Since January 2020 Elsevier has created a COVID-19 resource centre with free information in English and Mandarin on the novel coronavirus COVID-19. The COVID-19 resource centre is hosted on Elsevier Connect, the company's public news and information website.

European Journal of Operational Research 304 (2023) 99–112

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Modularity maximization to design contiguous policy zones for pandemic response

Milad Baghersad*, Mohsen Emadikhiav, C. Derrick Huang, Ravi S. Behara

Department of Information Technology & Operations Management, College of Business, Florida Atlantic University, Boca Raton, FL 33431-0991, USA

ARTICLE INFO

Article history:
Received 5 February 2021
Accepted 5 January 2022
Available online 13 January 2022

Keywords:
OR in disaster relief
Contiguous community detection
Modularity maximization
Pandemic response coordination
Column-generation algorithm

ABSTRACT

The health and economic devastation caused by the COVID-19 pandemic has created a significant global humanitarian disaster. Pandemic response policies guided by geospatial approaches are appropriate additions to traditional epidemiological responses when addressing this disaster. However, little is known about finding the optimal set of locations or jurisdictions to create policy coordination zones. In this study, we propose optimization models and algorithms to identify coordination communities based on the natural movement of people. To do so, we develop a mixed-integer quadratic-programming model to maximize the modularity of detected communities while ensuring that the jurisdictions within each community are contiguous. To solve the problem, we present a heuristic and a column-generation algorithm. Our computational experiments highlight the effectiveness of the models and algorithms in various instances. We also apply the proposed optimization-based solutions to identify coordination zones within North Carolina and South Carolina, two highly interconnected states in the U.S. Results of our case study show that the proposed model detects communities that are significantly better for coordinating pandemic related policies than the existing geopolitical boundaries.

1. Introduction

COVID-19 triggered devastating social and economic impacts around the world in a short amount of time. As of November 2021, more than 46 million have contracted the virus, and over 750 thousand people have died in the U.S. alone (CDC, 2021). Although pandemics are primarily a public health crisis, subsequent economic, social and political consequences are also significant. The total cost of COVID-19 in the U.S. is estimated to be more than $16 trillion (Cutler & Summers, 2020), larger than the costs of any previous manmade or natural disasters. The initial response to COVID-19 and the associated fiscal actions and lockdowns have resulted in $11.7 trillion, or close to 12% of global GDP, of negative economic impacts in the initial phase of the pandemic as of September 2020 (IMF, 2020). Further, the COVID-19 pandemic has had knock-on effects on other humanitarian disasters worldwide through restricted travel and support for ongoing relief efforts, because governments are looking inwards at protecting their own citizens and economies while pledging little for required international humanitarian support (The Lancet, 2020). Therefore, it is critical for decision-makers to respond with policy measures that save lives, contain economic fallout, and speed up recovery to minimize negative pollical and social impacts (Tanrisever, Shahmanzari, Eryarsoy & Şensoy, 2021).

In addition to any national vaccination effort, coordination of nonpharmaceutical interventions (NPIs) across governmental jurisdictions has been recommended as a critical element to combat the contagious disease, especially in decentralized and federal countries (OECD, 2020; Ruktanonchai, Floyd, Lai & Steele, 2020). Applying a broad and uniform NPI, such as full lockdowns, simultaneously in all regions may not be ideal and sometimes practical (if not impossible), as communities within a state or country may be at a very different level of an outbreak (Dhillon & Karan, 2020). This is highlighted by Dr. Anthony Fauci, U.S. President's Chief Medical Advisor and Director of the National Institute of Allergy and Infectious Diseases: "*We have to realize that we're a large country that has outbreaks in different regions, different states, different cities, that have different dynamics, and different phases in which they are in.*" (Watts, 2020). However, although decentralized governments may benefit from the flexibility of having customized responses within each jurisdiction, a lack of policy coordination may hinder the effectiveness (Gordon, Huberfeld & Jones, 2020), as the viruses do not stop at the jurisdictions' boundaries. Indeed, recent studies show significant policy differences from one region to another (Holtz, Zhao, Benzell & Aral, 2020). In the U.S., for example, the pandemic response is assigned to states and subsequent municipalities. Federal agencies, such as the Centers for Disease

* Corresponding author.
E-mail addresses: mbaghersad@fau.edu (M. Baghersad), memadikhiav@fau.edu (M. Emadikhiav), dhuang@fau.edu (C.D. Huang), rbehara@fau.edu (R.S. Behara).

https://doi.org/10.1016/j.ejor.2022.01.012
0377-2217/

Control and Prevention (CDC) and Federal Emergency Management Agency (FEMA), have limited authority to mandate local authorities to take united actions (Gordon et al., 2020). In this setting, two neighboring jurisdictions with a high social connection may have drastically different policies, such as closure or opening of restaurants and gyms, at the same time, rendering the effort to stop disease spreading ineffective.

Current research shows that a "place-based approach," where policies are customized and coordinated across interconnected locations, is effective in response to a contagious disease like COVID-19 (OECD, 2020). In a recent study, Ruktanonchai et al. (2020) find that appropriate coordination across countries that are interconnected by people movement can significantly increase the effectiveness of NPIs. The authors show that a resurgence of COVID-19 could happen five weeks earlier in Europe if well-connected countries end their interventions without coordinating with neighbors. Other studies have also shown that population mobility is an influential force in the transmission of COVID-19 (Chang, Pierson, Koh & Leskovec, 2021; Kraemer, Yang, Gutierrez & Scarpino, 2020). Moses (2021) recommends algorithmic thinking to reduce chaos and slow-downs in the COVID-19 vaccination rollout. Shahzamal, Mans, de Hoog and Jurdak (2020) develop an effective vaccination strategy, called the individual's movement-based vaccination strategy, where individuals are vaccinated based on their movement relative to public places without the need for detailed contact tracing information. Therefore, detecting communities based on the natural movement of people can be more effective for coordinating NPIs and vaccination campaigns, compared to relying on predefined geopolitical boundaries or at a national level.

Despite the importance of identifying coordination groups to combat a pandemic based on the natural movement of people, studies about finding the optimal set of locations or jurisdictions to create such groups are limited. In this study, we focus on identifying optimal coordination communities for applying NPIs and deploying vaccination strategies based on population mobility. To this end, a new mixed-integer quadratic-programming model based on the modularity maximization problem is developed. The model uses population mobility to identify highly interconnected locations and joins them to form coordination communities. The model guarantees that coordination communities are geographically contiguous, an important factor for effective coordination. It also allows for adjusting the number of jurisdictions assigned to a coordination community based on the policymakers' preferences. To our knowledge, such a community detection model based on the natural movement of people that guarantees the contiguity of communities does not exist. Due to the complexity of the problem, we develop a column-generation algorithm to solve the problem. The column-generation approach is coupled with an iterative pricing procedure to improve its performance. A heuristic algorithm is also developed to find high-quality initial solutions for the mixed-integer quadratic programming model and the column generation algorithm. In particular, we find that using the proposed heuristic algorithm as an initial solution can significantly boost the performance of the commercial solver to solve the mixed-integer quadratic programming model. The computational results show that the proposed solution methods are capable of efficiently solving instances of various sizes.

The remainder of this paper is structured as follows. The theoretical background is presented in Section 2. Section 3 presents the problem description and model formulation, and the column-generation algorithm is described in Section 4. We apply the model and algorithms to identify coordination communities using U.S. mobility data in Section 5 and present a case study in Section 6. Finally, Section 7 summarizes our findings and provides future research directions.

2. Theoretical background

In the search for optimal pandemic management, it has become clear that policy coordination across agencies and levels of government is necessary (Holtz et al., 2020). From the infectious disease epidemiological viewpoint, existing geopolitical or administrative borders are entirely arbitrary, as viruses can easily spread across cities, counties, states, and countries. Using commuting data, Laroze, Neumayer and Plümper (2021) show the spread of COVID-19 is spatially dependent across local districts in England between March and June of 2020. Fajgelbaum, Khandelwal, Kim and Schaal (2020) argue that spatially differentiated lockdowns can be substantially better economically compared to blanket lockdown orders. Furthermore, studies have shown that, because of mismatches between administrative structure and optimal governance, existing jurisdictional boundaries can act almost as a barrier to key elements of effective pandemic management (e.g., Ki, Kwak & Song, 2020), and Trein, Biesbroek, Bolognesi and Meyer (2021) suggest that policy coordination and integration should consider such specific context. Examining the patterns of population movement, effectively captured by mobile phone data, can reveal coordination zones for optimized pandemic policy (Glaeser, Gorback & Redding, 2020; Holtz et al., 2020).

The problem of identifying optimal coordination zones based on the natural movement of people is grounded on two independent but connected streams of research: community detection in complex networks and districting problems. Table 1 summarizes the relevant prior studies and highlights the key themes and characteristics of these two streams.

Community detection aims to identify sub-networks or local communities based on relationships/interactions between the nodes such that nodes within the same community are more connected among themselves than with nodes in other communities (Newman, 2006; Newman & Girvan, 2004). Identifying local communities reveals how a complex network is organized and can help decision-makers to focus on the sub-regions (Fortunato & Hric, 2016), because sub-networks often have different properties at the local level than at the level of the entire network (Newman & Girvan, 2004). In practice, the community detection problem and its extensions have been used in different fields, including detecting partitions of the human brain (Ashourvan, Telesford, Verstynen and Bassett, 2019), and maximizing the spread of influence in social networks in viral marketing (Huang, Shen, Meng & He, 2019).

Despite its wide applications, a precise or universally accepted definition of what creates a community is not available, and as a result, a variety of methods for detecting and measuring the quality of communities has been proposed, tested, and implemented (Fortunato & Hric, 2016; Rosvall, Delvenne, Schaub & Lambiotte, 2019). The most popular methodologies for solving community detection problems are optimization-based models, in which communities are identified based on maximizing a quality function (Lancichinetti & Fortunato, 2009; Fortunato & Hric, 2016). The modularity function, developed by Newman and Girvan (2004), evaluates weights of links between nodes inside the same communities relative to weights of links between nodes in the separate communities and is among the most commonly used in such optimization-based models for community detection. Because modularity maximization is hard (Brandes, Delling, Gaertler & Wagner, 2006), several algorithms have been proposed to efficiently detect communities in a large network based on modularity optimization (Blondel, Guillaume, Lambiotte & Lefebvre, 2008; Clauset, Newman & Moore, 2004; Traag, Waltman & van Eck, 2019). Recently, modularity density is introduced to remedy modularity optimization's limitation of resolution limit in very large networks by normalizing the weights considering the number of nodes in

Table 1
Characteristics of relevant studies on community detection and districting problems.

References	Problem	Modularity function	Contiguity	Interactions between nodes	Solution approach	Application
Aloise et al. (2010)	Community detection	√	×	√	Column-generation algorithm	Generic
Ashourvan et al. (2019)	Community detection	×	×	√	Heuristic	Partitioning human brain
Bergey, Ragsdale and Hoskote (2003a), 2003b)	Districting	×	√	×	Heuristic	Electrical power districting
Blondel et al. (2008)	Community detection	√	×	√	Heuristic (Louvain algorithm)	Generic problem
Bozkaya et al. (2003)	Districting	×	√	×	Heuristic	Political districting
Camacho-Collados, Liberatore and Angulo (2015)	Districting	×	√	×	Heuristic	Police districting
Caro et al. (2004)	Districting	×	√	×	Commercial solver	School districting
Carvajal et al. (2013)	Districting	×	√	×	Branch-and-cut algorithm	Forest Planning
Clauset et al. (2004)	Community detection	√	×	√	Heuristic	Generic
Costa (2015)	Community detection	√*	×	√	Commercial solver	Generic
Dugošija, Savić and Maksimović (2020)	Districting	×	√	×	Commercial solver	Political districting
Farughi, Tavana, Mostafayi and Santos Arteaga (2020)	Districting	×	√	×	Heuristic	Healthcare districts
Fryer Jr & Holden (2011)	Districting	×	×	×	Heuristic	Political districting
Han et al. (2020)	Districting	×	√	×	Commercial solver	Machinery maintenance
Haase and Müller (2014)	Districting	×	√	×	Column-generation algorithm	Salesforce deployment
Hess et al. (1965)	Districting	×	×	×	Heuristic	Political districting
Hojati (1996)	Districting	×	√	×	Lagrangian relaxation	Political districting
Huang et al. (2019)	Community detection	×	×	√	Heuristic	Viral marketing
Kim (2018)	Districting	×	√	×	Heuristic	Political districting
Latapy and Pons (2004)	Community detection	×	×	√	Heuristic (Walktrap algorithm)	Generic
Li et al. (2008)	Community detection	√*	×	√	Commercial solver	Generic
Newman and Girvan (2004)	Community detection	√	×	√	Heuristic	Generic
Newman (2006)	Community detection	√	×	√	Heuristic	Generic
Önal et al. (2016)	Districting	×	√	×	Commercial solver	Conservation management
Ricca and Simeone (2008)	Districting	×	√	×	Heuristic	Political districting
Ruktanonchai et al. (2020)	Community detection	√	×	√	Heuristic (Walktrap algorithm)	Pandemic response
Santiago and Lamb (2017)	Community detection	√*	×	√	Heuristic	Generic
Sato and Izunaga (2019)	Community detection	√*	×	√	Branch-and-price algorithm	Generic
Shirabe (2005)	Districting	×	√	×	Commercial solver	Generic
Shirabe (2009)	Districting	×	√	×	Commercial solver	Generic
Shirazi, Albadvi, Akhondzadeh and Teimourpour (2020)	Community detection	√	×	√	Heuristic (Louvain algorithm)	Physicians' specialty
Traag et al. (2019)	Community detection	√	×	√	Heuristic (Leiden algorithm)	Generic
Wang, Zheng, Qian and Liang (2017)	Community detection	×	×	√	Heuristic	Protein complexes
Zhang, Liu, Li and Wen (2020)	Community detection	√	×	√	Heuristic	Drug discovery
This study	Community detection	√	√	√	Column-generation algorithm	Pandemic response

* Modularity density function is considered as the objective function instead of the original modularity function.

each community (Costa, 2015; Li, Zhang, Wang & Chen, 2008; Santiago & Lamb, 2017).

Community detection based on modularity maximization has been studied extensively in the literature (Aloise, Cafieri, Caporossi & Liberti, 2010; Traag et al., 2019). However, as highlighted in Table 1, such algorithms do not guarantee the physical contiguity of nodes in proposed communities, an important factor for policy coordination for pandemic management. As an example, Ruktanonchai et al. (2020) applied the Walktrap algorithm, a heuristic developed by Latapy and Pons (2004), to detect communities in Europe based on the movement among subdivisions of countries (NUTS3, or Nomenclature of Territorial Units for Statistics). One of the identified subcommunity based on mobility patterns includes Estonia, Iceland, two areas of Italy, and part of Greece, while another subcommunity includes Slovenia, Sweden, and part of Croatia. Such mobility-based communities are scattered in different regions and countries, making the coordination of pandemic policies impractical, if not impossible.

Ignoring contiguity of communities in the community detection problem may result in a community structure with not only the inclusion of nodes that are far apart but, perhaps more important, also the exclusion of nodes that have a slightly lower degree of connection but are in the same region. In reality, NPI policy coordination among connected jurisdictions with a high level of movement is useful in combating pandemics, while long-distance movements among jurisdictions, even frequently, require different policies such as quarantine. For example, in response to the COVID-19 surge in June 2020, three contiguous states—New York, New Jersey, and Connecticut—coordinated their nonpharmaceutical interventions while jointly implemented 14-day quarantine require-

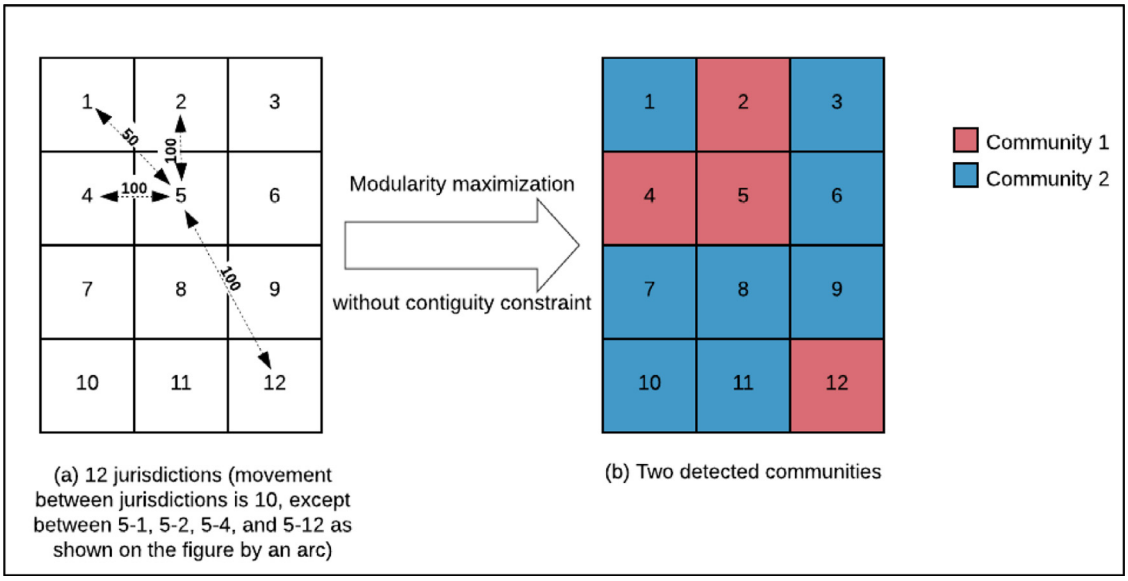


Fig. 1. A community detection example without considering contiguity constraint.

ments for visitors coming to the tri-state region from states with high COVID-19 infection rate (CBS New York, 2020). To further illustrate this issue, consider the fictitious case of 12 jurisdictions presented in Fig. 1a. Assume that the movement between each pair of jurisdictions is 10, except for the following: movement between (5 and 2), (5 and 4), and (5 and 12) is 100, and movement between (5 and 1) is 50. Applying the modularity maximization algorithm to find zones of high level of intra-movement, we get two communities, as shown in Fig. 1b. Although the movements and geography imply that jurisdictions 1, 2, 4, and 5 are inherently a community, the simple modularity maximization assigns the disjointed jurisdiction 12 to the community with 2, 4, and 5 because of its strong connection to jurisdiction 5, while assigning jurisdiction 1 to a different community. This example shows that modularity maximization does not necessarily produce contiguous community structure, which is important to NPI policy coordination.

The issue of contiguity is treated extensively in the districting problem, which refers to joining small areas (or units) to form larger areas (or districts) based on specific criteria. There is a wide variety of applications for the districting problem in the literature, such as political districting (Bozkaya, Erkut & Laporte, 2003; Garfinkel & Nemhauser, 1970; Hojati, 1996), school redistricting (Bulka, Carr, Jordan & Rheingans, 2007; Caro, Shirabe, Guignard & Weintraub, 2004), designing compact and contiguous conservation reserves (Önal, Wang, Dissanayake & Westervelt, 2016), and districting of service regions for agricultural machinery maintenance (Han, Hu, Mao & Wan, 2020). As each specific application of the districting problem is unique with its own objectives, several criteria and subsequently mathematical models and solution algorithms have been developed. The political districting problem, which is the original application of the districting problem, aims to divide a given territory into a number of districts, where a number of seats are assigned to each district usually based on its population (Ricca & Simeone, 2008; Ricca, Scozzari & Simeone, 2013). Several criteria, such as compactness, which refers to how odd the shape of a district is, have been proposed to achieve fairness and avoid gerrymandering in political districting (Ricca & Simeone, 2008; Shirabe, 2009). In their seminal paper, Hess, Weaver, Siegfeldt and Zitlau (1965), for example, use the sum of squared distances from each person to the center of his/her assigned district as a measure of compactness. They modeled the political districting problem as a location-allocation problem without considering the contiguity of districts and proposed an iterative heuristic procedure for solving the model. Other studies define compactness in very different ways, such as the distance between persons within the same district compared to the minimum possible distance (Fryer Jr & Holden, 2011) and the closeness between selected areas or units (Kim, 2018).

Districting problems often require solutions with geographical contiguity. A district is contiguous when one can travel between any two points in the district without going outside of it. The geographical contiguity constraint adds to the complexity of the districting problem and requires additional treatments to simple location-allocation. Shirabe (2005 and 2009), for the first time, use the network flow technique to mathematically formulate the contiguity constraint, and this technique has also been used in other applications such as sales team deployment (Haase & Müller, 2014) and service maintenance networks (Han et al., 2020). More recently, Carvajal, Constantino, Goycoolea and Weintraub (2013) deploy the notion of node-cut set to mathematically formulate the contiguity constraint in a harvest scheduling problem with the aim of maximizing profits. However, as shown in Table 1, these techniques are designed strictly for solving districting problems, and, to our knowledge, no models have been developed to identify contiguous communities based on complex relationships between vertices in a network. The current study attempts to address this research gap and applies the community detection model based on modularity maximization to the movement of people, subject to the geographic contiguity and cardinality (i.e., size of the districts) constraints in the districting problems. As such, our study contributes to the body of knowledge by extending the existing studies from two independent streams of research to solve a relevant problem of community detection with contiguity constraint that requires theories from both.

In addition, both community detection and districting problems are difficult problems to solve (Blondel et al., 2008; Haase & Müller, 2014; Santiago & Lamb, 2017). In this study, we present a column-generation algorithm that extends previous exact methods (e.g., Aloise et al., 2010, and Sato & Izunaga, 2019) by incorporating contiguity and cardinality constraints while maximizing modularity, coupled with an iterative pricing procedure, to improve the computational performance of the algorithm. We also propose a fast heuristic algorithm to find high-quality solutions to initiate the column-generation procedure. In so doing, our study contributes to

the development of computational methods for related problems, as the results show that the solution methods can efficiently solve instances of various sizes.

3. Problem description and formulation

We consider a *geographic region*, such as a country, that is divided into a number of mutually exclusive and collectively exhaustive *spatial units* (SUs), such as zip codes or counties. People frequently move within and between these SUs for various purposes, such as work, shop, leisure, etc. To reduce the spread of disease during a pandemic, the SUs are responsible for applying targeted NPIs only within their borders. However, as disease spread does not follow the geographic boundaries, the SUs may be better off coordinate their efforts with other close-by SUs. The goal here is to identify *communities* of highly interconnected SUs within the geographic region boundaries based on the natural movement of people. To do so, our objective is to maximize network modularity (Newman & Girvan, 2004), which is a function of people's mobility between SUs. In order to facilitate policy coordination, the communities should be contiguous, and the number of SUs that constitute a community can be constrained. A community is contiguous when one can travel between any two points within the community without going outside of it. We call this problem the modularity maximization with contiguity constraints (MCC).

To formally define the MCC,[1] let us represent a geographic region by a weighted connected undirected graph $G = (V, E)$, where V and E respectively denote the set of vertices and edges. A vertex $i \in V$ represents a SU. Then, the modularity of the geographic region, denoted by M, is calculated using Eq. (1):

$$M = \frac{1}{2m} \sum_{i,j \in V} \left[w_{ij} - \frac{k_i k_j}{2m} \right] \delta(d_i, d_j) \tag{1}$$

where:

$$m = \frac{1}{2} \sum_{i,j \in V} w_{ij} \tag{2}$$

$$k_i = \sum_{j \in V} w_{ij} \tag{3}$$

In Eq. (1), w_{ij} represents the weight of the edge between vertices $i \in V$ and $j \in V$(i.e., the mobility between i and j), d_i denotes the community to which vertex (or SU) $i \in V$ is assigned, and $\delta(d_i, d_j)$ equals 1 if $d_i = d_j$ and 0 otherwise. Eqs. (2) and (3) calculate auxiliary parameters m and k_i, where m is the total weights of the undirected graph, and k_i is the sum of the weights of the edges attached to vertex i.

We define and formulate contiguity using the flow analogy presented by Shirabe (2005). Contiguity can be defined in terms of a graph where each SU is a vertex and edges connecting each pair of SUs represent adjacency. Using this view, contiguity is equivalent to the notion of connectedness in graph theory. Therefore, one can check contiguity (connectedness) of a set of SUs (or a sub-network), S, by verifying the following condition: Starting from an arbitrary SU (vertex) in S, one can reach every other SU in S by following a sequence of adjacency edges. To check the contiguity condition, Shirabe proposed finding paths between every SU and one specific SU (named sink) in S, which is similar to the movement of fluid from multiple sources to a single sink in a connected network.

[1] We present a summarized list of notations in Appendix A within Supplementary Materials.

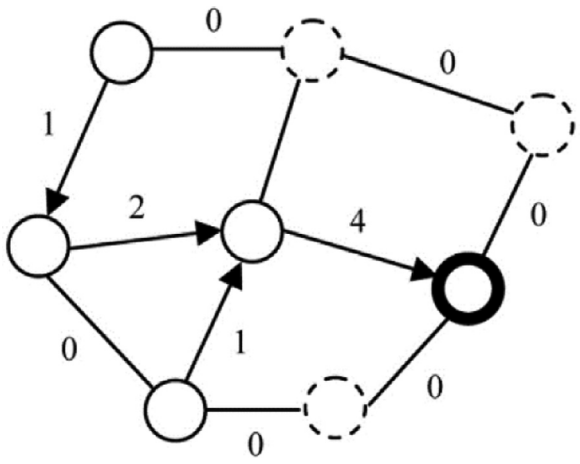


Fig. 2. A flow representation of a contiguous district (Shirabe, 2009).

In order to apply this analogy in this study, a community is interpreted as a sub-network of the SUs assigned to that community, in which one SU is a sink and every other SU provides one unit of supply. For a community to be contiguous, the supply sent from each SU must ultimately arrive at the sink by only passing through the SUs within the sub-network. Fig. 2 illustrates the idea. The solid circles and dashed circles represent the assigned and non-assigned SUs to the community, respectively. The bold circle is designated as the sink, and the numbers on the arrows represent the flow. In this way of formulation, how each unit of supply travels in the sub-network is not important. Instead, we are concerned whether each supply unit can ultimately reach the sink at least in one way. It is intuitively apparent that a disconnected sub-network requires more than one sink to consume all supply units, therefore violating the contiguity condition.

To mathematically formulate the contiguity based on the flow analogy, let binary decision variable x_{id} equal 1 iff SU $i \in V$ is assigned to community $d \in D$, continuous decision variable y_{ijd} determine the flow from SU $i \in V$ to SU $j \in V$ within community $d \in D$, and binary decision variable q_{id} equal 1 iff SU $i \in V$ is designated as the sink for community $d \in D$. Let A_i denote the set of immediate connected vertices to vertex $i \in V$, i.e., $A_i = \{j \in V | \{i, j\} \in E\}$. Then, the contiguity constraint is expressed as a set of linear equations as follows:

$$\sum_{i \in V} q_{id} \leq 1, \quad \forall d \in D, \tag{4}$$

$$\sum_{i \in V} x_{id} \leq C \sum_{i \in V} q_{id}, \quad \forall d \in D, \tag{5}$$

$$\sum_{j \in A_i} y_{ijd} - \sum_{j \in A_i} y_{jid} \geq x_{id} - C q_{id}, \quad \forall i \in V, \quad \forall d \in D, \tag{6}$$

$$\sum_{j \in A_i} y_{jid} \leq (C-1) x_{id}, \quad \forall i \in V, \quad \forall d \in D, \tag{7}$$

$$x_{id}, q_{id} \in \{0, 1\}, \quad \forall d \in D, \quad \forall i \in V, \tag{8}$$

$$y_{jid} \geq 0, \quad \forall d \in D, \quad \forall i \in V, \quad j \in A_i . \tag{9}$$

Constraints (4) ensure that at most one sink is assigned to each community. Constraints (5) ensure that the cardinality of selected SUs in a community is less than or equal to C, the maximum allowable number of SUs to constitute a community, and that the sink of a community is selected from the pool of SUs assigned to that community (one SU should be sink in each community unless no SU can be assigned to that community). Constraints (6)

indicate that the net out-flow from an SU assigned to a community is positive unless that SU is designated to be the sink of that community. The two terms on the left-hand side of this constraint calculate the total outflow and total inflow of SU i, respectively. Accordingly, if SU i is assigned to community d and not a sink (i.e., $x_{id} = 1$ and $q_{id} = 0$), then SU i *must* have a positive supply (positive net outflow), which is the primary purpose of the constraint. If SU i is assigned to community d and it is a sink (i.e., $x_{id} = 1$ and $q_{id} = 1$), then SU i can have a positive demand (negative net outflow), with a maximum amount of C-1. Constraints (7) ensure that there is no inflow into any SU that does not belong to that community (where $x_{id} = 0$, then $y_{jid} = 0$), and that the total inflow to the SUs assigned to each community does not exceed C-1. Since the net outflow from an SU assigned to a community is positive (unless that SU is designated to be the sink of that community), the constraints ensure that supply from each SU in the community reach the sink, and therefore the contiguity condition of each community holds. Finally, constraints (8) and (9) show the type of decision variables.

The objective of the MCC is to find the set of communities D, where any community $d \in D$ is a collection of contiguous vertices (SUs). The MCC is formulated as a standard mixed-integer quadratic-programming model (*MIQP*):

$$Maximize\ M = \frac{1}{2m} \sum_{d \in D} \sum_{i,j \in V} \left[w_{ij} - \frac{k_i k_j}{2m} \right] x_{id}\ x_{jd} \qquad \text{(MIQP-1)}$$

$$subject\ to: \sum_{d \in D} x_{id} = 1,\ \ \forall i \in V, \qquad \text{(MIQP-2)}$$

and constraints (4–9).

Objective function (MIQP-1) maximizes total modularity. The quadratic terms $x_{id}x_{jd}$ in the objective function can be easily linearized through standard McCormick transformations (McCormick, 1976). Constraints (MIQP-2) assign each SU to exactly one community.

4. Solution methods

Commercial solvers such as GUROBI[2] and CPLEX[3] are available to solve *MIQP*. In addition, we present a heuristic and a column-generation algorithm to solve the MCC.

4.1. The heuristic algorithm

The heuristic algorithm is designed in three phases: new community creation, community expansion, and a local search for solution improvement. In the first phase of the algorithm, a new community is created by randomly choosing a SU that does not belong to existing communities. The selected SU is assigned to the new community as the first member. The size of the new community is determined by randomly selecting a number between 2 and C using a discrete uniform distribution.

The second phase expands the community created in the first phase by selecting a SU from the subset of SUs that are not assigned to any communities and are adjacent to at least one SU within the community. If the subset of adjacent SUs without any communities has more than one member, the SU with the strongest connection to the community, i.e., with the highest sum of mobility with community members is selected. Note that the set of adjacent SUs is considered explicitly in the algorithm to preserve the contiguity of each community. This set of adjacent SUs is updated when a new SU is added to the community. Expansion of the community continues while the set of adjacent SUs is not empty, and the size of the community is less than the size determined in the first phase. The process of creating new communities (phase one) and expanding them (phase two) is repeated until all SUs are assigned to a community. Then, the modularity of constructed communities is calculated based on Eq. (1).

After creating an initial solution using the first two phases, a local search is applied to improve the quality of the solution. In this third phase, for each SU i, a new modularity value is calculated for joining i to SU j if the following three conditions are satisfied: (1) the size of the community that j belongs to is less than C, (2) i is adjacent to the community of SU j, and (3) removing i from its current community does not affect contiguity of its original community. If the new modularity value is greater than the current value, SU i is placed to the community of SU j, and the modularity value is updated. This process is repeated for all SUs over and over until no improvement can be achieved.

Next, the algorithm repeats this process (phases one to three) for a fixed number of times (*maxIterations*) and returns the solution with the highest modularity value. The pseudo-code of the heuristic algorithm is presented in Appendix B.

4.2. The column-generation algorithm

We present a column-generation algorithm that is tailored to solve an exponentially sized reformulation of *MIQP*. Let Ω represent the set of all possible *configurations*. A *configuration* $c \in \Omega$ is a tuple $\langle V_c,\ \theta_c \rangle$, where V_c denotes a subset of contiguous SUs ($V_c \subseteq V$) and $|V_c| \leq C$. The second element, θ_c, is the modularity value of configuration $c \in \Omega$, i.e.:

$$\theta_c = \frac{1}{2m} \sum_{i,j \in V_c} \left[w_{ij} - \frac{k_i k_j}{2m} \right]$$

We denote the set of configurations that include SU $i \in V$ by Ω_i. Let binary decision variable χ_c equal 1 iff configuration $c \in \Omega$ is selected. The MCC can alternatively be formulated as the following exponentially sized model (henceforth referred to as *EM*):

$$Maximize\ M = \sum_{c \in \Omega} \theta_c\ \chi_c \qquad \text{(EM-1)}$$

subject to:

$$\sum_{c \in \Omega_i} \chi_c = 1,\ \ \forall i \in V, \qquad \text{(EM-2)}$$

$$\chi_c\ \in \{0, 1\},\ \ c \in \Omega\ . \qquad \text{(EM-3)}$$

Objective function (EM-1) maximizes the total modularity for the whole geographic region. Constraints (EM-2) ensure that each SU is assigned to exactly one community. Each selected configuration in the solution of *EM* indicates the set of SUs that constitute a community. It is straightforward to show that *MIQP* and *EM* are equivalent. Since $|\Omega|$ exponentially grows with the number of SUs in an instance, enumerating all possible configurations is impractical. Hence, we dynamically generate columns (configurations) through column-generation (Gilmore & Gomory, 1961).

A column-generation algorithm is implemented on the linear programming (LP) relaxation of the exponentially sized model that includes a subset of the columns, referred as the *restricted master problem* (RMP). Then new columns with positive reduced costs are dynamically generated using a *pricing problem* (in the case of maximization). The algorithm iterates between the pricing problem and the RMP and terminates if no columns with positive reduced costs can be found. It is shown that the LP relaxation of RMP that is solved through column generation and LP relaxation of the master problem (the exponential model with all possible columns) are

[2] https://www.gurobi.com/.
[3] https://www.ibm.com/products/ilog-cplex-optimization-studio.

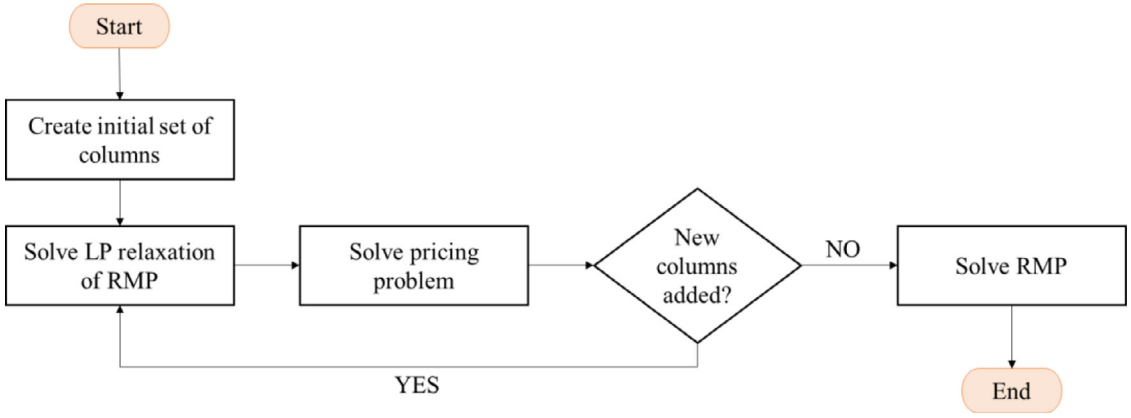


Fig. 3. The column-generation algorithm.

equal. In the case of integer-programming models with many variables, a branch-and-price algorithm (Barnhart, Johnson, Nemhauser & Vance, 1998) can be used where a column generation is solved on each node of the branch-and-bound algorithm (Lawler & Wood, 1966). The column-generation algorithm used to solve our problem is summarized in Fig. 3.

For the MCC, the RMP is similar to *EM*, except it is solved over a subset of configurations ($\tilde{\Omega} \subseteq \Omega$). Here we only solve the LP relaxation of the RMP on the root node of the branch-and-bound tree using the column-generation algorithm. When no additional columns can be added, we solve RMP to get an integer solution. As discussed in Section 5, the computational experiments, in many cases, we are able to prove optimality without branching. For generating the feasible initial set of columns, we use the heuristic algorithm. We also run *MIQP* for a few seconds to seek additional improvements in the initial feasible solution and upper bounds before starting the column-generation procedure.

To find improving columns for the LP relaxation of the RMP, we use a pricing problem. Let γ_i be the dual values associated with Constraints (EM-2) in the RMP. The pricing problem is similar to *MIQP*, and it is solved to find a contiguous configuration. Let binary decision variable x_i equal 1 iff spatial unit $i \in V$ is selected, continuous decision variable y_{ij} determine the flow from SU $i \in V$ to SU $j \in V$, and binary decision variable q_i equals 1 iff SU $i \in V$ is designated as the sink. The pricing problem (**P**) is:

$$\textit{Maximize} \ \frac{1}{2m}\sum_{i,j\in V}\left[w_{ij}-\frac{k_i k_j}{2m}\right]x_i\,x_j-\sum_{i\in V}\gamma_i\,x_i \tag{P-1}$$

subject to:

$$q_i \le x_i, \ \forall i \in V, \tag{P-2}$$

$$\sum_{i\in V} q_i \le 1 \tag{P-3}$$

$$\sum_{i\in V} x_i \le C\sum_{i\in V} q_i, \tag{P-4}$$

$$\sum_{j\in A_i} y_{ij} - \sum_{j\in A_i} y_{ji} \ge x_i - Cq_i, \ \forall i \in V, \tag{P-5}$$

$$\sum_{j\in A_i} y_{ji} \le (C-1)x_i, \ \forall i \in V, \tag{P-6}$$

$$x_i, q_i \in \{0,1\}, \ \forall i \in V, \tag{P-7}$$

$$y_{ji} \ge 0, \ \forall\, i \in V, \ j \in A_i\ . \tag{P-8}$$

The objective function (P-1) maximizes reduced cost for the newly generated column. Similar to the constraints in *MIQP*, Constraints (P-2) to (P-8) enforce contiguity of the configuration generated by the pricing problem and ensure that the cardinality of the generated configuration is at most C (similar to Eqs. (4) to (9)). Our computational experiments show that commercial solvers can optimally solve the pricing problem quickly.

To enhance the performance of the column-generation algorithm, instead of adding a single new column (*newColumn*) in every iteration of the algorithm, we use an iterative pricing procedure (IPP) to generate multiple compatible columns, which are those that can simultaneously be selected by RMP. This procedure is summarized in the IPP Algorithm below.

IPP Algorithm

Set $V' = V$;
Set *newColumn* = FALSE;
While $|V'| > 0$:
  Solve P over SUs in V';
  If $v^*(\mathbf{P}) > 0$:
    Add the set of selected SUs ($\overline{X^*}$) to $\tilde{\Delta}$; $V' \leftarrow V' \backslash \overline{X^*}$; *newColumn* = TRUE;
  If *newColumn* == TRUE and $|V'| > 0$:
    Solve P over SUs in V';
    If $v^*(\mathbf{P}) > 0$:
      Add the set of selected SUs ($\overline{X^*}$) to $\tilde{\Delta}$; and $V' \leftarrow V' \backslash \overline{X^*}$;
  Else:
    Break;
  If *newColumn* == FALSE:
Break;

In the IPP Algorithm, $v^*(\mathbf{P})$ denotes the optimal objective value of the pricing problem. The IPP Algorithm indicates that in each iteration of column generation procedure, after finding the column with the most positive reduced cost, we remove the SUs that are selected by the pricing problem and resolve the pricing problem. This procedure continues until no SUs are left or no new column with positive reduced cost is found. Through this process, we ensure not to add the columns that are already added to set $\tilde{\Delta}$. The IPP Algorithm helps to find high-quality feasible integer solutions more quickly by adding multiple compatible columns with positive reduced costs that may be simultaneously selected through the restricted master problem. In the next section, we highlight the advantages of coupling the IPP Algorithm with the column-generation algorithm for this problem.

5. Computational experiments

To examine the efficiency of the proposed algorithms, we conduct computational experiments of real mobility data (from Camber System, https://cambersystems.com) in the U.S. We also use

a set of synthetic instances as a robustness check for the performance of our algorithms.

States consisting of 5 to 254 counties (as SUs) are included in these experiments, with each state representing an instance. Every instance contains mobility information between all county pairs and an adjacency matrix to specify the neighbors for each county (A_i). To eliminate the possibility that counties always having higher mobility with adjacent counties compared to remote, non-contiguous counties and thus make the introduction of contiguity and cardinality constraints more challenging while maximizing modularity, we create synthetic instances based on randomly generated mobility data between counties (by random shuffling of the mobility matrix). Overall, in our experiments, we use 36 real instances and 36 synthetic instances.[4] We also perform sensitivity analyses by changing parameter C, the maximum number of counties that constitute a community. We consider $C = \{5, 10, 15\}$. For both real and synthetic instance sets, we label an instance as *small* if the number of counties ($|V|$) in its corresponding state is less than 70 ($|V| < 70$), and we label it as *large* if $|V| \geq 70$. For both synthetic and real instance sets, we have 20 and 16 *small* and *large* instances, respectively.

All the experiments are performed on a standard personal computer.[5] We use GUROBI 9.1.1 as the standard mixed-integer (quadratic) programming solver to directly solve the optimization model and as the oracle used within the column-generation algorithms. We test the following five algorithms:

1. *H*: the heuristic algorithm described in Section 4.1,
2. *MIP*: directly solve *MIQP* with GUROBI,
3. *MIP+*: directly solve *MIQP* with GUROBI with the solution gained from the heuristic algorithm as an initial solution,
4. *CG1*: column-generation algorithm by solving the pricing problem only once in each iteration, and,
5. *CG2*: column-generation algorithm that is coupled with the IPP Algorithm.

We also should note that, since the coefficients in the objective functions (MIQP-1) and (P-1) are very small in our datasets, to avoid numerical issues that may arise as a result of commercial solvers optimality tolerance gap, we eliminate the constant term $\frac{1}{2m}$ from the objective function and multiply it back to the objective value after the model is solved. In comparing the performance of algorithms, we set a time limit of 60 min for solving each instance by *MIP*, *MIP+*, *CG1*, and *CG2*. The value of *maxIterations* parameter in the heuristic algorithm is set to 20, selected based on the trial-and-error approach. The solution time for the heuristic algorithm *H* is negligible (less than 30 s).

5.1. Solution quality

Tables 2 and 3 summarize the average of best solutions found by each algorithm.

As can be seen in Table 2, for small instances, MIP, MIP+, CG1 and CG2 show similar performance in finding feasible solutions. However, for large instances, *CG2* outperforms the other algorithms with *MIP* being the worst. The heuristic algorithm, *H,* finds solutions that are on average 3.9% lower than our best performing algorithm, *CG2*. Particularly, we see that for large instances, *H* is able to find better solutions than *MIP*. We also find significant improvements in the solutions found by *MIP+* compared to *MIP*. Specifically, for large instances, we observe that MIP+ is able to find solutions with 7.8% higher modularity, highlighting the benefits of generating initial solutions using *H* for GUROBI to solve *MIQP*. Note that modularity values closer to 1.0 indicate a strong community structure. In practice, modularity is often in the 0.3 – 0.7 range for most networks (Newman & Girvan, 2004).

Table 2
Average modularity for real instances by each algorithm.

		Algorithm				
$\|V\|$	C	*H*	*MIP*	*MIP+*	*CG1*	*CG2*
< 70	5	0.397	0.411	0.411	0.411	0.411
	10	0.426	0.441	0.441	0.440	0.441
	15	0.422	0.443	0.443	0.443	0.443
≥ 70	5	0.503	0.495	0.530	0.535	0.535
	10	0.589	0.551	0.599	0.606	0.608
	15	0.602	0.571	0.615	0.608	0.620

Table 3
Average modularity over synthetic instances by each algorithm.

		Algorithm				
$\|V\|$	C	*H*	*MIP*	*MIP+*	*CG1*	*CG2*
< 70	5	0.400	0.421	0.425	0.434	0.434
	10	0.436	0.444	0.454	0.465	0.469
	15	0.407	0.453	0.460	0.462	0.473
≥ 70	5	0.513	0.505	0.565	0.587	0.589
	10	0.604	0.524	0.639	0.640	0.669
	15	0.627	0.565	0.657	0.629	0.677

Similar to real data, for the synthetic instances (Table 3), we find that *CG2* outperforms other algorithms. Specifically, for large instances we observe than *CG2* is able to find solutions with 17.6%, 3.8%, and 4.1% higher modularity compared to *MIP*, *MIP+*, and *CG1*, respectively. We also observe that for large instances, *H* outperforms *MIP* with respect to the solution quality (e.g., 0.627 vs 0.565 with $C = 15$.) We also see that *MIP+* finds solutions with 9% higher modularity compared to *MIP*, highlighting the benefits of the proposed heuristic algorithm.

5.2. Computational performance of the algorithms

We start this subsection by comparing the performance of CG1 and CG2 to analyze the impact of the proposed iterative pricing procedure (IPP). Table 4 summarizes these results. In Table 4, *#solved* indicates the number of instances for which the column-generation algorithm terminates and reports valid bounds within the 1-hour time limit. Column $time_s$ represents the runtimes (in seconds) averaged over only those instances that the column-generation procedure terminates within the time limit. Column *#opt* indicates the number of instances that are proved to be optimal. Column *M* represents average modularity as defined by Eq. (1).

As we can see in Table 4, for both datasets and by setting different values of *C*, *CG2* outperforms *CG1*, highlighting that utilizing the IPP boosts the performance of the column-generation algorithm. For real instances with $C = 5$, both *CG1* and *CG2* can solve 35 out of 36 instances within 60 min with CG2 performing better in runtimes (averaged 201 s vs. 324 s). By increasing *C*, the performance of both algorithms degrades since a larger *C* is creating a larger pool of columns with positive reduced costs that are dynamically generated through the pricing problem. Our results indicate that with $C = 15$, *CG2* is still able to solve 24 out of 36 instances

[4] Due to contractual agreements with the data provider, we are not able share the real and synthetic instances (that are generated by shuffling movements of the real instances). For replication purposes, we generated a random set of synthetic instances that are available at https://github.com/mohsen-emadikhiav/MCC-Instances.git. The data generation procedure and summary of the results for these second set of synthetic instances are presented in Appendix C. The codes are available upon request.

[5] Intel(R) Core (TM) i7-9700 CPU @ 3.60GHz with 32.0 GB RAM and using 1 thread.

Table 4
Summary of the results of CG1 and CG2 for real and synthetic instances.

		CG1				*CG2*			
Data set	*C*	*#solved*	$time^*_s$	*#opt*	*M*	*#solved*	$time^*_s$	*#opt*	*M*
Real	5	35	324	23	0.466	35	201	23	0.466
Instances	10	25	687	21	0.514	34	816	27	0.515
	15	19	623	17	0.516	24	657	22	0.522
Synthetic	5	33	840	17	0.502	34	552	17	0.503
Instances	10	14	353	13	0.543	17	444	13	0.558
	15	13	130	12	0.536	15	475	12	0.564

* Averaged over only those instances that the column-generation procedure terminates within time limit (in seconds).

Table 5
Summary of computational results for each algorithm on real instances.

		MIP			*MIP+*			*CG1*			*CG2*		
$\|V\|$	*C*	*#opt*	*time**	*gap*†	*#opt*	*time**	*gap*†	*#opt*	*time**	*gap*†	*#opt*	*time**	*gap*†
<	5	16	166	7%	16	214	8%	16	34	0.22%	16	30	0.22%
70	10	18	450	4%	18	344	8%	17	181	0.11%	17	86	0.05%
	15	19	414	9%	19	291	6%	17	398	10%	18	226	0.02%
≥	5	0	–	43%	0	–	27%	7	771	5%	7	450	5%
70	10	0	–	40%	0	–	17%	4	2412	27%	10	1714	9%
	15	1	3205	36%	0	–	15%	0	–	33%	4	2298	31%

* Averaged over only those instances that are proved to be solved optimally (in seconds).
† Averaged over only those instances that are not solved optimally (in percentage).

Table 6
Summary of computational results for each algorithm on synthetic instances.

		MIP			*MIP+*			*CG1*			*CG2*		
$\|V\|$	*C*	*#opt*	*time**	*gap*†	*#opt*	*time**	*gap*†	*#opt*	*time**	*gap*†	*#opt*	*time**	*gap*†
<	5	12	33	49%	12	59	43%	14	76	1%	14	55	2%
70	10	12	15	45%	12	21	38%	13	128	48%	13	59	26%
	15	12	12	40%	12	20	33%	12	26	52%	12	17	39%
≥	5	0	–	91%	0	–	49%	3	1524	14%	3	898	9%
70	10	0	–	158%	0	–	34%	0	–	42%	0	–	35%
	15	0	–	191%	0	–	30%	0	–	44%	0	–	29%

* Averaged over only those instances that are proved to be solved optimally (in seconds).
† Averaged over only those instances that are not solved optimally (in percentage).

(22 of which are proved to be optimal) while *CG1* solves 19 out of 36 instances.

The performance of both algorithms diminishes on synthetic instances. This is expected as by randomization of mobility between counties, finding contiguous communities with high modularity becomes more challenging. With $C = 5$, *CG2* is still able to solve 34 out of 36 instances (vs. 33 with *CG1*), but when *C* is increased to 15, only 15 out of 36 instances are solved within the time limit (vs. 13 with *CG1*). The average run times, on the other hand, are slightly worse for *CG2* in synthetic instances when *C* is 10 or 15, due to the long run time of the instance solved by *CG2* that *CG1* could not solve.

We also find that *CG2* dominates *CG1* in finding feasible solutions with higher modularity for all instance groups. *CG1* struggles to find better feasible solutions with increasing *C*. For instance, *CG1* finds solutions with a lower modularity value as *C* is increased from 10 to 15 for synthetic instances (0.543 to 0.536). On the other hand, *CG2* can consistently improve feasible solutions as we increase *C* for both instance groups.

Tables 5 and 6 summarize the computational results for *MIP*, *MIP+*, *CG1*, and *CG2* on real and synthetic instances, respectively. Column *gap* represents the optimality gap (in percentage), and is calculated as $\frac{UB-LB}{LB} \times 100$, where *UB* and *LB* respectively denote the best upper bound and lower bound found by an algorithm. We calculate a trivial upper bound by solving the relaxation of linearized *MIQP* for those instances that *CG1* and *CG2* are not able to prove an upper bound through column generation within the time limit, we use this trivial bound to calculate *gap*. Column *time* (in seconds) is averaged over only those instances that are proved to be solved optimally.

As can be seen in Table 5, for the smaller instances, the performance of the algorithms for the number of instances that are solved optimally are similar, with *MIP* and *MIP+* performing slightly better. On the other hand, for those small instances that are not solved optimally, *CG2* proves significantly lower optimality gaps. As expected, increasing the size of the instances degrades the performance of the algorithms. For large instances, *CG2* solves 21 instances optimally, while *MIP*, *MIP+*, and *CG1*, optimally solve only 1, 0, and 11 instances, respectively. On the other hand, for large instances, as *C* increases, *CG1* and *CG2* result in higher optimality gaps compared to *MIP+*. This is mainly due to the weak trivial bound that we use for those instances for which the column generation procedure does not terminate. We also see that *MIP+* proves much lower optimality gaps compared to *MIP*, which is because *MIP+* is able to find higher quality feasible solutions with the help of the heuristic algorithm.

As can be seen in Table 6, *CG1* and *CG2* perform better with respect to the number of synthetic instances that are solved optimally. *CG2* is able to prove much smaller optimality gaps on average. For example, on smaller instances with $C = 5$, the optimality gap of *CG2* is 2% on average, while *MIP* and *MIP+* respectively get 49% and 43% average optimality gaps. The performance of *CG2* de-

Table 7
Summary statistics of movements between counties for the eight-week timeframe.

Variable	Obs.	Mean	Median	Std. Dev.	Skewness
Movements between counties	1189	27,087	395	99,162	8.23

grades as C increases. For example, on larger instances, similar to the real instance, we see that the performance of all algorithms degrades significantly. *CG1* and *CG2* solve 3 instances to optimality, while *MIP* and *MIP+* cannot solve any of the instances. We also find that with a smaller C, *CG2* proves a significantly smaller optimality gap (9% on average) compared to other algorithms (e.g., 49% for *MIP+*). However, similar to other circumstances, the performance of *CG1* and *CG2* degrades as C increases.

Overall, among our algorithms, for both real and synthetic instances, we find that *CG2* is performing better than the others in finding higher quality solutions in a shorter time while proving smaller optimality gaps on average. The outperformance of CG2 is more significant particularly for those instances with tighter cardinality constraints ($C=5$). Moreover, we let *MIP* run for 12 h on the synthetic instances to better understand its behavior and highlight the benefits of our proposed algorithms. The summary of our results of these runs are presented in Appendix D. We find that the performance of *MIP* improves in finding higher quality solutions. However, with 12-hour execution, *MIP* still cannot perform as good as the proposed column-generation algorithms (that is executed for an hour) in terms of number of problems solved optimally and the average optimality gaps.

6. Analysis of modeling results

To analyze the effect of the formulation of possible policy coordination zones for pandemic responses based on movement of people across counties based on our model, we select two states in the U.S. that are highly interconnected by people movement and apply the proposed community detection model in Section 6.1 to identify optimal coordination communities within these interconnected states. Then, in Section 6.2, using COVID-19 data, we evaluate if the identified communities are more suitable for the coordination of NPIs and vaccination programs than current state-based policy making. Finally, Section 6.3 provides managerial insights from the case study.

6.1. Case study

We consider counties within the two states of North and South Carolina as the SUs in our model. There are 146 counties in total in these two states. The county adjacency data is downloaded from the U.S. Census Bureau website. Based on the adjacency data, the counties in our sample have a minimum of 2 and a maximum of 9 neighbors, with an average of 5.44 and a median of 5 neighbors. Next, we collected the movement of people between the 146 counties for the eight-week period between January 20 and March 15, 2020. We specifically select this timeframe as it captures the natural movement of people prior to any major pandemic-related lockdown in the U.S. The mobility data, provided by Camber System (https://cambersystems.com) after signing a data usage agreement, is collected using aggregated and anonymous location data of smartphones. The number of movements (transitions) between counties in the original dataset is reported in four-hour blocks and includes 208,470 records, with a total of 32,206,144 movements. As an example, Table E1 in Appendix E shows the movement from Lexington County, South Carolina, to Richland County, South Carolina, on January 20, 2020.

We calculated the total number of movements between each pair of counties by aggregating data over the eight weeks. Table 7 provides summary statistics of movements between counties for the eight-week timeframe. In summary, there are 1189 unique between counties movements in the aggregated dataset. The average number of movements is 27,087, with a median of 395 and a standard deviation of 99,162. The movement distribution is highly skewed to the right, which indicates that there are exceptionally strong connections among some counties (and thus good candidates to establish local communities). The maximum number of movements is between Lexington County, South Carolina, and Richland County, South Carolina (1527,033 total transitions). Table E2 in Appendix E summarizes the top 20 movements between counties. All of the top 20 movements are within states borders, except for the movement between Mecklenburg County, North Carolina, and York County, South Carolina, which is ranked as number 7. The top 20 inter-state movements are also reported in Table E3 in Appendix E, where the ranking column shows the overall ranking in the dataset.

The case study is solved using *CG2* algorithm with different C values. For all scenarios, the algorithm was able to find near optimal solutions (with less than 1% optimality gap) within 4 h. Table 8 summarizes the results, and Fig. 4 shows counties in the same community in the same color on a map. (List of all counties and their assigned community with respect to C is reported in Appendix F). We first notice that the contiguity constraint is satisfied in all cases. When the number of counties in a community is limited to 5 ($C=5$), the model creates 35 communities with a modularity value of 0.5952 and a minimum (maximum) number of counties per community of 1 (5). In this case, the average number of counties per community is greater than 4, which indicates most communities contain the maximum number of counties allowed in the communities. Additionally, three communities have counties from both states. When the number of counties in a community is limited to 10 ($C=10$), the model creates 23 communities with a modularity value of 0.7195. A large number of communities (11 out of 23) contain the maximum number of counties allowed, and there are six communities that contain only one county. When the number of counties in a community is limited to 15 ($C=15$), the model creates 13 communities, and the modularity value is 0.7411.

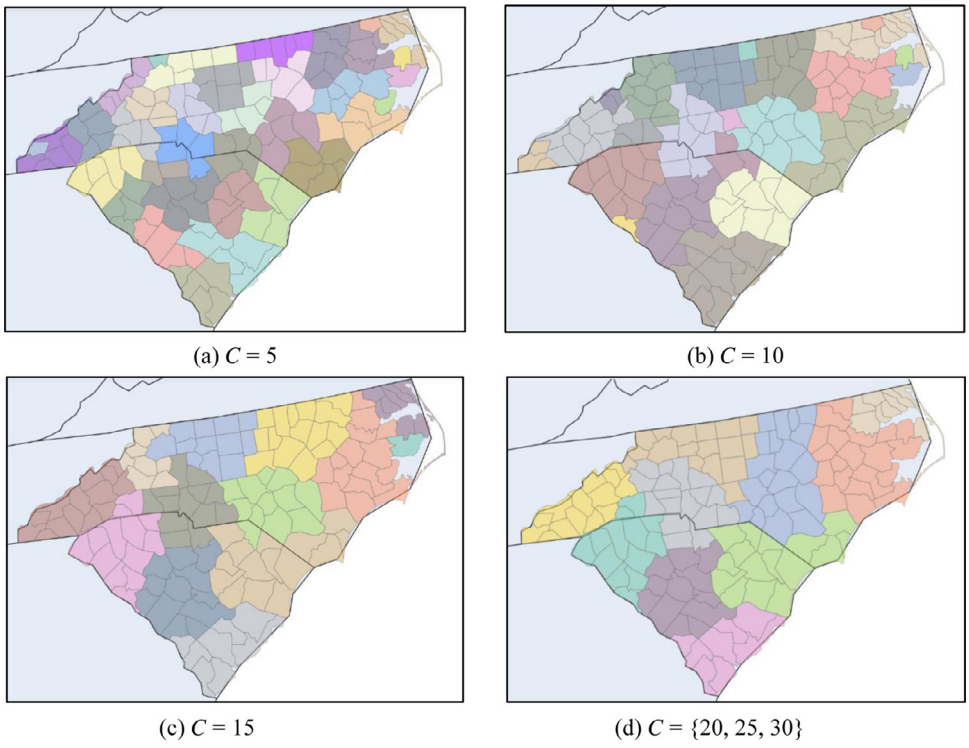
The modularity value increases with increasing C initially, but once an optimal solution is reached (in this case, where the largest community has 20 counties), a further increase of C does not produce better solutions. Fig. 4(d) shows the resulting communities when C is 20, 25, or 30. All communities contain at least eight counties, and the largest communities (two out of ten) include 20 counties. The average number of counties per community is 14.60, and three communities include counties from both states.

6.2. Cluster validation using COVID-19 data

In this section, we aim to demonstrate that the identified clusters/communities, based on mobility, are in fact more suitable for coordinating pandemic NPIs than state-wide coordination. To this end, we compare the impact of COVID-19 on the communities created based on the proposed model with that on individual states. Mobility data prior to any major lockdown in the U.S. is used in the previous section to identify coordination communities for pandemic responses within North and South Carolina. Since counties in the same community are highly interconnected with people's movements, we expect that the impact of COVID-19 on counties

Table 8
Results of case study.

C	N. of communities	N. of counties in the smallest community	N. of counties in the largest community	Average counties per community	Modularity
5	35	1	5	4.17	0.5952
10	23	1	10	6.35	0.7195
15	13	1	15	11.23	0.7411
20	10	8	20	14.60	0.7505
25	10	8	20	14.60	0.7505
30	10	8	20	14.60	0.7505



(a) $C = 5$ (b) $C = 10$ (c) $C = 15$ (d) $C = \{20, 25, 30\}$

Fig. 4. Communities based on mobility data within North Carolina and South Carolina.

within a community to be similar; that is, the number of cases and deaths due to COVID-19 should have lower variation within communities compared to those between communities or cross states boundaries. To officially test this argument, a clustering validation method is used in this section.

Clustering validation methods, which evaluate the goodness of clustering results, are commonly used to study the success of clustering algorithms (Liu, Li, Xiong, Gao, & Wu, 2010). These methods are divided into two categories: external clustering validation and internal clustering validation. External validation measures are used when the "true" cluster of elements is known, while internal validation measures are appropriate when the "true" cluster information is not available for comparison. In general, evaluating the performance of clustering algorithms when the true clusters are not known is more challenging than when the true clusters are known. Since, in our problem, the true cluster of counties is not known, we use a popular internal validity measure known as Caliński-Harabasz (CH) index (Caliński & Harabasz, 1974) to evaluate the success of our proposed model (Liu et al., 2010; Łukasik et al., 2016). The CH index calculates the ratio of the between-clusters sum of squares (β) and within clusters sum of squares (ω) and is adjusted based on the total number of clusters (κ) (i.e., communities in our study) and vertices (n) (i.e., counties in our study) within each cluster, as shown in Eq. (10):

$$CH = \frac{\beta}{\omega} \times \frac{n-\kappa}{\kappa-1} \tag{10}$$

where

$$\beta = \sum_{i=1}^{\kappa} n_i E^2(s_i, s) \tag{11}$$

$$\omega = \sum_{i=1}^{\kappa} \sum_{x \in s_i} E^2(x, s_i) \tag{12}$$

Eqs. (11) and (12) calculate the between-clusters sum of squares and the within clusters sum of squares, respectively, where n_i represents the number of counties in community i (S_i), s_i is the center of S_i, s is the grand center, and $E(x_1, x_2)$ calculates the Euclidean distance between x_1 and x_2. A higher value of CH index means the SUs are more homogenous within communities and more heterogeneous between communities. Therefore, a clustering method with a higher CH index value is preferred.

To better illustrate the idea of using CH index, consider nine interconnected counties shown in Fig. 5a. Assume that three different clustering methods are applied to detect communities, and their results are presented in Figs. 5b to 5d. COVID-19 data of each

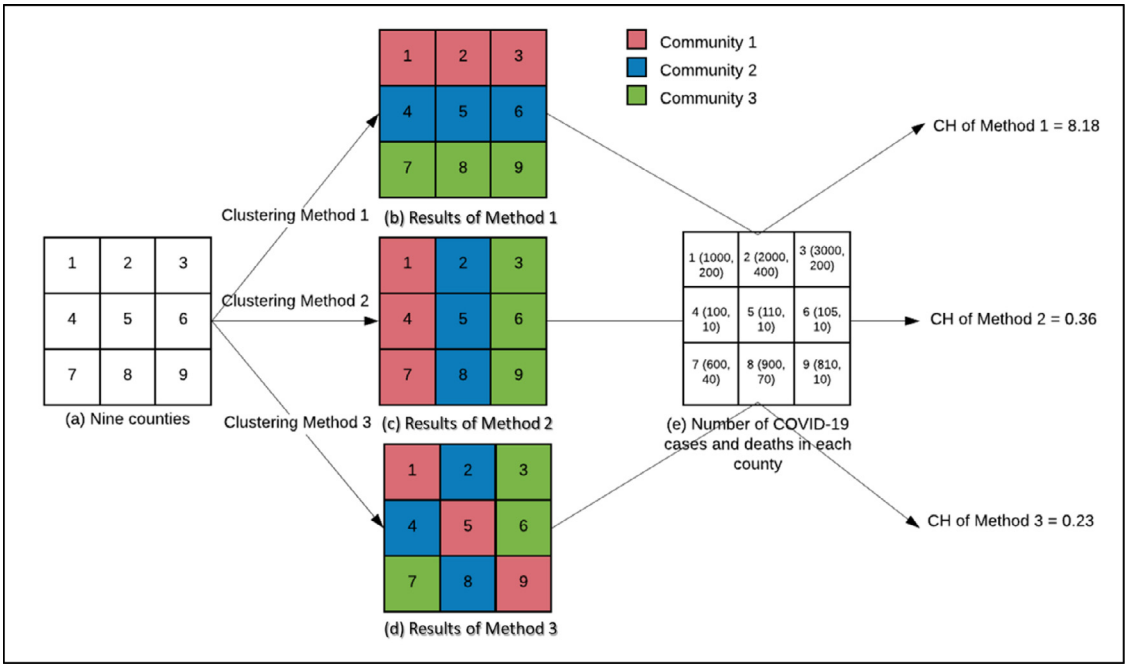


Fig. 5. An example of CH index values for three different clustering methods.

Table 9
CH values.

Communities	CH value
$C = 5$	1.62
$C = 10$	1.02
$C = 15$	1.24
$C = \{20, 25, 30\}$	1.36
States	0.68

county, which are available after the pandemic starts, can be used to evaluate the performance of the three clustering methods. Suppose the numbers in the parentheses in Fig. 5e denote the confirmed number of COVID-19 cases and deaths for each county, respectively. It is easy to visually see that the first clustering method is better than the other two methods because it creates more homogenous communities with respect to the impact of COVID-19. We obtain a similar conclusion using CH index. The CH index value of the first clustering method, calculated based on Eqs. (5) to 7, is 8.18, which is higher than the CH index values of the other two clustering methods (i.e., 0.36 and 0.23). Therefore, as our example demonstrates, CH index can be used to identify the clustering method that provides more homogenous communities.

To calculate CH index in our case study, we first collected the number of confirmed cases and deaths due to COVID-19 per county per day for each county in the two states for a period of eight weeks (June 1, 2020 to July 26, 2020) from the Johns Hopkins University tracking website (Dong, Du, & Gardner, 2020). Then, the total number of confirmed cases and deaths in each county were calculated for the eight-week period for comparison. Next, we calculate CH index values of the community structures in this case study (for C = 5, 10, 15, {20, 25, 30}) as well as the state structure (two communities each representing the individual states).

Table 9 provides a summary of CH index values. All communities created based on the movement of people are more homogenous than considering counties within states as their own community. For instance, communities based on $C = 5$ and $C = \{20, 25, 30\}$ have CH values (respectively 1.62 and 1.36) almost twice as the CH value of communities based on states borders (0.68). To check the robustness of the results, we also calculated the CH index values based on only the number of cases and then also the number of deaths, and the CH index values calculated based on the number of cases and deaths separately are very similar to those reported in Table 9.

This case study shows the close interconnections, based on people's movement, between counties within and between two states, and the communities created with our modularity-based model exhibit more homogeneous COVID-19 impacts than individual states. The results highlight the importance of considering coordinating NPIs and vaccination programs across counties and states based on people's natural movement without considering the geopolitical borders.

6.3. Implications for pandemic management

This case study provides insight into managerial and policy decisions in combating pandemics. There is significant variability between COVID-19 policies in North Carolina (NC) and South Carolina (SC) since the start of COVID-19 (NASHP, 2021). Original stay-at-home orders in NC were from March 30, 2020 through May 22, 2020 with violations punishable as a Class 2 misdemeanor, while SC had a shorter window from April 6, 2020 through May 4, 2020 with lax enforcement. Mask mandates also differ in similar fashion between the two states, while there were no state-wide travel restrictions in either state since the beginning of the pandemic. Additionally, reopening policies and responses to the rapidly spreading Delta variant in 2021 in both states were just as divergent. Our analysis shows that up to four border communities include counties from both states. Given that the profile of people at risk is similar in both states, local county governments from both states belonging to border communities identified in this study should collaborate to create a community-level policy and response strategy including vaccinations, masking, and limits on inside gathering, instead of following disparate state-level policy. This is especially vital given the highly transmissible Delta variant (and future variants to come).

Our analysis, along with a case study, shows that the proposed model can create communities that are significantly better

for coordinating pandemic related policies than those solely based on the existing geopolitical borders. This result is validated using COVID-19 data. The consequence of non-coordination across state borders has been significant in some instances. For example, it has resulted in an influx of COVID-19 patients from western parts of Idaho where masks and vaccinations are not mandated, to hospitals across the border in Washington state. This has overwhelmed hospital capacity in the latter, deepening the crisis in that state. Since the proposed communities are created based on the natural movement of people, jurisdictions within a community are well-connected and potentially in the same phase of pandemic. In addition, to facilitate the coordination of NPI policies, our model guarantees the contiguity of detected communities and limits the maximum number of jurisdictions within a community. As such, the proposed model can be used to identify communities of contiguous locations for applying customized and coordinated policies response to a pandemic.

We are beginning to see local responses by cities and school districts despite state mandates, but the effectiveness of such local policies can be enhanced by joining those administrative units into communities, as identified in this study, to provide a pathway to coordinated reopening. In a federally structured country like the U.S., in addition to an effective national pandemic policy, an appropriately coordinated multi-level approach of regional and local policy is required. As any of the communities of counties identified in our case study outperform the community as defined by state boundaries, this study indicates that a new community level of governance is required, based on population movement and beyond existing geographic boundaries of state and county, for effective pandemic management. Our model provides a framework to identify which county jurisdictions need to work together to manage transboundary public health crises. This is especially valid for contagious viruses like Influenza (flu) and COVID-19, where the movement and interactions among people are the essential mechanisms behind transmitting. Further, although this study is specifically designed to coordinate pandemic response policies, the model may be used in other fields to identify contiguous communities, such as in marketing for clustering customers based on their locations and daily movements.

7. Conclusion

The existing literature suggests that coordinating policies across interconnected locations can improve the effectiveness of pandemic interventions, especially in decentralized and federal countries (OECD, 2020; Ruktanonchai et al., 2020). This study proposes a new community detection model to identify communities based on the natural movement of people for coordinating pandemic related policies. To do so, we develop a new mixed-integer quadratic-programming model to maximize modularity subject to the geographic contiguity constraints. The optimization model uses the natural movement of people as inputs to calculate modularity, and the number of jurisdictions in each community or zone can be adjusted based on the policy maker's requirements. To improve computation, we also develop a column-generation algorithm that is coupled with an iterative pricing procedure (*CG2*) to solve larger instances. Our computational experiments highlight that *CG2* outperforms the standard mixed-integer quadratic-programming model using a commercial solver.

There are several possible opportunities to extend this research. First, we use contiguity requirements as hard constraints in the optimization models. A potential extension is to consider a bi-objective optimization model where one can analyze the implications of balancing modularity and contiguity in community detection. Another possible direction of future research is to incorporate other commonly used community detection objectives such as compactness or modularity density functions while enforcing contiguity requirements. Strengthening the *MIQP* formulation by finding valid inequalities is another possible direction for future researchers. Lastly, our column-generation algorithm is designed to solve the pricing problem optimally in each iteration. While in most cases this approach is effective and the pricing problems are solved quickly, we find that for larger instances, it may take a bit longer (up to a few minutes), which increases the algorithms' overall run time. As a future extension, a combination of heuristics and exact methods in the iterations of the pricing problem can be incorporated to solve the optimization problem more quickly while ensuring valid optimization bounds.

Acknowledgments

The authors would like to gratefully acknowledge Camber Systems for making the mobility data available for this study.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ejor.2022.01.012.

References

Aloise, D., Cafieri, S., Caporossi, G., … Liberti, L. (2010). Column generation algorithms for exact modularity maximization in networks. *Physical Review E, 82*(4), 46112.

Ashourvan, A., Telesford, Q. K., Verstynen, T., … Bassett, D. S. (2019). Multi-scale detection of hierarchical community architecture in structural and functional brain networks. *PloS one, 14*(5), Article e0215520.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., … Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research, 46*(3), 316–329.

Bergey, P. K., Ragsdale, C. T., & Hoskote, M. (2003a). A decision support system for the electrical power districting problem. *Decision Support Systems, 36*(1), 1–17.

Bergey, P. K., Ragsdale, C. T., & Hoskote, M. (2003b). A simulated annealing genetic algorithm for the electrical power districting problem. *Annals of Operations Research, 121*(1–4), 33–55.

Blondel, V. D., Guillaume, J.-. L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment, 2008*(10), P10008.

Bozkaya, B., Erkut, E., & Laporte, G. (2003). A tabu search heuristic and adaptive memory procedure for political districting. *European Journal of Operational Research, 144*(1), 12–26.

Brandes, U., Delling, D., Gaertler, M., … Wagner, D. (2006). Maximizing modularity is hard. Available at ArXiv: https://arxiv.org/abs/physics/0608255v2.

Bulka, B., Carr, R., Jordan, E., & Rheingans, P. (2007). Heuristic search and information visualization methods for school redistricting. *AI Magazine, 28*(3), 59.

Camacho-Collados, M., Liberatore, F., & Angulo, J. M. (2015). A multi-criteria police districting problem for the efficient and effective design of patrol sector. *European Journal of Operational Research, 246*(2), 674–684.

Caro, F., Shirabe, T., Guignard, M., & Weintraub, A. (2004). School redistricting: Embedding GIS tools with integer programming. *Journal of the Operational Research Society, 55*(8), 836–849.

Carvajal, R., Constantino, M., Goycoolea, M., … Weintraub, A. (2013). Imposing connectivity constraints in forest planning models. *Operations Research, 61*(4), 824–836.

CBS New York. (2020). *N.Y., N.J., Conn. Announce joint travel advisory, all travelers from states with high coronavirus infection rates must quarantine*. CBS NEW York https://newyork.cbslocal.com/2020/06/24/n-y-n-j-conn-announce-joint-travel-advisory-all-visitors-from-states-with-high-coronavirus-infection-rates-must-quarantine/.

CDC. (2021). *CDC COVID Data Tracker. The Centers for Disease Control and Prevention* https://covid.cdc.gov/covid-data-tracker/#datatracker-home.

Chang, S., Pierson, E., Koh, P. W., … Leskovec, J. (2021). Mobility network models of COVID-19 explain inequities and inform reopening. *Nature, 589*(7840), 82–87.

Clauset, A., Newman, M. E. J., & Moore, C. (2004). Finding community structure in very large networks. *Physical Review E, 70*(6), 66111.

Costa, A. (2015). MILP formulations for the modularity density maximization problem. *European Journal of Operational Research, 245*(1), 14–21.

Cutler, D. M., & Summers, L. H. (2020). The COVID-19 pandemic and the $16 trillion virus. *JAMA, 324*(15), 1495–1496.

Dhillon, R. S., & Karan, A. (2020). The U.S. needs smarter lockdowns Now. *Harvard business review - Economics & society*. Retrieved from https://hbr.org/2020/08/the-u-s-needs-smarter-lockdowns-now. Accessed January 2, 2022.

Dong, E., Du, H., & Gardner, L. (2020). An interactive web-based dashboard to track COVID-19 in real time. *The Lancet infectious diseases, 20*(5), 533–534.

Dugošija, D., Savić, A., & Maksimović, Z (2020). A new integer linear programming formulation for the problem of political districting. *Annals od Operations Research, 288*, 247–263.
Fajgelbaum, P., Khandelwal, A., Kim, W., … Schaal, E. (2020). Optimal lockdown in a commuting network. In National Bureau of Economic Research Working Paper Series, No. 27441.
Farughi, H., Tavana, M., Mostafayi, S., & Santos Arteaga, F. J. (2020). A novel optimization model for designing compact, balanced, and contiguous healthcare districts. *Journal of the Operational Research Society, 71*(11), 1740–1759.
Fortunato, S., & Hric, D. (2016). Community detection in networks: A user guide. *Physics Reports, 659*, 1–44.
Fryer, R. G., Jr, & Holden, R. (2011). Measuring the compactness of political districting plans. *The Journal of Law and Economics, 54*(3), 493–535.
Garfinkel, R. S., & Nemhauser, G. L. (1970). Optimal political districting by implicit enumeration techniques. *Management Science, 16*(8) B-495.
Gilmore, P. C., & Gomory, R. E. (1961). A linear programming approach to the cutting-stock problem. *Operations Research, 9*(6), 849–859.
Glaeser, E. L., Gorback, C., & Redding, J. R. (2020). JUE Insight: How much does Covid-19 increase with mobility? Evidence from New York and four other Us Cities. *Journal of Urban Economics*. https://doi.org/10.1016/j.jue.2020.103292.
Gordon, S. H., Huberfeld, N., & Jones, D. K. (2020). What federalism means for the US response to coronavirus disease 2019. *JAMA Health Forum, 1*(5), Article e200510.
Haase, K., & Müller, S. (2014). Upper and lower bounds for the sales force deployment problem with explicit contiguity constraints. *European Journal of Operational Research, 237*(2), 677–689.
Han, J., Hu, Y., Mao, M., & Wan, S. (2020). A multi-objective districting problem applied to agricultural machinery maintenance service network. *European Journal of Operational Research, 287*(3), 1120–1130.
Hess, S. W., Weaver, J. B., Siegfeldt, H. J., … Zitlau, P. A. (1965). Nonpartisan political redistricting by computer. *Operations Research, 13*(6), 998–1006.
Hojati, M. (1996). Optimal political districting. *Computers & Operations Research, 23*(12), 1147–1161.
Holtz, D., Zhao, M., Benzell, S. G., … Aral, S. (2020). Interdependence and the cost of uncoordinated responses to COVID-19. *Proceedings of the National Academy of Sciences, 117*(33) 19837 LP –19843.
Huang, H., Shen, H., Meng, Z., … He, H. (2019). Community-based influence maximization for viral marketing. *Applied Intelligence, 49*(6), 2137–2150.
International Monetary Fund (IMF). (2020). *Fiscal monitor: Policies for the recovery*. Retrieved from https://www.imf.org/en/Publications/FM/Issues/2020/09/30/october-2020-fiscal-monitor .
Ki, N., Kwak, C.-. G., & Song, M. (2020). Strength of strong ties in intercity government information sharing and county jurisdictional boundaries. *Public Administration Review, 80*(1), 23–35.
Kim, M. J. (2018). Multiobjective spanning tree based optimization model to political redistricting. *Spatial Information Research, 26*(3), 317–325.
Kraemer, M. U. G., Yang, C.-. H., Gutierrez, B., … Scarpino, S. V. (2020). The effect of human mobility and control measures on the COVID-19 epidemic in China. *Science, 368*(6490) 493 LP –497.
Lancichinetti, A., & Fortunato, S. (2009). Community detection algorithms: a comparative analysis. *Physical review E, 80*(5), Article 056117.
Lancet, T. (2020). Humanitarian crises in a global pandemic. *The Lancet, 396*(10249), 447.
Laroze, D., Neumayer, E., & Plümper, T. (2021). COVID-19 does not stop at open borders: Spatial contagion among local authority districts during England's first wave. *Social Science & Medicine, 270*, Article 113655.
Latapy, M., & Pons, P. (2004). Computing communities in large networks using random walks. Available at ArXiv: https://arxiv.org/abs/cond-mat/0412368v1.
Lawler, E. L., & Wood, D. E. (1966). Branch-and-bound methods: A survey. *Operations Research, 14*(4), 699–719.
Li, Z., Zhang, S., Wang, R.-. S., … Chen, L. (2008). Quantitative function for community detection. *Physical Review E, 77*(3), 36109.
Liu, Y., Li, Z., Xiong, H., Gao, X., & Wu, J. (2010). Understanding of internal clustering validation measures. In *Proceedings of the 2010 IEEE international conference on data mining*.
McCormick, G. P. (1976). Computability of global solutions to factorable nonconvex programs: Part I — Convex underestimating problems. *Mathematical Programming, 10*(1), 147–175.
Moses, M. E. (2021). How to Fix the Vaccine Rollout: A computational biologist charts a fair and efficient course for vaccine distribution. *Nautilus* Retrieved from https://nautil.us/issue/95/escape/how-to-fix-the-vaccine-rollout. Accessed January 2, 2022.
NASHP (2021). States' COVID-19 Public Health Emergency Declarations and Mask Requirements, Retrieved from https://www.nashp.org/governors-prioritize-health-for-all/. Accessed July 21, 21.
Newman, M. E. J. (2006). Finding community structure in networks using the eigenvectors of matrices. *Physical Review E, 74*(3), 36104.
Newman, M. E. J., & Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review E, 69*(2), 26113.
OECD. (2020). *Policy responses to coronavirus (COVID-19), The territorial impact of COVID-19: Managing the crisis across levels of government* http://www.oecd.org/coronavirus/policy-responses/the-territorial-impact-of-covid-19-managing-the-crisis-across-levels-of-government-d3e314e1/.
Önal, H., Wang, Y., Dissanayake, S. T. M., & Westervelt, J. D. (2016). Optimal design of compact and functionally contiguous conservation management areas. *European Journal of Operational Research, 251*(3), 957–968.
Ricca, F., Scozzari, A., & Simeone, B. (2013). Political districting: From classical models to recent approaches. *Annals of Operations Research, 204*(1), 271–299.
Ricca, F., & Simeone, B. (2008). Local search algorithms for political districting. *European Journal of Operational Research, 189*(3), 1409–1426.
Rosvall, M., Delvenne, J., Schaub, M. T., & Lambiotte, R. (2019). Different approaches to community detection. *Advances in Network Clustering and Blockmodeling*, 105–119.
Ruktanonchai, N. W., Floyd, J. R., Lai, S., … Steele, J. E. (2020). Assessing the impact of coordinated COVID-19 exit strategies across Europe. *Science, 369*(6510), 1465–1470.
Santiago, R., & Lamb, L. C. (2017). Efficient modularity density heuristics for large graphs. *European Journal of Operational Research, 258*(3), 844–865.
Sato, K., & Izunaga, Y. (2019). An enhanced MILP-based branch-and-price approach to modularity density maximization on graphs. *Computers & Operations Research, 106*, 236–245.
Shahzamal, M., Mans, B., de Hoog, F., … Jurdak, R. (2020). Vaccination strategies on dynamic networks with indirect transmission links and limited contact information. *PloS One, 15*(11), Article e0241612.
Shirabe, T. (2005). A model of contiguity for spatial unit allocation. *Geographical Analysis, 37*(1), 2–16.
Shirabe, T. (2009). Districting modeling with exact contiguity constraints. *Environment and Planning B: Planning and Design, 36*(6), 1053–1066.
Shirazi, S., Albadvi, A., Akhondzadeh, E., … Teimourpour, B. (2020). A new application of community detection for identifying the real specialty of physicians. *International Journal of Medical Informatics, 140*, Article 104161.
Tanrisever, F., Shahmanzari, M., Eryarsoy, E., & Şensoy, A. (2021). Optimizing Disease Containment Measures during a Pandemic. Available at SSRN: https://ssrn.com/abstract=3795643
Traag, V. A., Waltman, L., & van Eck, N. J. (2019). From Louvain to Leiden: Guaranteeing well-connected communities. *Scientific Reports, 9*(1), 5233.
Trein, P., Biesbroek, R., Bolognesi, T., … Meyer, I. (2021). Policy coordination and integration: A research agenda. *Public Administration Review, 81*(5), 973–977.
Wang, J., Zheng, W., Qian, Y., & Liang, J. (2017). A seed expansion graph clustering method for protein complexes detection in protein interaction networks. *Molecules, 22*(12), 2179.
Watts, A. (2020). US is in a "very critical time right now" as states begin to reopen, Fauci says. *CNN* Retrieved from https://www.cnn.com/us/live-news/us-coronavirus-update-04-23-20/h_bd4b95dc5c116cf935288209e250fc6e. Accessed January 2, 2022.
Zhang, Y., Liu, Y., Li, Q., … Wen, C. (2020). LILPA: A label importance based label propagation algorithm for community detection with application to core drug discovery. *Neurocomputing, 413*, 107–133.