Communication Papers of the 20th Conference on Computer Science and Intelligence Systems (FedCSIS) pp. 55–62
DOI: 10.15439/2025F1746
ISSN 2300-5963 ACSIS, Vol. 45

Modeling and optimizing flow networks with several constrains using sequential dynamical systems

Jens Dörpinghaus*†, Michael Tiemann*†, Robert Helmrich*‡,
* Federal Institute for Vocational Education and Training (BIBB), Bonn, Germany
† University of Koblenz, Germany,
Email: jens.doerpinghaus@bibb.de, https://orcid.org/0000-0003-0245-7752
‡ University of Bonn, Bonn, Germany

***Abstract*—This paper introduces a novel framework for modeling and optimizing flow networks with multiple constraints using Sequential Dynamical Systems (SDS). We have extended the Sequential Flow Network (SFN) definition to encompass both Sequential Flow Networks (SFN) and Bounded Sequential Flow Networks (bSFN), which incorporate directional constraints and weighted transitions. These models can be utilized to simulate intricate real-world applications, such as educational pathways and labor market issues. In order to optimize local flow to specific nodes with minimal global impact, we propose three novel approaches: a linear programming formulation and two greedy heuristics. The evaluation metrics employed are defined as a means to balance local improvement and global disturbance. The efficacy of these methods is evaluated through experimentation on both artificial and real-world-inspired random networks. Notwithstanding the encouraging results and observations yielded by the experimental analysis of random graphs, suggestions for further research will be made in order to overcome the limitations of the present study.**

I. Introduction

A CONSIDERABLE number of real-world problems manifest as phenomena or dynamics over networks and graphs. For instance, urban traffic and transportation networks can be represented as graphs, as illustrated in [1]). As another example, the spread of disease on social contact graphs is naturally represented in graphs [2]. Other examples include packet flow in information and engineering networks, such as cell phone communication, gene annotation and gene regulatory network (GRN), or optimization of SDS schedules, see [3]. Thus, SDSs use networks for modeling, simulation, and analysis. Networks are also widely used in other context for data modeling and analysis, see for example [4].

The generic class of Graph Dynamical Systems (GDS) is distinct from other dynamical systems. These systems operate on discrete time and may utilize a finite number of states. Consequently, classical dynamical systems theory and tools frequently prove to be inapplicable. While GDS are rooted in discrete mathematics, algebra, combinatorics, graph theory, and probability theory, they are primarily utilized within the context of computer simulation.

The central research question guiding this study is concerned with the utilization of SDSs in the modeling and optimization with several constraints of flow networks on natural numbers, characterized by linear transition functions. The primary emphasis of this study will be on real-world problems derived from labor market research. A comprehensive literature review will precede a concise discussion of the methods, tools, and theory for validation and theoretical insights of SDSs that are necessary for modeling flow networks with SDSs. In this study, we will propose three methodologies for addressing the aforementioned problem. The first is an approach based on linear programming, and the second and third are greedy heuristics. Subsequent to this discussion, we will present and analyze a series of experimental results. The study's conclusions and outlooks are articulated in the final section.

II. Literature Review

Research on GDS and SDS remains limited. For a comprehensive introduction to SDSs, see [5] and [3]. A close relationship exists between these models and Generalized Cellular Automata with parallel update schemes, see [6]. SDSs with sequential update schemes were introduced between 1999 and 2001 by Barrett et al, see [7]. Another related concept is that of stochastic graph dynamical systems, see [8].

The examination of flow networks in graphs is not a novel concept; see [9], [10]. However, to the best of our knowledge, there is a lack of literature on modeling flow in graphs with dynamical systems. The optimization of flow, whether local or global, is a subject of study within the framework of classical dynamical systems theory, see [11]. This concept has also been explored in the context of distributed systems [12], chemical systems [13], and traffic networks [14]. However, these issues are frequently addressed through the implementation of optimization methodologies or, in certain instances, artificial intelligence algorithms, see [15], [16]

In summary, the field of SDSs is not generally associated with flow networks. The objective of this study is to examine the feasibility of leveraging methodologies from linear programming to enhance the operational efficiency of flow networks in SDSs, which are characterized by multiple constraints.

III. Method

In this section, we will first introduce Sequential Dynamical Systems (SDS) and then develop the novel concept of flow networks modeled with SDS. Subsequently, the issue of local optimization of these flow networks will be presented.

The following three approaches will be introduced: a linear programming (LP) approach and two greedy approaches.

A. Sequential Dynamical System

An SDS consists of the following parts, which we will illustrate with a continuous example introduced by [3]. However, we will not strictly follow their notation and will add other remarks that focus on our research. First, we need a **Graph** $G = (V, E)$ with vertices V and edges E.

Example III.1. *For example we may use a circular graph on four nodes:* $V = \{v_0, ..., v_3\}$ *and* $E = \{(v_0, v_1), (v_1, v_2), (v_2, v_3), (v_3, v_0)\}$.

Each node has a particular **vertex state** x_i from a state set K, for example $K = \mathbb{F}_2 = \{0, 1\}$. This results in a **system state**, which is also called the *configuration* of an SDS. For G in the example III.1 the system state contains four vertex states:

$$x = (x_0, x_1, x_2, x_3).$$

Next, we use a G**-local function** $F_i : K^n \to K^n$, which is also called a *local transition function*. It takes the system state as input.

For each vertex $v_i \in V$ $f_{v_i} : K^{d(v_i)+1} \to K$ is called the *vertex function*. Here, $d(v)$ denotes the degree of vertex v and $N(v)$ its closed neighborhood. The input set is vertex state of the node v_i and the vertex state of all its neighbors, denoted by $x[N(v_i)]$. We can define the local function F_v node-wise as by

$$F_{v_i} = F_i = (x_0, ..., x_{i-1}, f_{v_i}(x[N(v_i)]), x_{i+1}, ..., x_n)$$

Example III.2. *Continuing Example III.1 we may set*

$$F_0(x_0, x_1, x_2, x_3) = (nor_3(x_0, x_1, x_3), x_1, x_2, x_3)$$
$$F_1(x_0, x_1, x_2, x_3) = (x_0, nor_3(x_0, x_1, x_2), x_2, x_3)$$
$$F_2(x_0, x_1, x_2, x_3) = (x_0, x_1, nor_3(x_1, x_2, x_3), x_3)$$
$$F_3(x_0, x_1, x_2, x_3) = (x_0, x_1, x_2, nor_3(x_2, x_3, x_0))$$

It is important to note that this function can only change the state of vertex i. We apply the updates sequentially and therefore need to define a **order**, e.g. $\pi = (0, 1, 2, 3)$.

To start the system, we define a **initial state** $x^0 = (x_0^0, ..., x_n^0)$. Typically, the context provides sufficient clarity regarding the intended state, thereby obviating the necessity for dual indexes.

Example III.3. *Continuing example III.1, we may set*

$$x^0 = (x_0, x_1, x_2, x_3) = (1, 1, 0, 0)$$

By applying the maps we get $(1, 1, 0, 0) \overset{F_0}{\longmapsto} (0, 1, 0, 0) \overset{F_1}{\longmapsto} (0, 0, 0, 0) \overset{F_2}{\longmapsto} (0, 0, 1, 0) \overset{F_3}{\longmapsto} (0, 0, 1, 0)$.

Effectively we have applied the composed map $F_3 \circ F_2 \circ F_1 \circ F_0$. In a more algorithmic perspective, each step of an SDS involves n substeps:

Algorithm 1 System Update

```
for i = 1 to n do
    x_{π(i)} = f_{π(i)}(x[N(v_{π(i)})])
end for
```

In summary, A SDS is thus defined by a graph G, $\mathbf{F}_G = (F_v)_{v \in V}$, which is the vertex-indexed family of vertex functions and π:

Definition III.4 (Sequential Dynamical System, SDS, see [3])**.** *Let* $G = (V, E)$ *be a graph, let* $(f_v)_{v \in V}$ *be a family of vertex functions, and let* $\pi = (v_{\pi(1)}, v_{\pi(2)}, ..., v_{\pi(n)})$ *be a permutation of the vertices of G. The sequential dynamical system (SDS) is the triple*

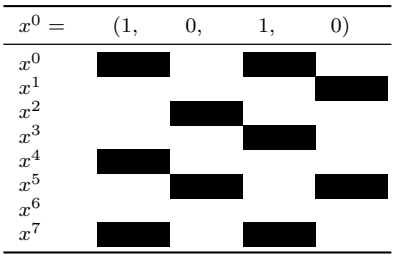
$$(G, (F_v)_v, \pi).$$

Its associated SDS-map is $[\mathbf{F}_G, \pi] : K^n \to K^n$ *defined by*

$$[\mathbf{F}_G, \pi] = F_{\pi_n} \circ F_{\pi_{n-1}} \circ ... \circ F_{\pi_1}.$$

Often, scenarios are considered where G is undirected. Thus, if not specified, we will assume that G is undirected. The application of the G-local map F_v is the *update of vertex* v, and the application of $[\mathbf{F}_G, \pi]$ is a *system update*, see Algorithm 1.

To visualize the behavior of an SDS we may use a table representing all node states in a table, where each column represents a particular node state and each row represents a system update. In the case of $K = \mathbb{F}_2$, a vertex state that is zero is represented as a white square and a vertex state that is one is represented as a black square. Consider the previous example:



This table represents the so-called *forward orbit* of $x = (1, 0, 1, 0)$:

Definition III.5 (Forward Orbit, see [3])**.** *Let* x *be a system state of a SDS with system update function* $[\mathbf{F}_G, \pi]$*. The forward orbit is given by*

$$O^+(x) = (x, [\mathbf{F}_G, \pi](x), [\mathbf{F}_G, \pi]^2(x), [\mathbf{F}_G, \pi]^3(x), ...).$$

However, this only represents the *forward orbit* of an initial state. To visualize a complete SDS we may use Phase spaces, see [3] for more details. It is obvious that we can only visualize a finite state of system states with these approaches. Thus, it is common to analyze the dynamics of sequential dynamical systems defined using classical Boolean functions. They have several nice properties, including symmetry:

Definition III.6. *We say that a function $f(x_1, ..., x_n)$ is symmetric if the order in which we describe its inputs does not change the output: i.e. if $f(x_1, ..., x_n) = f(x_{\pi(1)}, ..., x_{\pi(n)})$, for any permutation π.*

Some functions like nor, nand, or and and are all symmetric. Other functions may not, especially the linear functions which we will consider in the next section are generally not symmetric.

B. Flow Networks

We will now consider flow networks. Usually they are studied as another problem in graph theory where shortest paths can be useful [9], [10]. In fact, the Maximum Flow Problem is e so useful that we can apply them to many practical problems. Suppose we want to transport a good from one point to another, for example water, natural gas, oil, or electricity. In these networks edges refer to some kind of pipe or route to transport these good, not only pipes, but also roads, or railways. Here, the question is: How can we send as much as possible? Or in other words: How can we maximize the flow?

However, when simulating flow with dynamical systems, the question is usually rather: How can we manipulate the flow so that particular nodes get more goods, for example without affecting the whole other nodes? So for example, how can one particular factory get more resources without the need that other factories decrease their production.

In this case, a flow network $G = (V, E)$ is a directed graph comprising, here every node $v \in V$ may be a source node and a target node. We set $K = \mathbb{N}$. Each edge $e_i \in E$ has a weight $w(e_i) \in \mathbb{R}$ which identifies the share of how many goods from the source node "go" to the target node. Attention: This is not the capacity, but a real share. When we define $E^-(v)$ as the set of incoming edges for node $v \in V$, we can define the the generic class of SDS flow networks:

Definition III.7 (SDS Flow Network (SFN))**.** *Let $G = (V, E)$ be a directed weighted graph with edge weights $w : E \to \in \mathbb{R}$. Let $(f_v)_{v\in V}$ be a family of vertex functions with $f_v : K^{d(v_i)+1} \times E^-(v) \to K$, and let $\pi = (v_{\pi(1)}, v_{\pi(2)}, ..., v_{\pi(n)})$ be a permutation of the vertices of G. The sequential dynamical system (SDS) defined by $(G, (F_v)_v, \pi)$ is called a sequential flow network (SFN).*

In summary, the basic difference to an SDS is the usage of weights in the vertex functions. The difference to generic flow networks is that every node is a source and target node at the same time. However, these networks may have further constrains and we will develop some of them by starting with a very simple example and extend it.

We have n nodes $v_1, ..., v_n$ and we may group all other nodes in q groups $V_1, ...V_q$ to keep track of the update order. This step is not technically necessary but allows to define constrains.

We can model a small network with educational pathways, as described in Figure 1. Here, $V = \{\underbrace{V_0}_{v_0}, \underbrace{V_1}_{v_1,v_2,v_3}, \underbrace{V_3}_{v_4,v_5}\}$,

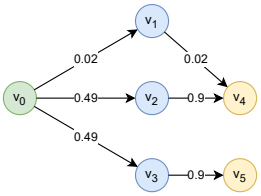


Fig. 1: An example of educational pathways. Green nodes are in V_0, blue nodes in V_1 and yellow nodes in V_2

and we have six nodes with a certain weight, see Figure 1. Here, the group V_0 represents the 'incoming' actors on the labor market, for example school-leavers, V_1 represents school-leaving certificates and V_3 vocational or academical education. Since education and qualification is usually not 'lost', we can assume that a particular share of people with a given qualification achieve another qualification during a given time, for example a year. Here, $w(e_i) \in [0, 1]$. In our artificial and simplified example, 49% of all people entering the labor market receive school leaving certificate v_3 and 90% of them get a vocational degree v_5. Note, that in this example we only allow edges between nodes u, v with $u \in V_i$ and $v \in V_{i+1}$.

With this, We can now define the the G-local function as

$$F_i(x_0, ..., x_n) = (x_1, ..., \underbrace{x_i + \sum_{j\in N^-(v_i)} w((v_j, v_i))\, x_j}_{\text{position } i}, ..., x_n).$$

This function sums over all incoming edges and adds the share defined as edge weight of the system state of the incoming node.

By using the reverse order $\pi = (n, n-1, ..., 2, 1, 0)$ we can now define an SDS $(G, (F_v)_v, \pi)$ and compute the forward orbit for a given initial state, for example

$$x = (10000, 2000, 80000, 40000, 130000, 110000).$$

Thus, we will apply the composed map $F_5 \circ F_4 \circ F_3 \circ F_2 \circ F_1 \circ F_0$ step by step as follows:

$$F_5 = (10000, ..., \underbrace{\underbrace{110000}_{=x_5} + (\underbrace{0.9}_{=w((v_3,v_5))} \cdot \underbrace{40000}_{=x_3})}_{=146{,}000})$$

$$F_4 = (...., \underbrace{\underbrace{130000}_{=x_4} + (\underbrace{0.02}_{=w((v_1,v_4))} \cdot \underbrace{2000}_{=x_1}) + (\underbrace{0.9}_{=w((v_2,v_4))} \cdot \underbrace{80000}_{=x_2})}_{=202{,}040}, ...)$$

$$F_3 = (...., \underbrace{\underbrace{40000}_{=x_3} + (\underbrace{0.49}_{=w((v_0,v_3))} \cdot \underbrace{10000}_{=v_0})}_{=44{,}900}, ...)$$

Here, the colors refer to the coloring in Figure 1. So we get the following forward orbit:

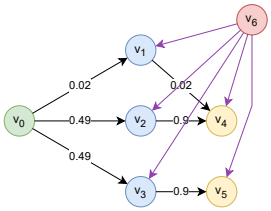


Fig. 2: An example of educational pathways. Green nodes are in V_0, blue nodes in V_1 and yellow nodes in V_2, red nodes in V_3

	v_0	v_1	v_2	v_3	v_4	v_5
x^0	10,000	2,000	80,000	40,000	130,000	110,000
x^1	10,000	2,200	84,900	44,900	203,040	146,000

As we can see, the number of people with a certain education raises every time step. This is not very natural, and we can thus extend the model by adding negative weights $w(e_i) \in [-1, 1]$ and adjusting the G-local function with

$$u(x_i, x_j) = \begin{cases} e((v_j, v_i))x_j & e((v_j, v_i)) \geq 0 \\ e((v_j, v_i))x_i & e((v_j, v_i)) < 0 \end{cases}$$

$$F_i(x_0, ..., x_n) = (x_1, ..., \underbrace{x_i + \sum_{j \in N^-(v_i)} u(x_i, x_j)}_{\text{position } i}, ...). \quad (1)$$

With this, we can also model an increasing flow with backward egdes. Extending our previous example with another node in V_3 and backward edges with a small share to all other nodes, see Figure 2, we can model the share of people leaving the labor market, for example because of death or retirement. Here, $e(v, v_6) = -0.01 \, \forall v \in V \setminus \{v_0, v_6\}$.

Again, we can compute the forward orbit. Let

$$x = (10000, 2000, 80000, 40000, 130000, 110000, 0)$$

be the inital state. Then

$$F_6 = (10,000, ..., 0)$$

$$F_5 = (10,000, ..., \underbrace{110000 - (0.01 \cdot 110000) + (0.9 \cdot 40000)}_{=144,900}, 0)$$

$$F_4 = (...., \underbrace{130000 - (0.01 \cdot 130000) + (0.9 \cdot 80000) + (0.02 \cdot 2000)}_{=200,740}, ...)$$

$$F_3 = (...., \underbrace{40000 - (0.01 \cdot 40000) + (0.49 \cdot 10000)}_{=44,500}), ...)$$

$$F_2 = (...., \underbrace{80000 - (0.01 \cdot 80000) + (0.49 \cdot 10000)}_{=84,100}), ...)$$

$$F_1 = (10,000, \underbrace{2000 - (0.01 \cdot 2000) + (0.02 \cdot 10000)}_{=2,180}), ...)$$

$$F_0 = (10,000, ..., 0)$$

	v_0	v_1	v_2	v_3	v_4	v_5	v_6
x^0	10,000	2,000	80,000	40,000	130,000	110,000	0
x^1	10,000	2,180	84,100	44,500	200,740	144,900	0

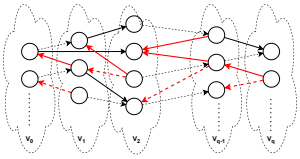


Fig. 3: Illustration of the concept of Bounded SDS Flow Networks: The positive flow goes from left to right, the negative flow from right to left

Which leads to the following forward orbit: We can make two observations: First, the vertex state of v_6 never changes. However, if we add 'backward'-edges, e.g. (v_5, v_6) with $w(v_5, v_6) = -w(v_6, v_5)$, this simulates the desired behavior. The second observation is that the newly added negative edges conflict with our assumption that edges are only between two nodes in groups V_i and V_{i+1}. This rule is not realistic, especially since some education may rely on other education in the same group, for example elementary school as dependency for higher school degrees. Thus, we define a specific subset of SDS flow networks:

Definition III.8 (Bounded SDS Flow Network (bSFN))**.** *Let $G = (V, E)$ be a directed weighted graph with edge weights $w : E \to \in \mathbb{R}$. Let $(f_v)_{v \in V}$ be a family of vertex functions with $f_v : K^{d(v_i)+1} \times E^-(v) \to K$, and let $\pi = (v_{\pi(1)}, v_{\pi(2)}, ..., v_{\pi(n)})$ be a permutation of the vertices of G.*

All n nodes $v_0, ..., v_n$ are grouped in q groups $V_0, ...V_q$. We allow edges $e = (u, v)$ with $w((u, v)) \geq 0$ only between nodes $u \in V_i, v \in V_j$ with $i \leq j$ or edges $e = (u, v)$ with $w((u, v)) < 0$ only between nodes $u \in V_i, v \in V_j$ with $j \leq i$.

The sequential dynamical system (SDS) defined by $(G, (F_v)_v, \pi)$ is called a bounded flow network (bSFN).

If not otherwise mentioned, we will assume that π is the reverse order $(n, ..., 1, 0)$. This means that the flow in this network is bounded by V_0 and V_q, see Figure 3 for an illustration. We can now study scenarios where we apply methods to optimize the local flow in these networks.

C. Local Optimization of Flow Networks

In general, the local optimization of flow networks refers to the maximization of the incoming flow to one particular node. So let $v_\iota \in V_i$ with $0 < i < q$ and x an initial system state and $x' = [\mathbf{F}_G, \pi](x)$. How can we change G so that we maximize x'_ι whereas we want to keep $\delta_j = |x_j - x'_j| \; \forall j \in [0, ..., n]$, $j \neq \iota$ minimal. In other words, after the system update the system state for v_ι should be maximized whereas ideally all other nodes have the same system state or change minimally.

While generally we can use or add new positive nodes from all nodes in V_i and all V_j with $j < i$ and use or add new

negative nodes from all nodes in V_i and all V_j with $j > i$, we may restrict the candidate set for positive edges to $C^+ \subseteq E$ and the candidate set for negative edges to $C^- \subseteq E$.

We set $c^+_{ij} = w((v_i, v_j))$ for $v_i, v_j \in V$, $i \leq j$ if $(v_i, v_j) \in E$ and else $c^+_{ij} = 0$. Similarly, we set $c^-_{ij} = w((v_i, v_j))$ $v_i, v_j \in V$, $i \geq j$ if $(v_i, v_j) \in E$ and else $c^+_{ij} = 0$. In Figure 4 we describe the adjacency matrix representation, where each entry holds the weight of an edge an 0 if no edge exists. The lower triangular part (green) represents values in c^+, the upper triangular part the values in c^-:

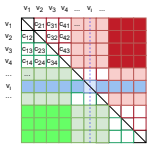


Fig. 4: Adjacency matrix representation

It is easy to see that the updated value of v_ι can be computed by considering the ι-th row of the matrix. So we want to maximize

$$x'_\iota = x_\iota + \sum_{v_i \in V;\ (i,\iota) \in E\ i<\iota} c^+_{i\iota} x_i - \sum_{v_i \in V;\ (i,\iota) \in E\ i>\iota} c^-_{i\iota} x_\iota.$$

However, only those values of $c_{i\iota}$ that are in C^+ or C^- are subject to optimization. All other variables are fixed. In addition, this formula does not update those vertex states which are updated before $\pi^{-1}(\iota)$, leading to an approximation of the optimal solution.

However, if we restrict the update schema to $\pi = (n - 1, ..., 1, 0)$, the update of v_i does not influence any other nodes in the current system update:

Lemma III.9. *Let $\pi = (n-1, ..., 1, 0)$ be the update schema for a Bounded SDS Flow Network on n nodes. Then, the vertex state of x_i is not included in the vertex update function of any other node v_j with $j < i$.*

Proof. Let v_j be a node with a vertex update function that includes x_i with $i > j$. Then, according to Formula 1, and edge (v_i, v_j) with positive weight would exist. This is a contradiction to Definition III.8. □

More generic, changing the value of x'_ι influences all other summands for other update values x'_j, $j \neq \iota$ which is represented by the ι-th column in the adjacency matrix. If π is the ordering $(0, 1, ..., n)$, this only affects the lower triangular part, if $\pi = (n, ..., 1, 0)$ it only affects the upper triangular part.

From now on, we will assume the update schema is $\pi = (n - 1, ..., 1, 0)$. Since – unless C^+ and C^- includes edges not connected to v_ι – in the first system update only x_ι is changed, we will compare $[\mathbf{F}_G, \pi]^2(x)$ to $[\mathbf{F}_{G'}, \pi]^2(x)$ where G' is the graph with adjusted weights. We can then compute the distance between the expected and modified vertex state of all other nodes as

$$\delta = |[\mathbf{F}_G, \pi]^2(x) - [\mathbf{F}_{G'}, \pi]^2(x)|$$

In summary, we can model this as optimization problem using a linear program:

$$\max \quad z = x'_\iota - \delta \tag{2}$$

$$\text{s.t.} \quad x'_\iota = x_\iota + \sum_{v_i \in C^+} c^+_{i\iota} x_i - \sum_{v_i \in C^-} c^+_{i\iota} x_\iota \tag{3}$$

$$\delta = \sum_{j=0,...,n-1} |([\mathbf{F}_G, \pi]^2(x))_j - ([\mathbf{F}_{G'}, \pi]^2(x))_j| \tag{4}$$

$$c^+_{ij} \in [0, 1] c^-_{ij} \in [-1, 0] \text{ for all } c^+_{ij} \in C^+, c^-_{ij} \in C^- \tag{5}$$

Here, line 3 gives the objective function for the updated system state, omitting the update of other node's state. Line 4 returns the influence of changing edge weights, using the modified state of v_ι.

Example III.10. *Coming back to our initial example in the last section, and let $C^+ = \{e_{1,4}, e_{2,4}, e_{3,4}\}$, the optimization approach is as follows:*

$$\max \quad z = 2000c14 + 80000c24 + 40000c34 - \delta + 128700.0 \tag{6}$$

$$\text{s.t.} \quad \delta = -83.2 + 4160c14 + 163300c24 + 84100c34 \tag{7}$$

The optimal solution in this trivial case is $c_{14} = c_{24} = c_{34} = 1$.

However it is also possible to use a greedy approach to maximize the local flow. The simplest approach would just set the value for all incoming edges to node v_ι to the maximum positive value. This means, if we restrict the edge weights to the range $[-1, 1]$ to set all edge weights in C^+ to 1 and in C^- to zero:

Algorithm 2 Greedy 1

Ensure: bSDS-FN $G = (V, E)$, edge weights restricted to $[a, b]$, allowed edge sets C^+ and C^-, and a node $v_\iota \in V$

1: **for** $(v, v_\iota) \in C^+$ **do**
2: $\quad w((v, v_\iota)) = b$
3: **end for**
4: **for** $(v, v_\iota) \in C^-$ **do**
5: $\quad w((v, v_\iota)) = \max(0, a)$
6: **end for**

However, this brute-force approach will significantly influence parts of the network. The changes are limited to those nodes, which are reachable from node v_ι. Considering Example 2, we see that changing x_4 or x_5 has no influence on other nodes. We can measure this influence using the betweenness centrality measure [17], [18], which is the sum of the fraction of all-pairs shortest paths that pass this particular

node; it was first introduced by [19]. Given a node v, it calculates all shortest paths in a network $P_v(k,j)$ for all beginning and ending nodes $k,j \in V$. If $P(k,j)$ denotes the total number of paths between k and j, the importance of v is given by the ratio of both values. Thus, the betweenness centrality according to [20] is given by

$$bc(v) = \sum_{k\neq j, v\neq k, v\neq j} \frac{P_v(k,j)}{P(k,j)} \cdot \frac{2}{(n-1)(n-2)},$$

Coming back to Example 2, we see that the bc-value of all nodes is zero, except $bc(v_1) = bc(v_2) \approx 0.01$, and $bc(v_3) \approx 0.08$.

Algorithm 3 Greedy 2

Ensure: bSDS-FN $G = (V,E)$, edge weights restricted to $[a,b]$, allowed edge sets C^+ and C^-, and a node $v_\iota \in V$

1: **for** $(v,v_\iota) \in C^+$ **do**
2: $\quad w((v,v_\iota)) = \min\{\max\{w((v,v_\iota)) \cdot bc(v_\iota), bc(v_\iota)\}, b\}$
3: **end for**
4: **for** $(v,v_\iota) \in C^-$ **do**
5: $\quad w((v,v_\iota)) = \max\{\min\{w((v,v_\iota)) \cdot bc(v_\iota), -bc(v_\iota)\}, a\}$
6: **end for**

D. Evaluation Metrics

As we have discussed in the last sectrion, the optimization approach has two foci: First, it tries to optimize the flow towards one particular node, second, it tries to keep the influence on the whole network flow minimal. Thus, while we want to maximize

$$\delta_k(x_\iota) = x_\iota^k - x_\iota^{k-1},$$

we want to minimize

$$\Delta_k(X) = \sum_{i\neq\iota} |x_i^k - x_i^{k-1}|.$$

Thus, our evaluation will be based on both metrics.

IV. Experimental Results

We used Python 3.11.2 with Pulp and NetworkX for creating random instances and implement the greedy heuristic as well as the Linear Program. We used GLPK (GNU Linear Programming Kit) 5.01 to solve the linear program. Random instances are build for a particular number of nodes n and a given probability $p \in [0,1]$ that two nodes are connected when not violating the conditions defined previously. Edge weights are randomly chosen from the same interval. We define C^+ and C^- with all possible direct edges. For all instances, we used 40 iterations to compare the results with Δ_k and δ_k with $k = 2$.

A. Small networks

In Table I we present the average, minimal and maximal error rates for $n = 200$ and $n = 400$ and different values for p. For a visualization of the corresponding values we refer to Figures 5 and 6.

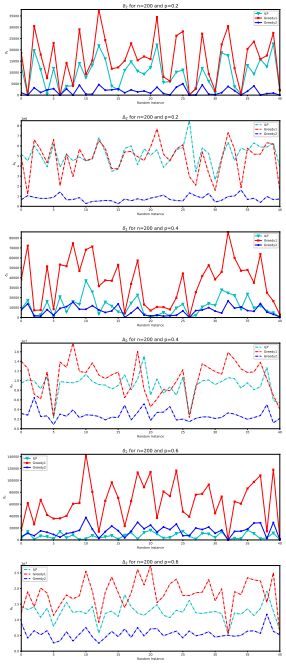


Fig. 5: Measures for different values of p and $n = 200$

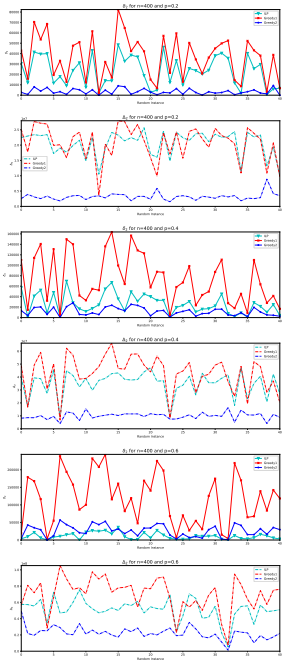


Fig. 6: Measures for different values of p and $n = 400$

For flow networks which are not so dense ($p = 0.2$) we see that the LP-approach outperforms Greedy1 with both metrics. However, for flow networks with more edges, Greedy1 produces better local improvements δ, but also much higher global errors Δ. Greedy2 generally produces best results, while keeping the global error low – but also with lower local

improvement. However, comparing to the IP solutions tend to be at least comparable and for larger p IP even outperforms Greedy2. The overall behavior is similar for larger instances, see the results for $n = 400$.

B. Real-world inspired random networks

The German Labor Market Ontology (GLMO) was developed to facilitate the modeling of labor market flows. The ontology in question was developed principally on the basis of two sources: the multilingual European ontology for occupations and skills, known as ESCO, and the German classification of occupations, denoted as KldB. In addition to the comprehensive classification system of KldB occupations, the GLMO encompasses sets of skills, tools, and educational training relevant to the German labor market. These competencies are delineated by the BA and will be designated as BA skills, tools, and educational training in the subsequent discussion. In the ontology, the concepts are organized in a hierarchical structure and mapped to KldB-occupational unit groups [21], [22]. In [23], the ontology was expanded by incorporating data from BERUFENET, an online portal that provides information on KldB occupational titles. BERUFENET organizes these occupational titles into distinct study fields, activity fields, and activity areas. In addition, this encompasses mappings to associated or alternative occupations, supplementary qualifications, and other CVET categories from KURSNET, along with information fields comprising extensive additional information (e.g., competencies, abilities, knowledge, and skills, cf. [24], [25]). It is possible to create a comprehensive model of the German labor market that incorporates educational pathways, in conjunction with additional data, such as information regarding individuals with specific training.

In order to analyze the efficacy of our approach for these networks, we created an additional set of larger random instances. Here, 2,000 nodes were utilized, and the probability p set at 0.4. The results are presented in Figure 7 and Table II. It is evident that Greedy2 demonstrates superior performance in comparison to IP for larger instances. In order to determine the most efficacious course of action, a rigorous examination of the pertinent real-world applications is imperative. This involves a judicious evaluation of the relative merits of a modest yet substantial increase, characterized by a limited global impact, as compared to a more pronounced increase accompanied by a concomitant global error (IP). Therefore, given the consideration of these three approaches, it can be concluded that a universal solution does not exist.

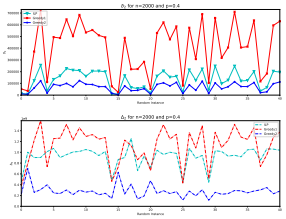


Fig. 7: Measures for different values of p and $n = 2000$

V. Conclusions and Outlook

In this study, a novel approach for modeling and optimizing flow networks with multiple constraints was introduced. This approach was developed using the framework of Sequential Dynamical Systems (SDS). The extension of the system dynamics (SDS) framework to encompass sequential flow networks (SFN) and a subclass of bounded SDS flow networks (BSFN) has yielded a highly versatile structure for the simulation and analysis of complex networked systems. In these systems, each node possesses the capacity to function as both a source and a sink, thereby enhancing their operational flexibility.

We hereby present three approaches for local flow optimization within these networks: a linear programming formulation and two greedy heuristics. Through extensive experimentation on both random and real-world inspired random network scenarios, it was found that the linear programming method generally yields superior local improvements in node-specific flow while maintaining control over global impact. However, for large-scale networks or instances with high edge densities, the Faster Greedy2 heuristic offers a compelling balance between performance and computational efficiency.

The findings of this study underscore the efficacy of SDS-based flow modeling in facilitating precise manipulation of local flows within the confines of bounded and interpretable constraints. This attribute renders it particularly well-suited for applications in educational pathways, labor market modeling, and other socioeconomic systems. The incorporation of negative weights facilitates the incorporation of realistic dynamics, such as attrition or regression in states, thereby enhancing the expressiveness of traditional flow models.

A number of promising avenues emerge for future research. Firstly, extending the model to accommodate stochastic or time-dependent edge weights has the potential to enhance realism, particularly in dynamic environments such as transportation or information networks. Secondly, the incorporation of machine learning algorithms to adaptively learn optimal edge weights from historical data has the potential to enhance the system's predictive capabilities and adaptability.

Additionally, the exploration of other update schemes or hybrid SDS models has the potential to enhance scalability and model expressiveness. Another area of interest involves the formalization of stability and convergence properties of flow dynamics for bSFN, especially in feedback-rich or cyclic networks. The potential practical utility of this methodology can be demonstrated by its application to real-world datasets, such as longitudinal educational or employment data. Such an application could also drive domain-specific innovation.

References

[1] J. Hackl and B. T. Adey, "Estimation of traffic flow changes using networks in networks approaches," *Applied Network Science*, vol. 4, no. 1, p. 28, 2019.

TABLE I: Minimum, average and maximum for the two measures δ_k and Δ_k

			δ_2			Δ_2		
			min	avg	max	min	avg	max
$n = 200$	$p = 0.2$	LP	2621090.37	5152700.64	8508910.71	0.00	9615.49	22671.63
		Greedy1	1195140.97	4750200.37	7694436.14	0.00	16071.90	37858.45
		Greedy2	249131.21	781855.68	1633661.57	0.00	1713.43	5039.86
	$p = 0.4$	LP	2233706.46	9096606.18	15119007.11	314.91	11148.71	37028.85
		Greedy1	2376102.21	10494505.55	6504125.29	408.99	36787.32	16679.52
		Greedy2	885100.59	2801082.14	17792735.64	1.14	6697.86	86151.74
	$p = 0.6$	LP	5465072.63	12447128.95	19132120.24	274.29	6722.21	16558.73
		Greedy1	5502528.56	18335531.33	27356699.78	439.05	64364.35	143897.45
		Greedy2	2474992.97	5625483.43	11857886.62	0.87	15242.24	37814.45

			δ_2			Δ_2		
			min	avg	max	min	avg	max
$n = 400$	$p = 0.2$	LP	9849239.34	20549663.28	25666389.85	729.96	23253.02	48495.46
		Greedy1	3744006.19	20687091.99	27944082.81	2113.98	36346.70	82402.39
		Greedy2	1522814.45	3193905.60	8906831.00	3.26	3446.25	9280.32
	$p = 0.4$	LP	8724475.30	35006775.26	47207071.83	314.54	26304.02	69638.76
		Greedy1	6788012.04	42726164.84	66155919.59	3927.77	73750.12	163657.35
		Greedy2	3963835.51	9984506.68	16366638.62	392.56	11682.35	28340.86
	$p = 0.6$	LP	5318927.62	51140901.87	75497480.76	38.78	11219.22	36393.41
		Greedy1	7516038.04	69934679.46	105264267.88	2970.70	118048.94	243847.50
		Greedy2	1209262.25	22555721.87	49223081.37	294.02	26522.69	56786.72

TABLE II: Minimum, average and maximum for the two measures δ_k and Δ_k

			δ_2			Δ_2		
			min	avg	max	min	avg	max
$n = 2000$	$p = 0.4$	LP	440188141.23	921787895.81	1255290402.09	8280.66	135184.65	255070.07
		Greedy1	376467304.46	1130567882.79	1585149455.70	23580.63	405392.55	730495.57
		Greedy2	111937948.72	279409397.33	699888936.51	3068.38	67414.42	123757.16

[2] R. Shirzadkhani, S. Huang, A. Leung, and R. Rabbany, "Static graph approximations of dynamic contact networks for epidemic forecasting," *Scientific Reports*, vol. 14, no. 1, p. 11696, 2024.

[3] H. Mortveit and C. Reidys, *An introduction to sequential dynamical systems*. Springer Science & Business Media, 2007.

[4] J. Dörpinghaus, S. Schaaf, and M. Jacobs, "Soft document clustering using a novel graph covering approach," *BioData mining*, vol. 11, pp. 1–20, 2018.

[5] M. A. Porter and J. P. Gleeson, "Dynamical systems on networks," *Frontiers in Applied Dynamical Systems: Reviews and Tutorials*, vol. 4, p. 29, 2016.

[6] C. L. Barrett and C. M. Reidys, "Elements of a theory of computer simulation i: sequential ca over random graphs," *Applied Mathematics and Computation*, vol. 98, no. 2-3, pp. 241–259, 1999.

[7] C. L. Barrett, H. S. Mortveit, and C. M. Reidys, "Elements of a theory of simulation ii: sequential dynamical systems," *Applied Mathematics and Computation*, vol. 107, no. 2-3, pp. 121–136, 2000.

[8] C. Barrett, H. B. Hunt III, M. V. Marathe, S. Ravi, D. J. Rosenkrantz, and R. E. Stearns, "Modeling and analyzing social network dynamics using stochastic discrete graphical dynamical systems," *Theoretical Computer Science*, vol. 412, no. 30, pp. 3932–3946, 2011.

[9] R. Diestel, *Graphentheorie*, 3rd ed. Berlin: Springer-Verlag, 2006.

[10] S. O. Krumke and H. Noltemeier, *Graphentheoretische Konzepte und Algorithmen*, 2nd ed. Wiesbaden: Vieweg + Teubner, 2009.

[11] H. Ronellenfitsch and E. Katifori, "Global optimization, local adaptation, and the role of growth in distribution networks," *Physical review letters*, vol. 117, no. 13, p. 138301, 2016.

[12] J. F. Mota, J. M. Xavier, P. M. Aguiar, and M. Püschel, "Distributed optimization with local domains: Applications in mpc and network flows," *IEEE Transactions on Automatic Control*, vol. 60, no. 7, pp. 2004–2009, 2014.

[13] B. Grimstad, B. Foss, R. Heddle, and M. Woodman, "Global optimization of multiphase flow networks using spline surrogate models," *Computers & Chemical Engineering*, vol. 84, pp. 237–254, 2016.

[14] S. Scellato, L. Fortuna, M. Frasca, J. Gómez-Gardenes, and V. Latora, "Traffic optimization in transport networks based on local routing," *The European Physical Journal B*, vol. 73, pp. 303–308, 2010.

[15] G. V. Puskorius and L. A. Feldkamp, "Neurocontrol of nonlinear dynamical systems with kalman filter trained recurrent networks," *IEEE Transactions on neural networks*, vol. 5, no. 2, pp. 279–297, 1994.

[16] O. San, R. Maulik, and M. Ahmed, "An artificial neural network framework for reduced order modeling of transient flows," *Communications in Nonlinear Science and Numerical Simulation*, vol. 77, pp. 271–287, 2019.

[17] J. Dörpinghaus, V. Weil, C. Düing, and M. W. Sommer, "Centrality measures in multi-layer knowledge graphs," *Annals of Computer Science and Information Systems*, 2022.

[18] J. Dörpinghaus, V. Weil, and M. W. Sommer, "Towards modeling and analysis of longitudinal social networks," *Applied Network Science*, vol. 9, no. 1, p. 52, 2024.

[19] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, pp. 35–41, 1977.

[20] M. O. Jackson, *Social and Economic Networks*. Princeton: University Press, 2010.

[21] J. Dörpinghaus, J. Binnewitt, S. Winnige, K. Hein, and K. Krüger, "Towards a german labor market ontology: Challenges and applications," *Applied Ontology*, no. 18(4), pp. 1–23, 2023.

[22] J. Dörpinghaus and M. Tiemann, "Vocational education and training data in twitter: Making german twitter data interoperable," *Proceedings of the Association for Information Science and Technology*, vol. 60, no. 1, pp. 946–948, 2023.

[23] D. Martić, A. Fischer, and J. Dörpinghaus, "Extending the german labor market ontology with online data," in *INFORMATIK 2024*. Gesellschaft für Informatik eV, 2024, pp. 2019–2030.

[24] A. Fischer and J. Dörpinghaus, "Web mining of online resources for german labor market research and education: Finding the ground truth?" *Knowledge*, vol. 4, no. 1, pp. 51–67, 2024.

[25] J. Dörpinghaus, D. Samray, and R. Helmrich, "Challenges of automated identification of access to education and training in germany," *Information*, vol. 14, no. 10, p. 524, 2023.