An Algorithm for Computing Simple k-Factors

Henk Meijer[a], Yurai Núñez-Rodríguez[b,*], David Rappaport[b,1]

[a]*Roosevelt Academy, NL-4330 AB Middelburg, The Netherlands*
[b]*School of Computing, Queen's University, Kingston, ON, Canada, K7L 3N6*

Abstract

A k-factor of graph G is defined as a k-regular spanning subgraph of G. For instance, a 2-factor of G is a set of cycles that span G. 2-factors have multiple applications in Graph Theory, Computer Graphics, and Computational Geometry [5, 4, 6, 14]. We define a simple 2-factor as a 2-factor without degenerate cycles. In general, simple k-factors are defined as k-regular spanning subgraphs where no edge is used more than once. We propose a new algorithm for computing simple k-factors for all values of $k \geq 2$.

Key words: graph algorithms, graph factors, k-factors, simple k-factors, 2-factors

1. Introduction

In his book "Algorithmic Graph Theory", Gibbons [5] defines a *k-factor* of a graph $G = (V, E)$ as a k-regular spanning subgraph of G. According to his definition, a 1-factor is a perfect matching and a 2-factor is a set of disjoint cycles that span G. He also proposes an algorithm to find a 2-factor based on an algorithm that finds a perfect matching of a bipartite graph. The algorithm produces a set of cycles that span the original graph; however, cycles may be *degenerate*, that is, cycles may include the same edge twice. For example, consider $V = \{v_1, v_2, v_3, v_4\}$, $E = \{(v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_1)\}$, a 2-factor of G is defined by $F = \{c_1, c_2\}$, where $c_1 = v_1, v_2, v_1$ and $c_2 = v_3, v_4, v_3$ (see Figure 1). In this example one edge is repeated twice in each cycle. This contradicts the original definition since a subgraph of a simple graph cannot be a multigraph. For the case of $k > 2$, one can also refer to a k-factor, as *degenerate* if it uses one edge multiple times.

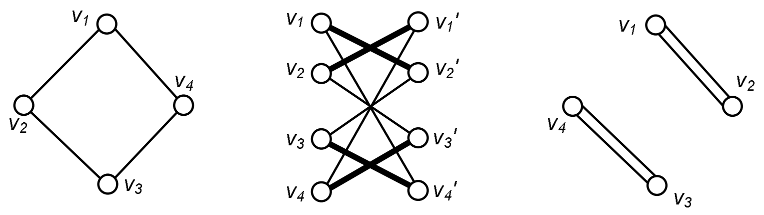

Figure 1: *Left:* Input graph. *Middle:* Auxiliary bipartite graph where the perfect matching is computed. *Right:* Resulting 2-factor. Degenerate cycles are marked with double lines.

Definition 1. *Consider $G = (V, E)$. A **simple k-factor** of G is a k-factor that is not degenerate.*

Notice that, according to the previous definition, both G and a k-factor of G, F, may include multiple edges (connecting the same pair of vertices) as long as F is a subgraph of G. However, no edge can be included more than once in F.

*Corresponding author
Email addresses: h.meijer@roac.nl (Henk Meijer), yurai@cs.queensu.ca (Yurai Núñez-Rodríguez), daver@cs.queensu.ca (David Rappaport)
[1]Supported by an NSERC of Canada Discovery Grant.

Simple k-factors, as opposed to general k-factors, do not only provide a more suitable definition for some problems; they are also more suitable for applications. In Computer Graphics, 2-factors are used in algorithms for fast rendering of 3D scenes. A 2-factor of the dual graph of a triangulated scene defines a (partial) linear order of the triangles involved. This allows for efficient *stripification* and compression of large sets of triangles as required for fast rendering [6]. Similarly, 2-factors can be used for stripification of large sets of quadrilaterals and tetrahedra [4]. In all these cases, it is desirable to obtain a small number of long cycles and, preferably, a Hamiltonian cycle. In this regard, degenerate cycles exhibit the worst case thus demonstrating the preference for simple 2-factors.

The idea of computing 2-factors for obtaining subgraphs as close as possible to Hamiltonian cycles has also been used on solid grid graphs [14]. Umans and Lenhart present an algorithm that by merging cycles of a 2-factor produces a Hamiltonian cycle, provided that the graph is Hamiltonian. Some of the elements used for their proofs are better defined if only simple cycles are considered. In fact, Umans [15] proposed two different algorithms for computing simple 2-factors. These algorithms will be reviewed in Section 3.

In the next section we give an overview of the 1-factor problem (i.e. the perfect matching problem) as it is fundamental for most 2-factor algorithms. In section 3 we review previous methods for computing 2-factors and simple 2-factors. Our approach to compute simple k-factors is presented in Section 4. We also provide a comparative analysis of our algorithm with respect to previous approaches for the simple 2-factor problem.

2. Preliminaries

The 2-factor algorithm presented by Gibbons [5] and other algorithms that produce simple 2-factors have one thing in common: they rely on maximum matching algorithms.

Our approach to computing k-factors uses the maximum matching algorithm by Micali and Vazirani [8, 11]. Given a graph $G = (V, E)$, with $|V| = n$ and $|E| = m$, their algorithm computes a maximum matching in $O(n^{1/2}m)$ time. This is not the most efficient algorithm for maximum matching in general graphs. Mucha and Sankowski [9] propose a randomized algorithm that runs in $O(n^{\omega})$ time, where $\omega < 2.38$ is the exponent in the complexity of the multiplication of $n \times n$ matrices. The latter is obviously more efficient if the graph is dense. Other algorithms are also more efficient than Micali and Vazirani's for special types of graphs. For example, Mucha and Sankowski [10] also propose an $O(n^{\omega/2})$ algorithm for planar graphs. Klein et al. [7] propose an $O(n^{4/3} \log n)$ algorithm for perfect matching in planar bipartite graphs. Biedl et al. [2] present an algorithm for perfect matching in 3-regular bridgeless graphs that runs in $O(n(\log n)^4)$ and $O(n)$ if the graph is also planar. The first of these two algorithms can be improved to $O(n \log^3 n \log \log n)$ using Thorup's data structure [13]. The reason we use Micali and Vazirani's algorithm is that the graphs to which we apply the maximum matching algorithm, as will be seen in Section 4, are not 3-regular, bipartite, planar, or dense, in general.

3. Previous work

To our knowledge, there are no published efficient algorithms for finding simple k-factors in general graphs. In their surveys, Akiyama and Kano [1], and more recently, Plummer [12], include multiple results describing classes of graphs for which k-factors exist. In some cases an algorithm that computes such a k-factor is also given. One such algorithm is due to Petersen. Petersen's result establishes the existence of simple $2k$-factors in $2m$-regular graphs, $m \geq k$, and his proof leads to the construction of a simple $2k$-factor.

Gopi and Eppstein [6] propose an algorithm to compute simple 2-factors of 3-regular graphs. Their algorithm computes a perfect matching of the input graph using the algorithm of Biedl et al., as mentioned above. The edges that are not in the computed matching define a simple 2-factor.

Diaz-Gutierrez and Gopi [4] present two different methods to compute simple 2-factors of graphs of maximum degree 4. The *two pass graph matching method* consists of computing a perfect matching on the input graph, removing the edges in the matching from the input graph, and computing a new matching on the remaining subgraph. The simple 2-factor is defined by the edges in the union of both perfect matchings. The authors acknowledge that their algorithm does not work on graphs with an odd number of vertices even when these are 4-regular. In fact, there are graphs with an even number of vertices where this algorithm also fails (see Figure 2). This suggests that algorithms for computing 2-factors need to compute all edges at once, as opposed to using multiple phases.

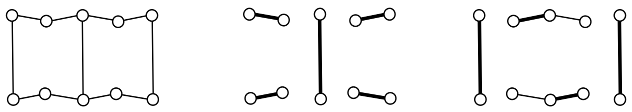
Figure 2: Example graph with an even number of vertices where the two pass matching method fails. *Left:* Input graph. *Middle:* Perfect matching obtained after the first pass. *Right:* Maximum matching (thick lines) obtained after removing the edges in the first pass matching from the input graph.

The second method by Diaz-Gutierrez and Gopi is called the *template substitution algorithm*. In this method, vertices of degree 4 are replaced by *templates*, as shown in Figure 3, to obtain an *inflated graph*. A perfect matching of the inflated graph can be translated into a simple 2-factor of the input graph given that exactly two *outside* vertices of a template connect to vertices of other templates. The authors claim that their templates can also replace vertices of degree less than 4; however, no details are provided.

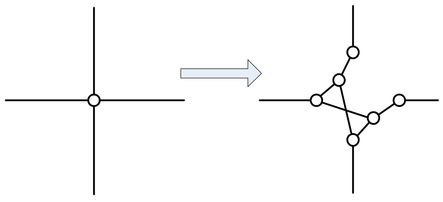
Figure 3: Template substitution as in the algorithm by Diaz-Gutierrez and Gopi [4].

Umans [15] proposes two algorithms for computing simple 2-factors of general graphs. The first algorithm is based on linear programming and, although simple to describe (as a set of linear equations), it is affected by the underlying complexity of linear programming (see Chvátal's book [3] for more details on the complexity of linear programming). The second approach consists of a set of transformations that produce a graph similar to the one obtained through the template substitution algorithm (see Figure 4 for an example). Thus, a 2-factor of the original graph is found via a perfect matching of the transformed graph.

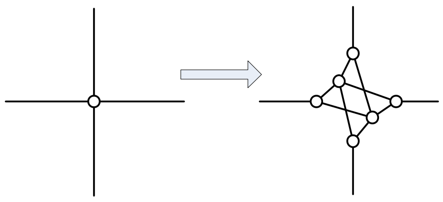
Figure 4: Transformation of a degree 4 vertex of the input graph according to Umans' algorithm [15].

4. An algorithm for general simple *k*-factors

In this section we present a new algorithm to compute simple *k*-factors on general graphs. Our technique also relies on matching algorithms. We use a set of *gadgets* that resemble the templates by Diaz-Gutierrez and Gopi and the transformations by Umans, but generalize to vertices of any degree and to *k*-factors for any possible *k* value.

The set of gadgets is defined as $\{\Gamma_{2,2}, \Gamma_{2,3}, \ldots, \Gamma_{3,3}, \Gamma_{3,4}, \ldots, \Gamma_{4,4}, \Gamma_{4,5}, \ldots\}$ as shown in Figure 5. These gadgets consist of *outer vertices*, *inner vertices*, *core vertices*, *outer edges*, *inner edges*, and *core edges*. A gadget $\Gamma_{k,d}$, $2 \leq k \leq d$, has d outer vertices, d inner vertices, and k core vertices. Core edges connect core vertices to inner vertices, inner edges connect inner vertices to outer vertices, and outer edges connect outer vertices of different gadget instances. Each inner vertex is adjacent to exactly one outer vertex, while each core vertex is adjacent to all inner vertices. Vertices within the same class are not adjacent to one another. In our algorithm (see Algorithm 1 below), a vertex v of degree d of the input graph G is replaced by a copy of the $\Gamma_{k,d}$ gadget, v'. The d outer edges of v' are connected to other outer vertices of the gadgets corresponding to the d neighbours of v. As a result, a new graph G' is obtained. We adopt the terminology of Diaz-Gutierrez and Gopi and call G' the *inflated graph* of G. Obviously, a simple k-factor cannot be found in graphs with minimum degree less than k. Thus, we assume that all vertices of G have degree k or more.

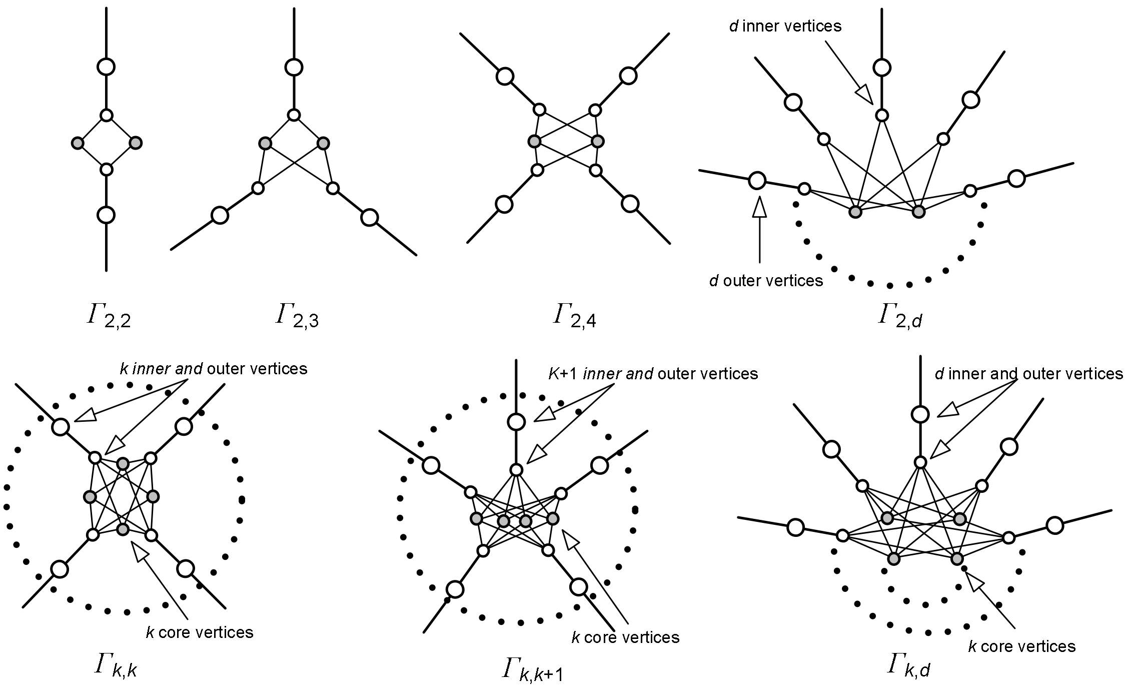

Figure 5: Gadgets used in Algorithm 1 to compute a simple k-factor. Instances of gadget $\Gamma_{k,d}$ replace degree d nodes of the input graph. Outer vertices are represented by larger circles, inner vertices by smaller circles, and core vertices by shaded circles.

Algorithm 1. *Computation of simple k-factors*
Input: *A general graph G*
Output: *A simple k-factor F, if one exists, or NULL otherwise*

```
1. If G has a vertex of degree less than k, stop and return NULL
2. Build the inflated graph G' from G by the process described above
3. Compute a maximum matching M of G'
4. If M is not a perfect matching, stop and return NULL
5. Initially define F as a graph that contains all the vertices of G and no edges
6. Add all the edges of G that correspond to the outer edges of M to F
7. return F
```

We provide the following lemma in order to prove that our algorithm correctly computes a simple k-factor of the input graph.

Lemma 1. *Let G be a graph with minimum degree at least k. G has a simple k-factor if and only if the corresponding inflated graph G' has a perfect matching.*

Proof. $\Longrightarrow$ Let F be a simple k-factor of G. Also let v be a vertex of G of degree d and let $v' \subseteq G'$ be the copy of $\Gamma_{k,d}$ that replaces v. Notice that the degree of v in F is k. A perfect matching of G' can be constructed such that the k outer edges of v' that correspond to the k matched edges incident to v in F are part of the matching. This leaves $d-k$ outer of v' unmatched. The $d-k$ unmatched outer vertices are then matched to their corresponding inner vertices. Finally, the k unmatched inner vertices are matched to the k core vertices, producing a perfect matching for G'.

$\Longleftarrow$ We prove that in a perfect matching M of G' each instance of a gadget $\Gamma_{k,d}$, $2 \leq k \leq d$, has k outer edges in the matching and, therefore, the corresponding edges in G define a simple k-factor. Recall that the gadgets $\Gamma_{k,d}$, contain k core vertices. The core vertices must be matched in M to k inner vertices. Therefore, the remaining $d-k$ inner vertices must be matched to their adjacent $d-k$ outer vertices, leaving exactly k outer vertices unmatched within v'. These have to be matched with other k vertices of other gadget instances through k distinct outer edges. □

The correctness of Algorithm 1 follows from Lemma 1. We state this explicitly in the following theorem.

Theorem 2. *Let G be a general graph. Algorithm 1 correctly computes a simple k-factor whenever the input graph has one.*

4.1. Analysis of the algorithm

The overall performance of our algorithm is based on the performance of the maximum matching algorithm of choice. In general, we use the algorithm by Micali and Vazirani [8, 11]. Let $G = (V, E)$ be the input graph, with $|V| = n$, and $G' = (V', E')$ be the inflated graph obtained from G. Notice that the computation of the maximum matching is the most expensive step of the algorithm. The remaining steps take at most linear time in the number of vertices or edges of the inflated graph. Thus the overall performance of our algorithm is $O(|V'|^{1/2}|E'|)$. In the following we analyze how $|V'|$ and $|E'|$ relate to n in order to express the complexity of the algorithm as a function of the input size.

Since the k-factor problem is easy to solve on complete graphs, we base our analysis on *near-complete* graphs. We define a near-complete graph to be a graph that is not complete but still has $O(n^2)$ edges and the degree of every vertex is $O(n)$. According to our algorithm, after substituting by Γ gadgets, $|V'|$ and $|E'|$ are $O(n^2)$ and $O(n^2k)$ respectively. This leads to an overall $O(n^3k)$ time complexity.

For graphs with vertex degree bounded by a constant D, the set of Γ gadgets leads to an algorithm that is as efficient as Micali and Vazirani's for maximum matching. In this case, the time complexity follows from $|V'| = O((D + k)n) = O(n)$ and $|E'| = O(Dkn + Dn) = O(n)$, resulting in $O(n^{1.5})$ overall.

When computing simple 2-factors of 3-regular graphs our algorithm produces an inflated graph with vertices of degree 2 and 3. Therefore, Petersen's Theorem (see [2]) does not apply and our algorithm is less efficient than Gopi and Eppstein's algorithm [6]: theirs is $O(n \log^3 n \log\log n)$ while ours is $O(n^{1.5})$.

4.2. Gadgets for large k-values

Obviously, the larger the value of k, the more complex the gadgets and therefore the slower the computation of k-factors. However, for large values of k ($k \geq d/2$), a more suitable set of gadgets exists. We define gadget $\Gamma'_{d-k,d}$ as a copy of $\Gamma_{d-k,d}$ that does not contain inner vertices or inner edges (See Figure 6). In the set of $\Gamma'_{d-k,d}$ gadgets, core edges directly connect core vertices to outer vertices.

In order to establish that Algorithm 1 also computes a simple k-factor using the $\Gamma'_{d-k,d}$ gadgets, we prove the following lemma.

Lemma 3. *Let G be a graph with minimum degree at least k. G has a simple k-factor if and only if the inflated graph G' obtained by using the $\Gamma'_{d-k,d}$ gadgets has a perfect matching.*

Proof. $\Longrightarrow$ Let F be a simple k-factor of G. Also, let v be a vertex of G of degree d and let $v' \subseteq G'$ be the copy of $\Gamma'_{d-k,d}$ that replaces v. Notice that the degree of v in F is k. A perfect matching of G' can be constructed such that the k outer edges of v' that correspond to the k matched edges incident to v in F are part of the matching. This leaves $d-k$ outer vertices and $d-k$ core vertices of G' unmatched. These are then pairwise matched.

$\Longleftarrow$ We prove that in a perfect matching M of G' each instance of a gadget $\Gamma'_{d-k,d}$, $2 \leq k \leq d$, has k outer edges in the matching and, therefore, the corresponding edges in G define a simple k-factor. Recall that the gadgets $\Gamma'_{d-k,d}$, contain $d-k$ core vertices. The core vertices must be matched in M to $d-k$ outer vertices. Therefore, the remaining k outer vertices must be matched to other k vertices of other gadget instances through k distinct outer edges. □

The advantage of the Γ' gadgets becomes more evident in dense graphs, where vertices have high degrees, and large k-values. We follow with a similar analysis as presented for the Γ set of gadgets (see Section 4.1). Let $G = (V, E)$ be the input graph, with $|V| = n$, and $G' = (V', E')$ be the inflated graph obtained from G by using the Γ' gadgets.

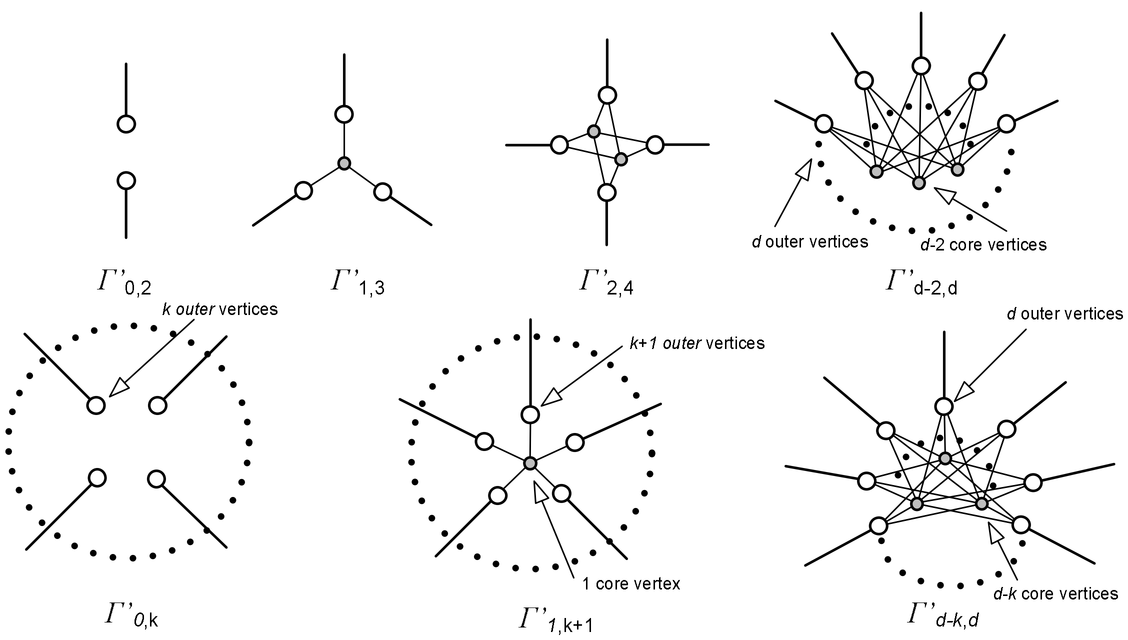

Figure 6: Alternate set of gadgets used in the k-factor version of Algorithm 1 for large values of k. Only outer and core vertices are needed. Core vertices appear as shaded circles.

Assume G is near-complete but $d(v) - k$ is bounded by a constant C for any vertex v, where $d(v)$ denotes the degree of vertex v. Then $|V'| = O(Cn + n^2) = O(n^2)$ and $|E'| = O(Cn^2 + n^2) = O(n^2)$. This results in $O(n^3)$ running time, which is a considerable improvement over the $O(n^3 k)$ bound for the general case.

Note that for Algorithm 1 gadgets of the Γ and Γ' sets may be combined. Therefore, the best performance is obtained if the gadget that substitutes a vertex v of degree d of the input graph G is $\Gamma_{k,d}$ for $k < d/2$, or $\Gamma'_{d-k,d}$ otherwise.

5. Conclusions

In many applications simple k-factors are more useful than k-factors. To our knowledge, efficient algorithms that compute simple k-factors existed only for special types of graphs, such as simple 2-factors of general graphs, simple 2-factors of 3-regular graphs (more efficiently), or $2k$-factors of regular $2m$ graphs, $k \leq m$. We present a new algorithm for computing a simple k-factor of a graph that fits any value of $k \geq 2$. Our algorithm, in addition to being simple to implement, is asymptotically as optimal as the available algorithms for maximum matching for the case of bounded degree graphs.

6. Acknowledgements

We would like to thank the editor and referees for their helpful comments and suggestions.

References

[1] J. Akiyama and M. Kano. *Book of Factors and Factorizations of Graphs*. June, 2007. Online version: http://gorogoro.cis.ibaraki.ac.jp/web/papers/FactorGraphVer1A4.pdf

[2] T. C. Biedl, P. Bose, E. D. Demaine, and A. Lubiw. Efficient algorithms for Petersen's matching theorem. *Journal of Algorithms*, 38(1):110–134, 2001.

[3] V. Chvátal. *Linear Programming*. W. H. Freeman, San Francisco CA, 1983.

[4] P. Diaz-Gutierrez and M. Gopi. Quadrilateral and tetrahedral mesh stripification using 2-factor partitioning of the dual graph. *The Visual Computer*, 21(8-10):689–697, 2005.

[5] A. Gibbons. *Algorithmic Graph Theory.* Cambridge University Press, 1985.

[6] M. Gopi and D. Eppstein. Single-strip triangulation of manifolds with arbitrary topology. *Computer Graphics Forum (EUROGRAPHICS)*, 23(3):371-379, 2004.

[7] P. Klein, S. Rao, M. Rauch, S. Subramanian. Faster shortest-path algorithms for planar graphs. *Proc. 26th Annual ACM Symposium on Theory of Computing (STOC'94)*, ACM Press, 27–37, 1994.

[8] S. Micali and V. V. Vazirani. An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. *Proc. 21st Annual Symp. on Foundations of Computer Science (FOCS '80)*, 17–27, 1980.

[9] M. Mucha and P. Sankowski. Maximum matchings via gaussian elimination. *Proc. 45th Annual Symp. on Foundations of Computer Science (FOCS '04)*, 248–255, 2004.

[10] M. Mucha and P. Sankowski. Maximum matchings in planar graphs via gaussian elimination. *Algorithmica*, 45(1):3–20, 2006.

[11] P. A. Peterson and M. C. Loui. The general maximum matching algorithm of Micali and Vazirani. *Algorithmica*, 3(1):511–533, 1988.

[12] M. D. Plummer. Graph factors and factorization: 1985-2003: A survey. *Discrete Mathematics*, 307:791-821, 2007.

[13] M. Thorup. Near-optimal fully-dynamic graph connectivity. *Proc. 32nd Annual ACM Symp. on Theory of Computing*, 343–350, 2000.

[14] C. Umans and W. Lenhart. Hamiltonian cycles in solid grid graphs. *Proc. 38th Annual Symp. on Foundations of Computer Science (FOCS '97)*, 496–505, 1997.

[15] C. Umans. An algorithm for finding Hamiltonian cycles in grid graphs without holes. *Honors thesis*, Williams College, 1996.