An Evolutionary Algorithm for an Optimization Model of Edge Bundling

Joelma Ferreira, Hugo Nascimento and Les Foulds

Institute of Informatics, Federal University of Goiás, Alameda Palmeiras, Campus Samambaia, Goiânia, Brazil

Keywords: Edge Bundling, Optimization, Evolutionary, Approximate, Algorithm.

Abstract: This paper presents two edge bundling optimization problems that address minimizing the total number of bundles, in conjunction with other aspects, as the main goal. A novel evolutionary edge bundling algorithm for these problems is described. The algorithm was successfully tested by solving two 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 formally defined as optimization models.

1 INTRODUCTION

A graph is a mathematical structure for representing inherent relationships between 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.

Traditional edge bundling approaches are not problem-specific methods but instead, are application-oriented. They are not dedicated to solve 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.

The consequence is that those 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.

Conversely, in a previous paper (Ferreira et al., 2017), edge bundling was formally defined and investigated as an optimization problem for the first time. Specifically, it was formulated as a constrained combinatorial optimization problem in which the total number of bundles is to be minimized and only edges with a common node[1] can be bundled together. The problem of bundling only adjacent edges appears to provide a better sense of proximity between edges and nodes and has been shown to be NP-hard. A more advanced problem was also defined by imposing a maximum angle (α) constraint between any pair of adjacent edges to be bundled. Therefore, the problem became minimizing the number of edge bundles while respecting the α-angle constraint. It is termed the Angle-Based Edge Bundling Problem (ABEB). Ferreira et al. (2017) highlights how complex and relevant edge bundling optimization is.

In the present paper, we continue the investigation into the formal modeling of edge bundling and propose a third problem. The new problem is denoted as the Compatibility-Based Edge Bundling Problem (CBEB) and it involves minimizing the total number of bundles while maximizing a multi-objective function that incorporates well-known edge compatibility measures. We also propose an approximate evolutionary algorithm, called here Evolutionary Edge Bundling (EEB), for ABEB and CBEB. As far as we know, EEB is the first heuristic method ever proposed for these problems[2].

Experiments with the evolutionary algorithm show that it produces close-to-optimal solutions for some of the nontrivial ABEB instances tested. The results reinforce the importance of optimization approaches for edge bundling, not only as a way of visualizing a graph with less visual clutter, but also as a

[1]Edges with a common node are termed "adjacent" and those without are termed "nonadjacent'.

[2] Ferreira et al. (2017) presented only an integer linear formulation for the ABEB, not a solution method.

Ferreira, J., Nascimento, H. and Foulds, L.
An Evolutionary Algorithm for an Optimization Model of Edge Bundling.
DOI: 10.5220/0006626901320143
In *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2018) - Volume 3: IVAPP*, pages 132-143
ISBN: 978-989-758-289-9

way 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 (Ferreira et al., 2017). Some compatibility and aesthetic criteria used as constraints and as part of the objective function of our problems are then discussed in Section 4. Section 5 provides the formal definitions of ABEB and CBEB. Section 6 introduces the evolutionary edge bundling algorithm aimed at solving these two problems and Section 7 describes the experimental results. Section 8 discusses a method (EEB) for rendering the edge bundling solutions as a final step of 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, the approach developed by Holten (2006), called *hierarchical edge bundling*, uses hierarchy tree branches for edge routing; in (Cui et al., 2008), a method called *geometric edge bundling* forms bundles by routing edges along a mesh generated by a triangulation algorithm; the *force-directed edge bundling* technique (Holten and van Wijk, 2009; Selassie et al., 2011) splits the edges into segments that attract each other as a basis for bundling edges; the *skeleton-based edge bundling* method (Ersoy et al., 2011) 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 (Hurter et al., 2012) computes a density map and moves the edges in the gradient direction of where the bundles will be formed.

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 et al. (2012) 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 happens because 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 et al. (2012) proposed an algorithm called *node-based edge bundling* that bundles and "knots" edges nearer to the common node of adjacent edges. Nocaj and Brandes (2013) refined the technique of (Peng et al., 2012) and proposed a method called *stub bundling* that joins only edges that share the same endpoint with the aim of visualizing unambiguous graphs and retrieving the exact source and target of each edge. Their method uses the angle between consecutive edges as the criterion for choosing which edges to be joined together.

Even though the methods of Peng et al. and Nocaj and Brandes 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, the formulation of an optimization problem is required, defining decision variables, constraints and objectives. In order to address the aforementioned need, the present paper investigates the problem of finding the "best" configuration of bundles (involving only adjacent edges) that maximizes a given objective function.

3 EDGE BUNDLING AS AN OPTIMIZATION PROBLEM

There is a present 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 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 (2015) and Lhuillier et al. (2017) have discussed those complementary aspects when presenting 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 decision of how to segment the edges. Lhuillier et al. (2017) define a *bundle* as a set of paths that share similarities and *edge bundling* as the method that creates bundles and trails.

McKnight (2015) affirms that many existing edge bundling approaches do not generate bundles directly. In fact, for most edge bundling algorithms, the output is just a graph drawing and the search for a good layout is a heuristic based on an informal or implicit definition of the desirable optimization goals.

Consequently, we take a different direction by formally defining edge bundling as an optimization problem. We also aim at a solution that has a bundle configuration as an output. For the structure of the solution, we follow the definition given by McKnight that focus on computing bundles directly, but we consider a bundle as a set of edges (not a set of edge segments as McKnight does).

Thus, in (Ferreira et al., 2017), 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 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 $S = (E_1, E_2, ..., E_n)$ a partition of E (not necessarily disjoint), $n \in \mathbb{N}^+$. Further, let R be a function that takes G, D and S, and renders a bundled graph drawing version of D called D^R, given some extra necessary information, such as rules for routing the edges. The general edge bundling problem is hence to determine the partition 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 (minimized or maximized), 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[3] as a question to be addressed by the optimization problem. This can be done in two ways: (1) by solving 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 that 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 to 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 amount 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).

[3]Defining the precise paths on which the bundled and possibly the unbundled edges travel through.

4 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 of the EEB framework, since they determine the constraints and objectives of the problems being investigated. In the next two subsections, some measures used in the evolutionary algorithm for obtaining such features are detailed, and some aesthetics for edge bundling are discussed.

4.1 Compatibility Measures

Generally, edge bundling techniques decide which edges will be bundled together (Holten and van Wijk, 2009; Nguyen et al., 2011). The proposed evolutionary formulation focuses on two measures introduced by Holten and van Wijk (2009)[4]: **Angle compatibility**, $C_a \in [0,1]$, that avoids joining perpendicular edges; and **Scale compatibility**, $C_s \in [0,1]$, which prevents joining edges that differ in length. The total compatibility, $C \in [0,1]$, between two edges is $C = C_a \times C_s$.

4.2 Aesthetics for Edge Bundling

A graph drawing method usually tries 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, etc (Battista et al., 1998). 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. (2016) restricted edge bundling to the end segments of edges and allowed at most one crossing per bundle. Alam et al. (2016) proposed a formulation focused on minimizing the number of bundled crossings for circular graphs but no

[4] Holten and van Wijk (2009) proposed two other measures, position and visibility, but they were not explicitly employed in our approach because they are covered by the adjacency constraint.

proof of its NP-completeness, nor an approximation algorithm were presented. Saga (2016) proposed two measures to quantitatively evaluate edge bundling: edge lengths and area occupation. The effectiveness of those measurements was not discussed.

We believe that, besides reducing the 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 formalized as a multi-objective optimization problem, where the layout is generated according to a given set of aesthetics. We propose and attempt to implement strategies for some of the following edge bundling aesthetic criteria:

(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.

5 EDGE BUNDLING PROBLEMS

In this paper, two edge bundling problems are investigated that tightly combine Definition 1 with some aesthetics and compatibility measures.

Problem 1 (Angle-Based Edge Bundling Problem (ABEB)). Given a drawing D of a graph $G = (V,E)$, a function $\gamma_{e_j e_k}$ that returns the smaller angle[5] between any pair of adjacent edges e_j and e_k from E in the drawing, and an angle α, $0 \le \alpha \le 180°$, the *ABEB problem*[6] is to determine a partition of E into disjoint subsets $E_1, E_2, \ldots, E_n$, $E = \cup_{i=1}^{n} E_i$, that minimizes n, subject to each E_i inducing a star subgraph G_i (i.e, all edges in E_i share a same node), and $\gamma_{e_{ij} e_{ik}} \le \alpha$ for every two edges e_{ij}, $e_{ik} \in E_i$.

ABEB is the original problem proposed in (Ferreira et al., 2017). 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.

For a given input graph drawing, there may be many solutions with the same number of bundles and satisfying the angle constraints, but with a different composition of the sets $E_1, E_2, \ldots, E_n$. In order to distinguish between these solutions, the additional objective of maximizing the compatibility value between edges is added to the original ABEB. The aim is to produce solutions in which the edges in E_i are as compatible as possible. The precise definition of this new problem is given below.

Problem 2 (Compatibility-Based Edge Bundling Problem (CBEB)). Let D be a drawing of a graph $G = (V,E)$. For D, let $C(a,b)$ be the compatibility measure between each pair of edges $a, b \in E$ (as previously defined in Section 4.1), and $C_{E_i} = \sum_{p,q \in E_i} C(p,q)$ the total compatibility of a bundle E_i, defined as the sum of C for all pairs of edges in the bundle. The *CBEB problem* is to determine a partition 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, subject to each E_i inducing 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:

$$f = w_1 \times C_G + w_2 \times \frac{1}{n} \tag{1}$$

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

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_G.

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. A technique that has frequently been able to address this challenge successfully in many situations is so-called evolutionary computation, a generic population-based approximate metaheuristic optimization algorithm from artificial intelligence. An evolutionary algorithm (Bäck, 1996) (EA) is a subset of evolutionary computation, An EA uses mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, and selection.

[5]We define the *smaller angle* as the lesser angle between two adjacent edges with regard to the clockwise and counter-clockwise orientations between the edges.

[6]The original name of this problem is $EB{-}star_\alpha$ but, for the sake of clarity, it will be called ABEB here.

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

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 ABEB (or CBEB) is simply $I = (E_1, E_2, \ldots, E_n)$, that is, the sets of bundles of edges $E_i \subset E$ $(1 \leq i \leq n)$, $1 \leq |E_i| \leq |E|$. The cardinality of individuals is variable, but cannot exceed a given maximum cardinality, $n \leq |E|$. 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 $E = \cup_{i=1}^{n} E_i$ as specified by ABEB (CBEB). Figure 1 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.

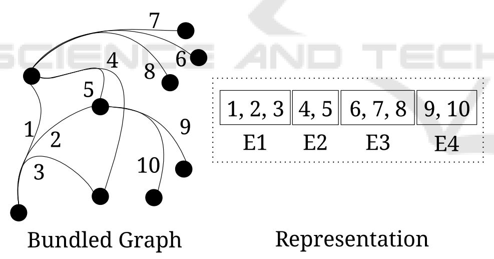


Figure 1: Individual representation scheme.

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 or CBEB (see Section 5). For the CBEB case, an adaptation was done via the f function, in order to further penalize solutions with bundles having very low compatibility values. For this, the thresholds T_a and T_s were defined, representing lowest acceptable values for C_a and C_s respectively. Then, for the pair of edges $p, q \in E_i$ that induces the smallest compatibility $C(p,q)$ among all other pairs in E_i, if $C_a(p,q) < T_a$ or $C_s(p,q) < T_s$ then C_{E_i} is set to -1.

6.3 Initialization

In order to explore the search space at points that are distributed as evenly as possible, two methods for generating the initial population were devised. The first method is a heuristic method based on solutions for the minimum vertex cover problem (Skiena, 1990), while the second algorithm selects the initial population randomly using pseudo-random numbers. Each method generates half of the initial population.

In the first strategy, a minimal vertex cover set A of V 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 one taking edges from a randomly chosen star subgraph. The edges that comprise a bundle are also chosen at random, but only compatible edges, (relative to a given threshold) can be joined together. For the ABEB and CBEB problems, the threshold represents the maximum value of the desired constraints (maximum angle and/or minimum compatibility measures). 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 also showed that a population created using only this method can lead to premature convergence.

In order to address the problem of premature convergence, the other strategy initializes the second half of the population randomly. It creates individuals by randomly choosing adjacent edges to create bundles, while still ensuring feasibility and threshold satisfaction. As a consequence, this method can produce uninteresting individuals but it facilitates an increase in population diversity.

After an individual is generated, both strategies enable bundles to be shuffled in their positions. This improves the efficiency of the crossover operator, creating more interesting random individuals and also preventing premature convergence.

If any individual is found duplicated, a removal function is used to replace or discard it. We use the approach of Saroj (2012) 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 edges present in each bundle, and on the fitness) is used to quickly check 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 offspring produced 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. After repeating this process *t* times, where *t* is the number of individuals in the population, 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 2). In the two-point crossover, the points are also defined by the size of the smallest individual, dividing it in three equall-sized parts. The middle parts are then swapped.

During the crossover process, infeasible offspring (that violate the disjoint-set constraint or do not cover all edges) may be produced and need to be repaired. Thus, the latest occurrence of duplicated edges are removed and missing edges are inserted in new solitary bundles in the final solution representation. This is illustrated in Figure 2, 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.

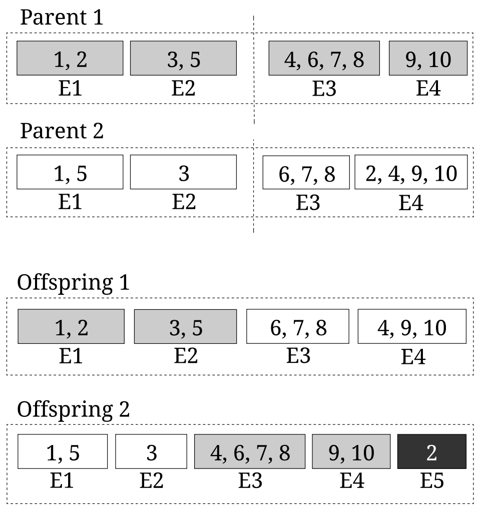


Figure 2: A example of one point crossover operation.

- **Split mutation** replaces a randomly chosen bundle by dividing it into two subbundles.
- **Move mutation** randomly selects a bundle and moves one of its edges to another 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 each time. There is also a constant probability of applying it.

7 EXPERIMENTAL RESULTS

Experiments for testing our approach EEB were conducted with one synthetic graph and eight real-world graphs. The graphs are: Synthetic (G1 – 20 nodes, 28 edges) (Ferreira et al., 2017), ZacharyClub (G2 – 34 nodes, 78 edges) (Zachary, 1977), PlanarGD2015 (G3 – 66 nodes, 101 edges) (ISGCI, 2015), Dolphin (G4 – 62 nodes, 160 edges) (Girvan and Newman, 2002), a connected version of MovieLens (G5 – 160 nodes, 161 edges) (MovieLens, 2017), LesMiserables (G6 – 77 nodes, 254 edges) (Knuth, 1993), BooksUSPolitics (G7 – 105 nodes, 401 edges) (Newman, 2006), Flare Software Class (G8 – 220 nodes, 709 edges) (Holten, 2006) and the USAirline (G9 – 235 nodes, 1297 edges) from an unknown source.

We executed two sets of experiments, one for each of the ABEB and CBEB problems discussed in Section 5. The initial layout of the graphs with fixed nodes was predefined in an input file. The experiments consisted of running the evolutionary algo-

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	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
G9	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

rithm for 100 independent trials for each graph and angle parameter. For the ABEB, the angle parameter was progressively fixed as $\alpha = 30^\circ, 45^\circ, 70^\circ$. For the CBEB, the angle parameters were considered, but they were employed for calculating T_a thresholds using the formula $T_a = 1 - \alpha/180$. Furthermore, for the CBEB, the weights of the components of the objective function were chosen empirically as $w_1 = 0.2$ and $w_2 = 0.8$ (see Section 5). Other parameters defined for both 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 were executed on a MacBook Pro with an Intel Core i7 processor of 2.9 GHz and 8GB of 1600MHz-DDR3 RAM.

7.1 Results for the ABEB Problem

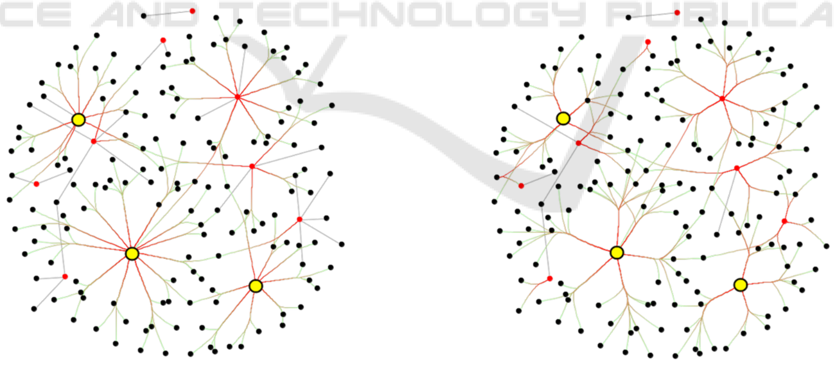
The first experiment aimed to produce solutions with the minimum number of bundles, and all edge angles in a bundle never higher than a given α. Table 1 summarizes the results for the various graphs. The first three columns consist of general information. The fourth column shows the number of bundles of the best solution in 100 trials. The amount of trials in which a solution with that quality appeared is presented in parenthesis. 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 eight columns are the standard deviation and the standard error of the fitness values. The last column shows the average of the total runtime.

Analyzing the table, one can see that the average of the number of bundles in the best solutions was nearby 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 3 illustrates the best solutions obtained for the graph G5 with angle constraints $\alpha = 30^\circ$ and $\alpha = 70^\circ$. A comparison between Figures 3(a) and (b) shows the effect on the bundling as the α angle increases, joining more edges. This is very noticeable for the edges connected to the highlighted nodes (drawn as circles in light yellow).

In general, when solving the ABEB, the EEB method produced good results in terms of the number

Table 2: 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 $T_s = 0.890$.

Graph	**Edges**	$\alpha(^\circ)$	**Bundles of best solution**	**Avg. compatibility**	**Avg. bundles**	**Avg. fitness**	**Std. dev. fitness**	**Std. error fitness**	**Avg. time (s)**
		30	20 (3)	9.48	20.01	1.9354	0.02551	0.002551	2
G1	28	45	16 (2)	13.27	18.17	2.6990	0.29839	0.029839	2
		70	16 (2)	20.22	16.82	4.0925	0.24830	0.024830	2
		30	56 (2)	22.69	58.49	4.5518	0.24592	0.024592	11
G2	78	45	48 (1)	35.87	50.72	7.1901	0.49003	0.049003	13
		70	36 (1)	69.97	39.10	14.0145	0.86055	0.086055	15
		30	81 (1)	23.23	83.86	4.6546	0.33595	0.033595	12
G3	101	45	76 (1)	34.94	81.10	6.9985	0.80816	0.080816	13
		70	65 (1)	84.31	69.05	16.8739	0.61844	0.061844	18
		30	129 (1)	29.49	132.78	5.9044	0.40687	0.040687	31
G4	160	45	112 (1)	50.48	119.24	10.1020	0.66873	0.066873	39
		70	89 (1)	108.68	94.89	21.7440	1.45149	0.145149	47
		30	76 (1)	136.71	83.90	27.305	2.18483	0.218483	33
G5	161	45	60 (1)	218.55	69.79	43.7209	4.24025	0.424025	36
		70	50 (1)	387.42	52.88	77.5001	5.80255	0.580255	35
		30	174 (1)	121.04	180.17	24.2121	2.08592	0.208592	103
G6	254	45	137 (1)	206.14	147.32	41.2327	3.00882	0.300882	111
		70	97 (1)	426.76	101.97	85.3606	4.74743	0.474743	101
		30	305 (1)	170.56	312.62	34.1143	1.68656	0.168656	300
G7	401	45	260 (1)	266.62	270.62	53.3276	2.48894	0.248894	315
		70	209 (1)	501.12	213.89	100.2274	4.68943	0.468943	363
		30	452 (1)	410.16	464.50	82.0344	4.58910	0.458910	684
G8	709	45	379 (1)	663.31	395.09	132.6634	4.62701	0.462701	834
		70	312 (1)	1108.79	318.49	221.7608	9.51946	0.951946	828
		30	530 (1)	2159.99	535.93	431.9990	14.91452	3.986073	18551
G9**	1297	45	428 (1)	3249.19	437.40	649.8400	22.67482	5.854614	11110
		70	318 (1)	5774.74	327.73	1154.9496	57.56356	14.862847	6859



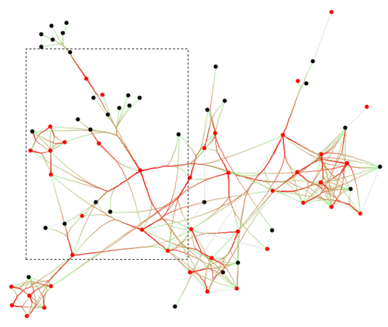
(a) (b)

Figure 3: MovieLens (G5) for ABEB with different angle constraints, (a) $\alpha = 30^\circ$ and (b) $\alpha = 70^\circ$.

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.

7.2 Results for the CBEB Problem

The second experiment was conducted in order to improve the results by introducing the compatibility criteria (CBEB). Table 2 summarizes the experimental results for this problem. It follows the same structure as Table 1 and includes one column representing the average compatibility measure of the best solutions in 100 trials. The stochastic nature of the al-


(a)

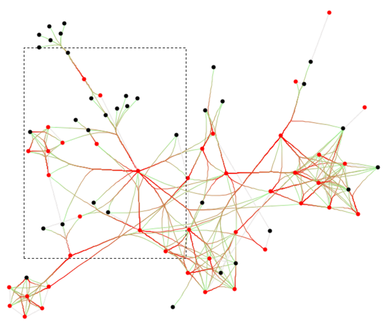

(b)

Figure 4: LesMiserables (G6) with $\alpha = 70°$, (a) ABEB solution with 82 bundles and (b) CBEB solution with 97 bundles.

gorithm associated with the size of the search space produced 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 (G9) graph, first because the experiment was repeated only for 15 independent trials for this graph 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 4(b) shows the best solution produced by EEB for the LesMiserables (G6) graph for the CBEB. Note that, although the number of bundles has increased, the solution for CBEB has bundles having more similar edges than the solution produced for ABEB (Figure 4(a)). The figures show that some bundles present in the ABEB solution split and some other were created in the CBEB solution.

Another additional feature of the CBEB is that, as a multi-objective problem, the solution with the lowest number of bundles is not always produced. This is because the final solution depends on the weights in the objective function and on the compatibility values of the current results.

7.3 Comparison of EEB and an Exact Approach for 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 graphs Synthetic and PlanarGD2015 for the ABEB problem. 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 (Ferreira et al., 2017). The exact method is implemented using

Table 3: Comparison of number of bundles generated by EEB with an exact method (IP) for ABEB problem.

Graph	**Edges**	$\alpha(°)$	**EEB**	**IP**	**Time EEB (s)**	**Time IP (s)**
G1	28	30	17	17	2	0.16
		45	16	16	1	0.16
		70	13	13	1	0.35
		90	11	11	1	0.39
		110	10	10	1	0.14
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 Gurobi solver (Optimization, 2017) and a more powerful computer (a DELL M630 server with 128 GB of RAM and 2 processors with 10 cores each – 40 visible cores of 3-3.6Ghz). The EEB method ran on the simpler computer previously described. More alpha angles were used this time. The results are reported in Table 3. The fourth and fifth columns present the number of bundles of the best solutions by EEB and 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 runtime.

The values in 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 with a maximum angle of 110°, implying that 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.

8 VISUALIZATION

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

In order to allow an adequate visual interpretation of the EEB results, mainly the comprehension of the relation patterns between the nodes, two visualizations are proposed in the present work. The first one shows bundled edges as a colored cubic Bézier curve, going from red to green by default. The red color represents the source (the central node of the bundles), and the destination is colored green. In bundles with just one edge, that edge is drawn as a straight line and is colored light gray. The nodes themselves are colored as black and red. Red means nodes corresponding to a vertex cover set of the graph.

The rendering/routing method of the visualization is a specialized implementation of the force-based edge bundling approach of Holten and van Wijk (2009) that receives the bundles as input and draws them individually. The force-based algorithm guarantees 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 compute the route of the bundles to maximize symmetry (see Figures 3 and 4). This visualization is identified as the "primary view". It allows the reader to identify connections between nodes and to trace individual edges.

Even though the previous visualization can reveal relation patterns at the node level, for graphs with a high level of clutter, edge crossings can still impede their legibility. To reduce the visual edge crossings and highlight the relation patterns, we followed the strategy of Peng et al. (2012); Bruckdorfer et al. (2012) 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 5(d) illustrates this visualization. In addition, the parts of an edge incident to its endpoints can still be colored using different colors. All of these aspects, like colors, transparency level and highlight options can be chosen by the user. Such a smooth visualization allows tracking of the individual edges and is similar to the layout used by the Side-Knot approach (Peng et al., 2012) (see Figure 5(c)).

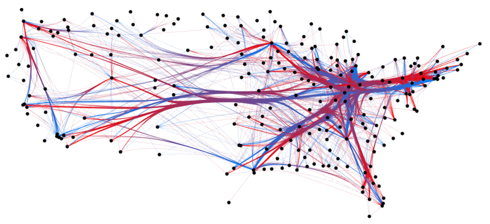
9 DISCUSSION

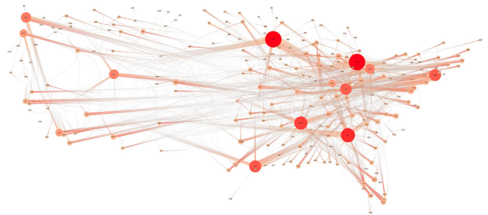
As was mentioned earlier, there is a difference between most classical edge bundling methods and the approach proposed here. The earlier methods attempt to create drawings with reduced clutter. Some of these methods have focused on minimizing various aspects of the drawing that can be mathematically formalized, for example, reducing ink or edge crossing. However, they do not search for an optimal solution, but instead, merely employ heuristic methods to produce solutions (Pupyrev et al., 2011). 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.

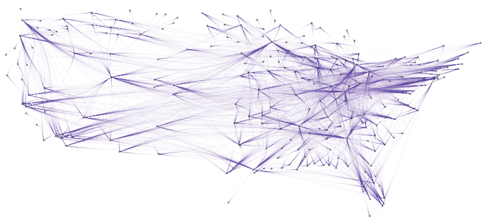
As can be observed in Figure 5 (a), methods based on force (such as divided edge bundling) produce significantly different layouts from the ones created by EEB, since they have no focus on showing connection trends of a node. Therefore, it is hard to establish which nodes are connected to each other. Nevertheless, *stub bundling* (Nocaj and Brandes, 2013) and *sideknot* (Peng et al., 2012) approaches (Figures 5 (b) and (c)) are more similar to EEB. They join only adjacent edges with the aim of indicating directions at the endpoints of the edges, highlight node-level connections and trace individual edges. Those methods, however, are not designed for an optimization-based problem as there is no formal mathematical definition of the problem being investigated.

In addition, 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 tries to find a partition containing only adjacent edges respecting 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" (Nocaj and Brandes, 2013). 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 Figures 5 (b)-(d) reveals the efficiency of EEB for the CBEB problem, with threshold $\alpha = 70°$. The algorithm generated 241 bundles with more than one edge


(a) *Divided Edge Bundling*


(b) *Stub Bundling* (Nocaj and Brandes, 2013)


(c) *SideKnot* (Nocaj and Brandes, 2013)

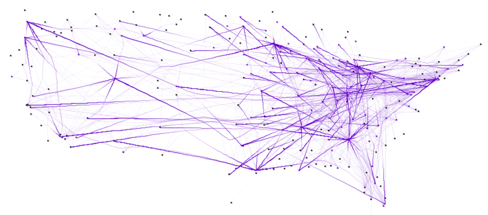

(d) EEB with partial edges, $\alpha = 70°$.

Figure 5: Comparison of solutions for the USAirlines (G9).

and 77 single-bundled edges (318 bundles in total). It is clear that the our group of bundles is markedly different from the solutions generated by the Stub Bundling and the SideKnot methods, since in our approach, they are formed respecting 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.

Finally, the evolutionary edge bundling algorithm attempted to satisfy, implicitly or explicitly, the aesthetics proposed in Section 4. For instance, the minimization criteria of the total number of bundles, the maximization of compatibility between edges, and the maximization of the number of edges per bundle are addressed by the objective function of CBEB (and partially by ABEB) in EEB. Furthermore, bundling only adjacent edges tends to minimize the ambiguity of tracing edges. Symmetry was given by the use of a force-directed edge bundling algorithm for rendering the results of EEB. The minimization of the total number of crossings between edges and bundles was addressed by using partial edge drawings.

10 CONCLUSIONS

This paper describes a new approach for edge bundling by an approximate evolutionary algorithm (EBB) in order to optimize edges to be grouped into edge bundles. We examined angle-based edge bundling which reduces angles between adjacent edges, and compatibility-based edge bundling that group edges with compatible directions and lengths. A previously defined combinatorial optimization problem (ABEB) and a new problem (CBEB) are discussed and solved using the evolutionary method. As far as we know, this is the first evolutionary algorithm for edge bundling modeled as a combinatorial optimization problem. The method was implemented and tested on a number of graphs, showing to be efficient at finding a near-optimal solution when the goal is to create bundled graphs with the minimum number of bundles by joining only adjacent edges. This work examined just two particular edge bundling optimization problems. Many other problems remain open. Future research could focus on the expansion of the approach to extend the fitness function for multi-objective search, in order to deal with new aesthetics criteria. Investigating a complete rendering function that allows a more intrinsic way to route edges and bundles, in order to produce smoother and more readable drawings, may also be fruitful. Finally, conducting some user-controlled studies may possibly establish if the EEB and new optimization-based edge-bundling problems are visually appealing or effective to help users to understand overall patterns.

ACKNOWLEDGEMENTS

This work has been supported by FAPEG-Brazil.

REFERENCES

Alam, J., M.Fink, and Pupyrev, S. (2016). The bundled crossing number. In *24th Internat. Symp. Graph Drawing*, pages 399–412.

Angelini, P., Bekos, M., Kaufmann, M., Kindermann, P., and Schneck, T. (2016). 1-fan-bundle-planar drawings of graphs. In *24th Internat. Symp. Graph Drawing – Poster*, pages 634–636.

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford Univ. Press.

Battista, G., Eades, P., Tamassia, R., and Tollis, I. G. (1998). *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall PTRr, 1st edition.

Bruckdorfer, T., Cornelsen, S., Gutwenger, C., Kaufmann, M., Montecchiani, F., Nöllenburg, M., and Wolff, A. (2012). Progress on partial edge drawings. *CoRR*, abs/1209.0830.

Cui, W., Zhou, H., Qu, H., Wong, P., and Li, X. (2008). Geometry-based edge clustering for graph visualization. *Trans. Vis. Comput. Graph*, 14(6):1277–1284.

Ersoy, O., Hurter, C., Paulovich, F., Cantareiro, G., and Telea, A. (2011). Skeleton-based edge bundling for graph visualization. *Trans. Vis. Comput. Graph*, 12(17):2364–2373.

Ferreira, J. M., Nascimento, H. A. D., Quigley, A. J., and Foulds, L. R. (2017). Computational complexity of edge bundling problemss. `http://inf.ufg.br/biblioteca-digital`. Technical report, Federal University of Goiás.

Girvan, M. and Newman, M. (2002). Community structure in social and biological networks. *Natl. Acad. Sci. USA*, 99(12):7821–7826.

Holten, D. (2006). Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *Trans. Vis. Comput. Graph*, 12(5):741–748.

Holten, D. and van Wijk, J. (2009). Force-directed edge bundling for graph visualization. *Comput. Graph. Forum*, 28(3):983–990.

Hurter, C., Ersoy, O., and Telea, A. (2012). Graph bundling by kernel density estimation. *Comput. Graph. Forum*, 31(3):865–874.

ISGCI (2015). Information system on graph classes and their inclusions. `http://www.csun.edu/gd2015/topics.htm`. [Online; accessed 02-November-2017].

Knuth, D. (1993). *The Stanford GraphBase: A Platform for Combinatorial Computing*. Addison-Wesley.

Lhuillier, A., Hurter, C., and Telea, A. (2017). State of the art in edge and trail bundling techniques. *Comput. Graph. Forum*, 36(3):619–645.

McKnight, R. L. (2015). Low-stretch trees for network visualization. Master dissertation, University of British Columbia.

MovieLens (2017). Movielens. `http://www.eecs.wsu.edu/~yyao/`. [Online; accessed 02-November-2017].

Newman, M. E. J. (2006). Modularity and community structure in networks. *National Academy of Sciences*, 103(23):8577–8582.

Nguyen, Q. H., Hong, S., and Eades, P. (2011). Tgi-eb: A new framework for edge bundling integrating topology, geometry and importance. *19th Internat. Symp. Graph Drawing*, 7034:123–135.

Nocaj, A. and Brandes, U. (2013). Stub bundling and confluent spirals for geographic networks. In *21st Internat. Symp. Graph Drawing*, pages 388–399.

Optimization, G. (2017). Gurobi optimizer quick start guide. `http://www.gurobi.com/documentation/`. [Online; accessed 14-May-2017].

Peng, D., N. Lu, W. C., and Peng, Q. (2012). Sideknot: Revealing relation patterns for graph visualization. In *IEEE Pacific Visualization Symposium*, pages 65–72.

Pupyrev, S., Nachmanson, L., and Kaufmann, M. (2011). Improving layered graph layouts with edge bundling. In *Internat. Symp. Graph Drawing*, pages 329–340.

Saga, R. (2016). Quantitative evaluation for edge bundling by difference of edge lengths and area occupation. In *HCI International Posters Extended Abstracts*, pages 287–290.

Saroj, D. (2012). A non-revisiting genetic algorithm for optimizing numeric multi-dimensional functions. *Intern. Journal on Comput. Sciences and Applic.*, 2(1):83–93.

Selassie, D., Heller, B., and Heer, J. (2011). Divided edge bundling for directional network data. *Trans. Vis. Comput. Graph*, 17(12):2354–2363.

Skiena, S. (1990). *Minimum Vertex Cover. § 5.6.2 in Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*. Addison-Wesley.

Zachary, W. (1977). An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473.