mathematics

MDPI

Article

Generalized Shortest Path Problem: An Innovative Approach for Non-Additive Problems in Conditional Weighted Graphs

Adrien Durand *, Timothé Watteau, Georges Ghazi * and Ruxandra Mihaela Botez *

Laboratory of Applied Research in Active Control, Avionics and AeroServoElasticity (LARCASE), École de Technologie Supérieure (ÉTS), Université de Québec, Montréal, QC H3C 1K3, Canada; timothe.wt@gmail.com
* Correspondence: adrien.durand1709@gmail.com (A.D.); georges.ghazi@etsmtl.ca (G.G.); ruxandra.botez@etsmtl.ca (R.M.B)

Abstract: The shortest path problem is fundamental in graph theory and has been studied extensively due to its practical importance. Despite this aspect, finding the shortest path between two nodes remains a significant challenge in many applications, as it often becomes complex and time consuming. This complexity becomes even more challenging when constraints make the problem non-additive, thereby increasing the difficulty of finding the optimal path. The objective of this paper is to present a broad perspective on the conventional shortest path problem. It introduces a new method to classify cost functions associated with graphs by defining distinct sets of cost functions. This classification facilitates the exploration of line graphs and an understanding of the upper bounds on the transformation sizes for these types of graphs. Based on these foundations, the paper proposes a practical methodology for solving non-additive shortest path problems. It also provides a proof of optimality and establishes an upper bound on the algorithmic cost of the proposed methodology. This study not only expands the scope of traditional shortest path problems but also highlights their computational complexity and potential solutions.

Keywords: universal shortest path problem; line graphs; graph cost functions; optimization techniques

MSC: 68R10

Citation: Durand, A.; Watteau, T.; Ghazi, G.; Botez, R.M. Generalized Shortest Path Problem: An Innovative Approach for Non-Additive Problems in Conditional Weighted Graphs. *Mathematics* **2024**, *12*, 2995. https://doi.org/10.3390/math12192995

Academic Editor: Andrea Scozzari

Received: 26 July 2024
Revised: 14 September 2024
Accepted: 21 September 2024
Published: 26 September 2024

1. Introduction

Among the wide range of challenges addressed in graph theory, the problem of finding the shortest path between two nodes (or vertices), commonly known as the shortest path problem (SPP), is one of the most fundamental and most widely studied. Indeed, a variety of real-world problems, ranging from network routing in communication systems [1], robotics [2], transportation logistics [3,4] and trajectories optimization [5–9], depend on the efficient resolution of the SPP. In fact, regardless of the application context, the problem can always be formulated to determine a path between two vertices of the graph that minimizes the sum of the weights of the edges that compose it. Although this is a universal problem, solving the shortest path remains a topic of interest for many researchers.

Over several decades, Dijkstra's algorithm, introduced by Edsger W. Dijkstra in 1956 [10], and the A* algorithm, developed by Peter Hart, Nils Nilsson, and Bertram Raphael in 1968 [11], have been used as fundamental methods for solving the SPP. Basically, Dijkstra's algorithm finds the shortest path from a starting node to all other nodes in a weighted graph by iteratively selecting the node giving the smallest known distance and by updating the distances to its neighboring nodes. The A* algorithm, on the other hand, can be seen as an improvement of Dijkstra's algorithm, as it uses heuristics to prioritize paths that appear to be the smallest, thus often finding the shortest path more efficiently. While practical in many problems, these algorithms can unfortunately be computationally expensive and slow for very large graphs, particularly when there are complex constraints

between nodes and segments. These algorithms can also be inefficient, or even fail to find a solution, when the cost function to be minimized is non-additive.

The difficulty of solving the SSP is not always due to the choice of the optimization algorithm, but rather to the way in which the problem is defined. Loui [12] highlighted this issue by criticizing the rigidity of classical SPP models and pointed out the relevance of incorporating probabilistic weights in these methods under certain circumstances. To address this problem, two solutions were proposed: (1) minimizing the expected values of the weights, and (2) setting bounds for each weight and minimizing a combined function. Another solution proposed by Loui was the use of dynamic programming, which consists of solving a problem by dividing it into sub-problems, then solving these sub-problems incrementally from the simplest to the most complex, storing their intermediate results.

The use of weighted graphs with random variables, instead of predetermined fixed variables, is a common approach in the literature. This approach was used by Raj et al. in [13] to improve the safety of hazardous goods transport routes. Their objective was to identify the safest path in a graph, while imposing a variance constraint to mitigate paths with excessive uncertainties. In fact, the modeling of the variance is particularly interesting from an optimization point of view, as it does not reduce the cost function to a simple sum of weights. Indeed, the variance, which is quadratic by nature, requires a more complex calculation and can be determined for each path p using a binary vector $x \in [0,1]^{|E|}$, such that $x_\alpha = 1$ if $\alpha \in p$, or $x_\alpha = 0$ otherwise. Starting from this definition, the variance of a path can be computed using the covariance matrix $Q = [q_{\alpha,\beta}] \in \mathbb{R}^{|E|\times|E|}$, which is symmetric and positive definite, with rows and columns indexed by the segments of the graph. The interaction cost between two arcs α and β is reflected by the sum of the off-diagonal entries $q_{\alpha,\beta} + q_{\beta,\alpha}$, while the linear cost of an arc γ is represented by the diagonal element $q_{\gamma,\gamma}$. The variance of a path p is then obtained by using the formula: $\mathrm{Var}(p) = x^T Q x$.

While Raj et al. [13] treated the variance as a constraint, Sen et al. [14] considered it as a component of the cost function (or as the cost function itself). In their study, which focused on the application of a shortest path algorithm for road traffic, they implemented a multi-objective optimization to balance the expected travel time and its variance. The concept of considering a cost function of the form $f(p) = x^T Q x$ is referred to as the quadratic shortest path problem (QSPP). Hu and Sotirov in [15] proposed a solution to this problem using semi-definite programming relaxation methods for directed graphs. They used the alternating direction method of multipliers to find solutions and demonstrated that their bounds were the strongest for this problem at that time. Many methods have also been effective in addressing the QSPP, such as proposed by Rostami et al. [16,17] and by Hu [18].

In several studies, the weighting of segments in shortest path calculations has been treated using random variables. This technique is typically used when it is difficult to predetermine the weights of segments in a graph. Weiss and Kaminka [19] proposed a slightly different modeling approach by exploring shortest path computation techniques when obtaining the exact segment weights is algorithmically expensive. To deal with this complexity, they approximated the weights with a certain level of confidence using a cost estimation function that provides bounds on the actual value of the weights. However, this approach, while practical, requires the development of a shortest path algorithm that works with these estimated weights, which may affect the optimality of the solution.

Turner and Hamacher in [20] introduced the concept of the universal shortest path problem (USPP). This concept consists of incorporating and then extending previously known variants, such as the largest edge cost (bottleneck SSP) and the difference between the largest and smallest edge cost (balanced SPP). In a subsequent study [21], Turner developed strongly polynomial-time algorithms to solve the USPP with equality constraints. These efforts highlight the continuous evolution and diversification of methodologies to address the complex challenges posed by shortest path problems.

An alternative modeling approach for addressing the SPP involves considering graph weights not as real scalar values but as vectors. This vector-based method, adopted by Jiang et al. [22], assigns distinct physical meanings to each vector component. For instance,

in the context of a road network, vector weights might represent the length of a road segment, the degree of congestion, and the probability of delays. The SPP can then be solved by using a cost function that mathematically combines the vector components, reducing it to a traditional cost function. However, this approach does not offer any significant improvements; it simply organizes the data differently while solving the SPP using Dijkstra's or A* algorithms. It is therefore more adapted for multi-objective problems, where the objective is to optimize multiple criteria as considered by Salzman et al. in [23] to solve the SPP with two objective functions using heuristic methods. The vector-based approach may seem trivial, because once the problem is modeled as a weighted graph with associated quantities, engineers can easily apply standard physical formulas to derive a relevant cost. However, the real challenge often lies in the initial modeling of the problem as a weighted graph, for which this approach does not provides solutions.

Vidhya and Saraswathi [24] addressed the SPP under fuzzy conditions using trapezoidal intuitionistic fuzzy numbers (TrIFNs). These TrIFNs were used to model the uncertainties and inaccuracies associated with the arcs in a graph. The problem was formulated as a bi-objective problem: minimizing costs and travel time. Each arc of the graph was associated with a fuzzy cost $\tilde{c}_{ij}$ and a fuzzy time $\tilde{t}_{ij}$.

The literature review clearly shows that in the majority of studies (if not all), a rigid representation of the shortest path problem has always been considered. No current approach envisages a weighting system that considers the position of a segment in the graph, such as adapting the weights depending on the previous nodes (or segments) on the path. It has also been observed that the shortest path problem and its variants offer numerous opportunities for research and application. However, despite the popularity of the problem, the costs associated with a path are typically computed in only three ways: through (1) additive scalar cost functions of the form $f(p) = \sum_{\alpha \in p} c_\alpha$, (2) quadratic scalar cost functions $f(p) = x^T Q x$, or (3) cost functions that incorporate vector weights. Consequently, it seems both interesting and necessary to generalize the cost functions or the methods of weighting the graphs to a wider range of problems.

This paper proposes to extend the concept of cost functions in traditional SPPs by including non-additive functions. It provides a classification of these functions and introduces a method for solving the SPP using a non-additive cost function with conditional weighting. Driven by practical engineering needs, the method is applied to a specific case within the aeronautical sector. In this case study, the integration of geometric constraints requires the use of non-additive cost functions, illustrating the practical application and relevance of the proposed method.

The remainder of this paper is organized as follows: In Section 2, the notations used throughout the paper are clarified. In Section 3, different types of cost functions are classified by introducing new elements of analysis for finite graphs. The application of the line graph and the estimation of the graph size following a sequence of iterated line graphs are explored in Section 4. Insights from the previous sections are then combined in Section 5 to optimally solve the shortest path problem using a k-additive cost function. An application example highlighting the relevance of this method is presented in Section 6. The paper concludes with final remarks and conclusions in the last section.

2. Notations for Graphs

Two types of graphs are considered in this study: undirected graphs and directed graphs (or digraphs). These sets will be denoted as $\mathcal{G}$ and $\mathcal{G}_d$, respectively. Basically, a graph $G \in \mathcal{G}$ can be defined as a set of vertices (or nodes) V connected by a set of edges (or segments) E, and is mathematically denoted as $G = (V, E)$. A digraph is a type of graph where the edges have a direction associated with them. This aspect means that the connections between vertices are not bidirectional, as in a undirected graph, but follow a specific direction from one vertex to another.

To distinguish edges from vertices, the following notations are used: vertices are represented by lowercase Latin letters (e.g., u, v, etc.), while edges are represented by

lowercase Greek letters (e.g., α, β, etc.). Based on these notations, the following properties can be established:

- If $G \in \mathcal{G}$ is an undirected graph, the vertices that constitute an edge are interchangeable. Thus, any edge $\alpha \in E$ defined as a set of nodes such as $\alpha = \{u, v\}$ with $u, v \in V$, can be equivalently expressed as $\alpha = \{v, u\}$. Consequently, an edge is a subset of V composed of two elements.
- If $G' \in \mathcal{G}_d$ is a digraph, an edge can be seen as an ordered pair of two elements that are obviously not interchangeable. Consequently, for any edge $\alpha \in E$ defined as $\alpha = (u, v)$ with $u, v \in V$, a direction must be specified. For this purpose, the first element of the directed edge, $\alpha_1 = u$, is referred to as the tail of the edge α, while the second element, $\alpha_2 = v$, is referred to as the head of the edge α.

Both directed and undirected graphs can be weighted. A weighted graph is defined as $G = (V, E, w)$ where $w : E \to \mathbb{R}$ is a function that assigns a real weight to each edge.

Finally, the set of neighbors of a vertex is denoted as $\mathcal{N}(v)$, and is defined as follows: $\mathcal{N}(v) = \{u \in V | (v, u) \in E\}$. This set includes all vertices u that are directly connected to the vertex (or node) v by an edge in the graph.

3. Classification of Graph Cost Functions

As discussed in Section 1, the effectiveness of solving the SPP depends not only on the definition of the graph, but also on the nature of the cost function. This function is essential for mathematically evaluating the efficiency of a path in terms of metrics to be minimized. The objective of this section is to define three categories of cost functions applicable to any SPP: (1) additive, (2) non-additive, and (3) k-additive. In addition, this section provides mathematical definitions to characterise these categories using techniques of differential analysis on finite graphs.

3.1. Elements of Differential Analysis on a Finite Graph

Calculus on finite weighted graphs is a well-studied field. Dodziuk [25] initiated the study of the Laplacian operator in the discrete domain, highlighting several properties of the continuous operator that transfer well to discrete representation. Subsequent work by Woess [26] and McDonald and Meyers [27] further developed this framework by considering the spaces of vertex or edge functions as a Hilbert space $\mathcal{H}$. This approach involves defining an inner product in $\mathcal{H}(V)$ and $\mathcal{H}(E)$, which facilitates the application of calculus concepts. The mathematical tools developed in these studies also allow the use of differential operators, such as the weighted graph derivative $\partial_{x_i} f(x_j)$, or the weighted gradient (∇_w) and divergence (∇_w^*), respectively, defined as $\nabla_w f(x_i, x_j) = \partial_{x_i} f(x_j)$ with $f \in \mathcal{H}(V)$ and $\langle f, \nabla_w^* F \rangle_{\mathcal{H}(V)} = \langle \nabla_w f, F \rangle_{\mathcal{H}(E)}$ with $F \in \mathcal{H}(E)$.

In this study, various concepts from differential calculus are applied to graphs in order to characterize both the structure of the graph and its associated cost function. For further information on this subject, readers are referred to the studies of Friedman and Tillich [28–30]. These studies introduce a specialized form of "calculus" for graphs, allowing graph theory to make new connections with functional analysis. Such an innovative approach has been applied effectively in various domains, including image processing, machine learning, and network analysis [31–34].

Let $f : \mathcal{P} \to \mathbb{R}^+$ be a cost function associated with a graph G, where $\mathcal{P}$ denotes the set of all paths in $G = (V, E)$. A path $p \in \mathcal{P}$ is an ordered list of nodes (or vertices) without repetition, such that any two consecutive nodes in the list are connected by an edge. Consequently, it can be written that $E \subset \mathcal{P}$. Also, given the absence of node repetition in any path, the number of possible paths is finite, i.e., $|\mathcal{P}| < \infty$.

We can introduce the *space of real path functions* $\mathcal{H}(\mathcal{P})$, which is a $|\mathcal{P}|$-dimensional Hilbert space such as:

$$\mathcal{H}(\mathcal{P}) = \{f : \mathcal{P} \to \mathbb{R}\} \tag{1}$$

The variations of $f \in \mathcal{H}(\mathcal{P})$ can be analysed by examining the function ∂f (i.e., the differential of f) which can be defined as follows:

$$\begin{aligned} \partial f : E \times \mathcal{P} &\to \mathbb{R} \\ (\alpha, p) &\mapsto \partial f(\alpha | \alpha \tilde{\in} p) = f(\alpha \tilde{\in} p) - f(p \setminus \alpha) \end{aligned} \tag{2}$$

where $\alpha \tilde{\in} p$ means that the edge α is included in the path p, and $f(p \setminus \alpha)$ refers to the evaluation of the cost function on the edges of p excluding the edge α.

Equation (2) can be re-written as:

$$\partial f(\alpha | p_1 \alpha p_2) = f(p_1 \alpha p_2) - (f(p_1 u_1) + f(u_2 p_2)) \, , \alpha = (u_1, u_2) \tag{3}$$

to express the variation of f with respect to an edge $\alpha \tilde{\in} p$ that connects two paths p_1 and p_2 within $\mathcal{P}$, such that $p = p_1 \alpha p_2$. This equation is useful to describe the effect of including the edge α in a given path p.

Similarly, Equation (4):

$$\Delta_{i=1,2} \partial f(\alpha | p_i \alpha) = |\partial f(\alpha | p_1 \alpha) - \partial f(\alpha | p_2 \alpha)| \tag{4}$$

can be used to quantify the difference in ∂f between two paths $p_1 \alpha$ and $p_2 \alpha$ sharing the same "terminal" edge α. This equation is useful to compare the effect of connecting the edge α to two given paths, p_1 and p_2.

Finally, using the mathematical definitions introduced in this section, we can establish the following property for a generalized cost function f.

Property 1. *The cost function f can be expressed as the sum of its local differentials by considering the complete path p, such that:*

$$f(p) = \sum_{\alpha \tilde{\in} p} \partial f(\alpha | p) \tag{5}$$

This representation of f is referred to as the "differential form of f".

3.2. Additive and Non-Additive Cost Functions

In graph theory, an additive cost function is a function where the total cost is "simply" the sum of individual costs associated with each edge that compose a path. This type of function is commonly used in problems where the cost can be incrementally accumulated without considering the interaction between edges or nodes.

Mathematically, an additive cost function can be defined on the graph set $\mathcal{P}$ as follows:

$$f \in \mathbb{F}_G(0) \text{ with } \mathbb{F}_G(0) = \{ f \in \mathcal{H}(\mathcal{P}) | \forall p, \alpha \in \mathcal{P} \times E, \;\; \Delta_i \partial f(\alpha | p_i) = 0 \} \tag{6}$$

$\mathbb{F}_G(0)$ is then the set of additive cost function. Based on this definition, a non-additive cost function is any cost function associated with graph G that does not belong to the set $\mathbb{F}_G(0)$, such that:

$$f \notin \mathbb{F}_G(0) \Rightarrow f \in \overline{\mathbb{F}_G(0)} \tag{7}$$

3.3. k-Additive Cost Function

A k-additive cost function generalizes additive cost functions and refines the concept of non-additive cost functions by incorporating interactions among up to k components of a path. In other words, a k-additive cost function is a function for which the value of the cost is influenced by interactions among up to k components. Interactions beyond the k^{th} component are assumed to be negligible or zero. This type of cost function is particularly useful in problems where the cost associated with an edge in a graph is primarily influenced by the properties of adjacent edges.

In addition, the k-additivity nature of a cost function can be oriented. Therefore, we can categorize the following three sets according to their orientation:

Definition 1. *The k-additive on the left cost function set:*

$$\mathbb{F}_G^L(k) := \{f \in \mathcal{H}(\mathcal{P}) | \forall p, \alpha \in \mathcal{P} \times E |\ \ l(p) \geq k \Rightarrow \Delta_i \partial f(\alpha | q_i p \alpha) = 0\} \tag{8}$$

Definition 2. *The k-additive on the right cost function set:*

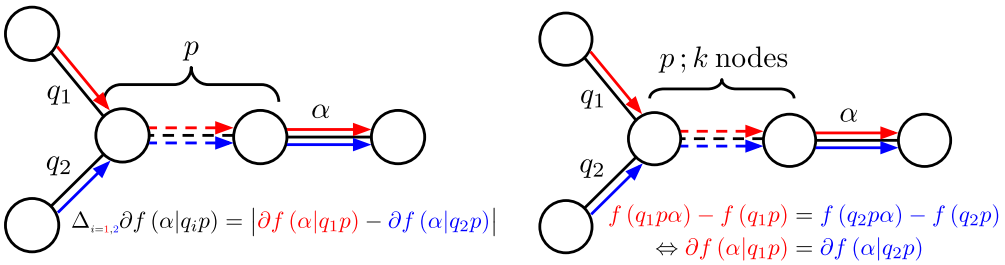
$$\mathbb{F}_G^R(k) := \{f \in \mathcal{H}(\mathcal{P}) | \forall p, \alpha \in \mathcal{P} \times E |\ \ l(p) \geq k \Rightarrow \Delta_i \partial f(\alpha | \alpha p q_i) = 0\} \tag{9}$$

Definition 3. *The k-additive on the left and right cost function set:*

$$\mathbb{F}_G^{LR}(k) := \left\{ f \in \mathcal{H}(\mathcal{P}) | \forall p_1, p_2, \alpha \in \mathcal{P}^2 \times E |\ \ \begin{cases} l(p_1) \geq k \\ l(p_2) \geq k \end{cases} \Rightarrow \Delta_i \partial f(\alpha | q_i p_1 \alpha p_2 q_i') = 0 \right\} \tag{10}$$

We now understand that an additive cost function $\mathbb{F}_G(0)$ is a 0-additive cost function. Specifically, if in the definition it is found that $l(p) = 0$, this indicates that p is an empty path, which is in line with the definition given in Equation (6)

The main concepts presented in this section are illustrated in Figure 1, which represents a section of a graph G.



(**a**) (**b**)

Figure 1. Representation of differential analysis tools on finite graph. (**a**) Representation of $\Delta\partial f(\alpha|p)$. (**b**) Representation of the condition that $f \in \mathbb{F}^L(k)$.

Figure 1a illustrates the difference in ∂f along the edge α between two paths q_1 and q_2. Figure 1b, on the other hand, shows that if ∂f on edge α remains constant when coming from two distinct paths q_1 and q_2, both distant by k nodes, then the cost function is k-additive. Specifically, the equality between $f(q_1 p\alpha) - f(q_1 p)$ and $f(q_2 p\alpha) - f(q_2 p)$, representing the differential of f on the edge α from paths $q_1 p$ and $q_2 p$ (i.e., $\partial f(\alpha|q_1 p)$ and $\partial f(\alpha|q_2 p)$, respectively), confirms these k-additive characteristics.

Property 2. *Let us consider the three sets $\mathbb{F}_G^L(k)$, $\mathbb{F}_G^R(k)$ and $\mathbb{F}_G^{LR}(k)$, as defined in Equations (8)–(10). Thus, we can write:*

$$\forall k \in \mathbb{N} : \begin{cases} \mathbb{F}_G^L(k) \subseteq \mathbb{F}_G^L(k+1), \\ \mathbb{F}_G^R(k) \subseteq \mathbb{F}_G^R(k+1), \\ \mathbb{F}_G^{LR}(k) \subseteq \mathbb{F}_G^{LR}(k+1) \end{cases} \tag{11}$$

Proof of Property 2. Let $f \in \mathbb{F}_G^L(k)$. By Definition 1 of $\mathbb{F}_G^L(k)$ given in Equation (8), we have:

$$\forall p, e \mid l(p) \geq k, \Delta_i \partial f(q_i p e) = 0 \tag{12}$$

This expression states that for any path p and node e where the length of p (i.e., $l(p)$) is at least k, the variation Δ_i in the partial derivative ∂f of the function f along the concatenated path $q_i p e$ equals zero.

We define $p' = vp$ with $p = p^1 \dots p^k$, and $p^1 \in \mathcal{N}(v)$, which implies $p' \in \mathcal{P}$ and $l(p') = k+1$. Therefore, we have:

$$\Delta_i \partial f(q'_i p' e) = \Delta_i \partial f(q'_i vpe) \tag{13}$$

Given that $q''_i = q'_i v$, $\forall i$, it follows that:

$$\Delta_i \partial f(q'_i p' e) = \Delta_i \partial f(q''_i pe) \tag{14}$$

Or, $\Delta_i \partial f(q''_i pe) = 0$, which implies that $f \in \mathbb{F}^L_G(k+1)$. This sequence of equalities demonstrates that the function f, which shows no change in differential after extending the path beyond k nodes, belongs to $\mathbb{F}^L_G(k+1)$.

For $\mathbb{F}^R_G(k) \subseteq \mathbb{F}^R_G(k+1)$, and $\mathbb{F}^{LR}_G(k) \subseteq \mathbb{F}^{LR}_G(k+1)$ the proof is conducted with the same methodology and inverting path order. □

Using the mathematical expressions and properties defined above, we can now represent a cost function as the sum of its local variations, as detailed in Property 3.

Property 3. *$f \in \mathbb{F}_G(k)$ is k-additive because we can reduce its differential form (c.f. Equation (5)) to a sum of terms with the path considered being only k nodes long :*

$$f(p) = f(p^1 \dots p^k) + \sum_{i=k}^{l(p)-1} \partial f(p^i, p^{i+1} | p^{i+1-k} \dots p^{i+1}) \tag{15}$$

This formulation is particularly useful for shortest path problems when the cost functions are non-additive. By using the local variations, we can better understand how the costs accumulate along the different segments of a path, even when the overall cost function is not simply the sum of the costs of each segment.

There are certain cases where the form of the cost function adds complexity to solving the shortest path problem. This complexity arises when the variations in the cost function depend on paths of arbitrary lengths. Such a situation occurs with k-additive functions when $k \to \infty$. This set of functions, which includes all other sets of functions, represents the most challenging category to analyze within this context.

Definition 4. *The general cost function set on G can be defined as:*

$$\mathbb{F}^X_G(\infty) := \mathbb{F}^X_G(k) \bigcup \overline{\mathbb{F}^X_G(k)}, \forall k \in \mathbb{N}, X \in \{L, R, LR\} \tag{16}$$

Using Definition 4, we can conclude that if a function belongs exclusively to $\mathbb{F}^X_G(\infty)$ and to no other defined set of functions, it cannot be solved using the methods presented in this paper for the shortest path problem.

3.4. Examples of Classification

Here, we provide two examples of cost functions which can be classified based on the definitions proposed in the previous sub-sections.

3.4.1. A Classical $\mathbb{F}_G(0)$ Function

Let us consider that each edge $\alpha \in E$ of a graph G is associated with a weight w_α. A classical and trivial cost function would be to compute the cost of a path by summing the weights of its constituent edges. This cost function is commonly used in many shortest path problems. Although it is a simple model, it is often sufficient to solve the problem under consideration. It can be defined as follows:

$$f(p) = \sum_{\alpha \tilde{\in} p} w_\alpha \tag{17}$$

Given the structure of the cost function, it can be easily demonstrated that $f \in \mathbb{F}_G(0)$, meaning that f is additive.

3.4.2. A Simple General $\mathbb{F}_G^L(k)$ Function

Let us consider the sliding product window function f_n, defined such that:

$$f_n(p) = \sum_{i=1}^{l(p)-n} \left(\prod_{k=i}^{i+n} p^k \right) \tag{18}$$

where $p^k \in \mathbb{R}$.

Property 4. *Using the definition proposed in Section 3.3, it can be shown that:*

$$f_n(p) \in \mathbb{F}_G^L(n) \cap \overline{\mathbb{F}_G^L(n-1)}. \tag{19}$$

Proof of Property 4. Let $\alpha = (u_1, u_2) \in E$, and $p = p^1 \dots p^{l(p)} : p^i \in V$. Using the definition of $f_n(p)$ in Equation (18), $f_n(p\alpha)$ can be developed as follows:

$$\begin{aligned} f_n(p\alpha) &= \sum_{i=1}^{l(p)-n} \left(\prod_{k=i}^{i+n} p^k \right) + u_1 \left(\prod_{k=l(p)-n+1}^{l(p)} p^k \right) + u_1 u_2 \left(\prod_{k=l(p)-n+2}^{l(p)} p^k \right) \\ &= f_n(p) + u_1 \left[\left(\prod_{k=l(p)-n+1}^{l(p)} p^k \right) + u_2 \left(\prod_{k=l(p)-n+2}^{l(p)} p^k \right) \right] \end{aligned} \tag{20}$$

Then, by using the definition of ∂f given in Equation (2), we can write:

$$\begin{aligned} \partial f_n(\alpha | p\alpha) &= f_n(p\alpha) - f_n(p) \\ &= u_1 \left[\left(\prod_{k=l(p)-n+1}^{l(p)} p^k \right) + u_2 \left(\prod_{k=l(p)-n+2}^{l(p)} p^k \right) \right] \end{aligned} \tag{21}$$

By assuming that $p' = v_i p\alpha$ where $v_i \in V$ can vary, we can study if the variation of v_i modifies the cost $f(\alpha | p')$:

- $l(p) \geq n \Rightarrow \Delta_i \partial f_n(\alpha | v_i p\alpha) = 0$, thus $f_n \in \mathbb{F}_G^L(n)$
- $l(p) = n-1 \Rightarrow \Delta_i \partial f_n(\alpha | v_i p\alpha) = \partial f_n(\alpha | p\alpha) \Delta v_i \neq 0$, thus $f_n \in \overline{\mathbb{F}_G^L(n-1)}$

This result demonstrates that based on $l(p)$, the cost function f_n may either belong to the set $\mathbb{F}_G^L(n)$ or to the set $\overline{\mathbb{F}_G^L(n-1)}$. Consequently, f_n is a member of the intersection of these two sets, implying that $f_n(p) \in \mathbb{F}_G^L(n) \cap \overline{\mathbb{F}_G^L(n-1)}$. □

4. Line Graph Application for k-Additive Functions

In 1932, Whitney [35] introduced a new construction for undirected graphs, called *line graphs*. This concept was further extended to directed graphs (*line digraph*) with the study proposed by Harary and Norman [36]. Line digraphs are particularly useful in applications where the relationships between the edges of a graph are as important as the relationships between the vertices themselves. Consequently, they can be used to account for constraints in a graph by converting problems stated in terms of edge connectivity into equivalent problems stated in terms of vertex connectivity, which are often easier to analyze and solve using existing graph algorithms.

4.1. Introduction to Line Graphs

Basically, a line digraph $H = (V_H, E_H) \in \mathcal{G}_d$ of a given digraph $G = (V_G, E_G) \in \mathcal{G}_d$ is a graph that represents the adjacencies between the edges of $G = (V_G, E_G)$. The line graph is constructed using a transformation denoted as $H = L_d(G)$, and based on the following conditions:

$$\begin{aligned} V_H &= \{u | u = \alpha \in E_G\} \\ E_H &= \{(\alpha, \beta) | \alpha, \beta \in V_H | \alpha_2 = \beta_1\} \end{aligned} \tag{22}$$

In Equation (22), the first condition means that each vertex of $H = (V_H, E_H)$ corresponds to an edge of the original graph $G = (V_G, E_G)$, while the second condition implies that two vertices in $H = (V_H, E_H)$ are connected by an edge if and only if their corresponding edges in $G = (V_G, E_G)$ share a common vertex.

The transformation $H = L_d(G)$ can be seen as a mapping application from the set of digraphs to itself, denoted as: $L_d : \mathcal{G}_d \to \mathcal{G}_d$. It is important to note that there is a similar mapping application $L : \mathcal{G} \to \mathcal{G}$ for undirected graphs; however, here, we only focus on directed graphs. Indeed, any undirected graph G can be converted into a directed graph G' by transforming each undirected edge $\{u, v\} \in E_G$ into two directed edges $(u, v) \in E_{G'}$ and $(v, u) \in E_{G'}$. Therefore, without loss of generality, the discussion in the remainder of this paper will focus exclusively on directed graphs. Consequently, the notation $L_d(\cdot)$ will be simplified to the more general notation $L(\cdot)$.

Figure 2 illustrates a example of the process of generating a line digraph $G^1 = L(G^0)$ from the original digraph G^0. As shown in this figure, the first step of the transformation involves replacing each edge of the original digraph G^0 with a vertex in G^1. The new vertices in G^1 are labeled to indicate the direction of the original edge in G^0 they represent. For example, the edge from vertex 1 to 4 in G^0 becomes a vertex in G^1 labeled as "14" (i.e., 1 to 4). Subsequently, all vertices created in G^1 are connected with directed edges based on a specific rule: if two edges in G^0 share a common vertex, and one edge's arrival vertex is the other edge's departure vertex, then the corresponding vertices in G^1 are connected by a directed edge. For example, since G^0 has edges from 1 to 4 and from 4 to 2, then in G^1, the vertex representing "14" will have a directed edge to the vertex representing "42".

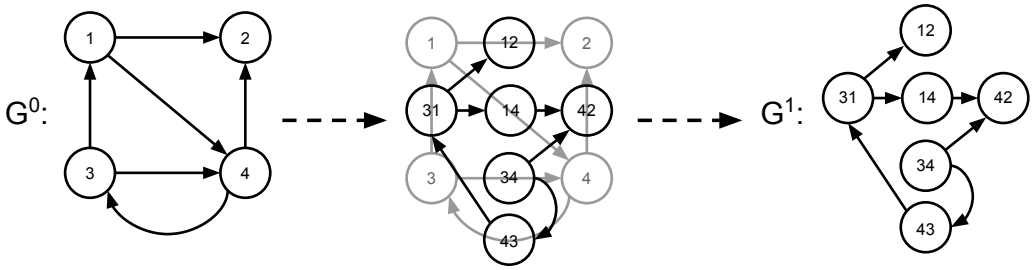


Figure 2. Illustration of line graph transformation.

In this paper, we will use several iterations of $L(\cdot)$. The sequence of graphs G built by the iteration of $L(\cdot)$ can be noted as:

$$G^0,\ G^1 = L(G^0),\ G^2 = L(L(G^0)) = L^2(G^0),\ \ldots,\ G^m = L^m(G^0) \tag{23}$$

This sequence was studied by van Rooij et al. [37], who demonstrated that if a graph contains a cycle, the sequence will never end with an empty graph. They also found that the size of graphs could increase without bounds. This aspect can cause problems if $L(\cdot)$ must be numerically iterated a significant number of times.

Figure 3 illustrates a more general procedure for generating the transformation $L(G^{n-1})$ from the graph G^n.

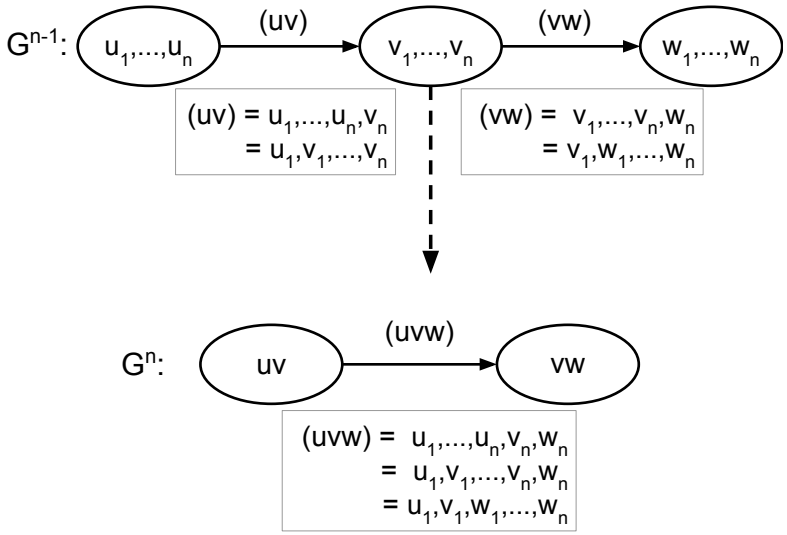


Figure 3. General transformation from $G^{n-1} = L^{n-1}(G)$ to $G^n = L^n(G)$.

4.2. Algebraic Formulation of the Adjacency Matrix of a Line Graph Sequence

One of the fundamental tools in graph analysis is the adjacency matrix. This binary matrix captures all connections between the vertices of a graph, thereby defining its structure. However, graph transformations completely redefine these connections. Consequently, it becomes necessary to determine the equivalent adjacency matrix for a line graph $G^{(n)}$ that results from applying the transformation $L^n(G)$ multiple times.

For this purpose, we introduce the application $\mathcal{E}(\cdot)$, which is a matrix transformation that squares the size of the matrix. This transformation is defined as follows:

$$\begin{aligned} \mathcal{E} : \mathcal{M}_n &\to \mathcal{M}_{n^2} \\ \mathbb{A} &\mapsto \mathcal{E}(\mathbb{A}) = (\mathbb{A} \otimes \mathbb{A}) \odot \Delta_n \end{aligned} \tag{24}$$

where $\odot$ represents the Hadamard product, or element-wise product, and $\Delta_n \in \mathcal{M}_{n^2}(\{0,1\})$ is built such that:

$$\Delta^n = \begin{bmatrix} D_1^n & \dots & D_n^n \\ \vdots & & \vdots \\ D_1^n & \dots & D_n^n \end{bmatrix}, \text{ with } [D_k^n]_{i,j} = \delta_{i,k} \tag{25}$$

and D_k^n is filled with zeros, except on the k^{th} row filled with ones.

Theorem 1. *Let $\mathbb{A} \in \mathcal{M}_n(\{0,1\})$ be the adjacency binary matrix associated with the digraph $G \in \mathcal{G}_d$. The matrix composed of non-zeros rows and columns from $\mathcal{E}(\mathbb{A})$ is the adjacency matrix of $L(G)$.*

Proof of Theorem 1. Let $G, H \in \mathcal{G}_d$, such that $L(G) = H$, and let $\mathbb{A}_G$ be the adjacency matrix associated with G. In this proof, we will construct the adjacency matrix $\mathbb{A}_H$ associated with H.

Let us assume that we do not know if an edge exists; we will then consider all possibilities. Each node u in V_G can potentially be connected to any other node v in V_G, including itself (if we allow self-loops (u,u)). This results in $|V_G|^2$ possible connections. If we denote $|V_G| = n$, therefore, the size of the matrix $\mathbb{A}_H$ will be $n^2 \times n^2$.

Let us keep the order of the nodes used in the adjacency matrix $\mathbb{A}_G$ as follows:

$$u_1, u_2, \cdots, u_n$$

We choose the order of the nodes for $\mathbb{A}_H$ as follows:

$$(u_1,u_1),(u_1,u_2),\cdots,(u_1,u_n),(u_2,u_1),\cdots(u_2,u_n),\cdots,(u_n,u_n)$$

Thus, we can say that:

- There could be a one in the line $k = in + v$, corresponding to $\alpha_k = (u_i, u_v)$ in $\mathbb{A}_H$ only if $[\mathbb{A}_G]_{i,v} = 1$; meaning that $\alpha_k = (u_i, u_v) \in E_G$.
- There could be a one in the k^{th} line, with $k = in + v$, and the l^{th} row, with $l = jn + w$, corresponding to the connection (α_k, α_l) only if $v = j$; meaning that $(\alpha_k)_2 = (\alpha_l)_1$ (i.e., the second node of α_k is equal to the first node of α_k).

Using Properties 1–3, we can deduce that:

$$[\mathbb{A}_H]_{k,l} = [\mathbb{A}_G]_{i,v} \times [\mathbb{A}_G]_{j,w} \times \delta_{v,j} \tag{26}$$

where $k = in + v$ and $l = jn + w$.

By definition of the Kronecker product ($\otimes$) and using the definition of $\mathcal{E}(\cdot)$ and Δ^n in Equations (24) and (25), respectively, we can write:

$$A_H = (\mathbb{A}_G \otimes \mathbb{A}_G) \odot \Delta_n = \mathcal{E}(\mathbb{A}_G) \tag{27}$$

This last result thus demonstrates that $\mathcal{E}(\mathbb{A}_G)$ is the adjacency matrix of A_H. □

The result of Theorem 1 can be generalized to a line digraph G^m resulting from m-transformations, using the following theorem:

Theorem 2. *Let $\mathbb{A} \in \mathcal{M}_n(\{0,1\})$ be the adjacency binary matrix associated with the digraph $G \in \mathcal{G}_d$. The matrix defined by:*

$$\mathcal{E}^m(\mathbb{A}) = \left[\bigotimes_{i=1}^{2^m} \mathbb{A}\right] \odot K_m \quad : \quad K_1 = \Delta_n \; ; \; K_{m+1} = (K_m \otimes K_m) \odot \Delta_{n^{2^{m+1}}} \tag{28}$$

is the adjacency matrix of $G^m = L^m(G)$

Proof of Theorem 2. Let $\mathbb{A}_{G^m} \in \mathcal{M}_n$. We can demonstrate the result shown in Theorem 2 using a proof by induction. For this purpose, let us denote $(*)$ as the property we want to demonstrate.

- For $m = 1$: the proof of Theorem 1 demonstrates that the property $(*)$ is true.
- For $m + 1$: we assume that the property $(*)$ is true for a given m, meaning that:

$$\mathbb{A}_{G^m} = \left[\bigotimes_{i=1}^{2^m} \mathbb{A}\right] \odot K_m \tag{29}$$

is the adjacency matrix of $G^m = L^m(G)$.

Using the result of the proof of Theorem 1, we can say that $\mathbb{A}_{G^{m+1}} = \mathcal{E}(\mathbb{A}_{G^m})$ is the adjacency matrix of $L(G^m) = G^{m+1}$, and we can write:

$$\mathcal{E}(\mathbb{A}_{G^m}) = (\mathbb{A}_{G^m} \otimes \mathbb{A}_{G^m}) \odot \Delta_X \tag{30}$$

To determine the value of X, we need the size of $\mathbb{A}_{G^m}$. If $\mathbb{A}_{\mathbb{G}}$ is a $n \times n$ matrix, then $\mathbb{A}_{G^1}$ is a $n^2 \times n^2$ matrix. By induction: $\mathbb{A}_{G^m}$ is a $n^{2^m} \times n^{2^m}$ so $X = n^{2^{m+1}}$, since:

$$\begin{aligned} \mathcal{E}(\mathbb{A}_{G^m}) &= (\mathbb{A}_{G^m} \otimes \mathbb{A}_{G^m}) \odot \Delta_{n^{2^{m+1}}} \\ &= \left(\left\{ \left[\bigotimes_{i=1}^{2^m} \mathbb{A}\right] \odot K_m \right\} \otimes \left\{ \left[\bigotimes_{i=1}^{2^m} \mathbb{A}\right] \odot K_m \right\} \right) \odot \Delta_{n^{2^{m+1}}} \end{aligned} \tag{31}$$

Using the direct product and Kronecker product rules, and arranging the terms of the equations, we obtain :

$$\mathcal{E}(\mathbb{A}_{G^m}) = \left[\bigotimes_{i=1}^{2\times 2^m} \mathbb{A}\right] \odot (K_m \otimes K_m) \odot \Delta_{n^{2^{m+1}}} \tag{32}$$

Based on the definition of K_{m+1}, we can write:

$$\mathcal{E}(\mathbb{A}_{G^m}) = \left[\bigotimes_{i=1}^{2^{m+1}} \mathbb{A}\right] \odot K_{m+1} \tag{33}$$

This last result implies that $\left[\bigotimes_{i=1}^{2^{m+1}} \mathbb{A}\right] \odot K_{m+1}$ is the adjacency matrix of $L^{m+1}(G)$. Thus, the property $(*)$ is true for $m+1$.

In conclusion, the property $(*)$ is true for all $m \in \mathbb{N}^*$ □

Finally, Theorem 3 can be used to quantify the upper bound of the complexity of the iterated line graphs in terms of the number of edges, considering that while the structure becomes increasingly interconnected, it is still finite and bounded as a function of the number of vertices n and the number of iterations m.

Theorem 3. *For any graph G, the number of edges of its associated m^{th} line graph $L^m(G)$ is bounded. This aspect implies: $\forall G \in \mathcal{G},\ |V_G| = n \Rightarrow |E_{L^m(G)}| \leq n^{m+2}$*

Proof of Theorem 3. Let us define the application $\mathcal{S}(\cdot)$ as the sum of all terms of a matrix, such as:

$$\begin{aligned} \mathcal{S} :\ & \mathcal{M} \to \mathbb{R} \\ & \mathbb{A} \mapsto \mathcal{S}(\mathbb{A}) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} \end{aligned} \tag{34}$$

The application $\mathcal{S}(\cdot)$ can be used to count the number of edges in a graph given its adjacency matrix. Specifically, for a graph $G \in \mathcal{G}$ with its associated adjacency matrix $\mathbb{A}_G$, applying $\mathcal{S}(\mathbb{A}_G)$ yields the number of edges, represented as $|E_G|$.

Considering that $\mathbb{A}$ and K_m are binary matrices, we can write :

$$\mathcal{S}(\mathcal{E}^m(\mathbb{A})) \leq \mathcal{S}(K_m) \tag{35}$$

We need to know the value of $\mathcal{S}(K_m)$ depending on m. Let $K_m \in \mathcal{M}_n$ and $K_{m+1} = (K_m \otimes K_m) \odot \Delta_n$. Thus, $[K_{m+1}]_{n(i-1)+v,n(j-1)+w} = [K_m]_{iv} \times [K_m]_{jw} \times \delta_{jv}$, and:

$$\mathcal{S}(K_{m+1}) = \sum_{k=1}^{n^2}\sum_{l=1}^{n^2} [K_{m+1}]_{kl} \tag{36}$$

Let $k = n(i-1)+v$ and $l = n(j-1)+w$ with $i, j, v, w \in [\![1, n]\!]$. We can therefore rewrite the previous equation as follows:

$$\begin{aligned}\mathcal{S}(K_{m+1}) &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{v=1}^{n}\sum_{w=1}^{n}[K_m]_{iv}\times[K_m]_{jw}\times\delta_{jv}\\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{w=1}^{n}[K_m]_{ij}\times[K_m]_{jw}\\ &= \sum_{i=1}^{n}\sum_{w=1}^{n}\underbrace{\left(\sum_{j=1}^{n}[K_m]_{ij}\times[K_m]_{jw}\right)}_{[K_m^2]_{iw}}\\ &= \sum_{i=1}^{n}\sum_{w=1}^{n}[K_m^2]_{iw} = \mathcal{S}(K_m^2)\end{aligned} \tag{37}$$

which means that $\forall m \in \mathbb{N}^*$, $\mathcal{S}(K_{m+1}) = \mathcal{S}(K_m^2)$.

Moreover, if a graph has n nodes, it implies that $K_1 \in \mathcal{M}_{n^2}$. We find that $\mathcal{S}(K_1) = n^3$. Additionally, $\forall m \in \mathbb{N}^*$, the relationships $\mathcal{S}(K_m^2) = n \cdot \mathcal{S}(K_m)$ holds.

By integrating all this information, we deduce that $\mathcal{S}(K_m) = n^{m-1} \cdot \mathcal{S}(K_1) = n^{m+2}$.

Finally, we can conclude that $|E_{L^m(G)}| \leq n^{m+2}$. □

As shown in Figure 4, the size of the iterated line graph could increase exponentially with the number of transformations m. For instance, by considering a digraph G with $n = 10$ nodes, the associated 6^{th} line digraph, i.e., $L^6(G)$, will have a maximum of 10^7 nodes. This aspect represents one of the main drawbacks of using multiple transformations, as it could become too time-consuming to generate the m^{th} line digraph.

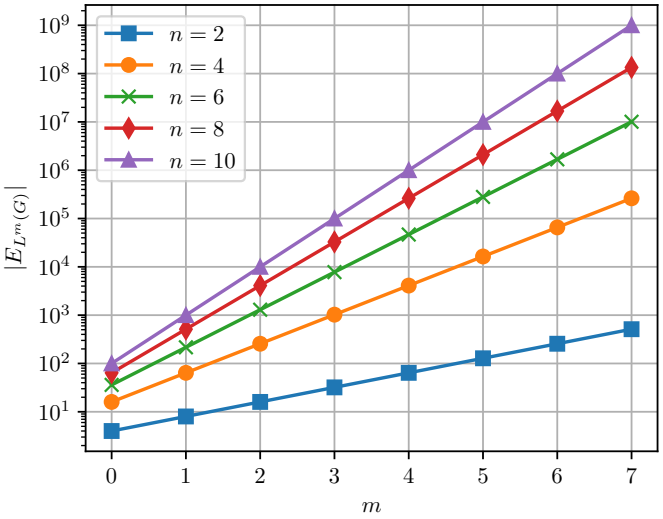


Figure 4. Evolution of the maximal size of the digraph.

5. Application of Line Graph Sequences for Non-Additive Shortest Path Problems

This section provides a comprehensive procedure for solving the shortest path problem using non-additive cost functions. Several algorithms are proposed to be used for (1) constructing the line digraph, (2) adding conditional weights to the generated line digraph to include constraints, and (3) adapting Dijkstra's algorithm to the line digraph. In addition, a proof of the optimality of the proposed methodology is also provided.

5.1. Problem Definition

Let consider the following optimization problem associated with a digraph $G = (V_G, E_G) \in \mathcal{G}_d$:

$$\min_{p \in \mathcal{P}_G} f(p) | f \in \mathbb{F}_G(k)$$
$$s.t. \begin{cases} p_1 = s \\ p_n = t, \text{ with } n = l(p) \\ \forall i \in [\![1, n]\!] \mid (p_i, p_{i+1}) \in E_G \end{cases} \tag{38}$$

This problem is similar to the shortest path problem, except that the cost function is a k-additive cost function. The only constraints are that the first and the last vertices of the path are imposed: s for the source, and t for the target, and that the path can only follow the edges of the graph.

5.2. Proposed Methodology—Conditional Weighting Shortest Path (CWSP)

To solve the optimization problem presented in Equation (38), it is first necessary to construct a substitute graph, and then to use the properties of the k-additive cost functions established in Section 3 to add weights to this substitute graph. Extra weighted edges can be introduced to connect the original graph G with the substitute graph H. Finally, Dijkstra's algorithm can be adapted and applied to the weighted substitute graph to solve the optimization problem in Equation (38). Algorithms 1–3 can be applied for that purpose.

Algorithm 1: Construction of a line graph $L(G)$

```
Input: Graph G = (V_G, E_G).
Result: L(G).
V_{L(G)} ← {(α) : α ∈ E_G}
E_{L(G)} ← ∅
for α ∈ E_G do
    for β ∈ E_G do
        if α_2 = β_1 then
            E_{L(G)} ← E_{L(G)} ∪ {(α, β)}
        end
    end
    L(G) ← (V_{L(G)}, E_{L(G)})
end
```

Algorithm 1 is proposed for constructing the line digraph $L(G)$ associated with the digraph G. This transformation converts the original problem, which is stated in terms of edge connectivity, into a substitute (or equivalent) problem expressed in terms of vertex connectivity. Figures 5 and 6 illustrate an example of the application of Algorithm 1. Figure 5a shows the original digraph G, while Figure 5b shows the resulting line digraph $L(G) = H$. The line digraph in Figure 5b is then completed to include all entering nodes, as shown in Figure 6. These nodes represent the entry nodes into the original graph, i.e., all nodes that have only one connection with another node in the original digraph G.

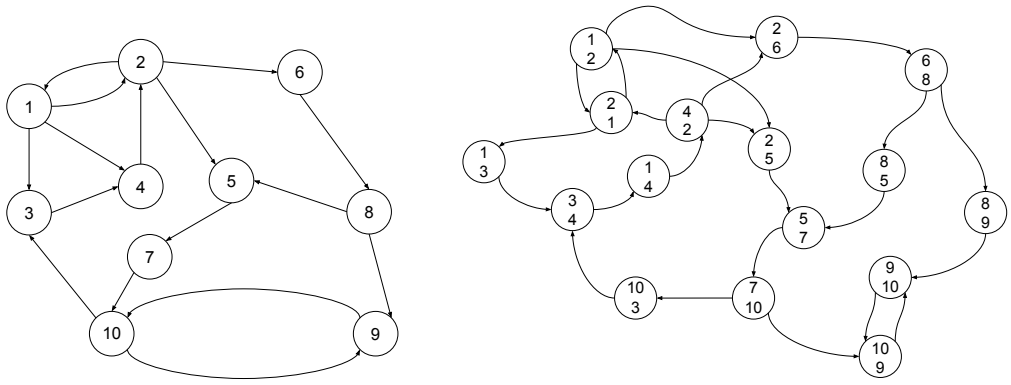
Once the line digraph $L(G)$ is constructed, it serves as the substitute graph. The edges of this new graph must be weighted based on the static weights present in the original digraph G, as well as on various constraints typically captured by the k-non-additive cost function $f(p)$. This process is performed with Algorithm 2. For instance, using the the sliding product window $f_{swp}(p)$ as defined in Equation (18), and considering the line digraph in Figure 6, we can determine the weight for the edge $((2,6),(6,8))$ to be $\partial f_2(6,8|2,6,8) = 96$. Similarly, for an entering edge such as $(3,(3,4))$, the weight would be $f_{swp}(3,4) = 12$.

Algorithm 2: Conditional weighting of a substitute graph

Input: Graph: $G = (V_G, E_G)$.
Cost function : f.
k order of additivity of f
Result: $H = (V_H, E_H, W_H)$.
$H \leftarrow L^k(G)$ ▷ Using Algorithm 1
for $(p, q) \in E_H$ **do**
  ▷ $l(p) = l(q) = k + 1$
  $(u_1 \dots u_{k+1}) \leftarrow p$
  $(v_1 \dots v_{k+1}) \leftarrow q$
  $\beta \leftarrow (u_{k+1}, v_{k+1})$ ▷ $\beta \in E_G$
  $w_\alpha \leftarrow \partial f(\beta|\alpha)$
end
$V_H \leftarrow V_H \bigcup V_G$
for $p \in V_H$ **do**
  $p_1 \leftarrow u$
  $\gamma \leftarrow (u, p)$ ▷ γ is an enter edge from G to H
  $E_H \leftarrow E_H \bigcup \{\gamma\}$
  $w_\gamma \leftarrow f(p \dots p_k)$
end
$W_H \leftarrow \{w_\alpha : \alpha \in E_H\}$

Algorithm 3: Dijkstra for a substitute graph

Input: Graph $H = (V_H, E_H)$, source node s, and target node t.
Result: p^*
for $v \in V_H$ **do**
  cost[v] $\leftarrow \infty$
  predecessor[v]$\leftarrow \emptyset$
end
cost[s] $\leftarrow 0$
$Q \leftarrow V_H$
while $Q \neq \emptyset$ **do**
  $u \leftarrow$ extract-min(Q)
  for $\alpha = (u, v)$ **do**
    if $cost[v] > cost[u] + w_\alpha$ **then**
      cost[v] $\leftarrow$ cost[u] $+ w_\alpha$ predecessor[v] $\leftarrow u$
    end
    if $v_{k+1} = t$ **then**
      $u_{last} = v$
      break
    end
  end
end
$p^* = (t)$
$u = u_{last}$
while $l(u) = k + 1$ **do**
  $v = u$
  $u \leftarrow$ predecessor[v]
  $p^* \leftarrow (u_{k+1}, p^*)$
end
$p^* \leftarrow (v_1 \dots v_k, p^*)$



(**a**) (**b**)

Figure 5. Illustration of a digraph G and its transformation $H = L(G)$. (**a**) Original digraph G. (**b**) Resulting line digraph $H = L(G)$.

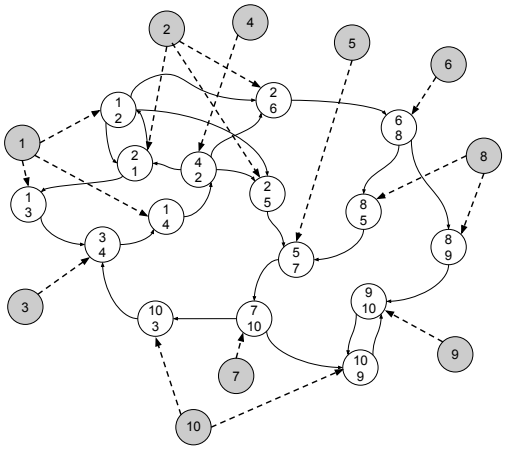


Figure 6. Line graph with entering nodes.

Finally, using the conditional weighted line digraph $H = (V_H, E_H, W_H)$ obtained with Algorithm 2, the shortest path from the source node s to the target node t in H can be determined using Dijkstra's algorithm (see Algorithm 3). It is important to consider that a node v in H represents a series of node in G. This configuration implies that the target node $t \in V_G$ is reached when the last element of $v \in V_H$ is t, and the length of v is $k+1$. The stop criterion in Dijkstra's algorithm will be achieved when $v_{k+1} = t$.

The set of the three Algorithms 1–3 constitutes the conditional weighting shortest path (CWSP) method for solving a shortest path problem when the cost function is k-left-additive $\mathbb{F}^L(k)$ or k-right-additive $\mathbb{F}^R(k)$.

5.3. Proof of Optimality and Algorithmic Cost

The optimal path p^* returned by Algorithm 3 represents the shortest path between the source node s and the target node t within the substitute graph H. This optimal path can be transferred to the original graph G, and the next Property 5 guarantees that the transferred path is indeed the optimal path that minimizes the cost function described in Equation (38).

Property 5. *The path returned by Algorithm 3 is a path that minimizes the cost function $f(p)$, such that:*

$$p^* \in \operatorname{argmin}\{f(p) | p \in \mathcal{P},\ p_1 = s,\ p_n = t\} \tag{39}$$

Proof of Property 5. First of all, because $f \in \mathbb{F}_G(k)$, we can break it into the sum of local differentials using Property 1:

$$f(p) = \sum_{\alpha \tilde{\in} p} \partial f(\alpha|p) \tag{40}$$

Let us define $q \in \mathcal{P}_H$, such that:

$$\begin{aligned} q_1 &= p_1^* \\ q_i &= p_i^* \dots p_{k+i}^* \text{ for } i = 1 \dots n-k \end{aligned} \tag{41}$$

and let us define $T_H = \{v \in V_H | v_{k+1} = t\}$.

Dijkstra's algorithm ensures that:

$$q \in \text{argmin} \left\{ \sum_{\alpha \tilde{\in} q} w_\alpha \mid q_1 = s,\ q_n \in T_H \right\}$$

and:

$$\sum_{\alpha \tilde{\in} q} w_\alpha = f(p_1^* \dots p_k^*) + \sum_{\alpha \tilde{\in} q \setminus q_1} \partial f(\beta|\alpha) \ \text{ with } \ \beta = (\alpha_{k+1}\alpha_{k+2}) \tag{42}$$

Since $l(\alpha \setminus \beta) = k$ and $f \in \mathbb{F}_G(k)$, we find that $\Delta_r \partial f(\beta|r\alpha) = 0$. In addition, $\forall \alpha \tilde{\in} q$, it follows that $\alpha \tilde{\in} p^*$. Therefore, we can substitute $\partial f(\beta|\alpha) = \partial f(\beta|p^*)$.

We can also change the sum of variables as follows:

$$\alpha \tilde{\in} q \setminus q_1 \to \beta \tilde{\in} p^* \setminus (p_1^* \dots p_k^*) := p' \tag{43}$$

which leads to:

$$\sum_{\alpha \tilde{\in} q} w_\alpha = f(p_1^* \dots p_k^*) + \sum_{\beta \in p'} f(\beta|p^*) = f(p^*) \tag{44}$$

In conclusion, p^* is indeed the minimum solution of the k-additive cost function problem. □

In addition to assessing optimality, it might also be interesting to determine the algorithmic cost of the proposed CWSP method. This cost can be evaluated in terms of the maximum number of iterations required to solve the optimization problem. This parameter depends on two aspects: the size of the initial graph $|V_G|$, and the order of the cost function k. Property 6 can be used to determine this cost.

Property 6. *Consider a graph $G = (V_G, E_G)$ such as $|V_G| = n$, and an associated cost function $f \in \mathbb{F}^X(k)$, where $X = \{R \text{ or } L\}$. Thus:*

1. *The algorithmic cost of computing $L^k(G)$ is at highest order $\mathcal{O}\left(m + n^{k+2}\right)$.*
2. *The algorithmic cost of the CWSP algorithm is, at highest order, $\mathcal{O}(n^{k+1}(n + (k+1)\log(n)))$.*

Proof of Property 6.

1. The algorithm cost of computing $L(G)$ is $\mathcal{O}(|E_G|)$:

Theorem 1 establishes that at a given iteration i, the number of edges is bounded by $|E_{L^i(G)}| \leq n^{i+2}$. Summing all the costs, we obtain the following result:

$$\begin{aligned} \text{cost} &= \sum_{i=0}^{m} |E_{L^i(G)}| \\ \text{cost} &= m + n^3 + n^4 + \ldots + n^{m+2} \\ \text{cost} &\propto m + n^{m+2} \end{aligned} \tag{45}$$

2. The algorithmic cost of the CWSP with a Fibonacci heap [38]:

Since $|E_{L^k(G)}| \leq n^{k+2}$, and using the definition of $L(\cdot)$, we can write :

$$|V_{L^k(G)}| = |E_{L^{k-1}(G)}| \tag{46}$$

which implies:

$$|V_{L^k(G)}| \leq n^{k+1} \tag{47}$$

The computational cost of Dijkstra's algorithm is $\mathcal{O}|E| + |V|\log(|V|)$, and it is used in the last step with Algorithm 3. By substitution we can upper-bound the number of iterations, n_{it}:

$$\begin{aligned} n_{it} &= |E_{L^k(G)}| + |V_{L^k(G)}| \cdot \log(|V_{L^k(G)}|) \\ &\leq n^{k+2} + n^{k+1}\log(n^{k+1}) \\ &\leq n^{k+1}(n + (k+1)\log(n)) \end{aligned} \tag{48}$$

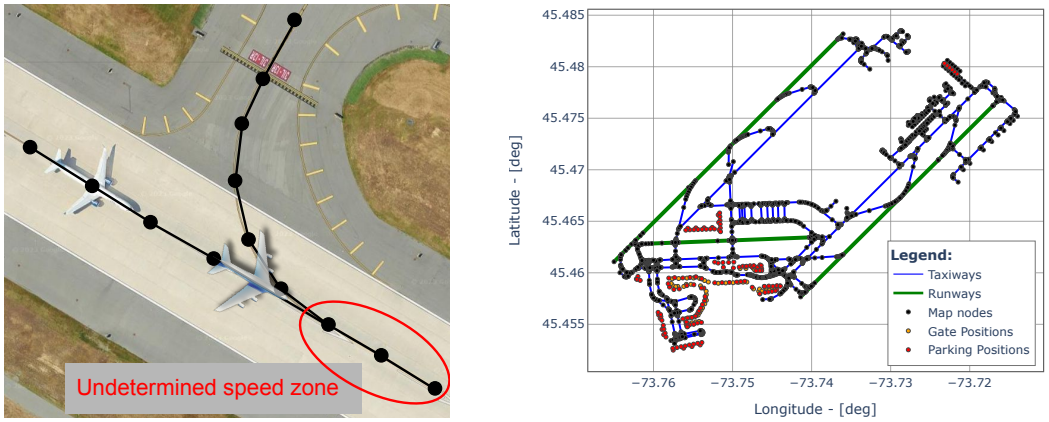
Therefore, the computational cost is at most $\mathcal{O}(n^{k+1}(n + (k+1)\log(n)))$. □

6. Application: Case Study in Airport Trajectory Optimization

One of the main motivations for the development of the shortest path technique with conditional weighting is to address problems where constraints, which can be geometric or physical, can lead to the generation of paths that are infeasible in practice.

A typical example is the management of aircraft ground trajectories at airports. Indeed, airports impose various types of restrictions and rules that pilots must comply with, such as directional taxiways and prohibited turns to avoid sharp maneuvers or to account for the size (or weight) of the aircraft. In addition, aircraft can accelerate or decelerate as they enter or exit turns. Consequently, their ground speeds are influenced by their current trajectory and the segment they will taxiing next. As illustrated in Figure 7a, these factors create areas within the airport where the ground speed of an aircraft cannot be predetermined. In the segments in the circled area in Figure 7a, the aircraft ground speed can vary according to three scenarios: the aircraft can (1) maintain its ground speed if it continues on the straight segment; (2) decelerate if it enters a turn; or (3) accelerate if it exits a turn. Additionally, certain trajectories are prohibited due to restrictions on maneuvers, such as sharp turns. For instance, as shown in Figure 7a, the aircraft is not authorized to perform a left turn to enter the turn segment, or a right turn to exit the turn segment.

Due to these constraints, optimizing the ground trajectory of an aircraft within a graph that replicates the topology of an airport, as illustrated in Figure 7b, presents significant challenges.



(**a**) (**b**)

Figure 7. Illustration of a graph for an airport and constraint during ground operations. (**a**) Example of restrictions along the path of an aircraft during taxi operation. (**b**) Example of a graph for Montreal Trudeau International Airport (CYUL).

6.1. Model

In this section, we evaluate the effectiveness of the conditional weighting shortest path method proposed in this paper for solving the shortest path problem in an airport, taking into account constraints on sharp turns.

The cost function for this application is based on the total distance traveled by the aircraft, while avoiding sharp turns. All sharp turns are determined *a priori* on the original graph using a classification method that considers four nodes (or three edges) to verify if the middle segment is allowed. Turn constraints are included in the cost function with a barrier function $b(\cdot)$, defined as:

$$\begin{aligned} &b : V^4 \to \{1, +\infty\} \\ &b(v_{i-1}, \ldots, v_{i+2}) = \begin{cases} 1 & \text{, if the turn is authorized} \\ +\infty & \text{, if the turn is not authorized} \end{cases} \end{aligned} \tag{49}$$

By considering the function:

$$d : V^2 \to \mathbb{R}^+ \tag{50}$$

which returns the distance between two points of the airport, the cost function can be expressed as follows:

$$f(p) = d(v_1, v_2) + \sum_{i=2}^{l(p)-2} d(v_i, v_{i+i}) \cdot b(v_{i-1}, \ldots, v_{i+2}) + d(v_{n-1}, v_n) \cdot c(v_{n-3}, \ldots, v_n) \tag{51}$$

Using the definitions and properties introduced in Section 3.3, we can confirm that $f(p)$ belongs to $\mathbb{F}_G^{LR}(2)$ (see Equation (10)). Indeed, the left–right k-additivity nature of the cost function comes from the need to consider information about the edges preceding and succeeding a given edge to accurately classify sharp turns.

6.2. Methodology and Results

To compare and validate the CWSP method, the optimization problem described in Equation (38) was applied to Miami International Airport. In addition, 50 different scenarios were considered to find the shortest path between two randomly selected points on the airport graph, each separated by at least 20 nodes. For each scenario, the CWSP method using the non-additive cost function specified in Equation (51) was applied. At the same time, Dijkstra's algorithm was also applied; however, since it cannot handle non-additive cost functions, the cost function for this algorithm was limited to the total distance traveled by the aircraft:

$$f^*(p) = \sum_{i=1}^{l(p)-1} d(v_i, v_{i+1}) \tag{52}$$

which is additive by nature.

It should be noted that the optimal solution found by Dijkstra's algorithm was further evaluated using the cost function described in Equation (51). This evaluation determined whether the solution provided by Dijkstra's algorithm was reliable (i.e., feasible) or not. If the value of the cost function was finite, this indicated that Dijkstra's algorithm had found a viable solution. Conversely, if the value of the cost function was infinite, this suggested that the solution was not feasible in practice, as it included at least one prohibited sharp turn.

Figure 8 shows examples of results comparisons for four different problems. In this figure, the trajectory found by Dijkstra's algorithm is highlighted in green, while the trajectory found by the CWSP method is highlighted in blue. At a first glance, both strategies were able to find a trajectory between the source and destination nodes. However, when the trajectory found by Dijkstra's algorithm was evaluated using the cost function in Equation (51), an infinite cost value was observed. This result can be attributed to the fact

that the solution provided by Dijkstra's algorithm contains sharp turns that are forbidden. These turns are marked with red circles for each problem. In contrast, the CWSP method successfully found a better and feasible solution without any forbidden turns.

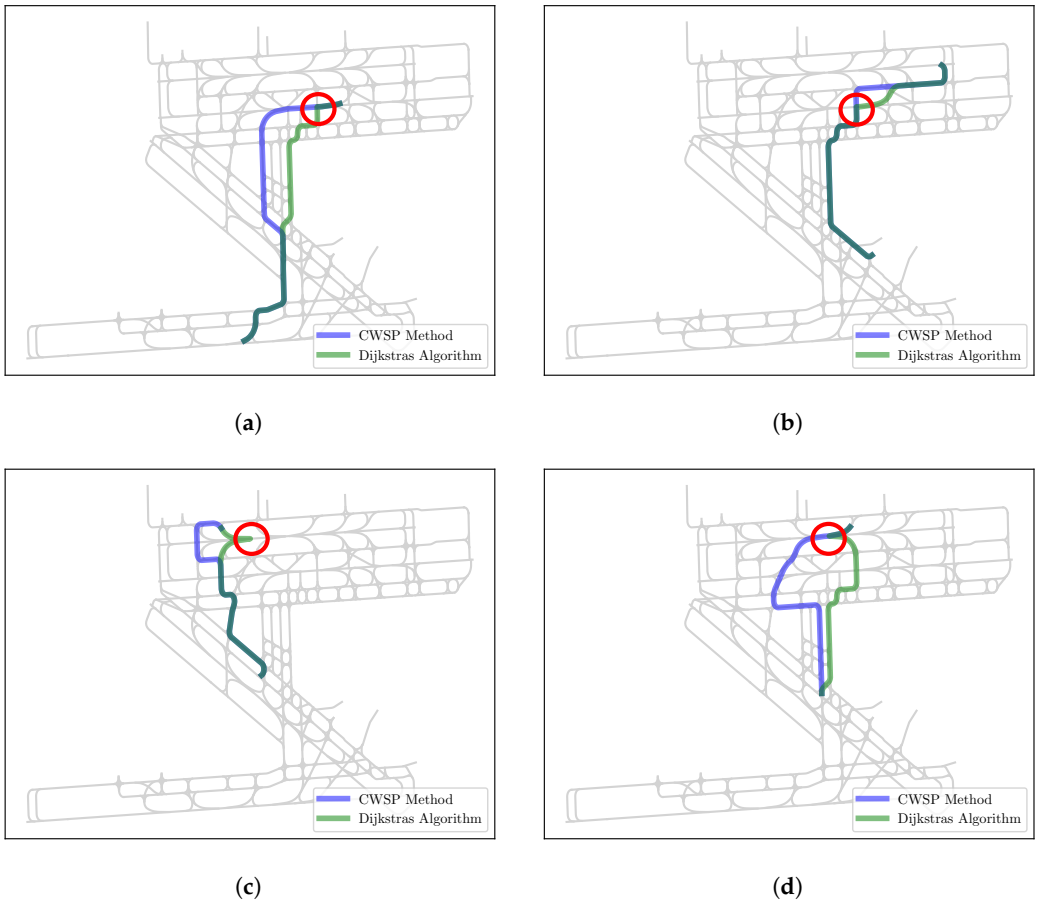


Figure 8. Examples of results; comparison between a trajectory found by the CWSP method and a trajectory found using Dijkstra's algorithm. (**a**) Problem #4. (**b**) Problem #40. (**c**) Problem #47. (**d**) Problem #49.

This analysis was repeated for all 50 problems. A table summarizing the results for each problem is included in the Appendix A. By analyzing the results, it was found that the CWSP method successfully solved all problems, consistently finding the shortest feasible trajectory while avoiding sharp turns. Dijkstra's algorithm was also able to find a solution, but only 28% of these solutions were reliable (i.e., feasible without forbidden turns). Interestingly, as shown in the table in the Appendix A, whenever Dijkstra's algorithm found an acceptable solution, it was identical to the solution found by the CWSP method. This analysis is significant because it shows that the CWSP method is not only capable of finding the shortest path in the sense of Dijkstra's algorithm, it is also well-suited to handle complex graphs with constraints or to deal with non-additive cost functions.

7. Conclusions

In this paper, we presented a generalization of classical cost functions on graphs, traditionally called "additive" functions, which are part of the space of real edge functions. This generalization leads to the development of real path functions, a category known as non-additive or k-additive. While these functions are applicable to various shortest path problems, their complexity makes traditional shortest path algorithms designed to optimize real edge functions unsuitable.

To address a shortest path problem, our first step was to develop tools for characterizing and analyzing path functions. By studying the variations of these functions, we were able to classify them into distinct sets.

Then, for a subset of these non-additive function sets, we proposed a technique for solving the shortest path problem. This technique relies on the weighting of a substitute graph, which is determined by successive iterations of transforming a graph into its associated line graph. The proposed method has proven effective on a simple problem that could not have been solved using Dijkstra's algorithm alone, demonstrating its potential for application to numerous physical and engineering problems requiring complex modeling. This article establishes an initial framework and provides an exact solution that represents a significant advance in the field of complex shortest path problems.

However, the computational cost of such a method is difficult to bound because it strongly depends on the topology of the initial graph. Furthermore, the complexity of the cost function rapidly increases with the size of the replacement graph used to accurately solve the problem. Finally, this method, with its solid proof of convergence, can be used to validate the efficiency of faster heuristic methods for solving the shortest path problem with non-additive path functions.

The research initiated in this paper can be extended in several directions. Firstly, the upper bound provided for the size of the linear graph sequence is highly dependent on the structure of the initial graph. A tighter limit could potentially be obtained by examining how the sequence of linear graphs evolves as a function of the topology of the initial graph. Furthermore, it has been observed that when k is large, the cost of constructing and weighting the line graph becomes significant. It would be useful to study and limit the error introduced when approximating a k-additive cost function by a $(k-n)$-additive cost function, where $n < k$. This approach could lead to faster solutions to the problem while quantifying the associated error. Finally, the general concept of non-additive cost functions encompasses different types of functions. The methodology proposed in this paper is suitable for functions in $\mathbb{F}(k)$. However, we believe that exact methods applicable to functions in $\mathbb{F}(\infty)$ which are similar to quadratic forms such as $f(p) = x^T Q x''$, are also worth exploring.

Author Contributions: Conceptualization, A.D.; funding acquisition, G.G. and R.M.B.; methodology, A.D. and T.W.; project administration, G.G.; supervision, G.G. and R.M.B.; validation, A.D. and T.W.; writing—original draft, A.D.; writing—review and editing, G.G. and R.M.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC: RGPIN-2022-03864), and by the Fonds de Recherche ÉTS sur les Changements Climatiques (FRECC).

Data Availability Statement: Dataset available on request from the authors.

Acknowledgments: This research was performed at the Laboratory of Applied Research in Active Controls, Avionics and AeroServoElasticity research (LARCASE). The authors would like to thank the Fond de Recherche ÉTS sur les Changements Climatiques (FRECC) for their support of this project.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

SPP	Shortest Path Problem
USPP	Universal Shortest Path Problem
QSPP	Quadratic Shortest Path Problem
CWSP	Conditional Weighting Shortest Path
LARCASE	Laboratory of Applied Research in Active Control, Avionics and AeroServoElasticity

$\mathcal{G}$	Set of graphs
$\mathcal{P}$	Set of paths in a graph
$\mathcal{H}(\mathcal{P})$	Space of real path functions
$G = (V, E)$	General graph
$f(\cdot)$	General cost function
$\partial f(\cdot\|\cdot)$	differential of f
$\Delta_{i=1,2} X_i$	Variation on any quantity X_i
$p \backslash \alpha$	Path p deprived of edge α
$\mathbb{F}_G^X(k)$	Set of k- additive cost function in direction X
$\mathbb{A}$	Adjacency matrix
Δ^n	Binary matrix define in Equation (25)
$L(G)$	Line graph of G

Appendix A. Results for All 50 Problems

N°Problem	Dijkstra's Algorithm Solution Cost Ignoring Constraints	Dijkstra's Algorithm Solution Cost Considering Constraints	Conditional Weighting Method Considering Constraints
1	1959.5	$+\infty$	2017.1
2	3945.8	$+\infty$	3983.9
3	768.1	768.1	768.1
4	2327.3	$+\infty$	2655.8
5	1128.2	1128.2	1128.2
6	308.8	$+\infty$	836.8
7	1014.6	1014.6	1014.6
8	2061.4	$+\infty$	2094.6
9	2297.5	$+\infty$	3940.0
10	1434.3	1434.3	1434.3
11	1720.5	$+\infty$	1889.7
12	2567.8	$+\infty$	4500.3
13	946.7	$+\infty$	1047.1
14	593.0	$+\infty$	648.0
15	607.2	$+\infty$	690.3
16	1389.8	$+\infty$	1828.7
17	2884.8	$+\infty$	2962.4
18	2751.1	$+\infty$	2958.0
19	2044.1	2044.1	2044.1
20	2023.3	2023.3	2023.3
21	1913.3	$+\infty$	2074.1
22	2249.4	2249.4	2249.4
23	3877.6	$+\infty$	4133.8
24	1340.1	$+\infty$	1421.6
25	3225.7	$+\infty$	3258.5

N°Problem	Dijkstra's Algorithm Solution Cost Ignoring Constraints	Dijkstra's Algorithm Solution Cost Considering Constraints	Conditional Weighting Method Considering Constraints
26	2898.6	$+\infty$	2954.3
27	426.0	$+\infty$	1317.5
28	519.1	519.1	519.1
29	2282.8	$+\infty$	2695.8
30	2071.3	$+\infty$	2128.9
31	1838.7	1838.7	1838.7
32	2562.8	$+\infty$	2782.2
33	1408.9	1408.9	1408.9
34	365.6	$+\infty$	1266.3
35	1948.7	$+\infty$	2055.0
36	3694.3	$+\infty$	5595.6
37	951.5	$+\infty$	2937.5
38	1988.0	$+\infty$	1992.8
39	1294.4	1294.4	1294.4
40	2533.9	$+\infty$	2603.4
41	2934.4	2934.4	2934.4
42	2945.9	2945.9	2945.9
43	2512.1	2512.1	2512.1
44	1635.1	1635.1	1635.1
45	477.9	477.9	477.9
46	1506.4	1506.4	1506.4
47	1762.6	$+\infty$	1775.0
48	1380.4	$+\infty$	2324.4
49	1729.8	$+\infty$	2080.2
50	990.7	$+\infty$	1192.2
Number of solution	Not applicable	14	50

References

1. Wang, R.; Zhou, M.; Wang, J.; Gao, K. An Improved Discrete Jaya Algorithm for Shortest Path Problems in Transportation-Related Processes. *Processes* **2023**, *11*, 2447. [CrossRef]
2. Wahhab, O.; Al-Araji, A.S. An Optimal Path Planning Algorithms for a Mobile Robot. *Iraqi J. Comput. Commun. Control Syst. Eng.* **2021**, *21*, 44–58. [CrossRef]
3. Rosyida, I.; Asih, T.S.N.; Waluya, S.; Sugiyanto. Fuzzy Shortest Path Approach for Determining Public Bus Route (Case study: Route planning for "Trans Bantul bus" in Yogyakarta, Indonesia). *J. Discret. Math. Sci. Cryptogr.* **2021**, *24*, 557–577. [CrossRef]
4. Priliana, C.Y.; Rosyida, I. The Ambulance Route Efficiency for Transporting Patients to Referral Hospitals Based on Distance and Traffic Density Using the Floyd-Warshall Algorithm and Google Traffic Assistance. In Proceedings of the 4th International Seminar on Science and Technology (ISST 2022), Palu, Indonesia, 2–3 November 2022; Atlantis Press: Amsterdam, The Netherlands, 2023; pp. 349–360. [CrossRef]
5. Murrieta Mendoza, A.; Beuze, B.; Ternisien, L.; Botez, R.M. Branch & Bound-Based Algorithm for Aircraft VNAV Profile Reference Trajectory Optimization. In Proceedings of the 15th AIAA Aviation Technology, Integration, and Operations Conference, Dallas, TX, USA, 22–26 June 2015; American Institute of Aeronautics and Astronautics: Reston, VA, USA. 2015. [CrossRef]
6. Murrieta-Mendoza, A.; Hamy, A.; Botez, R.M. Four- and Three-Dimensional Aircraft Reference Trajectory Optimization Inspired by Ant Colony Optimization. *J. Aerosp. Inf. Syst.* **2017**, *14*, 597–616. [CrossRef]
7. Murrieta-Mendoza, A.; Botez, R.M.; Bunel, A. Four-Dimensional Aircraft En Route Optimization Algorithm using the Artificial Bee Colony. *J. Aerosp. Inf. Syst.* **2018**, *15*, 307–334. [CrossRef]
8. Murrieta-Mendoza, A.; Romain, C.; Botez, R.M. 3D Cruise Trajectory Optimization Inspired by a Shortest Path Algorithm. *Aerospace* **2020**, *7*, 99. [CrossRef]

9. Durand, A.; Toulet, M.; Ghazi, G.; Botez, R.M. An Innovative Approach to Aircraft Ground Trajectory Optimization using a Bi-Directional A* Algorithm. In Proceedings of the Canadian Aeronautics and Space Institute (CASI) AERO23 Conference, Ottawa, ON, Canada, 14–16 November 2023.
10. Dijkstra, E.W. A note on two problems in connexion with graphs. *Numer. Math.* **1959**, *1*, 269–271. [CrossRef]
11. Hart, P.E.; Nilsson, N.J.; Raphael, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [CrossRef]
12. Loui, R.P. Optimal paths in graphs with stochastic or multidimensional weights. *Commun. ACM* **1983**, *26*, 670–676. [CrossRef]
13. Sivakumar, R.A.; Batta, R. The variance-constrained shortest path problem. *Transp. Sci.* **1994**, *28*, 309–316. [CrossRef]
14. Sen, S.; Pillai, R.; Joshi, S.; Rathi, A.K. A Mean-Variance Model for Route Guidance in Advanced Traveler Information Systems. *Transp. Sci.* **2001**, *35*, 37–49. [CrossRef]
15. Hu, H.; Sotirov, R. On solving the quadratic shortest path problem. *INFORMS J. Comput.* **2020**, *32*, 219–233. [CrossRef]
16. Rostami, B.; Malucelli, F.; Frey, D.; Buchheim, C. On the Quadratic Shortest Path Problem. In Proceedings of the Experimental Algorithms, Paris, France, 29 June–1 July 2015; Springer: Berlin/Heidelberg, Germany, 2015; pp. 379–390. [CrossRef]
17. Rostami, B.; Chassein, A.; Hopf, M.; Frey, D.; Buchheim, C.; Malucelli, F.; Goerigk, M. The Quadratic Shortest Path Problem: Complexity, Approximability, and Solution Methods. *Eur. J. Oper. Res.* **2018**, *268*, 473–485. [CrossRef]
18. Hu, H.; Sotirov, R. A Polynomial Time Algorithm for the Linearization Pproblem of the QSPP and its Applications. *arXiv* **2018**, arXiv:1802.02426.
19. Weiss, E.; Kaminka, G.A. A Generalization of the Shortest Path Problem to Graphs with Multiple Edge-Cost Estimates. In Proceedings of the ECAI 2023, Kraków, Poland, 30 September–4 October 2023.
20. Turner, L.; Hamacher, H.W. On Universal Shortest Paths. In Proceedings of the Operations Research Proceedings 2010: Selected Papers of the Annual International Conference of the German Operations Research Society, 1–3 September 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 313–318. [CrossRef]
21. Turner, L. Variants of the Shortest Path Problem. *Algorithmic Oper. Res.* **2011**, *6*, 91–104.
22. Jiang, S.; Feng, Z.; Zhang, X.; Wang, X.; Rao, G. A Multi-dimension Weighted Graph-Based Path Planning with Avoiding Hotspots. In Proceedings of the Knowledge Graph and Semantic Computing: Semantic, Knowledge, and Linked Big Data, Singapore, 19–22 September 2016; pp. 15–26. [CrossRef]
23. Salzman, O.; Felner, A.; Hernández, C.; Zhang, H.; Chan, S.H.; Koenig, S. Heuristic-Search Approaches for the Multi-Objective Shortest-Path Problem: Progress and Research Opportunities. In Proceedings of the Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, Macao, China, 19–25 August 2023; pp. 6759–6768. [CrossRef]
24. Vidhya, K.; Saraswathi, A. A Novel Method for Finding the Shortest Path with Two Objectives Under Trapezoidal Intuitionistic Fuzzy Arc Costs. *Int. J. Anal. Appl.* **2023**, *21*, 121–121. [CrossRef]
25. Dodziuk, J. Difference Equations, Isoperimetric Inequality and Transience of Certain Random Walks. *Trans. Am. Math. Soc.* **1984**, *284*, 787–794. [CrossRef]
26. Woess, W. *Random Walks on Infinite Graphs and Groups*; Cambridge Tracts in Mathematics, Cambridge University Press: Cambridge, UK, 2000. [CrossRef]
27. McDonald, P.; Meyers, R. Diffusions on Graphs, Poisson Problems and Spectral Geometry. *Trans. Am. Math. Soc.* **2002**, *354*, 5111–5136. [CrossRef]
28. Friedman, J.; Tillich, J.P. Calculus on Graphs. *arXiv* **2004**, arXiv:cs/0408028. [CrossRef]
29. Friedman, J.; Tillich, J.P. Laplacian Eigenvalues and Distances Between Subsets of a Manifold. *J. Differ. Geom.* **2000**, *56*, 285–299. [CrossRef]
30. Friedman, J.; Tillich, J.P. Wave Equations for Graphs and the Edge-Based Laplacian. *Pac. J. Math.* **2004**, *216*, 229–266. [CrossRef]
31. Elmoataz, A.; Lezoray, O.; Bougleux, S. Nonlocal Discrete Regularization on Weighted Graphs: A Framework for Image and Manifold Processing. *IEEE Trans. Image Process.* **2008**, *17*, 1047–1060. [CrossRef] [PubMed]
32. Gilboa, G.; Osher, S. Nonlocal Operators with Applications to Image Processing. *Multiscale Model. Simul.* **2009**, *7*, 1005–1028. [CrossRef]
33. Desquesnes, X.; Elmoataz, A.; Lézoray, O. Eikonal Equation Adaptation on Weighted Graphs: Fast Geometric Diffusion Process for Local and Non-Local Image and Data Processing. *J. Math. Imaging Vis.* **2013**, *46*, 238–257. [CrossRef]
34. Mahmood, F.; Shahid, N.; Skoglund, U.; Vandergheynst, P. Adaptive Graph-Based Total Variation for Tomographic Reconstructions. *IEEE Signal Process. Lett.* **2018**, *25*, 700–704. [CrossRef]
35. Whitney, H. Congruent Graphs and the Connectivity of Graphs. *Am. J. Math.* **1992**, *54*, 150–168. [CrossRef]
36. Harary, F.; Norman, R.Z. Some Properties of Line Digraphs. *Rend. Del Circ. Mat. Palermo* **1960**, *9*, 161–168. [CrossRef]
37. van Rooij, A.C.M.; Wilf, H.S. The Interchange Graph of a Finite Graph. *Acta Math. Acad. Sci. Hung.* **1965**, *16*, 263–269. [CrossRef]
38. Fredman, M.L.; Tarjan, R.E. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM* **1987**, *34*, 596–615. [CrossRef]