Math. Model. Nat. Phenom.
Vol. 6, No. 6, 2011, pp. 96-107
DOI: 10.1051/mmnp/20116606

Using DNA Self-assembly Design Strategies to Motivate Graph Theory Concepts

J. Ellis-Monaghan[1] * **and G. Pangborn**[2]

[1] Department of Mathematics, Saint Michael's College, Colchester, VT 05404
[2] Department of Computer Science, Saint Michael's College, Colchester, VT 05404

Abstract. A number of exciting new laboratory techniques have been developed using the Watson-Crick complementarity properties of DNA strands to achieve the self-assembly of graphical complexes. For all of these methods, an essential step in building the self-assembling nanostructure is designing the component molecular building blocks. These design strategy problems fall naturally into the realm of graph theory. We describe graph theoretical formalism for various construction methods, and then suggest several graph theory exercises to introduce this application into a standard undergraduate graph theory class. This application provides a natural framework for motivating central concepts such as degree sequence, Eulerian graphs, Fleury's algorithm, trees, graph genus, paths, cycles, etc. There are many open questions associated with these applications which are accessible to students and offer the possibility of exciting undergraduate research experiences in applied graph theory.

Key words: branched junction molecules, self-assembly, DNA complexes, graphs
AMS subject classification: C05, 92-01, 92E99

1. Introduction

Because of the great promise of nanotechnology, in particular for biomolecular computing, drug delivery and biosensors (see [1, 23, 13, 18]), recent research has focused on self-assembly of

*Corresponding author. E-mail: jellis-monaghan@smcvt.edu

nanoscale geometric constructs, notably graphs. A number of techniques have been developed using the Watson-Crick complementarity properties of DNA strands to achieve the self-assembly. Several different graphs have been constructed from self-assembling DNA molecules, including cubes [2], truncated octahedra [24], rigid octahedra [19], and more recently tetrahedra, dodecahedra, and buckyballs [9]. A 3D crystalline lattice has also been constructed [25]. Recent origami methods have resulted in DNA folding into 2D images and designs (see [16, 10]), and these methods are adaptable to 3D structures [3].

For all of these methods, an essential step in building the self-assembling nanostructure is designing the component molecular building blocks, and moreover determining where in the final structure they will appear. For the many structures being built that are essentially graphs, these design strategy problems fall naturally into the realm of graph theory.

We review graph theoretical formalism to support these applications, and then suggest several graph theory exercises to introduce these applications into a standard undergraduate graph theory class. The applications give a natural framework for motivating central concepts such as degree sequence, Eulerian graphs, Fleury's algorithm, trees, graph genus, paths, cycles, etc. The applications also lead naturally to discussions of well known graph theoretical problems such as the Chinese postman problem and the cycle double cover conjecture. There are many open questions associated with these applications which are accessible to students and offer the possibility of exciting undergraduate research experiences in applied graph theory.

2. Graph Theory Conventions

Our graph theory conventions are standard, following, for example, [22]. However, we quickly review a few graph theory concepts central to the discussions below. The applications we consider often involve *Eulerian graphs*, where the degree of every vertex is even. An *Eulerian digraph* has directed edges with equal indegree and outdegree at each vertex. A *walk* traverses consecutive edges in a graph (following the direction of the arrows in the case of a digraph) allowing repeated edges and vertices; a *trail* allows repeated vertices but not edges; and a *path* repeats neither. A *circuit* is a closed trail, and a *cycle* is a closed path.

A cellular embedding of a graph in an orientable surface is a representation of the graph as a subset of the sphere with n handles attached so that every face is a 2-cell. An embedding of a graph in a nonorientable surface is similarly the representation of the graph as a subset of the sphere with n crosscaps attached. The minimum (respectively maximum) *orientable genus* of a graph G is the smallest (respectively largest) n for which G has a cellular embedding. The minimum and maximum *nonorientable genus* are defined analogously. We can also consider the *ribbon graph* of the embedding: the surface with boundary that is an epsilon-neighborhood of the embedded G. The boundary cycles of the ribbon graph correspond bijectively to the faces of the embedded G.

3. Graph theoretical design strategies for branched junction molecule, or tile methods

One construction method for self-assembly of DNA structures uses k-armed branched junction molecules, star-shaped molecules whose centers form the vertices of the structure, and whose arms are double strands of DNA with one strand extending beyond the other. The longer strand forms a cohesive end at the end of the arm that can bond to any other cohesive end with complementary Watson-Crick bases, as in Figure 1, thus forming the edge of the graph. See [13] for a survey of current construction methods, and also [6, 15]. A target graph is constructed from these branched molecules, with a vertex of degree k realized by a k-armed branched molecule, and where two vertices in the target graph have an edge between them if and only if their two branched molecules have an arm joined between them via the complementary bases on the cohesive ends. Thus, the joined arms form the edges of the target graph. The resulting DNA complex is *complete* if it has no unmatched cohesive ends. In general (except for various lattices), we assume the final realization of a target graph is a complete complex.

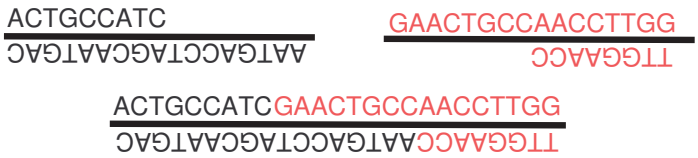

Figure 1: Joining two cohesive ends.

Following [11, 20], we call the combinatorial abstraction of the branched molecule a *tile*, and the abstraction of a cohesive end together with its complementary cohesive end a *bond-edge*. Cohesive ends are distinguished by *cohesive end types* (letter labels) such that a cohesive end labeled with an unhatted letter can adjoin to a cohesive end labeled with its complementary hatted label. For example, cohesive end types a and $\hat{a}$ would represent complementary sequences of bases, and so could form a bond-edge, as in Figure 1. More specifically, let $S = \{a, b, \ldots\}$ be a set of labels and let $\hat{S} = \{\hat{a}, \hat{b}, \ldots\}$ be a corresponding set of hatted labels. Combinatorially, a tile may be thought of as a vertex with half-edges labeled by elements of either S or $\hat{S}$, as in Figure 2. In the case of flexible armed molecule models, it is unnecessary to distinguish among permutations of cohesive end types about a vertex, while in geometrically constrained constructions, the relative positions of the labeled half-edges is a critical consideration.

A graph G is formed from a collection of tiles, called a *pot*, by identifying the half-edges in pairs, each pair forming an edge, subject to the restriction that a symbol is always paired with its hatted version and vice-versa. (See Figures 3 and 4.) This is equivalent to finding an edge-labeled orientation of the graph, with the arrows pointing from the unhatted to the hatted half edges making up the labeled edge, as in Figure 5. The pot constructing the graph in Figure 5 consists of two tiles: $t_1 = \{a, a, a, a\}$ and $t_2 = \{\hat{a}, \hat{a}, c, \hat{c}\}$.

We say that a pot *realizes* a graph G if a molecule with the same structure as G can be con-

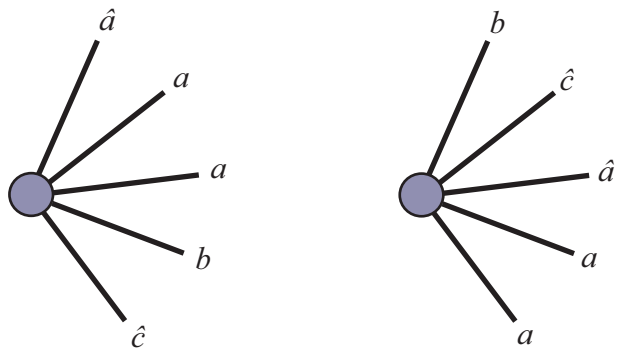

Figure 2: Two representations of the same tile type.

structed from branched junction molecules represented by the tile types listed in the pot. The combinatorial description of a pot lists each tile type exactly onces. However, in the chemical construction process, an arbitrarily large number of the branched junction molecules represented by each tile type are present. The (chemical) challenge is assuring that enough of these branched junction molecules attach to one another in the desired configuration for a reasonable yield of the targeted construct.

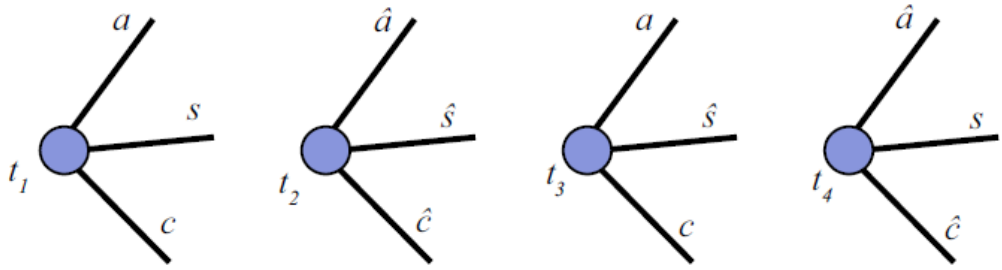

Figure 3: Four tiles comprising a pot P.

This leads to two new graph invariants and several new combinatorial problems for graphs. The branched junction molecules cost time, money, and effort to make. Thus, given a graph, we need to determine the minimum number of tile and bond-edge types required to construct a DNA complex realizing the graph. Furthermore, we must specify the combinatorial structure of the tiles in such a minimum set. This is equivalent to finding an edge-labeled orientation of the graph, using a minimal alphabet for the labels, with as few different labelings of the half edges about the vertices as possible, and furthermore specifying the set of different vertex types (the resulting tiles).

This problem is further confounded by the constraints of several different experimental settings. Because in the actual assembly process there are typically arbitrarily many tiles of each type present, a pot that realizes a graph G may realize many other graphs as well. In particular, if a pot realizes a graph G, then it could possibly realize any covering graph H of G. (A graph H covers a

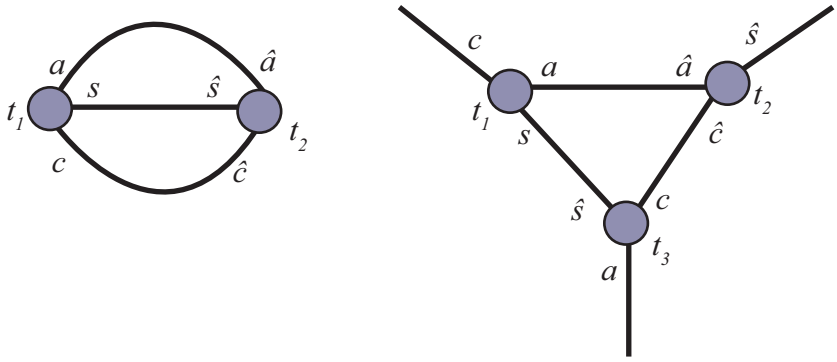

Figure 4: A complete complex and an incomplete complex, both formed from the pot P.

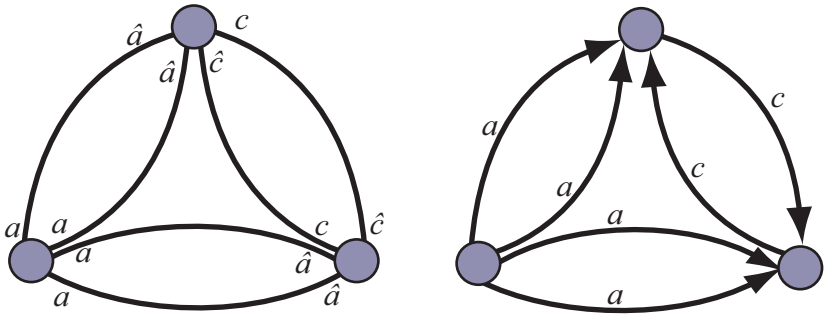

Figure 5: Edge labelings of a complete complex.

graph G if there is a surjective function from the vertices of H to the vertices of G that induces a surjection on the edges incident to every vertex.) For example, by identifying antipodal points we see that the cube covers K_4. Thus, any pot that realizes K_4 will also realize a cube. Fortunately, in practice this has not yet been a problem (it is hard enough to get the desired construct to self-assemble, let alone anything bigger). However, for some experiments it may be important for the pot not to realize anything smaller than the desired graph. Thus, if say a cube is desired, the pot must realize the cube, but not permit the incidental construction of K_4's.

Problem: Assuming flexible armed branched junction molecules, given a graph G, determine the minimum number of tile types and bond-edge types necessary to construct G under the following laboratory conditions, and furthermore find explicit pots realizing these minimums.

- *Scenario 1.* Allowing the possibility that graphical complexes of smaller size (that is, complexes representing graphs with fewer vertices) than the target graph could be created from the pot that builds the target graph;
- *Scenario 2.* Allowing the possibility that graphical complexes with the same number of vertices as, but not isomorphic to, the target graph could be created from the set of tiles that builds the target graph, but require that no smaller complexes can be built from the pot that builds the target graph;

- *Scenario 3.* Requiring that no complexes with less than or equal to the number of vertices of the target graph, except the target graph itself, can be built from the pot that builds the target graph.

These problems provide a good opportunity to introduce degrees sequences, Eulerian graphs, augmenting edges, Fleury's algorithm, cycles, trees, and chromatic numbers. The exercises below are drawn from work described in [5].

Exercise 1. *For scenario one, show that you can achieve an optimal solution for Eulerian graphs using one bond-edge type and a minimum number of tile types by using an Eulerian circuit construction. Use augmenting edges to show that this can be extended to a construction for a general graph, and give an upper bound on the number of tile types used. (If possible, give an example for which the upper bound is not attained.)*

Exercise 2. *Describe an optimal set of tiles for constructing a cycle of length n in scenario two. (The same pot will work for scenario three.)*

Exercise 3. *Show that the chromatic number of a graph provides a lower bound on the number of tile types needed in scenario three.*

For rigid armed tiles, the geometric setting introduces significant challenges. If a specific embedding of the graph in 3-space is desired (e.g. a specific polyhedra), or if the arms on the branched junction molecules are either short or reinforced for rigidity, then constraints such as restricted edge lengths and prescribed angles at the vertices must be built into the model.

Thus, for rigid armed tiles, the octet truss (see Figures 6 and 7) provides a reasonable geometric model. Currently, branched junction molecules with 2, 4, 5, 6, 8, and 12 arms have been fabricated (see [18] for an overview), so the 12-regular octet truss accurately reflects current capacity. All the edges are unit length, and the even spacing of the edges about the vertices prohibits any unattainable angles. However, this is a very challenging problem. To begin with, it is first necessary to determine what graphs are even subgraphs of the octet truss, and then it is necessary to find good drawings of them in it. The octet truss, being the only semi-regular tiling of 3-space is the next obvious 3D structure after the regular cubic lattice in which to draw graphs. There has been considerable work done for cubic lattice graph drawing (see e.g. [14]), but 3D orthogonal graph drawing was motivated by computer chip layouts, and attempts to minimize volume while allowing some edge bending, so is not directly applicable. Then, even assuming an embedding of the graph has been found, the minimum tile and bond-edge types must still be determined and a minimum pot constructed.

Because of these difficulties, it is reasonable to reverse the question here: instead of starting with a specific graph and trying to construct it, determine those graphs that can be most easily constructed within the geometric constraints of an octet truss.

This application provides a good opportunity to introduce geometric graph theory, skeletons of platonic solids, three dimensional tilings, rigidity, and related topics such as grid graphs. Related work is described in [8].

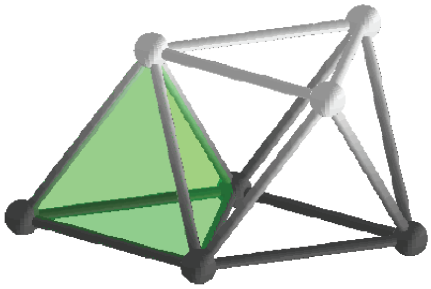

Figure 6: The basic space-filling tetrahedron/octahedron unit.

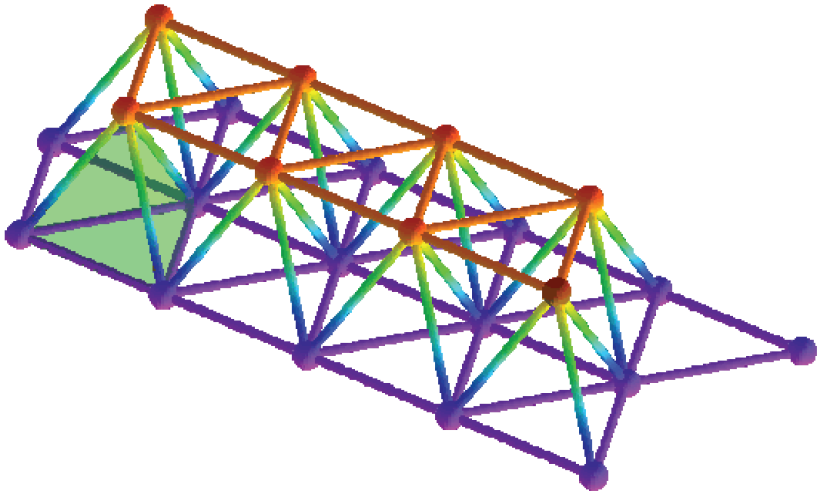

Figure 7: A portion of the octet truss.

Exercise 4. *Determine those Platonic solids naturally occurring in the octet truss, and find the minimum number of tile types necessary to construct them.*

4. Graph theoretical design strategies for linear strand methods

More recently, origami methods have emerged, wherein a single scaffolding strand of DNA traces the edges of the construct exactly once, and then short helper strands bond to this strand to fold and staple it into the desired configuration (see [16]). While originally applied to solid 2D structures, and later solid 3D structures, that are filled by the strands of DNA, this technique may also be adapted to graphical structures such as the wireframe beachball recently constructed by [3].

For origami constructions, questions focus on finding a threading of the graph or multigraph, that is, a walk for the scaffolding strand through the graph, ideally covering each edge exactly

once. Because a DNA strand is oriented from one end to another, if an edge must be covered more than once, the second time it is traversed, it must be traversed in the opposite direction from the first time it was traversed. The threading may be just an Eulerian circuit if the graph happens to be Eulerian. However, for non-Eulerian graphs, the problem of adapting either the graph or the circuit to enable the construction raises new mathematical questions. [12] gives a topology-based existence theorem for threadings of arbitrary graphs, but (necessarily) allows undesirable double-backs. Obviously, if the construction can be adapted by adding augmenting edges to make the graph Eulerian, a threading can easily be found. However, the problem becomes more challenging with the added constraints that the embedding of the graph in 3-space is often fixed ahead of time, and that, to the extent possible, the threading should follow faces and hence not cross itself at any vertex.

This application provides an opportunity to discuss algorithms for finding Euler circuits in Eulerian graphs, and augmenting edges in graphs with odd degree vertices. While finding a threading is not quite the Chinese postman problem (CPP) (in the CPP, an edge may be traversed in the same direction twice, which is not the case here), this application can provide a nice segue to the CPP. Because the octahedron and cube in the exercise below are 3D structures, this is also a good opportunity to introduce the Schlegel diagrams that give 2D drawings that are easier to work with.

Exercise 5. *Find a threading of an octahedron. Find one that does not cross itself, or prove that there are none. Find one that crosses itself at every vertex, or prove that there are none.*

Exercise 6. *Find a threading of a cube. Here it is not possible to cover each edge exactly once, so what is the minimum number of edges that must be doubled covered? Which edges might be good choices for being traversed more than once? (Remember that here, if an edge is traversed more than once, it must be traversed in two different directions.)*

Exercise 7. *Does a threading exist for any minimal set of doubled edges of the cube.* I.e. *first determine the minimum number of edges that must be covered twice by a threading, and then ask if any arbitrarily chosen set of edges of that size has a threading with just those edges traversed twice, once in each direction. Prove that such a threading always exists, or exhibit a minimal set of edges for which there is no threading.*

Exercise 8. *If augmenting edges are added to a cube, an Eulerian cycle provides a threading of the augmented graph. What might be a good set of augmenting edges? Find a threading for the cube with these augmenting edges. Again, is there a threading that does not cross itself?*

The earliest approach to self-assembly may be found in the techniques of [1, 24, 2], and, with modification, [19]. This method uses single strands of DNA to trace out each edge of a graph, once in each direction, binding to themselves along segments of Watson-Crick complements to form double helixes along the edges and stable branched junctions at the vertices. (See Figure 8.) The sequences of nucleotides are carefully specified so that only the two sides of an edge are complements of, and hence bind to, each other. As an expository convenience, we will refer to this method as *double tracing* because of the mathematics involved.

With the double tracing method, questions focus on determining ds and DS, respectively the minimum and maximum numbers of circular single strands needed to construct the graph with

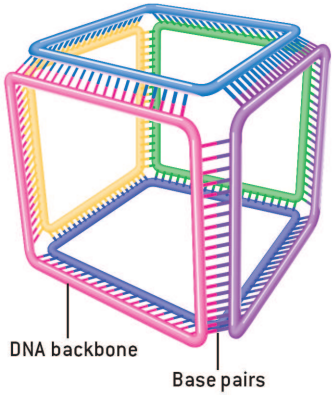

Figure 8: Schematic of cube from [17] (twisting of DNA strands not shown). This uses six circular strands, but as few as two are possible.

a detachment-free bidirectional double covering of the edges. (Detachment-free means the path cannot separate at a vertex, for example by doubling back at an edge. Figure 9 shows examples of forbidden detachments.) A *strand* may then be interpreted as a facial walk of a graph embedded in an orientable surface, or equivalently as the boundary of an orientable ribbon graph. It follows from the Euler polyhedral formula that any strand number for a graph may be given by $2 - V + E - 2g$, where V and E are the numbers of vertices and edges, respectively, of the graph, and g is the genus of an oriented surface into which the graph is embedded. Thus, if γ is the minimum genus of a graph, and Γ is the maximum genus, then $ds = 2 - V + E - 2\Gamma$ and $DS = 2 - V + E - 2\gamma$, so the minimum strand number corresponds to the maximum genus and vice versa.

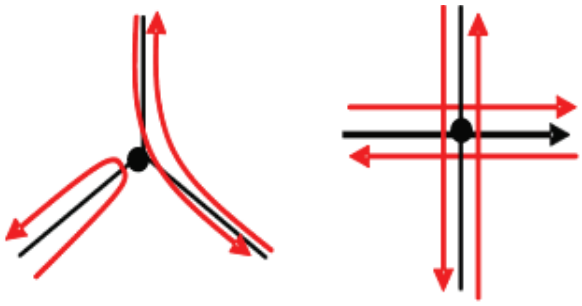

Figure 9: Forbidden tracing configurations causing detachment of the construct.

Both ds and DS are equally important: fewer strands means fewer to design, and perhaps fewer incomplete or incorrect constructs, hence higher yields, but shorter strands may be more stable, so a large number of shorter strands may be preferable for some constructions. For example, the octahedron of [19] uses only one long strand, albeit with brances, while the cube of [17] uses six. The relation between the strand numbers and genus means that for many important classes of graphs ds and DS are known, but in terms of the minimum and maximum genus. Genus questions can be quite difficult; for example, minimum genus was shown in [21] to be NP-hard in general.

Maximum genus is more tractable, with a general algorithm in [7] that finds the maximum genus of a graph with e edges, v vertices, and maximum degree d in $O(evd\log^6 v)$ time. However, the application requires not only the strand numbers, but also the intervals along each strand where it adheres to another.

While double covering of edges naturally suggests the cycle double cover conjecture (CDC), the double tracing construction method has both stronger and weaker requirements. In the CDC, a cycle may not repeat an edge, while in DNA constructs this is allowed. Also, orientation is not considered in the CDC, but a DNA strand has a natural orientation, and so the construction requires that each edge be covered once in each direction.

This construction method provides an interesting new application of graph genus, and a natural segue to the CDC. Related work can be found in [4].

Exercise 9. *Find the minimum and maximum strand numbers for the cube.*

Exercise 10. *The octahedron of [19] uses"1.7 kilobase single stranded DNA" in the construction. Use Euler's formula to show that this cannot be a linear strand that traces a single face, but must have some 'branches' or 'spurs' on it (it does).*

Acknowledgements

We thank the many students who have worked with us on projects related to self-assembing nanostructures including Laura Beaudin, Nick Bruno, Thomas Dickerson, Andrew Gilbert, Jacob Girard, Akie Hashimoto, Brian Hopper, Dan Lewis, Paul Jarvis, Dave Miller, Andrew Parent, and Mary Spuches.

Support for this work was provided by the National Science Foundation through award 1001408. This work has also been partially supported by the Vermont Space Grant Consortium through NASA grant number NNX10AK67H. Its contents are solely the responsibility of the authors and do not necessarily represent the official views of the NSF or NASA.

References

[1] L. Adleman. *Molecular computation to solutions of combinatorial problems*. Science, 266 (1994), 1021–1024.

[2] J. Chen, N. Seeman. *Synthesis from DNA of a molecule with the connectivity of a cube*. Nature, 350 (1991), 631–633.

[3] H. Dietz, S. Douglas, W. Shih. *Folding DNA into twisted and curved nanoscale shapes*. Science, 325 (2009), 725–730.

[4] J. Ellis-Monaghan. *Transition polynomials, double covers, and biomolecular computing*. Congressus Numerantium, 166 (2004), 181–192.

[5] J. Ellis-Monaghan, G. Pangborn, L. Beaudin, N. Bruno, A. Hashimoto, B. Hopper, P. Jarvis, D. Miller. *Minimal tile and bond-edge types for self-assembling DNA graphs*. Manuscript.

[6] K. Freedman, J. Lee, Y. Li, D. Luo, V. Sbokeleva, P. Ke. *Diffusion of single star-branched dendrimer-like DNA*. J. Phys. Chem. B, 109 (2005), 9839–9842.

[7] M. Furst, G. Gross, L. McGeoch. *Finding a maximum-genus graph imbedding*. J. Assoc. Comput. Mach., 35 (1988), 523–534.

[8] J. Girard, A. Gilbert, D. Lewis, M. Spuches. *Design optimization for DNA nanostructures*. American Journal of Undergraduate Research, 9, no. 4 (2011), 15–32.

[9] Y. He, T. Ye, M. Su, C. Zhuang, A. Ribbe, W. Jiang, C. Mao. *Hierarchical self-assembly of DNA into symmetric supramolecular polyhedra*. Nature, 452 (2008), 198–202.

[10] B. Hogberg, T. Liedl, W. Shih. *Folding DNA origami from a double-stranded source of scaffold*. J. Am. Chem. Soc., 131 (2009), 9154–9155.

[11] N. Jonoska, G. McColm, A. Staninska. *Spectrum of a pot for DNA complexes*. In DNA, 2006, 83–94.

[12] N. Jonoska, N. Seeman, G. Wu. *On existence of reporter strands in DNA-based graph structures*. Theoretical Computer Science, 410 (2009), 1448–1460.

[13] T. LaBean, H. Li. *Constructing novel materials with DNA*. Nano Today, 2 (2007), 26–35.

[14] B. Landfraf. *Drawing Graphs Methods and Models*. Springer-Verlag, 2001, ch. 3D Graph Drawing, 172–192.

[15] D. Luo. *The road from biology to materials*. Materials Today, 6 (2003), 38–43.

[16] P. Rothemund. *Folding DNA to create nanoscale shapes and patterns*. Nature, 440 (2006), 297–302.

[17] N. Seeman. *Nanotechnology and the double helix*. Scientific American, 290 (2004), 64–75.

[18] N. Seeman. *An overview of structural DNA nanotechnology*. Mol. Biotechnol., 37 (2007), 246–257.

[19] W. Shih, J. Quispe, G. Joyce. *A 1.7 kilobase single-stranded DNA that folds into a nanoscale octahedron*. Nature, 427 (2004), 618–621.

[20] A. Staninska. *The graph of a pot with DNA molecules*. in Proceedings of the 3rd annual conference on Foundations of Nanoscience (FNANO'06), April 2006, 222–226.

[21] C. Thomassen. *The graph genus problem is NP-complete*. J. Algorithms, 10 (1989), 568–576.

[22] D. West. *Introduction to Graph Theory*. Prentice-Hall, Englewood Cliffs, NJ, 2000.

[23] H. Yan, S. Park, G. Finkelstein, J. Reif, T. LaBean. *DNA-templated self-assembly of protein arrays and highly conductive nanowires*. Science, 301 (2003), 1882–1884.

[24] Y. Zhang, N. Seeman. *Construction of a DNA-truncated octahedron*. J. Am. Chem. Soc., 116 (1994), 1661–1669.

[25] J. Zheng, J. Birktoft, Y. Chen, T. Wang, R. Sha, P. Constantinou, S. Ginell, C. Mao, N. Seeman. *From molecular to macroscopic via the rational design of a self-assembled 3D DNA crystal*. Nature, 461 (2009), 74–77.