DIMACS Technical Report 2005-02
January 2005

Integer Programming Solutions for Several Optimization Problems in Graph Theory

by

Jamiru Luttamaguzi[1,2]
Dept of Math. and Computer Science
Elizabeth City State Univ.
jluttamaguzi@mail.ecsu.edu

Michael Pelsmajer
Dept of Applied Mathematics
Illinois Institute Technology
pelsmajer@iit.edu

Zhizhang Shen
Dept of Computer Science
Plymouth State Univ.
zshen@mail.plymouth.edu

Boting Yang
Dept of Computer Science
University of Regina
boting@cs.uregina.ca

[1]This report is based on the work originated during the DIMACS Reconnect Satellite Conference 2004 at Lafayette College on "Experimental Algorithmics, with a Focus on Branch and Bound for Discrete Optimization Problems", June 20 - June 26, 2004. A paper based on this report has been accepted by the 2005 International Conference on Computers and Their Applications (CATA2005) for presentation and publication in its proceedings.
[2]The authors would like to thank both Cynthia Phillips of Sandia National Laboratories, and Jeffrey Linderoth of Lehigh University for their help. The authors also wish to thank Dr. Jonathan Berry of Sandia National Laboratories for facilitating the conference.

DIMACS is a collaborative project of Rutgers University, Princeton University, AT&T Labs–Research, Bell Labs, NEC Laboratories America and Telcordia Technologies, as well as affiliate members Avaya Labs, HP Labs, IBM Research and Microsoft Research. DIMACS was founded as an NSF Science and Technology Center.

ABSTRACT

In this paper, we discuss how to solve several graph theory based optimization problems by following an *integer programming* approach. In particular, we discuss such problems as the space-filling problem, the bandwidth problem, and the cut-width problem. We present respective IP formulations, and carry out some preliminary output analysis.

1 Introduction

Non-trivial real-life discrete optimization problems are often involved with a large quantity of variables, thus lead to a huge amount of computation. Since it is quite important to solve such problems for the benefit of our lives and the society as a whole, a tremendous amount of effort has been spent over the years in developing the associated computer hardware and software technology, and as a result, many optimization solvers based on linear and integer programming techniques [14], such as CPlex [4], Minto[12], PICO [22], etc., have come to life.

When solving a problem by applying the linear programming (LP) technique, or its more constrained cousin of integer programming (IP), where all the involved variables have to be integers, as a first step, we need to provide an LP (IP) formulation of the problem, consisting of an objective function that needs to be either maximized or minimized, together with the definition of the variables used within, as well as a set of the associated constraints. Once such a formulation is completed, its *LP relaxation,* namely the problem minus the integral constraint(s), is sent to an LP solver for a solution and such technology as branch-bound and cutting planes will then be applied to seek an IP solution [14]. In general, the LP solver, often based on a revised version of the famous Simplex algorithm [14, §1.2], is computationally effective, thus does not pose a technical challenge. On the other hand, since many of the IP problems are NP-hard[6], there does not exist a general effective procedure for solving IP problems, but many research results, as well as applicable techniques, exist for solving general IP problems, and some specific ones.

In this paper, we explore the likelihood of solving several graph theory based optimization problems by following an *integer programming (IP)* approach. After a "mild" introduction of the integer programming techniques with a few basic problems, we will discuss several related graph theory oriented problems. For each of them, we will formulate its IP representation, and present solutions obtained via an IP solver.

2 Basic IP formulations

In this section, we introduce a few integer programming formulation techniques and apply them to solve several rather basic, but often encounted, problems.

2.1 Permutation in integer programming

When solving graph theory related problems, we often have to relabel the vertices of the involved graphs. More specifically, given a graph $G(=(V,E))$, we have to find out a rearrangement for the ordered set of $\{v_1, v_2, \ldots, v_n\}$, where $|V| = n$. Technically, we are looking for a permutation of the ordered set of $\{1, 2, \ldots, n\}$, which can be formulated in IP, as follows:

Let x_i^k, $i, k \in [0, n-1]$, be binary decision variables that specify the fact that i is mapped to the position k in the permutation. That is, for all $i, k \in [0, n-1], x_i^k \in \{0, 1\}$, $x_i^k = 1$ iff $i \mapsto k$. To characterize the 1-1 correspondence nature of such a mapping, we add in the following constraints: for all $i \in [0, n-1]$, $\sum_{k\in[0,n-1]} x_i^k = 1$. This equality specifies that any specific member maps to exactly one position in the linear order. Similarly, we apply the following constraint to ensure the injectivity and surjectivity of this mapping: for all $k \in [0, n-1]$, $\sum_{i\in[0,n-1]} x_i^k = 1$.

In the rest of the paper, we refer to the binary variables, $x_i^k, i, k \in [0, n-1]$, together with the set of the constraints, $\sum_{k\in[0,n-1]} x_i^k = 1$ and $\sum_{i\in[0,n-1]} x_i^k = 1$, as the $Permutation(x, i, k, n)$ constraint.

2.2 Logic operations in integer programming

In formulating linear programming expressions, we often need to express logic operations. Some of these operations can be easily converted to integer programming statements. For example, let x and z be two binary variables, such that $z = \neg x$. Then based on the relationship $z = 1 - x$, we immediately have the following two constraints: $z \leq 1 - x$ and $z \geq 1 - x$.

We also need to express other logic operations: For example, when stating the fact that $i \mapsto k$ and $j \mapsto l$, if we use a binary variable x_i^k to mean that $i \mapsto k$, the above requirement is equivalent to the logic conjunction of x_i^k and x_j^l, namely, $x_i^k \wedge x_j^l$. However, the latter expression is not a linear expression, thus not allowed in linear programming. We now discuss a technique that linearizes a logic conjunction. Let x, y, z be binary decision variables such that $z = x \wedge y$. To ensure the conjunctive property that $x = y = 1$ if $z = 1$, we add in two constraints: $z \leq x$ and $z \leq y$. Indeed, if either x or y equals to 0, then the above pair of constraints require that z be 0 as well. On the other hand, if $z = 0$, under the above two constraints, both x and y could be 1. Hence, to ensure the other half of the equivalency, namely, if both x and y equal to 1, then so is z, we add on the constraint: $x + y \leq z + 1$. Under this constraint, if $z = 1$, then x and/or y could be 0. Hence, it does take all the three constraints to faithfully characterize the logic conjuction operation.

In the rest of the paper, we refer to the constraints, $z \leq x$, $z \leq y$ and $x + y \leq z + 1$, with parameters x, y, and z, as the $Conjunction(x, y; z)$ constraint.

For the logic NOR operation $z = \neg(x \vee y)$, we can use the following constraints to express it: for binary variables x, y and z, $x + z \leq 1$, $y + z \leq 1$ and $x + y + z \geq 1$. We later refer to these three constraints with parameters x, y, and z, as the $NOR(x, y; z)$ constraint.

3 The space filling problem

We sometimes need to linearly order the nodes of a two-dimensional (2D) mesh structure such that the *clustering* property of the mesh nodes is preserved in the linear order as much as possible. For example, when assigning working processes with a mesh structure to a set of processors, to achieve load balancing and minimize interprocessor communication, the first step is to linearly order all the mesh nodes such that processes adjacent in the mesh structure are mapped to close positions in the linear order. This problem is often referred to as the *space filling problem* in the literature [2]. The opposite direction of such a mapping, namely mapping a list of working processes to a 2D mesh structure, referred to as "inverse space filling", is also found useful [23, 15].

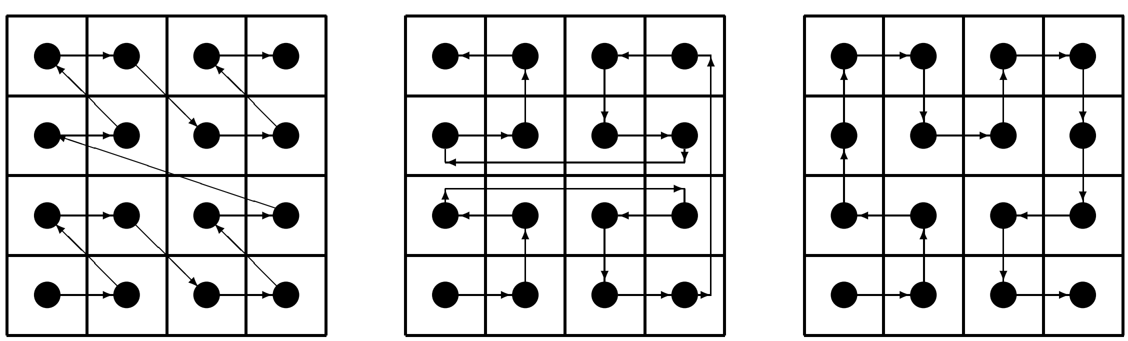
Another application of such a linear order is found in design access methods for multi-dimensional database, where a database is interpreted as a multi-dimensional space, with each attribute specifying a dimension. Components of such a space is then partitioned and kept in disk pages. It is observed that as the distance between two tuples in such a space decreases, the chance increases that they will get retrieved together by a range query, which tries to collect all tuples $(A_1, A_2, \cdots, A_n)$, such that for all $i \in [1, n], l_i \leq A_i \leq u_i$, where $l_i(u_i)$ is the lower(upper) bound specified for the A_i attribute. Thus, it makes sense to keep those neighboring data within proximity in such a partition [18].

Other applications in such fields of image compression, geographic information systems, graph drawing, scientific visualization, VLSI layout, etc., are also discussed in [21, §1] and [13].

Traditionally, such a mapping from a multi-dimensional space to a linear order is called a *space filling curve* [25]. Although Peano is the first to suggest such a curve in the continuous sense, which fills the 2D plane without "holes" [1] [20], Hilbert's elegant design [9], its discrete variants to be exact[2], is often used in implementing such a linear order of all

the nodes in a 2D mesh due to its attractive clustering preservation property [13].

Several such curves have been widely applied in practice, including the z-ordering as proposed in [1] and [18, §3.2], the Gray curve based on the Gray code as proposed in [5], and the $H-$curve as suggested in [15]. Many other such curves have been collected and discussed in [24]. We demonstrate some of the aforementioned space filling curves in the following Figure 1.



z-curve Gray curve Hilbert curve

Figure 1: Various space filling curves

To provide a general context, we define the general *space filling problem* as follows: To

[1]Technically, such a curve continuously maps the closed unit interval $I = [0, 1]$ into the closed unit square $S = [0, 1]^2$.

[2]Indeed, the original continuous Hilbert curve is defined to be the limit of a family of discrete space filling curves. The one shown in Figure 1 is an instance for a 4×4 mesh.

linearly order the vertex set of a general graph, (V, E), such that a pre-specified metric for adjacent vertices[3] is minimized as much as possible.

To place the space-filling problem for a 2D mesh in this general context, we relabel the vertices via the well-known row-major ordering, r. In particular, for all $i \in [0, m-1], j \in [0, n-1]$, we let

$$r(i,j) \;\; = \;\; i * n + j.$$

Notice that this mapping itself immediately linearly orders the mesh nodes, but it is known [15] that this order does not preserve proximity well, as compared with, e.g., the Hilbert curve. Thus, we use it to merely provide a linear labeling of all the nodes in a 2D mesh, so that we can represent this specific structure as a general graph (V, E), such that $V = \{v_0, v_1, \cdots, v_{m*n-1}\}$, together with the adjacency relation for the vertices in the associated grid graph.

If we follow the aforementioned row major ordering, starting with the top-left corner, the z-curve, the Gray curve and the Hilbert curve correspond, respectively to $(12, 13, 8, 9, 14, 15, 10, 11, 4, 5, 0, 1, 6, 7, 2, 3)$[4], $(12, 13, 9, 8, 11, 10, 14, 15, 3, 2, 6, 7, 4, 5, 1, 0)$, and $(12, 13, 9, 8, 4, 0, 1, 5, 6, 2, 3, 7, 11, 10, 14, 15)$.

Although analytical results have been derived for some of the specific graphs, mainly the multi-dimensional grids with finite granularity [13, 17], no formal proof of the optimality of any of the aforementioned curves for *general graphs* exists, in terms of best preserving various clustering property. Thus, it is still interesting to explore other mappings that might lead to the enforcement of the desired locality property, particularly for the case of general graphs. In this paper, we explore an alternative approach to solving this problem: *formulate an IP expression of this problem, and let the computers, IP solvers to be exact, to come up with such curves according to pre-specified proximity metric.*

Indeed, numerous definitions of such clustering proximity have appeared in literature. For example, in [15], given any two distinct mapped positions, i and j, in a space-filling curve, such metrics as the Manhattan distance, the Euclidean distance and the maximum metric, are used to study the proximity of i and j in a 2D mesh. For example, with respect to the Manhattan distance, $d(i, j)$, for any two mapped positions i and j in the Hilbert curve, it is shown in [15, Theorem 3] that $d(i, j) \le 3\sqrt{|i-j|} - 2$.

Another rather global proximity measurement of the average number of clusters relative to a subspace is suggested in [13], where a *cluster* is defined as *a collection of vertices inside the subspace that are mapped consecutively under a space filling curve.* This measurement is deemed important since it determines the number of non-consecutive disk accesses, when associated with multi-dimensional database applications. It is shown [13, §6], e.g., that the number of such clusters is a quarter of the perimeter of a query rectangle for the Hilbert curve; while that of the z-curve is more than a third.

[3] Incidentally, we can certainly further generalize this problem by introducing another parameter, α, so that the problem will address the locality property for all the α-adjacent vertices, namely those vertices whose distance, in relation to a certain metric, is α or less.

[4] This sequence represents the mapping of $12 \mapsto 0, 13 \mapsto 1, \cdots, 3 \mapsto 15$.

We now suggest another such measurement of a global flavor: Given a graph $G(=(V,E))$. Let $v_i, v_j \in V$, $(v_i, v_j) \in E$, and let f specify a permutation of the ordered set $\{0, \cdots, |V|-1\}$. It is clear that the less $|f(v_i) - f(v_j)|$ is, the closer v_i and v_j are under the space filling curve as f specifies. Therefore, the space-filling curve with a smaller sum of all such difference intuitively preserves the proximity of the adjacent vertices better in a global sense.

We notice that the sum of the difference of the mapped position of the adjacent vertices corresponding to the z-curve, Gray curve, the $H-$curve, and the Hilbert curve is, 60, 72, 74, and 64, respectively. This explains why it is pointed out in [18, §3.2] that if two vertices are close in the 2D mesh, then they are *likely* to be close in the $z-$order.

Thus, relative to this measurement, the general space filling problem can be described as follows: *Given a graph $G = (V,E)$, how to find a linear ordering of V, i.e., a one-to-one function $f : V \rightarrow \{0, 1, ..., |V| - 1\}$, such that for all the adjacent vertices, the sum of the difference of their mapped positions in such an order is minimized.*

It is clear that, under this specific proximity measurement, the general space filling problem turns into the well known *minimum linear arrangement problem,* namely, the problem number GT42, as listed in [6]. This problem is known to be NP-hard, and the associated decision problem is NP-complete, even for a bipartite graph.

We now start to formulate an integer programming expression for this problem. Let x_i^k be binary decision variables such that for all $i, k \in [0, |V|-1]$, $x_i^k = 1$ iff $v_i \mapsto k$; and, for all $i, j, k, l \in [0, |V|-1]$, let $y_{i,j}^{k,l}$ be binary decision variables such that $y_{i,j}^{l,k} = x_i^k \wedge x_j^l$.

Since, for all the adjacent pair of vertices in V, we want to minimize the sum of the difference between their positions in a linear order, we have the following objective function:

$$\min \sum_{0 \leq i,j,k,l < |V|, (v_i, v_j) \in E} |k - l| y_{i,j}^{k,l},$$

subject to, for all $i, k \in [0, |V|-1]$, the $Permutation(x, i, k, |V|)$ constraint and, for all $i, j, k, l \in [0, |V|-1]$, the $Conjunction(x_i^k, x_j^l; y_{i,j}^{k,l})$ constraint, as discussed in Section 2.

We have run the resulted program on several sample meshes with the branch-bound based MINLP solver made available in the NEOS IP solver site [16], a $2 \times 2, 2 \times 3, 3 \times 3,$ and 4×4 mesh. The solver sent back optimal solutions for the first two cases, but only sent back feasible solutions for the latter two cases due to the restriction on the stack space. We collect these results in Figure 2.

We observe the result for the $2{\times}2$ mesh coincides with the Hilbert curve, and the result for the 2×3 mesh is a mixture of the Hilbert curve and the z-curve, while these two standard curves are only defined for $2^d \times 2^d$ meshes. On the other hand, the *feasible solution*[5] obtained via an unknown heuristics by the IP solver for the case of 4×4 mesh corresponds to the following sequence: $(3, 2, 7, 6, 1, 5, 4, 0, 8, 9, 12, 13, 10, 11, 14, 15)$. It is easy to calculate the sum of the difference of the mapped position of the adjacent vertices according to this sequence is 61, that is %1.67 more than that associated with the z-curve, but %4.9 less than that associated with the Hilbert curve, %18 less than this sum associated with the Gray

[5]The optimal solution will certainly be at least as good as this feasible solution.

curve, and %21.3 less compared with the $H-$curve. Thus, this feasible solution obtained via the IP approach is (much) better than some of the standard curves.

4 The bandwidth problem

The *bandwidth problem* tries to label the vertices of a given graph G such that the maximum difference between adjacent vertices is minimized. More specifically: *Given a graph $G = (V, E)$, how to find a linear ordering of V, i.e., a one-to-one function $f : V \to \{0, 1, \ldots, |V|-1\}$, such that the bandwidth of the ordering, i.e.,* $\max_{(v_i,v_j)\in E} |f(v_i) - f(v_j)|$*, is minimized?*

Bandwidth minimization has many applications by itself. For example, in VLSI design, sometimes we want to arrange a set of circuit components in a line on a circuit board such that the length of the longest wire (and hence time delay) is minimum. We can place this problem in the context of the bandwidth problem by representing each circuit component by a vertex and every wire connecting two components by an edge. Another application is storing large objects, such as images, on a storage device. From each image there is a set of images we can go to next via, e.g., the hyperlinks. In order to minimize the time for searching these images, we should arrange these linked images so that they are not far away from each other on the device.

It is well known that the decision problem associated with the bandwidth minimization problem again is NP-complete [6, 19]. Surprisingly, it remains NP-complete even if the input graph is a tree with a maximum vertex degree of 3 [7], a very strong condition indeed. Blache, Karpinski and Wirtgen [3] showed that there is no polynomial time approximation scheme (PTAS) for the bandwidth problem unless $P = NP$, even for trees. They also showed that there are no polynomial time approximation algorithms for general graphs with an approximation ratio better than 1.25, and for trees with an approximation ratio better than $7/6(\approx 1.167)$. Karpinski, Wirtgen and Zelikovsky [11] presented an approximation algorithm for dense graphs with ratio 3 if every vertex has degree $\Theta(|V|)$. Haralambides, Makedon and Monien [8] gave an approximation algorithm for caterpillars with ratio $O(\log |V|)$. Kloks,

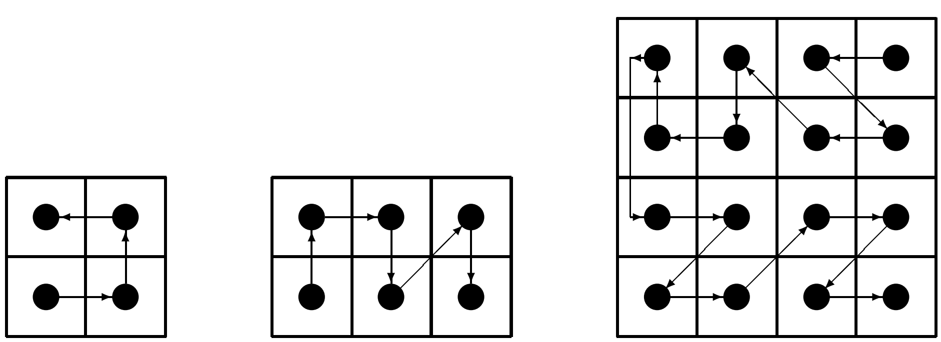


Figure 2: IP results

Kratsch and Müller [10] gave an approximation algorithm for asteroidal triple-free graphs with a ratio of 2.

It is clear that the linear programming expression for this bandwidth problem differs from that of the space-filling problem only in the objective function. In fact, the objective function for the bandwidth problem can be described as $\min_{0 \leq i,j,k,l < |V|, (v_i,v_j) \in E} \max(|k-l| y_{i,j}^{k,l})$.

However, linear programming does not handle the minimax problem well. Thus, we convert the above objective function into the following standard IP problem by minimizing the ceiling of the absolute value of all such differences, denoted by an integer variable S : $\min S$, subject to the constraint: for all $0 \leq i,j,k,l < |V|$ and $(v_i, v_j) \in E$, $|k-l| y_{i,j}^{k,l} \leq S$.

We certainly also need to add in such constraints as $Permutation(x, i, k, |V|)$ for all $i, k \in [0, |V|-1]$, and $Conjunction(x_i^k, x_j^l; y_{i,j}^{k,l})$, for all $i, j, k, l \in [0, |V|-1]$, together with variables $x_i^k, i, k \in [0, |V|-1]$, and $y_{i,j}^{l,k}, i, j, l, k \in [0, |V|-1]$, as defined in the previous section.

We similarly coded the above formulation into an executable program, and ran it on several sample graphs with the MINLP solver. For the 2×3 mesh, we obtained the optimal solution for this problem: $(2, 5, 1, 4, 0, 3)$, with its maximum difference between the mapped position of any two adjacent vertices being 2. For the 4×4 mesh, we obtained a feasible solution, due to an stack overflow error[6]: $(0, 4, 15, 6, 2, 8, 1, 11, 3, 7, 14, 5, 12, 13, 9)$, with the aforementioned maximum difference being 10. We notice that this quantity for the z-curve, the Gray curve, the $H-$curve, and the Hilbert curve is, 6, 11, 11, and 15, respectively. Thus, again, we find a curve with a smaller objective function value compared with almost all of the above standard curves.

We also ran this bandwidth program on the well-known Peterson graph, and obtained a feasible solution which gives the maximum difference between the mapped positions of any adjacent vertices in this mapping of 6. We notice that if we don't permute the input at all, i.e., letting $0 \mapsto 0, 1 \mapsto 1, \cdots, 9 \mapsto 9$, we would have a result of 5. After enforcing $0 \mapsto 0$,we did obtain a feasible solution $(0, 1, 6, 8, 4, 5, 2, 9, 3, 7)$, which gives a minimum bandwidth of 5. Needless to say, neither of the proposed curves is defined for such general graphs as the Peterson graph.

Incidentally, we suspect that one of the reasons that we ran into many stack overflow errors is because the graphs we have been working with are highly symmetrical, which leads to much unnecessary work on the solver part. One way to break such symmetry is indeed coercing a few of such mappings. The question as what variables to be initialized, and to what values, certainly waits for an answer through future research.

5 The cut-width problem

The goal of the load balancing problem is that, besides preserving the locality property, once we cut the number of vertices evenly among k processors, thus balancing their load, we want to minimize the inter-processor communication among those processors. To start with, we

[6]The default stack maximum size is set at 5,000.

might want to cut down the connection between those non-adjacent vertices. As an example, if we consider $M(2,2)$, if we follow the classic Hilbert space filling curve, by assigning the following mapping, $(2,3,1,0)$, then assigning the first two processes to the first processor, and the other two to the second processor, to achieve an apparent load balancing. We will have one "cross edge" between two non-adjacent vertices, i.e., 2 and 0. On the other hand, if we give the mapping $(2,1,3,0)$, there will be three such cross edges instead, namely, between 2 and 3, 0 and 2, and between 0 and 1.

This consideration leads to the *graph partition problem,* which asks for the minimization of the number of cross edges between non-adjacent nodes as a result of a $k(\geq 2)$ partition. As a special case, when we take $k = n$, we have the following *cut-width problem,* which is the problem of labeling the vertices of a given graph G such that the maximum number of cross edges is minimized. Formally, it can be described as follows: *Given a graph $G = (V, E)$, how to find a linear ordering of V, i.e., a one-to-one function $f : V \to \{0, 1, 2, ..., |V| - 1\}$, such that the cut-width of the ordering, i.e.,* $\max_{0\leq c\leq |V|-2} |\{(v_i, v_j) \in E : f(i) \leq c < f(j)\}|$, *is minimized?*

Once again, we have to identify a linear order for the vertices, and we are interested in the relationship between the positions of two adjacent vertices in such an order. Thus, the previously defined binary decision variables, x_i^k and $y_{i,j}^{k,l}$, together with the familiar *Permutation* and *Conjunction* constraints are handy. The strategy now, however, is to minimize the maximum number of cut edges in any vertex, c, where $c \in [0, |V| - 2]$. Since we face another minimax problem, we introduce the variable S into the objective function: $\min S$, subject to the following constraint: for $c \in [0, |V| - 2]$, $i, j \in [0, |V| - 1], (v_i, v_j) \in E$, $\sum_{(k\leq c<l)\vee(l\leq c<k)} y_{i,j}^{k,l} \leq S$

We have coded the above formulation into an executable program, and ran it with the MINLP solver on several sample graphs. For the 2×3 mesh, we obtained the optimal solution for this problem: $(1, 2, 5, 4, 0, 3)$, with the sought after minimum cut-width being 3, occuring at the mapped position of 1. For the 4×4 mesh, we obtained a feasible solution, again due to a stack overflow error: $(12, 13, 10, 9, 8, 4, 0, 5, 1, 6, 2, 3, 7, 11, 15, 14)$, with its minimum cut-width being 7, occuring at 9. We notice that the cut-width for the z-curve, the Gray curve,the $h-$curve, and the Hilbert curve is, 5, 6, 7, and 6, respectively, for the 4 $\times 4$ mesh.

6 Concluding remarks

In this paper, we explored the issue of following the integer programming approach to solve several graph related problems. We provided the IP formulation of some of the basic operations, as well as more general problems, and made analysis for the preliminary data that we have so far achieved. In particular, we demonstrated several techniques in formulating an IP expression for these problems, which we believe are also useful in expressing other problems.

For all the three problems we discussed in this paper, we have coded each of their IP formula into executable program, and ran it on several sample graphs with the MINLP solver made available in the NEOS IP solver site [16]. They works very well, but run very slowly. Because of the NP-hard nature of all four problems we have discussed, it is not a surprise at all that we only solved problems of small size. In fact, as pointed out in [21, §5], for the space filling problem, a mesh with size 5×5 should be considered large for the branch-and-bound algorithm. Compared with traditional approaches of applying analytical, or experimental, techniques to solve these NP-hard problems, the IP approach is preferable in some cases, since, as we demonstrated in numerous cases, the IP approach is general and flexible, it is relatively easy to formulate, and particularly appropriate when only a solution is needed.

To find optimal solutions to the larger instances of those problems, besides relying on the progress of computer technology, particularly, the improved ability of IP solvers, we also have to make these IP formulation more efficient. Among future works are the following:

- Investigate various problem specific cutting planes to add on more effective constraints to increase the instance size for various problems that can be effectively solved to optimality.

  For example, in the space filling problem, the corners are always mapped to either the first or the last positions. An conjecture is that this might always occur in some optimal solution. Indeed, if we could prove this is always the case, then we could add stronger constraints to exploit this structural property to further expedite the searching process.

- In the line of the work done in [21], investigate approximation algorithms for such problems as the bandwidth problem, the cut-width problem, etc., based on the requirement of finding a permutation for the vertices, a key property shared with the space filling problem.

- Compare the results we obtain via the IP approach with those already obtained via other approaches. For example, for the space-filling problem, many other patterns have been suggested for the mesh structures and other specific graphs [21, Table 1]. Although the data we have generated does not show any of the existing patterns, it seems that some of the patterns we have obtained is a mixture of the existing patterns, with the same or slightly higher objective values.

  Noticing that no lower bound of the objective value has been obtained, it is certainly interesting and important to carry out more experiments to see if a value lower than those associated with the existing patterns can be obtained.

- A significant advantage of the IP approach is that we can now explore the properties of general graphs. Thus, another line is to carry out experiments on general graphs to investigate, e.g., if similar patterns can be found on other topological structures.

References

[1] Abel, D., and Smith, J., A data structure and algorithm based on a linear key for a rectangle retrieval problem, *Computer Vision, Graphics and Image Processing,* 27(1) (1983), 19-31.

[2] Aluru, S., and Sevilgen, F., Parallel Domain Decomposition and Load Balancing Using Space-Filling Curves, in *Proc. of the Fourth International Conference on High-Performance Computing,* December 18 - 21, 1997 Bangalore, India, 230-235.

[3] Blache, G., Karpinski, M., and Wirtgen, J., On approximation intractability of the bandwidth problem, *Technical Report TR98-014, ECCC,* 1998.

[4] *CPlex,* http://www-fp.mcs.anl.gov/otc/Guide/SoftwareGuide/Blurbs/cplex.html.

[5] Faloutsos, C., Multiattribute hashing using gray codes, in *Proc. of ACM SIGMOD Conf.,* 227-238, 1986.

[6] Garey, M. R., and Johnson, D. S., *Computers and Intractability: A Guide to the theory of NP-completeness,* W. H. Freeman, San Francisco, 1979.

[7] Garey, M. R., Graham, D., Johnson, D. S., and Knuth, D., Complexity results for bandwidth minimization, *SIAM J. Appl. Math.,* 34:477-495, 1978.

[8] Haralambides, J., Makedon, F., and Monien, B., Bandwidth minimization: an approximation algorithm for caterpillars, *Math. Systems Theory 24,* 169-177, 1991.

[9] Hilbert, D., Uber Die Stetige Abbildung Einer Linie auf Flächenstück, *Math. Ann.* 38:459-450, 1891.

[10] Kloks, T., Kratsch, D., and Mller, H., Approximating the bandwidth for asteroidal triple-free graphs, in *Proc. 3rd Ann. European Symp. on Algorithms Lecture Notes in Comput. Sci. 979,* Springer-Verlag, 434-447, 1995.

[11] Karpinski, M., Wirtgen, J., and Zelikovsky, A., An approximation algorithm for the bandwidth problem on dense graphs, *Technical Report TR97-017, ECCC,* 1997.

[12] Savelsbergh, M., Lindroth, J., *Functional Description of MINTO, a Mixed INTeger Optimizer,* 2004.

[13] Moon, B., Jagadish, H. V., Faloutsos, C., Saltz, J., Analysis of the clustering properties of the Hilbert space filling curves, *IEEE Tran. on Knowledge and Data Engi.* 13(1):124-141, Jan/Feb 2001.

[14] Nemhauser, G., and Wolsey, L. A., *Integer and Combinatorial Optimization,* John Wiley & Sons, Inc., New York, 1999.

[15] Niedermeier, R., Erinhardtm, K, and Sanders, P., Towards optimal locality in mesh-indexings, in *Proc. of the Eleventh International Symposium on Fundamentals of Computation Theory,* Springer-Verlag, 364-375, 1997.

[16] *Neos Servers for Optimization,* http://www-neos.mcs.anl.gov/neos/.

[17] Orenstein, J., *Algorithms and data structures for the implementation of a relational database,* Ph.D. Thesis, McGill University, 1983.

[18] Orenstein, J., Spatial query processing in an object-oriented data base system, in *Proc. of ACM SIGMOD Conf.,* 326-336, 1986.

[19] C. Papadimitriou. The NP-completeness of the bandwidth minimization problem. Computing, 16:263-270, 1976.

[20] Peano, G. Sur une Courbe qui Remplit Toute une Aire Plane, *Math. Ann.,* 36:157160, 1890.,

[21] Pepit, J., Experiments on the minimum linear arrangement problem, *J. of Experimental Algorithmics* 8:(Article 2.3), Jan 2003.

[22] *PICO - Parallel Integer and Combinatorial Optimization,* http://www.cs.sandia.gov/ caphill/proj/pico.html.

[23] Philkington, J., Baden, S., Dynamic partitioning of non-uniform structured workloads with spacefilling curves, *IEEE Tran. on Parallel and Dist. Sys.* 7(3):288-300, March 1996.

[24] Sagan, H., *Space-Filling Curves,* Uniersitext. Springer-Verlag, 1994.

[25] Simmons, G., *Introduction to Topology and Modern Analysis,* McGraw-Hill Book Company, Inc., New York, 1963.