Improved Approximation Algorithms for Graph-TSP in k-Regular Bipartite Graphs

Jeremy Karp
R. Ravi (Advisor)

September 3, 2013

Abstract

We prove new results for approximating Graph-TSP. Specifically, we provide a polynomial-time $\frac{9}{7}$-approximation algorithm for cubic bipartite graphs and a $(\frac{9}{7} + \frac{1}{21(k-2)})$-approximation algorithm for k-regular bipartite graphs, both of which are improved approximation factors compared to previous results. Our approach involves finding a cycle cover with relatively few cycles, which we are able to do by leveraging the fact that all cycles in bipartite graphs are of even length along with our knowledge of the structure of cubic graphs.

1 Introduction

The traveling salesman problem is one of most well known problems in combinatorial optimization, famous for being hard to solve precisely. For several decades, the best approximation algorithm for the Metric-TSP problem, where distances between cities must obey the triangle inequality, was Christofides' algorithm [1976], which achieves a $\frac{3}{2}$-approximation factor. Until recently, the related Graph-TSP problem - a refinement of the Metric-TSP problem, where problem instances are obtained by taking the metric completion of an undirected, unweighted graph - did not have any approximation algorithms that achieved a better approximation factor than Christofides' more general algorithm. However, a wave of recent papers (Gamarnik, Lewenstein, and Sviridenko [2005]; Aggarwal, Garg, and Gupta [2011]; Boyd, Sitters, van der Ster, and Stougie [2011]; Gharan, Saberi, and Singh [2011]; Mömke and Svensson [2011]; Correa, Larré, and Soto [2012]; Sebő and Vygen [2012]) have provided significant improvements over Christofides' classic algorithm.

It is conjectured (Goemans [1995]) that a solution at most $\frac{4}{3}$ of the lower bound identified by the solution to the subtour relaxation linear program will always exist. This ratio is the one found in the worst known examples and is called the linear programming integrality gap of the problem. Currently, the best known approximation algorithm for Graph-TSP has an approximation factor of $\frac{7}{5}$ (Sebő and Vygen [2012]). However, algorithms with smaller approximation factors have been found for Graph-TSP instances generated by specific subclasses of graphs. In particular, algorithms for Graph-TSP in cubic graphs (where all nodes have degree 3) have drawn significant interest as this appears to be the simplest class of graphs that captures much of the complexity of the general case. The best known approximation algorithm for Graph-TSP in cubic graphs is due to Correa, Larré, and Soto [2012], whose algorithm achieves an approximation factor of $(\frac{4}{3} - \frac{1}{61236})$. Lastly, before discussing the contributions of this paper, we would like to credit an algorithm developed by Aggarwal, Garg, and Gupta [2011] for Graph-TSP in 3-edge-connected cubic graphs with a $\frac{4}{3}$-approximation factor, which is similar in spirit to the algorithm presented in this paper.

1.1 Overview

In this paper, we will present an algorithm to solve Graph-TSP, which guarantees a solution with at most $\frac{9}{7}n-2$ edges in cubic bipartite graphs. The best possible solution to Graph-TSP is a Hamiltonian cycle, which has exactly n edges, so this algorithm has an approximation factor of $\frac{9}{7}$. We also provide a $(\frac{9}{7} + \frac{1}{21(k-2)})$-approximation algorithm for k-regular bipartite graphs.

Proposition 1. *Any k-regular bipartite graph $G = (A \cup B, E)$ contains k edge-disjoint perfect matchings, $M_1, M_2, \ldots, M_k$ such that $E = (M_1 \cup M_2 \cup \ldots \cup M_k)$. Furthermore, we can decompose E into these k matchings in polynomial time.*

Proof. We will prove this by induction. As a base case, if $k = 1$ then every node has degree 1, so the edge set of the graph is a single perfect matching. As an induction hypothesis, assume that $k \geq 2$ and all $(k-1)$-regular bipartite graphs have $k-1$ edge-disjoint perfect matchings. Then, we can show that Hall's condition is satisfied. Any subset of nodes on one side of the partition $A' \subseteq A$ is incident on exactly $k|A'|$ edges. Let $B' = N(A')$, the subset of B that neighbors A'. B' is incident on exactly $k|B'|$ edges, and by definition B' is incident on $k|A'|$ edges that are also incident on A'. Then $k|B'| \geq k|A'|$ so $|N(A')| = |B'| \geq |A'|$. Therefore, we conclude that G has a perfect matching, M_1. We can use the Hopcroft-Karp algorithm [1973] to find this matching in $O(|E|\sqrt{|A \cup B|})$ in the worst case. If we remove M_1 from G we now have $G' = (A \cup B, E \backslash M_1)$, a $(k-1)$-regular bipartite graph. From the induction hypothesis, this subgraph has $k-1$ additional edge-disjoint perfect matchings, proving the claim. In total, this process takes $O(k|A|\sqrt{|A|}) = O(n^{\frac{5}{2}})$ in the worst case because $k \leq |A| = \frac{n}{2}$. In cubic bipartite graphs, $k = 3$ so the running time in these graphs is $O(n^{\frac{3}{2}})$ in the worst case. □

Note that in a cubic bipartite graph, we can apply Proposition 1 to find 3 edge-disjoint perfect matchings. The union of any 2 of these matchings forms a 2-factor. The following proposition demonstrates the close relationship between 2-factors and Graph-TSP tours in connected graphs.

Proposition 2. *Any 2-factor with k cycles in a connected graph can be extended into a spanning Eulerian multigraph with the addition of exactly $2(k-1)$ edges. This multigraph contains exactly $n + 2(k-1)$ edges in total.*

Proof. Contract each cycle in the 2-factor into a single vertex. The resulting contracted graph is connected and has k nodes. We can easily compute a spanning tree in this graph, which contains exactly $(k-1)$ edges. We claim that adding two copies of each of the edges in this spanning tree to the 2-factor yields a spanning Eulerian multigraph. Every node is incident on two edges in the 2-factor and every edge of the spanning tree is doubled. Then every node has even degree in the multigraph we obtain. Furthermore, edges we added from the spanning tree in the condensed graph ensures that the multigraph is connected, so this multigraph is Eulerian. Every node is covered by the 2-factor, so this multigraph is also spanning. Any 2-factor contains exactly n edges, adding two copies of $(k-1)$ edges results in a multigraph with exacly $n + 2(k-1)$ edges in total, completing the proof. □

We present an algorithm, $BIGCYCLE$, which begins by finding a 2-factor with at most $\frac{n}{7}$ cycles. Then, it applies the method used to prove Proposition 2 to generate a spanning Eulerian subgraph from this 2-factor containing at most $n + 2 \times (\frac{n}{7} - 1) = \frac{9}{7}n - 2$ edges.

The fundamental challenge in developing this approximation algorithm was creating a method to find cycle covers with relatively few cycles. To do this, we first remove all 4-cycles from the graph. When $BIGCYCLE$ finds a 4-cycle, it condenses it into two neighboring vertices, each of which has two other edges corresponding to the edges exiting the original 4-cycle. This contraction has the property that if a 2-factor through this condensed graph has average cycle length of x then we can easily expand the graph back to the original instance and this 2-factor will have average cycle length at least x. This process is described in more detail in Section 2.

Once the algorithm has condensed every 4-cycle in the graph, it generates a 2-factor in the condensed graph. If the resulting 2-factor has no 6-cycles, then we can expand the 4-cycles and this will be our solution. If the 2-factor does have a 6-cycle, then we contract either this 6-cycle or a larger subgraph that includes this 6-cycle, using methods described in detail in Section 3. It is worth noting that while it does not disrupt our analysis, these contraction operations can potentially disconnect the graph. We call the new subgraphs we introduce during contraction operations "gadgets" and the vertices and edges that form these gadgets "super-vertices" and "super-edges", respectively. In contrast, we refer to subgraphs, vertices, and edges that were entirely contained in the original graph as "organic". We then repeat this process, contracting any 4-cycles formed due to contractions in the graph, constructing a new 2-factor, and if this 2-factor contains a 6-cycle with no super-vertices or super-edges we contract it. When we have found a 2-factor whose 6-cycles (if any)

all contain super-vertices or super-edges, then we expand the contracted graph using the method described in Section 3 to obtain a 2-factor in the original graph with average cycle size at least 7. Equivalently, this 2-factor contains at most $\frac{n}{7}$ cycles. This upper bound on the number of cycles is proved in Theorem 2 in Section 4.7. We can then apply Proposition 2 to obtain a solution with at most $(\frac{n}{7} - 1) \times 2 + n = \frac{9}{7}n - 2$ edges, as claimed at the beginning of this section.

1.2 Roadmap

The rest of this paper is organized as follows: Section 2 describes a simple $\frac{4}{3}$-approximation to Graph-TSP in cubic bipartite graphs. Algorithms with this same approximation factor already exist for bridgeless cubic graphs. The primary purpose of this section is to provide intuition for the methods used to prove our main result, as the methods used to prove the $\frac{9}{7}$-approximation are in many ways an extension of those used to prove the $\frac{4}{3}$-approximation. This algorithm also demonstrates how bipartiteness can be used to our advantage when developing approximation algorithms. In Section 3, we describe the operations and gadgets used in the $\frac{9}{7}$-approximation and provide a formal description of our algorithm. Then, we prove our approximation factor guarantee in Section 4. In Section 5, we provide a $(\frac{9}{7} + \frac{1}{21(k-2)})$-approximation algorithm for k-regular bipartite graphs. Sections 6-12 are appendices, with diagrams detailing exactly how the graph expansions mentioned in the paper are performed, in all cases.

2 A Simple $\frac{4}{3}$-Approximation Algorithm for Graph-TSP in Cubic Bipartite Graphs

The approach taken throughout this paper to find good solutions to the Graph-TSP problem is to find 2-factors with relatively few cycles and then apply Proposition 2 to obtain a spanning Eulerian multigraph with relatively few total edges. One way to limit the number of cycles in a 2-factor is enforce a minimum cycle length for the cycles in the 2-factor. The algorithm in this section finds a 2-factor where all cycles are of length at least 6. Then, the 2-factor contains at most $\frac{n}{6}$ cycles. Applying Proposition 2, we get a spanning Eulerian multigraph with at most $n + 2(\frac{n}{6} - 1) = \frac{4}{3}n - 2$ edges.

A method used throughout this paper is to systematically replace certain subgraphs containing 4-cycles (and 6-cycles in the $\frac{9}{7}$-approximation algorithm) with other subgraphs. We will refer to these replacement subraphs as "gadgets". More formally, we will define the term "gadget" as follows:

Definition 1. A gadget is a subgraph that is isomorphic to one of the subgraphs shown in Figures 2, 4, 6, 8, 10, 12, 14, 16, and 18. Gadgets are used to replace other subgraphs containing 4- or 6-cycles.

Another term used throughout this paper is "organic," which we will also define formally.

Definition 2. A subgraph is organic if it consists entirely of nodes and edges contained in the original graph. In other words, none of the nodes or edges in the subgraph are part of a gadget which replaced a subgraph in the original graph. For a single edge to be organic, both its end-nodes must be organic.

Theorem 1. *Given a cubic bipartite graph G_0 with n vertices, there is a polynomial time algorithm that computes a spanning Eulerian subgraph H in $2G_0$ with at most $\frac{4}{3}n - 2$ edges.*

Proof. If G_0 contains no squares, then we construct a 2-factor, F_0, in the graph. Since G_0 is bipartite, square-less, and has no parallel edges, it contains no cycles of length less than 6. Then, F_0 has at most $\frac{n}{6}$ connected components. We can apply Proposition 2 to this 2-factor to obtain a spanning Eulerian mutligraph with at most $\frac{4}{3}n - 2$ edges.

If G_0 is not square-less, then we compute a 2-factor, F_0, in this graph and check if any of its cycles are of length 4. If not, then all cycles are of length 6 or greater and so we similarly apply Proposition 2, as described in the previous paragraph, giving us a solution with at most $\frac{4}{3}n - 2$ edges. Let $i = 0$.

Suppose then, that F_i contains at least one cycle of length 4. We select any one of these cycles and if its four edges leaving the cycle connect to four distinct vertices, we define a new graph G_{i+1} by taking G_i and replacing this cycle with the gadget shown in Figure 2. Note that this gadget and all other gadgets maintain the bipartiteness and cubicity of the graph. Similarly, not that this contraction operation and all successive contraction operations reduce the number of nodes and edges in the graph.

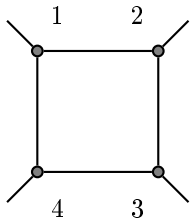

Figure 1: A square covered by the 2-factor, F_i, in graph G_i

Figure 2: We define a new graph and 2-factor G_{i+1} and F_{i+1}, where this gadget which replaces the square from the previous figure.

If two of the cycle's exiting edges connect to a common vertex, then we replace this cycle with the gadget shown in Figure 4.

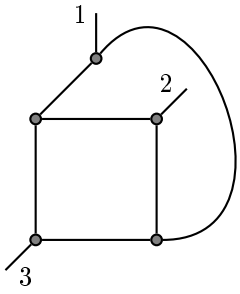

Figure 3: A square covered by the 2-factor, F_i, in graph G_i

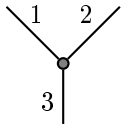

Figure 4: The gadget which replaces this configuration in graph G_{i+1}

If the both pairs of the cycle's diagonally opposite exiting edges connect to common vertices, then we replace this cycle with a single "super-edge" as shown in Figure 6.

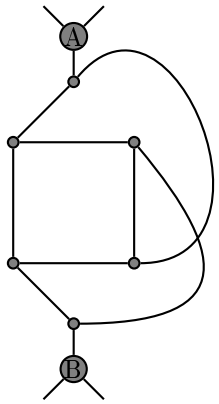

Figure 5: A square covered by the 2-factor, F_i, in graph G_i

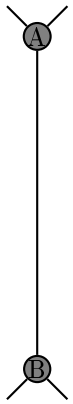

Figure 6: The super-edge which replaces this configuration in graph G_{i+1}

We now repeat this process on the condensed graph, G_{i+1}. We find a new 2-factor, F_{i+1}, and check if it contains any organic cycles of length 4. In other words, we are checking to see if there are any 4-cycles made up of nodes that were all in the original graph. If the current 2-factor contains any organic 4-cycles, then we condense these cycles into the appropriate gadget, shown in figures 2, 4, and 6.

Eventually, we will find a 2-factor, F_k, with no organic 4-cycles in the condensed graph. We now expand our gadgets, expanding the gadgets we compressed last first (last in first out). When we expand a super-edge, if this edge was used in the 2-factor, then we can weave this 2-factor through the configuration (see Figures 5 and 50) we get when we expand the super-edge, which lengthens the cycle that contained the super-edge. If the super-edge was not part of the 2-factor, then we expand the super-edge and add a new 6-cycle which covers the 6 nodes added from this expansion. These expansions are documented in detail in Section 6.5.

When we expand a gadget from Figure 2 or Figure 4, we can simply extend the 2-factor, F_i, through the gadget's expansion (Figures 1 and 3, respectively) and we obtain a 2-factor, F_{i-1}, in the expanded graph, G_{i-1}, with all cycles of greater or equal length than those in the 2-factor in the graph before we expanded the gadget. Specifically, the cycle passing through the gadget we expanded most recently will be of length at least 6 because it had length at least 4 before expanding the gadget and these expansions always increase the length of the cycle by at least 2. These expansions are documented in detail in Sections 6.1-6.4.

After expanding all of the gadgets, we have a 2-factor, F_0, in the original graph G_0 with minimum cycle length at least 6. Then, F_0 is composed of at most $\frac{n}{6}$ cycles. Using Proposition 2, we get a a connected spanning Eulerian subgraph with at most $n + 2 \times (\frac{n}{6} - 1) = \frac{4}{3}n - 2$ edges. □

One easily overlooked challenge in designing this algorithm was identifying gadgets for which upon expansion, we can keep long cycles long and not introduce smaller cycles. Section 6 documents every expansion operation needed to run this algorithm and demonstrates that each expansion does not introduce any 4-cycles into the 2-factor.

3 A $\frac{9}{7}$-Approximation Algorithm for Graph-TSP in Cubic Bipartite Graphs

3.1 Overview

We can extend the approach described in Section 2 to obtain an improved approximation guarantee. The biggest difference in the approach in the next algorithm is that we will condense our graph so it becomes nearly square-less, instead of just removing squares that are contained in a specific 2-factor. To do this, at every stage of the algorithm, if there is a configuration depicted in Figures 1, 3, or 5, then we replace it with the gadgets from Figures 2, 4, or 6, respectively. We saw in the previous section that given a 2-factor, F_i, in a condensed graph, G_i, we can expand these gadgets and find a 2-factor, F_{i-1}, for the expanded graph, G_{i-1}, by including all edges from F_i, and weaving through the nodes added to the graph by expanding the gadgets. Furthermore, the cycles of $F_{i-1}$are at least as large as the corresponding cycles in F_i and do not increase the number of cycles.

A consequence of the previous paragraphs is that throughout this algorithm, we can essentially think of the graph as square-less. In a square-less graph, all 2-factors will have an average cycle length of at least 6, so all 2-factors will have at most $\frac{n}{6}$ cycles, which results in a $\frac{4}{3}$-approximation. In order to improve our approximation guarantee, we need to target 6-cycles, as well as 4-cycles. The algorithm we present finds a 2-factor in which every 6-cycle corresponds to a distinct cycle of size 8 or larger. Then, we can find a 2-factor in which every large cycle and its corresponding 6-cycles have average cycle length of at least 7 (Lemma 6 in Section 4.7). We then show that this is enough to conclude that the 2-factor contains at most $\frac{n}{7}$ cycles (Theorem 2 in Section 4.8). This is primary contribution of this paper.

In the following section, we introduce some new gadgets. These gadgets are used in a similar fashion as the gadgets presented in Section 2: when our 2-factor contains organic 6-cycles, we will use these gadgets to condense our graph and remove these cycles. We then repeat this process (condensing squares that appear along the way) and compute a new 2-factor in the condensed graph until we obtain a 2-factor with no organic 6-cycles. We show that expanding the graph can create a small number of new 6-cycles in our 2-factor, but we are able to account for them, ensuring that the bounds described in the previous paragraph must hold.

As a side note, the only way for a square to remain in our graph is if there is a $K_{3,3}$. Unless the original graph is simply $K_{3,3}$, in which case we can trivially find a Hamiltonian cycle, then the original graph cannot contain a $K_{3,3}$ as it is connected. However, as noted in Section 1.1, the some of the compression opeartions we introduce in this section can create multiple connected components. Then, if our algorithm results in a $K_{3,3}$ occurring in the compressed graph, when we compute a 2-factor, the $K_{3,3}$ will be covered by a single 6-cycle, which we show in our analysis to expand a set of cycles with desirable properties.

3.2 Some New Gadgets

In this section, we present six new configurations to be replaced by gadgets, which we use in the algorithm. We will give these configurations the names $0-P$, $1-P$, $2-P$, $W2-P$, $SV2-P$, and $SE2-P$. The "P" comes from the word "parasite", which we thought the $1-P$ resembled during the development of the algorithm. The $W2-P$ is named as it is because it is a "winged" $2-P$. The $SV2-P$ is named as it is because the gadget that replaces it is a "super-vertex". Similarly, the $SE2-P$ is named as it is because the gadget that replaces it is a "super-edge". The first gadget is two super-vertices which replace a simple 6-cycle, $0-P$.

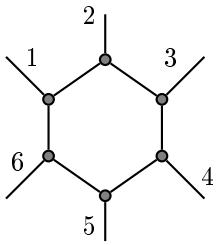

Figure 7: A simple 6-cycle: $0-P$

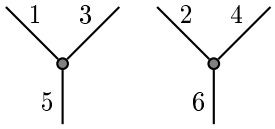

Figure 8: The gadget which replaces $0-P$: $0-P-G$

The remaining gadgets are special cases of 6-cycles. Note that every $1-P$ contains a $0-P$, all $2-P$s contain a $1-P$, and $SE2-P$s, $SV2-P$s, and $W2-P$s are special cases of $2-P$s.

The motivation to use these additional gadgets comes out of necessity, to prevent large numbers of 6-cycles from being introduced into the 2-factor during the expansion phase of the algorithm. For example, Figures 59-60 in Section 7.2 document an expansion that turns an $x+y+4$-cycle passing through a gadget which replaced a $0-P$ into two cycles of lengths $x+3$ and $y+5$. Since the algorithm will condense $1-P$s before $0-P$s, this ensures that y, the length of a path, is at least 3, meaning that the $y+5$-cycle is not a 6-cycle. The motivation for introducing the $2-P$ and $W2-P$ is similar. Notice that if edges 1 and 3 or edges 2 and 4 out of a $W2-P$ are incident on a common node, then the gadget we use to replace the $W2-P$ will have a doubled edge. This motivates the introduction of the $SV2-P$ and $SE2-P$ subgraphs, which allow us to avoid introducing doubled edges into the graph.

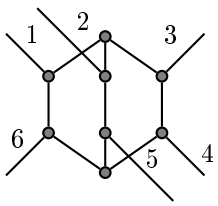

Figure 9: The 1-parasite: $1-P$

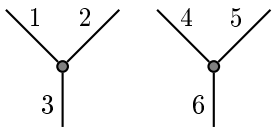

Figure 10: The gadget which replaces $1-P$: $1-P-G$

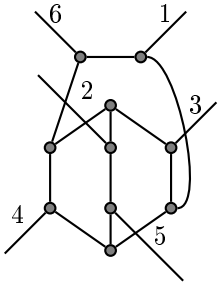

Figure 11: The 2-parasite: $2-P$

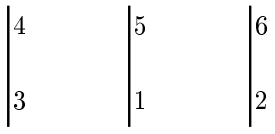

Figure 12: The gadget which replaces $2-P$: $2-P-G$

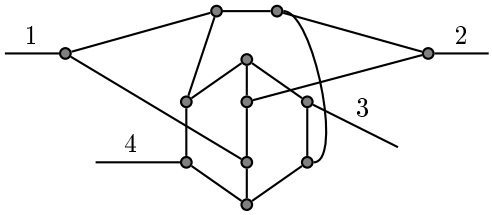

Figure 13: The winged 2-parasite: $W2-P$

Figure 14: The gadget which replaces $W2-P$: $W2-P-G$

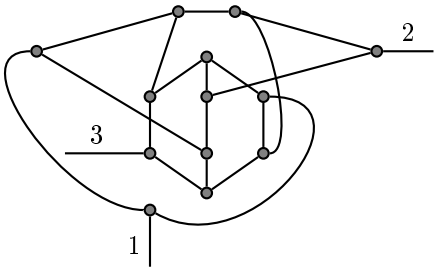

Figure 15: The super-vertex 2-parasite: $SV2-P$

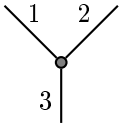

Figure 16: The gadget which replaces $SV2-P$: $SV2-P-G$

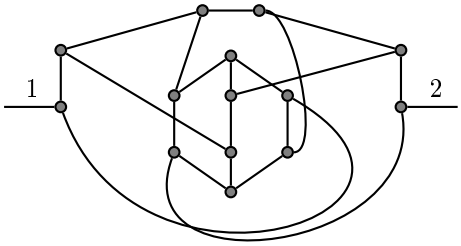

Figure 17: The super-edge 2-parasite: $SE2-P$

Figure 18: The gadget which replaces $SE2-P$: $SE2-P-G$

In the next subsection, we present a detailed description of the algorithm.

3.3 The Algorithm

Algorithms 1-8 present complete pseudocode for the BIGCYCLE algorithm. The remainder of this section explains and motivates the operations performed by the pseudocode.

Algorithm 1 BIGCYCLE(G)

Input: An undirected, unweighted, cubic, bipartite graph $G = (V, E)$
$\mathcal{G} \leftarrow$ An empty global variable
$F_{compressed} \leftarrow$COMPRESS($G$)
$F \leftarrow$EXPAND($F_{compressed}$)
$TSP \leftarrow$PROP2(G,F)
Return TSP

Algorithm 2 COMPRESS(G)

Input: An undirected, unweighted, cubic, bipartite graph $G = (V, E)$
$i \leftarrow 0$
$G_i \leftarrow G$
$\mathcal{G} \leftarrow \{G_i\}$ #$\mathcal{G}$ is the same global variable initialized in BIGCYCLE
$\mathcal{G} \leftarrow \mathcal{G} \cup$COMPRESS_SUB($G_i$,$SE2-P$,$SE2-P-G$,$i$)
$i \leftarrow |\mathcal{G}| - 1$
$G_i \leftarrow G_i \in \mathcal{G}$
$\mathcal{G} \leftarrow \mathcal{G} \cup$COMPRESS_SUB($G_i$,$SV2-P$,$SV2-P-G$,$i$)
$i \leftarrow |\mathcal{G}| - 1$
$G_i \leftarrow G_i \in \mathcal{G}$
$\mathcal{G} \leftarrow \mathcal{G} \cup$COMPRESS_SUB($G_i$,$W2-P$,$W2-P-G$,$i$)
$i \leftarrow |\mathcal{G}| - 1$
$G_i \leftarrow G_i \in \mathcal{G}$
$\mathcal{G} \leftarrow \mathcal{G} \cup$COMPRESS_SUB($G_i$,$2-P$,$2-P-G$,$i$)
$i \leftarrow |\mathcal{G}| - 1$
$G_i \leftarrow G_i \in \mathcal{G}$
$\mathcal{G} \leftarrow \mathcal{G} \cup$COMPRESS_SUB($G_i$,$1-P$,$1-P-G$,$i$)
$i \leftarrow |\mathcal{G}| - 1$
$G_i \leftarrow G_i \in \mathcal{G}$
$\mathcal{G} \leftarrow \mathcal{G} \cup$COMPRESS_SUB($G_i$,$0-P$,$0-P-G$,$i$)
$i \leftarrow |\mathcal{G}| - 1$
$G_i \leftarrow G_i \in \mathcal{G}$
$F \leftarrow$2FACTOR(G_i) #We know F contains no organic 6-cycles, otherwise
#the last call of COMPRESS_SUB would not have terminated
Return F

Algorithm 3 EXPAND(F)

Input: A 2-factor F
$i \leftarrow |\mathcal{G}| + 1$ #$\mathcal{G}$ is the same global variable initialized in BIGCYCLE
While $i > 0$:
  $F' \leftarrow$The edges not in F_i, which pass through the corresponding subgraph/2-factor replacement in Sections 6-12, which is uniquely determined by the gadget induced by $V_i \backslash V_{i-1}$ in G_i, the subgraph induced by $V_{i-1} \backslash V_i$ which the gadget replaced in G_{i-1}, and the 2-factor F_i
  #Note that graphs G_i, G_{i-1} are stored in global variable $\mathcal{G}$
  $F_{i-1} \leftarrow (F_i \cap E_i) \cup F'$
  $F_{i-1} \leftarrow$LOCAL(G_i,G_{i-1},F_i,F_{i-1})
  $i \leftarrow i - 1$
Return F_0

Algorithm 4 PROP2(G, F)

Input: An undirected, unweighted, cubic, bipartite graph $G = (V, E)$
and a 2-factor F
$k \leftarrow$The number of cycles in F
$ST \leftarrow$A set of $k-1$ edges connecting all the cycles in F, found using the method described in the proof of Proposition 2
$TSP \leftarrow$A multiset containing one copy of each edge in F and two copies of each edge in ST
Return TSP

Algorithm 5 COMPRESS_SUB(G, S, T, i)

Input: An undirected, unweighted, cubic, bipartite graph $G = (V, E)$,
a subgraph S, a gadget T, and an integer i.
$G_i \leftarrow G$
$\mathcal{G}_{temp} \leftarrow \emptyset$
$F_i \leftarrow$2FACTOR(G_i)
While [(G_i contains a square not in a $K_{3,3}$) or (F_i has a 6-cycle that is part of an organic subgraph S)]:
  While (G_i contains a square not in a $K_{3,3}$):
    $G_{i+1} \leftarrow$SQUARES(G_i)
    $\mathcal{G}_{temp} \leftarrow \mathcal{G}_{temp} \cup G_{i+1}$
    $i \leftarrow i + 1$
    $F_i \leftarrow$2FACTOR(G_i)
  IF F_i has a 6-cycle that is part of an organic subgraph S:
    $G_{i+1} \leftarrow$The resulting graph, after replacing S in G_i with gadget T
    $\mathcal{G}_{temp} \leftarrow \mathcal{G}_{temp} \cup G_{i+1}$
    $i \leftarrow i + 1$
    $F_i \leftarrow$2FACTOR(G_i)
Return $\mathcal{G}_{temp}$

Algorithm 6 2FACTOR(G)

Input: An undirected, unweighted, cubic, bipartite graph $G = (V, E)$
$M_1 \leftarrow$FindPerfectMatching(G)
$M_2 \leftarrow$FindPerfectMatching($(V, E \backslash M_1)$)
$M_3 \leftarrow E \backslash (M_1 \cup M_2)$
For every $e \in E$:
  If e is a super-edge which replaced an organic subgraph depicted in Figure 5:
    $x_e = 1$
  Else:
    $x_e = 0$
$a \leftarrow$argmin$\{\sum_{e \in M_a} x_e\}$
$F \leftarrow E \backslash M_a$
Return F

A sub-routine we will use frequently in this algorithm involves checking if the current graph contains any squares not in $K_{3,3}$s, then condenses the graph by replacing these squares with the appropriate gadgets (shown in Figures 1-6 in Section 2). This process is repeated until the only squares in the graph (if any) are part of $K_{3,3}$s. This procedure is called *SQUARES*.

Algorithm 7 SQUARES(G)

Input: An undirected, unweighted, cubic, bipartite graph $G = (V, E)$
If G contains a square not in a $K_{3,3}$: #Can be determined in $O(n)$ checking the #neighborhood of radius 3 of every node
  $G \leftarrow$The resulting graph, after replacing the square in G with the appropriate gadget among those depicted in Figures 2, 4, and 6.
Return $G_i = (V_i, E_i)$

Algorithm 8 LOCAL(G_i,G_{i-1},F_i,F_{i-1})

Input: Two undirected, unweighted, cubic, bipartite graphs G_i and G_{i-1} and two 2-factors in these graphs, F_i and F_{i-1}
If the expansion from G_i to G_{i-1} was a $0-P$ expansion that introduced an organic 6-cycle in the configuration shown in Figure 21 into 2-factor F_{i-1}:
  $F_{i-1} \leftarrow$The resulting 2-factor, after exchanging the edges in F_{i-1} shown in Figure 21 with those shown in Figure 22
Return F_{i-1}

3.3.1 Finding a "good" 2-factor in the condensed graph G_k

We start the algorithm by receiving a connected cubic bipartite graph. Call this graph G_0. Note that this algorithm can be used on disconnected graphs by applying the algorithm to each connected component. If G_0 is a $K_{3,3}$ then we compute a 2-factor in this graph, which will be a Hamiltonian cycle, and return this cycle as our solution. Otherwise, we run *SQUARES* until we are returned a graph with no squares except possibly inside of $K_{3,3}$s. Let i be the number of times *SQUARES* was run, and G_i is the compressed graph at the end of this process. Next, construct a 2-factor, F_i, in G_i. Recall from Definition 2 that subgraphs, specifically the configurations $SE2-P$, $SV2-P$, $W2-P$, $2-P$, $1-P$, $0-P$, are called organic if they are made up entirely of nodes and edges that were in the original graph. If F_i contains no organic 6-cycles, then we advance to the next phase of the algorithm, described in Section 3.3.2. In this case, $k = i$.

If F_i does contain an organic 6-cycle, C, then we check if the current compressed graph G_i contains organic specialized configurations in the following order (ordered from most specialized to most general): $SE2-P$, $SV2-P$, $W2-P$, $2-P$, $1-P$, $0-P$. We choose the first organic configuration on the list (the most specialized configuration) we can find in G_i and replace this configuration with the corresponding gadget, outputting graph G_{i+1} to reflect this change. We then run *SQUARES* until we have removed any 4-cycles generated as a consequence of replacing a parasite with one of our gadgets, obtaining a new compressed graph G_j. We construct a new 2-factor F_j and repeat the process in this paragraph until we have a 2-factor F_k with no organic 6-cycles, in a condensed graph G_k, where k is the total number of gadget replacement operations performed during this phase of the algorithm.

Pseudo code for this step can be found in Algorithm 2.

3.3.2 Expanding a 2-factor in G_k into a 2-factor in G_0

The process of expanding F_k and G_k so that we get back to the original graph G_0 with a desireable 2-factor F_0 is very similar to the process described in Section 2. In this section, We will describe this process in more detail.

We will reverse the process described in Section 3.3.1 by replacing our gadgets in compressed graph G_i with the original configuration from the earlier graph G_{i-1} in the reverse order of that in which we replaced the configurations. In other words, the gadgets we inserted last are those which we first with their original configuration. We call this process "expanding" because each one of these operations adds vertices and edges

to the graph. After we have made each replacement to expand the graph, F_i is no longer a 2-factor in G_{i-1} because the new nodes added by the most recent expansion step are not covered by F_i. However, we can add edges to F_i so that it becomes a 2-factor, F_{i-1} in the graph after this expansion step. It may not be immediately clear that this is always possible. In fact, one of the bigger challenges in developing this algorithm was choosing a set of gadgets where this property holds. Figures 19 and 20 show an example of how this process works, and detailed figures covering all possible instances are presented in the appendices, found in Sections 6-12.

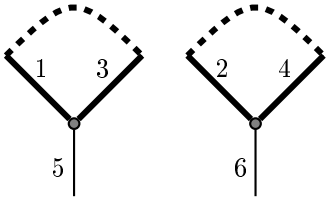

Figure 19: A pair of super-vertices in G_i. The bolded edges are included in 2-factor F_i. The dashed bold edges represent a path, included in F_i.

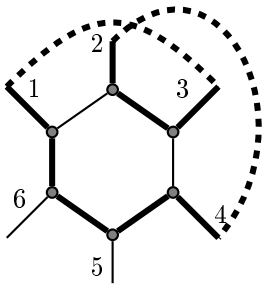

Figure 20: 2-factor F_{i-1} after expanding the super-vertices from Fig. 15.

At each expansion, we are able to extend F_i into a set of edges F_{i-1} , which will be a 2-factor in the expanded graph, G_{i-1}. In order to optimize the performance of $BIGCYCLE$, we must impose one extra operation in this phase of the algorithm. After each expansion of a $0-P$ that introduces an organic 6-cycle, C_1, into the 2-factor, we will perform a local search to see if C_1 and the nearby edges of the newly expanded 2-factor F_{i-1} contained in the surrounding portion of the graph are in the position shown in Figure 21. If they are, then we update F_{i-1} so that it covers this region with one fewer cycle, as shown in Figure 22. In addition to being an effective heuristic to reduce the number of cycles in our final 2-factor, this operation allows us to improve the approximation factor we prove in Section 4 by removing an otherwise troubling corner case (see Remark 1, Figure 26, and Lemma 5).

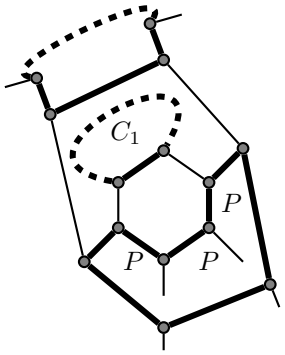

Figure 21: The configuration $BIGCYCLE$ searches for after expanding any $0-P$ that introduces an organic 6-cycle. The edges labeled "P" are the protected edges from the previous expansion.

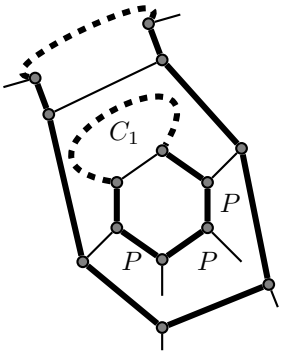

Figure 22: The updated 2-factor F_{i-1} after the configuration in Figure 21 is corrected.

At this point, we can repeat this process of replacing gadgets with their original configurations and adding edges to the 2-factor until we have expanded the graph back to the original input G_0 and have a 2-factor, F_0, in this graph. In Section 4 we prove that F_0 has at most $\frac{n}{7}$ cycles, which allows us to obtain the $\frac{9}{7}$-approximation factor.

Pseudo code for this step can be found in Algorithm 3.

3.3.3 Obtaining a good final solution by adding edges to F_0

We now have a 2-factor F_0, which contains k cycles. We now apply the method described in the proof of Proposition 2 to obtain a solution with $n + 2(k-1)$ edges. In Section 4 we prove that F_0 has at most $\frac{n}{7}$ cycles, so this gives us a solution of at most $\frac{9}{7}n - 2$ edges.

Pseudo code for this step can be found in Algorithm 4.

4 Accounting for 6-Cycles

In the proof of our approximation guarantee, the limitation on producing a lower approximation factor comes from the possibility that some proportion of our final 2-factor's cycles will be of length 6. Most operations the algorithm performs while expanding the 2-factor from the condensed to the original graph (Section 3.3.2) result in cycles of length 8 or larger, so in this section we will look at the few operations that create organic 6-cycles in detail. To account for 6-cycles, we show that every organic 6-cycle can be put in correspondence with some long cycle of length 8 or longer. Then, Lemma 6 demonstrates that the average cycle length of any long cycle and its corresponding set of 6-cycles is sufficiently long to ensure that our final cycle cover has relatively few cycles, even if some of them are 6-cycles.

Figures 23 and 24, taking the dashed lines to be paths of lengths x and y, demonstrate how a $(y+7)$-cycle can turn into a a 6-cycle and a $(y+5)$-cycle if $x = 3$:

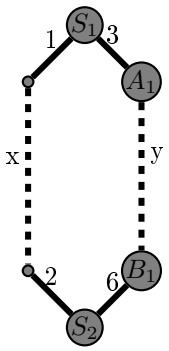

Figure 23: A cycle in F_i, a 2-factor over the condensed graph G_i. S_1 and S_2 are two super-vertices which replaced a standard hexagon. The dashed lines represent paths of length 3.

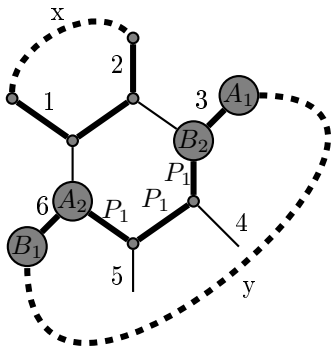

Figure 24: The cycle from Figure 23, after expanding S_1 and S_2

We will carefully analyze this and several other cases that form the bottleneck in our analysis, which occur when expanding our graph back to its original state. We will account for organic 6-cycles by creating a correspondence from every 6-cycle in the final 2-factor F_0 to some larger cycle in F_0. This way, if we can show that every large cycle of length $k \geq 8$ in F_0 is affiliated with at most $f(k)$ 6-cycles, then the average

cycle length is at least $\min_{k \geq 8} \frac{6 \times f(k)+k}{f(k)+1}$. Once we have placed a lower-bound on the average cycle length in this manner (Lemma 6), we can easily determine our approximation factor (Theorems 2 and 3).

4.1 Expanding $0-P$ gadgets

Appendix B in Section 7 documents, in detail for all cases, the process of winding the 2-factor through a $0-P$ after the algorithm has expanded the $0-P$'s replacement gadget. An examination of this appendix confirms that the example shown in Figures 23 and 24 is the only type of operation involving the $0-P$ configuration that can introduce an organic 6-cycle into the 2-factor during an expansion.

Remark 1. If an expansion operation of the type shown in Figures 23 and 24 were to occur, then it is not possible for the nodes corresponding to A_1 and B_1 in these figures to be the super-vertices of a $0-P$'s gadget whose expansion introduces an organic 6-cycle into the 2-factor. If this were to happen, then edges corresponding to (A_1, B_2) and (B_1, A_2) would have to be in the position of either edges 1 and 2 or edges 3 and 6 in Figure 23. In the first case, the graph G_{i-1} would have contained a $1-P$ and would have been compressed differently at the time when A_1 and B_1 would have been created during a compression (see Figure 25). Then, at this stage, the algorithm would have have replaced a $1-P$, $2-P$, or $W2-P$ at this time, not a $0-P$, which would have been necessary to create A_1 and B_1 as specified. In the second case, upon expanding, the graph G_{i-1} and 2-factor F_{i-1} would be in the configuration shown in Figure 21, prompting a local improvement so that this expansion no longer introduces an organic 6-cycle. Figure 25 shows the case when (A_1, B_2) is in the position of edge 1 in Figure 23, and Figure 26 shows the case when (A_1, B_2) is in the position of edge 3 in Figure 23.

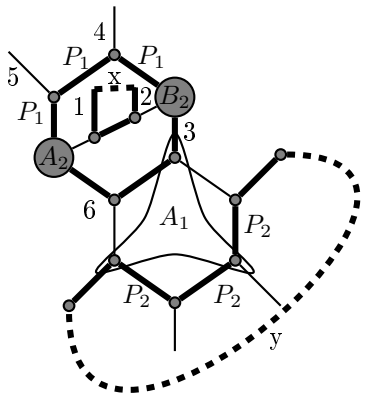

Figure 25: The cycles from Figure 24, after expanding A_1 and B_1, if these nodes had been a $0-P$'s gadget whose expansion introduced a 6-cycle, where edge (A_1, B_2) was in the position of edge 1 in Figure 23. We can see in this figure that nodes A_2 and B_2 are part of an organic $1-P$. Then, A_1 and B_1 cannot be a $0-P$'s gadget in this configuration, otherwise the *BIGCYCLE* algorithm would have performed different operations, compressing this $1-P$ or some other $1-P$, $2-P$, or $W2-P$ instead of the $0-P$ that A_1 and B_1 replaced.

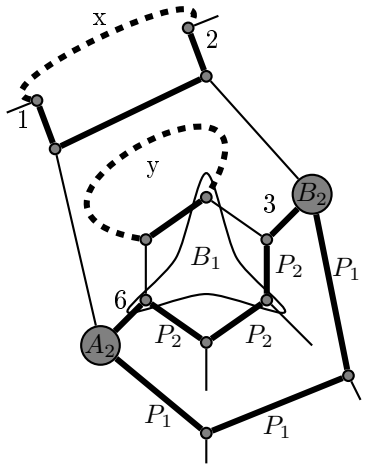

Figure 26: The cycles from Figure 24, after expanding A_1 and B_1, if these nodes had been a $0-P$'s gadget whose expansion introduced a 6-cycle, where edge (A_1, B_2) was in the position of edge 3 in Figure 23. We can see in this figure the cycle containing path y corresponds to 6-cycle C_1 in the configuration shown in Figure 21. Then, the algorithm would have updated the 2-factor to the configuration shown in Figure 22, making path y part of a 10-cycle. Then, because of this local correction, this expansion step would not have introduced an organic 6-cycle into the 2-factor.

We now give a definition and prove two lemmas about "protected edges", organic paths which help us analyze the performance of the *BIGCYCLE* algorithm. We use this term because protected edges cannot be separated from each other in the 2-factor during subsequent expansion operations.

Definition 3. Protected edges are identified during the "expanding" phase of the algorithm (described in Section 3.3.2), when the expansion operations shown in Figures 23-24, 27-28, 29-30, and 31-32 are performed. In the expansion shown in Figures 23-24, the three edges in the 2-factor that connect node A_2 to B_2 in Figure 24, labeled "P_1", are protected. In the expansion shown in Figures 27-28 the five edges in the 2-factor that connect node A_2 to B_2 in Figure 28, labeled "P", are protected. In the expansion shown in Figures 29-30, the four edges in the middle of the path connecting node B_1 to B_2 in Figure 30, labeled "P", are protected. In the expansion shown in Figures 31-32, the five bolded edges in the middle of the path connecting node A_2 to B_2 in a cycle of the form shown in Figure 34, labeled "P", are protected.

Lemma 1. *Let P_1 and P_2 be the sets of protected edges corresponding to two 6-cycles, C_1 and C_2, respectively. Then $P_1 \cap P_2 = \emptyset$.*

Proof. Suppose not, for the sake of contradiction. Then there is some edge e such that $e \in P_1$ and $e \in P_2$. All four expansion operations (described in detail in sections 4.1-4.4) that introduce organic 6-cycles and identify protected edges are such that there is some integer i where $e \notin E_{i+1}$ but $e \in E_i$. Then, the expansion operation from G_{i+1} to G_i will introduce both C_1 and C_2 to 2-factor F_i as organic 6-cycles. This is not possible, as the expansion operations which introduce organic 6-cycles all introduce exactly one 6-cycle into the 2-factor, proving the lemma. □

Lemma 2. *Let P_1 and P_2 be the sets of protected edges corresponding to two 6-cycles, C_1 and C_2, respectively. Then, $V(P_1) \cap V(P_2) = \emptyset$.*

Proof. Suppose not, for the sake of contradiction. Then there is some node v such that $v \in V(P_1)$ and $v \in V(P_2)$. All four expansion operations (described in detail in sections 4.1-4.4) that introduce organic 6-cycles and identify nodes that are endpoints of protected edges are such that there is some integer i where $v \notin V_{i+1}$ but $v \in V_i$. Then, the expansion operation from G_{i+1} to G_i will introduce both C_1 and C_2 to 2-factor F_i as organic 6-cycles. This is not possible, as the expansion operations which introduce organic 6-cycles all introduce exactly one 6-cycle into the 2-factor, proving the lemma. □

Definition 4. A 6-cycle is "affiliated" with the cycle in the 2-factor containing its protected edges.

Proposition 3. *A cycle, C, of length x has at most $\frac{x}{3}$ affiliated 6-cycles that were formed during the expansion of a $0-P$ gadget.*

Proof. Suppose for the sake of contradiction that this cycle has $y > \frac{x}{3}$ affiliated 6-cycles that were formed during the expansion of a $0-P$ gadget. Each of these 6-cycles was formed during a distinct expansion operation (no expansion operation introduces more than one organic 6-cycle), so the protected edges for each of these cycles are disjoint. Each of these 6-cycles has 3 protected edges, so $3y > x$ of the edges in C are protected edges affliated with 6-cycles that were formed during the expansion of a $0-P$ gadget. This is a contradiction because C has fewer than $3y$ total edges. □

4.2 Expanding $1-P$ gadgets

Appendix C in Section 8 documents, in detail for all cases, the process of winding the 2-factor through a $1-P$ after the algorithm has expanded the $1-P$'s replacement gadget. An examination of this appendix confirms that the example shown in Figures 27 and 28 is the only type of operation involving the $1-P$ configuration that can introduce an organic 6-cycle into the 2-factor during an expansion.

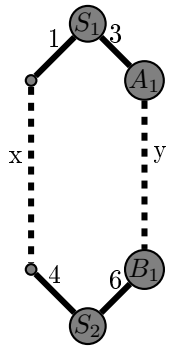

Figure 27: A cycle in F_i, a 2-factor over the condensed graph G_i. S_1 and S_2 are two super-vertices which replaced a $1-P$.

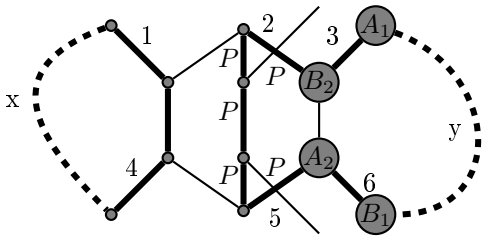

Figure 28: The cycle from Figure 27, after expanding S_1 and S_2

The difference in this case is that now five edges are "protected", rather than three for the $0-P$, when the graph is expanded. To see this, consider Figure 28, where we can see that the nodes on the dark path from B_3 to A_3 cannot be super-vertices. For the nodes along this path to become part of a 6-cycle from expanding $0-P$ or $1-P$ gadgets, they must be in between two corresponding super-vertices located exactly five edges apart on one of the cycles of the 2-factor. The path between A_3 and B_3 in Figure 28 is five edges, so any sufficiently long path that contains these nodes has more than five edges, meaning that if these edges were to become part of a different cycle through expanding $0-P$ or $1-P$ gadgets, it would have to be a cycle of length at least 8. These edges are protected (Definition 3) like those discussed in Section 4.1.

Proposition 4. *A cycle, C, of length x has at most $\frac{x}{5}$ affiliated 6-cycles that were formed during the expansion of a $1-P$ gadget.*

Proof. Suppose for the sake of contradiction that this cycle has $y > \frac{x}{5}$ affiliated 6-cycles that were formed during the expansion of a $1-P$ gadget. Each of these 6-cycles was formed during a distinct expansion operation (no expansion operation introduces more than one organic 6-cycle), so the protected edges for each of these cycles are disjoint. Each of these 6-cycles has 5 protected edges, so $5y > x$ of the edges in C are protected edges affliated with 6-cycles that were formed during the expansion of a $1-P$ gadget. This is a contradiction because C has fewer than $5y$ total edges. □

This "bad" expansion, then, is very similar to the "bad" expansion of $0-P$ gadgets. In fact, the main difference is that expanding these $1-P$ gadgets is less costly because such an expansion protects more edges than the corresponding $0-P$ gadget expansion. Consequently, in our worst-case analysis, we will tend to discuss the $0-P$ gadget expansion as this will be sufficient to analyze worst-case performance of the algorithm.

4.3 Expanding $2-P$ gadgets

Appendix D in Section 9 documents, in detail for all cases, the process of winding the 2-factor through a $2-P$ after the algorithm has expanded the $2-P$'s replacement gadget. An examination of this appendix confirms that the example shown in Figures 29 and 30 is the only type of operation involving the $2-P$ configuration that can introduce an organic 6-cycle into the 2-factor during an expansion.

The instances where expanding a $2-P$'s gadget can split off a 6-cycle are substantially different than those we examined for $0-P$ and $1-P$ gadgets. This is because $2-P$s are replaced by super-edges rather than super-vertices. Figures 29 and 30 demonstrate this operation:

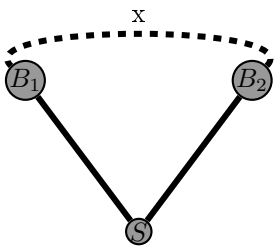

Figure 29: A cycle in F_i, a 2-factor over the condensed graph G_i. The edges (S, B_1) and (S, B_2) are two super-edges which replaced a $2-P$. The dashed line is a path of length x, where $x \geq 4$ and even.

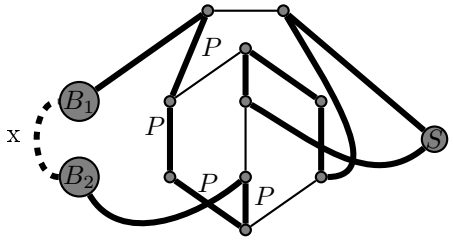

Figure 30: The cycle from Figure 29, after expanding the two super-edges

In these figures, we see that this expansion requires two super-edges to be directly neighboring each other in a cycle of the 2-factor. When this expansion is performed, the two super-edges are split off and form a 6-cycle while the larger cycle they came from increases in length by four. These four new edges are protected (Definition 3).

Proposition 5. *A cycle, C, of length x has at most $\frac{x}{4}$ affiliated 6-cycles that were formed during the expansion of a $2-P$ gadget.*

Proof. Suppose for the sake of contradiction that this cycle has $y > \frac{x}{4}$ affiliated 6-cycles that were formed during the expansion of a $2-P$ gadget. Each of these 6-cycles was formed during a distinct expansion operation (no expansion operation introduces more than one organic 6-cycle), so the protected edges for each of these cycles are disjoint. Each of these 6-cycles has 4 protected edges, so $4y > x$ of the edges in C are protected edges affliated with 6-cycles that were formed during the expansion of a $2-P$ gadget. This is a contradiction because C has fewer than $4y$ total edges. □

4.4 Expanding super-edges which replaced squares

Figure 31: A super-edge in G_i which is not covered by the 2-factor F_i

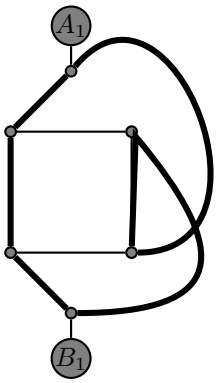

Figure 32: The subgraph in G_{i-1} that replaced the super-edge in Figure 31. Bold edges indicate those included in 2-factor F_{i-1}.

Expanding super-edges, not in the 2-factor, of the type shown in Figure 31 can introduce organic 6-cycles into the 2-factor. However, Algorithm 6 finds a 2-factor by computing three disjoint perfect matchings and taking the union of the two perfect matchings that contain the most of these dangerous super-edges. Then, strictly fewer than $\frac{1}{2}$ of the super-edges of the type shown in Figure 31 will not be included in the 2-factor. Then, we can put each super-edge of this type not in the 2-factor in correspondence with a super-edge of the same type that is included in the 2-factor (Figure 33). In accordance with Definition 3, the seven bold edges between nodes A_2 and B_2 in the corresponding large cycle (shown in Figure 34) will be the newly introduced 6-cycle's protected edges.

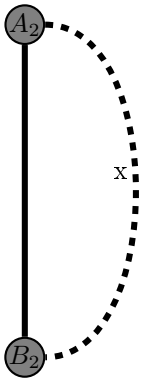

Figure 33: A super-edge in G_i, of the same type as shown in Figure 28, which is included in the 2-factor F_i

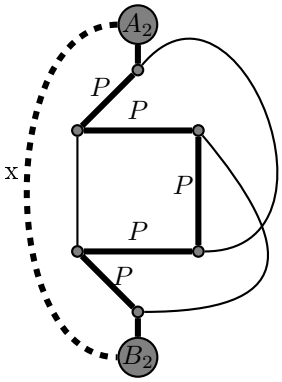

Figure 34: The subgraph in G_{i-1} that replaced the super-edge in Figure 33. The 5 bold edges labeled "P" will be the 6-cycle in Figure 32's protected edges.

4.5 Expanding all other gadgets

A crucial point in the analysis that follows is that only the expansions documented in Sections 4.1-4.4 can introduce organic cycles into F_i that were not present in F_{i+1}. Confirming this fact requires checking all possible ways a 2-factor can pass through each gadget to ensure that all expansions that can introduce organic 6-cycles are properly analyzed. Diagrams documenting every one of these other expansions are shown in the Section 6-12, and a careful examinitaion of these sections confirms that all other expansion operations are not capable of introducing an organic 6-cycle into the 2-factor. These other operations result either in converting one cycle into one larger cycle, converting one cycle into two large (length of at least 8) cycles, or converting two cycles into one or two large cycles. Diagrams documenting every one of these other expansions are shown in the Sections 6-12.

4.6 Protected Edges

We call the edges identified in Definition 3 "protected" because they form organic paths in our 2-factor, so these paths will remain part of the 2-factor regardless of the future expansion operations performed. Now that protected edges are defined, we can establish a correspondence between protected edges and 6-cycles in our 2-factor. Each of the expansion operations that identifies protected edges also introduces a 6-cycle into the 2-factor following this operation. We define the following correspondence between 6-cycles in our final 2-factor (at the end of the algorithm, in the fully expanded graph) and protected edges.

Definition 5. For a given 6-cycle, C, in the final 2-factor, F_0, consider the value i such that C is a cycle of F_i but not of F_{i+1}. Then, it was the expansion operation from G_{i+1} to G_i which "finalized" this cycle. Then, the protected edges identified during this "finalizing" operation are defined to be the **protected edges corresponding to C.**

Every 6-cycle in the final 2-factor will have a set of protected edges it corresponds to, as definied in Definition 5. To see this, observe that our initial 2-factor in the most condensed graph we consider will not have any organic 6-cycles by the construction of our algorithm. Then, if such a cycle exists in the final 2-factor, it necessarily entered the 2-factor through one of the special expansion operations described in this section. Each of these operations identifies protected edges, so these will be the protected edges corresponding to the 6-cycle in question.

Remark 2. During the compression phase of the algorithm, apart from squares, only orgranic subgraphs are replaced with gadgets. Then, each expansion operation, apart from those involving squares, replaces a non-organic gadget with an organic subgraph.

Lemma 3. *If a 6-cycle, C_1, has its corresponding protected edges in another 6-cycle, C_2, then C_2 has 5 corresponding protected edges, and these protected edges are all in a cycle of length at least 8. Furthermore, C_1 and C_2 must have been introduced into the 2-factor during the expansion of a $0-P$ and $1-P$, respectively.*

Proof. In the preceeding subsections of this Section 4, we show all ways an organic 6-cycle can appear in the final 2-factor. The appendices validate this claim, as all possible expansions are examined in detail, and all expansions not included in Section 4 do not produce organic 6-cycles. Immeditely following each of these three special expansion operations, all newly identified protected edges are in cycles of length at least 8.

Then suppose, for the sake of contradiction, that in the final 2-factor, C_1 has a protected edge in another 6-cycle, C_2, but the lemma does not hold. The operation that brought C_1 into the 2-factor F_{i-1} in graph G_{i-1} is one that replaced some gadget in a condensed graph, G_i, with a $0-P$, $1-P$, $2-P$, or a small 2-cut as depicted in Figures 24, 27, 29, and 31, respectively. We now claim that such a situation cannot occur if the expansion operation replaced the gadget with a $1-P$, $2-P$, or a small 2-cut.

Consider the $1-P$ case first, as shown in Figures 27-28. In this case, C_1 is the left cycle in Figure 28 and has five protected edges (the path from A_2 to B_2). Furthermore, we see from Remark 2 that immediately after this expansion the path from A_2 to B_2 in Figure 28 is entirely organic, meaning none of the nodes or edges (including A_2 and B_2) are part of gadgets. The only way C_1's protected edges could end up in a 6-cycle in the final 2-factor is if another special expansion operation split C_2 into two cycles. For this to

happen, however, either two super-vertices which together form a gadget must be a distance of exactly 5 edges away from each other in a cycle or two super-edges must be directly next to each other in a cycle of the 2-factor. However, in this case, C_1's protected edges form an organic path of length 5, which prevents this from occuring, as the two closest possible super-vertices would be A_1 and B_1, which are separated by 7 edges and the two closest possible super-edges would be (A_2, B_1) and (A_1, B_2), which are separated by 5 edges.

Similarly, consider if the expansion operation that first introduced C_1 into the 2-factor replaced a gadget with a $2-P$ or a small 2-cut of the type shown in Figure 32. The same reasoning as in the previous paragraph applies here, too, as C_1 would have at least six protected edges appearing consecutively in a $2-P$ or small 2-cut in these cases. Any corresponding super-vertices or super-edges in this cycle are separated by too many organic edges and nodes to split off some of the protected edges into a new 6-cycle.

The only remaining possibility is that C_1 was introduced through an expansion operation that replaced a gadget with a $0-P$, as shown in Figure 24. In this case, the only way these protected edges could be split off into an organic 6-cycle is if nodes A_1 and B_1 are a gadget. If this gadget was one that replaced a $0-P$, then we see by Remark 1 that the expansion of this gadget cannot introduce an organic 6-cycle into the 2-factor. Then, A_1 and B_1 must have been a $1-P$'s gadget. Thus, C_2 was necessarily introduced into the 2-factor through the expansion of a $1-P$'s gadget, as depicted in Figures 27-28. Then, C_2 has 5 corresponding protected edges of its own. We conclude that these protected edges will be in a cycle of length at least 8 because we demonstrated earlier in this proof that if protected edges are identified from expanding a $1-P$, these protected edges cannot be part of a 6-cycle in the final 2-factor. The final 2-factor does not contain any odd cycles (due to bipartiteness of the original graph) and contains no 4-cycles so we conclude that C_2's protected edges are part of a cycle of length at least 8, proving the lemma. □

Lemma 4. *If a 6-cycle, C_1, has its corresponding protected edges in another 6-cycle, C_2, then C_2's protected edges are in a third cycle, C_3, and either C_3 has length at least 10 or it contains no other protected edges.*

Proof. By Lemma 3, we know that C_2 has 5 protected edges in a third cycle, C_3, of length at least 8. Suppose then, for the sake of contradiction, that this lemma is violated. Then, C_3 contains C_2's 5 protected edges, some set of other protected edges, and has length less than 10. By Lemma 3 and the fact that the graph is bipartite, we conclude that C_3 must have length 8. Protected edges from the same expansion cannot get separated from each other, as there are no super-vertices or super-edges along a path of protected edges, and protected edges come in sets of 3, 4, and 5 edges, depending on the expansion operation. C_3 contains the five protected edges from C_2 as well as another set of protected edges, so this additional set of protected edges must be a set of three edges. This is only possible if the additional set of protected edges were introduced by expanding a $0-P$ gadget, under the circumstances depicted in Figure 23-24. Such an expansion would require the nodes corresponding to A_2 and B_2 in Figure 28 to be a gadget. However, by Lemma 3, A_2 and B_2 must be organic, as they are part of the $1-P$ whose expansion introduced C_2 into the 2-factor and C_2's protected edges into C_3. Since A_2 and B_2 are organic, they cannot be a $0-P$'s gadget, proving that C_3 contains no protected edges other than C_2's protected edges. This contradicts our assumption, proving the lemma. □

4.7 Analyzing the Worst Case

The purpose of the preceeding subsections was to prepare us to compute a lower bound on the average cycle length of our final 2-factor F. We have shown for all of the operations which can produce 6-cycles that there is a way to associate each 6-cycle with some set of protected edges which will remain together in a large cycle for the rest of the algorithm. Then, we can analyze the maximum number of 6-cycles which can be associated with some cycle C of length $k \geq 8$. Before we do this, let's define for each cycle of length at least 8 a set of corresponding 6-cycles (which may be empty):

Definition 6. For any cycle, C_i, of length at least 8, in the 2-factor F we will define a set of 6-cycles, S_{C_i} which correspond to C_i. For each 6-cycle, C, in F, we say C is an element of S_{C_i} iff C's protected edges are in C_i or if C's protected edges are in another 6-cycle C', whose protected edges are in C_i.

Note that Definition 6 is constructed such that each 6-cycle in F corresponds to exactly one large cycle in F. When a set of protected edges are identified, they form an organic path in some cycle of the current 2-factor. Any operation that could separate the protected edges into disjoint cycles would require there to be a super-vertex or super-edge along the path of protected edges, so this cannot happen. This means that each 6-cycle corresponds to at most one large cycle. We also know that each 6-cycle in the final 2-factor is introduced by an expansion operation that identifies corresponding protected edges. From Lemma 3, these protected edges either end up in a large cycle, or they end up in a 6-cycle whose protected edges are in a large cycle. This proves the other side of my claim, that each 6-cycle corresponds to at least one large cycle.

The following proposition will be useful in the upcoming proof of Lemma 5:

Proposition 6. *Suppose there is a cycle C in a preliminary 2-factor F_i which is the union of four edge disjoint paths: P_1, X, P_2, Y. Furthermore, suppose that P_1 and P_2 are organic, P_1 shares its endpoints with endpoints of X and Y, and P_2 shares its endpoints with the remaining endpoints of X and Y. Then the final 2-factor F does not contain a cycle C' which is the union of P_1, e_1, P_2, e_2, where e_1 and e_2 are single edges separating P_1 and P_2 in C', unless both X and Y are single edges.*

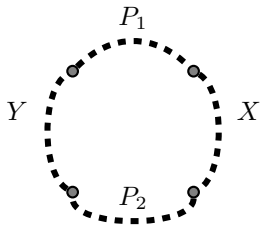

Figure 35: A cycle C, composed of four edge disjoint paths P_1, X, P_2, Y

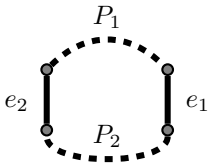

Figure 36: A cycle C', composed of P_1, e_1, P_2, e_2. Proposition 6 proves that if C is a cycle in the computed 2-factor in a compressed graph G_i, then C' will never be a cycle in the final 2-factor F.

Proof. Suppose, for the sake of contradiction, that F contained such a cycle C' where $C \neq C'$. The algorithm does not perform any expansion operations that combine two organic paths in separate cycles into a single cycle connected by two single edges. Then, none of the final i expansions the algorithm performs can separate P_1 and P_2 into two different cycles. $|C| > |C'|$, so, for F to contain C', one of these final i expansions must have shortened at least one of the paths connecting P_1 to P_2 while keeping both P_1 and P_2 in the same segment. The algorithm does not perform any expansion operations on a cycle containing two non-adjacent organic paths resulting in a new, shorter cycle containing both non-adjacent organic paths, which contradicts the assumption that F contains C'. □

Lemma 5. *Every cycle C of length 8 in the final 2-factor F has at most one corresponding 6-cycle.*

Proof. Suppose, for the sake of contradiction, that there exists an 8-cycle C in F which has k corresponding 6-cycles, C_1, C_2, ..., C_k, where $k \geq 2$. By Definition 6, for $1 \leq i \leq k$, C_i has protected edges, which are located either in C or in C_j for some $j \neq i$, $1 \leq j \leq k$.

First, let's consider when there are values i, j such that C_i's protected edges are in C_j. C is of length 8, so by Lemma 4, it must be the case that $k = 2$ and without loss of generality, $i = 1$ and $j = 2$. Additionally, we know due to Lemma 3 that C_1 was introduced through the expansion of a $0 - P$ and C_2 was introduced through the expansion of a $1 - P$.

If the expansion of the $1 - P$ that introduced C_2 into a preliminary 2-factor F_x also introduced C into F_x, then C could not be an 8-cycle, which is a contradiction. To see this, see Figure 28, which describes this class

of expansion operation. If C were an 8-cycle, then the dashed path connecting nodes A_1 and B_1 in this figure is of length 1, but this would mean that the algorithm compressed a $1-P$ at a time when there was a square present in graph, which is also a contradiction. Then, the remaining possibility is that immediately following the expansion of the $1-P$ that introduced C_2 into F_x, the cycle's protected edges were in a non-organic cycle C', of the form shown on the right side of Figure 28. A later expansion operation must introduce C, resulting in the nodes corresponding to A_2 and B_2 in Figure 28 being connected by a path of length 3. This expansion cannot be one where a square is expanded because none of these operations shorten the length of the cycles involved. Then any other potential expansion would contradict the algorithm, as a path of length 3 from A_2 to B_2 would form a square, and the algorithm would have previously contracted a $1-P$, $2-P$, $W2-P$, $SV2-P$, or $SE2-P$ when a square is present in the graph.

We have now ruled out the possiblility of cycle C_i having protected edges in a cycle other than C. By Lemma 1, we know that the protected edges of two cycles C_i and C_j are disjoint. Then, we can easily see that $k < 3$. Each 6-cycle has at least 3 protected edges in C, and C is an 8-cycle, so if $k \geq 3$, then either C would need to have more than 8 edges or the 6-cycles would need to share protected edges, which is not possible.

We must now also show that we obtain a contradiction when $k = 2$. By Lemma 2, we know that any two sets of protected edges do not share any vertices. Then if C contains two sets of protected edges, they must be separated by at least one edge on each side of the cycle. Then, $8 = |C| \geq |P_1| + |P_2| + 2$, where P_1 and P_2 are the protected edges of C_1 and C_2, respectively. $P_i \geq 3$, and if either P_1 or P_2 is at least 4 then $|P_1| + |P_2| + 2 \geq 9$, so the only possibility we need to consider is when $|P_1| = |P_2| = 3$. This would require that both 6-cycles are introduced from expanding $0-P$s. All other possibilities would result in C containing more than 8 protected edges, which is not possible. We now demonstrate that this case results in a contradiction.

We now know that both 6-cycles are introduced through the expansion of two $0-P$s, as all other cases result in a contradiction. The specific expansion of this type that can introduce organic 6-cycles is described in detail in Section 4.1 and shown in Figures 23-24. If P_1 and P_2 are ever contained in different cycles of a preliminary 2-factor F_x, then some expansion operation will eventually bring these two sets of protected edges into the same cycle. However, observe that the algorithm does not perform any expansion operations that combine two organic paths in separate cycles into a single cycle, where the two paths share an endnode or are separated by a single edge on both sides. Then P_1 and P_2 will necessarily be separated by at least two edges on one side and at least one edge on the other side. By Proposition 6, no future expansions could result in P_1 and P_2 being contained in a single 8-cycle. This would contradict the assumptions that C is an 8-cycle and contains two sets of protected edges introduced through the expansion of two $0-P$s.

The other possibility is that the expansion of the second $0-P$ introduces P_2 directly into a cycle that contains P_1. By Remark 1, we know that the nodes corresponding to A_1 and B_1 shown Figure 24 cannot be super-vertices for a $0-P$ whose expansion introduces an organic 6-cycle into a preliminary 2-factor F_x. Then, there must be at least 7 edges between any two super-vertices for a $0-P$ whose expansion introduces an organic 6-cycle into F_x. If A_1 or B_1 or either edge (A_2, B_1) or (B_2, A_1) are non-organic and get expanded before the expansion that introduces the 6-cycle into F_x, then these expansions will replace these non-organic subgraphs with organic subgraphs, so the new nodes in the place of A_1 or B_1 will never be two super-vertices for a $0-P$ whose expansion introduces a 6-cycle into F_x. Then, after the expansion introducing the second 6-cycle, the cycle containing both sets of protected edges in F_x will have at least 10 edges, the 7 edges in between the two super-vertices that replaced the second $0-P$ and the second 6-cycle's set of 3 protected edges. By Proposition 6, no future expansions can result in both P_1 and P_2 being contained in a single 8-cycle. This contradicts the assumptions that C is an 8-cycle and contains two sets of protected edges introduced through the expansion of two $0-P$s.

We have proved that all cases that could result in the existance of an 8-cycle C in the final 2-factor F that has two or more corresponding 6-cycles leads to a contradiction, proving the lemma. □

We are now prepared to prove the next lemma, regarding average cycle length of a large cycle and its set of

corresponding 6-cycles:

Lemma 6. *For any cycle C in 2-factor F of length k such that $k \geq 8$ and its set of corresponding 6-cycles, the average length of this set of cycles is at least 7.*

Proof. First, consider the simple case where C has no corresponding 6-cycles. The set of cycles we are considering in this case is just a single cycle of length $k \geq 8$. $k \geq 8 > 7$, so in this case, the set of cycles has length at least 7.

Now, consider the case when all of the corresponding 6-cycles have their protected edges contained in the large cycle C. To be clear, the only way this condition could be violated is if some 6-cycle has its protected edges in another 6-cycle, whose edges are contained in C. Each of the expansion operations which included one of the corresponding 6-cycles in the 2-factor protects at least 3 edges, and these protected edges are contained in C, so at most $\frac{k}{3}$ 6-cycles can correspond to cycle C. If $k \geq 10$, then the average cycle length among cycle C and its corresponding 6-cycles is at least

$$\frac{\lfloor \frac{k}{3} \rfloor \times 6 + k}{\lfloor \frac{k}{3} \rfloor + 1} \geq 7$$

If $k = 8$ then by Lemma 5, C has at most one corresponding 6-cycle, so the average length of C and its corresponding 6-cycle is also 7.

We must also consider the case when at least one corresponding 6-cycle, C_1, has its protected edges contained in another 6-cycle, C_2.

If $|C| = 8$ and contains C_2's protected edges then C has at least two corresponding 6-cycles, C_1 and C_2. This contradicts Lemma 5.

Next, consider if $|C| = 10$ and C contains C_2's 5 protected edges in the final 2-factor. C cannot contain another set of 5 protected edges, due to Lemma 2, because this would require these protected edges to share a node with C_2's protected edges. Then, in addition to C_1 and C_2, C can have at most one additional corresponding 6-cycle, otherwise C would contain more than 10 protected edges. In this case, C has at most 3 corresponding 6-cycles, so the average length of C and its corresponding 6-cycles is at most $\frac{10+3\times6}{4} = 7$.

The only remaining case is when $|C| \geq 12$ and at least one corresponding 6-cycle, C_1, has its protected edges contained in another 6-cycle, C_2. By Lemma 3, each 6-cycle has at least 3 protected edges in C or its protected edges are in another 6-cycle whose 5 protected edges are in C. So, if a corresponding 6-cycle's protected edges are not in C, then there is another 6-cycle corresponding to C for which these two 6-cycles contribute 5 protected edges to C. Then, each 6-cycle on average contributes at least $\frac{5}{2}$ protected edges to C, so there are at most $\frac{2k}{5}$ 6-cycles corresponding to C. Then, the average cycle length among cycle C and its corresponding 6-cycles is at least

$$\begin{aligned}\frac{\lfloor \frac{2k}{5} \rfloor \times 6 + k}{\lfloor \frac{2k}{5} \rfloor + 1} &\geq \frac{36}{5} \text{ (because } k \geq 12) \\ &> 7\end{aligned}$$

In all possible cases, C and its corresponding 6-cycles have average length of at least 7, which proves the lemma. □

4.8 Main Theorems

Theorem 2. *Given a cubic bipartite graph G with $n > 6$ vertices, there is a polynomial time algorithm that computes a 2-factor with at most $\frac{n}{7}$ cycles*

Proof. In Definition 6, we establish a correspondence between each 6-cycle in the 2-factor computed by $BIGCYCLE$ and another cycle of the 2-factor with length at least 8. Then, we can split the 2-factor into disjoint subsets of cycles by letting $S_C \cup C$ be one of these subsets for each cycle C of length at least 8 in the 2-factor. The average cycle length in the 2-factor is at least as long as the smallest of the average cycle lengths of the subsets of cycles we identified. Lemma 6 establishes that the average length of a cycle each subset of cycles is at least 7, so this is also a lower bound on the average cycle length in the 2-factor. A 2-factor over n vertices with average cycle length at least 7 cannot have more than $\frac{n}{7}$ cycles, otherwise the 2-factor would cover more than n vertices. This proves that $BIGCYCLE$ produces a 2-factor with at most $\frac{n}{7}$ cycles.

It is not difficult to show that the first two steps of $BIGCYCLE$, which compute this 2-factor, runs in polynomial time. Every contraction operation removes at least 2 vertices from the graph, meaning the algorithm performs at most $O(n)$ contractions and expansions. In between each contraction, the algorithm will search for other subgraphs to contract and will compute a 2-factor in the current graph. The algorithm does this search for subgraphs by checking if each node is part of a subgraph that can be contracted. There is a constant neighborhood around each node, v, that needs to be checked to determine if v is part of a subgraph eligible for contraction, so it takes $O(1)$ to check each node in this manner. Computing a 2-factor in a cubic bipartite graph takes $O(n^{\frac{3}{2}})$ in the worst case (Proposition 1). Then, the contraction phase of the algorithm runs in $O(n^{\frac{5}{2}})$ in the worst case.

Each expansion operation requires scanning the cases in the appendices to determine how to update the 2-factor. There are a fixed number of cases to check, so this takes $O(1)$ at each step. After each expansion operation the algorithm also checks the local neighborhood of the location of the expansion to see if the local update shown in Figures 21 and 22 needs to be performed. The size of the neighborhood is constant, so this opearation also takes $O(1)$ at each step. Then, the expansion phase of the algorithm takes $O(n)$ time in the worst case. Therefore, together, the contraction and expansion phases of the algorithm take $O(n^{\frac{5}{2}})$ time in the worst case. It takes $O(n^2)$ to compress all the cycles into super-vertices and then $O(n^2)$ to find a spanning tree, so the entire algorithm runs in $O(n^{\frac{5}{2}})$ in the worst case. □

Theorem 3. *Given a cubic bipartite G with n vertices, there is a polynomial time algorithm that computes a spanning Eulerian multigraph H in G with at most $\frac{9}{7}n - 2$ edges.*

Proof. If $n = 6$ then G is a $K_{3,3}$. We can find a Hamiltonian cycle in a $K_{3,3}$ in constant time and is a solution with n edges, satisfying the theorem. For the remainder of the proof we will assume that $n > 6$. Theorem 2 proves that steps 1 and 2 of $BIGCYCLE$ (Algorithms 4 and 5, respectively) produces a 2-factor with at most $\frac{n}{7}$ cycles for the required class of graphs. Proposition 2 demonstrates that Step 3 of $BIGCYCLE$ successfully extends the 2-factor into a spanning Eulerian multigraph with at most $\frac{9}{7}n - 2$ edges.
It is easy to see that this algorithm runs in polynomial time. We know from Theorem 2 that we can compute the required 2-factor in $O(n^2)$. Once we have done this, we can compute the compressed graph in $O(n)$. A minimum spanning tree is then computed, doubled, and added to form our multigraph in $O(n)$ as well, as the original graph (and the compressed graph as well) has $O(n)$ edges. The total running time of the algorithm, then, is $O(n^3)$ in the worst case. □

5 An Extension to k-regular bipartite graphs

In this section, we demonstrate how the $BIGCYCLE$ algorithm can be used as a subroutine to produce an improved approximation algorithm for k-regular bipartite graphs. The main idea in this algorithm is that k-regular bipartite graphs contain cubic subgraphs on which we can run $BIGCYCLE$ to obtain solutions to the cubic subgraphs, which will also be solutions to the original k-regular bipartite graphs. If the cubic subgraph we find is composed entirely of connected components of size 8 and larger, then we will get a solution with at most $\frac{9}{7}n - 2$ edges. However, if some of the components are of size 6 (in a cubic bipartite graph these will be $K_{3,3}$s), then the 2-factor we compute may have between $\frac{n}{6}$ and $\frac{n}{7}$ cycles, which gives us a solution of size x where $\frac{9}{7}n - 2 \leq x \leq \frac{4}{3}n - 2$. Algorithm 6 provides the pseudo-code for selecting cubic subgraph from a k-regular bipartite graph containing a small number of $K_{3,3}$s. In the analysis that follows,

we will bound the number of $K_{3,3}$s in the cubic subgraph computed by Algorithm 6, allowing us to prove a specific approximation factor. The subroutine CountK33, used in Algorithm 6, takes a graph as input and returns the number of connected components of the graph that are $K_{3,3}$s.

Algorithm 9 An algorithm to find a cubic subgraph from a k-regular bipartite graph: CUBIC

```
Input: A connected, undirected, unweighted, k-regular, bipartite graph, G_0 =
(V, E)
For i = 1 to k:
  M_i ←FindPerfectMatching(G_{i-1})
  G_i ← G_{i-1}\M_i
  If i = 3:
    G_cubic ← (V, M_1 ∪ M_2 ∪ M_3)
  If i > 3:
    G_temp ← (V, M_1 ∪ M_2 ∪ M_i)
    If CountK33(G_temp)<CountK33(G_cubic):
      G_cubic ← G_temp
End Loop
Return G_cubic
```

Lemma 7. *For any k-regular bipartite graph where $k \geq 4$, G, at most $\frac{n}{6(k-2)}$ $K_{3,3}$s are contained in CUBIC(G), the cubic bipartite graph outputted by Algorithm 6.*

Proof. Consider an arbitrary k-regular bipartite graph G. Note that the nodes of any 6-cycle in $(V, M_1 \cup M_2)$ will form a $K_{3,3}$ in $(V, M_1 \cup M_2 \cup M_i)$ for at most 1 value of i because the k matchings $M_1, \ldots, M_k$ are edge-disjoint. There are at most $\frac{n}{6}$ 6-cycles in $M_1 \cup M_2$. In the worst case, the nodes of each of these 6-cycles can form a $K_{3,3}$ in $(V, M_1 \cup M_2 \cup M_i)$ for exactly one value of i where $3 \leq i \leq k$. Then, by the pigeonhole principle, there is some value i where $3 \leq i \leq k$ where $(V, M_1 \cup M_2 \cup M_i)$ contains at most $\frac{\frac{n}{6}}{k-2} = \frac{n}{6(k-2)}$ $K_{3,3}$s. Algorithm 6 finds M_i where $(V, M_1 \cup M_2 \cup M_i)$ contains the fewest $K_{3,3}$s, so the algorithm will necessarily output three edge-disjoint matchings with at most $\frac{n}{6(k-2)}$ $K_{3,3}$s, proving the lemma. □

Theorem 4. *Given a k-regular bipartite G with n vertices where $k \geq 4$, there is a polynomial time algorithm that computes a spanning Eulerian multigraph H in G with at most $(\frac{9}{7} + \frac{1}{21(k-2)})n - 2$ edges.*

Proof. First, we will find a cubic subraph of G, G_{cubic}, by running Algorithm 6 on G. In each component of G_{cubic} that is a $K_{3,3}$ we will find a 6-cycle covering these nodes. This can be done in constant time by taking any walk through this component that does not visit a node twice as long as this is possible, then returning to the first node. In every other connected component, run the Contract and Expand phases of the $BIGCYCLE$ algorithm, which will find a 2-factor over this component containing at most $\frac{n_i}{7}$ cycles, where n_i is the number of nodes in the connected component. The upper bound of $\frac{n_i}{7}$ cycles is proven by Theorem 2. By Lemma 7, there are x_1 nodes in $K_{3,3}$s within G_{cubic}, where $x_1 \leq \frac{n}{k-2}$. These nodes are covered by $\frac{x_1}{6}$ cycles. Then, there x_2 nodes in the remaining components and $x_1 + x_2 = n$. These x_2 nodes are covered by at most $\frac{x_2}{7}$ cycles. Then, the overall 2-factor of G has at most $\frac{x_1}{6} + \frac{x_2}{7}$ cycles. The following calculations compute an upper bound on these cycles in terms of n:

$$\begin{aligned}
\frac{x_1}{6} + \frac{x_2}{7} &= \frac{7x_1 + 6x_2}{42} \\
&= \frac{6n + x_1}{42} \\
&= \frac{n}{7} + \frac{x_1}{42} \\
&\leq \frac{n}{7} + \frac{n}{42(k-2)}
\end{aligned}$$

By Proposition 2, this 2-factor can be extended into a spanning Eulerian multigraph in G with at most $n + 2(\frac{n}{7} + \frac{n}{42(k-2)} - 1) = (\frac{9}{7} + \frac{1}{21(k-2)})n - 2$ edges, proving the theorem. □

6 Appendix A: Squares

The purpose of the appendices is to demonstrate in detail how the algorithm *BIGCYCLE* "winds" a 2-factor through the gadgets (4-cycles, $0-P$s, $1-P$s, $2-P$s, and $W2-P$s) as it expands the condensed graph back to its original state. Appendix A contains information on winding the 2-factor through square gadgets. Appendix B contains this information for $0-P$s, and $1-P$s, $2-P$s, and $W2-P$s are described in Appendices C, D, and E, respectively.

6.1 Gadget is covered by three edges of a single cycle

If a gadget that replaced a square is covered by two disjoint cycles of F_i, then the internal edge of the gadget must be included from F_i. Then, F_i must include either edge 1 or 3 and either edge 2 or 4. However, while there are four orientations to consider, they are all symmetric to each other, so there is only one case to consider. In this case, we start with a cycle of length $x+3$ in F_i and are returned a single cycle of length $x+5$ in F_{i-1}. $x+3 \geq 6$, so $x+5 \geq 8$, meaning this class of expansions cannot introduce an organic 6-cycle into the 2-factor.

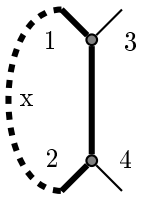

Figure 37: A cycle of length $x+3$ that passes through a gadget that replaced a square.

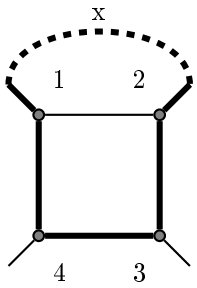

Figure 38: The cycle from the previous figure, after expanding the gadget, is now of length $x+5$.

6.2 Gadget is covered by two cycles

If a gadget that replaced a square is covered by two disjoint cycles of F_i, then the internal edge of the gadget must be excluded from F_i. Then, there is only one possible orientation in which F_i could cover the nodes of this gadget. In this case, we start with cycles of lengths $x+2$ and $y+2$ in F_i and are returned a single cycle of length $x+y+6$ in F_{i-1}. $x+2 \geq 6$ and $y+2 \geq 6$, so $x+y+6 > 8$, meaning this class of expansions cannot introduce an organic 6-cycle into the 2-factor.

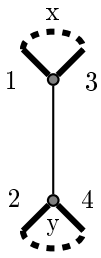

Figure 39: Two cycles of lengths $x+2$ and $y+2$ that pass through a gadget that replaced a square.

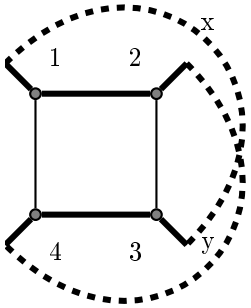

Figure 40: The cycles from the previous figure, after expanding the gadget, now form a single cycle of length $x+y+6$.

6.3 Gadget is covered by four edges of a single cycle

If a gadget that replaced a square is covered by four edges of a single cycle of F_i, then there are two orientations in which F_i could pass through these edges. However, these two cases are symmetric, so there is only one case to consider. In this case, we start with a cycle of length $x + y + 4$ in F_i and are returned a single cycle of length $x + y + 6$ in F_{i-1}. $x + y + 4 \geq 6$, so $x + y + 6 \geq 8$, meaning this class of expansions cannot introduce an organic 6-cycle into the 2-factor.

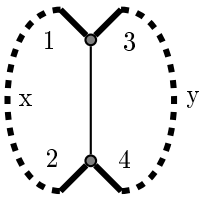

Figure 41: A cycle of length $x+y+4$ that passes through a gadget that replaced a square.

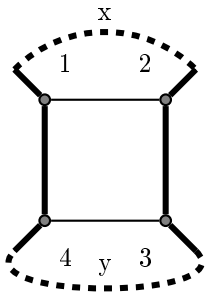

Figure 42: The cycle from the previous figure, after expanding the gadget, is now of length $x + y + 6$.

6.4 Super-vertex replaces a square

A super-vertex that replaced a square is necessarily covered by F_i. Then, there are three ways we can select the edge in each of the super-vertices to exclude from F_i. However, the cases where edges 2 and 3 are excluded from F_i are symmetric so we will examine only the second of these cases. Then, there are only two case to consider. In all cases, we start with a cycle of length $x + 2$ in F_i and are returned a single cycle of length $x + 6$ in F_{i-1}. $x + 2 \geq 6$, so $x + 6 > 8$, meaning this class of expansions cannot introduce an organic 6-cycle into the 2-factor.

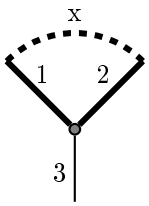

Figure 43: A cycle of length $x + 2$ that passes through a gadget that replaced a square.

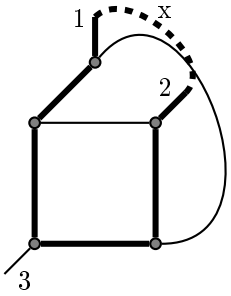

Figure 44: The cycle from the previous figure, after expanding the gadget, is now of length $x+6$.

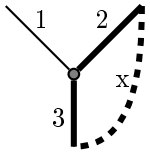

Figure 45: A cycle of length $x + 2$ that passes through a gadget that replaced a square.

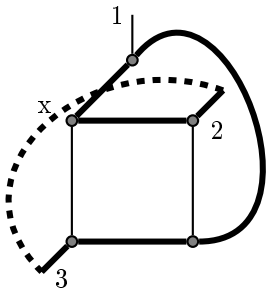

Figure 46: The cycle from the previous figure, after expanding the gadget, is now of length $x+6$.

6.5 Super-edge replaces a square

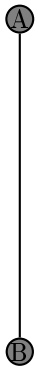

Figure 47: The super-edge is not included in the 2-factor.

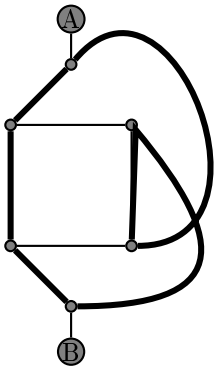

Figure 48: A cycle of length 6 passes through the square after the super-edge in the previous figure is expanded. The impact of these 6-cycles on the algorithm's result is analyzed in Sections 4.4 and 4.6-7.

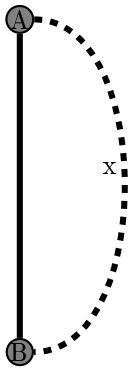

Figure 49: A cycle of length $x + 1$ that passes through a gadget that replaced a square.

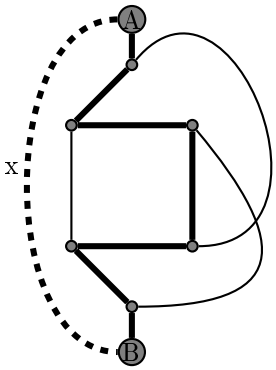

Figure 50: The cycle from the previous figure, after expanding the gadget, is now of length $x+7$.

7 Appendix B: $0 - P$s

7.1 Gadget is covered by two cycles

If a $0 - P$ gadget is covered by two disjoint cycle in F_i, then, there are three ways we can select the edge in each of the super-vertices to exclude from F_i. However, without loss of generality, we can fix the edge of the first super-vertex to exclude from F_i. Then, there are only three cases to consider. In each of these cases, we start with cycles of lengths $x+2$ and $y+2$ in F_i and are returned either a single cycle of length $x+y+8$ or two cycles of lengths $x+4$ and $y+4$ in F_{i-1}. $x+2 \geq 6$ and $y+2 \geq 6$, so $x+y+8 > 8$, meaning the first case cannot introduce an organic 6-cycle into the 2-factor. Similarly, we conclude in the second case that $x+4 \geq 8$ and $y+4 \geq 8$ so neither the $x+4$ or $y+4$ cycles can be organic 6-cycles.

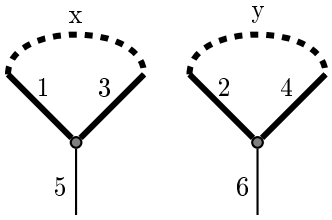

Figure 51: Two cycles of lengths $x+2$ and $y+2$ that pass through a gadget that replaced a $0-P$.

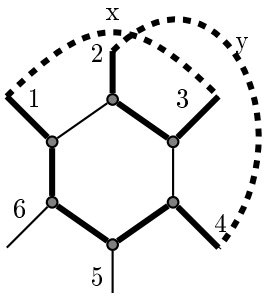

Figure 52: The cycles from the previous figure, after expanding the gadget, now form a single cycle of length $x + y + 8$.

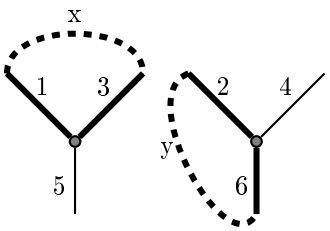

Figure 53: Two cycles of lengths $x+2$ and $y+2$ that pass through a gadget that replaced a $0-P$.

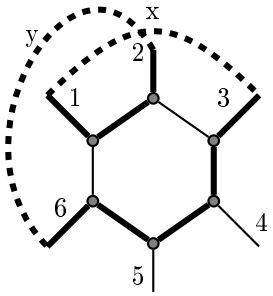

Figure 54: The cycles from the previous figure, after expanding the gadget, now form a single cycle of length $x+y+8$.

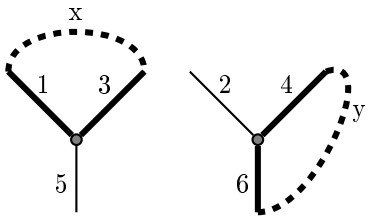

Figure 55: Two cycles of lengths $x+2$ and $y+2$ that pass through a gadget that replaced a $0-P$.

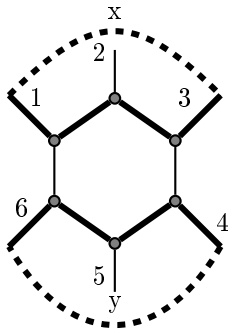

Figure 56: The cycles from the previous figure, after expanding the gadget, now form two cycles of lengths $x+4$ and $y+4$, respectively.

7.2 Gadget is covered by one cycle

If a $0-P$ gadget is covered by a single cycle in F_i, then, there are three ways we can select the edge in each of the super-vertices to exclude from F_i. However, without loss of generality, we can fix the edge of the first super-vertex to exclude from F_i. Then, in each of these three configurations, we examine the two orientations in which the cycle can pass through the two super-vertices. In all 6 cases, we start with a cycle of lengths $x+y+4$ in F_i and are returned either a single cycle of length $x+y+8$, two cycles of lengths $x+3$ and $y+5$, or two cycles of lengths $x+5$ and $y+3$ in F_{i-1}. $x+y+4 \geq 6$, so $x+y+8 > 8$, meaning the first case cannot introduce an organic 6-cycle into the 2-factor. In the later two cases the $x+3$ or $y+3$ cycle can be an organic 6-cycle, but this is the expansion examined in detail in Sections 4.1 and 4.6-7.

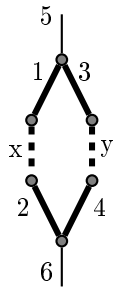

Figure 57: A cycle of length $x+y+4$ that passes through a gadget that replaced a $0-P$.

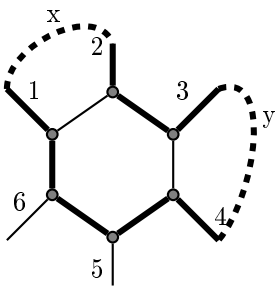

Figure 58: The cycle from the previous figure, after expanding the gadget, is now of length $x+y+8$.

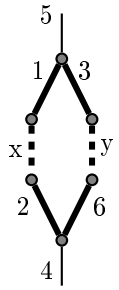

Figure 59: A cycle of length $x+y+4$ that passes through a gadget that replaced a $0-P$.

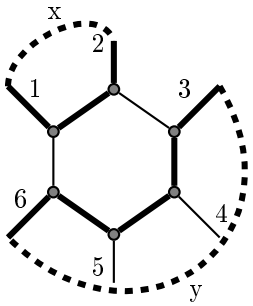

Figure 60: The cycle from the previous figure, after expanding the gadget, is now two cycles, of lengths $x+3$ and $y+5$, respectively. This expansion can produce an organic 6-cycle if $x=3$ and the cycle is organic. The impact of these 6-cycles on the algorithm's result is analyzed in Sections 4.1 and 4.6-7.

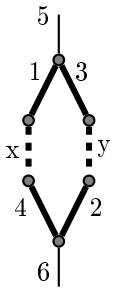

Figure 61: A cycle of length $x+y+4$ that passes through a gadget that replaced a $0-P$.

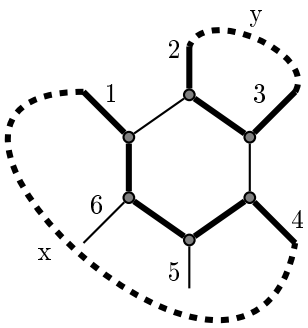

Figure 62: The cycle from the previous figure, after expanding the gadget, is now two cycles, of lengths $y+3$ and $x+5$, respectively. This expansion can produce an organic 6-cycle if $y=3$ and the cycle is organic. The impact of these 6-cycles on the algorithm's result is analyzed in Sections 4.1 and 4.5-6.

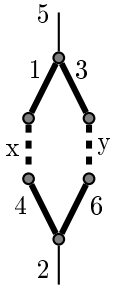

Figure 63: A cycle of length $x+y+4$ that passes through a gadget that replaced a $0-P$.

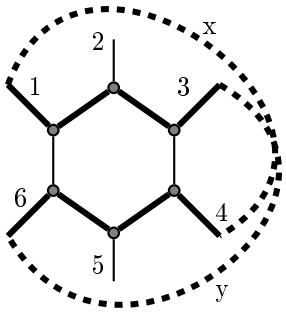

Figure 64: The cycle from the previous figure, after expanding the gadget, is now of length $x+y+8$.

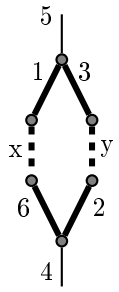

Figure 65: A cycle of length $x+y+4$ that passes through a gadget that replaced a $0-P$.

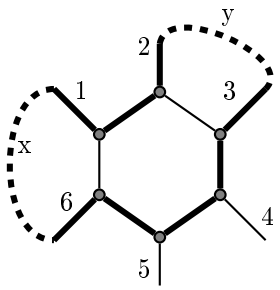

Figure 66: The cycle from the previous figure, after expanding the gadget, is now of length $x+y+8$.

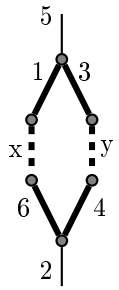

Figure 67: A cycle of length $x+y+4$ that passes through a gadget that replaced a $0-P$.

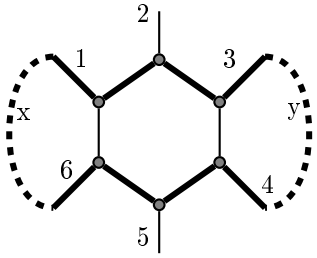

Figure 68: The cycle from the previous figure, after expanding the gadget, is now of length $x+y+8$.

8 Appendix C: $1-P$s

8.1 Gadget is covered by two cycles

If a $1-P$ gadget is covered by a two cycles in F_i, then, there are three ways we can select the edge in each of the super-vertices to exclude from F_i. However, without loss of generality, we can fix the edge of the first super-vertex to exclude from F_i. Then, we only have to consider three cases. In each of these cases, we start with two cycles of lengths $x+2$ and $y+2$ in F_i and are returned a single cycle of length $x+y+10$ in F_{i-1}. $x+2 \geq 6$ and $y+2 \geq 6$, so $x+y+10 > 8$, meaning that none of these expansions can introduce an organic 6-cycle into the 2-factor.

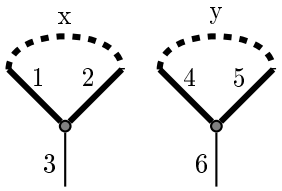

Figure 69: Two cycles of lengths $x+2$ and $y+2$ pass through a gadget that replaced a $1-P$.

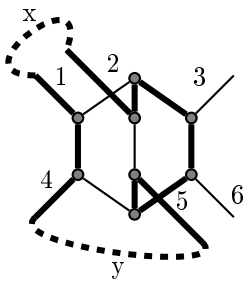

Figure 70: The cycles from the previous figure, after expanding the gadget, are now a single cycle of length $x+y+10$.

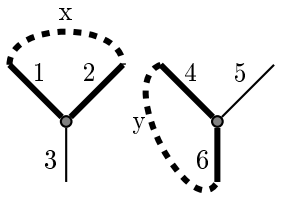

Figure 71: Two cycles of lengths $x+2$ and $y+2$ pass through a gadget that replaced a $1-P$.

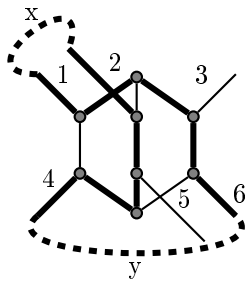

Figure 72: The cycles from the previous figure, after expanding the gadget, are now a single cycle of length $x+y+10$.

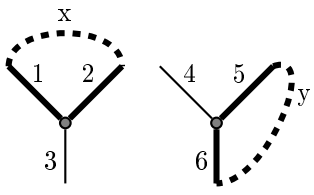

Figure 73: Two cycles of lengths $x+2$ and $y+2$ pass through a gadget that replaced a $1-P$.

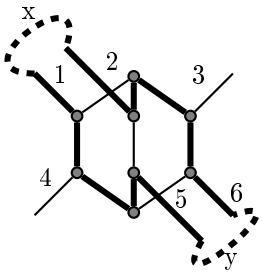

Figure 74: The cycles from the previous figure, after expanding the gadget, are now a single cycle of length $x+y+10$.

8.2 Gadget is covered by one cycle

If a $1-P$ gadget is covered by a single cycle in F_i, then, there are three ways we can select the edge in each of the super-vertices to exclude from F_i. However, without loss of generality, we can fix the edge of the first super-vertex to exclude from F_i. Then, in each of these three configurations, we examine the two orientations in which the cycle can pass through the two super-vertices. In all 6 cases, we start with a cycle of lengths $x+y+4$ in F_i and are returned a single cycle of length $x+y+10$, two cycles of lengths $x+5$ and $y+5$, or two cycles of lengths $x+3$ and $y+7$ in F_{i-1}. $x+y+4 \geq 6$, so $x+y+10 > 8$, meaning the first case cannot introduce an organic 6-cycle into the 2-factor. In the second case, neither the $x+5$ and $y+5$ cycles can be organic 6-cycles, otherwise the $1-P$ the gadget replaced would have been part of an organic $2-P$, which would have been contracted instead of the $1-P$. In the third case, the $x+3$ cycle can be an organic 6-cycle, but this is the expansion examined in detail in Sections 4.2 and 4.6-7.

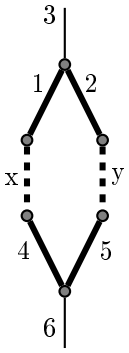

Figure 75: A cycle of length $x+y+4$ passes through a gadget that replaced a $1-P$.

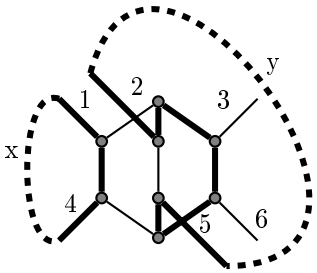

Figure 76: The cycle from the previous figure, after expanding the gadget, is now two cycles, of lengths $x+3$ and $y+7$, respectively. This expansion can produce an organic 6-cycle if $x=3$ and the cycle is organic. The impact of these 6-cycles on the algorithm's result is analyzed in Sections 4.2 and 4.6-7.

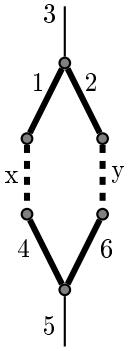

Figure 77: A cycle of length $x+y+4$ passes through a gadget that replaced a $1-P$.

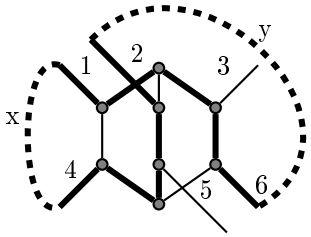

Figure 78: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x+y+10$.

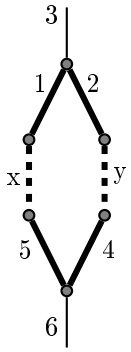

Figure 79: A cycle of length $x + y + 4$ passes through a gadget that replaced a $1 - P$.

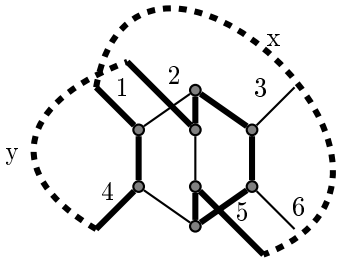

Figure 80: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x + y + 10$.

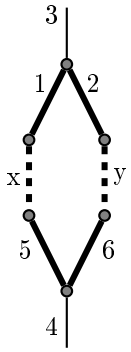

Figure 81: A cycle of length $x + y + 4$ passes through a gadget that replaced a $1 - P$.

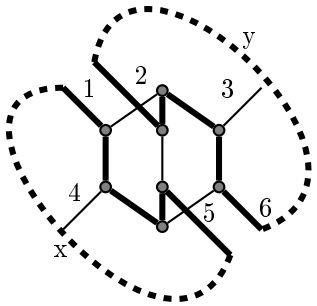

Figure 82: The cycle from the previous figure, after expanding the gadget, is now two cycles, of lengths $x+5$ and $y+5$, respectively. Note that x and y cannot be 1, otherwise this $1-P$ would be part of a $2-P$. This is not possible, otherwise the $2-P$ would have been contracted instead of this $1-P$.

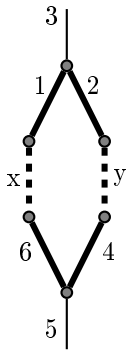

Figure 83: A cycle of length $x + y + 4$ passes through a gadget that replaced a $1 - P$.

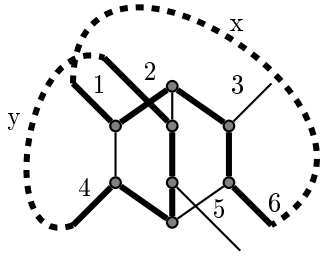

Figure 84: The cycle from the previous figure, after expanding the gadget, is now two cycles, of lengths $x+5$ and $y+5$, respectively. Note that x and y cannot be 1, otherwise this $1-P$ would be part of a $2 - P$. This is not possible, otherwise the $2 - P$ would have been contracted instead of this $1 - P$.

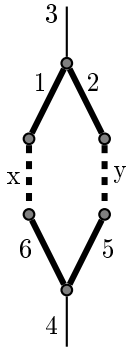

Figure 85: A cycle of length $x + y + 4$ passes through a gadget that replaced a $1 - P$.

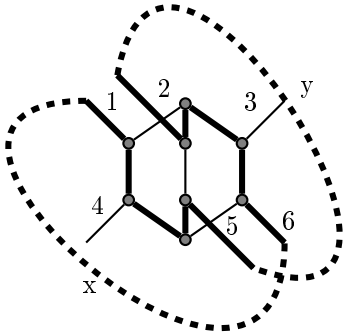

Figure 86: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x + y + 10$.

9 Appendix D: $2 - P$s

9.1 Zero super-edges are covered by 2-factor

If none of the super-edges of a $2 - P$ gadget are covered by F_i, then expanding this gadget returns a cycle of length 10.

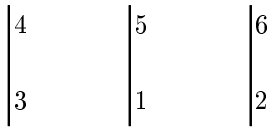

Figure 87: None of the super-edges in the gadget that replaced a $2-P$ are included in the 2-factor.

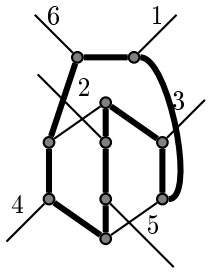

Figure 88: After expanding the gadget, the $2-P$ is covered by a cycle of length 10.

9.2 One super-edge is covered by 2-factor

If exactly one edge of a $2-P$ gadget is covered by a cycle of F_i, then, to ensure we examine every case, we examine all three ways we can select the super-edge to include in F_i. In all three cases, we start with a cycle of length $x+1$ in F_i and are returned either a single cycle of length $x+11$ in F_{i-1}. $x+1 \geq 6$, so $x+11 \geq 8$, meaning the resulting cycle in F_{i-1} cannot be a cycle of length 6.

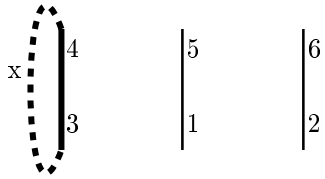

Figure 89: A cycle of length $x+1$ passes through a gadget that replaced a $2-P$.

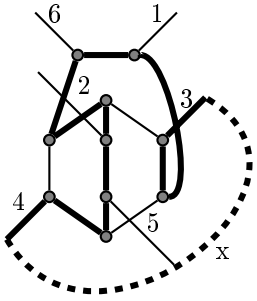

Figure 90: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x+11$.

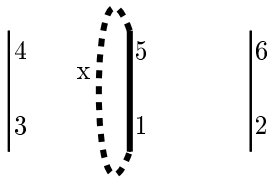

Figure 91: A cycle of length $x+1$ passes through a gadget that replaced a $2-P$.

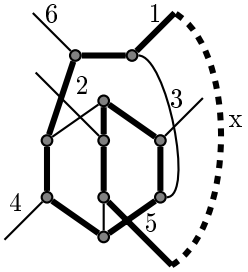

Figure 92: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x+11$.

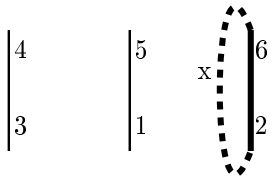

Figure 93: A cycle of length $x+1$ passes through a gadget that replaced a $2-P$.

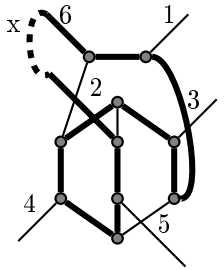

Figure 94: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x+11$.

9.3 Two super-edges are covered by 2-factor

9.3.1 Two cycles pass through gadget

If two edges of a $2-P$ gadget are covered by two disjoint cycles in F_i, then, to ensure we examine every case, we examine all three ways we can select the super-edge to exclude from F_i. In all three cases, we start with cycles of lengths $x+1$ and $y+1$ in F_i and are returned either a single cycle of length $x+y+12$ or two cycles of lengths $x+5$ and $y+7$ in F_{i-1}. $x+1 \geq 6$, and $y+1 \geq 6$, so $x+5 \geq 8$, $y+7 \geq 8$, and $x+y+12 \geq 8$, meaning the resulting cycle or cycles in F_{i-1} cannot be a cycle of length 6.

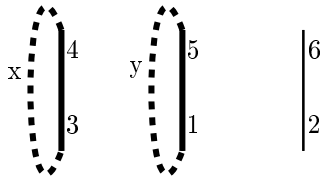

Figure 95: Two cycles of lengths $x+1$ and $y+1$ pass through a gadget that replaced a $2-P$.

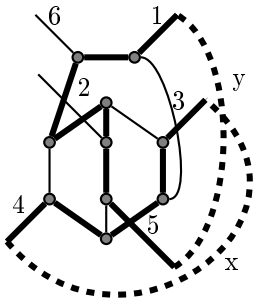

Figure 96: The cycles from the previous figure, after expanding the gadget, are now cycles of lengths $x+5$ and $y+7$, respectively.

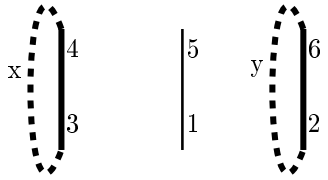

Figure 97: Two cycles of lengths $x+1$ and $y+1$ pass through a gadget that replaced a $2-P$.

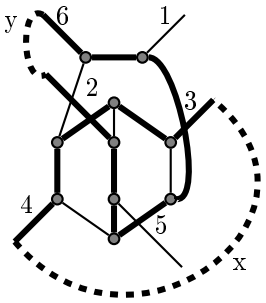

Figure 98: The cycles from the previous figure, after expanding the gadget, are now cycles of lengths $x+5$ and $y+7$, respectively.

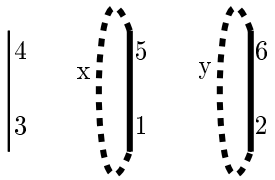

Figure 99: Two cycles of lengths $x+1$ and $y+1$ pass through a gadget that replaced a $2-P$.

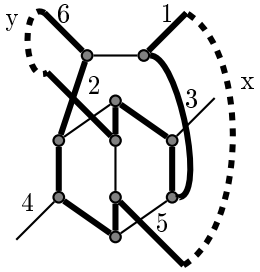

Figure 100: The cycles from the previous figure, after expanding the gadget, are now a single cycle of length $x+y+12$.

9.3.2 One cycle passes through gadget

If two edges of a $2-P$ gadget are covered by a single cycle in F_i, then, to ensure we examine every case, we examine all three ways we can select the super-edge to exclude from F_i and then within each of these arrangements, we examine both orientations in which the two super-edges in the same cycle can be connected. In all six cases, we start with a cycle of length $x+y+2$ in F_i and are returned a single cycle of length $x+y+12$ in F_{i-1}, except for the expansion shown in Figures 111-112, which is analyzed in detail in Sections 4.3 and 4.6-7. $x+y+2 \geq 6$, so $x+y+12 > 8$, meaning the resulting cycle in F_{i-1} cannot be a cycle of length 6, except for in the special case we have identified.

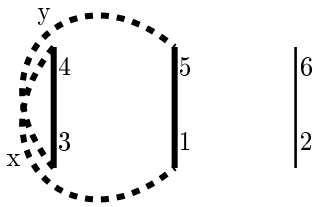

Figure 101: A cycle of length $x+y+2$ passes through a gadget that replaced a $2-P$.

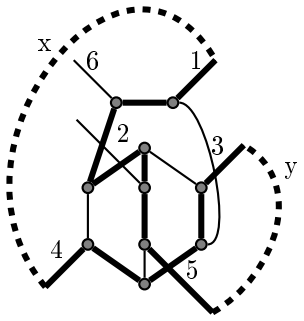

Figure 102: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x+y+12$.

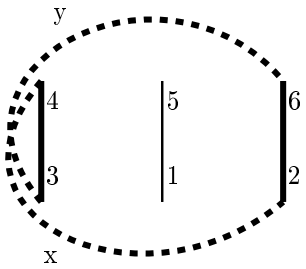

Figure 103: A cycle of length $x + y + 2$ passes through a gadget that replaced a $2 - P$.

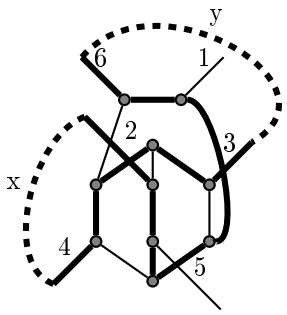

Figure 104: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x + y + 12$.

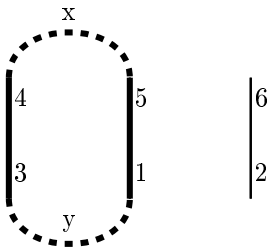

Figure 105: A cycle of length $x + y + 2$ passes through a gadget that replaced a $2 - P$.

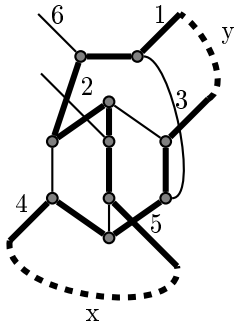

Figure 106: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x + y + 12$.

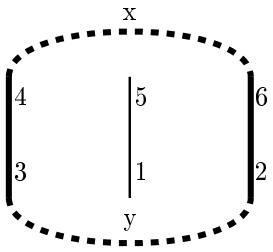

Figure 107: A cycle of length $x + y + 2$ passes through a gadget that replaced a $2 - P$.

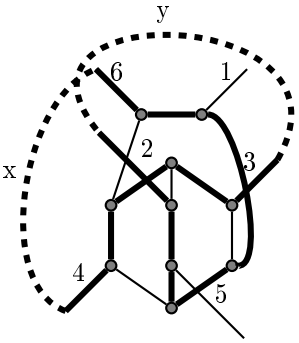

Figure 108: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x + y + 12$.

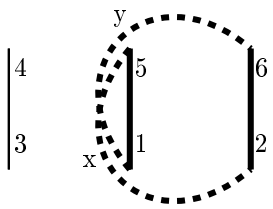

Figure 109: A cycle of length $x + y + 2$ passes through a gadget that replaced a $2 - P$.

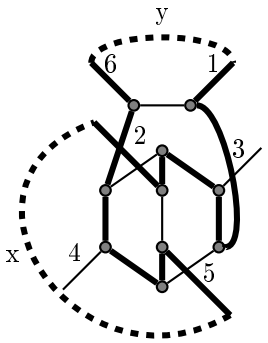

Figure 110: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x + y + 12$.

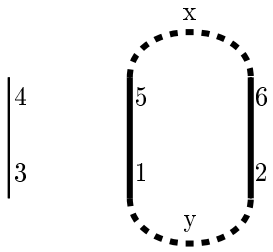

Figure 111: A cycle of length $x + y + 2$ passes through a gadget that replaced a $2 - P$.

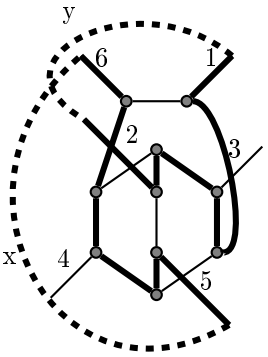

Figure 112: The cycle from the previous figure, after expanding the gadget, is now two cycles of lengths $x + 6$ and $y + 6$, respectively. Both x and y cannot have length 0, otherwise this $2 - P$ would be part of a $W2 - P$. This expansion can produce an organic 6-cycle if either x or y is a 0-length path and the cycle is organic. The impact of these 6-cycles on the algorithm's result is analyzed in Sections 4.3 and 4.6-7.

9.4 Three super-edges are covered by 2-factor

9.4.1 Three cycles pass through gadget

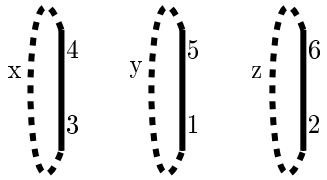

Figure 113: Three cycles of lengths $x+1$, $y+1$, and $z+1$ pass through a gadget that replaced a $2-P$.

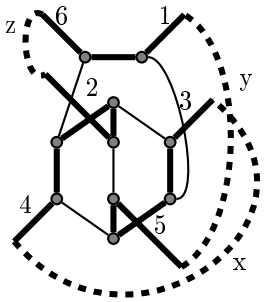

Figure 114: The cycles from the previous figure, after expanding the gadget, are now a single cycle of length $x+y+z+13$.

9.4.2 Two cycles pass through gadget

If a $2-P$ gadget is covered by two disjoint cycles in F_i, then two of the super-edges must be part of the same cycle, with the third super-edge in a different cycle. To ensure we examine every case, we examine all three ways we can select the super-edge to appear in a separate cycle and then within each of these arrangements, we examine both orientations in which the two super-edges in the same cycle can be connected. In all cases we start with cycles of lengths $x+y+2$ and $z+1$ in F_i and are returned a single cycle of length $x+y+z+13$ in F_{i-1}. $x+y+2 \geq 6$ and $z+1 \geq 6$, so $x+y+z+13 > 8$, meaning the resulting cycle in F_{i-1} cannot be a cycle of length 6.

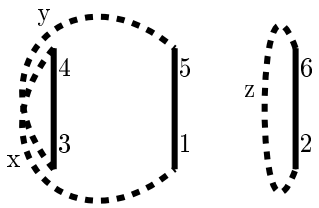

Figure 115: Two cycles of lengths $x+y+2$ and $z+1$ pass through a gadget that replaced a $2-P$.

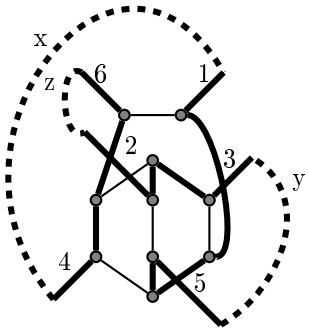

Figure 116: The cycles from the previous figure, after expanding the gadget, are now a single cycle of length $x+y+z+13$.

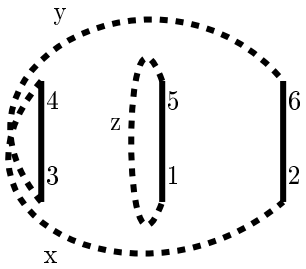

Figure 117: Two cycles of lengths $x + y + 2$ and $z+1$ pass through a gadget that replaced a $2-P$.

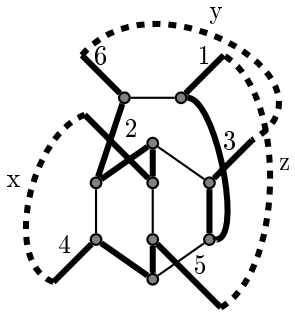

Figure 118: The cycles from the previous figure, after expanding the gadget, are now a single cycle of length $x + y + z + 13$.

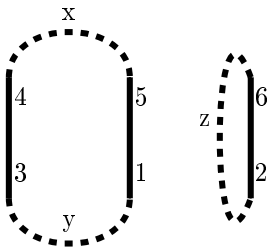

Figure 119: Two cycles of lengths $x + y + 2$ and $z+1$ pass through a gadget that replaced a $2-P$.

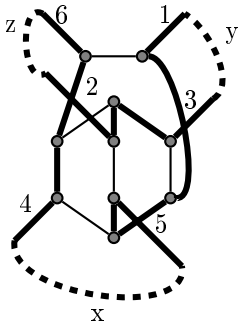

Figure 120: The cycles from the previous figure, after expanding the gadget, are now a single cycle of length $x + y + z + 13$.

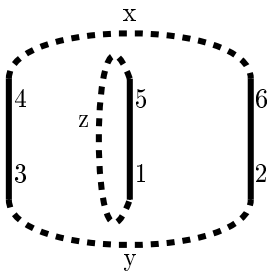

Figure 121: Two cycles of lengths $x + y + 2$ and $z+1$ pass through a gadget that replaced a $2-P$.

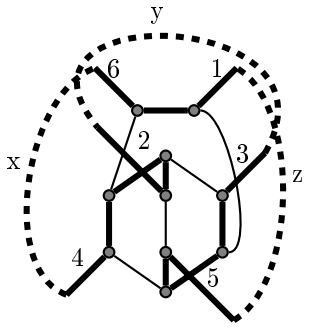

Figure 122: The cycles from the previous figure, after expanding the gadget, are now a single cycle of length $x + y + z + 13$.

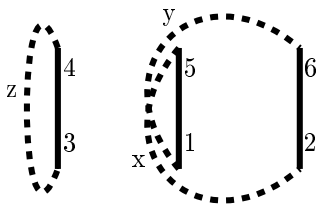

Figure 123: Two cycles of lengths $x + y + 2$ and $z+1$ pass through a gadget that replaced a $2-P$.

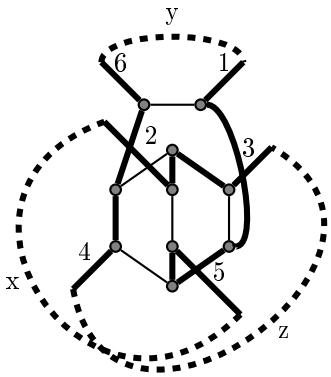

Figure 124: The cycles from the previous figure, after expanding the gadget, are now a single cycle of length $x + y + z + 13$.

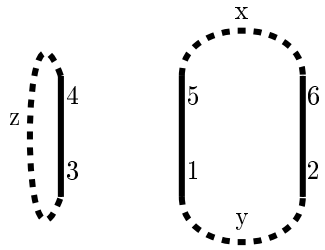

Figure 125: Two cycles of lengths $x + y + 2$ and $z+1$ pass through a gadget that replaced a $2-P$.

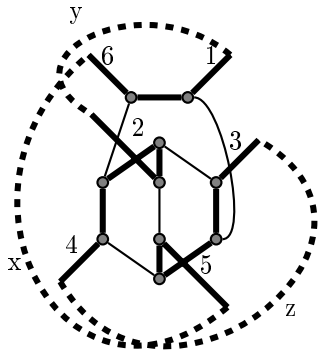

Figure 126: The cycles from the previous figure, after expanding the gadget, are now a single cycle of length $x + y + z + 13$.

9.4.3 One cycle passes through gadget

If a $2 - P$ gadget is covered by a single cycle in F_i, then without loss of generality, we can assume the cycle passes through the (4,3) super-edge first. To ensure we examine every case, we examine all four sides of the two remaining super-edges the 2-factor could enter after exiting edge 4. Then, after exiting the other side of this second super-edge, we consider the two orientations in which the 2-factor could enter the third super-edge. In total, this gives us eight cases to consider. In all cases we start with a cycle of lengths $x + y + z + 3$ in F_i and are returned a single cycle of length $x + y + z + 13$ in F_{i-1}. $x + y + z + 3 \geq 6$, so $x + y + z + 13 > 8$, meaning the resulting cycle in F_{i-1} cannot be a cycle of length 6.

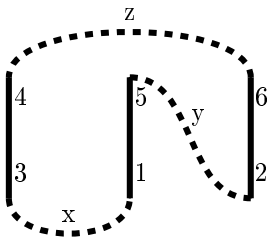

Figure 127: A cycle of length $x+y+z+3$ passes through a gadget that replaced a $2 - P$.

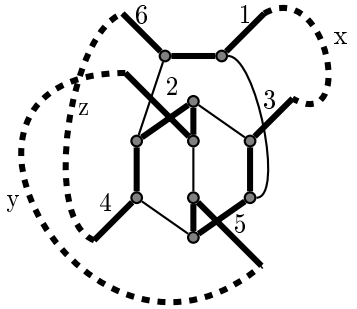

Figure 128: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x + y + z + 13$.

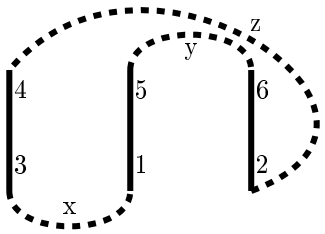

Figure 129: A cycle of length $x+y+z+3$ passes through a gadget that replaced a $2-P$.

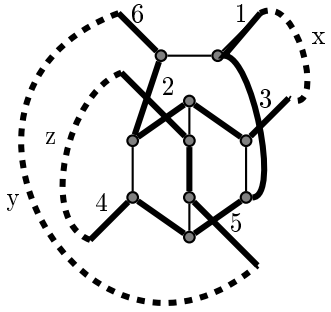

Figure 130: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x+y+z+13$.

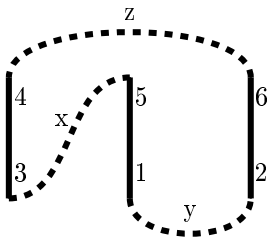

Figure 131: A cycle of length $x+y+z+3$ passes through a gadget that replaced a $2-P$.

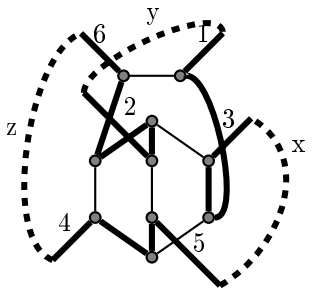

Figure 132: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x+y+z+13$.

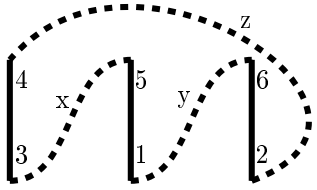

Figure 133: A cycle of length $x+y+z+3$ passes through a gadget that replaced a $2-P$.

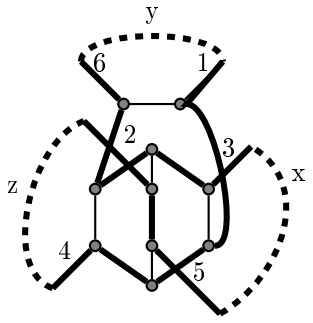

Figure 134: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x+y+z+13$.

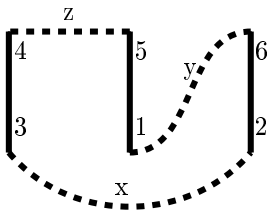

Figure 135: A cycle of length $x+y+z+3$ passes through a gadget that replaced a $2-P$.

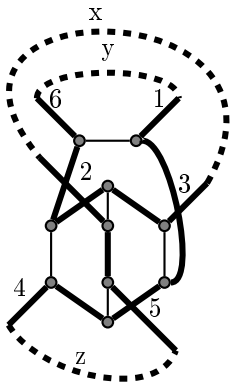

Figure 136: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x+y+z+13$.

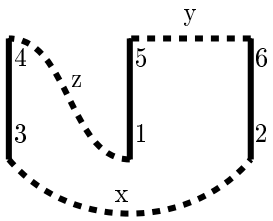

Figure 137: A cycle of length $x+y+z+3$ passes through a gadget that replaced a $2-P$.

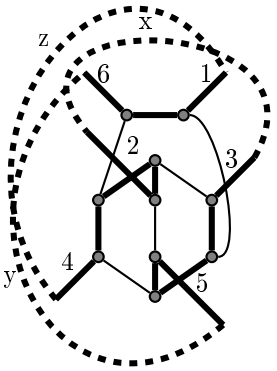

Figure 138: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x+y+z+13$.

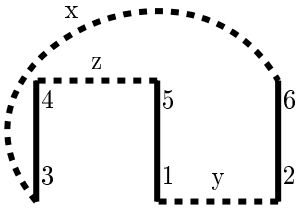

Figure 139: A cycle of length $x+y+z+3$ passes through a gadget that replaced a $2-P$.

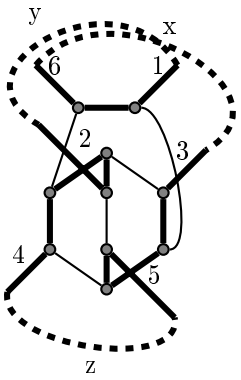

Figure 140: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x+y+z+13$.

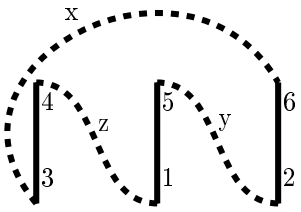

Figure 141: A cycle of length $x+y+z+3$ passes through a gadget that replaced a $2-P$.

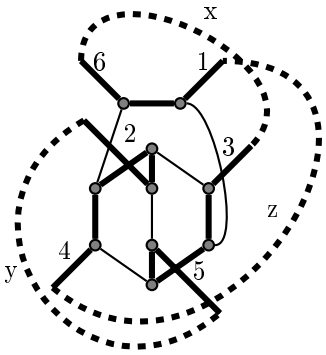

Figure 142: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x+y+z+13$.

10 Appendix E: $W2-P$s

10.1 Gadget is covered by two cycles

If a $W2-P$ gadget is covered by two disjoint cycles in F_i, then the internal edge cannot be part of the 2-factor. This leaves only the single possibility depicted in Figures 143-144, which takes two cycles of lengths $x+2$ and $y+2$ in F_i and returns a single cycle of length $x+y+14$ in F_{i-1} after the expansion. $x+2$ and $y+2$ are both at least 6, so $x+y+14>8$, meaning the resulting cycle in F_{i-1} cannot be a cycle of length 6.

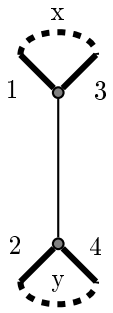

Figure 143: Two cycles of lengths $x+2$ and $y+2$ pass through a gadget that replaced a $W2-P$.

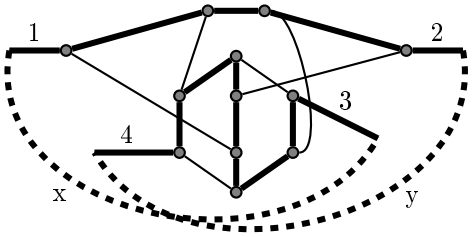

Figure 144: The cycles from the previous figure, after expanding the gadget, are now a single cycle of length $x+y+14$.

10.2 Gadget is covered by one cycle

If a $W2-P$ gadget is covered by a single cycle in F_i, then we consider cases when the internal edge is part of F_i and those when the internal edge is not included. First consider the cases when the internal edge is included in F_i. Then F_i passes through either edge 1 or 3 and either edge 2 or 4. Each of these possibilities takes a cycle of length $x+3$ in F_i and returns a cycle of length $x+13$ in F_{i-1}. $x+3 \geq 6$, so $x+13 > 8$, meaning the resulting cycle in F_{i-1} cannot be a cycle of length 6. The case when exiting edges 1 and 4 are included in F_i is symmetric to the case when exiting edges 2 and 3, so only the first of these cases is included in this appendix. Then, there are three unique cases to consider where the internal edge is included in F_i. We consider these cases in Figures 145-150.

Next, consider when the internal edge is not included in F_i. Then all four exiting edges of the gadget must be used. There are two cases to consider where F_i can pass through these four edges, because after exiting edge 1, the cycle can re-enter the gadget at either edge 2 or 4. Each of these possibilities takes a cycle of length $x+y+4$ in F_i and returns a cycle of length $x+y+14$ in F_{i-1}. $x+y+4 \geq 6$, so $x+y+14 > 8$, meaning the resulting cycle in F_{i-1} cannot be a cycle of length 6. We consider both cases in Figures 151-154.

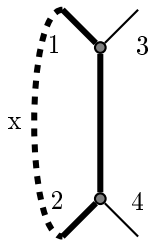

Figure 145: A cycle of length $x+3$ passes through a gadget that replaced a $W2-P$.

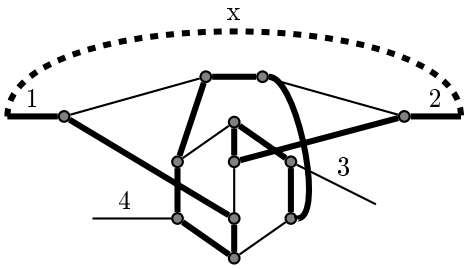

Figure 146: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x+13$.

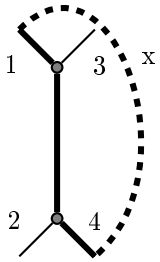

Figure 147: A cycle of length $x+3$ passes through a gadget that replaced a $W2-P$.

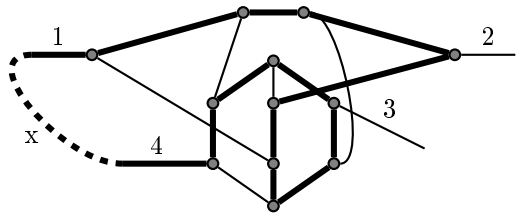

Figure 148: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x+13$.

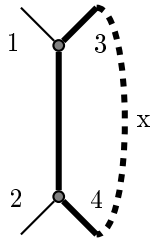

Figure 149: A cycle of length $x+3$ passes through a gadget that replaced a $W2-P$.

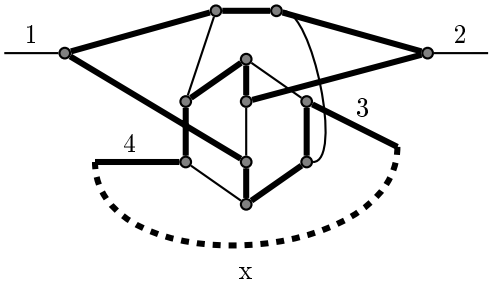

Figure 150: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x+13$.

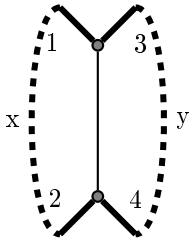

Figure 151: A cycle of length $x+4$ passes through a gadget that replaced a $W2-P$.

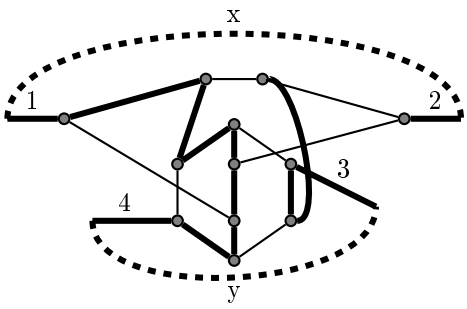

Figure 152: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x+y+14$.

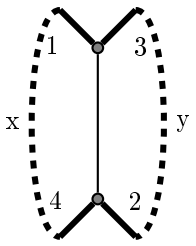

Figure 153: A cycle of length $x+4$ passes through a gadget that replaced a $W2-P$.

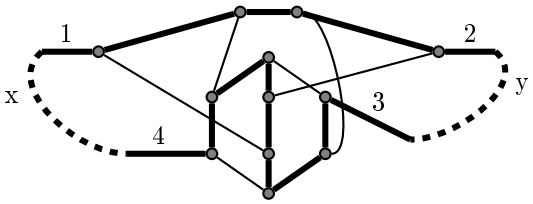

Figure 154: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x+y+14$.

11 Appendix F: $SV2-P$s

Two edges of each $SV2-P$ gadget is covered by a cycle in F_i. Then, there are three cases to consider, when each of the gadget's edges are excluded from F_i. In each of these cases, we start with a cycle of length $x+2$ in F_i and are returned a cycle of length $x+14$. $x+2 \geq 6$, so $x+14 > 8$, meaning these expansion operations cannot introduce an organic 6-cycle into the 2-factor.

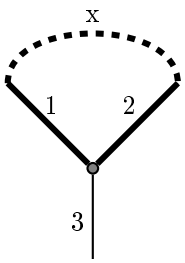

Figure 155: A cycle of length $x+2$ passes through a gadget that replaced a $SV2-P$.

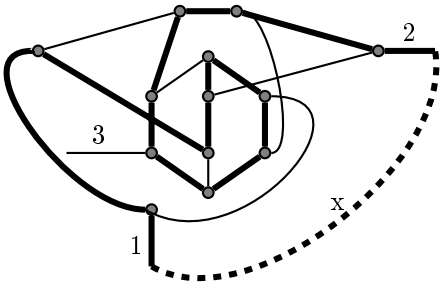

Figure 156: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x+14$.

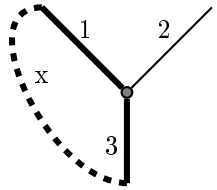

Figure 157: A cycle of length $x+2$ passes through a gadget that replaced a $SV2-P$.

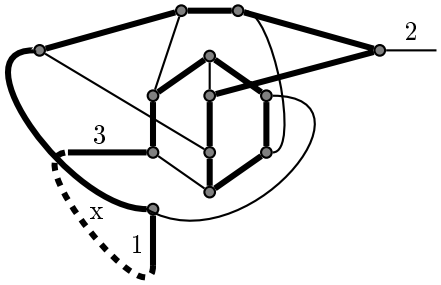

Figure 158: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x+14$.

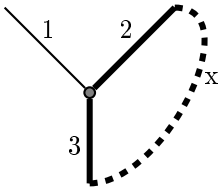

Figure 159: A cycle of length $x+2$ passes through a gadget that replaced a $SV2-P$.

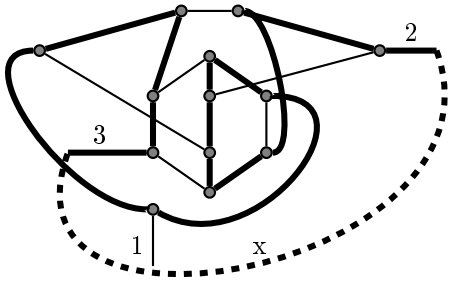

Figure 160: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x+14$.

12 Appendix G: $SE2-P$s

There are two cases to consider for a $SE2-P$ gadget, when the edge is included in F_i and when it is not. In both cases, expanding the gadget cannot introduce an organic 6-cycle to F_{i-1}. We consider both cases in Figures 162-164.

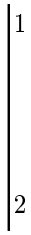

Figure 161: The super-edge that replaced a $SE2-P$, which is not included

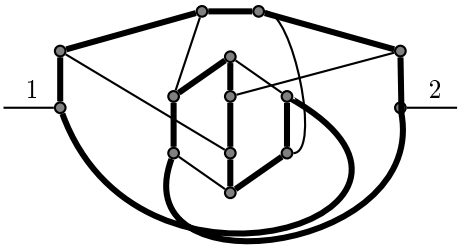

Figure 162: The cycle from the previous figure, after expanding the gadget, is now a cycle of length 14.

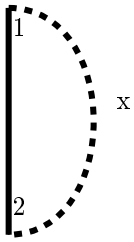

Figure 163: A cycle of length $x+1$ passes through a gadget that replaced a $SE2-P$.

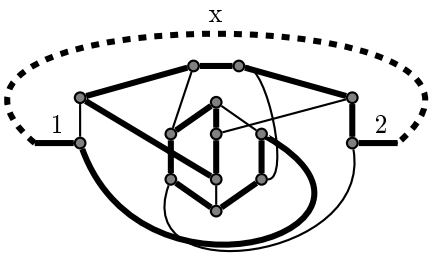

Figure 164: The cycle from the previous figure, after expanding the gadget, is now a cycle of length $x+15$.

References

[1] Nishita Aggarwal, Naveen Garg, and Swati Gupta. A 4/3-approximation for tsp on cubic 3-edge-connected graphs. *arXiv preprint arXiv:1101.5586*, 2011.

[2] Sylvia Boyd, René Sitters, Suzanne van der Ster, and Leen Stougie. Tsp on cubic and subcubic graphs. In *Integer Programming and Combinatoral Optimization*, pages 65–77. Springer, 2011.

[3] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, DTIC Document, 1976.

[4] José R Correa, Omar Larré, and José A Soto. Tsp tours in cubic graphs: beyond 4/3. In *Algorithms–ESA 2012*, pages 790–801. Springer, 2012.

[5] David Gamarnik, Moshe Lewenstein, and Maxim Sviridenko. An improved upper bound for the tsp in cubic 3-edge-connected graphs. *Oper. Res. Lett.*, 33(5):467–474, sep 2005.

[6] Shayan Oveis Gharan, Amin Saberi, and Mohit Singh. A randomized rounding approach to the traveling salesman problem. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 550–559. IEEE, 2011.

[7] Michel X Goemans. Worst-case comparison of valid inequalities for the tsp. *Mathematical Programming*, 69(1-3):335–349, 1995.

[8] John E Hopcroft and Richard M Karp. An n^5/2 algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973.

[9] T Momke and Ola Svensson. Approximating graphic tsp by matchings. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 560–569. IEEE, 2011.

[10] András Sebő and Jens Vygen. Shorter tours by nicer ears: 7/5-approximation for graphic tsp, 3/2 for the path version, and 4/3 for two-edge-connected subgraphs. *arXiv preprint arXiv:1201.1870*, 2012.