SN Computer Science (2021) 2:83
https://doi.org/10.1007/s42979-021-00462-6

ORIGINAL RESEARCH

Application of Tolerance Graphs to Combat COVID-19 Pandemic

Dean Crnković[1] · **Andrea Švob**[1]

Received: 27 November 2020 / Accepted: 9 January 2021 / Published online: 8 February 2021

Abstract
Tolerance graphs were introduced in 1982 by Golumbic and Monma as a generalization of interval graphs. In this paper, we propose several applications of tolerance graphs in fighting COVID-19. These applications include finding cliques of a certain size, and calculating the chromatic number of a graph, the problems that are in general NP-complete but for tolerance graphs can be solved in polynomial time.

Keywords Pandemic · Tolerance graph · Perfect graph · Chromatic number · Clique

Introduction

Nowadays, COVID-19 is causing terrible loss of lives all around the world, and also has a devastating impact on economics. Computer aid in this battle can be crucial. Mathematics and computer science can work out many models to help to fight the pandemic. Graph theory is a part of mathematics that studies graphs, which are mathematical structures that can be used to model certain relations between object. Graphs arise as a natural tool in modeling various problems related to the COVID-19 pandemic. They are also one of the basic parts of making artificial intelligence use; for example tracing the contacts of infected people, doing some medical analysis for understanding the virus and disease dynamics, and, in general, predicting the dynamics of COVID-19 pandemic. Recently, many papers dealt with this topic. For example, Zhu et al. [1] propose an novel epidemic model which can be used to calculate the spread process of COVID-19 epidemic in Wuhan city and to estimate an unknown data. In [2], Postavaru et al. calculate the number of infections considering the chaotic contributions, etc.

In this paper, we will show how graph theory, in particular tolerance graphs, can be used to model various situation that occur during an pandemic and help fighting COVID-19 by slowing down the process of spreading the virus. This approach can be used in fighting other pandemics too.

Preliminaries

A *graph* $G = (V, E)$ consists of a finite set V of vertices together with a set E of edges, where an edge is a subset of the vertex set of cardinality 2. Some literature calls this a simple graph (a graph without loops or multiple edges). For basic definitions and further reading in graph theory, we refer the reader to [3].

An *induced subgraph* of a graph is a subgraph formed from a subset of the vertices of the graph and all of the edges connecting pairs of vertices in that subset.

A *clique*, C, in a graph $G = (V, E)$ is a subset of the vertices, $C \in V$, such that every two distinct vertices are adjacent. A *maximal clique* is a clique that cannot be extended by including one more adjacent vertex, i.e., a maximal clique is a clique not contained in any larger clique. A *maximum clique* of a graph G is a clique, such that there is no clique with more vertices. Moreover, the clique number $\omega(G)$ of a graph G is the number of vertices in a maximum clique in G.

A *vertex coloring* of a graph $G = (V, E)$ is a map $c : V \to S$, such that $c(v) \neq c(w)$ whenever v and w are adjacent. Some literature calls this a proper coloring. Given an

This article is part of the topical collection "Computer Aided Methods to Combat COVID-19 Pandemic" guest edited by David Clifton, Matthew Brown, Yuan-Ting Zhang, and Tapabrata Chakraborty.

✉ Andrea Švob
asvob@math.uniri.hr

Dean Crnković
deanc@math.uniri.hr

[1] Department of Mathematics, University of Rijeka, Radmile Matejčić 2, 51000 Rijeka, Croatia

integer k, a k-coloring c is a map that assigns to each vertex v of the graph G an integer $c(v)$ chosen in the set $\{1, 2, \ldots, k\}$ (the set of colors). The smallest integer k, such that G has k-coloring, is the chromatic number that is usually denoted by $\chi(G)$. A graph G having that $\chi(G) = k$ is called k-chromatic, and if $\chi(G) \leq k$, we call G k-colorable.

Tolerance graphs were introduced in 1982 by Golumbic and Monma [4], at the 13th Southeastern Conference on Combinatorics, Graph Theory and Computing held at Boca Raton, as a mathematical model of tolerance to generalize some of the well-known applications associated with interval graphs. The motivation was the need to solve several scheduling problems in various situations. After the introduction of tolerance graphs, properties and several variations of the graphs were studied in the literature, for example in [5, 6].

The notation and most important definitions used in this work follow the book [5]. First, we have to remark that we are interested in a collection of intervals on the real line. In this paper, the intervals will come from various problems connected with COVID-19, but in general, they might arise from other applications. We are interested in a class of graphs called intersection graphs. Let $\mathcal{F}$ be a collection of set. The *intersection graph* of $\mathcal{F}$ is defined as the graph obtained by assigning a distinct vertex to each set in $\mathcal{F}$ and joining two vertices by an edge when their corresponding sets have nonempty intersection. Before defining tolerance graphs, we will introduce another family of intersection graphs, i.e., interval graphs which are actually the special case of tolerance graphs. An *interval graph* $G = (V, E)$ is a graph for which each vertex $v \in V$ is associated with an real interval I_v, and two vertices are connected by an edge in G if the associated intervals have nonempty intersection. The set of intervals $\{I_v \mid u \in V\}$ is an interval graph representation of G. In other words:

$$uv \in E(G) \Longleftrightarrow I_u \cap I_v \neq \emptyset, \quad \text{for all } u, v \in V(G).$$

Tolerance graphs are defined in a similar manner as interval graphs, but here, the sizes of intersecting intervals will have the crucial role. Exact definition of tolerance graphs is given as follows. A graph $G = (V, E)$ is a *tolerance graph* if each vertex $v \in V$ can be assigned a closed interval I_v and a tolerance $t_v \in \mathbb{R}^+$, such that $xy \in E$ if and only if $|I_x \cap I_y| \geq \min\{t_x, t_y\}$. A collection $\langle I, t\rangle$ of intervals and tolerances is called a *tolerance representation*, $I = \{I_x \mid x \in V\}$, $t = \{t_x \mid x \in V\}$. If, for all $v \in V$, $t_v \leq |I_v|$, the graph is called a *bounded tolerance graph* and the representation is called a *bounded tolerance representation*.

An important property of tolerance graphs is that tolerance graphs are perfect graphs. A graph G is perfect if, for all induced subgraphs H of G, the chromatic number of H equals the number of vertices in a largest clique in H. The perfect graph theorem, proved by Lovász in [7], states that a graph is perfect if and only if its complement graph is also perfect. Furthermore, the strong perfect graph theorem of Chudnovsky, Robertson, Seymour, and Thomas [8] states that a graph is perfect if and only if it has neither odd cycles of length at least 5 nor their complements as induced subgraphs. Both of these theorems were conjectured by Berge.

One of the highest importance why one should use perfect graphs is the following. Perfect graphs have the property that for some problems that are in general NP-complete, they allow polynomial-time algorithms for obtaining the solution. In [9], Karp proved that computing the chromatic number of any graph is NP-complete, while in [10], it was shown that for perfect graphs, the chromatic number can be computed in polynomial time. Furthermore, finding cliques of a certain size is an NP-complete problem and determining the clique number of the graph is an NP-hard problem. However, for perfect graphs, these problems can be solved in polynomial time (see [10]). For more information, see [11].

For the computations dealing with the problems given in this paper, one can use Python [12] and SageMath [13]. Furthermore, Cliquer [14] is a computer program that is efficient in finding cliques of a certain size and determining the clique number of a graph.

Tolerance Graphs in Fighting COVID-19 Pandemic

In this section, we propose four applications of tolerance graphs which could be used in combating COVID-19 pandemic by limiting the spread of the disease. Each application is followed by a computer program written in SageMath [13]. The only demanding parts of the presented algorithms (and programs) are the parts where the clique number (and a maximal clique) or the chromatic number (and a $\chi(G)$-coloring) of a graph G has to be determined. Hence, the complexities of the given algorithms depend only on the complexities of the algorithms for finding the clique number and the chromatic number implemented in a particular software.

As an illustration of the proposed applications of tolerance graphs, in "Appendix", we give an example related to Sect. 3.2 based on real data, namely the flight schedule of the Venice Airport Marco Polo found at their web page https://www.veneziaairport.it/en/.

Search for Critical Events

One of the main things that epidemiologist in this crisis must do to diminish a spread of the virus is to determine the events that can be labeled as critical, meaning that many people could be infected on these events. Epidemiologists could give an advice that these events should be forbidden,

or the number of people that participate in such events should be restricted. This problem can be modeled in the following way. Let G be a graph whose set of vertices correspond to the set of intervals $I = \{I_1, \ldots, I_v\}$. Each vertex is represented by $I_j = [s_j, e_j]$, where s_j and e_j are the initial time when person j is infected and the ending time of the infection, respectively. To each vertex, we add the tolerance, t_i, $i = 1, \ldots, v$ which is determined by the epidemiologists. If $|I_a \cap I_b| \leq \min\{t_a, t_b\}$, for persons a and b, then the vertices corresponding to persons a and b are connected. In the defined tolerance graph, we search for large cliques of particular sizes (or maximal cliques). Looking at a particular clique, epidemiologists could investigate if there was an event that these people were present. If such an event is found, one can conclude that events of this type are critical, and then, precaution measures could be taken to decrease the spread of the disease on similar events.

Below, we give a computer program written in SageMath [13] as an implementation of the algorithm described:

```
input: list of triples (s_j,e_j,t_j) where (s_j,e_j) denotes a
closed interval (representing each vertex) on the real line and t_j is a positive
value i.e. tolerance.
int_tol = [(s_j,e_j,t_j) for j in [1,...,v]]
def critical_events(int_tol):
g = graphs.ToleranceGraph(int_tol)
max_c = g.cliques_maximum()
num_c = g.clique_number()
return [num_c,max_c]
```

We remark here that as output we get all maximum cliques. We might be interested in finding smaller cliques, so we can find all cliques of particular size using the function:

```
list(sage.graphs.cliquer.all_cliques(g)).
```

In the above definition of the interval graph, we might consider alternative approach. Namely, to detect critical events, it may be important to find a situation where there are a large number of people that got infected at the same time, and hence, interval might be the epidemiologist's (interval) estimate of the time that the infection started. In this way, one gets much sparser graph reducing the noise of edges connecting people that have already been infected.

Scheduling Flights

One of the real-life problems that can be modeled with tolerance graphs is connected with scheduling flights during the pandemic crisis. We are analyzing the following problem. The airport X has a rule that the passengers that are using domestic flights must be at the gates 2 h before the flight, and those getting international flights 3 h before. To decrease the possibility of spreading the disease COVID-19 among passengers, the airport wants to schedule flights a and b for different gates if the period of time when the passengers for these two flights should be at the gates overlap for more than 30 min. Under this condition, we are searching for minimal number of different gates that can be used each day at the airport X. Let $I_j = [s_j - 2, s_j]$, where s_j is the time of the flight j scheduled for the day if the flight is domestic and let $I_j = [s_j - 3, s_j]$, where s_j is the time of the flight j scheduled for the day if the flight is international. Furthermore, let $I = \{I_1, \ldots, I_x, I_{x+1}, \ldots, I_{x+y}\}$ be the set of intervals for each flight scheduled for the exact day with x domestic and y international flights. Each interval represents a vertex of the graph G. To each vertex, we add the tolerance, $t_i, i = 1, \ldots, v$ which depend on the number of passengers that are expected at the ith flight. If $|I_a \cap I_b| \leq \min\{t_a, t_b\}$, for flights a and b, the same gates can be used. The minimum number of gates that must be used is equal to the chromatic number of the corresponding tolerance graph (each color corresponds to a gate).

A corresponding computer program written in SageMath [13] is given below:

```
input: list of triples (s_j,e_j,t_j) where (s_j,e_j) denotes a
closed interval (representing each vertex) on the real line and t_j
is a positive value i.e. tolerance.
int_tol = [(s_j,e_j,t_j) for j in [1,...,v]]
def schedule_flights(int_tol):
g = graphs.ToleranceGraph(int_tol)
from sage.graphs.graph_coloring import chromatic_number
cronum = chromatic_number(g)
from sage.graphs.graph_coloring import all_graph_colorings
GraphColoring = all_graph_colorings(g, cronum)
return [g, cronum, next(GraphColoring)]
```

Use of Common Classrooms in Live Teaching During the Pandemic

During the pandemic crisis, there are some meetings and lectures that could be organized as live teaching, i.e., teaching at universities. One of the main problems that appears with this item is the use of common classrooms within different courses. There arise the problem of making optimal schedule of the use of particular classroom. Optimal here means use of minimal possible classrooms for various courses. Let v be the number of courses in one day at some study program that should be organized in the campus building with exactly x rooms. For each course, there exists exact schedule, i.e., an interval or a period of time of the day in each the course must be organized. The time includes the 1 h extra which is supposed to be used for cleaning between two uses. Under this condition, we are searching for the minimal number of different classrooms that can be used each day at the campus building.

Let $I_j = [s_j, e_j + 1]$, where s_j and e_j are the starting and ending time, respectively, of the class j scheduled for the day. We suppose here that the 60 min (1 h) are added for the cleaning. Furthermore, let $I = \{I_1, \ldots, I_v\}$ be the set of intervals for each course scheduled for the exact day with v courses per day. Each interval represents a vertex of the graph G. To each vertex, we add the tolerance t_j, $j = 1, \ldots, v$, where tolerance $t_j$represents the number of students enrolled a course. If $|I_x \cap I_y| \leq \min\{t_x, t_y\}$, for classes x and y, the same classroom can be used. The minimum number of rooms that must be used is equal to the chromatic number of the corresponding tolerance graph (each color corresponds to a classroom). We remark here that the same approach can be used for organizing meeting and many other events.

Below, we give a corresponding computer program written in SageMath [13]:

```
input: list of triples (s_j,e_j+1,t_j) where (s_j,e_j) denotes a
closed interval (representing each vertex) on the real line
and t_j is a positive value i.e. tolerance.
int_tol = [(s_j,e_j+1,t_j) for j in [1,...,v]]
def classroom(int_tol):
g = graphs.ToleranceGraph(int_tol)
from sage.graphs.graph_coloring import chromatic_number
cronum = chromatic_number(g)
from sage.graphs.graph_coloring import all_graph_colorings
GraphColoring = all_graph_colorings(g, cronum)
return [g, cronum, next(GraphColoring)]
```

Museum Visits

One of the main issues for epidemiologists is to detect the contacts of a person infected with the virus. Here, we propose a model that can be helpful in handling this issue. While the pandemic is present in our lives, museum visits, exhibitions, and many other similar events are usually organized within precise rules and schedules. Tolerance graphs can help us to give a list of possible contacts to epidemiologists.

Let $I_j = [s_j, e_j]$, where s_j and e_j are the starting and ending time, respectively, of a person (or a group of people) j visit to a particular museum. Let v be the number of people (or different groups of people) scheduled for visiting museum for each day and let $I = \{I_1, \ldots, I_v\}$ be the set of these intervals. Each interval represents a vertex of the graph G. To each vertex, we add a tolerance t_j, $j = 1, \ldots, v$, where the tolerance t_j represents the period of time that epidemiologist defined for them as a duration of contact that could lead to infection. If $|I_x \cap I_y| \geq \min\{t_x, t_y\}$, for persons (groups) x and y, and a person x is infected by the virus, then epidemiologists may prescribe appropriate measures for the person y. The same approach can be used for other similar events that include different groups of people visiting the same place.

A corresponding computer program written in SageMath [13] is given below:

```
input: list of triples (s_j,e_j,t_j) where (s_j,e_j) denotes a
closed interval (representing each vertex) on the real line and
t_j is a positive value i.e. tolerance.
int_tol = [(s_j,e_j,t_j) for j in [1,...,v]]
def museum_vis(int_tol):
g = graphs.ToleranceGraph(int_tol)
ver = g.vertices()
neigh=[ g.neighbors(v) for v in range(len(ver)) ]
return neigh
```

Conclusion

In this paper, we propose four applications of tolerance graphs in fighting COVID-19. One application is determining events that bring high risk of spreading the disease. The second application is scheduling flights during a pandemic, and the third application is making optimal schedule for sharing common classrooms in live teaching during a pandemic. The fourth application is detecting contacts of a person infected with the virus in a case of museum visits or similar events. These applications are based on finding (large) cliques of a certain size and calculating the chromatic number of a graph, the problems that are in general NP-complete but for tolerance graphs can be solved in polynomial time.

Appendix

Here, we give the application of the algorithm presented in Sect. 3.2 to the departure flights scheduled at the Venice Airport Marco Polo, Italy, for December 28, 2020. The schedule was found at their web page https://www.veneziaairport.it/en/. The labels I and D mark whether the flight is domestic or international (Table 1).

For the domestic flights, we set the tolerance of the corresponding vertex to 0.3 h, and for international flights, the tolerance of the corresponding vertex is set to 0.2 h. Below, we give a computer program written in SageMath that gives us the minimum number of gates needed with respect to our settings:

Table 1 Departure—Venice Airport, December 28, 2020

Flight	City	Time	I or D
0	Amsterdam	06:30	I
1	Rome Fiumicino	06:50	D
2	Barcelona	09:35	I
3	London Stansted	09:45	I
4	Istanbul Airport	10:15	I
5	Vienna	10:50	I
6	Bari	11:10	D
7	Zurich	11:30	I
8	Bruxelles	11:50	I
9	Bari	12:15	D
10	Amsterdam	12:20	I
11	Paris Charles de Gaulle	12:40	I
12	Palermo	12:50	D
13	Brindisi	13:20	D
14	Catania	13:35	D
15	Rome Fiumicino	15:25	D
16	Lisbona	16:10	I
17	Frankfurt	17:00	I
18	Lamezia Terme	17:30	D
19	Cagliari	17:40	D
20	Paris Charles de Gaulle	18:05	I
21	Naples	18:50	D
22	Madrid	19:25	I
23	London Gatwick	20:30	I
24	Catania	22:00	D

```
sage: inttol = [(3.30,6.30,0.2), (4.5,6.5,0.3), (6.35,9.35,0.2), (6.45,9.45,0.2),
(7.15,10.15,0.2), (7.50,10.50,0.2), (9.10,11.10,0.3), (8.30,11.30,0.2),
(8.50,11.50,0.2), (10.15,12.15,0.3),(9.20,12.20,0.2),(9.40,12.40,0.2),
(10.50,12.50,0.3),(11.20,13.20,0.3),(11.35,13.35,0.3),(13.25,15.25,0.3),
(13.10,16.10,0.2),(14,17,0.2),(15.30,17.30,0.3),(15.40,17.40,0.3),(15.05,18.05,0.2),
(16.50,18.50,0.3),(16.25,19.25,0.2),(17.30,20.30,0.2),(20,22,0.3)]
 sage: def schedule_flights(int_tol):
 ....:     g = graphs.ToleranceGraph(int_tol)
 ....:     from sage.graphs.graph_coloring import chromatic_number
 ....:     cronum = chromatic_number(g)
 ....:     from sage.graphs.graph_coloring import all_graph_colorings
 ....:     GraphColoring = all_graph_colorings(g, cronum)
 ....:     return [g, cronum, next(GraphColoring)]
 ....:
 sage: schedule_flights(inttol)
 [Graph on 25 vertices, 7,
  {0: [0, 2, 10, 16, 21, 24],
   1: [1, 3, 11, 15, 18, 23],
   2: [4, 9, 17],
   3: [5, 12, 19],
   4: [6, 13, 20],
   5: [7, 14, 22],
   6: [8]}]
```

The program output gives us the information that the minimum number of gates that we need is 7. Furthermore, we got the following solution: the gate 0 is used for flights 0, 2, 10, 16, 21, and 24; the gate 1 is used for flights 1, 3, 11, 15, 18, and 23; the gate 2 is used for flights 4, 9, and 17; the gate 3 is used for flights 5, 12, and 19; the gate 4 is used for flights 6, 13, and 20; the gate 4 is used for flights 7, 14, and 22; and the gate 6 is used for the flight 8.

Acknowledgements The authors would like to thank the anonymous referees for helpful suggestions that improved the paper.

Author Contributions This is a joint collaboration with both authors contributing substantially throughout.

Funding This work has been fully supported by Croatian Science Foundation under the project 6732.

Compliance with ethical standards

Conflict of interest The authors declare that there is no conflict of interest.

References

1. Zhu H, Li Y, Jin X, Huang J, Liu X, Qian Y, Tan J. Transmission dynamics and control methodology of COVID-19: a modeling study. Appl Math Model. 2021;89:1983–98.
2. Postavaru O, Anton SR, Toma A. COVID-19 pandemic and chaos theory. Math Comput Simul. 2021;181:138–49.
3. Diestel R. Graph Theory. New York: Springer; 2012.
4. Golumbic MC, Monma CL. A generalization of interval graphs with tolerances. In: Proceedings of the thirteenth Southeastern conference on combinatorics, graph theory and computing (Boca Raton, Fla., 1982), Congr. Numer., vol., 35; 1982. pp. 321–31.
5. Golumbic MC, Trenk AN. Tolerance graphs. Cambridge: Cambridge University Press; 2004.
6. Golumbic MC, Jamison RE. Rank-tolerance graph classes. J Graph Theory. 2006;52:317–40.
7. Lovász L. A characterization of perfect graphs. J Combin Theory B. 1972;13:95–8.
8. Chudnovsky M, Robertson N, Seymour P, Thomas R. The strong perfect graph theorem. Ann Math. 2006;164:51–229.
9. Karp RM. Reducibility among combinatorial problems. In: Miller RE, Thatcher JW, editors. Complexity of computer computations. New York: Plenum Press; 1972. p. 85–103.
10. Grötschel M, Lovász L, Schrijver A. Polynomial algorithms for perfect graphs. Ann Discr Math. 1984;21:325–56.
11. Jungnickel D. Graphs, networks and algorithms. 2nd ed. Berlin, Heidelberg, New York: Springer; 2005.
12. Python Software Foundation. Python Language Reference, version 3.9. 2020. http://www.python.org.
13. Developers TS. Sage Mathematics Software (version 7.3). 2016. http://www.sagemath.org
14. Niskanen S,Östergård PRJ. Cliquer user's guide, version 1.0, Communications Laboratory, Helsinki University of Technology, Espoo, Finland, Tech.Rep. T48; 2003.