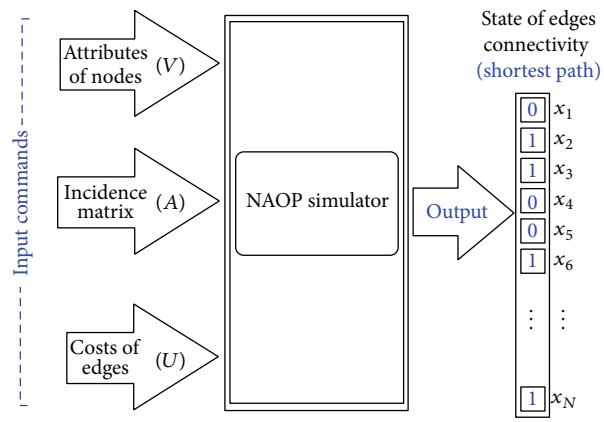

Figure 1: Synoptic representation of the system describing the NAOP based SPP solver concept. The input commands (or external commands) V and U are used, respectively, for assigning attributes to nodes and values (or costs) to edges. The matrix A is the incidence matrix of the graph under investigation. The output x_i ($i = 1, 2, 3, \ldots, N$) expresses the belonging (i.e., value is 1) or not (i.e., value is 0) to the shortest path (SP) of all edges of the graph under investigation.

This section provides a full description of the proposed NAOP concept for finding shortest paths in complex and dynamically reconfigurable graphs/networks. The abstract input/out logical diagram of this concept is schematically presented in Figure 1. The proposed concept does take three basic inputs as illustrated in Figure 1.

The first input is an external command V (see Figure 1), which is used for assigning one of the three possible attributes (i.e., source, destination, or intermediate) to each of the M nodes of the graph under investigation. The command V is defined as follows: $\vec{V} = [\delta_1, \delta_2, \ldots, \delta_{M-1}, \delta_M]^T$, whereby $\delta_j = 1$ when the node with index j is a source node, $\delta_j = -1$ when the node with index j is a destination node, and $\delta_j = 0$ when the node with index j is a candidate intermediate node on the shortest path (in fact, after origin and destination nodes are both fixed, all remaining nodes of the graph are "candidate" intermediate nodes on the final shortest path).

The second input A (see Figure 1) expresses the topology of the graph under investigation. The incidence matrix denoted by A is $M \times N$ matrix, whose elements can only take three possible values from the set $\{-1, 0, 1\}$. In essence, the matrix A expresses the states of connectivity/incidence of all edges to nodes (incidence matrix). A directed edge going onto a node is denoted in the matrix A by -1 while 1 denotes a directed edge leaving a node. The state of an edge, which is not connected to any node, is expressed by 0. In the matrix A, each column corresponds to a given edge and each row does correspond to a given node. This is clearly illustrated in (10b).

The third input U (see Figure 1) is used for assigning weight values to all N (directed) edges of the graph. The vector U is defined in terms of the elements c_i as follows: $\vec{U} = [c_1, c_2, \ldots, c_{N-1}, c_N]^T$. The coefficients c_i represent the weight values (i.e., costs) of all edges of the graph.

The output in Figure 1 expresses the state of connectivity (i.e., belonging to the SP) of all edges. The states of edges are represented by the variables x_i. A value $x_i = 1$ means that the edge with index i is part of the shortest path; otherwise, $x_i = 0$ means it is not.

The concept developed here is general and scalable regarding graph size and magnitude and possibly bidirectional connectivity between nodes.

The NAOP concept in Figure 1 has already been successfully applied for solving both nonlinear ordinary differential equations (ODEs) and partial differential equations (PDEs) [43, 44]. The aim of this paper is to demonstrate that the NAOP concept (see Figure 1) can also be efficiently used as a universal framework to realize a robust SPP solver (called NAOP SPP simulator). Some aspects of the "universality" flavor of this NAOP SPP solver are expressed amongst others by the fact that this concept is applicable to various types of graphs/networks regardless of size, magnitude, and nature/form of the costs of edges (negative and/or positive values). And although it is not addressed in this paper (this issue will be extensively handled in a subsequent work), this concept is also applicable for cases where the path cost is non-additive, that is, nonlinear (e.g., see a possible case where SPP is combined with network/graph reliability considerations (in such a case, every edge has 2 weights: a deterministic weight (w_i) related to the SPP problem and a probability value (p_i) related to the graph reliability perspective; then the path cost is $c_1 + c_2$, whereby c_1 is the sum of all w_i belonging to the path and c_2 is the product of all p_i belonging to the path) or a case where the contribution of every edge belonging to the total path cost related to the "driving energy consumption perspective" is the sum of (a) edge length and (b) square of the edge's maximum (or current) driving speed).

During the past/last decades, tremendous attention has been devoted to the development of new methods, concepts, algorithms, and tools based on the neurocomputing paradigm (e.g., use of neural networks) for finding shortest paths in networks/graphs [1–9, 13, 17, 29, 30, 35, 36, 41, 42]. However, already published works are mainly just focusing on proof-of-concept examples and do not solve a series of fundamental issues related amongst other things to a systematic and general (or universal) methodological framework for efficiently solving shortest path problems in graphs of complex topologies. An open challenge is related to suggesting and validating a successful and systematic concept for controlling and mastering the following key performance metrics: accuracy, precision, robustness, flexibility, reconfigurability, and universality.

The key contribution of this paper is to develop a comprehensive NAOP SPP solver concept for finding shortest paths in complex and dynamically reconfigurable graphs which satisfactorily addresses and solves the abovementioned key challenges. The overall superiority of this concept is demonstrated in extensive benchmarking where the above-listed performance metrics are used.

The next section presents with full details the steps involved in the design of the NAOP SPP solver concept. The derivation of the related coupled ODE mathematical models is systematically performed.

3. General Methodology for Modeling the Shortest Path Problem through NAOP: Expression of the Lagrange Function and Derivation of the NAOP Mathematical Model

3.1. Transforming SPP Problem into an Optimization Problem. We now consider the derivation of the Lagrange function corresponding to the shortest path problem. We provide a description of the key steps involved in the mathematical modeling of the shortest path problem as a multivariable nonlinear optimization problem. Finally, we explain the process leading to the derivation of the set of coupled nonlinear ODEs, which correspond to the model of the NAOP SPP solver simulator.

The first step relates to the derivation of the corresponding Lagrange function (see (4)), which is viewed as the total energy of the system. In this context, the optimization problem is represented by an objective function (see (1)) subject to constraints ((2) and (3)). These constraints make the problem NP-hard. In order to solve NP-hard problems, the constrained optimization problem can be transformed into an unconstrained optimization problem by using relaxation methods. Several methods have been proposed by the relevant literature (see [9, 13, 17–22, 28, 29]). Some of these methods are direct substitution, constrained variation, and Lagrange multipliers [6], just to name a few.

The shortest path problem can be considered as a multivariable constrained optimization problem. This type of problem can be formulated mathematically by (1), (2), (3), and (4). In these equations, $\vec{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_{M-1}, \lambda_M]^T$ is the vector of multipliers variables λ_m and $\vec{\gamma} = [\gamma_1, \gamma_2, \ldots, \gamma_{N-1}, \gamma_N]^T$ is the vector of multipliers variables γ_n. The decision variable $\vec{x} = [x_1, x_2, \ldots, x_{N-1}, x_N]^T$ is the state vector of all edges x_i involved in the graph under consideration. Thus, the dimension of $\vec{x}$ is equal to N ($\dim(\vec{x}) = N$), where N is the size of the graph. The function $f(\vec{x})$ represents the total cost of the graph under investigation. The optimization constraints $g_m(\vec{x}) = 0$ are formulated in order to ensure the connectivity of each node belonging to the shortest path, while the constraints $h_n(\vec{x}) = 0$ impose the binarization of all components of the decision variable vector $\vec{x}$. These last-named constraints ($h_n(\vec{x}) = 0$) are formulated to ensure the convergence of all components x_i of the vector $\vec{x}$ to binary variables (0 or 1). Thus, according to the definition of the optimization constraints, the indexes i, m, and n are defined as follows: $i = 1, 2, 3, \ldots, N$; $m = 1, 2, 3, \ldots, M$; $n = 1, 2, 3, \ldots, N$. The integers M and N represent, respectively, the magnitude and the size of the graph. Hence,

$$\text{Min} \quad [f(\vec{x})] = 0 \tag{1}$$

$$g_m(\vec{x}) = 0 \tag{2}$$

$$h_n(\vec{x}) = 0 \tag{3}$$

$$\widetilde{L} = f(\vec{x}) + \sum_{m=1}^{M} \lambda_m g_m(\vec{x}) + \sum_{n=1}^{N} \gamma_n h_n(\vec{x}). \tag{4}$$

The second step exploits the concept of neuron dynamics as an optimization strategy that maps the optimization problem unto the energy of a neural network (e.g., a Hopfield network) in order to find the optimal solution. In this context, we represent this energy by the Lagrange form and the minimization of the Lagrange function leads to a stable state (due to the already proven convergence of the Basic Differential Multiplier Method (BDMM); see [38, 43, 44]).

Applying the BDMM to the Lagrange function in (4) leads to the derivation of

$$\dot{x}_i = -\alpha \partial_{x_i}\left(\tilde{L}\right), \tag{5a}$$

$$\dot{\lambda}_m = \beta \partial_{\lambda_m}\left(\tilde{L}\right), \tag{5b}$$

$$\dot{\gamma}_n = \beta \partial_{\gamma_n}\left(\tilde{L}\right). \tag{5c}$$

The symbols ∂_{x_i}, ∂_{λ_m}, and ∂_{γ_n} denote the partial derivatives with respect to x_i, λ_m, and γ_n, respectively. As already mentioned above, the variables x_i are the components of the vector of decision variables $\vec{x}$, while λ_m and γ_n are components of the vectors of multiplier variables $\vec{\lambda}$ and $\vec{\gamma}$, respectively.

The set of (5a), (5b), and (5c) is the characteristic model of BDMM [38, 43, 44]. This model (from which the coefficients of the coupled ODEs for a given graph are calculated) reveals the coupling between the dynamics of decision variable x_i with the dynamics of multiplier variables λ_m and γ_n. The parameters α and β are step sizes for updating decision variables and multiplier variables, respectively. The values of these parameters are chosen through a stability or convergence analysis (see Section 3.3). A similar study (i.e., stability analysis) is considered (in [43, 44]) and it is demonstrated that the sign of the damped mass matrix depends on the parameters α and β. Specifically, it is demonstrated (see [43, 44]) that the convergence is ensured for positive definite damped mass matrix. Thus, the values of α and β must be chosen accordingly. However, the convergence analysis in [43, 44] was considered in the design of a general and robust framework for ultrafast solving of ordinary and/or partial differential equations. Our aim in this paper is to extend the convergence analysis developed in [43, 44] to the case of the design of a novel, general, and robust SPP solver concept to which we have assigned the acronym of NAOP SPP solver. The output of the NAOP SPP solver is expressed by the solutions x_i of (5a). x_i are decision variables of the optimization process. These solutions, which reveal the state of edges connectivity, will converge to binary variables (0 or 1), 0 meaning that this edge does not belong to the SP and 1 meaning that it does.

The next subsection (see Section 3.2) shows how the set of (5a), (5b), and (5c) is used to derive the coupled mathematical model of the NAOP SPP solver simulator.

3.2. Derivation of the Mathematical Model of the NAOP SPP Solver Simulator. We provide in this part a full explanation of the systematic concept leading to the ODE based mathematical modeling of the shortest path problem as well as the design and implementation of the appropriate NAOP SPP solver simulator. Our focus is on finding the shortest path from a source node (s) to a destination node (t). The paths in the graph $G = (V, E)$ can be expressed by the objective function in (1), where the function $f(\vec{x})$ in (6) represents the total cost corresponding to the graph under investigation. In (6), N is the size of the graph. Hence,

$$f(\vec{x}) = \sum_{i=1}^{N} c_i x_i. \tag{6}$$

The minimization of the total cost in (6) corresponds to the shortest path calculation whereby this path is defined starting from a given source node (s) to a destination node (t).

The next step of the modeling process is related to the formulation of the optimization problem's constraints. Two key constraints are considered in order to fulfill some key requirements of the problem formulation. These requirements are summarized by the following conditions or constraints: (a) connectivity of nodes (by edges) and the participation of incident edges belonging to the path regarding, respectively, inflow nodal degree and outflow nodal degree and (b) binarization of the decision variables x_i, which are used to express the belonging or not of any edge to the shortest path.

The first-mentioned constraint (a) is modeled mathematically by (7). In (7), x_i is the state of an edge. The graph is assumed to be directed. In case of undirected graph, meaning bidirectionality between all connected node pairs, each direction is represented by a separate directed edge:

$$\left(\sum_{\substack{i=1\\ i\neq j}}^{N} a_{ij}x_i - \delta_j\right) = 0, \quad \text{Node } j \ \left(j = 1, \ldots, M\right). \tag{7}$$

For each node j, on the corresponding line/row in matrix A we have 3 groups of values which are reflected in (7): (i) a group of ingoing edges represented by values $a_{ij} = -1$, (ii) a group of outgoing edges represented by values $a_{ij} = +1$, and (iii) a group of nonincident edges (i.e., edges not connected to node j) represented by values $a_{ij} = 0$. The parameter δ_j is used to assign one of the three attributes (i.e., source, destination, or intermediate) to node j as follows: $\delta_j = 0$ (if j is an intermediate node); $\delta_j = +1$ (if j is a source node); $\delta_j = -1$ (if j is a destination node).

The previously mentioned constraint (b) is related to the binarization of all state variables, which are used to express the belonging to the shortest path (SP) of any edge. This constraint is modeled mathematically by

$$x_i\left(x_i - 1\right) = 0. \tag{8}$$

Thus, according to (4), the objective function in (6) can be combined with the constraints formulated in (7) and (8) to derive the Lagrange expression given in

$$\tilde{L} = f(\vec{x}) + \sum_{j=1}^{M} \lambda_j \left(\sum_{\substack{i=1\\ i\neq j}}^{N} a_{ij}x_i - \delta_j\right) + \sum_{i=1}^{N} \gamma_i x_i \left(x_i - 1\right). \tag{9}$$

In (9), $\vec{x}$, $\vec{\lambda}$, and $\vec{\gamma}$ are vectors of, respectively, N, M, and N components as already described above.

The expression $\widetilde{L}$ in (9) is further substituted into the mathematical expression of the BDMM technique (see the expression in (5a), (5b), and (5c)).

The BDMM application to (9) is done as follows: note that each decision variable and each multiplier variable are considered as a particular direction in a multidimensional space. In this context, the dimension of the multidimensional space is always equal to $N + M + N$ (i.e., N decision variables x_i, then M multiplier variables of type λ_j, and N multiplier variables of type γ_i). Thus, the BDMM application according to (9) corresponds to making partial derivations with respect to each particular direction. In each of these partial derivations only the variable expressing that particular direction is considered "dependent variable" in (9) and all other variables (or directions) are taken/handled as constant variables/values in the partial derivation process. Each partial derivation with respect to a particular variable will result in a related ordinary differential equation. Thus, we will obtain a total of $N + M + N$ coupled nonlinear differential equations, which can be grouped and expressed in a matrix form as indicated in (10a) and (10b). These equations ((10a) and (10b)) constitute the NAOP SPP solver model used for shortest path finding. This set of (10a) and (10b) is a general mathematical model, which is applicable (for shortest path detection) to any graph regardless of its type (i.e., directed or undirected graph, etc.) and also regardless of both its magnitude and its size. Hence,

$$\begin{aligned} \dot{\vec{x}} &= \alpha\left[-A^T\vec{\lambda} + (1-2\vec{x})\,\vec{\gamma} - \vec{U}\right], \\ \dot{\vec{\lambda}} &= \beta\left[A\vec{x} - \vec{V}\right], \\ \dot{\vec{\gamma}} &= \beta\left[\vec{x}\,(\vec{x}-1)\right]. \end{aligned} \tag{10a}$$

In (10a), the matrix A (graph's incidence matrix) is expressed as follows:

$$A = \begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & \cdots & x_i & \cdots & x_N \end{array} \\ \begin{bmatrix} 1 & -1 & \cdots & 1 & \cdots & -1 \\ -1 & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \cdots & -1 & \cdots & 1 \end{bmatrix} \end{array} \begin{array}{c} \\ N_1 \\ N_2 \\ \\ N_j \\ \\ N_M \end{array} \tag{10b}$$

The matrix A is the incidence matrix of the graph and A^T denotes the transpose of A. This matrix is of dimension $M \times N$, where M and N are, respectively, the magnitude (number of nodes) and the size (number of edges) of the graph. The elements of A, which are illustrated in (10b), express specific connectivity between nodes and edges. For example, the first column of A shows the connection between nodes N_1 and node N_2 through the edge x_1 in the direction from N_1 to N_2 denoted as $N_1 \xrightarrow{x_1} N_2$. Similarly, the other columns of A express the following connectivity: column-2 ($N_1 \xleftarrow{x_2} N_j$), column-i ($N_1 \xrightarrow{x_i} N_M$), and column-$N$ ($N_1 \xleftarrow{x_N} N_M$). Thus, the derivation of the incidence matrix A is immediate to a given graph.

Overall, for any given graph, one should first give IDs to all nodes by numbering them in any order with natural numbers; the same is done for giving IDs to all edges (here also the numbering order is not important). After all nodes and edges have been given an ID, the matrix A can be determined and it does express the current graph topology. It should be mentioned here that the use of the variables x_i is twofold: the variables x_i are used for the identification of the edges belonging to the graph. These variables are also solutions of (10a) and (10b). Thus, the variables x_i do therefore depict the state of all edges of the graph. Specifically, the solutions $x_i = 1$ reveal the edges involved in the shortest path, while $x_i = 0$ stand for those edges, which are not included in the shortest path.

Important Remark. Consider the following:

> The application examples (see Section 4) demonstrate the straightforward application of the general concept in (10a) and (10b) to the cases of directed graphs with positive costs of edges, directed graphs with mix of positive and negative costs of edges, and so forth. All the graph examples envisaged in this paper are already considered in [41, 42]. Benchmarking of our results against the results published in [41, 42] is further considered. Besides robustness, the core metric for comparison is the number of iterations needed by each method (i.e., the methods in [41, 42] and our NAOP SPP solver concept) until the convergence to the exact shortest path is achieved. Several scenarios are envisaged in which the nodes are assigned different attributes (source, destination, or intermediate).

According to (9), the following important statements can be made to explicitly describe the set of (10a) and (10b) in order to allow easy use of this set of coupled equations as a general modeling framework, applicable to all graph architectures for shortest path detection. Hence, we have the following statements:

(i) $\vec{x}$ is the vector of decision variables with components x_i. Thus, as already mentioned above, $\dim(\vec{x}) = N$, where N is the size of the graph.

(ii) $\vec{\lambda}$ is a first vector of multiplier variables with components λ_m. Thus, as already mentioned above, $\dim(\vec{\lambda}) = M$, where M is the magnitude of the graph.

(iii) $\vec{\gamma}$ is a second vector of multiplier variables with components γ_n. Thus, as already mentioned above, $\dim \vec{\gamma} = N$, where N is the size of the graph.

(iv) $\vec{U}$ is a vector of all edges' weights with components c_i. Thus, $\dim(\vec{U}) = N$.

(v) $\vec{V}$ is the vector for assigning attributes to nodes, that, is nodes' configuration vector. The components of $\vec{V}$ are denoted by δ_j. Thus, as already mentioned above, $\dim(\vec{V}) = M$, where M is the magnitude of the graph.

Comment on (10a) and (10b). The set of (10a) and (10b) contains three key fundamental parameters which are the external commands $\vec{V}$, $\vec{U}$, and A. The derivation (or calculation) of these key fundamental parameters is straightforward for a given graph architecture:

(i) The matrix A defines the topology of the graph under investigation. This matrix provides the connections between nodes, the identities of connectors (edges), and the valency of each node belonging to the graph.

(ii) The vector $\vec{U}$ offers the possibility of dynamically externally varying the costs of edges (weight values) c_i. This is performed through an external command (see Figure 1).

(iii) The vector $\vec{V}$ offers the possibility of performing dynamic reconfigurability of the graph nodes using an external command (see Figure 1). This reconfigurability offers the possibility of defining each node-j of the graph to be either source node ($\delta_j = +1$) or destination node ($\delta_j = -1$) or intermediate node ($\delta_j = 0$). The command δ_j is thus used for assigning a specific attribute to each node of the graph.

(iv) In essence, the strong points of the general mathematical model in (10a) and (10b) when compared with the traditional methods, concepts, and algorithms for solving shortest path problems are fourfold:

(a) The high flexibility of the set of (10a) and (10b), which can easily be adaptable to any new graph topology.

(b) The universality of the set of (10a) and (10b), which is valid for all graphs regardless of the type, the magnitude, the size, and the topology.

(c) The excellent dynamic reconfigurability through an external command without any need for retraining (contrarily to this, most traditional SPP solver methods do generally need new retraining or equivalent reprocessing in case of dynamic change of parts of the graph's attributes). The reconfigurability potential is used to dynamically change not only edge weights but also "source-destination" node pairs through external commands without any need for retraining.

(d) The robustness and correctness of the model in (10a) and (10b), which are very good, are underscored by the fact that the model always converges to the exact shortest path and, further, there is no faulty SP detection ever observed after very extensive stress testing of the concept.

If one uses the dynamic systems' terminology, (10a) and (10b) do describe indeed a nonlinear oscillator system that is externally excited by 3 inputs vectors describing the graph under consideration (matrix A, vector $\vec{U}$, and vector $\vec{V}$) and its oscillations over time do converge to fix points (or equilibrium points) that represent the shortest path of the graph described by the 3 inputs. These inputs constitute the signature of any graph.

An important remark to be underscored here is that the set of (10a) and (10b) is expressed in terms of two specific parameters α and β, which are step sizes for updating decision variables and multiplier variables, respectively. The aim of the next subsection (see Section 3.3) is to demonstrate the possibility of efficiently using these two parameters as control parameters to ensure/guarantee that the set of (10a) and (10b) always converges to the exact shortest path whatever the topology, size, magnitude of the graph, and so forth could be. The derivation of a systematic framework to control the convergence properties of the NAOP SPP solver in (10a) and (10b) is of necessary importance in order to guarantee excellent scalability potential of (10a) and (10b) when investigating SPP in complex graphs of huge size and high magnitude. Thus, the analytical condition established below (see Section 3.3) will be considered in order to improve the robustness of the optimization process and facilitate fast convergence to the exact shortest path.

The convergence properties of the NAOP SPP simulator are addressed (or considered) in the next subsection and analytical criteria are derived under which the NAOP SPP simulator always converges to the exact shortest path.

It is worth mentioning that the relevant literature has clearly demonstrated that the convergence of the analysis methods/concepts for solving shortest path problems (SPP) based on traditional neural networks (ANN) (see [45] and the references therein) and the concepts based on recurrent neural networks (RNN) (see [46] and the references therein) is an important issue to which full attention should be devoted. However, the literature does not propose an analytical framework that could help to systematically predict and control the convergence properties of both ANN and RNN concepts under various possible changes/configurations (e.g., variations in graph topology, choice of source-destination pair-node, size, and magnitude.). This remark is important to justify the analytical concept developed in this paper in order to provide a systematic analytical concept that efficiently addresses the issue related to convergence.

3.3. Convergence of the NAOP SPP Solver Concept under All Possible Combinations or Settings of Parameters. The key important issue in this context is how to ensure convergence of the NAOP SPP solver concept for all possible parameter settings of (10a) and (10b). These parameters are A, $\vec{V}$, $\vec{U}$, α, and β. However, since the first three parameters are derived according to a given graph under investigation, we will be deriving a general analytical framework for convergence regardless of these parameters (A, $\vec{V}$, and $\vec{U}$). Thus, the analytical expression derived as a general framework for convergence will be controlled by two specific and fundamental parameters which are the step size for updating decision

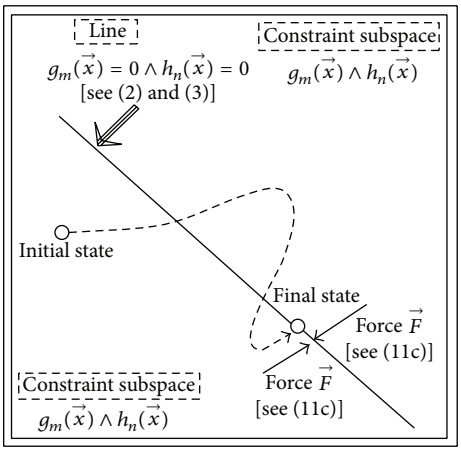

Figure 2: The sign flip concept is illustrated as a key step towards the convergence of the BDMM algorithm. Damped oscillations are exhibited around the constraint subspace leading to constraints minimization (system's final state).

neurons α and the step size for updating multiplier neurons β.

The convergence problem was already addressed in our previous contributions (see [43, 44]) when dealing with the design of a universal framework based on NAOP for solving ordinary differential equations (ODEs) and partial differential equations (PDEs). Our aim here is to provide the full/complete steps of the analytical process for ensuring convergence of the NAOP concept for shortest path detection. Further, we provide (see the Appendix) an in-depth analytical study of the convergence properties of the NAOP SPP solver concept. Such in-depth development was not considered in our previous papers ([43, 44]). Further, another motivation of carrying out an in-depth analytical study to ensure convergence of the NAOP framework for solving shortest path problems (SPP) is justified by our wish to demonstrate the possibility of controlling the convergence properties of the NAOP based solver for SPP through specific and well-known parameter settings. This will offer the possibility of using NAOP as a general and robust framework for the investigation of shortest path problems (SPP).

Regarding the convergence, we are providing here a practical/physical comment of the analytical results we want to achieve/derive. Indeed, the stability (or convergence) of the optimization process is ensured by the "sign flip" concept [43, 44]. This concept is illustrated in Figure 2. The "sign flip" concept can be explained as follows. As the parameters α and β vary, the system in (5a), (5b), and (5c) (or equivalently in (10a) and (10b)) exhibits damped oscillations around the constraints subspace defined by the relation $g_m(\vec{x}) = 0 \wedge h_n(\vec{x}) = 0$. The damped oscillations exhibited around the constraints subspace $g_m(\vec{x}) = 0 \wedge h_n(\vec{x}) = 0$ are characterized by the alternation (in time domain) of the sign of the constraint functions (i.e., $g_m(\vec{x}) < 0 \wedge h_n(\vec{x}) < 0$, $g_m(\vec{x}) > 0 \wedge h_n(\vec{x}) > 0$, and vice versa) during the optimization process in order to converge to a point located on the constraint subspace defined by the equation $g_m(\vec{x}) = 0 \wedge h_n(\vec{x}) = 0$ (see Figure 2). The damped mass matrix A_{ij} is used to provide an insight of the abovementioned convergence. Indeed, the damped mass matrix controls the energy dissipation within the system. Thus, according to (15) a positive value of the damped mass matrix is an insight that the total energy E_T within the system decreases with time and, finally, the system settles down into the state where the energy is minimized. This characterizes attraction of the system state to the constraint subspace (stability). At the attraction point (or convergence state (see Figure 2)), all constraints are fulfilled. However, when the damped mass matrix is negative, the total energy within the system increases with time. A negative damped mass matrix is an insight that all constraints are not fulfilled (instability). To tackle this problem, the learning rate parameters (α and β) can be tuned/monitored (independently of the other system's parameters A, $\vec{V}$, and $\vec{U}$) in order to bring the system model in (5a), (5b), and (5c) (or equivalently in (10a) and (10b)) into its stable state (i.e., convergence).

A full/complete detail of the analytical steps involved in the stability/convergence analysis of (10a) and (10b) is provided in the Appendix. The results provided in the Appendix can be summarized by the following important steps.

Step 1. The first step is the derivation of the characteristic ODE model for decision variables x_i. The Lagrange function in (4) is combined with (5a), (5b), and (5c) to derive the following nonlinear ordinary differential equation (see analytical details in the Appendix):

$$\ddot{x}_i + \sum_{j=1}^{N} \left[A_{ij} \dot{x}_j \right] + \vec{F} = 0. \tag{11a}$$

The expression in (11a) is a second-order ordinary differential equation with a single and dissipative coefficient denoted by A_{ij}. This coefficient represents the damped mass matrix defined as follows:

$$A_{ij} = \alpha \left[\frac{\partial^2 f}{\partial x_i \partial x_j} + \sum_{m=1}^{M} \lambda_m \frac{\partial^2 g_m}{\partial x_i \partial x_j} + \sum_{n=1}^{N} \gamma_n \frac{\partial^2 h_n}{\partial x_i \partial x_j} \right]. \tag{11b}$$

In (11a), $\vec{F}$ is an internal force producing the potential energy into the system. This force is expressed as follows:

$$\vec{F} = 2\alpha\beta \left[\sum_{m=1}^{M} g_m \frac{\partial g_m}{\partial x_i} + \sum_{n=1}^{N} h_n \frac{\partial h_n}{\partial x_i} \right]. \tag{11c}$$

In (11a), (11b), and (11c), $\ddot{x}_i = d^2 x_i / dt^2$ represents the force of inertia, A_{ij} is the damping matrix, and $\vec{F}$ is an internal force, which creates the potential energy (i.e., the internal energy) of the system. This force $\vec{F}$ is needed in order to fulfill the constraints (by moving the attractor from a given initial point chosen in the constraint subspace to a final point located on the characteristic line $g_m(\vec{x}) = 0 \wedge h_n(\vec{x}) = 0$).

Step 2. The second step is the derivation of the mathematical expression of the total energy of the system. This expression is further used in Step 3 for the global stability analysis. The Lyapunov function is derived as the total energy of the system $E_T(x_i)$. The total energy is expressed as the sum of kinetic and potential energies denoted by $E_C(x_i)$ and $E_P(x_i)$, respectively. These energies are expressed by the following mathematical expressions while considering the state variable x_i in (11a), (11b), and (11c):

$$E_P(x_i) = \int \vec{F} \partial x_i, \tag{12a}$$

$$E_C(x_i) = \sum_{i=1}^{N} \left[\frac{1}{2} (\dot{x}_i)^2 \right]. \tag{12b}$$

Thus, the total energy is expressed as follows:

$$E_T(x_i) = \sum_{i=1}^{N} \left[\frac{1}{2} (\dot{x}_i)^2 \right] + \int \vec{F} \partial x_i. \tag{13a}$$

Substituting the expression of $\vec{F}$ defined in (11c) into expression (13a) leads to the following expression of the total energy (see analytical details in the Appendix):

$$E_T = \sum_{i=1}^{N} \left[\frac{1}{2} (\dot{x}_i)^2 \right] + \alpha\beta \left[\sum_{m=1}^{M} \left(g_m^2 \right) + \sum_{n=1}^{N} \left(h_n^2 \right) \right]. \tag{13b}$$

The total energy in (13b) is positive definite for all x_i when α and β are of the same sign. Thus, according to the first criterion of the Lyapunov theorem, the total energy in (13b) (i.e., the Lyapunov function) is positive definite for all state variables x_i. Hence, the first criterion of the Lyapunov theorem for global stability is fulfilled (i.e., $E_T(x_i) > 0$ for all x_i).

Step 3. The third step is the derivation of the second criterion of the Lyapunov theorem for global stability. This criterion states that the first derivative of the Lyapunov function must be less than or equal to zero for all state variables x_i. Thus, the aim of this step is to derive the analytical condition under which the first derivative of the Lyapunov function (i.e., the total energy) must be less than or equal to zero (i.e., $\dot{E}_T(x_i) \leq 0$ for all x_i). Using the mathematical expression in (13b), the first derivative of the total energy can be evaluated and expressed as follows:

$$\begin{aligned} \dot{E}_T = & \sum_{i=1}^{N} [\dot{x}_i \ddot{x}_i] \\ & + 2\alpha\beta \left[\sum_{m=1}^{M} \left(g_m \frac{dg_m}{dt} \right) + \sum_{n=1}^{N} \left(h_n \frac{dh_n}{dt} \right) \right]. \end{aligned} \tag{14}$$

Using the expression in (14), an in-depth analytical development is conducted (see full details in the Appendix) and finally the first derivative of the total energy is expressed into the following simplified form:

$$\dot{E}_T(x_i) = -\sum_{i=1}^{N} \sum_{j=1}^{N} \left(\dot{x}_i A_{ij} \dot{x}_j \right). \tag{15}$$

The expression in (15) is the characteristic mathematical expression used to illustrate and underscore the significance of the "sign flip." Indeed as already mentioned, the Lyapunov theorem for global stability analysis states that the system is stable if the first derivative of the total energy (i.e., the Lyapunov function) in (15) is less than or equal to zero. In (15), $\dot{x}_i$ and $\dot{x}_j$ are, respectively, the time derivative of decisions variables in the directions i (row of the matrix A_{ij}) and j (column of the matrix A_{ij}). Thus, $\dot{x}_i$ and $\dot{x}_j$ are orthogonal and therefore $\dot{x}_j$ is the transpose of $\dot{x}_i$ denoted by $\dot{x}_j = [\dot{x}_i]^T$.

According to the well-known theorem of linear algebra (regarding the definition of the sign of a matrix), the damped mass matrix A_{ij} is positive definite if and only if $(\dot{x}_i A_{ij} \dot{x}_j) \geq 0$ for all x_i ($\dot{x}_j$ is the transpose of $\dot{x}_i$). Thus, since the matrix A_{ij} is expressed in terms of α and β, these parameters are dynamically monitored (as control parameters) during the optimization process in order to achieve a positive definite damped mass matrix A_{ij}. A positive definite A_{ij} leads to $dE_T/dt = \dot{E}_T(x_i) \leq 0$ according to (15). Thus, according to the Lyapunov theorem (for global stability), the mathematical model in (10a) and (10b) is stable and, consequently, the NAOP SPP solver platform for shortest path detection always converges to the exact solution (i.e., the desired shortest path) during the optimization process.

Step 4. The fourth step is convergence monitoring and control. During our various numerical simulations, the condition in (15) is being coded (algorithmically) into a feedback loop in order to dynamically monitor and control the convergence properties of the NAOP platform for shortest path detection as illustrated in the flowchart presented in Figure 3. The values of the parameters α and β are evaluated in order to guarantee that the condition $\dot{E}_T(x_i) \leq 0$ is fulfilled for all $\dot{x}_i$ and $\dot{x}_j$. When this condition is observed/realized, the NAOP optimization platform always converges to the exact solution (revealing the exact/desired shortest path).

Comment on the Convergence Analysis. The advantage of considering the convergence/stability analysis is justified by the fact that a fundamental analytical relationship is derived under which straightforward convergence to the exact solution (i.e., exact shortest path) of the NAOP model (in (10a) and (10b)) for solving shortest path problems (SPP) is always observed regardless of the following key metrics: (a) the graph structure (directed, undirected, and/or mix of directed and undirected edges), (b) the graph size, (c) the costs of edges (high or small costs values), (d) the sign of costs (negative, positive, and/or mix of positive and negative costs), and (e) the magnitude of the graph. This latter metric is related to the scalability issue. This issue can be systematically tackled using the key analytical relationship or expression in (15). This expression can be appropriately used to ensure

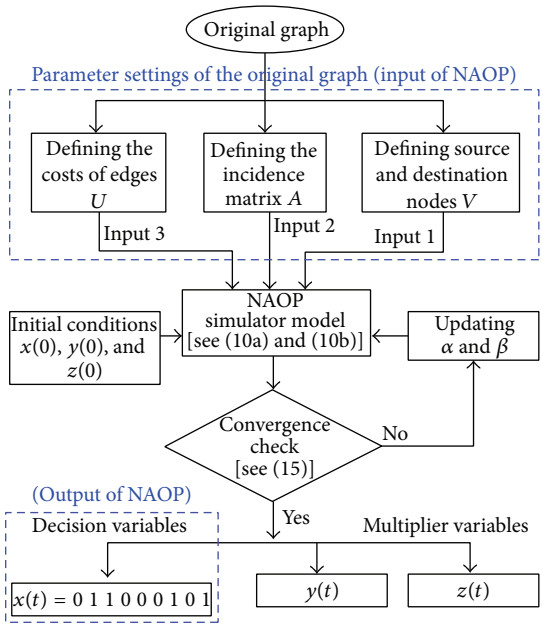

Figure 3: Flowchart revealing the functioning principle of the NAOP simulator for shortest path detection in dynamically reconfigurable graphs. This flowchart is valid for all graphs. The rule for updating alpha and beta is defined according to the expression in (15) (i.e., $\dot{E}_T(x_i) \leq 0$).

fast convergence even in case of graphs with huge amount of nodes. Specifically, it is demonstrated that the parameters α and β can be monitored (or varied) systematically and that a judicious choice of these parameters ensures fast convergence to the desired/exact solution (i.e., exact shortest path).

3.4. Flowchart of the NAOP Concept for Solving Shortest Path Problems (SPP) in Dynamically Reconfigurable Graphs. The aim of this subsection is to provide a full understanding of the functioning principle of the NAOP simulator for shortest path detection. The flowchart of the functioning principle of the NAOP simulator is proposed in Figure 3. The NAOP simulator is modeled by (10a), (10b), and (15). The known parameters A, V, and U are used as coefficients of (10a) and (10b). These parameters are obtained from the graph under investigation. The parameters α and β are dynamically monitored in order to ensure the condition of convergence expressed by $dE_T/dt = \dot{E}_T(x_i) \leq 0$ for all state variables x_i. This condition is controlled through a feedback loop defined by (15). Thus, expression (15) is essential to ensure convergence of the NAOP simulator to the exact shortest path.

Thus, for a given graph under investigation the art consists of collecting the input parameters A, V, and U which are further used by the NAOP simulator modelled by (10a), (10b), and (15). Mathematical expressions (10a), (10b), and (15) constitute a coupled mathematical model of the NAOP simulator. The coupled model is applicable to all types of graphs independently of their topology/structure, size, magnitude, and so forth.

For any given new graph, the corresponding parameters A, V, and U are determined. Further, according to the new values of A, V, and U, an update of the control parameters α and β may be performed if necessary (according to the flowchart in Figure 3). The update of the control parameters α and β allows fast convergence of the NAOP SPP solver concept to the exact shortest path. This interesting result is of necessary importance when considering graph scalability analysis since the control parameters α and β could be used to guarantee fast convergence to the exact shortest path, specifically when investigating shortest path problems (SPP) in graphs of huge magnitude and/or size.

During the optimization process (see Figures 2 and 3), the system modelled by (10a), (10b), and (15) converges to an equilibrium point (i.e., fixed point). At equilibrium point (i.e., final state (see Figure 2)), the state variables denoted by x_i are solutions of (10a) and (10b). These solutions reveal the edges belonging to the shortest path ($x_i = 1$) and those that do not belong to the shortest path ($x_i = 0$) (see Output of NAOP in Figure 3). The initial conditions $x(0)$, $y(0)$, and $z(0)$ (see Figure 3) are randomly selected. Interesting to be underscored here is that the values of initial conditions do not affect the convergence properties (robustness) or the exact shortest path detection (accuracy). This conclusion underscores the robustness of the NAOP SPP solver concept developed.

The next section will be concerned with a demonstration of the straightforward application of the NAOP SPP simulator model to several graph examples. Benchmarking of the NAOP SPP solver concept developed in this paper is further conducted, leading to a comparison of the results obtained (using NAOP) with the results provided by two well-known and proven efficient concepts for SP detection selected from the literature. These last-named concepts are the dynamic neural network concept developed in [42] (benchmarking 1) and the dependent variable Hopfield neural network developed in [41] (benchmarking 2). Examples of graphs published in [41, 42] are considered, specifically (a) graphs with unidirectional edges, (b) graphs with positive edges, (c) graphs with mixed positive and negative costs of edges, and so forth. All these examples have been considered either in [41] or in [42] and are selected here once again for benchmarking purposes.

3.5. General Algorithm for Determining the Fundamental Parameters (A, U, and V) of the Original Graph. This subsection proposes the algorithm (in the form of Matlab subcodes) for solving shortest path problems. The proposed algorithm is developed according to the analytical concept developed, which has led to the derivation of the general mathematical modelling framework in (10a) and (10b). The three steps below summarize the general procedure when dealing with a given graph:

(i) The first step consists of determining the fundamental parameters A, V, and U of a given graph under investigation. These parameters are the parameter settings of the mathematical model of the shortest path solver (see (10a) and (10b)).

(ii) The second step consists of inserting A, V, and U in the solver given in (10a) and (10b).

The algorithm can be depicted in the following pseudocode format:

BEGIN

(1) Identify the original graph under investigation:

(a) Extract/read the incidence matrix (A) of the graph.
(b) Choose the source and destination nodes (V).
(c) Assign values to edges (U).

(2) Use the parameters A, V, and U as coefficients of the NAOP model in (10a) and (10b):

(a) Define the size of A.
(b) Define the dimension of V.
(c) Define the dimension of U.

(3) Solve (10a) and (10b) under the convergence condition $Q \geq 0$ and read the shortest path (SP) as solution $x(t)$ of (10a) and (10b):

(a) Define (10a) and (10b) as a function.
(b) Define the order of (10a) and (10b).
(c) Read A, V, and U as coefficients of (10a) and (10b).
(d) Define the state variables of (10a) and (10b).
(e) Define initial conditions on the state variables of (10a) and (10b).
(f) Define a range (or a window) in which the control parameters α and β are dynamically monitored.
(g) Evaluate the condition for convergence of NAOP:

(i) Declare the variables $\dot{x}_i$, $\dot{x}_j$, and A_{ij}.
(ii) Write a loop to evaluate the value of $Q = \dot{x}_i A_{ij} \dot{x}_j$:
(A) If $Q \geq 0$, read the solutions $x(t)$ of (10a) and (10b).
(B) If $Q < 0$, tune/monitor α, β dynamically until $Q \geq 0$ and read the solutions $x(t)$.

(h) Read/depict the SP through solutions $x(t)$ of (10a) and (10b):

(A) If $x(t) = 1$, select the corresponding edges as belonging to the SP.
(B) If $x(t) = 0$, ignore the corresponding edges.
(C) Display the SP solution involving all edges belonging to the SP (as depicted above).

END

4. Application of the NAOP Spp Solver Concept to Shortest Path Detection: Benchmarking with Cases of Selected Graph Examples in [41, 42]

4.1. Simulation Results and Benchmarking with [42]. Reference [42] presents a concept based on dynamic neural networks (DNN) for solving shortest path problems in graphs with various weight values or costs of edges (e.g., positive, negative, and mix of positive and negative costs). It is demonstrated that the DNN concept undergoes a transient behavior before converging to the exact shortest path. It is also demonstrated in [42] that the concept developed (i.e., the DNN) does not converge under some specific parameter settings (e.g., initial condition). Our aim in this subsection is to demonstrate that the NAOP SPP developed in this paper can overcome the limitations of the concept developed in [42]. Specifically, we demonstrate the possibility of shortening or cancelling the transient phase duration using NAOP; we also demonstrate that the NAOP concept converges when considering the case in which the concept in [42] fails to converge. These two achievements/results can be considered to underscore the efficiency of the NAOP SPP solver concept developed in this paper.

The simulation is performed under the same parameter settings as those in [42]. These parameters are (a) graph topology, (b) source-destination pairs, (c) costs of edges, and (d) initial values for the respective ODE models. Our aim here is to demonstrate the effectiveness and efficiency of the NAOP based SPP solver concept especially in light of a direct comparison with the concept presented in [42]. The effectiveness is demonstrated by the capability of always robustly detecting the exact and truly best shortest path in each of the application examples considered in [42]. Further, the efficiency is expressed by the extreme fast convergence of the NAOP SPP solver concept to the exact shortest path when compared with the results in [42]. The metrics for comparison (benchmarking) considered here are the robustness and the simulation time, this last correlating with the number of iterations required to achieve convergence.

4.1.1. Application Example 1 (Taken from [42]). In this application example, the graph considered in "example 5.2" from [42] presents the case of a directed graph of magnitude 4 and size 5 (see Figure 4). Edges in Figure 4 are made up of a mix of positive and negative weight values (costs).

According to the theory developed in Section 4, the parameters A, U, and V of (10a) and (10b) corresponding to the graph topology in Figure 4 are defined as follows:

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 \\ 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & -1 \end{bmatrix}; \quad (16)$$

$$U = [2, -1, -4, 3, -6]^T;$$

$$V = [1, 0, 0, -1]^T.$$

Table 1: Benchmarking results for application Example 1, Case 1.

Concept	Exact shortest path	Convergence	Convergence starts at (t)
Concept in [42]	$x_{12} \rightarrow x_{23} \rightarrow x_{34}$	Yes	≈50
NAOP concept	$x_{12} \rightarrow x_{23} \rightarrow x_{34}$	Yes	≈2

Table 2: Benchmarking results for application Example 1, Case 2.

Concept	Exact shortest path	Convergence	Convergence starts at (t)
Concept in [42]	No detection	No convergence	No convergence
NAOP concept	$x_{12} \rightarrow x_{23} \rightarrow x_{34}$	Yes	≈2

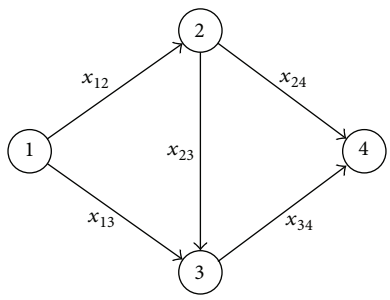

Figure 4: Graph under investigation for the application example denoted by Example 5.2 in [42].

Case 1 (convergence behavior with respect to both concepts). In this case, the initial values are chosen according to [42]. These values are $(\vec{x}(0), \vec{\lambda}(0), \vec{\gamma}(0)) = [-1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1]^T$. The results obtained by the concept in [42] and the results obtained by the NAOP SPP solver simulator are shown in Figures 5 and 6, respectively, for the graph topology in Figure 4.

The results in Figure 5 (this figure is provided by [42]) show the temporal evolution of the state variables $x_{ij}(t)$ in the simulation window $[0, 70]$. These state variables $x_{ij}(t)$ undergo a transient behavior before converging to their equilibrium point "0" or "1." In Figure 5(a), the state variables $x_{12}(t)$, $x_{23}(t)$, and $x_{34}(t)$ converge to "1." This convergence to "1" expresses the belonging of the edges x_{12}, x_{23}, and x_{34} to the shortest path, whereas in Figure 5(b) the state variables $x_{13}(t)$ and $x_{24}(t)$ converge to "0" which explains the notion that the edges x_{13} and x_{24} do not belong to shortest path.

Figure 6 presents the results obtained by our NAOP SPP solver concept. These results confirm those obtained in Figure 5 (i.e., $x_{12}(t) = x_{23}(t) = x_{34}(t) = 1$ and $x_{13}(t) = x_{24}(t) = 0$).

Hence, both concepts detect the exact shortest path. However, the significant difference between both concepts is the duration of the transient phase within the simulation window $[0, 70]$. This metric is considered to deduce the time needed by each concept to achieve the convergence of all state variables. In Figure 5 (this figure is provided by [42]), the transient phase is observed in the simulation window $[0, 50]$; hence, the convergence of state variables to binary variables "0" and "1" starts at $t \approx 50$, whereas in Figure 6 the results provided by the NAOP SPP solver show that the state variables undergo a very brief transient phase before converging to binary variables at $t \approx 2$.

Thus, this first benchmarking example clearly underscores (see Table 1) the fast convergence of the NAOP SPP solver concept proposed in this paper.

Case 2 (divergent behavior with respect to [42]). In a second example case, the initial values are taken as $(\vec{x}(0), \vec{\lambda}(0), \vec{\gamma}(0)) = [-7, 6, -5, 4, -3, 2, -1, 0, 1, -2, 3, -4, 5, -6]^T$ as mentioned in [42] for the application example given here in Figure 4. It is clearly reported in [42] that under these initial values the DNN concept developed in [42] does not converge. The aim of this comparison is to show that, under the same initial values, the NAOP concept converges to the exact shortest path. The results from [42] and the results from solving (10a) and (10b) (i.e., the mathematical model of NAOP) are shown, respectively, in Figures 7 and 8.

The results in Figure 7 (this figure is provided by [42]) show that the concept in [42] does not converge to binary values "0" and "1" for the considered initial values $[-7, 6, -5, 4, -3, 2, -1, 0, 1, -2, 3, -4, 5, 6]^T$. In contrast, the results provided by the NAOP SPP solver concept (see Figure 8) clearly show the robust convergence behavior for the same initial values. This conclusion also underscores the robustness of the NAOP based concept to changes in initial values (see conclusion in Table 2).

4.1.2. Application Example 2 (Taken from [42]). Our focus on this application example is justified by (a) the mix of positive and negative costs of edges, (b) the mix of unidirectional and bidirectional connections between nodes, and (c) the direct connections between all nodes. In this application example, the graph considered is the one given in Example 5.3 of [42]; it presents the case of a directed graph of magnitude 4 and size 7; see Figure 9.

According to the theory developed in Section 3, the parameters A, U, and V of (10a) and (10b) corresponding to the graph topology in Figure 9 are defined as follows:

$$A = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & -1 & 0 \\ -1 & 1 & 0 & -1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 1 & 1 \end{bmatrix}; \tag{17}$$

$$U = [2, -3, 6, -7, 3, 4, 5]^T;$$

$$V = [1, 0, 0, -1]^T.$$

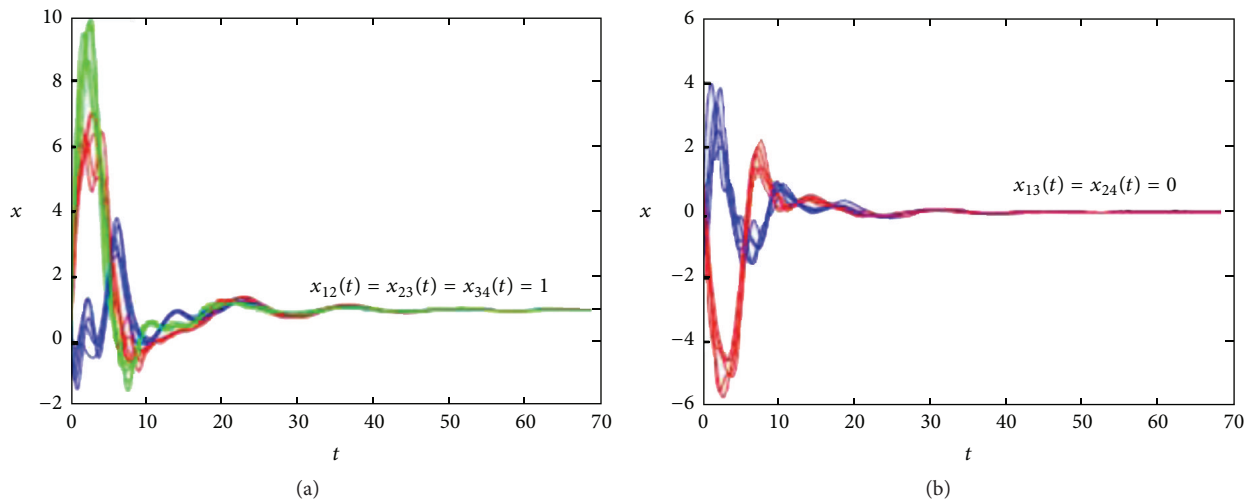

Figure 5: (a) Transient behavior of x_{12}, x_{23}, and x_{34} (edges belonging to the SPP) and (b) x_{13} and x_{24} (edges which do not belong to the SP). *These plots are provided by* [42] for the application example given in Figure 4 with the initial values $[-1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1]^T$.

Table 3: Benchmarking results for application Example 2.

Concept	Exact shortest path	Convergence	Convergence starts at (t)
Concept in [42]	$x_{12} \rightarrow x_{24}$	Yes	≈30
NAOP concept	$x_{12} \rightarrow x_{24}$	Yes	≈3

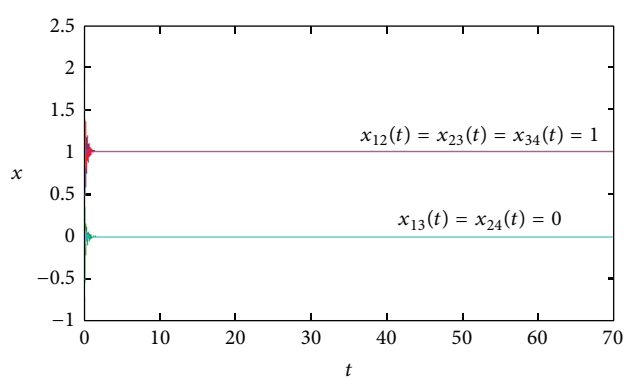

Figure 6: Transient behavior of x_{12}, x_{23}, and x_{34} (edges belonging to the SPP) and x_{13} and x_{24} (edges which do not belong to the SP). *This plot is obtained as a solution of our NAOP SPP solver concept expressed in (10a) and (10b)* for the application example given in Figure 4 with the same initial values as in Figure 5.

Table 3 just summarizes what is expressed in Figures 10 and 11. It is evident and confirmed once again that the NAOP based SPP solver is very robust and very much faster than the DNN concept developed in [42].

4.2. Simulation Results and Benchmarking with [41]. The Dependent Variables Hopfield Neural Network (DVHNN) is considered here because it is one of the leading traditional neural network (NN) concepts, which have been applied for solving shortest path problems (SPP). This promising concept for shortest path detection is extensively developed in [41]. However, some inherent limitations of this concept are pointed out by the authors of [41] such as (a) the lack of reconfigurability (this is explained by the fact that when changes occur in the graph topology (e.g., change of source or destination nodes, suppression of an edge belonging to the graph, and suppression of a node), new training (i.e., retraining) is needed in order to derive the new parameters of the DVHNN corresponding to the changes observed); (b) the accuracy of the DVHNN which deteriorates when the weights values (or costs of edges) become very high [41]; (c) the DVHNN which does not converge in case several paths (at least two paths) with identical total cost equal to the minimum cost (i.e., several equivalent shortest paths possibilities) exist in the graph for a given source-destination nodes pair; (d) lack of a systematic method (e.g., an analytical framework) that can be used to predict and control the convergence properties of the DVHNN concept through variation (or monitoring) of its two fundamental parameters μ_1 (this parameter is used in [41] for controlling convergence of the DVHNN) and μ_2 (this parameter is used in [41] for controlling accuracy, i.e., the solution quality provided by the DVHNN). The authors of [41] have further pointed out that the development of a systematic concept (e.g., analytical concept) that could be used to efficiently predict and control both convergence and accuracy of the DVHNN is of high importance. The current addressing of both convergence and accuracy of the DVHNN concept in [41] is essentially based on a trial and error process (i.e., a guess and check process). This justifies the importance of developing a systematic

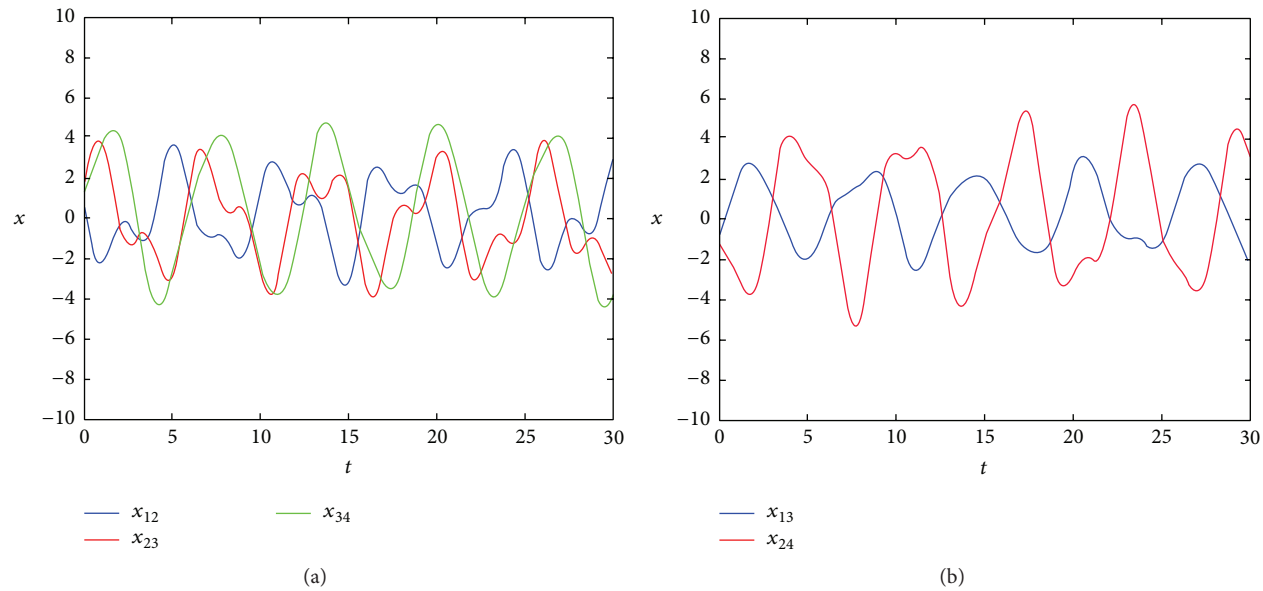

FIGURE 7: (a) Divergent behaviors of x_{12}, x_{23}, and x_{34} and (b) x_{13} and x_{24}. *These plots are provided by* [42] with the initial values $[-7, 6, -5, 4, -3, 2, -1, 0, 1, -2, 3, -4, 5, -6]^T$ for the application example given in Figure 4.

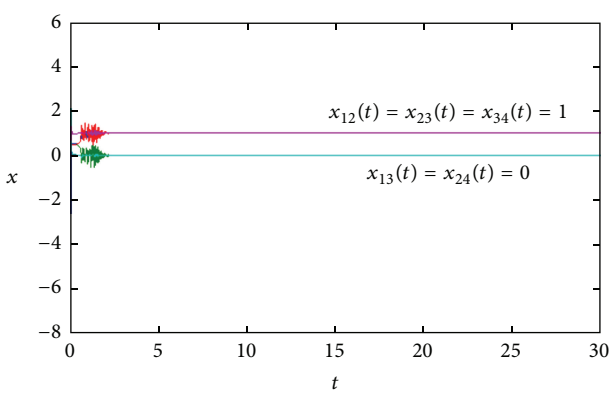

FIGURE 8: Transient behaviors of x_{12}, x_{23}, and x_{34} (edges belonging to the SP) and x_{13} and x_{24} (edges which do not belong to the SP). *This plot is obtained as a solution of (10a) and (10b)* for the application example in Figure 4 under the same initial values as in Figure 7 coming from reference to [42].

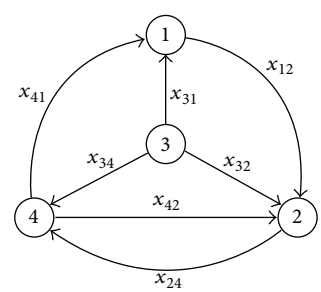

FIGURE 9: Graph under investigation for the application example denoted by 5.3 in [42].

concept or framework for addressing this underscored control and monitoring need of both convergence and accuracy. (e) Using the DVHNN, a tradeoff exists between convergence time and solution quality [41]. It is demonstrated in [41] that an improvement of the convergence time degrades the accuracy of the DVHNN concept and vice versa. The dependence between convergence and accuracy is addressed through monitoring of the ratio (μ_1/μ_2). However, even this monitoring of the ratio is also based on a trial and error process (see [41]).

Our aim in this subsection is to demonstrate that the NAOP concept developed can solve the above unsolved problems by the DVHNN. Specifically, we want to demonstrate that the NAOP SPP concept developed does not need retraining in case the abovementioned changes could occur in the graph topology. We also want to demonstrate that the accuracy of the NAOP SPP concept is not affected by the values of the costs assigned to edges (e.g., high or small costs of edges). Finally, we want to propose a systematic analytical concept that can be efficiently used to predict and control convergence of the NAOP SPP simulator to the exact (or desired) shortest path independently of the graph topology, size, magnitude, and so forth. Thus, the concept developed in this work could be viewed as a concrete prototype (i.e., model or framework) to overcome some limitations encountered by the DVHNN concept in [41].

Benchmarking is further considered between the concept developed in this work (i.e., NAOP SPP) and the concept developed in [41] (i.e., DVHNN). The performance metrics used for benchmarking are the computing performance (i.e., simulation time) and the robustness (i.e., convergence and correctness). The benchmarking is conducted by considering the graph example in Figure 12, which is considered in [41]. According to the theory developed in Section 3,

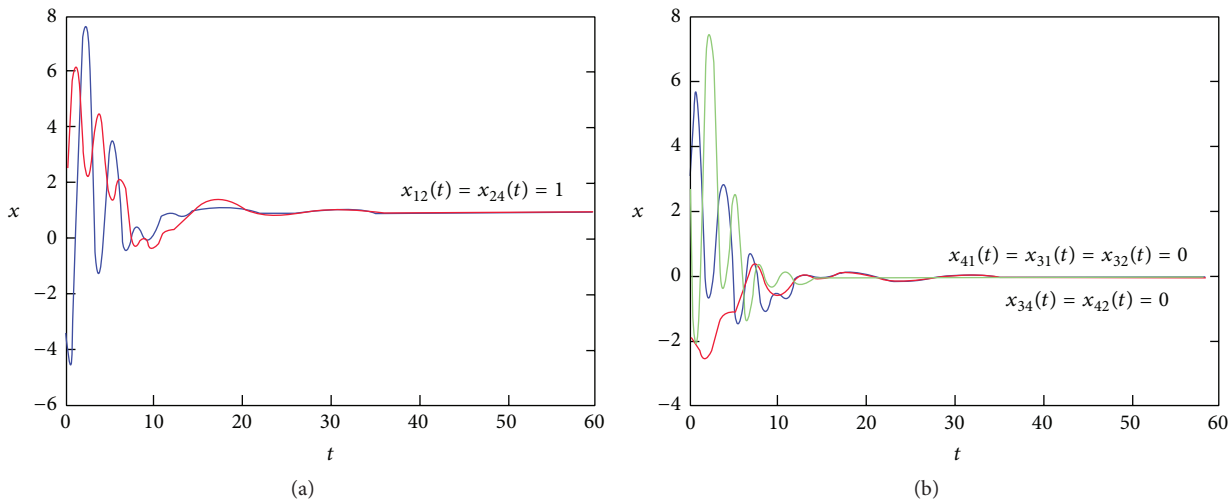

FIGURE 10: (a) Transient behaviors of x_{12} and x_{24} (edges belonging to the SPP) and (b) x_{31}, x_{32}, x_{34}, x_{41}, and x_{42} (edges which do not belong to the SP). *These plots are provided by* [42] with the initial values $\vec{x}(0) = [1, -1, 2, -2, 3, -3, 4]^T$ for the application example given in Figure 9.

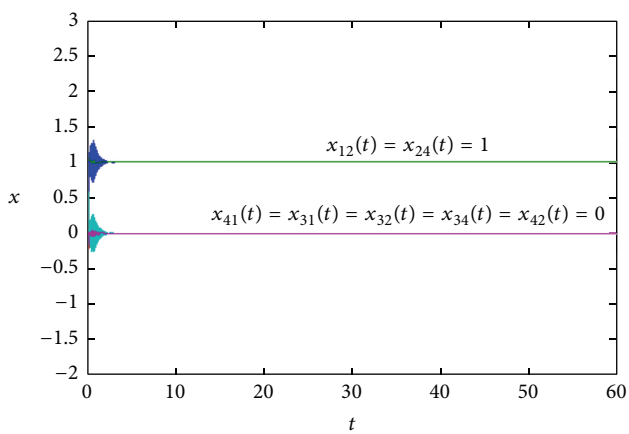

FIGURE 11: Transient behaviors of x_{12} and x_{24} (edges belonging to the SP) and x_{31}, x_{32}, x_{34}, x_{41}, and x_{42} (edges which do not belong to the SP). *This plot is obtained as a solution of (10a) and (10b)* for the application example given in Figure 9 under the same parameter settings as in Figure 10 with reference to [42].

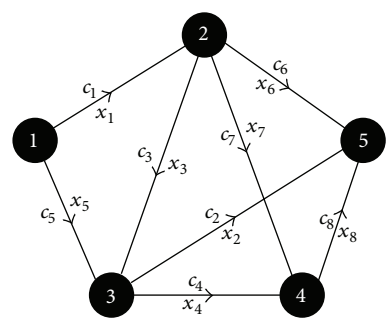

FIGURE 12: Topology of a directed graph of magnitude 5 and size 8. Both costs and states of edges are denoted, respectively, by c_i and x_i (this figure is considered in [41]).

the parameters A, U, and V of (10a) and (10b) corresponding to the graph topology in Figure 12 are defined as follows:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & -1 & 1 \\ 0 & -1 & 0 & 0 & 0 & -1 & 0 & -1 \end{bmatrix},$$
$$U = [c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8]^T, \quad V = [\delta_1, \delta_2, \delta_3, \delta_4, \delta_5]^T. \tag{18}$$

The fundamental parameters A, U, and V of the graph in Figure 12 are expressed in a general form, which allows the possibility of dynamically changing the source-destination pair and also the costs of edges. This shows the good reconfigurability potential of the NAOP SPP solver since the parameters A, U, and V are used as external commands (i.e., inputs) to the NAOP SPP concept.

The core concept of the DVHNN for finding SP is based on the derivation of Kirchhoff's constraints. These constraints express the dependent variables/neurons in terms of independent variables/neurons. Reference [41] proposes an algorithm for the selection of dependent variables. However, simple selection of the dependent variables can be performed by choosing spanning walk/path from source (s) to destination (t). This is walk involving all nodes of the graph. The state variables involved in the spanning walk/path are dependent variables/neurons while the other variables are independent neurons. Full details of the DVHNN concept are provided in [41].

It is important to mention that, according to the DVHNN concept, the dependent and independent variables are defined for a specific choice of source-destination pair (lack of reconfigurability). Due to the nonreconfigurability

Table 4: Results of the shortest path detection (in Figure 12) using the NAOP-simulator and Benchmarking between the NAOP paradigm and the DVHNN concept. A scenario corresponds to a specific choice of source-destination pair.

From source to destination $s \to t$	Shortest path results using NAOP-simulator		NAOP versus DVHNN			
			Sim. time (Tsim)		Convergence	
	Edges in the shortest path	Total cost of the path	NAOP (ms)	VDHNN (ms)[1]	NAOP	DVHNN
Small weights values: the cost of an edge with index "i" is "$C_i = 0.01 * i$"						
$1 \to 2$	x_1	0.1	0.45	48.4	Yes	Yes
$1 \to 3$	x_1, x_3	0.4	1.73	114	Yes	Yes
$1 \to 4$	x_1, x_3, and x_4	0.8	42.2	—	Yes	No
$2 \to 3$	x_3	0.3	1	77.3	Yes	Yes
$2 \to 4$	x_7	0.7	37.9	—	Yes	No
$2 \to 5$	x_2, x_3	0.5	7.6	89.2	Yes	Yes
$3 \to 5$	x_2	0.2	0.94	68.9	Yes	Yes
High weights values: the cost of an edge with index "i" is "$C_i = 1000 * i$"						
$1 \to 2$	x_1	1000	0.54	—	Yes	No
$1 \to 3$	x_1, x_3	4000	0.77	—	Yes	No
$1 \to 4$	x_1, x_3, and x_4	8000	70.3	—	Yes	No
$2 \to 3$	x_3	3000	0.22	—	Yes	No
$2 \to 4$	x_7	7000	85.4	—	Yes	No
$2 \to 5$	x_2, x_3	5000	0.19	—	Yes	No
$3 \to 5$	x_2	2000	0.16	—	Yes	No

[1]In this paper, the concepts have been all implemented in Matlab on a standard PC.

of the DVHNN concept, we have performed for each of the scenarios (*remark*: each scenario is defined by a specific source-destination pair) envisaged in Table 4 additional specific training of the DVHNN in order to determine the suitable system's parameters. It is important to notice that the simulation times in Table 4 (for the DVHNN) correspond only to the duration of the usage phases (training not considered). The duration of the training phase of DVHNN (for each of the scenarios listed in Table 4) was not constant; however, it was always greater than 5 seconds.

The results (in Table 4) show that the NAOP concept developed in this work leads to a better computing performance than the DVHNN concept (for both small and large values of the costs of edges). Further, the computing performance of the DVHNN degrades (i.e., becomes the worst) with increasing costs of edges. When the costs of edges are very high (e.g., greater than thousand (1000)), the DVHNN cannot converge. On the other hand, the accuracy of the NAOP SPP concept is preserved even for very high costs of edges.

Regarding the test for robustness, we have considered various critical and challenging cases where several paths (from source to destination) are depicted with the same minimum cost (i.e., paths with equivalent/identical costs equal to the minimum cost). As it appears in Table 4, the DVHNN cannot efficiently analyze some of the various scenarios envisaged. Further, the robustness of the DVHNN degrades with increasing values of the costs of edges. This is observed (in Table 4) by cases of nonconvergence (lack of convergence) of the VDHNN for shortest path finding. However, the results in Table 4 reveal the good robustness of the NAOP SPP simulator for shortest path finding. This robustness is characterized by successful and straightforward convergence to a routing path (i.e., a correct path, which corresponds to the path having the minimum cost) for the sample scenarios envisaged.

In Table 5, we do summarize the essential comparison between DVHNN and NAOP SPP concept while using the most relevant criteria. It is clear from the table that the NAOP SPP concept can overcome some limitations of the DVHNN.

5. Conclusion

The work presented in this paper has developed and validated a general theoretical concept based on nonlinear adaptive optimization (NAOP) for the efficient and robust solving of SPP problem cases in reconfigurable network graphs. The developed concept has been demonstrated as being capable of efficiently handling shortest path problems in both directed and undirected graphs, even in the case of large values of the edge costs. Further, it has been demonstrated that the developed concept can efficiently detect SP even in the particular cases where there exist several paths (SP) with identical minimum total cost. This result does significantly contribute to the enrichment of the relevant state of the art regarding shortest path problems since the traditional SP concepts and algorithms (e.g., the traditional neural network architectures, PSO, GA, etc.) cannot efficiently tackle

Table 5: Critical comparison of DVHNN and NAOP based concepts for SP determination.

Comparison criteria	DVHNN based SP determination	NAOP based SP determination
External reconfigurability regarding dynamic change of edges' weights	Yes	Yes
External reconfigurability regarding dynamic change of $s \rightarrow t$ pairs	No	Yes
External reconfigurability regarding dynamic change of network topology	No	*In principle yes*, although this particular aspect has not been considered in this paper. It will be addressed in a future paper
Ability to cope with negative edges' weights	No	Yes
Ability to cope with nonlinear path's weights	No	*In principle yes*, although this particular aspect has not been considered in this paper. It will be addressed in a future paper
Reliability of the convergence: this stands for valid and successful convergence; otherwise, it is a failure	No Many failure cases have been observed; this is especially the case when high values of weights are present	Yes
Computational speed[2]	Basically good if compared to the other NN based SP determination concepts	Very good (a difference of 1 to 2 order of magnitude better than DVHNN has been observed here)
Memory consumption need	High Because of the retraining need for each $s \rightarrow t$ pair	Very low

[2]In this paper, the concepts have been all implemented in Matlab on a standard PC.

such complexity or always ensure convergence [47]. The efficiency in this context is related to core performance criteria such as computing speed, robustness, and convergence. One of the most important conclusions to be underscored in this paper is that the NAOP based SPP solver concept developed can also be used, after appropriate adaptations, to efficiently model and handle further graph theoretical problems (e.g., traveler salesman problem, minimum spanning tree problem, and vehicle routing problem). The latter cited problems are being addressed in some subsequent works.

As proof of concept, extensive benchmarking has been carried out. The performance of the NAOP SPP concept developed has been sufficiently compared with the performances of two concepts taken from the literature, namely, the DVHNN in [41] and the DNN in [42]. The outcome of the benchmarking has confirmed the effectiveness of the novel NAOP based concept developed to efficiently handle (or tackle) SP problems.

A series of ongoing works under consideration do relate amongst others to (a) the extension of the concept developed (in this work) to the case where the cost of a path is a nonlinear function of the arc weights (e.g., for cases where the driving energy on a path must be minimized) and (b) the extension of the concept developed (in this work) to the case of stochastic graphs. These graphs are characterized by arc weights that are stochastic processes. Thus, extending the concept developed in this work to both nonlinear and stochastic graphs models is possible. Regarding the latter ones, this can be achieved by considering the key parameters, that is, the kth moments of the distributions characterizing the stochastic dynamics of each edge of the network/graph. The extension of the concept developed in this work to stochastic graphs will lead to the mathematical modeling of various stochastics-prone systems, for example, of sensor data with straightforward interesting applications in areas like intelligent transportation systems and many others.

Furthermore, a series of extremely interesting problem cases (for practical applications) for which the core concept of this paper will be extended in future works are (a) network reliability problems in case of both dependent and independent edges; (b) optimal flow problems under multiple source-destination scenarios; (c) optimal flow problems under multiple source-destination scenarios and under stochastic arc's weights conditions; (d) traveler salesman problem (TSP); (e) TSP with stochastic edge's weights; (f) resource allocation problems under stochastic conditions; (g) vehicle routing problems (VRP) under stochastic conditions; (h) scheduling problems under stochastic conditions, and so forth. All these problems are extremely challenging and do face a huge computational complexity that can be significantly alleviated

by the involvement of the novel concept developed in this work.

Appendix

This appendix provides full detail of the stability analysis carried out in order to guarantee fast convergence (to the exact solution) of the NAOP platform developed in (10a) and (10b) for solving shortest path problems (SPP). The analysis is systematically conducted and this led to the derivation of analytical conditions (expressed by a formula; see (A.38)) under which the NAOP system model in (10a) and (10b) always provides converging solutions. As already mentioned, these solutions, for example, the decision variables (or solutions x_i), express the exact/desired shortest path.

The attention devoted to the convergence/stability analysis is justified by the fact that a fundamental analytical relationship is derived under which straightforward convergence to the exact solution (i.e., exact shortest path) of the NAOP model (in (10a) and (10b)) for solving shortest path problems (SPP) is always observed regardless of the following key metrics: (a) the graph structure (directed, undirected, and/or mix of directed and undirected edges), (b) the graph size, (c) the costs of edges (high or small costs values), (d) the sign of costs (negative, positive, and/or mix of positive and negative costs), and (e) the magnitude of the graph. This latter metric is related to the scalability issue. This issue is systematically addressed here and a key analytical relationship is established analytically to ensure fast convergence even in case of graphs with huge amount of nodes. Specifically it is demonstrated that the parameters α and β can be monitored (or varied) systematically and that a judicious choice of these parameters ensures fast convergence to the desired/exact solution (i.e., shortest path).

The functions $f(\vec{x})$, $g_m(\vec{x})$, and $h_n(\vec{x})$ are expressed in terms of $\vec{x} = [x_1, x_2, \ldots, x_N]^T \in \mathfrak{R}$. Therefore, the total differentiation of these functions is expressed as follows:

$$\begin{aligned} df &= \left(\frac{\partial f}{\partial x_1}\right) dx_1 + \cdots + \left(\frac{\partial f}{\partial x_N}\right) dx_N, \\ dg_m &= \left(\frac{\partial g_m}{\partial x_1}\right) dx_1 + \cdots + \left(\frac{\partial g_m}{\partial x_N}\right) dx_N, \\ dh_n &= \left(\frac{\partial h_n}{\partial x_1}\right) dx_1 + \cdots + \left(\frac{\partial h_n}{\partial x_N}\right) dx_N. \end{aligned} \tag{A.1}$$

Equations (A.1) can be written in the following forms:

$$\begin{aligned} \frac{df}{dt} &= \left(\frac{\partial f}{\partial x_1}\right) \dot{x}_1 + \cdots + \left(\frac{\partial f}{\partial x_N}\right) \dot{x}_N, \\ \frac{dg_m}{dt} &= \left(\frac{\partial g_m}{\partial x_1}\right) \dot{x}_1 + \cdots + \left(\frac{\partial g_m}{\partial x_N}\right) \dot{x}_N, \\ \frac{dh_n}{dt} &= \left(\frac{\partial h_n}{\partial x_1}\right) \dot{x}_1 + \cdots + \left(\frac{\partial h_n}{\partial x_N}\right) \dot{x}_N, \end{aligned} \tag{A.2}$$

where the single overdot $\dot{(\ )} = d(\ )/dt$ denotes the first derivative. Thus, (A.2) can be expressed in the following compact form:

$$\frac{df}{dt} = \sum_{j=1}^{N} \left[\left(\frac{\partial f}{\partial x_j}\right) \dot{x}_j \right], \tag{A.3}$$

$$\frac{dg_m}{dt} = \sum_{j=1}^{N} \left[\left(\frac{\partial g_m}{\partial x_j}\right) \dot{x}_j \right], \tag{A.4}$$

$$\frac{dh_n}{dt} = \sum_{j=1}^{N} \left[\left(\frac{\partial h_n}{\partial x_j}\right) \dot{x}_j \right]. \tag{A.5}$$

The Lagrange function $\widetilde{L}(\vec{x}, \vec{\lambda}, \vec{\gamma})$ in (4) can be substituted into (5a) to obtain the following expression:

$$\begin{aligned} \dot{x}_i = -\alpha \Bigg[& \frac{\partial f(\vec{x})}{\partial x_i} + \frac{\partial}{\partial x_i} \left(\sum_{m=1}^{M} [\lambda_m g_m(\vec{x})] \right) \\ & + \frac{\partial}{\partial x_i} \left(\sum_{n=1}^{N} [\gamma_n h_n(\vec{x})] \right) \Bigg]. \end{aligned} \tag{A.6}$$

Applying the first derivative to (A.6) leads to

$$\begin{aligned} \ddot{x}_i = -\alpha \Bigg[& \frac{\partial}{\partial x_i} \left(\frac{df}{dt} \right) \\ & + \frac{\partial}{\partial x_i} \sum_{m=1}^{M} \left(\lambda_m \cdot \frac{dg_m}{dt} + g_m \cdot \frac{d\lambda_m}{dt} \right) \\ & + \frac{\partial}{\partial x_i} \sum_{n=1}^{N} \left(\gamma_n \cdot \frac{dh_n}{dt} + h_n \frac{d\gamma_n}{dt} \right) \Bigg]. \end{aligned} \tag{A.7}$$

Similarly, the Lagrange function $\widetilde{L}(\vec{x}, \vec{\lambda}, \vec{\gamma})$ in (4) can be substituted into (5b) to obtain the following expression of the first derivative of λ_m:

$$\begin{aligned} \frac{d\lambda_m}{dt} = +\beta \frac{\partial \widetilde{L}}{\partial \lambda_m} = +\beta \Bigg[& \frac{\partial f(\vec{x})}{\partial \lambda_m} \\ & + \frac{\partial}{\partial \lambda_m} \left(\sum_{m=1}^{M} [\lambda_m g_m(\vec{x})] \right) \\ & + \frac{\partial}{\partial \lambda_m} \left(\sum_{n=1}^{N} [\gamma_n h_n(\vec{x})] \right) \Bigg]. \end{aligned} \tag{A.8}$$

An important remark is that $\sum_{m=1}^{M} [\lambda_m g_m(\vec{x})]$ can be written in the following expanded form:

$$\begin{aligned} & \sum_{m=1}^{M} [\lambda_m g_m(\vec{x})] \\ & \quad = \lambda_1 g_1 + \lambda_2 g_2 + \lambda_3 g_3 + \cdots + \lambda_M g_M, \end{aligned}$$

$$\frac{\partial}{\partial \lambda_1}\left\{\sum_{m=1}^{M}\left[\lambda_m g_m(\vec{x})\right]\right\} = \frac{\partial}{\partial \lambda_1}\left\{\lambda_1 g_1 + \cdots + \lambda_M g_M\right\} = g_1,$$
$$\frac{\partial}{\partial \lambda_2}\left\{\sum_{m=1}^{M}\left[\lambda_m g_m(\vec{x})\right]\right\} = \frac{\partial}{\partial \lambda_2}\left\{\lambda_1 g_1 + \cdots + \lambda_M g_M\right\} = g_2, \tag{A.9}$$

and, finally,

$$\frac{\partial}{\partial \lambda_M}\left\{\sum_{m=1}^{M}\left[\lambda_m g_m(\vec{x})\right]\right\} = \frac{\partial}{\partial \lambda_M}\left\{\lambda_1 g_1 + \cdots + \lambda_M g_M\right\} = g_M. \tag{A.10}$$

Further,

$$\frac{\partial f(\vec{x})}{\partial \lambda_m} = 0$$
$$\frac{\partial}{\partial \lambda_m}\left(\sum_{n=1}^{N}\left[\gamma_n h_n(\vec{x})\right]\right) = 0 \tag{A.11}$$

since $f(\vec{x})$, γ_n, and $h_n(\vec{x})$ do not depend on λ_m.

Substituting (A.10) and (A.11) into (A.8) leads to the following expression:

$$\frac{d\lambda_m}{dt} = +\beta g_m. \tag{A.12}$$

Similar to the process leading to (A.12), it can be shown that

$$\frac{d\gamma_n}{dt} = +\beta h_n. \tag{A.13}$$

Substituting (A.3), (A.4), (A.5), (A.12), and (A.13) into (A.7) leads to the following mathematical expression:

$$\begin{aligned}\ddot{x}_i = -\alpha\Bigg[&\frac{\partial}{\partial x_i}\left(\sum_{j=1}^{N}\left[\left(\frac{\partial f}{\partial x_j}\right)\dot{x}_j\right]\right) + \cdots \\ &+ \frac{\partial}{\partial x_i}\left(\sum_{m=1}^{M}\left[\lambda_m \cdot \sum_{j=1}^{N}\left\{\left(\frac{\partial g_m}{\partial x_j}\right)\dot{x}_j\right\} + \beta g_m^2\right]\right) \\ &+ \cdots \\ &+ \frac{\partial}{\partial x_i}\left(\sum_{n=1}^{N}\left[\gamma_n \cdot \sum_{j=1}^{N}\left\{\left(\frac{\partial h_n}{\partial x_j}\right)\dot{x}_j\right\} + \beta h_n^2\right]\right)\Bigg].\end{aligned} \tag{A.14}$$

Expanding the summation on the indexes m and n in (A.14) leads to

$$\begin{aligned}\ddot{x}_i = -\alpha\Bigg[&\frac{\partial}{\partial x_i}\left(\sum_{j=1}^{N}\left[\left(\frac{\partial f}{\partial x_j}\right)\dot{x}_j\right]\right) + \cdots \\ &+ \frac{\partial}{\partial x_i}\left(\sum_{m=1}^{M}\left[\lambda_m \cdot \sum_{j=1}^{N}\left\{\left(\frac{\partial g_m}{\partial x_j}\right)\dot{x}_j\right\}\right]\right. \\ &\left.+ \sum_{m=1}^{M}\left[\beta g_m^2\right]\right) + \cdots \\ &+ \frac{\partial}{\partial x_i}\left(\sum_{n=1}^{N}\left[\gamma_n \cdot \sum_{j=1}^{N}\left\{\left(\frac{\partial h_n}{\partial x_j}\right)\dot{x}_j\right\}\right]\right. \\ &\left.+ \sum_{n=1}^{N}\left[\beta h_n^2\right]\right)\Bigg].\end{aligned} \tag{A.15}$$

Equation (A.15) can be rewritten as follows by permuting the summations on the indexes m and j and also n and j:

$$\begin{aligned}\ddot{x}_i = -\alpha\Bigg[&\frac{\partial}{\partial x_i}\left(\sum_{j=1}^{N}\left[\left(\frac{\partial f}{\partial x_j}\right)\dot{x}_j\right]\right) + \cdots \\ &+ \frac{\partial}{\partial x_i}\left(\sum_{j=1}^{N}\left[\sum_{m=1}^{M}\left\{\lambda_m\left(\frac{\partial g_m}{\partial x_j}\right)\dot{x}_j\right\}\right]\right. \\ &\left.+ \sum_{m=1}^{M}\left[\beta g_m^2\right]\right) + \cdots \\ &+ \frac{\partial}{\partial x_i}\left(\sum_{j=1}^{N}\left[\sum_{n=1}^{N}\left\{\gamma_n\left(\frac{\partial h_n}{\partial x_j}\right)\dot{x}_j\right\}\right]\right. \\ &\left.+ \sum_{n=1}^{N}\left[\beta h_n^2\right]\right)\Bigg].\end{aligned} \tag{A.16}$$

Expression (A.16) can be transformed as follows:

$$\begin{aligned}\ddot{x}_i = -\alpha\Bigg[&\left(\sum_{j=1}^{N}\left\{\frac{\partial}{\partial x_i}\left(\frac{\partial f}{\partial x_j}\right)\dot{x}_j\right\}\right) + \cdots \\ &+ \left(\sum_{j=1}^{N}\left\{\sum_{m=1}^{M}\left\{\lambda_m \frac{\partial}{\partial x_i}\left(\frac{\partial g_m}{\partial x_j}\right)\dot{x}_j\right\}\right\}\right.\end{aligned}$$

$$+\sum_{m=1}^{M}\frac{\partial}{\partial x_i}\left(\beta g_m^2\right)\Bigg)+\cdots$$
$$+\left(\sum_{j=1}^{N}\left\{\sum_{n=1}^{N}\left\{\gamma_n\frac{\partial}{\partial x_i}\left(\frac{\partial h_n}{\partial x_j}\right)\dot{x}_j\right\}\right\}\right.$$
$$\left.+\sum_{n=1}^{N}\frac{\partial}{\partial x_i}\left(\beta h_n^2\right)\right)\Bigg]. \tag{A.17}$$

By grouping in (A.17) the expressions involving summations on the index j, the following new expression is obtained:

$$\begin{aligned}\ddot{x}_i = -\alpha\Bigg[&\sum_{j=1}^{N}\left\{\left(\frac{\partial^2 f}{\partial x_i\partial x_j}\right)\dot{x}_j\right.\\&+\sum_{m=1}^{M}\left(\lambda_m\frac{\partial}{\partial x_i}\left(\dot{x}_j\frac{\partial g_m}{\partial x_j}\right)\right)\\&\left.+\sum_{n=1}^{N}\left(\gamma_n\frac{\partial}{\partial x_i}\left(\dot{x}_j\frac{\partial h_n}{\partial x_j}\right)\right)\right\}\Bigg]+\sum_{m=1}^{M}\frac{\partial}{\partial x_i}\left(\beta g_m^2\right)\\&+\sum_{n=1}^{N}\frac{\partial}{\partial x_i}\left(\beta h_n^2\right).\end{aligned} \tag{A.18}$$

Applying the partial derivative (in the direction x_i) to the products terms in (A.18) leads to the following expression:

$$\begin{aligned}\ddot{x}_i = -\alpha\Bigg[&\sum_{j=1}^{N}\left\{\left(\frac{\partial^2 f}{\partial x_i\partial x_j}\right)\dot{x}_j\right.\\&+\sum_{m=1}^{M}\left(\lambda_m\left(\dot{x}_j\frac{\partial^2 g_m}{\partial x_i\partial x_j}\right)+\lambda_m\left(\frac{\partial\dot{x}_j}{\partial x_i}\right)\left(\frac{\partial g_m}{\partial x_j}\right)\right)\\&\left.+\sum_{n=1}^{N}\left(\gamma_n\left(\dot{x}_j\frac{\partial^2 h_n}{\partial x_i\partial x_j}\right)+\gamma_n\left(\frac{\partial\dot{x}_j}{\partial x_i}\right)\left(\frac{\partial h_n}{\partial x_j}\right)\right)\right\}\\&+2\beta\sum_{m=1}^{M}\left(g_m\frac{\partial g_m}{\partial x_i}\right)+2\beta\sum_{n=1}^{N}\left(h_n\frac{\partial h_n}{\partial x_i}\right)\Bigg].\end{aligned} \tag{A.19}$$

Further, since the variables x_i and x_j are pairwise independent, the following expression can be derived:

$$\left(\frac{\partial\dot{x}_j}{\partial x_i}\right)=\left(\frac{\partial}{\partial x_i}\left[\frac{dx_j}{dt}\right]\right)=\left(\frac{d}{dt}\left[\frac{\partial x_j}{\partial x_i}\right]\right)=0. \tag{A.20}$$

Substituting (A.20) into (A.19) leads to the following simplified expression:

$$\begin{aligned}\ddot{x}_i = -\alpha\Bigg\{&\sum_{j=1}^{N}\left(\left[\frac{\partial^2 f}{\partial x_i\partial x_j}\right]\dot{x}_j+\dot{x}_j\sum_{m=1}^{M}\left[\lambda_m\frac{\partial^2 g_m}{\partial x_i\partial x_j}\right]\right.\\&\left.+\dot{x}_j\sum_{n=1}^{N}\left[\gamma_n\frac{\partial^2 h_n}{\partial x_i\partial x_j}\right]\right)+2\beta\left(\sum_{m=1}^{M}\left[g_m\frac{\partial g_m}{\partial x_i}\right]\right.\\&\left.+\sum_{n=1}^{N}\left[h_n\frac{\partial h_n}{\partial x_i}\right]\right)\Bigg\}.\end{aligned} \tag{A.21}$$

Expression (A.21) can be written in the following simplified form:

$$\begin{aligned}\ddot{x}_i+\Bigg\{&\sum_{j=1}^{N}\alpha\left(\left[\frac{\partial^2 f}{\partial x_i\partial x_j}\right]+\sum_{m=1}^{M}\left[\lambda_m\frac{\partial^2 g_m}{\partial x_i\partial x_j}\right]\right.\\&\left.+\sum_{n=1}^{N}\left[\gamma_n\frac{\partial^2 h_n}{\partial x_i\partial x_j}\right]\right)\dot{x}_j+2\alpha\beta\left(\sum_{m=1}^{M}\left[g_m\frac{\partial g_m}{\partial x_i}\right]\right.\\&\left.+\sum_{n=1}^{N}\left[h_n\frac{\partial h_n}{\partial x_i}\right]\right)\Bigg\}=0.\end{aligned} \tag{A.22}$$

Therefore, grouping the terms of (A.22) leads to the following fundamental expression describing the temporal evolution of the decision variables x_i (solutions of (10a) and (10b)) and outputs of the NAOP simulator in Figures 1 and 3:

$$\ddot{x}_i+\sum_{j=1}^{N}\left[A_{ij}\dot{x}_j\right]+\vec{F}=0. \tag{A.23a}$$

Equation (A.23a) is a second-order ordinary differential equation with a single and dissipative coefficient denoted by A_{ij}. This coefficient represents the damped mass matrix defined according to (A.22) as follows:

$$\begin{aligned}&A_{ij}\\&=\alpha\left[\frac{\partial^2 f}{\partial x_i\partial x_j}+\sum_{m=1}^{M}\lambda_m\frac{\partial^2 g_m}{\partial x_i\partial x_j}+\sum_{n=1}^{N}\gamma_n\frac{\partial^2 h_n}{\partial x_i\partial x_j}\right].\end{aligned} \tag{A.23b}$$

In (A.23a) $\vec{F}$ is an internal force producing the potential energy into the system. This force is expressed according to (A.22) as follows:

$$\vec{F}=2\alpha\beta\left[\sum_{m=1}^{M}g_m\frac{\partial g_m}{\partial x_i}+\sum_{n=1}^{N}h_n\frac{\partial h_n}{\partial x_i}\right]. \tag{A.23c}$$

The total energy (E_T) of the system is expressed as the sum of kinetic energy (E_C) and potential energy (E_P) by the following expression:

$$E_T=E_C+E_P. \tag{A.24}$$

According to (A.23a), (A.23b), and (A.23c), x_i is the state variable. Therefore, the energies (i.e., E_C and E_P) are expressed in terms of the state variable x_i (solution of (A.23a), (A.23b), and (A.23c)) as follows:

$$E_P(x_i) = \int F \cdot \partial x_i, \qquad E_C(x_i) = \sum_{i=1}^{N} \left[\frac{1}{2} (\dot{x}_i)^2 \right]. \tag{A.25}$$

Substituting (A.25) into (A.24) leads to the following expression of the total energy:

$$E_T(x_i) = \sum_{i=1}^{N} \left[\frac{1}{2} (\dot{x}_i)^2 \right] + \int F \cdot \partial x_i. \tag{A.26}$$

Thus, substituting (A.23c) into (A.26) leads to the following expression of the total energy:

$$E_T(x_i) = \sum_{i=1}^{N} \left[\frac{1}{2} (\dot{x}_i)^2 \right] + 2\alpha\beta \int \left[\sum_{m=1}^{M} g_m \frac{\partial g_m}{\partial x_i} + \sum_{n=1}^{N} h_n \frac{\partial h_n}{\partial x_i} \right] \partial x_i. \tag{A.27}$$

Equation (A.27) can be written in the following simplified form:

$$E_T(x_i) = \sum_{i=1}^{N} \left[\frac{1}{2} (\dot{x}_i)^2 \right] + \alpha\beta \int \left[\sum_{m=1}^{M} \frac{\partial (g_m^2)}{\partial x_i} + \sum_{n=1}^{N} \frac{\partial (h_n^2)}{\partial x_i} \right] \partial x_i. \tag{A.28}$$

In (A.28), the integral operator can be permuted with the summation operator to obtain the following expression:

$$E_T(x_i) = \sum_{i=1}^{N} \left[\frac{1}{2} (\dot{x}_i)^2 \right] + \alpha\beta \left(\left[\sum_{m=1}^{M} \int \partial (g_m^2) + \sum_{n=1}^{N} \int \partial (h_n^2) \right] \right). \tag{A.29}$$

Finally, an evaluation of the integral term in (A.29) leads to the following expression:

$$E_T(x_i) = \sum_{i=1}^{N} \left[\frac{1}{2} (\dot{x}_i)^2 \right] + \alpha\beta \left(\left[\sum_{m=1}^{M} (g_m^2) + \sum_{n=1}^{N} (h_n^2) \right] \right). \tag{A.30}$$

Using (A.30), the first derivative $\dot{E}_T(x_i) = d(E_T(x_i))/dt$ of the total energy is expressed as follows:

$$\dot{E}_T(x_i) = \sum_{i=1}^{N} [\dot{x}_i \ddot{x}_i] + 2\alpha\beta \left(\left[\sum_{m=1}^{M} \left(g_m \frac{dg_m}{dt} \right) + \sum_{n=1}^{N} \left(h_n \frac{dh_n}{dt} \right) \right] \right). \tag{A.31}$$

Using (A.4) and (A.5), the terms dg_m/dt and dh_n/dt are expressed in the direction i as follows:

$$\frac{dg_m}{dt} = \sum_{i=1}^{N} \left[\left(\frac{\partial g_m}{\partial x_i} \right) \dot{x}_i \right], \qquad \frac{dh_n}{dt} = \sum_{i=1}^{N} \left[\left(\frac{\partial h_n}{\partial x_i} \right) \dot{x}_i \right]. \tag{A.32}$$

Substituting (A.32) into (A.31) leads to the following expression of the first derivative of the total energy:

$$\dot{E}_T(x_i) = \sum_{i=1}^{N} [\dot{x}_i \ddot{x}_i] + 2\alpha\beta \sum_{m=1}^{M} g_m \left(\sum_{i=1}^{N} \left(\frac{\partial g_m}{\partial x_i} \right) \dot{x}_i \right) + 2\alpha\beta \sum_{n=1}^{N} h_n \left(\sum_{i=1}^{N} \left(\frac{\partial h_n}{\partial x_i} \right) \dot{x}_i \right). \tag{A.33}$$

The summation on the index i can be permuted with the summations on the indexes m and n, respectively. This algebraic manipulation leads to the following expression:

$$\dot{E}_T(x_i) = \sum_{i=1}^{N} [\dot{x}_i \ddot{x}_i] + 2\alpha\beta \sum_{i=1}^{N} (\dot{x}_i) \left[\sum_{m=1}^{M} g_m \frac{\partial g_m}{\partial x_i} + \sum_{n=1}^{N} h_n \frac{\partial h_n}{\partial x_i} \right]. \tag{A.34}$$

Therefore, the expression in (A.34) can be expressed in the following simplified form (by grouping the summations on the index i):

$$\dot{E}_T(x_i) = \sum_{i=1}^{N} \dot{x}_i \left(\ddot{x}_i + 2\alpha\beta \left[\sum_{m=1}^{M} g_m \frac{\partial g_m}{\partial x_i} + \sum_{n=1}^{N} h_n \frac{\partial h_n}{\partial x_i} \right] \right). \tag{A.35}$$

Using (A.23a), the following relationship is obtained:

$$-\sum_{j=1}^{N} [A_{ij} \dot{x}_j] = \ddot{x}_i + \vec{F}. \tag{A.36a}$$

Substituting (A.23c) into (A.36a) leads to the following expression:

$$-\sum_{j=1}^{N}\left[A_{ij}\dot{x}_j\right] = \left(\ddot{x}_i + 2\alpha\beta\left[\sum_{m=1}^{M} g_m \frac{\partial g_m}{\partial x_i} + \sum_{n=1}^{N} h_n \frac{\partial h_n}{\partial x_i}\right]\right). \tag{A.36b}$$

Further, substituting (A.36b) into (A.35) leads to the following expression of the first derivative of the total energy with respect to the independent variable (t):

$$\dot{E}_T(x_i) = \sum_{i=1}^{N}\left[\dot{x}_i\left(-\sum_{j=1}^{N}\left[A_{ij}\dot{x}_j\right]\right)\right]. \tag{A.37}$$

Finally, (A.37) can be expressed in the following simplified form:

$$\dot{E}_T(x_i) = -\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\dot{x}_i A_{ij}\dot{x}_j\right). \tag{A.38}$$

Equation (A.38) is the characteristic (or fundamental) equation expression used for the stability (or convergence) analysis. As already mentioned, the damped mass matrix A_{ij} depends on the parameters α and β. These parameters are dynamically chosen during the optimization process in order to guarantee a positive definite damped mass matrix A_{ij} for all state variables x_i. Thus, a positive definite damped mass matrix A_{ij} leads to $dE_T(x_i)/dt = \dot{E}_T(x_i) \leq 0$ (according to the expression in (A.38)). This condition is therefore coded (algorithmically) in order to insure convergence of the NAOP SPP simulator to the exact shortest path (according to the Lyapunov theorem for global stability analysis). When the condition $\dot{E}_T(x_i) \leq 0$ is fulfilled, the convergence of the NAOP SPP simulator to the exact shortest path is observed.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] J.-H. Kim and H. Myung, "Evolutionary programming techniques for constrained optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 2, pp. 129–140, 1997.

[2] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[3] S. Verblunsky, "On the shortest path through a number of points," *Proceedings of the American Mathematical Society*, vol. 2, no. 6, pp. 904–913, 1951.

[4] S. Kim, M. E. Lewis, and C. C. White III, "Optimal vehicle routing with real-time traffic information," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 2, pp. 178–188, 2005.

[5] S.-T. Liu and C. Kao, "Network flow problems with fuzzy arc lengths," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 1, pp. 765–769, 2004.

[6] A. R. Willms and S. X. Yang, "Real-time robot path planning via a distance-propagating dynamic system with obstacle clearance," *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, vol. 38, no. 3, pp. 884–893, 2008.

[7] C. Chenghui and S. Ferrari, "Information-driven sensor path planning by approximate cell decomposition," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 3, pp. 672–689, 2009.

[8] X. Zhu and W. E. Wilhelm, "Three-stage approaches for optimizing some variations of the resource constrained shortest-path sub-problem in a column generation context," *European Journal of Operational Research*, vol. 183, no. 2, pp. 564–577, 2007.

[9] S. Changming, "De-interlacing of video images using a shortest path technique," *IEEE Transactions on Consumer Electronics*, vol. 47, no. 2, pp. 225–230, 2001.

[10] J. Cong, A. B. Kahng, and K.-S. Leung, "Efficient algorithms for the minimum shortest path steiner arborescence problem with applications to VLSI physical design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 1, pp. 24–39, 1998.

[11] T. Deschamps and L. D. Cohen, "Fast extraction of minimal paths in 3D images and applications to virtual endoscopy," *Medical Image Analysis*, vol. 5, no. 4, pp. 281–299, 2001.

[12] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124–1137, 2004.

[13] L. Fu, "An adaptive routing algorithm for in-vehicle route guidance systems with real-time information," *Transportation Research Part B: Methodological*, vol. 35, no. 8, pp. 749–765, 2001.

[14] M.-Y. Kao, *Encyclopedia of Algorithms*, Springer, New York, NY, USA, 2008.

[15] E. Nardelli, G. Proietti, and P. Widmayer, "Finding the most vital node of a shortest path," *Theoretical Computer Science*, vol. 296, no. 1, pp. 167–177, 2003.

[16] A. Broder, R. Kumar, F. Maghoul et al., "Graph structure in the Web," *Computer Networks*, vol. 33, no. 1, pp. 309–320, 2000.

[17] R. E. Bellman, *Dynamic Programming*, Princeton University Press, 1957.

[18] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[19] R. Bellman, "On a routing problem," *Quarterly of Applied Mathematics*, vol. 16, pp. 87–90, 1958.

[20] L. R. Ford and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, NJ, USA, 1962.

[21] L. Fu, D. Sun, and L. R. Rilett, "Heuristic shortest path algorithms for transportation applications: state of the art," *Computers & Operations Research*, vol. 33, no. 11, pp. 3324–3343, 2006.

[22] G. Tsaggouris and C. Zaroliagis, "Non-additive shortest paths," in *Algorithms—ESA 2004*, vol. 3221 of *Lecture Notes in Computer Science*, pp. 822–834, Springer, Berlin, Germany, 2004.

[23] P. Narváez, K.-Y. Siu, and H.-Y. Tzeng, "New dynamic algorithms for shortest path tree computation," *IEEE/ACM Transactions on Networking*, vol. 8, no. 6, pp. 734–746, 2000.

[24] A. V. Goldberg, "Scaling algorithms for the shortest paths problem," *SIAM Journal on Computing*, vol. 24, no. 3, pp. 494–504, 1995.

[25] U. Zwick, "All pairs shortest paths using bridging sets and rectangular matrix multiplication," *Journal of the ACM*, vol. 49, no. 3, pp. 289–317, 2002.

[26] S. Pettie, "A new approach to all-pairs shortest paths on real-weighted graphs," *Theoretical Computer Science*, vol. 312, no. 1, pp. 47–74, 2004.

[27] M. Thorup, "Undirected single-source shortest paths with positive integer weights in linear time," *Journal of the ACM*, vol. 46, no. 3, pp. 362–394, 1999.

[28] J. C. Nash, "The (Dantzig) simplex method for linear programming," *Computing in Science & Engineering*, vol. 2, no. 1, pp. 29–31, 2000.

[29] C.-W. Ahn and R. S. Ramakrishna, "A genetic algorithm for shortest path routing problem and the sizing of populations," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 6, pp. 566–579, 2002.

[30] J. Zhang and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[31] H. Beigy and M. R. Meybodi, "Utilizing distributed learning automata to solve stochastic shortest path problems," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 14, no. 5, pp. 591–615, 2006.

[32] B. Moradabadi and H. Beigy, "A new real-coded Bayesian optimization algorithm based on a team of learning automata for continuous optimization," *Genetic Programming and Evolvable Machines*, vol. 15, no. 2, pp. 169–193, 2014.

[33] X. J. Wu and H. F. Xue, "Shortest path algorithm based on cellular automata extend model," *Computer Applications*, vol. 24, pp. 92–103, 2004.

[34] M. Wang, Y. Qian, and X. Guang, "Improved calculation method of shortest path with cellular automata model," *Kybernetes*, vol. 41, no. 3-4, pp. 508–517, 2012.

[35] C. W. Ahn, R. S. Ramakrishna, C. G. Kang, and I. C. Choi, "Shortest path routing algorithm using Hopfield neural network," *Electronics Letters*, vol. 37, no. 19, pp. 1176–1178, 2001.

[36] M. K. Mehmet Ali and F. Kamoun, "Neural networks for shortest path computation and routing in computer networks," *IEEE Transactions on Neural Networks*, vol. 4, no. 6, pp. 941–954, 1993.

[37] D.-C. Park and S.-E. Choi, "A neural network based multi-destination routing algorithm for communication network," in *Proceedings of the IEEE International Joint Conference on Neural Networks Proceedings*, vol. 2, pp. 1673–1678, IEEE, Anchorage, Alaska, USA, May 1998.

[38] J. J. Hopfield and D. W. Tank, "'Neural' computation of decisons in optimization problems," *Biological Cybernetics*, vol. 52, no. 3, pp. 141–152, 1985.

[39] U.-P. Wen, K.-M. Lan, and H.-S. Shih, "A review of Hopfield neural networks for solving mathematical programming problems," *European Journal of Operational Research*, vol. 198, no. 3, pp. 675–687, 2009.

[40] G. G. Lendaris, K. Mathia, and R. Saeks, "Linear Hopfield networks and constrained optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 29, no. 1, pp. 114–118, 1999.

[41] F. Araújo, B. Ribeiro, and L. Rodrigues, "A neural network for shortest path computation," *IEEE Transactions on Neural Networks*, vol. 12, no. 5, pp. 1067–1073, 2001.

[42] A. Nazemi and F. Omidi, "An efficient dynamic model for solving the shortest path problem," *Transportation Research Part C: Emerging Technologies*, vol. 26, pp. 1–19, 2013.

[43] J. C. Chedjou and K. Kyamakya, "A universal concept based on cellular neural networks for ultrafast and flexible solving of differential equations," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 4, pp. 749–762, 2015.

[44] J. C. Chedjou and K. Kyamakya, "A Novel general and robust method based on NAOP for solving nonlinear ordinary differential equations and partial differential equations by cellular neural networks," *Journal of Dynamic Systems, Measurement, and Control*, vol. 135, no. 3, Article ID 031014, 11 pages, 2013.

[45] S. X. Yang and C. Luo, "A neural network approach to complete coverage path planning," *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, vol. 34, no. 1, pp. 718–725, 2004.

[46] Z. Yi, J. C. Lv, and L. Zhang, "Output convergence analysis for a class of delayed recurrent neural networks with time-varying inputs," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 36, no. 1, pp. 87–95, 2006.

[47] K. E. Parsopoulos and M. N. Vrahatis, "On the computation of all global minimizers through particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 211–224, 2004.