Duologues in Social Networks: Quick and Low-Variance Paths

Stijn Hennissen
Department of Advanced Computing Sciences
Faculty of Science and Engineering
Maastricht University
Maastricht, The Netherlands
June 2023

***Abstract*—A social network is a set of connections between individuals. We modeled such a network as a graph and investigated how information spreads through isolated paths in this graph. We applied a labeling algorithm by Tung and Chew [20] to find multicriteria optimal paths that minimize the expected arrival time and variance in the arrival time of information at vertices in the graph. We also presented three speed-up heuristics to reduce the practical computational complexity of the labeling algorithm. Lastly, we looked at some real-world implications of our findings. We explained that two ways of reducing the expected arrival time and variance in the arrival time of information at vertices, are to add edges if the graph is sparse, or to increase information flow over specific edges if the graph is dense.**

***Index Terms*—information spreading, social networks, paths, arrival time, variance, Pareto-optimal paths, speed-up techniques, distribution fitting**

I. Introduction

A social network is a representation of connections between individuals in a community or other social group. This could be as small as people in a friend group, as big as all employees in a company, or even all citizens of a country. Studying social networks allows us to gain insight into social dynamics and structures between individuals.

In this thesis, we consider a social network where individuals are interconnected based on meetings they may have. A meeting is any situation where a single individual provides information to a single other individual. In the abstraction used in this thesis, we do not consider meetings between more than two people. The social network is represented by an edge-weighted directed graph $G = (V, E)$, where λ_{uv} is the weight of edge $(u, v) \in E$. The edge weights must be strictly positive. Each vertex in the network represents an individual and each edge represents a connection over which meetings can occur. Meetings along an edge are modeled as random events appearing according to an exponential distribution. Meetings are independent between edges and appear continuously for each edge. The weight of an edge is the rate parameter of the exponential distribution associated with that edge. The k'th meeting generated along (u, v) is denoted as $r_{uv}^{(k)}$ which appears $\mathfrak{X}_{uv}^{(k)} \sim \text{Exp}(\lambda_{uv})$ hours after the previous meeting $r_{uv}^{(k-1)}$, or at $\mathfrak{X}_{uv}^{(k)}$ hours if $k = 1$. Meeting $r_{uv}^{(k)}$ happens after $t_{uv}^{(k)} = \sum_{i=1}^{k} \mathfrak{X}_{uv}^{(i)}$ hours. Information is transmitted from u to v at the first meeting $r_{uv}^{(k)}$ after the moment at which u acquires the information.

Given the set $\mathfrak{P}_{uv}$, containing all paths a single piece of information can take between two vertices u and v in the network, we consider a path $P_{uv} \in \mathfrak{P}_{uv}$. We denote the arrival time of the information at vertex v starting from vertex u, traveling along P_{uv} as $\mathfrak{T}_{P_{uv}}$. We investigate what distribution $\mathfrak{T}_{P_{uv}}$ follows and use this to look at ways to choose an optimal path between this vertex pair.

We show that computing an optimal path for information to travel between a vertex pair in the network can be viewed as a multicriteria optimization problem (MCOP) with m criteria. We will define the MCOP as finding all Pareto-optimal (PO) paths between two vertices in a graph. The set of all PO solutions/paths for a given problem is defined as the Pareto-frontier of the problem.

We seek to answer the following research questions: Can we model the transfer of information along an isolated path in a way that allows us to compute, within polynomial time, the expected arrival time and variance in arrival time of information at any vertex along that path? Can we express the optimization of this expected arrival time and variance in arrival time as a discrete bi-criteria shortest path problem? Can we develop speed-up heuristics to an existing algorithm for solving multi-criteria shortest path problems in order to reduce the effective computational complexity? What measures could one take to reduce the expected arrival time and variance in the arrival time of information for vertices in a graph? Is calculating the Pareto-frontier useful for decision-making problems within the field of communication?

We start off by explaining the definitions and terminology used in this thesis. We then move on to the analysis of related work that we use as starting point for the algorithm described in this thesis. Afterward, we move on to the methods used. Here we describe how we model the problem and the algorithm we propose for finding a solution to this problem and talk about speed-up heuristics for this algorithm. We then move on to the experiments, followed by the results and discussion, where we answer our research questions and address the ethical implications of our work. We close off with our conclusion, including possible future work and our contribution to the field.

II. Definitions and Terminology

This section goes over some specific definitions used in this thesis. A complete summary of all symbols can be found in the appendix in section X-A.

A. Incident edges to a vertex

We denote all edges (u, v) coming out of u as $(u, v) \in E^{\text{OUT}}(u)$ and all edges (u, v) coming into v as $(u, v) \in E^{\text{IN}}(v)$.

B. Costs and Pareto-optimality

We define $c(x, y)$ to be the vector containing the cost of going from vertex x to vertex y for each of the m criteria. From this definition, we define $c(P_{uv})$ for $P_{uv} \in \mathfrak{P}_{uv}$, the cost of path P_{uv} from u to v, as

$$c_j(P_{uv}) = \sum_{(h,i)\in P_{uv}} c_j(h, i) \tag{1}$$

where $c_j(P_{uv})$ is the j'th component of the objective vector $c(P_{uv})$, corresponding to the j'th criterion. For two different paths $P_{uv} \in \mathfrak{P}_{uv}$ and $Q_{uv} \in \mathfrak{P}_{uv}$ we define

$$c_j(P_{uv}) \leq c_j(Q_{uv}), j = 1, 2, ..., m \tag{2}$$

as used in the following definition.

Definition II.1 (Dominating a path)**.** A path $P_{uv} \in \mathfrak{P}_{uv}$ from vertex u to vertex v *dominates* another path $Q_{uv} \in \mathfrak{P}_{uv}$ from u to v if inequality (2) holds for at least one j.

Using this definition we state the following:

Definition II.2 (Pareto-optimal path)**.** A path $P_{uv} \in \mathfrak{P}_{uv}$ from vertex u to vertex v is defined to be a *Pareto-optimal path* if there is no other path $Q_{uv} \in \mathfrak{P}_{uv}$ from u to v that *dominates* P_{uv} by definition II.1.

III. Related Work

The single-criterion shortest path problem has famously been solved using the algorithm presented by Dijkstra [4]. This algorithm has later been adapted by Hansen [7] for several bi-criterion cases. A few years later Martins [11] presented a generalization for the multicriteria case, based on Hansen's work. Martin's multicriteria algorithm [11] is based on lexicographically ordered search. While Martins in his paper [11] provides rigorous proof that his algorithm works mathematically, he states that his algorithm has not been implemented in practice and that performance would be highly dependent on the graph and data structure used. A new adaptation of Martins' algorithm [11] is later presented by Tung and Chew [20], that generalizes a bicriterion labeling scheme [19] coupled with a heuristic evaluation concept [12], [15] into the multicriteria case, using techniques developed for Martins' algorithm [11].

In this thesis, we seek to expand on the labeling algorithm proposed by Tung and Chew [20], by researching an open statement they leave in their paper: the combining function can be any strictly positive function. We will propose normalizing edge weights and using a weighted sum as the combining function.

IV. Methods

A. Fitting a distribution

Since the spread of information through meetings is stochastic by nature, modeled by random events, the arrival of information at specific individuals is as well. To get a concrete view of the arrival times of information, we find out if the arrival times at vertices along isolated paths in the graph follow a specific distribution and if we can calculate the parameters of this distribution.

In the model, we describe transferring information between vertices in an isolated path in the graph. We specify that meetings happen continuously for each edge and information is transferred from vertex u to vertex v by the first meeting $r_{uv}^{(k)}$ that happens after information arrives at vertex u, where the time at which $r_{uv}^{(k)}$ happens is independent of the time when information arrives at u. We compare this to the similar but computationally advantageous model that models the transfer of information as a Markov chain [10], meaning that once information arrives at vertex u, we generate a meeting r_{uv} that transfers information to vertex v. The difference between the Markov chain model and our original model is that meetings $r_{uv}^{(k)}$ are not generated continuously from the start, but one is only generated when information arrives at vertex u.

1) The Hypoexponential distribution: The advantage of the Markov chain model is that the distribution we try to find for the arrival times of information at specific vertices of an isolated path has already been studied and defined. This distribution is the Hypoexponential distribution [2] defined as a series of n exponential distributions, each with their own rate λ_i, the rate of the i'th exponential distribution, where all n exponentially distributed random variables X_i are independent. For these conditions, the random variable

$$X = \sum_{i=1}^{n} X_i \tag{3}$$

is hypoexponentially distributed.

The hypoexponential distribution is defined by the cumulative density function

$$\text{CDF: } F(x) = 1 - \boldsymbol{\gamma} e^{x\Theta}\mathbf{1} \tag{4}$$

and the probability density function

$$\text{PDF: } f(x) = -\boldsymbol{\gamma} e^{x\Theta}\Theta\mathbf{1}. \tag{5}$$

Here we define the matrix Θ as

$$\Theta = \begin{bmatrix} -\lambda_1 & \lambda_1 & \cdots & 0 & 0 \\ 0 & -\lambda_2 & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \ddots & -\lambda_{n-1} & \lambda_{n-1} \\ 0 & 0 & \cdots & 0 & -\lambda_n \end{bmatrix}, \tag{6}$$

and the vector γ as

$$\gamma = [1, 0, ..., 0]. \tag{7}$$

Additionally, $\mathbf{1}$ is a column vector of ones of the size n, and e^A is the matrix exponential of A. Lastly, the mean μ for the hypoexponential distribution is defined as

$$\mu = \sum_{i=1}^{n} \frac{1}{\lambda_i} \tag{8}$$

and the variance σ^2 is defined as

$$\sigma^2 = \sum_{i=1}^{n} \frac{1}{\lambda_i^2}. \tag{9}$$

We hypothesize that the hypoexponential distribution fits to describe the arrival times $\mathfrak{T}_{P_{uv}}$ of information at vertex v, starting from u, along the path $P_{uv} \in \mathfrak{P}_{uv}$. We could not formulate a proof for this assumption, although we ran simulations that seem to confirm our hypothesis. These simulations as well as the results are explained in detail in section V-B and VI-A. We use these results as backing to propose the following conjecture.

Conjecture IV.1. Given the path $P_{uv} \in \mathfrak{P}_{uv}$ from vertex u to vertex v and any edge $(i,j) \in P_{uv}$. Information moves from vertex i to vertex j when a meeting r_{ij} happens and information has already arrived at i. Meetings happen according to random exponential events, independent between edges. Model A says meetings appear continuously at each edge. Model B says a meeting is generated for edge (i,j) as soon as information arrives at vertex i. Define the arrival time $\mathfrak{T}_{P_{uv}}$ of information at vertex v, starting from vertex u, traveling along path P_{uv}. The distribution of random variables $\mathfrak{T}_{P_{uv}}$ have the same, hypoexponential, mean and variance for model A as for model B.

This conjecture allows us to rewrite the definition of a hypoexponential random variable in terms that fit our situation. For all edges (i,j) in $P_{uv} \in \mathfrak{P}_{uv}$, where edge (i,j) has weight λ_{ij}, generate an independent random variable $\mathfrak{X}_{ij} \sim \exp(\lambda_{ij})$ for each edge (i,j). We state that

$$\mathfrak{T}_{P_{u,u+1}} = \mathfrak{X}_{u,u+1} \tag{10}$$

and accordingly, state that

$$\mathfrak{T}_{P_{uj}} = \mathfrak{T}_{P_{ui}} + \mathfrak{X}_{ij}, \tag{11}$$

which means that the random variable

$$\mathfrak{T}_{P_{uv}} = \sum_{(i,j)\in P_{uv}} \mathfrak{X}_{ij} \tag{12}$$

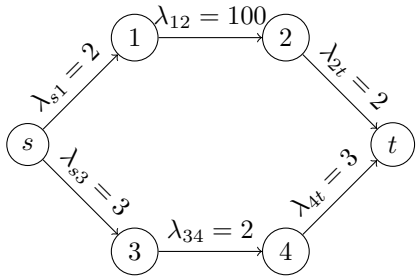
is hypoexponentially distributed, and represents the arrival time of the information at vertex v starting from vertex u, traveling along path P_{uv}. Now the hypoexponential distribution for our path P_{uv} has mean μ defined as

$$\mu = \sum_{(i,j)\in P_{uv}} \frac{1}{\lambda_{ij}} \tag{13}$$

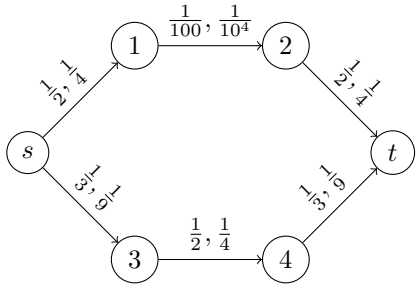
and variance σ^2 defined as

$$\sigma^2 = \sum_{(i,j)\in P_{uv}} \frac{1}{\lambda_{ij}^2}. \tag{14}$$

This definition of μ and σ^2 lends itself to considering the movement of information along paths in the graph on an edge-by-edge basis. Each edge (i,j) directly contributes to the mean and variance based on its rate parameter λ_{ij}. In the practical application that we study, it makes sense to try and minimize both the mean μ and the variance σ^2 when choosing which path information should take through the graph. By spreading information along edge (i,j), μ increases by $1/\lambda_{ij}$ and σ^2 increases by $1/\lambda_{ij}^2$. This means that to minimize both μ and σ^2, we need to minimize both $1/\lambda_{ij}$ and $1/\lambda_{ij}^2$, when choosing along which edge to pass information. Even though both terms are based on λ_{ij}, we cannot simplify this to maximizing just λ_{ij}. To reason why this simplification will not work, consider the graph



which when considering $1/\lambda_{ij}$ and $1/\lambda_{ij}^2$ instead of λ_{ij} turns into



When choosing between the two paths from s to t, $s-1-2-t$ minimizes $1/\lambda_{ij}$, while $s-3-4-t$ minimizes $1/\lambda_{ij}^2$. This highlights the problem that is encountered when trying to optimize both μ and σ^2. Situations will arise where optimizing one will diminish the other. In these situations, where we can not choose a single optimal path, we choose to find all Pareto-optimal paths, following the definition II.2.

B. Finding multicriteria Pareto-optimal paths

To find Pareto-optimal paths we use the algorithm introduced by Tung and Chew [20] as a basis. This algorithm finds all Pareto-optimal paths from a starting vertex s to a target vertex t, for m criteria. Even though the situation discussed in this thesis is the bicriterion case, the algorithm treats the multicriteria case and will be described as such.

The algorithm works by building step by step, starting from s, loopless paths through alternating evaluation and branching. As we progress, each vertex y will be given in succession a

sequence of labels. Label $\theta(y)$ is assigned every time vertex y is reached, from starting vertex s, and consists of three parts:

$$\theta(y) = [y; \theta(x); \alpha(y)] \tag{15}$$

The label indicates that vertex y was reached from vertex x with label $\theta(x)$. The vector $\alpha(y)$ indicates the cost of getting to y for each criterion and is defined by the relation

$$\alpha(y) = \alpha(x) + c(x, y), \tag{16}$$

where $\alpha(x)$ is obtained from the label $\theta(x)$. A label can be either tentative or permanent. A tentative label is available to choose as the next label to branch from, at which point it becomes permanent and can not be chosen again.

Since we need to consider all criteria when calculating a heuristic value for a label, we use a *combining function*, $\mathfrak{F}(\mathfrak{l})$ that, given a vector $l = (l_1, l_2, ..., l_m)$, combines the components of l into a single value. Tung and Chew [20] decided to use the summation of the components of l as their combining function, defining it as

$$\mathfrak{F}(l) = \sum_{l_i \in l} l_i. \tag{17}$$

Transitively from equation (1) and (17) we now define the combining function for a path $P_{uv} \in \mathfrak{P}_{uv}$ as

$$\mathfrak{F}(P_{uv}) = \sum_{j=1}^{m} c_j(P_{uv}). \tag{18}$$

Using equation (18) we define

$$h^*(u, v) = \min\ \mathfrak{F}(P_{uv}); P_{uv} \in \mathfrak{P}_{uv} \tag{19}$$

to be the cost value of the minimum cost path $P_{uv} \in \mathfrak{P}_{uv}$ from u to v using $\mathfrak{F}(c(i, j))$ as the cost of traveling along each edge $(i, j) \in P_{uv}$. In a similar vein, we define

$$q_j(u, v) = \min\ c_j(P_{uv}); j = 1, 2, ..., m; P_{uv} \in \mathfrak{P}_{uv} \tag{20}$$

to be the cost value of the minimum cost path $P_{j,uv} \in \mathfrak{P}_{j,uv}$ from u to v out of all criteria, given that each criterion's path is the minimum cost path for that criterion.

For a label $\theta(y)$ at vertex y, as given in equation (15), we calculate the evaluation value $e(\theta(y))$ as

$$e(\theta(y)) = \mathfrak{F}(\alpha(y)) + h^*(y, t) \tag{21}$$

Thus, $e(\theta(y))$ is the combined cost of the path from s to y plus the minimum combined future cost of reaching t from y. The evaluation value is used as the heuristic value of each label.

To initialize the algorithm to find all Pareto-optimal paths from vertex s to vertex t, we set the first tentative label $\theta(s) = [s; -; \alpha(s)$, where $\alpha(s) = \mathbf{0}$. We define the set of all tentative labels L to be initialized as $L = \{\theta\}$. Lastly, we define and initialize the set of all efficient objective vectors, all solutions, as $\Gamma = \emptyset$. After the algorithm is done Γ will contain the Pareto-frontier from s to t.

We consider the set L of all current tentative labels at an iterative step. If L is empty, the algorithm is done executing. Otherwise, choose the tentative label $\theta(y)$ with minimum evaluation value and make it permanent.

Case 1. If $y = t$, determine if

$$\alpha_j(t) < \xi_j; \xi \in \Gamma; j = 1, 2, ..., m \tag{22}$$

holds for all ξ and all j. If this is the case, $\theta(t)$ is not dominated by any existing solution and we add $\theta(t)$ to Γ. If this is not the case, reject $\theta(t)$ as a solution. To obtain the Pareto-optimal path, trace from $\theta(t)$ back to $\theta(s)$. We also check if

$$\xi_j < \alpha_j(t); \xi \in \Gamma; j = 1, 2, ..., m \tag{23}$$

holds for all ξ and all j, removing any ξ, for which this is not the case, from Γ.

Case 2. If $y \neq t$, consider the vertices $z \in E^{\text{OUT}}$. Reject vertex z if

$$\begin{aligned} \alpha_j(y) + c_j(y, z) + q_j(z, t) \geq \xi_j; \\ \xi \in \Gamma; j = 1, 2, ..., m \end{aligned} \tag{24}$$

holds for all j and all ξ, meaning any path that could be constructed by branching from z will be dominated by an existing solution. The remaining neighbors are assigned a tentative label according to

$$\theta(z) = [z; \theta(y); \alpha(y) + c(y, z)] \tag{25}$$

and are added to L.

1) The labeling algorithm: Algorithm (1) shows the pseudo-code for the labeling algorithm, as well as the functions used in it.

In practice, executing the labeling algorithm requires computing $q(y, t)$ and $h^*(y, t)$ for the target vertex t and every vertex $y \in V$. We can pre-compute these values before running the algorithm by running Dijkstra's algorithm [4] on the graph with the direction of all edges reversed, denoted by G'. Proof as to why this works can be found in Tung and Chew's work [20]. In case we want to find the Pareto-frontier for every vertex pair in the graph, we compute $q(u, v)$ and $h^*(u, v)$ for all $u, v \in V$, where $u \neq v$.

Algorithm 1 The labeling algorithm.

function $c(x, y)$
  return vector with cost of moving from x to y, per criterion

function $c(P_{uv})$
  return $\sum_{(i,j)\in P_{uv}} c(i, j)$

function $\mathfrak{F}(l = (l_1, l_2, ..., l_m))$
  return $\sum_{l_i \in l} l_i$

function $q_j(u, v)$
  return $\min\ c_j(P_{uv}); j = 1, 2, ..., m$

function $h^*(u, v)$
  return $\min\ \mathfrak{F}(P_{uv})$

function $e(\theta(y))$
  return $\mathfrak{F}(\alpha(y)) + h^*(y, t)$

procedure PO-PATHS(V, E, s, t)
  $\alpha(s) \leftarrow \mathbf{0}$
  $\theta(s) \leftarrow [s; -; \alpha(s)]$
  $L \leftarrow \{\theta(s)\}$
  $\Gamma \leftarrow \emptyset$
  while $L \neq \emptyset$ **do**
    $\theta(y) \leftarrow \underset{\theta(x)\in L}{\arg\min}\ e(\theta(x))$
    $L \leftarrow L \setminus \alpha(y)$
    if $y = t$ **and** $\neg(\xi \in \Gamma$ **dominates** $\alpha(y))$ **then**
      $\Gamma \leftarrow \Gamma \setminus \xi$ **for** $\alpha(y)$ **dominates** $\xi \in \Gamma$
      $\Gamma \leftarrow \Gamma \cup \theta(y)$
      continue
    for $\forall(y, z) \in E^{\text{OUT}}(y)$ **do**
      if $\neg(\xi \in \Gamma$ **dominates** $\alpha(y) + c(y, z) + q(z, t))$ **then**
        $\theta(z) \leftarrow [z; \theta(y); \alpha(y) + c(y, z)]$
        $L \leftarrow L \cup \theta(z)$

2) The combining function: The evaluation function of the algorithm is comprised of two pieces, $\mathfrak{F}(\alpha(z))$ and $h^*(z, t)$. Both pieces use the combining function $\mathfrak{F}(l)$, which is used to determine a heuristic value that incorporates all components of the vector l, where l represents the vector of all cost functions for the optimization criteria. Tung and Chew [20] define $\mathfrak{F}(l)$ as a summation of the components of l, but state that any strictly increasing combining function can be used.

We elaborate on this statement by noting that we only need to use a strictly increasing combining function if we allow non-positive edge weights. If we enforce strictly positive edge weights, we may choose any combining function. The reason behind requiring either of these two conditions is that we have to avoid non-positive evaluation value cycles in the graph since they could make the labeling algorithm stall.

3) Normalization: The first adjustment we make to the combining function is to normalize edge weights to a range from 0 to 1. This is done by taking all the edge weights for each criterion and dividing them by the largest edge weight for each criterion. We do this to eliminate the effect that criteria with bigger cost values could have on the evaluation function, effectively drowning out the influence of criteria with smaller cost values. This evens the playing field, allowing all criteria to contribute equally to deciding which label the algorithm will consider next. For example, consider two criteria: the first has edge weights between 1000 and 10000 and the second one has edge weights between 0 and 1. Without normalization, the combined evaluation value can be between 1000 and 10001, meaning that the effect of the second criterion is negligible compared to the first criterion. With normalization, however, the combined evaluation value can be between 0 and 2, where both criteria contribute equally.

4) Weighing the combining function: With all the cost vectors normalized we propose to weigh each criterion's cost vector in the combining function. This weighing effectively means we decide to prefer minimizing some criteria over others. The goal here is to find as many solutions as quickly as possible since the more solutions are found, the more the algorithm can prune using the rejection rule given by equation (24), meaning the algorithm will require fewer labels and less time to finish.

How we weigh the different criteria is based on what weighing scheme we choose. Many weighing schemes could be devised, but we will focus on two in particular. Which weighing scheme to use in practice for the best results will depend heavily on the problem that is being modeled and will most likely require some analysis of the underlying graph structure. Since this analysis is not the focus of this thesis we will weigh based on two example schemes that we expect would work in many general situations. The first scheme weighs based on a centrality measure, computed as the standard deviation in costs per criterion within the Pareto-frontier found. The second scheme weighs based on the label cost of the shortest path per criterion. Both schemes are explained in further detail in section V-C2 and V-C3.

The combining function $\mathfrak{F}$ will now be defined as

$$\mathfrak{F}(l, w) = \frac{w}{|w|} \cdot l \tag{26}$$

where l is any vector with m components, each representing a cost per criterion, and w is a vector containing the weights per component, normalized to between 0 and 1.

5) Iterative refinement: To compute the weights we use iterative refinement, getting a better and better estimation with each vertex for which we find the Pareto-frontier. To apply iterative refinement we initialize the weights $w = (1, 1, ..., 1)$. We iterate over all vertex pairs in the graph and compute the Pareto-frontier for each of them. After finding a Pareto-frontier we adjust the weights according to our weighing scheme. The new weights get used in the combining function during the next iteration of the algorithm. This results in the weights getting more adjusted to the structure of the graph with each Pareto-frontier that is computed.

V. Experiments

A. Graphs

All experiments make use of graphs or paths in graphs. In all experiments, these graphs are edge-weighted directed graphs. The graph structure is constructed using the *Social Circles: Google+* dataset, which is part of the *SNAP* database published by Stanford University [9]. The edge weights are obtained from data gathered in a study done by the author of this thesis. The study sought to gather information concerning the number of one-to-one meetings people have in a work environment. Data about meeting frequency was gathered from teaching staff at a Dutch high school. We fit a lognormal distribution to this data and use this distribution to generate edge weights representing meeting rates λ.

B. Simulations

To support the proposal that the hypoexponential distribution fits to describe the arrival times $\mathfrak{T}$ of information at specific vertices, we run a series of simulations. The aim of these simulations is to see how close the mean arrival times μ and variances σ^2, that we are able to calculate for the hypoexponential distribution using equations (13) and (14), are to the sample mean $\bar{X}$ and the sample variance s^2 that we get from the simulations. We run 10000 simulations of information traveling along a path containing 100 vertices

C. Labeling Algorithm Comparisons

In the experiments, we compare three different adjusted versions of the labeling algorithm to the base version proposed by Tung and Chew [20], where for each one we choose a different combining function $\mathfrak{F}$. For each version, we execute the algorithm to find Pareto-frontiers for ten vertex pairs, chosen at random, but consistent between experiments. We count the labels created while computing the ten Pareto-frontiers. For each version, these ten Pareto-frontiers are computed for 40 different graphs with consistently increasing densities.

The density of a graph is defined as

$$DEN_G = \frac{|E|}{|E_{MAX}|}, \tag{27}$$

where DEN_G stands for the density of graph G, $|E|$ is the number of edges in G, and $|E_{MAX}|$ is the maximum number of edges that could be in G. Since we use directed graphs we can rewrite this equation as

$$DEN_G = \frac{|E|}{|V|(|V|-1)}, \tag{28}$$

where $|V|$ is the amount of vertices in G [6].

In the end, we have 40 label counts per adjusted version, as well as for the base version. We will use this to compute the improvement achieved by the adjusted versions over the base version. The adjusted versions are as follows:

1) Normalized Sum (NS): For the first version, the normalized sum, we apply the normalization described in section IV-B3, and use the same combining function as the base version.

2) Weighted NS: Centrality: The second version builds upon the first version. This version applies the weighing procedure as described in section IV-B4, where the weights are calculated according to the following procedure:

After we compute a Pareto-frontier we look at each criterion and calculate the standard deviation for the cost of that criterion across all PO-paths in the Pareto-frontier, which we use as a measure of centrality. This standard deviation per criterion is added to the criterion's component in the weight vector.

3) Weighted NS: Label Cost: The third version also builds upon the first version and applies the weighing procedure as described in section IV-B4, where the weights are calculated according to the following procedure:

After we compute a Pareto-frontier we look at each criterion and find the PO-path that has the minimal label cost for that criterion, where the label cost of a path is defined as the sum of the number of labels created at each vertex on the path. We then add this label cost per criterion to that criterion's component in the weight vector.

D. Complexity measure

In their paper, Tung and Chew [20] state that the size of the set of labels given to a single vertex is bounded by

$$\sum_{k=0}^{|V|-2} \binom{|V|-2}{k} k!, \tag{29}$$

which derives to

$$\sum_{k=0}^{|V|-2} \frac{(|V|-2)!}{(|V|-k)!}. \tag{30}$$

By interpreting equations (29) and (30) we can state that the total amount of labels $|\mathfrak{L}|$ created by the algorithm grows as the number of vertices $|V|$ grows, and that $|\mathfrak{L}|$ grows faster than any other part of the algorithm. Because of this, we use the number of labels created by the algorithm as a measure of computational complexity in the experiments.

VI. Results

A. Simulation Results

After running the experiments we calculate the mean absolute percentage error (MAPE) and coefficient of determination score (R^2-score), for both the mean and variance, to determine the quality of our prediction. These measures are summarised in table I. We use $\bar{X}$ and s^2 as true values, and μ and σ^2 as predicted values.

	MAPE	$1 - R^2$-score
Mean	8×10^{-4}	8×10^{-7}
Variance	5×10^{-3}	1×10^{-4}

Table I
MAPE AND R^2-SCORES OF MEAN AND VARIANCE FROM SIMULATION.

B. Labeling Algorithm Results

For the different versions of the labeling algorithm, we look at how the total number of labels created grows as the density of the graph grows, as well as the percentage difference between the base version and the three adjusted versions.

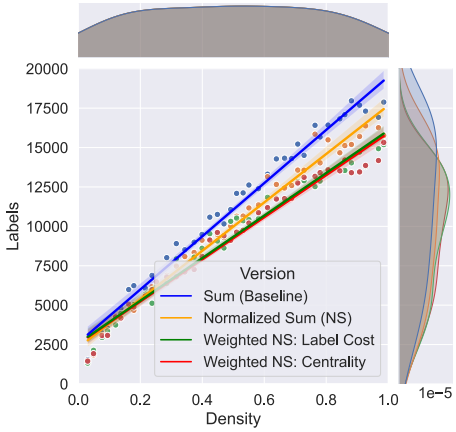


Figure 1. Total label count with density.

In figure (1) we see that the amount of labels grows linearly as the density of the graph grows.

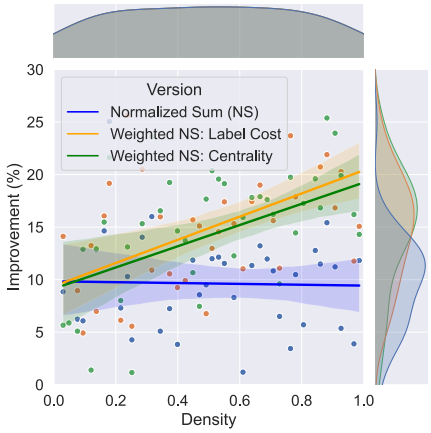


Figure 2. Percentage improvement with density.

In figure (2) we see that the NS version has a consistent improvement of around 10% over the base version. In contrast, we see that both weighted versions linearly increase their improvement over the base version from around 10% to around 20%, as the density of the graph increases, where the Label Cost version increases slightly faster than the Centrality version.

VII. Discussion

A. hypoexponential Conjecture

From the results of the simulations, there seems to be strong evidence that conjecture IV.1 holds for the paths that we simulated. It must be noted, however, that we only look at predicting the sample mean and sample variance. We do not analyze if the hypoexponential distribution as a whole fits the simulation data. More research would be required to make any concrete statement about the overall fit. If this were to be done, the conjecture could be adjusted to fit these new results.

B. The labeling algorithm

In optimizing the labeling algorithm we can already see significant results when normalizing edge weights in the graph. This is to be expected since the graphs we use are prone to exactly the problem normalizing was made to tackle: disproportionate edge weights. Since our second criterion is always the square of the first criterion, it is prone to grow and shrink to great proportions when compared to the first criterion. Normalizing makes it so that both criteria exert the same influence on the evaluation function, resulting in the algorithm making less arbitrarily biased decisions.

On top of the improvement provided by the normalization, we gain another improvement by applying weighing to the combining function. The interesting thing to notice is that the improvement gained by weighing grows linearly with the density of the graph. We expect that this happens because the weighing takes advantage of the inherent structure of the graph. If the graph is more densely connected, meaning more vertices are connected, there would be more structure present that the weighing can take advantage of.

There does not seem to be any significant difference between the two example weighing schemes presented in this thesis, most likely because they are so simple. It would be interesting to see if more case-specific weighing schemes, that have a basis in some underlying analysis of the graph, would be able to achieve even better performance increases.

C. Optimizing information spread

There is a question we want to discuss that could arise from this thesis. For the most part, we only look at identifying and quantifying the optimal paths in a graph for information to take. However, in a real-world application, it would make sense to ask: How do we improve information spread in a given graph? We decide to look at this from two possible angles. The first option is to encourage information to spread to more people, more quickly, by adding new edges. The second option is to encourage an increase in the rate at which meetings happen at an existing edge. The way we see it, both options have merit, depending on the existing structure of the graph [16].

The first option of adding more edges could be very effective in sparse graphs, graphs with a low density, so a small number of edges. Adding strategic edges in a graph like this can connect whole sections of the graph, that previously may have been (nearly) disconnected. On the other hand, if the graph is already very well connected or even fully connected, adding another edge will have little to no effect. In this case, it may be better to consider the second option.

The second option, increasing the rate at which meetings happen, would make sense in situations where the ability to

change the structure of the graph is limited, for example when the graph is already very dense, when hierarchical dynamics need to be preserved, or when sections of the graph are physically separated, such as with a company that has multiple locations. In this case, increasing the rate at which meetings happen over vital edges or edges that are a bottleneck, can be a good solution.

D. Usefulness of Pareto-frontiers

A large part of this thesis is focused on finding Pareto-frontiers, without really asking the question of whether using Pareto-frontiers is at all a good fit for the intended real-world application in social networks. Since their conceptualization, Pareto-frontiers have been applied to gain insight into decision-making problems in a plethora of fields, including medicine [13], communication for infrastructure [16], climate [3], machine learning [18], and physics [17]. For each specific application and field, it is important to make investigate how much of the decision-making process can be influenced by a Pareto-frontier that has been found. This is stressed by Ottosson et al. [13] in relation to the use of Pareto-frontiers for their study on decision-making in medical treatment planning. However, given the already numerous applications of Pareto-frontiers in communication networks [16], [8], [1], [14], [5], we are of the opinion that it is a valuable asset regarding decision-making in this field.

E. Ethics

When studying anything sociological in nature, like social networks, it is important to keep the ethical implications of possible real-world applications in mind. If at all possible, one should anonymize collected data as much as possible, as the way that people communicate with others is very personal and should be kept private, unless explicit permission has been given otherwise. At the other end of the research cycle, when results are presented, care should also be given to how and in what detail results are presented. Consider, for example, a situation where an individual in a social network has been identified as an information bottleneck. Discussing such a matter should be done with great care regarding those who might gain knowledge into the identity of this individual, as to avoid loss of face for this individual among their peers in the social network.

VIII. Conclusions

In this thesis, we found that it can be conjectured that, given our definition of information spreading along isolated paths in a graph, the expected arrival time and variance in the arrival time of information at any vertex on this path, can be predicted by the hypoexponential distribution. We showed how to express minimizing this expected arrival time and variance in the arrival time, by finding a multicriteria shortest path in the graph. We found it was effective to apply normalization and weighing as speed-up heuristics to the labeling algorithm by Tung and Chew [20] and achieved a significant reduction in practical computational complexity. We presented two ways of reducing the expected arrival time and variance in the arrival time of information at vertices. One could add edges if the graph is sparse or increase information flow over specific edges if the graph is dense. Lastly, we showed that in many cases the Pareto-frontier would be useful for decision-making problems, within the field of communication.

In the future, the research presented in this thesis could be expanded upon in multiple ways. One could investigate the effect of other weighing schemes that are more case-specific. Work could be put toward proving or disproving the conjecture presented in this thesis. Another question that could be investigated is how the current findings translate to a scenario where we do not look only at isolated paths, but expand the model to include trees or other more complex graph structures. One could investigate if the methods presented could be applied to distributions, related to the hypoexponential distribution, like the hyperexponential distribution.

The results of this thesis contribute to the field, first of all, by investigating and exploiting the open statement from Tung and Chew's paper [20]. In addition to this, this thesis proposes a conjecture relating two different existing models of information transfer to each other, which if proven could give insight into the underlying distribution, but further research is required to see if this conjecture has merit in the general case.

IX. References

[1] Aamir Akbar, Muhammad Ibrar, Mian Ahmad Jan, Ali Kashif Bashir, and Lei Wang. Sdn-enabled adaptive and reliable communication in iot-fog environment using machine learning and multiobjective optimization. *IEEE Internet of Things Journal*, 8(5):3057–3065, 2021.

[2] R. Bekker and P. Out. Scheduling admissions and reducing variation in bed demand. *Health Care Management Science*, 14(3):237–249, 2011.

[3] Jose G. Borges, Jordi Garcia-Gonzalo, Vladimir Bushenkov, Marc E. McDill, Susete Marques, and Manuela M. Oliveira. Addressing Multicriteria Forest Management With Pareto Frontier Methods: An Application in Portugal. *Forest Science*, 60(1):63–72, 04 2013.

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

[5] Joaquín Goñi, Andrea Avena-Koenigsberger, Nieves Velez de Mendizabal, Martijn P. van den Heuvel, Richard F. Betzel, and Olaf Sporns. Exploring the morphospace of communication efficiency in complex networks. *PLOS ONE*, 8(3):1–10, 03 2013.

[6] Jonathan L. Gross and Jay Yellen. *Graph theory and its applications*. CRC Press, 2005.

[7] Pierre Hansen. Bicriterion path problems. In *Multiple Criteria Decision Making Theory and Application: Proceedings of the Third Conference Hagen/Königswinter, West Germany, August 20–24, 1979*, pages 109–127. Springer, 1980.

[8] Stephen D. Krasner. Global communications and national power: Life on the pareto frontier. *World Politics*, 43(3):336–366, 1991.

[9] Jure Leskovec. Social circles: Google+, 2012.

[10] Andrei Andreevich Markov. Rasprostranenie predel'nyh teorem ischisleniya veroyatnostej na summu velichin svyazannyh v cep', zapiski akademii nauk po fiziko-matematicheskomu otdeleniyu, viii seriya 25 (3)(1908). translated into german, ausdehnung der satze uber die grenzwerte in der wahrscheinlichkeitsrechnung auf eine summe verketteter grossen. *Wahrscheinlichkeitsrechnung (translated by H. Liebmann), BG Teuber, Leipzig*, pages 272–298, 1912.

[11] Ernesto Queirós Vieira Martins. On a multicriteria shortest path problem. *European Journal of Operational Research*, 16(2):236–245, 1984.

[12] Nils J Nilsson. Problem-solving methods in. *Artificial Intelligence*, 5, 1971.

[13] Rickard O. Ottosson, Per E. Engström, David Sjöström, Claus F. Behrens, Anna Karlsson, Tommy Knöös, and Crister Ceberg. The feasibility of using pareto fronts for comparison of treatment planning systems and delivery techniques. *Acta Oncologica*, 48(2):233–237, 2009. PMID: 18752085.

[14] C. Papagianni, K. Papadopoulos, C. Pappas, N. D. Tselikas, D. T. Kaklamani, and I. S Venieris. Communication network design using particle swarm optimization. In *2008 International Multiconference on Computer Science and Information Technology*, pages 915–920, 2008.

[15] Judea Pearl. *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley Longman Publishing Co., Inc., 1984.

[16] Rafał Polak, Dariusz Laskowski, Robert Matyszkiel, Piotr Łubkowski, Łukasz Konieczny, and Rafał Burdzik. Optimizing the data flow in a network communication between railway nodes. In Mirosław Siergiejczyk and Karolina Krzykowska, editors, *Research Methods and Solutions to Current Transport Problems*, pages 351–362, Cham, 2020. Springer International Publishing.

[17] E. Sieni, P. Di Barba, and M. Forzan. Migration nsga: method to improve a non-elitist searching of pareto front, with application in magnetics. *Inverse Problems in Science and Engineering*, 24(4):543–566, 2016.

[18] Tran Anh Tuan, Long P. Hoang, Dung D. Le, and Tran Ngoc Thang. A framework for controllable pareto front learning with completed scalarization functions and its applications, 2023.

[19] Chi Tung Tung and Kim Chew. A bicriterion pareto-optimal path algorithm. *Asia-Pacific Journal of Operations Research*, 5:166–172, 1988.

[20] Chi Tung Tung and Kim Chew. A multicriteria pareto-optimal path algorithm. *European Journal of Operations Research*, 62:203–209, 1992.

X. Appendix

A. List of symbols and their meaning

Symbol	Meaning
G	graph
V	set of vertices in G
E	set of edges in G
λ_{uv}	edge weight of edge $(u, v) \in E$; rate parameter for exponential distribution of edge $(u, v) \in E$
$r_{uv}^{(k)}$	the k'th meeting for edge $(u, v) \in E$
$\mathfrak{X}_{uv}^{(k)}$	interarrival time between meetings $r_{uv}^{(k-1)}$ and $r_{uv}^{(k)}$; exponentially distributed random variable with rate parameter λ_{uv}
$t_{uv}^{(k)}$	time at which meeting $r_{uv}^{(k)}$ happens
$\mathfrak{P}_{uv}$	set containing all paths between vertices u and v in G
$\mathfrak{T}_{P_{uv}}$	arrival time of information at vertex u starting from vertex v, along path $P_{uv} \in \mathfrak{P}_{uv}$; conjectured hypoexponentially distributed random variable
m	number of criteria
$E^{\text{IN}}(v)$	set of all edges coming into vertex v
$E^{\text{OUT}}(v)$	set of all edges coming out of vertex v
$c(x)$	cost of parameter x
Θ	matrix parameter for the hypoexponential distribution
γ	vector parameter for the hypoexponential distribution
μ	mean of the hypoexponential distribution
σ^2	variance of the hypoexponential distribution
$\bar{X}$	sample mean of simulation; real value to the predicted μ
s^2	sample variance of the simulation; real value to the predicted σ^2
s	starting vertex for the labeling algorithm
t	target vertex for the labeling algorithm
$\theta(y)$	label for vertex y
$\alpha(y)$	cost of reaching vertex y from the starting vertex s
$\mathfrak{F}(l)$	combining function
w	weights per criterion used in the weighted combining function

$\mathfrak{F}(w, l)$	weighted combining function
$h^*(u, v)$	cost value of the minimum cost path $P_{uv} \in \mathfrak{P}_{uv}$ from u to v using $\mathfrak{F}(c(i, j))$ as the cost of traveling along each edge $(i, j) \in P_{uv}$
$q_j(u, v)$	cost value of the minimum cost path $P_{j,uv} \in \mathfrak{P}_{j,uv}$ from u to v out of all criteria, given each criterion's path is the minimum cost path for that criterion
$e(\theta(y))$	evaluation value for label $\theta(y)$
L	set of all tentative labels
Γ	set of all Pareto-optimal solutions; Pareto-frontier after labeling algorithm has finished
ξ	Pareto-optimal path in Γ
DEN_G	density of G