information

MDPI

Article

An Evolutionary Algorithm for an Optimization Model of Edge Bundling †

Joelma de M. Ferreira *, **Hugo A. D. do Nascimento** and **Les R. Foulds**

Institute of Informatics, Federal University of Goiás, Goiás 74690-900, Brazil; hadn@inf.ufg.br (H.A.D.d.N.); lesfoulds@gmail.com (L.R.F.)

* Correspondence: joelmaferreira@inf.ufg.br

† This manuscript is an extended version of our paper published in the proceedings of International Conference on Information Visualization Theory and Applications (IVAPP, 2017), Porto, Portugal, 27 February–1 March 2017, and published in the proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, Madeira, Portugal, 27–29 January 2018.

Received: 2 May 2018; Accepted: 23 June 2018; Published: 26 June 2018

Abstract: This paper discusses three edge bundling optimization problems that aim to minimize the total number of bundles of a graph drawing, in conjunction with other aspects, as the main goal. A novel evolutionary algorithm for edge bundling for these problems is described. The algorithm was successfully tested by solving the related problems applied to real-world instances in reasonable computational time. The development and analysis of optimization models have received little attention in the area of edge bundling. However, the reported experimental results demonstrate the effectiveness and the applicability of the proposed evolutionary algorithm to help resolve edge bundling problems by formally defining them as optimization models.

Keywords: edge bundling; optimization problem; evolutionary algorithm; approximate solution

1. Introduction

A graph is a mathematical structure for representing inherent relationships between discrete objects. It is often drawn as a node-link diagram, but, as the number of elements increases, its ability to effectively display information without visual clutter is a challenge. Several existing graph drawing techniques attempt to reduce visual clutter. An example is edge bundling, which has gained attention as a way to improve the readability of a graph drawing by merging geometrically close edges into bundles along a shared path (see Figure 1).

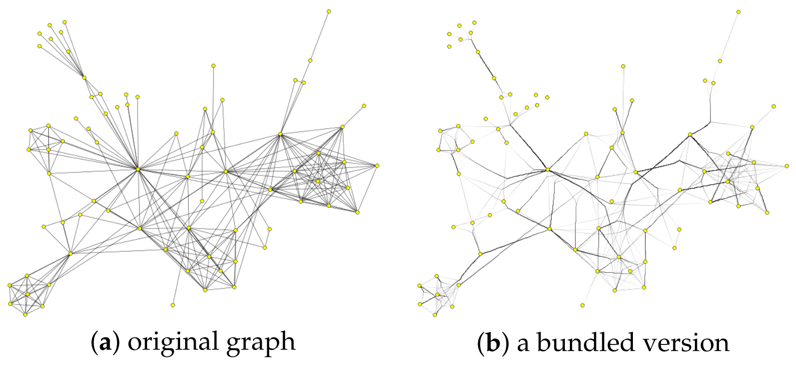


Figure 1. Illustration of the concept of edge bundling.

Traditional edge bundling approaches are not problem-specific methods, but, instead, are application-oriented. They are not dedicated to solving a mathematically-formulated problem, i.e., they do not follow a systematic and theory-guided process to propose and solve edge bundling optimization problems by defining decision variables and a given objective function to be optimized. Consequently, these approaches do not address the issue of identifying the "best" set of bundles based on a group of criteria that are mathematically defined in the form of an objective function.

Departing from tradition, Ferreira et al. [1] proposed the study of edge bundling as a formally defined combinatorial optimization problem. The authors focused on a class of edge bundling problems for which the set of edges composing every bundling has to be explicitly chosen as main problem-decision variables. In order to demonstrate their ideas, the authors formulated a constrained combinatorial optimization problem, called $EB-Star$, in which the total number of bundles has to be minimized and only edges with a common end node (i.e., edges that are adjacent) can be bundled together. As Peng et al. [2] affirm, bundling only adjacent edges seems to provide a better sense of edge relations of a graph at the node level. A more general and interesting problem was also defined by Ferreira et al. [1] by adding a new constraint to the $EB-Star_\alpha$ problem. The constraint consisted of imposing a maximum angle (α) condition that any pair of adjacent edges must satisfy in order to be bundled. That problem was called $EB-Star_\alpha$, which becomes the $EB-Star$ problem for high values of α. In the present paper, we refer to $EB-Star_\alpha$ as the *Angle-Based Edge Bundling Problem* (ABEB) in order to be consistent with other problems that are introduced. Ferreira et al. showed that $EB-Star$ is NP-hard and, therefore, $EB-Star_\alpha$ has the same complexity for high values of α. This justifies the use of heuristic methods for solving ABEB.

In the present paper, we continue an investigation into the formal modeling of edge bundling and propose two other problems, in addition to the ABEB. One problem is termed the Compatibility-Based Edge Bundling Problem (CBEB) which involves minimizing the total number of bundles while maximizing a multi-objective function that incorporates well-known edge compatibility measures. The other problem is an extension of the last one and allows the creation of bundles with non-adjacent edges and is termed General-Based Edge Bundling (GBEB). We also propose an approximate evolutionary algorithm for ABEB, CBEB and GBEB called here Evolutionary Edge Bundling (EEB). As far as we know, EEB is the first heuristic method ever proposed for these problems as Ferreira et al. [1] presented only an integer linear formulation for the ABEB, not a solution method.

Experiments with the evolutionary algorithm show that it produces close-to-optimal solutions for some of the nontrivial ABEB instances tested. The results emphasize the importance of optimization approaches to edge bundling, not only as a way of visualizing a graph with less visual clutter, but also as a means of systematically studying and comparing problem definitions, methods and solutions in this field. From now on, we consider only undirected graphs.

The remainder of the paper is organized as follows. Section 2 briefly surveys relevant work on various aspects of edge bundling. Section 3 presents a definition of edge bundling as a combinatorial optimization problem, as proposed in [1]. Section 4 provides the formal definitions of ABEB, CBEB and GBEB. Some compatibility and aesthetic criteria are then discussed in Section 5. Section 6 introduces the evolutionary edge bundling algorithm which aims to solve the ABEB, CBEB and GBEB problems and Section 7 describes some experimental results. Section 8 discusses a method for rendering the edge bundling solutions as a final step in the edge bundling process. A comparison of EEB with previous edge bundling techniques is discussed in Section 9. Finally, in Section 10, we draw some conclusions and discuss ideas for future research.

2. Related Work

The scientific literature on edge bundling is extensive and reports many edge bundling techniques. For instance, an approach called *hierarchical edge bundling* [3] uses hierarchy tree branches for edge routing; a method called *geometric edge bundling* [4] forms bundles by routing edges along a mesh generated by a triangulation algorithm; the *force-directed edge bundling* technique [5,6] splits the edges into segments that attract each other as a basis for bundling edges; the *skeleton-based edge bundling* method [7] generates a skeleton from the medial axes of groups of similar edges, and then attracts edges to this skeleton; and the *kernel density estimation edge bundling* method [8] computes a density map and moves the edges in order to create bundles.

Those and many other edge bundling techniques attempt to aggregate edges with similar properties into bundles, addressing mainly the question of specifying how these edges should be routed along the same path. The techniques produce solutions in which the coarse structure of the graph is revealed but fail to show connection patterns at the node level. Peng, Lu and Peng [2] affirm that sometimes it is more important to show the connection trends of a node rather than the overall network structure. For example, in graph drawings representing a route network, the actual relationships between interconnected components are usually more relevant than the coarse structure of the graph. This is due to the fact that many traditional edge bundling methods generate bundles that have common interior segments and multiple source and destination nodes, i.e., they "knot" the edges in the middle of the bundle.

Consequently, Peng, Lu and Peng [2] proposed an algorithm called *node-based edge bundling* that bundles and "knots" edges nearer to the common node of adjacent edges. Nocaj and Brandes [9] refined the technique of [2] and proposed a method called *stub bundling* that joins only edges that share the same endpoint. The method has the aim of visualizing unambiguous graphs and retrieving the exact source and target of each edge. The method uses the angle between consecutive edges as the criterion for choosing which edges to be joined together.

Even though the methods in [2,9] efficiently produce bundles with only adjacent edges, neither of them has the aim of explicitly minimizing the number of bundles. In order to achieve this goal, it may be needful to formulate an optimization problem having decision variables, constraints and objectives. For addressing the aforementioned need, the present paper investigates the problem of finding the "best" configuration of bundles (involving adjacent and non-adjacent edges) that optimizes a given objective function.

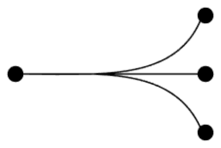
3. Edge Bundling as an Optimization Problem

There is a current lack of fundamental and theoretical principles that can be used to objectively measure the effectiveness of bundling techniques. Despite some attempts to formalize the presentation of certain edge bundling methods, bundling itself, as a technique, as yet lacks an underlying formalism that can unify previous methods. Most existing edge bundling definitions are vague, each being related to slightly different characteristics of the problem. Moreover, edge bundling is related to both joining edges and determining the paths of the edges.

Recently, McKnight [10] and Lhuillier, Hurter and Telea [11] discussed various complementary aspects of more complex mathematical formulations of edge bundling. McKnight defines a *bundle* as a set of two or more "edge segments", and *edge bundling* as the process of segmenting the edges. Lhuillier, Hurter and Telea [11] define a *bundle* as a set of paths that share similarities and *edge bundling* as a process that creates bundles and trails.

We now complement the definitions of bundle and edge bundling just mentioned. By analyzing the various existing approaches in this area, it is possible to identify distinct types of bundle structures and methodologies.

Bundles can be separated into two types, those composed of (i) only edges sharing a common end node (bundling of only adjacent edges), as in the methods described in [2,9,12–14], or of (more commonly) (ii) merged edges with multiple origin or destination nodes (see Figure 2).


(**a**) only adjacent edges

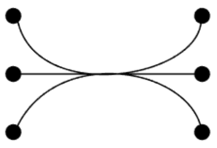
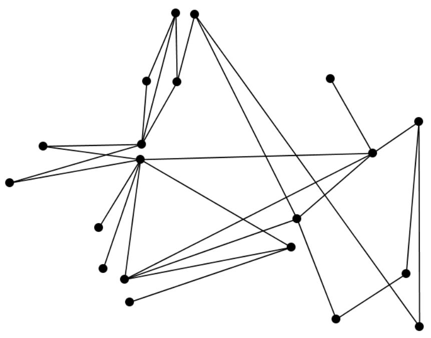
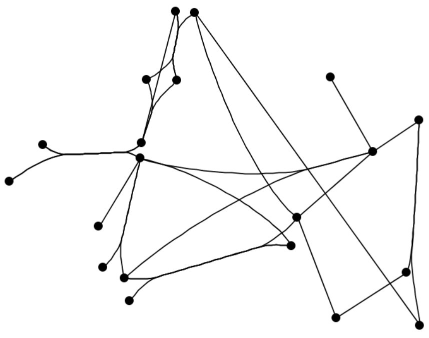

(**b**) multiple source or destination nodes

Figure 2. Type of bundles.

Regarding the bundling methodologies, an edge bundling method can construct bundles in an explicit or in an implicit way [15]. In explicit approaches, the edges are first clustered and then a rendering module draws each group of edges [7,16–19]. In implicit approaches, there is no pre-grouping of the edges. Instead, the bundles are considered as oriented-curved routes, each containing a set of edges. The related methods employ strategies for defining routes and making associations between them [5,8]. McKnight [10] affirms that many existing edge bundling approaches do not generate bundles directly (explicitly). In fact, most edge bundling algorithms output just a graph drawing and the identification of the individual bundles is indirect.

Figure 3 uses a synthetic graph (G1) to illustrate the difference between the solution generated by the explicit approach of the EEB model (described in Section 6) and the solution generated by the implicit Force-Based Edge Bundling approach [5], implemented in the d3.js library of visualization routines. The two solutions are visually rather different, which makes it difficult to compare them in terms of quality. It is especially challenging to make such comparisons when the specific elements (e.g., parameters and constraints) related to the edge bundling problem are ill defined. In fact, the comparison is meaningless as the underlying problems are significantly different from each other.


(**a**) original graph


(**b**) solution of the explicit method EEB

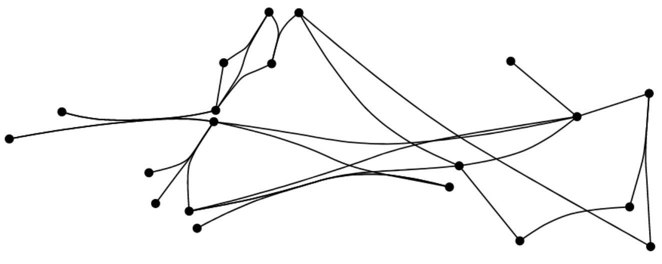

(**c**) solution of the implicit method FDEB [5] generated by d3.js

Figure 3. Comparison of solutions produced by implicit and explicit methodologies.

In general, the search for a "good" edge-bundling layout is carried out heuristically, and is based on an informal definition of given desirable optimization goals. In opposition, we take a different line for the investigation of edge bundling strategies by formally defining edge bundling as an optimization problem. We also aim at identifying a problem solution that represents an explicitly generated bundle configuration. For the structure of the solution, we follow the definition given in [10] that focuses on computing bundles directly. However, we consider a bundle as a set of edges (not just a set of edge segments as in [10]).

Thus, in [1], edge bundling was formulated as an optimization problem that attempts to find the "best" set of bundles in terms of some given parameters, goals and constraints. A general edge bundling optimization problem, that can be used as a framework for describing more specific problems, was defined by the authors as follows:

Definition 1. *Let $G = (V, E)$ be a graph and D a given unbundled node-link drawing of G in the plane. Consider the set $S = (E_1, E_2, ..., E_n)$, where E_i are subsets of E (not necessarily disjoint), $n \in \mathbb{N}^+$. Furthermore, let R be a function of G, D and S that renders a bundled graph drawing version of D (denoted by D^R), given some extra necessary information, such as rules for routing the edges. The* ***General Edge bundling Problem*** *is hence to determine the set S (here called bundles), with $E = \cup_{i=1}^{n} E_i$, so that a set F of objective functions, representing aesthetic edge bundling measurements of D^R are optimized, and a set P of constraints (defining mainly which edges can be bundled together) are satisfied.*

Note that Definition 1 enables the inclusion of the routing problem, which is to find the precise paths of the bundled (and possibly unbundled) edges. This is a question to be addressed as an optimization problem. This can be done in either of two ways: (1) by determining the routing as a second-level problem totally inside the rendering function R; or (2) by extending the formal definition in order to have extra variables that determine the routing and are used in R. In both cases, functions that evaluate the quality of the resultant edge-bundling drawings, produced by R, can be included in the set F, making the edge-routing problem more intrinsic in the optimization process. Some quality aspects that may influence the routing, such as minimizing the amount of ink for drawing the bundles, or minimizing the number of edge crossings, can be pursued using these approaches. However, in the present paper, the routing problem is not considered critical for the optimization process. Therefore, it was treated as a fully independent problem (see (1) above) and was considered an external, post-processing stage (see Section 8).

4. Edge Bundling Problems

In this paper, three edge bundling problems are investigated. The problems tightly combine Definition 1 with some of the common aesthetics and compatibility measures.

Problem 1 (Angle-Based Edge Bundling Problem (ABEB))**.** *Suppose a drawing D of a graph $G = (V, E)$, a function $\gamma_{e_j e_k}$ that returns the smaller angle between any pair of adjacent edges e_j and e_k from E in the drawing, and an angle α, $0 \leq \alpha \leq 180^\circ$ are given (We define the smaller angle as the smallest angle between two adjacent edges with regard to the clockwise and counter-clockwise orientations). The* ***Angle-Based Edge Bundling Problem*** *(was denoted by $EB-star_\alpha$ in [1] but, for the sake of clarity, it is denoted by ABEB here) is to determine a decomposition (We borrow the definition of decomposition of a graph G that, according to Arumugam, Hamid and Abraham [20], is "a collection ψ of edge-disjoint subgraphs $H_1, H_2, \ldots, H_r$ of G such that every edge of G belongs to exactly one H_i") of E into disjoint subsets $E_1, E_2, \ldots, E_n$, $E = \cup_{i=1}^{n} E_i$, that minimizes n, subject to the conditions that each E_i induces a star subgraph G_i (i.e, all edges in E_i share a same node), and $\gamma_{e_{ij} e_{ik}} \leq \alpha$ for every pair of edges e_{ij}, $e_{ik} \in E_i$.*

ABEB is the original problem proposed in [1], as mentioned before. Note that a set E_i represents a bundle containing only adjacent edges, and that each edge of the graph appears in exactly one bundle. As an example of constraint satisfaction, the two bundles $E_1 = \{e_1, e_2, e_3\}$ and $E_2 = \{e_1, e_5\}$ do not represent a solution to ABEB because the edge e_1 is in both of them. The objective of ABEB is to minimize n, the number of bundles, i.e., the size of the decomposition of E. This objective can be expressed as:

$$f = n. \tag{1}$$

For a given input graph drawing, there may be many solutions with the same number of bundles that satisfy the angle constraints, but with different set partitions $E_1, E_2, \ldots, E_n$. In order to distinguish between these solutions, the additional objective of maximizing a compatibility parameter between edges is added to the original ABEB problem. The aim is to produce solutions in which the edges in each E_i are as compatible as possible. The level of compatibility is given by a function C that evaluates (possibly using an unbundled drawing of the graph) how similar given edge pairs are, and whether or not they should be bundled together. The precise definition of this new problem is given below.

Problem 2 (Compatibility-Based Edge Bundling Problem (CBEB))**.** *Let D be a given drawing of a graph $G = (V, E)$. For D, let $C(a, b)$ be the* ***measure of compatibility*** *between a pair of edges $a, b \in E$, and let $C_{E_i} = \sum_{p,q \in E_i} C(p, q)$ be the* ***measure of total compatibility*** *of a bundle E_i, defined as the sum of the C values for all pairs of edges in the bundle. The* ***Compatibility-Based Edge Bundling Problem*** *is to determine a decomposition of E into disjoint subsets $E_1, E_2, \ldots, E_n$, $E = \cup_{i=1}^{n} E_i$, which maximizes $C_G = \sum_{i=1}^{n} C_{E_i}$ and minimizes n, subject to the condition that each E_i induces a star subgraph G_i.*

Thus, CBEB has two objectives: to maximize the sum of the compatibilities of the edge bundles and to minimize the number of edge bundles. These objectives may balance each other out when finding a minimal set of bundles. For the purpose of simplification, CBEB was converted into a single-objective problem aimed at maximizing the following weighted sum function:

$$f = w_1 \cdot C_G + w_2 \cdot \frac{1}{n}, \tag{2}$$

where $0 \leq w_i \leq 1$, $i = 1, 2$.

Note that the angle limit α is not a constraint of the problem any more, but angle compatibility is part of the objective function, embedded in C and, consequently, in C_G. Many edge compatibility measures can be included in C. The exact ones that we used in the current work are described in Section 5.

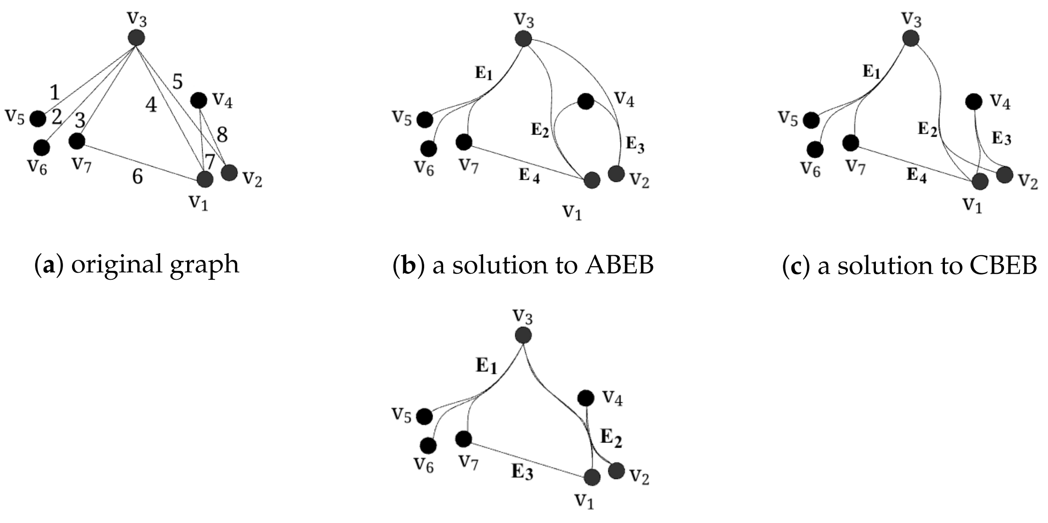
The classic edge bundling methods create bundles by joining the edges at their middle parts, not at their endpoints. However, the solutions to the ABEB and CBEB problems result in graph drawings with only adjacent edges. A more general problem is proposed, in which the adjacency constraint is relaxed. The definition of the resulting problem is given below:

Problem 3 (General-Based Edge Bundling Problem (GBEB))**.** *Let D be a given drawing of a graph $G = (V, E)$. For D, let $C(a, b)$ be the compatibility measure between each pair of edges $a, b \in E$, and let $C_{E_i} = \sum_{p,q \in E_i} C(p, q)$ be the total compatibility of a bundle E_i, defined as the sum of the C values for all pairs of edges in the bundle. The* ***GBEB problem*** *is to determine a decomposition of E into disjoint subsets $E_1, E_2, \ldots, E_n$, $E = \cup_{i=1}^{n} E_i$, that maximizes $C_G = \sum_{i=1}^{n} C_{E_i}$ and minimizes n.*

Comparing the definitions of CBEB and GBEB, one can see that the only aspect that makes the two problems different is the removal of the adjacency constraint in the second problem. However, as it will be discussed in the Section 7.4, this simple modification increases the search space of the feasible solutions. The objective function for GBEB is also defined by Equation (2).

Figure 4 illustrates solutions to the problems ABEB, CBEB and GBEB for a small graph, an angle $\alpha \approx 45^\circ$ and a compatibility measure C_G. Figure 4b is a possible solution to the problem ABEB, constrained by the conditions of (i) adjacency, (ii) having only disjoint sets and (iii) angles no more than $\alpha \approx 45^\circ$. The solution consists of the sets of edges $E_1 = \{1, 2, 3\}$, $E_2 = \{4, 7\}$, $E_3 = \{5, 8\}$ and $E_4 = \{6\}$. The solution to the problem CBEB, Figure 4c, although having the same amount of bundles as in (b), groups the edges in more compatible sizes, with $E_1 = \{1, 2, 3\}$, $E_2 = \{4, 5\}$, $E_3 = \{7, 8\}$ and $E_4 = \{6\}$. Finally, a solution to the problem GBEB (Figure 4d) was not restricted by the adjacency constraint and, therefore, has less bundles, with $E_1 = \{1, 2, 3\}$, $E_2 = \{4, 5, 7, 8\}$, $E_3 = \{6\}$. Note that the sets E_i for the three problems are disjoint, but, in the solution to GBEB, the non-adjacent edges 5

(v_2, v_3) and 7 (v_1, v_4) were joined at the same bundle E_2. Despite producing less bundles, solutions to GBEB may be more ambiguous in terms of the identification of the individual edges in each bundle.



(**a**) original graph (**b**) a solution to ABEB (**c**) a solution to CBEB

(**d**) a solution to GBEB

Figure 4. Comparison of possible solutions between the proposed problems.

5. Compatibility and Aesthetics

The terms "compatibility" and "aesthetics" suggest preferable features that a layout should possess, in order to help to reduce visual clutter. These features are important aspects since they determine the constraints and objectives of the problems being investigated. In the next two subsections, some measures of such features are described and some aesthetics for edge bundling are discussed.

5.1. Compatibility Measures

Generally, edge bundling techniques that decide which edges will be bundled together are based on compatibility measures. There are several measures including: geometric, semantic, importance, topology, neighborhood, temporal and connectivity [5,6,21–23].

The proposed evolutionary formulation focuses on the geometric measures introduced by Holten and Wijk [5], which are: *angle compatibility*, $C_a \in [0,1]$, that avoids joining perpendicular edges; *scale compatibility*, $C_s \in [0,1]$, that prevents joining edges that differ in length; *position compatibility*, $C_p \in [0,1]$, which avoids joining edges that are far apart; and *visibility compatibility* $C_v \in [0,1]$, which avoids joining edges that diverge in projection overlap. Holten and Wijk define the *total compatibility*, $C(P,Q) \in [0,1]$, between two edges P and Q as:

$$C(P,Q) = C_a(P,Q) \cdot C_s(P,Q) \cdot C_p(P,Q) \cdot C_v(P,Q). \tag{3}$$

We do not employ the latter four measures in the CBEB problem. In particular, the measures of position and visibility are left out. This is because they are, in some ways, covered by the adjacency constraint. As a result, only the angular and scale components were used to compute the total compatibility measure C.

For the problem GBEB, however, any edges of E can be joined. Thus, the complexity of selecting compatible edges increases. In order to ensure an acceptable level of compatibility, all four geometric measures proposed in [5] were used for this problem. In addition, a new compatibility measure $C_d \in [0,1]$, termed *distance compatibility*, is proposed to guarantee that pairs edges beyond a certain distance are not joined together. The new measure associates the distance between the midpoints of two edges P, Q with a threshold parameter:

$$C_d(P,Q) = 1 - \frac{||P_m - Q_m||}{k}, \tag{4}$$

where k is the Euclidean distance between the leftmost and rightmost nodes of the graph in the given input drawing D. The values P_m and Q_m are the midpoints of the two edges P and Q, respectively.

The measure $C_d(P,Q)$ was included in the total compatibility (Equation (3)) as a product term. It is important to note that the aim of this measure is similar to the objective of using C_p. Nevertheless, in our preliminary experiments, its inclusion in C allowed a more refined result, with bundles being visually more pleasing. Defining a single distance compatibility measure that combines aspects of both C_p and C_d is left as a topic for future research.

5.2. Aesthetics for Edge Bundling

A graph drawing method usually aims to produce drawings that are considered visually pleasing according to specific aesthetic criteria, for example, minimizing the number of edge crossings or maximizing the number of symmetries [24]. However, most of the known heuristics for edge bundling are not necessarily involved with the exploration and evaluation of the quality of a drawing with respect to such aesthetics.

There are few reported attempts to incorporate and optimize aesthetic criteria in edge bundling layouts. Recently, Angelini et al. [25] restricted edge bundling to the end segments of edges and allowed at most one crossing per bundle. Alam, Fink and Pupyrev [26] proposed a formulation focused on minimizing the number of bundled crossings for circular graphs but did not provide an analysis of its complexity, nor an approximation algorithm for it. Saga [27] proposed two measures to quantitatively evaluate edge bundling: edge lengths and area occupation. The effectiveness of those measures was not discussed.

We believe that, besides reducing clutter, an edge bundling technique could potentially improve the readability of a graph drawing if it was constructed to optimize one or more aesthetic criteria related to the bundling structure. As a result, edge bundling could possibly be formulated as a multi-objective optimization problem, where the layout is generated according to a given set of aesthetics. Some of the following edge bundling aesthetics criteria could possibly be used:

(i) minimizing the total number of bundles;
(ii) maximizing the compatibility of bundled edges;
(iii) maximizing the number of edges per bundle;
(iv) minimizing the ambiguity of the edges;
(v) maximizing the axial symmetry of the bundles;
(vi) minimizing the total number of crossings between edges and/or bundles.

As described before, the present paper investigates the first two aesthetics listed above as part of the objective function of the proposed edge bundling problems.

6. Evolutionary Edge Bundling (EEB)

One of the challenges of dealing with difficult combinatorial optimization problems is to develop algorithms that guarantee to find a reasonably good solution in an acceptable computational time. An approach that has frequently been able to address this challenge successfully in many situations is the so-called evolutionary algorithm (EA), a generic population-based optimization meta-heuristic from artificial intelligence that uses mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, and selection. An evolutionary algorithm [28] is a type of evolutionary computation that can employ a variety of solution representations and evolutionary steps.

Following that line, we present a novel EA for the edge bundling problems ABEB, CBEB and GBEB. The algorithm adopts the standard evolutionary cycle involving population initialization, evaluation, selection, recombination and mutation steps. The next subsections describe the solution representation and present details of the steps of the EA.

6.1. Representation Scheme

Given a graph $G = (V, E)$, the representation of an edge bundling solution (also called here an individual of a population), for problems ABEB, CBEB or GBEB, is simply $I = (E_1, E_2, \ldots, E_n)$, that is, the sets of bundles of edges $E_i \subset E$ $(1 \le i \le n)$, $1 \le |E_i| \le |E|$. The cardinality of individuals is variable, but cannot exceed a given maximum cardinality, $n \le |E|$. Constraint satisfaction is an integral part of the concept of a genetic operator. Therefore, we assume that the sets E_i are disjoint and that $E = \cup_{i=1}^{n} E_i$ is as specified by ABEB, CBEB or GBEB. Figure 5 illustrates the representation of a simple bundled graph with four bundles E_1, E_2, E_3 and E_4. The bundle E_1, for example, is composed of the edges $1, 2$ and 3. The routing of the edges and other visual aspects of the bundling are not included in our current solution representation. These elements are treated automatically by a rendering function at the last stage of the edge bundling process.

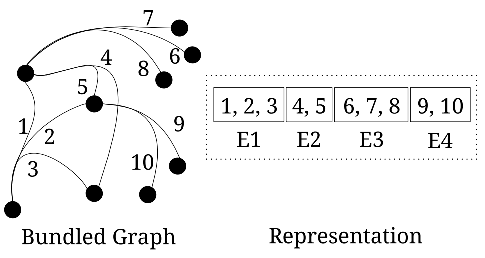


Figure 5. Individual representation scheme.

6.2. Fitness Function

The quality of each individual is evaluated according to a fitness function. This function is simply a maximization version of the objective function required to be optimized in either ABEB, CBEB or GBEB (see Section 4). For instance, the fitness function for ABEB was defined as $f = 1/n$. For CBEB and GBEB, the f function was adapted in order to further penalize solutions with bundles having very low compatibility values, as described in Section 6.7.

6.3. Initialization

In order to explore the search space of points that are distributed as evenly as possible, two methods for generating the initial population were devised. The first method is a heuristic based on solutions for the minimum vertex cover problem [29], while the second method creates individuals randomly using pseudo-random numbers. Each method generates half of the initial population.

In the first strategy, a minimal vertex cover subset A of V (see Algorithm 1) is generated by a heuristic process. Then, each node $v \in A$ is considered as the center of an induced star subgraph in G. Finally, edge bundling sets are created, each taking edges from a randomly chosen star subgraph. The edges that are members of a bundle are also chosen at random, but only compatible edges can be joined together. This approach was found to be effective, usually generating individuals near the optimal solution (related to the number of bundles) for populations of large size. However, preliminary experiments showed that a population that was created by using only this method can lead to premature convergence.

The second strategy produces individuals by randomly choosing adjacent edges to form bundles, while ensuring feasibility. This approach may result in uninteresting individuals, but it facilitates an increase in population diversity, helping to alleviate the problem of premature convergence.

Algorithm 1 The Verter Cover Algorithm

```
procedure VERTER-COVER($G$)
    $A \leftarrow \{\}$
    $V' \leftarrow V(G)$
    Sort $V'$ by decreased degree
    while $V'$ is not empty do
        Let $v$ be the next node of $V'$
        E′ ← Adjacent edges of $v$
        while E′ is not empty do
            Let $(u, v)$ be an arbitrary edge of $E'$
            Remove from $E'$ every edge incident on either $u$ or $v$
        end while
        $A \leftarrow A \cup \{v\}$
        Sort $V'$ by decreased degree
    end while
    return $A$
end procedure
```

After an individual is generated, both strategies enable the positions of the bundles to be shuffled. We found that this improves the efficiency of the crossover operator, often creating more interesting random individuals and preventing premature convergence.

If any individual is found to be duplicated, a removal procedure is run to replace or discard it. We use the approach of [30] in which any duplicated individual is replaced probabilistically with a mutated version of the best individual presented in the population. After applying a test, if the individual is still duplicated, then it is discarded. A checksum (based on the number of bundles, on the reference of edges present in each bundle and on the fitness value) is used for quickly checking the solution. The checksum is invariant with regard to the position of the bundles in the representation.

6.4. Selection

The strategy used for randomly selecting individuals for recombination is tournament selection with replacement. In addition, after recombining two individuals and mutating their two offspring, the steady state is applied, where the produced offspring compete for survival against the members of the current population. This is done by inserting the newborn offspring in the current population and removing the worst two individuals.

The above process is repeated t times, where t is the number of individuals in the population. Then, the best individual of the last generation is propagated to the new one, if it has a higher fitness. If this happens, the best individual randomly replaces a solution in the resultant population.

6.5. Crossover Operator

The one-point and two-point classic methods are applied (they are chosen randomly each time) to produce the recombination of the parental sets of bundles. This is accomplished by swapping the list of bundles of two parents in order to create a new random set I, representing two new individuals. First, the crossover points are selected.

In the one-point crossover, the selected point is at the middle part of the smallest parent (the smallest n value divided by two). The right-hand side of the parents are then swapped (see Figure 6). In the two-point crossover, the points are also defined by the size of the smallest individual, dividing this individual into three equally-sized parts. The middle parts are then swapped.

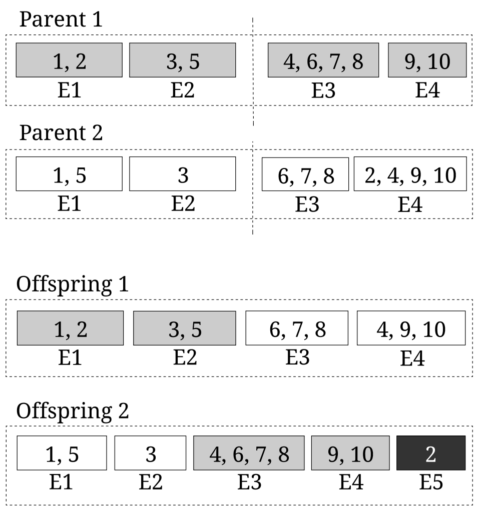


Figure 6. A example of the one point crossover operation.

During the crossover process, infeasible offspring (that violate the disjoint-set constraint or do not cover all edges) may be produced and must be repaired. The repair procedure involves removing duplicated edges (the last ones that were included) and inserting missing edges (these edges are added to new solitary bundles). This is illustrated in Figure 6, where edge 2 was removed from bundle E_4 in Offspring-1 and the same edge was inserted in a new bundle in Offspring-2.

6.6. Mutation Operators

The mutation operators specified for EEB are:

- **Join mutation** randomly selects two solitary bundles and merges them if their edges are adjacent.
- **Merge mutation** is similar to Join mutation. The difference is that it works with bundles of any size and only the first bundle is chosen at random. The other merged bundle is the first compatible one found by a sequential search in the representation of the individual.
- **Split mutation** replaces a randomly chosen bundle by dividing it into two sub-bundles.
- **Move mutation** randomly selects a bundle and moves one of its edges to the first compatible bundle found.
- **Remove mutation** randomly removes an edge from a bundle of size greater than one and creates a new solitary bundle with that edge.

The mutation operation to be used is chosen at random at each time. There is also a constant probability of applying it.

6.7. Penalty Scheme

The EEB approach uses the formulas defined in the Section 5 to calculate the compatibility C of pairs of edges (for the CBEB and GBEB problems). The total bundle compatibility, denoted by C_{E_i}, is defined as the sum of the compatibilities of all pairs of edges in the bundle. Nevertheless, this sum might not be realistic as it may hide pairs of edges with low compatibility in a bundle, if other high-compatibility edges are present.

A threshold-based penalty is proposed to solve this problem. The threshold T controls the penalty applied to solutions having pairs of edges with low compatibility. The value T is defined as

$$T = T_a \cdot T_s \cdot T_p \cdot T_v \cdot T_d, \tag{5}$$

where T_a, T_s, T_p, T_v and T_d are compatibility threshold constants set as the lowest acceptable values for C_a, C_s, C_p, C_v and C_d, respectively.

Assuming that $\min\_C_i$ is the smallest compatibility $C(p,q)$ among all pairs p and q in E_i, the total bundle compatibility C_{E_i} is now computed or penalized as follows:

$$C_{E_i} = \begin{cases} \sum C(p,q), \forall p,q \in E_i, & \text{if} \quad \min\_C_i \geq T, \\ p_e, & \text{if} \quad \min\_C_i < T, \\ 0, & \text{if} \quad |E_i| = 1, \end{cases} \tag{6}$$

with $p_e < 0$ a penalization constant. The compatibility of the graph, C_G, is defined as before.

The values of the compatibility thresholds and p_e are configurable by the user, who can control the final graph drawing. The possibility of controlling characteristics of the solution is one of the benefits of formulating edge bundling problems as optimization problems.

6.8. EEB for GBEB Problem

For the GBEB problem, the lack of an adjacency constraint necessitates some adaptations in the EEB algorithm. In particular, the population initialization process and the operators of mutation have to be changed. The population initialization now involves three methods, each creating 1/3 of the initial population.

The first method uses the same algorithm for initialization guided by the minimal vertex cover set, proposed in Section 6.3. The second method is similar to the random method presented in that section, but, in this case, the constraints such as adjacency, vertex cover and compatibility do not restrict the edges to be grouped. The third method is a new approach. It randomly chooses edges to be joined by considering only the main compatibility measures (edge adjacency is neglected). Five separate solutions are generated, each satisfying one of the compatibility thresholds: T_a, T_s, T_p, T_v and T_d.

The mutation operators described in Section 6.6 are also partially changed to remove the adjacency constraint. Furthermore, to prevent premature convergence, the version of EEB for GBEB employs an adapted multi-population model. Two different subpopulations are maintained, having independent evolution. After each evolutionary cycle, the best individual of each subpopulation is migrated to the other subpopulation.

7. Experiments

Experiments for testing the EEB approach were conducted with one synthetic graph and nine real-world graphs. The graphs are: G1—Synthetic (20 nodes, 28 edges) [1]; G2—ZacharyClub (34 nodes, 78 edges) [31]; G3—PlanarGD2015 (66 nodes, 101 edges) [32]; G4—Dolphin (62 nodes, 160 edges) [33]; G5—a connected version of MovieLens (160 nodes, 161 edges) [34]; G6—LesMiserables (77 nodes, 254 edges) [35]; G7—BooksUSPolitics (105 nodes, 401 edges) [36]; G8—Word adjacencies (112 nodes, 425 edges) [36]; G9—Flare Software Class (220 nodes, 709 edges) [3]; and G10—the USAirline (235 nodes, 1297 edges) from an unknown source.

We performed three sets of experiments, one for each of the ABEB, CBEB and GBEB problems discussed in Section 4. The initial layout of each graphs used was predefined as an input file. The experiments consisted of running the evolutionary algorithm for 100 independent trials (20 trials for GBEB) for each graph and for some angle parameters. For the ABEB, the angle parameter was progressively fixed as $\alpha = 30°, 45°, 70°$.

For the CBEB problem, the angle parameter was considered, but the calculation of the threshold T_a was carried out using the formula $T_a = 1 - \alpha/180$. The value of the scale compatibility was set at $T_s = 0.890$. The penalty value was fixed to $p_e = -1$.

For the GBEB problem, the angle parameter was also included and the formula of Holten and Wijk [5] was used for calculating T_a. The other compatibility values were empirically determined as $T_s = 0.70$, $T_p = 0.96$, $T_v = 0.72$ and $T_d = 0.96$. Because the choice of small values for the penalty constant p_e had no effect during the experiments, a dynamic scheme was implemented: each bundle was penalized individually according to the following dynamic scheme in (7):

$$p_{e_i} = \begin{cases} \sum C(p,q) \cdot (-3), & \text{if } \sum C(p,q) > 0, \forall\, p,q \in E_i, \\ -3, & \text{if } \sum C(p,q) = 0, \forall\, p,q \in E_i. \end{cases} \tag{7}$$

This is different from the CBEB problem, which determines a fixed value of p_e.

Other parameters defined for the three sets of experiments were: the population size $u = 150$; the crossover rate $p_c = 0.98$; and the mutation rate $p_m = 0.4$. The evolutionary cycle was repeated until there was no further improvement in the population for 500 consecutive iterations or the maximum number of generations (set at 16,500) was achieved. All tests for ABEB and CBEB were executed on a MacBook Pro (McIntosh Laboratory, Inc., Binghamton, New York, USA) with an Intel Core i7 processor (Intel Corporation, Santa Clara, California, USA) of 2.9 GHz and 8 GB of 1600 MHz-DDR3 RAM, and, due to the long processing time needed to solve an instance, for the GBEB problem, was used a more powerful computer a DELL M630 server (Dell Technologies, Round Rock, TX, USA) with 128 GB of RAM and two processors with 10 cores each and hyperthreading, resulting in 40 visible cores of 3–3.6 Ghz.

7.1. Results for the ABEB Problem

The aim of the first experiment was to produce solutions with the minimum number of bundles, with all edge angles in a bundle never higher than a given α. Table 1 summarizes the results for the various graphs.

The first three columns consist of general information. The fourth column shows the number of bundles of the best solution in 100 trials. The number of trials in which a solution with that property appeared is presented in parentheses. The fifth and the sixth columns present the average values of the number of bundles of the best solutions, and the average of their fitness, respectively, over the 100 trials. The seventh and the eighth columns are the standard deviation and the standard error of the fitness values. The last column shows the average of the total runtime.

Upon analyzing the table, it can seen that the average of the number of bundles in the best solutions was usually the best one found over the independent trials. In addition, the standard deviations were low, indicating that a majority of the generated solutions are positioned close to the mean fitness. Figure 7 illustrates the best solution obtained for the graph G5 with angle constraints $\alpha = 30°$ and $\alpha = 70°$. A comparison between Figure 7a,b shows the effect of the angle constraint on the bundling. Higher α angles lead to more edges being joined together. This is very noticeable for the edges connected to the highlighted nodes (drawn as lighter circles).

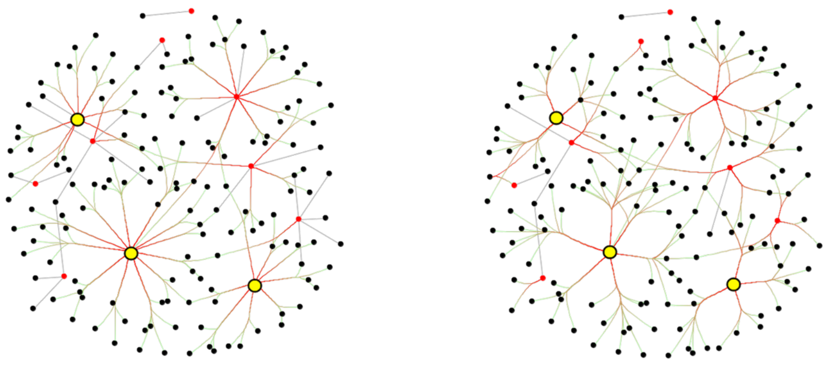

(**a**) (**b**)

Figure 7. MovieLens (G5) for ABEB with different angle constraints: (**a**) angle $\alpha = 30°$; (**b**) angle $\alpha = 70°$.

Table 1. Results for the ABEB problem, averaged over 100 independent runs, $u = 150$, $p_c = 0.98$ and $p_m = 0.4$.

Graph	Edges	$\alpha(^\circ)$	Bundles of Best Solution	Avg. Bundles	Avg. Fitness	Std. Dev. Fitness	Std. Error Fitness	Avg. Time (s)
G1	28	30	17 (100)	17.00	0.0588	0.00000	0.000000	2
		45	16 (100)	16.00	0.0625	0.00000	0.000000	1
		70	13 (89)	13.11	0.0763	0.00173	0.000173	1
G2	78	30	45 (1)	49.74	0.0201	0.00073	0.000073	6
		45	37 (3)	39.75	0.0252	0.00081	0.000081	6
		70	31 (6)	33.30	0.0301	0.00101	0.000101	5
G3	101	30	75 (4)	76.84	0.0130	0.00015	0.000015	9
		45	69 (1)	73.01	0.0137	0.00023	0.000023	9
		70	60 (4)	62.63	0.0160	0.00028	0.000028	9
G4	160	30	107 (4)	111.08	0.0090	0.00019	0.000019	20
		45	92 (6)	96.57	0.0104	0.00026	0.000026	19
		70	75 (2)	79.48	0.0126	0.00032	0.000032	18
G5	161	30	65 (2)	69.28	0.0144	0.00028	0.000028	11
		45	51 (1)	54.17	0.0185	0.00050	0.000050	11
		70	37 (1)	40.84	0.0245	0.00085	0.000085	10
G6	254	30	121 (4)	128.23	0.0078	0.00022	0.000022	156
		45	99 (1)	107.13	0.0093	0.00032	0.000032	40
		70	82 (4)	88.01	0.0114	0.00039	0.000039	29
G7	401	30	238 (1)	251.82	0.0040	0.00009	0.000009	101
		45	207 (1)	217.45	0.0046	0.00010	0.000010	88
		70	169 (2)	178.71	0.0056	0.00014	0.000014	66
G8	425	30	217 (1)	230.19	0.0043	0.00010	0.000010	103
		45	189 (1)	199.94	0.0050	0.00012	0.000012	85
		70	150 (1)	164.54	0.0061	0.00017	0.000017	71
G9	709	30	339 (1)	354.39	0.0028	0.00006	0.000006	198
		45	280 (3)	293.35	0.0034	0.00008	0.000008	167
		70	235 (2)	247.62	0.0040	0.00009	0.000009	167
G10	1297	30	338 (1)	355.85	0.0028	0.00008	0.000008	524
		45	281 (1)	295.32	0.0034	0.00009	0.000009	451
		70	221 (1)	236.72	0.0040	0.00012	0.000012	420

In general, when solving the ABEB, the EEB method usually produces good results in terms of the number of bundles and the α angle constraint. On the other hand, some bundles may have edges that differ significantly in length. Therefore, as expected, the ABEB usually produces bundles with low scale compatibility between the edges (see Figure 8).

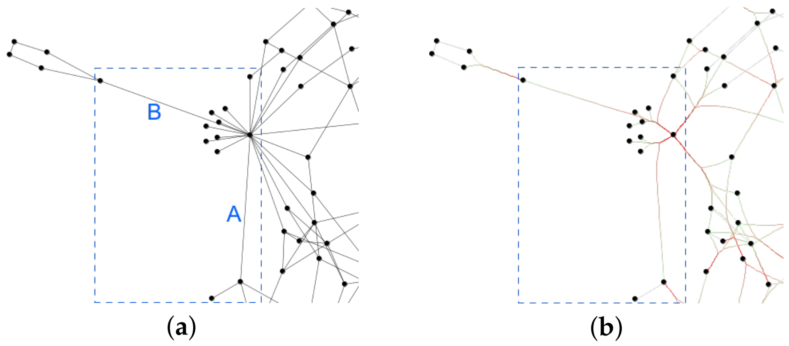


Figure 8. Bundles with edges that differ significantly in length: (**a**) original graph; (**b**) graph with bundles.

7.2. Comparison of EEB and an Exact Approach for the ABEB Problem

We now establish a qualitative point of view of the complexity of the edge bundling problems discussed in this paper and attempt to show that the EEB algorithm is a promising approach. To do this, we present (in terms of number of bundles and runtime execution) the best solution (over 100 independent executions) obtained by EEB for the ABEB problem for the graphs Synthetic (1), ZacharyClub (G2) and PlanarGD2015 (G3). We then compare numerical results with those generated by an exact method based on an integer programming (IP) model for the ABEB problem, published in [1]. The exact method was implemented using the Gurobi solver [37] and a more powerful computer (a DELL M630 server with 128 GB of RAM and two processors with 10 cores each and hyperthreading—resulting in 40 visible cores of 3–3.6 Ghz). The EEB method was executed on the simpler computer previously described. More α angles were used this time. The results are reported in Table 2. The fourth and fifth columns present the number of bundles in the best solutions found by EEB and by the exact IP method, respectively. A line with a dash means that no solution was produced by the IP method even after several hours of execution.

Table 2. Comparison of number of bundles generated by EEB with an exact method (IP) for ABEB.

Graph	Edges	$\alpha(°)$	EEB	IP	Time EEB (s)	Time IP (s)
G1	28	30	17	17	2	0.31
		45	16	16	1	0.41
		70	13	13	1	0.35
		90	11	11	1	0.17
		110	10	10	1	0.17
G2	78	30	45	45	6	12,070.30
		45	37	36	6	1103.77
		70	31	29	5	135.931
		90	27	25	4	445.188
		110	25	25	4	801.322
G3	101	30	75	74	9	125.65
		45	69	69	9	656.14
		70	60	58	9	1210.29
		90	53	50	9	1275.27
		110	48	-	9	-

The values in this table show that the IP method was effective in generating the optimal solutions only for small numerical instances and took an inordinate amount of time to solve some of the larger instances. For the graph G3 with 101 edges, the IP method failed to find a solution with a maximum angle of 110°. In general, the exact method is not practical for medium-sized ABEB and CBEB problems, as it requires excessive computation time.

Conversely, the EEB approach was usually successful in finding close-to-optimal solutions. In addition, the execution time of EEB is significantly lower than that of the IP method for the larger graphs, even when using a less powerful computer.

7.3. Results for the CBEB Problem

The second experiment was conducted in order to improve the earlier results by introducing the compatibility criteria of the CBEB problem. Table 3 summarizes the experimental results for this problem. It follows the same structure as Table 1 and includes one new column representing the average compatibility measure of the best solutions in 100 trials (for the USAirline graph, G10, fewer trials were conducted, due to the long processing time needed to solve this instance). The stochastic nature of the EEB algorithm associated with the size of the search space led to solutions significantly different from each other in terms of fitness. However, the standard error reflects a low sampling fluctuation.

Note that similar results did not occur for the USAirline graph, first because the experiment was repeated for only 15 independent trials and also because the number of edges significantly increased the size of the search space. These results highlight how the number of edges can dramatically increase the set of candidate solutions for the CBEB problem.

Figure 9b shows the best solution produced by EEB for the LesMiserables (G6) graph for the CBEB problem. Note that, although the number of bundles has increased, the solution for CBEB has bundles with more similar edges than the solution produced for ABEB (Figure 9a). The figures show that some of the bundles in the ABEB solution were split and some different bundles are part the CBEB solution.

As the CBEB problem has multiple objectives, best solutions to it are not always those with the least number of bundles. This is because the final solution depends on the weights in the objective function and on the compatibility values.

Table 3. Results for the CBEB problem, averaged over 100 independent runs, $u = 150$, $p_c = 0.98$, $p_m = 0.4$, $w_1 = 0.2$, $w_2 = 0.8$ and $T_s = 0.890$ (*G10 was used in 15 trials).

Graph	Edges	$\alpha(°)$	Bundles of Best Solution	Avg. Compa-Tibility	Avg. Bundles	Avg. Fitness	Std. Dev. Fitness	Std. Error Fitness	Avg. Time (s)
G1	28	30	20	9.48	20.01	1.935	0.026	0.003	2
		45	16	13.27	18.17	2.700	0.299	0.030	2
		70	16	20.22	16.82	4.093	0.248	0.025	2
G2	78	30	56	22.69	58.49	4.552	0.246	0.025	11
		45	48	35.87	50.72	7.190	0.490	0.050	13
		70	36	69.97	39.10	14.015	0.861	0.086	15
G3	101	30	81	23.23	83.86	4.655	0.336	0.034	12
		45	76	34.94	81.10	6.999	0.808	0.081	13
		70	65	84.31	69.05	16.874	0.618	0.062	18
G4	160	30	129	29.49	132.78	5.904	0.407	0.041	31
		45	112	50.48	119.24	10.102	0.669	0.067	39
		70	89	108.68	94.89	21.744	1.451	0.145	47
G5	161	30	76	136.71	83.90	27.305	2.185	0.219	33
		45	60	218.55	69.79	43.721	4.240	0.424	36
		70	50	387.42	52.88	77.500	5.803	0.580	35
G6	254	30	174	121.04	180.17	24.212	2.086	0.209	103
		45	137	206.14	147.32	41.233	3.009	0.301	111
		70	97	426.76	101.97	85.361	4.747	0.475	101
G7	401	30	305	170.56	312.62	34.114	1.687	0.169	300
		45	260	266.62	270.62	53.328	2.489	0.249	315
		70	209	501.12	213.89	100.227	4.689	0.469	363
G8	425	30	303	148.28	314.20	29.659	1.871	0.187	209
		45	267	241.85	274.03	48.373	3.121	0.312	240
		70	208	472.18	213.04	94.440	6.095	0.610	273
G9	709	30	452	410.16	464.50	82.034	4.589	0.459	684
		45	379	663.31	395.09	132.663	4.627	0.4631	834
		70	312	1108.79	318.49	221.761	9.519	0.952	828
G10*	1297	30	530	2152.49	536.33	430.500	15.501	4.002	19,184
		45	428	3249.19	437.40	649.840	22.675	5.855	11,110
		70	318	5774.74	327.73	1154.950	57.564	14.863	6859

7.4. Results for the GBEB Problem

The results for the GBEB problem are reported in Table 4. The fourth column of the table shows the number of bundles in the best solution. It can be seen that the number of bundles in solutions to GBEB is usually lower than in solutions to CBEB. Fewer trials were conducted due to the long processing time needed to solve this instance, emphasizing the complexity of the GBEB problem. The best solution was obtained from 20 trials and using a more powerful computer as described above.

For the BooksUSPolitics (G7) and Flare (G9), the experiments were repeated only for five independent trials. The results for USAirline graphs (G10) are related to the best solution obtained from one trial.

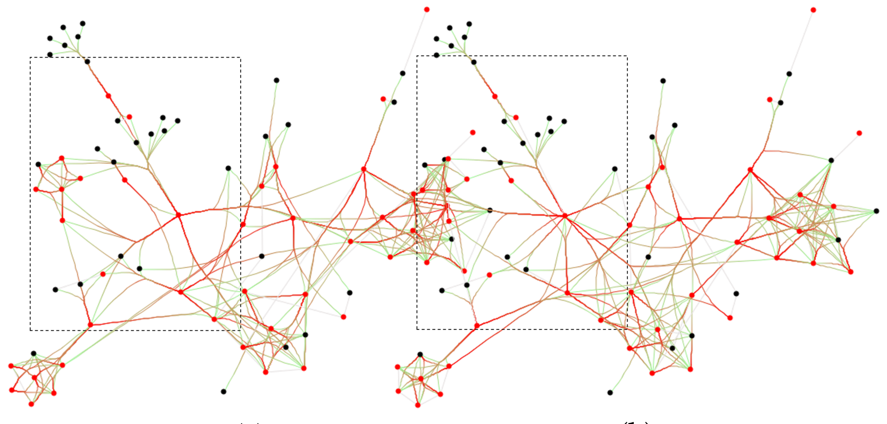


Figure 9. LesMiserables (G6) with $\alpha = 70°$: (**a**) ABEB solution with 82 bundles; (**b**) CBEB solution with 97 bundles.

Table 4. Results for the GBEB problem, averaged over 20 independent runs, $u = 150$, $p_c = 0.98$, $p_m = 0.4$, $w_1 = 0.4$, $w_2 = 0.6$, $T_s = 0.70$, $T_p = 0.96$, $T_v = 0.72$ and $T_d = 0.96$ (*G7 and *G9 were used in 5 trials, and G10** was used in 1 trial).

Graph	Edges	$\alpha(°)$	Bundles of Best Solution	Avg. Compa-Tibility	Avg. Bundles	Avg. Fitness	Std. Dev. Fitness	Std. Error Fitness	Avg. Time (s)
G1	28	30	12	189.54	12.65	75.863	5.595	1.251	30
		45	14	45.28	15.20	18.153	0.774	0.173	22
		70	16	36.96	16.05	14.821	0.564	0.126	22
G2	78	30	27	445.86	28.85	178.367	20.645	4.616	128
		45	38	94.77	39.35	37.923	3.245	0.725	312
		70	40	73.44	42.35	29.390	3.319	0.742	153
G3	101	30	48	402.42	49.45	160.981	13.570	3.034	198
		45	63	80.59	68.05	32.243	4.185	0.936	194
		70	67	56.12	72.95	22.456	4.026	0.900	185
G4	160	30	59	614.27	66.60	245.719	31.456	7.034	584
		45	89	122.76	96.10	49.111	8.323	1.861	686
		70	101	83.82	107.45	33.535	5.270	1.178	645
G5	161	30	46	1641.42	50.75	656.579	70.408	15.744	526
		45	62	378.49	67.65	151.405	17.442	3.900	511
		70	70	276.21	73.65	110.494	11.425	2.555	509
G6	254	30	88	1482.56	92.75	593.030	83.764	18.730	1670
		45	126	314.21	129.10	125.687	15.674	3.505	1899
		70	140	222.38	145.55	88.956	13.924	1.501	2154
G7*	401	30	160	2303.84	164.40	921.541	24.908	8.303	3351
		45	242	413.07	250.00	165.232	12.064	4.021	4374
		70	267	258.51	294.00	103.401	27.584	9.195	3955
G8	425	30	133	2527.92	137.15	1011.172	75.617	16.910	3032
		45	199	502.24	208.95	200.900	9.550	2.135	3991
		70	216	379.37	230.20	151.750	7.549	1.689	4215
G9*	709	30	294	2919.86	297.00	1167.947	52.315	23.396	6381
		45	429	594.10	435.20	237.641	9.467	4.234	7565
		70	461	446.09	473.40	178.438	10.286	4.600	8501
G10**	1297	30	404	12,801.08	404.00	5129.434	-	-	9489
		45	619	2359.84	619.00	943.937	-	-	12,503
		70	723	1481.55	723.00	592.623	-	-	13,558

The high standard deviation for the average of the fitness shows that the data are widely spread, especially for the smaller angles, with 20 independent trials performed.

The average of the runtime (reported in the last column) is much higher when compared to the other problems investigated in this paper. In particular, the results appear to imply that the removal of the adjacency constraint increases the difficulty of the problem.

The drawing of the best generated solution is consistent with the predetermined thresholds. Figures 10 and 11 display the solutions for the PlanarGD2015 and Word Adjacencies graphs, respectively. Note that the solutions have bundles with both adjacent and non-adjacent edges. Varying the maximum angle caused the sets of bundles and the overall structure of the drawing to be significantly different. In addition, the level of ambiguity has increased. For example, it is difficult to individually track the edges joined in the bundles.

7.5. Comparison of the Results for the Proposed Problems

Figures 12 and 13 compare the results of the three experiments. Figure 12 illustrates the average number of bundles in the best solution for each problem and given angle constraint parameter. In general, the solutions to the CBEB problem have less bundles than ABEB and the number of bundles in the solutions to GBEB problem is lower for the smaller graphs, with some exceptions when the angles $\alpha = 45°$ and $\alpha = 70°$.

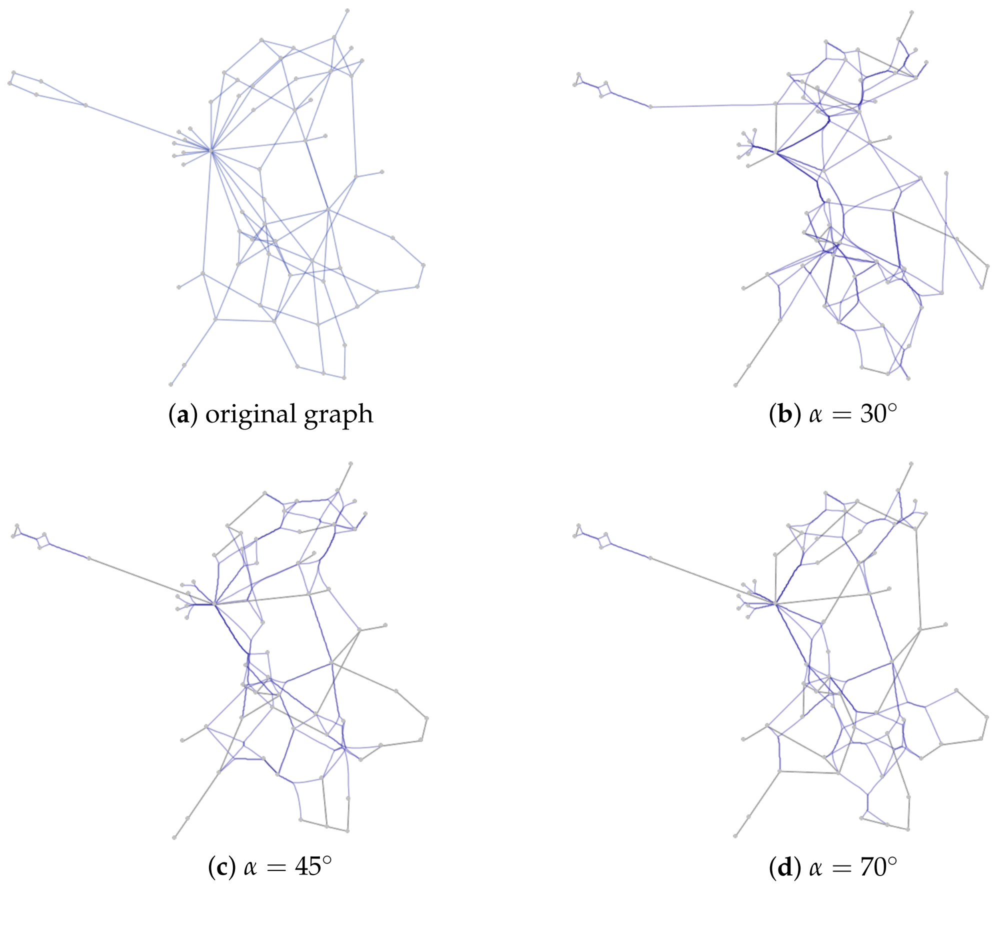
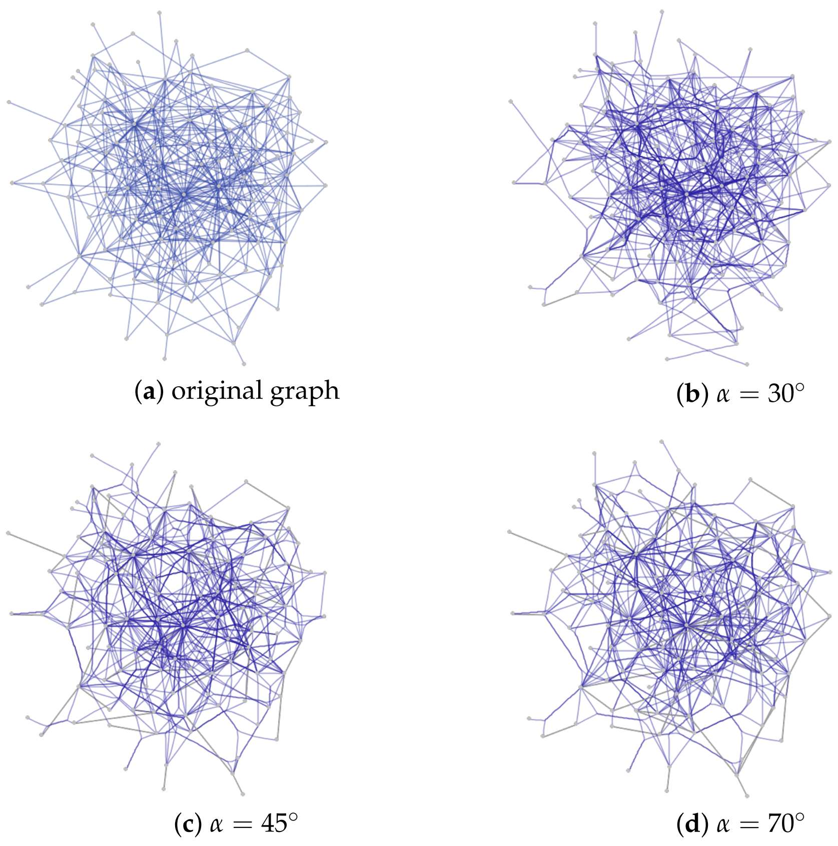


Figure 10. Effects of the angle in the Planar graph (G3) for the GBEB problem.


(**a**) original graph (**b**) $\alpha = 30°$

(**c**) $\alpha = 45°$ (**d**) $\alpha = 70°$

Figure 11. Effects of the angle in the Word Adjacencies (G8) for the GBEB problem.

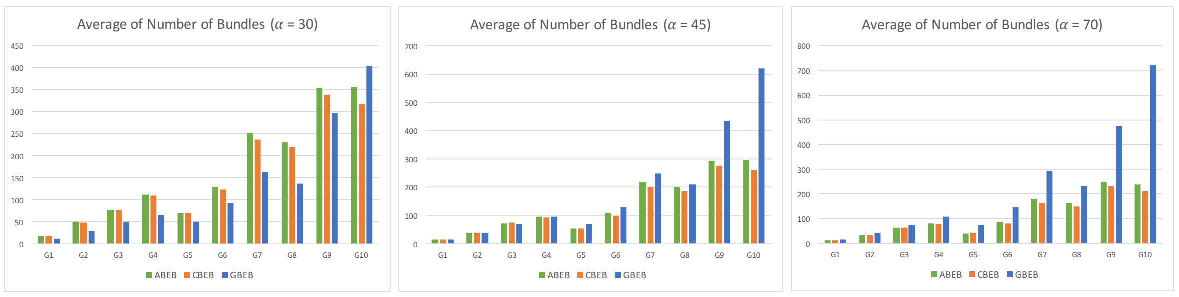


Figure 12. Comparison of average of number of bundles of the best solution for each problem and angle parameter.

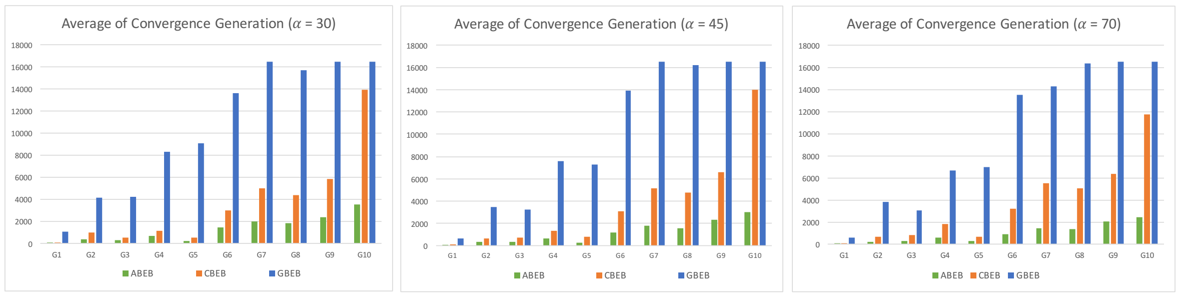
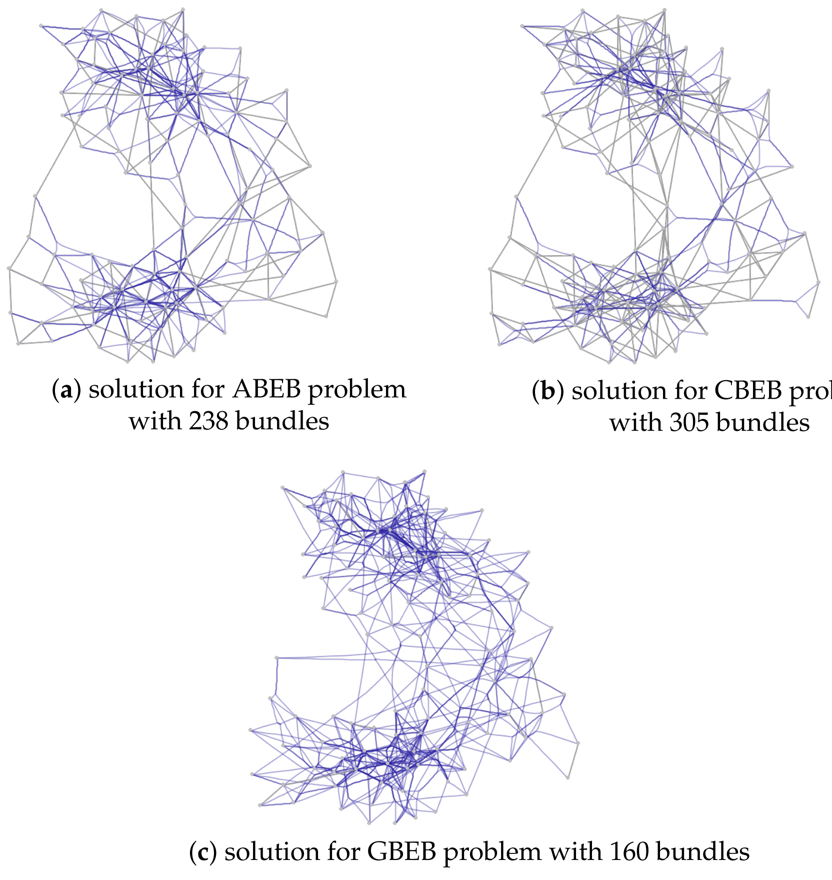


Figure 13. Comparison of average of convergence generation for each problem and angle parameter.

Figure 13 shows the average convergence, in terms of number of iterations. The ABEB problem was solved most quickly, with the GBEB problem requiring significantly more time for the ten graphs tested. As already mentioned, this behavior is a reflection of both the evolutionary method implemented (which differs from the earlier methods by using multiple populations) and the relative complexities of the problems.

Figure 14 depicts the solutions that were found for the graph BookUSPolitics with $\alpha = 30^\circ$ by implementing the EEB approach with the overall objective of minimizing the number of bundles. Note that the solutions to the three problems ABEB, CBEB and GBEB are quite different from each other. This is no surprise as ABEB prioritizes the angle constraint, CBEB optimizes compatibility measures and GBEB allows bundles with non-adjacent edges.

It is possible to introduce other objectives to the edge bundling optimization problems such as: minimizing the amount of clutter (or maximizing the white area), minimizing the number of edge crossings, minimizing the number of bundles with a single edge, etc. The purpose of this is to produce solutions according to what the user desires. However, it would be necessary to use a multi-objective optimization approach in order to optimize multiple objectives simultaneously.


(**a**) solution for ABEB problem with 238 bundles

(**b**) solution for CBEB problem with 305 bundles

(**c**) solution for GBEB problem with 160 bundles

Figure 14. Comparison of solutions of BookUSPolitics graph (G7) and $\alpha = 30^\circ$ for each problem.

Another interesting observation relates to the importance of defining an effective routing method for edge bundling. Even though the solution to the GBEB problem drawn in (Figure 14c) has a relatively low number of bundles, the drawing is more cluttered than the drawings of the other solutions. In the present paper, the EEB approach does not focus on, or attempt to optimize, the trajectory (routing) of the edges directly. Instead, the final edge bundling drawing is generated by a standard post-processing procedure that draws each bundle individually. As a consequence, larger bundles that intersect with several other bundles may make the drawing confusing. Possible ways of addressing this issue include:

using a better rendering function that considers the interaction between bundles and routes them rationally, increasing the weight of the compatibility criteria in the objective function in order to avoid long edges, or including edge crossing as a additional objective to be minimized in the edge bundling optimization problem.

7.6. Effects of Parameters at the Final Solution

Various parameters can be set up in the EEB approach. We believe that one of the benefits of investigating edge bundling as an optimization problem is the possibility that users may be able configure the parameters they consider important in order to find the “best solution” according to their needs. Later in the present section, we provide an analysis of the effect of the existence of multiple parameters in the final bundling solution.

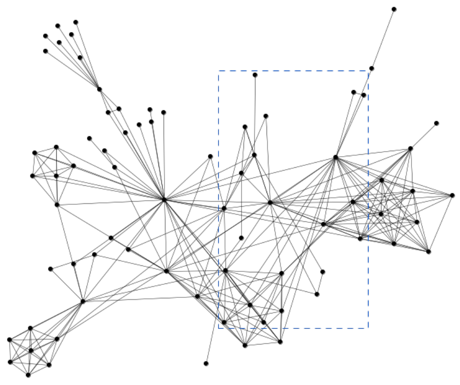
In general, in multi-objective problems, there is no single solution that is globally optimal, since there are multiple, possibly conflicting objectives [38]. This is the case with the CBEB and GBEB problems. Minimizing the number of bundles may mean worsening the compatibility value between the edges in a bundle. This is the kind of scenario that requires the intervention of a decision maker, through a precise definition of the search criteria. We attempted to address this issue by transforming the CBEB and GBEB problems into single-objective problems via weighted, linear functions whose weights reflect user preferences.

The weights w_1 and w_2 of the components of the objective function (see Equation (2)), for the CBEB and GBEB, were chosen empirically. Changes to these weights may affect the quality of solutions obtained, in terms of number of bundles and level of compatibility. Figure 15 shows various solutions to the CBEB problem for the graph LesMiserables, generated by varying the weight coefficients. The rectangles mark areas that have a different set of bundles. Various weights were tried and it was found that $w_1 = 0.2$ and $w_2 = 0.8$ achieved the best visual results for CBEB and $w_1 = 0.4$ and $w_2 = 0.6$ for GBEB.

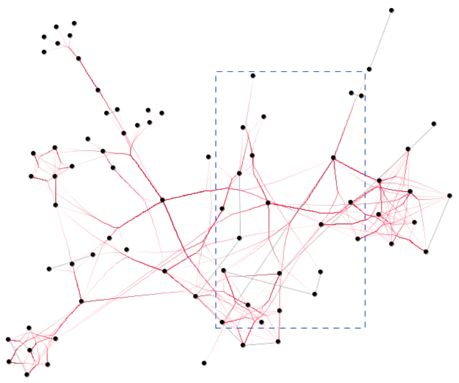
Another aspect that helps guide the search towards an optimal solution is the definition of thresholds for the compatibility measures, including α and part of the threshold T_α. From the results shown in Tables 1, 3 and 4, it can be seen that varying the angle parameter generated completely different numerical results and different drawings (see Figure 7). This shows that it is possible, via parametrization of the compatibility values, to give users the power to decide which type of drawing they want. For example, Figure 16c–e shows three different configurations for the graph USAirlines (G1) for the GBEB problem with different α parameters. Note that the visual solutions produced are not far from the one created by the MINGLE approach [18] (Figure 16b). However, varying the parameter generates different structural aspects of the graph drawing. We did not formalize detailed experiments involving the other thresholds, but preliminary tests with them demonstrated the possibility of controlling other aspects of the results by exploring those thresholds. The parameters described in the Section 7 were defined empirically and based on the personal perception of the authors.

Finally, the parameters of the evolutionary method are characteristics of the optimization approach and may be useful in determining both the speed of convergence and population diversity. The parameters of an evolutionary algorithm are the control inputs that enable the search to be as comprehensive and as fast as possible [28,39]. For the EEB approach, the values were chosen during an extensive preliminary testing process. We believe that adapting the values to the type of graph could have produced better results for the larger graphs. However, we have chosen to be as pragmatic as possible, keeping the set of parameters constant in all tests, so as to ensure that the results can be meaningfully compared.

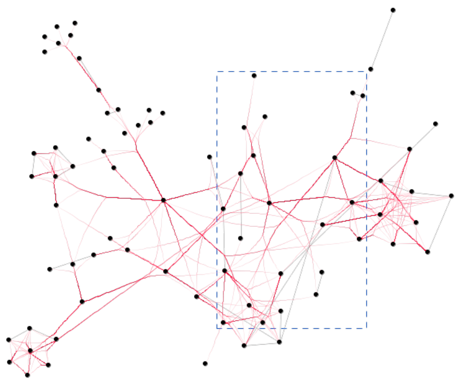
As expected, the results of the experiments indicate that different parameterization produces distinct solutions. In order to illustrate the importance of parameterization in edge bundling approaches Policis et al. [40] investigated a search for "optimal" parameters for edge bundling algorithms.



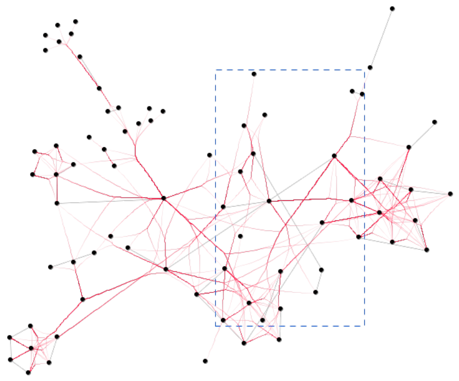
(**a**) original graph



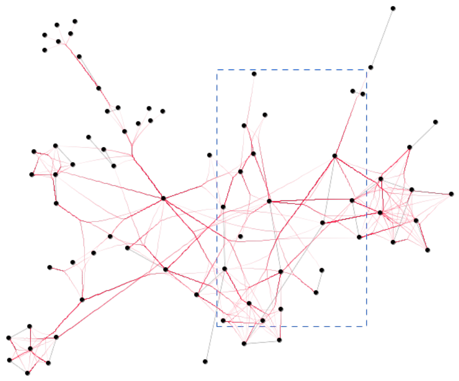
(**b**) $w_1 = 0.2$, $w_2 = 0.8$, bundles = 97 and compatibility = 479.29



(**c**) $w_1 = 0.8$, $w_2 = 0.2$, bundles = 93 and compatibility = 501.90

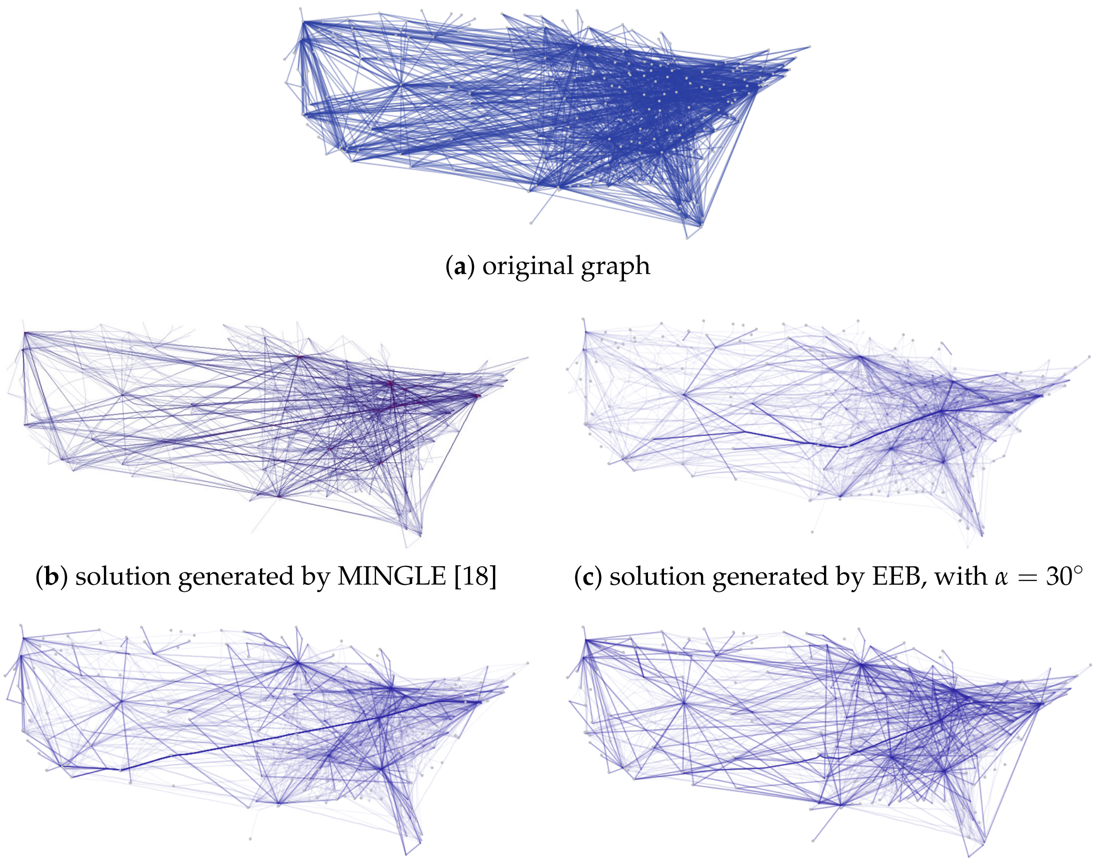


(**d**) $w_1 = 0.4$, $w_2 = 0.6$, bundles = 96 and compatibility = 486.70



(**e**) $w_1 = 0.6$, $w_2 = 0.4$, bundles = 95 and compatibility = 504.63

Figure 15. Effects of weight in the LesMiserables graph (G6), CBEB problem and $\alpha = 70°$.



(**a**) original graph

(**b**) solution generated by MINGLE [18]

(**c**) solution generated by EEB, with $\alpha = 30°$

(**d**) solution generated by EEB, with $\alpha = 45°$

(**e**) solution generated by EEB, with $\alpha = 70°$

Figure 16. Comparison of solutions for USAirline graph (G10), GBEB problem.

8. Visualization

Most existing edge bundling methods draw edges individually and then route them. In contrast, the EEB approach only outputs sets of edges representing bundles. Those groups of edges form star subgraphs. After the execution of the evolutionary algorithm, the subgraphs of the best-found solution can then be submitted to a rendering module to produce an edge bundling drawing.

In order to allow an adequate visual interpretation of the EEB results (especially the comprehension of the relational patterns between the nodes), two visualizations are now proposed. The first one shows bundled edges as a colored cubic Bézier curve, going from red to green by default (see Figures 7 and 9). The red color represents the source (the central node of the bundles) and the destination is colored green. Single edges that constitute a bundle by themselves are drawn as straight lines and colored light gray. Nodes that correspond to a vertex cover set of the graph are colored red and the others are colored black. Other combinations of colors could be used as we can see in the figures throughout this paper.

The rendering/routing method of the visualization is a specialized implementation of the force-directed edge bundling (FDEB) approach of Holten and Wijk [5] that receives the bundles as input and draws them individually. The version of the FDEB was implemented as defined by the authors. The EEB approach creates a list of bundles and passes it to the rendering function, which draws each subgraph induced by set of bundled edges individually. For example, if the final solution has 10 bundles, the FDEB will be run 10 times. The level of ambiguity may vary in the final drawing, depending on the parameterization made for the FDEB (number of iterations, etc). This is only a visual aspect, independent of the choice of bundles.

The force-based algorithm attempts to ensure that the individual axial symmetry of each bundle looks plausible. However, overall graph symmetry is sampled poorly for some graphs because the algorithm does not identify the route of the bundles to maximize symmetry (see Figures 7 and 9). This visualization is identified as the "primary view". It allows the user to identify connections between nodes and to trace individual edges.

Even though the just mentioned visualization process will reveal relational patterns at the node level, for graphs with a high level of clutter, edge crossings can still impede their legibility. To reduce the number of edge crossings and highlight the relational patterns, we followed the strategy of [2,41] that uses the idea of partial edge drawings. Thus, the drawing of a curve is divided into three parts, and the middle part is drawn using a partial transparency. Figure 17d illustrates this visualization. In addition, the parts of an edge incident with its endpoints can be colored using different colors. Various aspects (such as colors, transparency level and highlight options) can be chosen by the user. Such choices promote smooth visualization by enabling the tracking of individual edges. This is similar to the layout process used by the "sideknot" approach [2] (see Figure 17c).

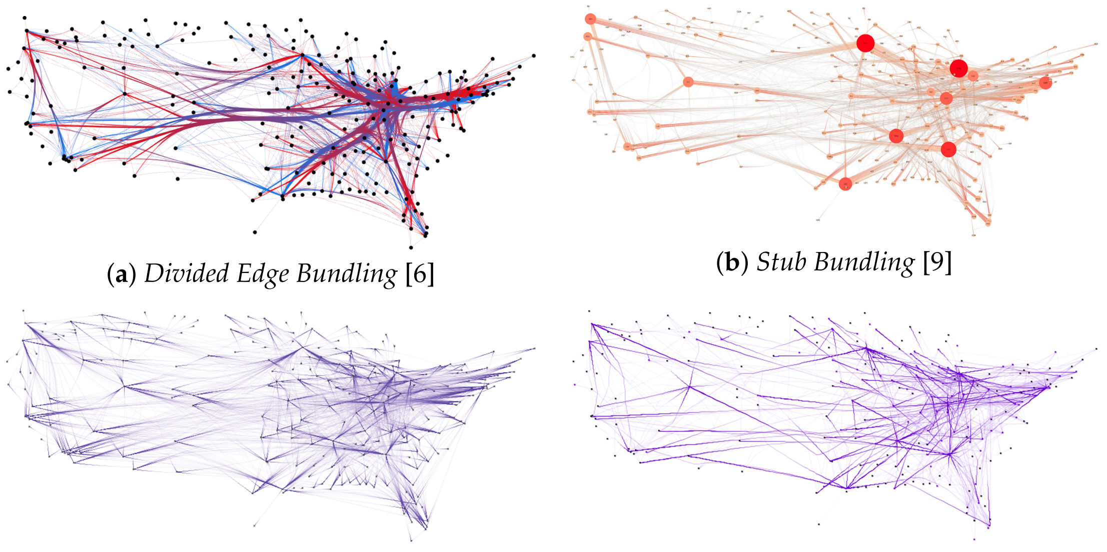

(**a**) *Divided Edge Bundling* [6]
(**b**) *Stub Bundling* [9]
(**c**) *SideKnot* [9]
(**d**) EEB with partial edges, $\alpha = 70°$.

Figure 17. Comparison of solutions for USAirlines (G10).

9. Discussion

As was mentioned earlier, there is a significant difference between most classical edge bundling methods and the approach proposed here. The earlier methods attempt to create drawings with reduced clutter. Some of these methods focus on minimizing a particular aspect of the drawing that can be mathematically formalized, for example, reducing ink used or the number edges that cross. However, they do not search for an optimal solution, but instead, merely employ heuristic methods to produce solutions [42]. On the other hand, our approach involves an optimization algorithm with the purpose of finding high-quality solutions for a formal and precise problem definition. Thus, comparing the EEB approach with previous methods in an informative manner is not straightforward, since the other approaches are not based on an explicit and well-defined edge-bundling optimization problem.

In addition, as can be observed in Figure 17a, methods based on force (such as *divided edge bundling* [6]) produce layouts that are significantly different from the ones created by EEB because they do not focus on showing the connection trends of a node. It is hard to establish which nodes are connected to each other in those layouts. Some exceptions are the *stub bundling* [9] and the *sideknot* [2] approaches (Figure 17b,c), which are more similar to EEB. They join only adjacent edges with the aim of indicating directions at the endpoints of the edges, highlighting node-level connections and tracing individual edges. Those methods, however, are not designed for an optimization-based problem as there is no formal mathematical definition of what is being investigated.

Furthermore, if we attempt to formally define the implicit problem studied by those techniques, then differences with the proposed approach are found. For example, the stub bundling approach attempts to find sets only containing adjacent edges with respect to the following constraints: "the angle between any two half-edges in a bundle must be at most α, and the angle between two consecutive edges in a bundle has to be at most γ" [9]. This is similar to the ABEB problem except that the edges are half-edges, i.e., they can be shared by two bundles and there is no clear attempt to minimize the number of edge bundles.

A quantitative comparison of the efficacy of earlier methods in terms of the number of bundles is not possible, since the traditional approaches usually do not provide that information. A visual comparison of Figure 17b–d reveals the efficiency of the EEB approach for the CBEB problem, with threshold $\alpha = 70°$. The algorithm generated 241 bundles with more than one edge and 77 single-bundled edges. It is clear that the group of bundles is markedly different from the solutions generated by the stub bundling and sideknot methods since, in our approach, they are formed with respect to model constraints. Overall, the experimental results suggest that the main disadvantage of the EEB framework is its long running time for large graphs. As with most population-based search algorithms, the running time for EEB is influenced by the population size, the number of generations, the size of the graph and other parameters. The evolutionary parameters were adjusted to attempt to find a near-optimal solution, which greatly increased the running time.

Possible runtime reduction strategies that might be implemented in the future include: performing recombination without the repair procedure (which would imply changing the representation of the individual solutions), using only the genetic operator of mutation, or updating the fitness function only for what has been changed in the individual solutions.

10. Conclusions

This paper describes a new approach to edge bundling via an approximate evolutionary algorithm (EBB). We examined maximum angle-based edge bundling that restricts angles between adjacent edges, compatibility-based edge bundling that group edges with compatible directions and lengths and general edge bundling that joined adjacent and non-adjacent edges in the same bundle. A previously defined combinatorial optimization problem (ABEB) and two new problems (CBEB and GBEB) were discussed and solved using the evolutionary algorithm. As far as we know, this is the first proposal of an evolutionary algorithm for edge bundling modeled as a combinatorial optimization problem. The algorithm was tested on a number of graphs and found to be efficient at finding near-optimal solutions when the goal is mainly to create bundled graphs with the minimum number of bundles and with only adjacent edges being joined. Furthermore, this appears to be a promising approach to the challenge of grouping non-adjacent edges. Many other edge bundling problems, based on extensions of the present work, can be devised and studied using EEB. Future research could focus on: extending the fitness function to include new compatibility and aesthetic criteria; investigating a new rendering function that improves the routing of the edges belonging to a bundle; and conducting some user-controlled studies in order to check whether the solutions for the proposed optimization-based edge-bundling problems are visually appealing or effective for users.

Author Contributions: This article is based in part, on the doctoral dissertation of the first author, J.d.M.F., who successfully completed a PhD in The Institute of Informatics, Federal University of Goiás, Goiás, Brazil, under the supervision of the second and third authors, H.A.D.d.N. and L.R.F. The contributions to the paper are as follows. Conceptualization, J.d.M.F., H.A.D.d.N. and L.R.F.; Methodology, J.d.M.F., H.A.D.d.N. and L.R.F.; Software, J.d.M.F.; Validation, J.d.M.F., H.A.D.d.N. and L.R.F.; Formal Analysis, J.d.M.F., H.A.D.d.N. and L.R.F.; Investigation, J.d.M.F., H.A.D.d.N. and L.R.F.; Resources, H.A.D.d.N.; Data Curation, J.d.M.F.; Writing—Original Draft Preparation, J.d.M.F.; Writing—Review & Editing, J.d.M.F., H.A.D.d.N. and L.R.F.; Visualization, J.d.M.F.; Supervision, H.A.D.d.N. and L.R.F.; Project Administration, H.A.D.d.N.; Funding Acquisition, H.A.D.d.N.

Funding: This research was funded by FAPEG-Brazil, grant number 25725.

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

References

1. Ferreira, J.M.; Nascimento, H.A.D.; Quigley, A.J.; Foulds, L.R. Computational Complexity of Edge Bundling Problems. Technical Report, Federal University of Goiás. 2017. Available online: http://inf.ufg.br/biblioteca-digital (accessed on 25 June 2018).
2. Peng, D.; Lu, N.; Chen, W.; Peng, Q. SideKnot: Revealing Relation Patterns for Graph Visualization. In Proceedings of the 2012 IEEE Pacific Visualization Symposium, Songdo, Korea, 28 February–2 March 2012; pp. 65–72.
3. Holten, D. Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data. *Trans. Vis. Comput. Graph.* **2006**, *12*, 741–748. [CrossRef] [PubMed]
4. Cui, W.; Zhou, H.; Qu, H.; Wong, P.; Li, X. Geometry-based Edge Clustering for Graph Visualization. *Trans. Vis. Comput. Graph.* **2008**, *14*, 1277–1284. [CrossRef] [PubMed]
5. Holten, D.; van Wijk, J. Force-directed Edge Bundling for Graph Visualization. *Comput. Graph. Forum* **2009**, *28*, 983–990. [CrossRef]
6. Selassie, D.; Heller, B.; Heer, J. Divided Edge Bundling for Directional Network Data. *Trans. Vis. Comput. Graph.* **2011**, *17*, 2354–2363. [CrossRef] [PubMed]
7. Ersoy, O.; Hurter, C.; Paulovich, F.; Cantareiro, G.; Telea, A. Skeleton-based Edge Bundling for Graph Visualization. *Trans. Vis. Comput. Graph.* **2011**, *12*, 2364–2373. [CrossRef] [PubMed]
8. Hurter, C.; Ersoy, O.; Telea, A. Graph Bundling by Kernel Density Estimation. *Comput. Graph. Forum* **2012**, *31*, 865–874. [CrossRef]
9. Nocaj, A.; Brandes, U. Stub Bundling and Confluent Spirals for Geographic Networks. In Proceedings of the 21st International Symposium on Graph Drawing, Bordeaux, France, 23–25 September 2013; pp. 388–399.
10. McKnight, R.L. Low-Stretch Trees for Network Visualization. Ph.D. Thesis, University of British Columbia, Vancouver, BC, Canada, 2015.
11. Lhuillier, A.; Hurter, C.; Telea, A. State of the Art in Edge and Trail Bundling Techniques. *Comput. Graph. Forum* **2017**, *36*, 619–645. [CrossRef]
12. Beck, F.; Puppe, M.; Braun, P.; Burch, M.; Diehl, S. Edge Bundling without Reducing the Source to Target Traceability. In Proceedings of the 2011 15th International Conference on Information Visualisation, London, UK, 13–15 July 2011; pp. 298–305.
13. Čermák, M.; Dokulil, J.; Katreniaková, J. Edge Routing and Bundling for Graphs with Fixed Node Positions. In Proceedings of the 2011 15th International Conference on Information Visualisation, London, UK, 13–15 July 2011; pp. 475–481.
14. Luo, S.J.; Liu, C.L.; Chen, B.Y.; Ma, K.L. Ambiguity-free Edge-bundling for Interactive Graph Visualization. *IEEE Trans. Vis. Comput. Graph.* **2012**, *18*, 810–821. [PubMed]
15. Hurter, C.; Puechmorel, S.; Nicol, F.; Telea, A. Functional Decomposition for Bundled Simplification of Trail Sets. *IEEE Trans. Vis. Comput. Graph.* **2018**, *24*, 500–510. [CrossRef] [PubMed]
16. Gansner, E.R.; Koren, Y. Improved Circular Layouts. In Proceedings of the 14th International Symposium on Graph Drawing, Karlsruhe, Germany, 18–20 September 2006; pp. 386–398.
17. Telea, A.C.; Ersoy, O. Image-based Edge Bundles: Simplified Visualization of Large Graphs. *Comput. Graph. Forum* **2010**, *29*, 843–852. [CrossRef]
18. Gansner, E.R.; Hu, Y.; North, S.; Scheidegger, C. Multilevel Agglomerative Edge Bundling for Visualizing Large Graphs. In Proceedings of the 2011 IEEE Pacific Visualization Symposium, Hong Kong, China, 1–4 March 2011; pp. 187–194.
19. Yamashita, T.; Saga, R. Cluster-based Edge Bundling Based on a Line Graph. In Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, Porto, Portugal, 27 February–1 March 2017; pp. 311–316.
20. Arumugam, S.; Hamid, I.S.; Abraham, V.M. Decomposition of Graphs into Paths and Cycles. *J. Discret. Math.* **2013**, *2013*, ID 721051. [CrossRef]
21. Kienreich, W.; Seifert, C. An Application of Edge Bundling Techniques to the Visualization of Media Analysis Results. In Proceedings of the 2010 14th International Conference Information Visualisation, London, UK, 26–29 July 2010; pp. 375–380.

22. Nguyen, Q.H.; Hong, S.; Eades, P. TGI-EB: A New Framework for Edge Bundling Integrating Topology, Geometry and Importance. In Proceedings of the 19th International Symposium on Graph Drawing, Eindhoven, The Netherlands, 21–23 September 2011; Volume 7034, pp. 123–135.
23. Nguyen, Q.; Edges, P.; Hong, S.H. StreamEB: Stream Edge Bundling. In *Lecture Notes in Computer Science, Proceedings of the 20th International Symposium on Graph Drawing, Redmond, WA, USA, 19–21 September 2012*; Didimo, W., Patrignani, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7704, pp. 400–413.
24. Battista, G.; Eades, P.; Tamassia, R.; Tollis, I.G. *Graph Drawing: Algorithms for the Visualization of Graphs*, 1st ed.; Prentice Hall: Englewood Cliffs, NJ, USA, 1998.
25. Angelini, P.; Bekos, M.; Kaufmann, M.; Kindermann, P.; Schneck, T. 1-Fan-Bundle-Planar Drawings of Graphs. In Proceedings of the 24th International Symposium on Graph Drawing, Athens, Greece, 19–21 September 2016; pp. 634–636.
26. Alam, M.J.; Fink, M.; Pupyrev, S. The Bundled Crossing Number. In Proceedings of the 24th International Symposium on Graph Drawing, Athens, Greece, 19–21 September 2016; pp. 399–412.
27. Saga, R. Quantitative Evaluation for Edge Bundling by Difference of Edge Lengths and Area Occupation. In Proceedings of the 18th International Conference, HCI International 2016, Toronto, Canada, 17–22 July 2016; pp. 287–290.
28. Bäck, T. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*; Oxford University Press: Oxford, UK, 1996.
29. Skiena, S. *Minimum Vertex Cover. § 5.6.2 in Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*; Addison-Wesley: Reading, MA, USA, 1990; p. 218.
30. Saroj, D. A Non-revisiting Genetic Algorithm for Optimizing Numeric Multi-dimensional Functions. *Int. J. Comput. Sci. Applic.* **2012**, *2*, 83–93. [CrossRef]
31. Zachary, W. An Information Flow Model for Conflict and Fission in Small Groups. *J. Anthropol. Res.* **1977**, *33*, 452–473. [CrossRef]
32. ISGCI. Information System on Graph Classes and Their Inclusions. 2015. Available online: http://www.csun.edu/gd2015/topics.htm (accessed on 2 November 2017).
33. Girvan, M.; Newman, M. Community Structure in Social and Biological Networks. *Proc. Natl. Acad. Sci. USA* **2002**, *99*, 7821–7826. [CrossRef] [PubMed]
34. MovieLens. 2017. Available online: http://www.eecs.wsu.edu/~yyao/ (accessed on 2 November 2017).
35. Knuth, D. *The Stanford GraphBase: A Platform for Combinatorial Computing*; Addison-Wesley: Reading, MA, USA, 1993.
36. Newman, M.E.J. Modularity and Community Structure in Networks. *Proc. Natl. Acad. Sci. USA* **2006**, *103*, 8577–8582. [CrossRef] [PubMed]
37. Optimization, G. Gurobi Optimizer Quick Start Guide. 2017. Available online: http://www.gurobi.com/documentation/ (accessed on 14 May 2017).
38. Collette, Y.; Siarry, P. *Multiobjective Optimization: Principles and Case Studies*; Springer: Berlin/Heidelberg, Germany, 2003.
39. Ahn, C.W. *Advances in Evolutionary Algorithms: Theory, Design and Practice*; Springer: Berlin/Heidelberg, Germany, 2006.
40. Polisciuc, E.; Cruz, A.; Machado, P.; Arrais, J.P. On the Role of Aesthetics in Genetic Algorithms Applied to Graph Drawing. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO'17), Berlin, Germany, 15–19 July 2017; pp. 1713–1720.
41. Bruckdorfer, T.; Cornelsen, S.; Gutwenger, C.; Kaufmann, M.; Montecchiani, F.; Nöllenburg, M.; Wolff, A. *Progress on Partial Edge Drawings*; Springer: Berlin/Heidelberg, Germany, 2013.
42. Pupyrev, S.; Nachmanson, L.; Kaufmann, M. Improving Layered Graph Layouts with Edge Bundling. In Proceedings of the 19th International Symposium on Graph Drawing, Konstanz, Germany, 21–24 September 2011; pp. 329–340.