Approximation Algorithms for Some Combinatorial Optimization Problems

by

Yao Xu

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science
University of Alberta

Abstract

Many real-world problems can be formulated as *combinatorial optimization* problems, thus making it very important to find efficient methods to solve them, both theoretically and practically. In this thesis, we consider several **NP**-*hard* combinatorial optimization problems, consisting of some classification problems and some computational biology problems, all of which can be formulated in terms of *graphs*; we focus on the design and analysis of *approximation algorithms* for these problems. The main techniques used to design and analyze approximation algorithms in this thesis include: *randomized rounding* (based on *linear programming relaxation*), *local search*, and *amortization*.

We investigate the following **NP**-hard problems in this thesis: the *maximum happy vertice* (MHV) problem, its complement the *minimum unhappy vertices* (MUHV) problem, the *maximum duo-preservation string mapping* (Max-Duo) problem, and the *k-path-partition* (k-PP) problem. The MHV and MUHV problems, which are actually labeling problems, and the k-PP problem can all be considered as classification problems. The Max-Duo problem is a string comparison problem, with applications in bioinformatics and data compression, and the k-PP problem was actually arising from a broadcasting problem in data communication networks. We present approximation algorithms for MHV and MUHV based on randomized linear programming rounding. For Max-Duo and k-PP with $k = 3$, we propose improved approximation algorithms mainly based on local search, and the performances of these approximation algorithms are all done through amortized analysis.

Acknowledgements

I am very grateful to everyone who has contributed in one way or anther to the completion of my thesis. I sincerely appreciate all their help and support.

First and foremost, I would like to thank my supervisor, Dr. Guohui Lin, for his guidance, advice, encouragement, and patience that have contributed tremendously to my four years of Ph.D. life. I feel very fortunate to have had a teacher and collaborator like Dr. Lin. I have learned quite a lot from discussing problems with him, and after every discussion, I could always find a right direction for the next step of my research. His enthusiasm for research and his way of thinking has been an inspiration to me. In addition to my research, Dr. Lin has also spent much time in teaching and providing valuable suggestions to my writing and my talks, for which I am really thankful. Besides, I have also benefited from Dr. Lin's support and advice outside of my research.

I am very thankful to Dr. Yong Chen, Dr. Longchen Liu, Dr. Taibo Luo, and Dr. Wenchang Luo, for long time discussions on problems we have been working on. In addition to them, I want to thank my other collaborators as well, Dr. Randy Goebel, Dr. Tao Jiang, Dr. Angsheng Li, Dr. Guohui Lin, Dr. Tian Liu, Dr. Eiji Miyano, Dr. Bing Su, Dr. Weitian Tong, and Dr. Peng Zhang. Additionally, I would also like to express my appreciation to my supervisory committee members and two other examiners, Dr. Zachary Friggstad, Dr. Randy Goebel, Dr. Linglong Kong, and Dr. Fangxiang Wu, for their time, extreme patience, and valuable suggestions.

I would also like to thank my references Dr. Guohui Lin, Dr. Randy Goebel, and Dr. Jia-Huai You, for their supportive letters of recommendation for my job applications.

Lastly, I want to give my appreciation to my family for their unconditional love and support. For all these years I have been studying and living abroad, my parents and my grandmother have always been very much supportive and encouraging. Most of all, I want to thank my husband Weitian for his love, support, encouragement, patience, and his faith in me. Although all my family are far away from me for most of my Ph.D. life, I can always feel their love and support, which are indispensable to the completion of my Ph.D. journey.

Contents

List of Figures **vii**

Acronyms **xii**

1 Introduction **1**

1.1 Preliminaries 2

1.1.1 Some graph terminologies 2

1.1.2 Notations of some basic complexity classes 4

1.1.3 Approximation algorithms 6

1.2 Techniques for approximation algorithms design and analysis 8

1.2.1 Randomized rounding 8

1.2.2 Local search 9

1.2.3 Amortized analysis 10

1.3 Problems discussed and thesis contributions 11

1.3.1 The vertex happiness problems 11

1.3.2 The maximum duo-preservation string mapping problem 14

1.3.3 The path partition problem 16

2 The Vertex Happiness Problems **18**

2.1 Introduction 18

2.1.1 Related work 21

2.1.2 Our contributions 25

2.1.3 Organization 26

2.2 A $1/\big(\Delta + 1/g(\Delta)\big)$-approximation for MHV 26

2.2.1 Probability that a vertex in $\mathcal{B}_0$ becomes happy 28

2.2.2 Probability that a vertex in $\mathcal{B}_1$ becomes happy 30

2.2.3 The approximation ratio analysis 31

2.3 The Sub-ML and the Sup-ML problems 36

2.4 The approximability of the MUHV problem 38

2.5 Improved approximation results for the MHV problem 43

2.5.1 A $2/k$-approximation for MHV 43

2.5.2 A hardness result for MHV 46

2.6 Concluding remarks and possible future work 48

3 The Maximum Duo-preservation String Mapping Problem **50**

3.1 Introduction 50

3.2 Preliminaries 54

3.3 On approximating the 2-Max-Duo problem 58

3.3.1 Properties for the graph H when $k = 2$ 58
3.3.2 An approximation algorithm for 2-MAX-DUO 65
3.3.3 Review of the reduction from MIS on cubic graphs 68
3.4 On approximating the general MAX-DUO problem 72
3.4.1 A local search algorithm $\mathcal{LS}$ 72
3.4.1.1 Operation REPLACE-5-BY-6 73
3.4.1.2 Operation REDUCE-5-BY-5 74
3.4.1.3 The local search algorithm $\mathcal{LS}$ 75
3.4.1.4 A better time complexity analysis for $\mathcal{LS}$ 75
3.4.2 Approximation ratio analysis for the algorithm $\mathcal{LS}$ 81
3.4.2.1 The amortization scheme 81
3.4.2.2 Value combinations of $\tau(e_{i,j} \leftarrow C^*(e_{i,j}))$ with $\omega(e_{i,j}) \geq 3$ 83
3.4.2.3 Ordered value combinations of $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot}))$ with $\omega(e_{i,j}) \geq 3$ 85
3.4.2.4 Edge combinations of $C(C^*(e_{i,j}))$ with $\omega(e_{i,j}) \geq 3$ 89
$e_{i,j}$ cannot be a parallel edge of M 89
$e_{i,j}$ is a singleton edge of M 90
3.4.2.5 An upper bound on $\omega(e)$ for $e \in C(C^*(e_{i,j})) - \{e_{i,j}\}$ . . . 101
3.4.2.6 An upper bound on $\omega(e)$ for $e \in C(C^*(e_{i,j}))$ known to be parallel 106
3.4.2.7 An upper bound on the average value of $\omega(e)$ 115
3.4.3 Lower bounds on the locality gap for the algorithm $\mathcal{LS}$ 117
3.4.3.1 An instance of MCBM 117
3.4.3.2 An instance of MAX-DUO 121
3.5 Concluding remarks and possible future work 123

4 The Path Partition Problem 126
4.1 Introduction 126
4.2 A local search approximation algorithm 128
4.2.1 Computing a 3-path partition with the least 1-paths 129
4.2.1.1 Step 1: computing a maximum matching 129
4.2.1.2 Step 2: computing a second maximum matching 130
4.2.1.3 Step 3: reducing 1-paths to the minimum 130
4.2.1.4 The main theorem 133
4.2.2 Local operations and their priorities 136
4.2.2.1 OPERATION 3-0-BY-0-2, highest priority 1 136
4.2.2.2 OPERATION 3-1-BY-0-3, priority 2 137
4.2.2.3 OPERATION 4-1-BY-1-3, priority 3 138
4.2.2.4 OPERATION 4-2-BY-1-4, lowest priority 4 138
4.2.3 The complete local search algorithm APPROX 140
4.3 Analysis of the approximation ratio 4/3 141
4.3.1 Distribution process Phase 1 142

4.3.2 Distribution process Phase 2 144
4.3.3 A tight instance of algorithm Approx 150
4.4 Concluding remarks and possible future work 150

5 Conclusions and Future Work **152**

Bibliography **154**

List of Figures

2.1 An instance of MHV on the left and a coloring scheme on the right, where the integer in the parenthesis on each vertex is its weight. In this coloring scheme, the total weight of happy vertices is $w(a) + w(d) + w(h) + w(g) = 9$. 19
2.2 A high-level description of Algorithm $\mathcal{P}$ for the MHV problem. 27
2.3 A high-level description of Algorithm $\mathcal{R}$ for the MHV problem. 27
2.4 The non-uniform Algorithm $\mathcal{A}$ for the MHV problem. 27
2.5 An illustration of $a_0(\lambda)$ and $a_1(\lambda)$ with $\lambda_2 \leq \lambda^*$. The thick line denotes the function $\min\{a_0(\lambda), a_1(\lambda)\}$. 34
2.6 An illustration of $a_0(\lambda)$ and $a_1(\lambda)$ with $\lambda^* < \lambda_2$. The thick line denotes the function $\min\{a_0(\lambda), a_1(\lambda)\}$. 35
2.7 A high-level description of Algorithm $\mathcal{P}$ for the MHV problem. 37

3.1 An instance of Max-Duo with two strings $A = (a, b, c, d, a, b, c)$ and $B = (b, c, d, c, a, b, a)$, and a common partition $\{a, bcd, ab, c\}$ that preserves three duos (b, c), (c, d) and (a, b), corresponding to the perfect matching shown in the figure. 52
3.2 A bipartite graph $G = (D^A, D^B, E)$ constructed from the two strings $A = (a, b, c, d, a, b, c)$ and $B = (b, c, d, c, a, b, a)$, and a compatible matching in G containing three edges $e_{2,1}, e_{3,2}, e_{5,5}$. 55
3.3 An instance of the k-Max-Duo problem with $A = (a, b, c, d, e, f, b, c, d, e)$ and $B = (f, b, c, d, e, a, b, c, d, e)$. Figure 3.3a is the graphical view as a bipartite graph $G_0 = (A, B, E_0)$, where a perfect matching consisting of the ten bold edges form into eight pairs of parallel edges, corresponding to the eight preserved duos $(a, b), (b, c), (c, d), (d, e), (f, b), (b, c), (c, d)$ and (d, e). Figure 3.3b shows the instance graph $H = (V, F)$ of MIS constructed from H, where the independent set $\{v_{1,6}, v_{2,7}, v_{3,8}, v_{4,9}, v_{6,1}, v_{7,2}, v_{8,3}, v_{9,4}\}$ corresponds to the eight pairs of parallel edges shown in Figure 3.3a, and consequently also corresponds to the eight preserved duos. In this instance, we have $k = 2$. Any maximum independent set of H must contain some of the degree-6 vertices, invalidating the $(1.6 + \epsilon)$-approximation for 2-Max-Duo proposed in [10]. 57
3.4 The square $S(i, i'; j, j')$ shown in bold lines. The two non-adjacent vertices $v_{i,j}$ and $v_{i',j'}$ of the square form a pair stated in Corollary 3.5; they have 6 common neighbors, of which two inside the square and four outside of the square. 61

3.5 An instance of the 2-Max-Duo problem with $A = (a, b, c, d, e, f, g, b, c, d, e, h, y, x)$ and $B = (g, b, c, d, e, h, a, b, c, d, x, y, e, f)$. The bipartite graph $G_0 = (A, B, E_0)$ is shown in Figure 3.5a and the instance graph $H = (V, F)$ of the MIS problem is shown in Figure 3.5b. There is a maximal series of 2 squares $\mathcal{S}^2(2, 8; 2, 8)$ in the graph G, with the four substrings "*bcd*". The bipartite graph $G'_0 = (A', B', E'_0)$ is shown in Figure 3.5c and the graph $H' = (V', F')$ is shown in Figure 3.5d, on $A' = (a, d, e, f, g, d, e, h, y, x)$ and $B' = (g, d, e, h, a, d, x, y, e, f)$. Applying the vertex contracting process on H also gives the graph H'. 62
3.6 A high-level description of the approximation algorithm for 2-Max-Duo. . 65
3.7 Illustration of the execution of our algorithm Approx on the instance shown in Figure 3.5. The independent set I_1 in the graph H_1 is shown in Figure 3.7a in filled circles, for which we did not apply the state-of-the-art approximation algorithm for the MIS problem. The independent set I in the graph H is shown in Figure 3.7b in filled circles, according to Corollary 3.9 the four vertices $v_{2,8}, v_{3,9}, v_{8,2}, v_{9,3}$ are added due to $v_{10,4} \in I_1$. The parallel edges of G_0 corresponding to the vertices of I are shown in Figure 3.7c, representing a feasible solution to the 2-Max-Duo instance shown in Figure 3.5. 67
3.8 The instance $I_u = (A_u, B_u)$ defined for each vertex $u \in V'$. Each two dots between a pair of substrings represent a substring of two distinct letters (x^i_u, y^i_u) in A_u and (y^i_u, x^i_u) in B_u, respectively, for $i = 1, \ldots, 6$. All these 12 letters x^i_u, y^i_u are distinct and also distinct from all the other letters in A_u and B_u. Each solid or dashed line connects a pair of common duos in A_u and B_u. The set of five duos connecting by solid lines is a unique optimal solution to I_u. 69
3.9 The gadget subgraph associated with the instance $I_u = (A_u, B_u)$, in which there are nine vertices corresponding to the nine common duos between A_u and B_u. 69
3.10 Four options of modifying instances I_u and I_v by only modifying the right side of A_u, B_u and the left side of A_v, B_v such that an optimal solution to $I_{H'}$ coincides with at most one of the optimal solutions to I_u and I_v. 70
3.11 Four different configurations for joining the two gadget subgraphs for the vertices $u, v \in V'$, in each of which a common duo is revised for the directed edge $(u, v) \in E'$. 71
3.12 A high-level description of the algorithm of finding an X'_1 in E'_1. 79
3.13 A high-level description of the algorithm of finding a compatible matching of G^* incident on U^A. 80

3.15 The only possible configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{2}, 1, \frac{1}{2}\right)$. We have $|C(C^*(e_{i,\cdot}))| = 3$, and $|\mathcal{N}_p(e_{i+2})| = |\mathcal{N}_p(e_{j'-2})| = 1$ in this configuration. It also represents the other three symmetric configurations where $|\mathcal{N}_p(e_{i+2})| = |\mathcal{N}_p(e_{i-2})| = 1$, $|\mathcal{N}_p(e_{j'+2})| = |\mathcal{N}_p(e_{j'-2})| = 1$, and $|\mathcal{N}_p(e_{j'+2})| = |\mathcal{N}_p(e_{i-2})| = 1$, respectively. (Recall that the edge $e_{i,j}$ is in bold solid line, the edges in vertical bold dashed lines are in $C^*(e_{i,\cdot})$, the edges in thin solid lines are in $C(C^*(e_{i,\cdot}))$, and the edges in thin dashed lines are in $\mathcal{N}_p(C(C^*(e_{i,\cdot}))$; the vertices in filled circles are surely incident with no edges of M and the vertices in hollow circles have uncertain incidence in M.) 92
3.16 The only possible configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{2}, 1, \frac{1}{3}\right)$. We have $|C(C^*(e_{i,\cdot}))| = 4$ and $|\mathcal{N}_p(e_{j'-2})| = 1$ in this configuration. It also represents the symmetric configuration where $|\mathcal{N}_p(e_{i-2})| = 1$. 92
3.17 The only possible configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(1, \frac{1}{2}, \frac{1}{3}\right)$. We have $|C(C^*(e_{i,\cdot}))| = 3$ and $|\mathcal{N}_p(e_{j'+2})| = 1$ in this configuration. 93
3.18 The only possible configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(1, \frac{1}{2}, \frac{1}{4}\right)$. We have $|C(C^*(e_{i,\cdot}))| = 4$ and $\mathcal{N}_p(e_{j'+2}) = \emptyset$ in this configuration. . 93
3.19 The only possible configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{3}, 1, \frac{1}{3}\right)$, where $|C(C^*(e_{i,\cdot}))| = 5$. 93
3.20 The two possible configurations of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{3}\right)$. We have $|C(C^*(e_{i,\cdot}))| = 4$. They also represent the symmetric case where $e_{i'-2} \in M$ instead of $e_{j'-2} \in M$. 94
3.21 The two possible configurations of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{4}\right)$. They are associated with the only possible edge combination of $C(C^*(e_{i,\cdot}))$ with $|C(C^*(e_{i,\cdot}))| = 5$, which also represents the symmetric case where $e_{i'-2} \in M$ instead of $e_{j'-2} \in M$. The first configuration shadows the second one. 95
3.22 The three possible configurations of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{3}, \frac{1}{2}, \frac{1}{3}\right)$, associated with three possible edge combinations of $C(C^*(e_{i,\cdot}))$ with $|C(C^*(e_{i,\cdot}))| = 4, 5, 5$, respectively. The configuration in Figure 3.22a also represents the symmetric case where $e_{i-2} \in M$ instead of $e_{j'-2} \in M$ and/or $e_{j'+2} \in M$ instead of $e_{i+2} \in M$. 96
3.23 The only possible configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{2}, \frac{1}{3}, \frac{1}{3}\right)$. We have $|C(C^*(e_{i,\cdot}))| = 4$ and $|\mathcal{N}_p(e_{j'+2})| = 1$. 96
3.24 The two possible configurations of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{3}, \frac{1}{2}, \frac{1}{4}\right)$, associated with two possible edge combinations of $C(C^*(e_{i,\cdot}))$ with $|C(C^*(e_{i,\cdot}))| = 5, 6$, respectively. The configuration in Figure 3.24a also represents the symmetric case where $e_{i-2} \in M$ instead of $e_{j'-2} \in M$. 97
3.25 The three possible configurations of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{2}, \frac{1}{3}, \frac{1}{4}\right)$, associated with three possible edge combinations of $C(C^*(e_{i,\cdot}))$ with $|C(C^*(e_{i,\cdot}))| = 4, 4, 5$, respectively. 98
3.26 The only possible configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{2}, \frac{1}{3}, \frac{1}{5}\right)$, where $|C(C^*(e_{i,\cdot}))| = 5$. 98
3.27 The only possible configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\right)$, where $|C(C^*(e_{i,\cdot}))| = 6$. 98

3.28 The four possible configurations of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$, associated with four possible edge combinations of $C(C^*(e_{i,\cdot}))$ with $|C(C^*(e_{i,\cdot}))| = 4, 5, 5, 5$, respectively. Figure 3.28a also represents the case where $e_{j'+2} \in M$ instead of $e_{i+2} \in M$; Figure 3.28c also represents the case where $e_{j'-2}, e_{i+2} \in M$ instead of $e_{i-2}, e_{j'+2} \in M$. 99

3.29 The two possible configurations of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{2}, \frac{1}{2}, 0\right)$, associated with the only possible edge combinations of $C(C^*(e_{i,\cdot}))$ with $|C(C^*(e_{i,\cdot}))| = 3$. Each of them also represents the case where $e_{i-2} \in M$ instead of $e_{j'-2} \in M$. 100

3.30 The two possible configurations of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{2}, 0, \frac{1}{2}\right)$, associated with the two possible edge combinations of $C(C^*(e_{i,\cdot}))$ with $|C(C^*(e_{i,\cdot}))| = 3$. The second configuration represents the other 15 symmetric cases exactly one of $i-2, j''-2, j''-1, j''$ is incident with an edge of M and exactly one of $i+2, j''', j'''+1, j'''+2$ is incident with an edge of M, but the two edges are not parallel to each other. 100

3.31 A bipartite graph $G = (V^A, V^B, E)$, where $V^A = \{1, 2, \ldots, 26\}$, $V^B = \{1', 2', \ldots, 26'\}$, and $E = M \cup M^*$. M consists of the 12 edges in solid lines and it is a maximal compatible matching in G; M^* consists of the 26 edges in dashed lines and it is an optimal compatible matching to the MCBM problem on G. 118

3.32 The corresponding bipartite graph $G = (V^A, V^B, E)$ constructed from two identical strings $A = B = (a, b, c, d, e, f, b, c, d, e, g)$, where $V^A = \{1, 2, \ldots, 10\}$, $V^B = \{1', 2', \ldots, 10'\}$, and $E = M \cup M^*$. M consists of the six edges in solid lines and it is a compatible matching in G; M^* consists of the ten edges in dashed lines and it is an optimal compatible matching to the MCBM problem on G. 121

3.14 A high-level description of the algorithm of expanding the current maximal compatible matching by swapping ρ edges out for $\rho + 1$ compatible edges. . 125

4.1 An instance of 3-PP. The two 2-paths and the two 3-paths represented by edges with red backgroud is a solution to 3-PP for this instance. 127

4.2 An alternating path v_0-w_1-u_1-w_2-u_2-. . .-w_i-u_i-u_{i+1} that saves the singleton v_0, where the first i paths are 3-paths and the last one is a 2-path. In the figure, solid edges are in the edge set E^* and dashed edges are outside of E^*. 132

4.3 A high-level description of Algorithm A for computing a 3-path partition in the graph $G = (V, E)$ with the least singletons. 133

4.4 The configuration of the expected collection for Operation 3-0-By-0-2, which has the highest priority 1, where solid black edges are in Q, dashed edges are in E but outside of Q, and the edges with red backgroud form the replacement collection. 136

4.5 The first class of configuration of the expected collection for Operation 3-1-By-0-3, which has priority 2.1, where solid black edges are in Q, dashed edges are in E but outside of Q, and the edges highlighted in red form the replacement collection. 137

4.6 The second class of configurations of the expected collection in Operation 3-1-By-0-3, which has priority 2.2, where solid black edges are in Q, dashed edges are in E but outside of Q, and the edges highlighted in red form the replacement collection. 138
4.7 The configuration of the expected collection for Operation 4-1-By-1-3, which has priority 3, where solid black edges are in Q, dashed edges are in E but outside of Q, and the edges highlighted in red form the replacement collection. 139
4.8 The three classes of configurations of the expected collections for an Operation 4-2-By-1-4, where solid black edges are in Q, dashed edges are in E but outside of Q, and the edges highlighted in red form a possible replacement collection. In every class, each dotted edge between $P_5 = u$-w-v and $P_6 = u'$-w'-v' corresponds to one configuration. 139
4.9 A high-level description of the local search algorithm Approx, where each "break" is meant to go to the next iteration of the loop. 140
4.10 Illustrations of the token distribution scheme in Phase 1, where solid edges are in $E(Q_2)$ and dashed edges are in $E(Q^*)$. In Fig. 4.10c, u or v can be either an endpoint or the midpoint of the corresponding 3-path of Q^*_3. . . . 143
4.11 An illustration of a 3-path $P_1 = v$-v'-$v'' \in Q^*_3$ with $\tau_1(P_1) = 3/2$, where u-v, u'-$v' \in E(Q_2)$, $P_3 \in Q_3$, with w being either the midpoint or an endpoint of P_3, and $P_2 \in E(Q^*_3)$ is represented by dashed edges, on which w is adjacent to u. 145
4.12 The cases when P_4 is a singleton of Q^*_1, where solid edges are in $E(Q_2)$ or $E(Q_3)$ and dashed edges are in $E(Q^*)$. x_1 is the tail of the pipe through which P_4 could receive $1/2$ token from the 2-path of u-v. 146
4.13 The cases when both P_4 and P_5 are in $Q^*_2 \cup Q^*_3$, where solid edges are in $E(Q_2)$ or $E(Q_3)$ and dashed edges are in $E(Q^*)$. In Fig. 4.13a, y_1 is the tail of the pipe through which P_5 receives $1/2$ token from the 2-path u-v; in Fig. 4.13b, x_1 is the tail of the pipe through which P_4 receives $1/4$ token from the 2-path u-v and y_1 is the tail of the pipe through which P_5 receives $1/4$ token from the 2-path u-v. 147
4.14 An illustration of y_2 in Fig. 4.13a being the tail of a pipe, say $z_1 \longrightarrow w' \longrightarrow y_2$, which could never happen due to Operation 4-2-By-1-4. 148
4.15 An illustration of y_1 being an endpoint of P_5 in Fig. 4.13a, where $P_5 = y_1$-y_2-y_3, solid edges are in $E(Q_2)$ or $E(Q_3)$ and dashed edges are in $E(Q^*)$. y_3 could be on a path of Q_1, Q_2, or Q_3. 149
4.16 A tight instance of 27 vertices, where solid edges represent a 3-path partition Q produced by Approx and dashed edges represent an optimal 3-path partition Q^*. The edges (u_i, v_i), $i = 1, 3, 5, 7, 9$, are in $E(Q_2) \cap E(Q^*)$, shown in both solid and dashed edges. In our distribution process, each of the nine 3-paths in Q^* receives 1 token from the 2-paths in Q. 150

Acronyms

CMIS constrained maximum induced subgraph 51, 52

Coloring graph coloring 12, 19, 46

Cubic-MIS maximum independent set on cubic graphs 68

FPT fixed parameter tractable 51

FPTAS fully polynomial-time approximation scheme 7

Hyp-MC hypergraph multiway cut 13, 22–25, 41–43, 49

ILP integer linear program or integer linear programming 8, 9

LP linear program or linear programming 8, 9, 21, 24–26, 44–46, 48

MC multiway cut 22–24

MCBM maximum compatible bipartite matching x, 53–55, 75, 117, 118, 120, 121, 123, 124

MCSP minimum common string partition 14, 15, 50, 51, 53, 68, 123

MHE maximum happy edges 12, 18, 21, 24

MHV maximum happy vertices ii, vii, 12, 13, 18–22, 24–27, 29, 31, 35–37, 43–49, 152

MIS maximum independent set vii, viii, 13, 15, 46–49, 52–54, 56–59, 62, 64, 66–69, 123, 124

MPSM maximum duo-preservation string mapping 14, 51

MUC multiway uncut 24

MUHV minimum unhappy vertices ii, 12, 13, 19, 20, 22, 25, 26, 38–43, 48, 49, 152

Max-Duo maximum duo-preservation string mapping ii, vii, viii, 14, 15, 51–60, 62, 65, 67, 68, 81, 117, 121–124, 152, 153

NPO **NP**-optimization 6, 7

NODE-MC node-weighted multiway cut 22–24

PP path partition ii, x, 16, 17, 126–128, 135, 141, 142, 150–153

PTAS polynomial-time approximation scheme 7

SC set cover 16, 17, 127, 128, 151

SUB-ML submodular multi-labeling 13, 20, 22, 25, 26, 36, 37, 39, 43, 48, 49

SUB-MP submodular multiway partition 22, 24–26, 37, 39, 48

SUP-ML supermodular multi-labeling 13, 20, 22, 24–26, 37, 38, 44, 46, 48, 49

SUP-MP supermodular multiway partition 24

Chapter 1

Introduction

In real life applications, many computational problems can be formulated as *combinatorial optimization problems*. According to [3], an *optimization problem* can be formally defined as follows.

Definition 1.1. An *optimization problem* $\mathcal{P}$ can be characterized by a quadruple ($\mathcal{I}_{\mathcal{P}}$, $\mathcal{F}_{\mathcal{P}}$, $\mathcal{V}_{\mathcal{P}}$, $\mathcal{G}_{\mathcal{P}}$), where

- $\mathcal{I}_{\mathcal{P}}$ is the set of instances of the problem;
- $\mathcal{F}_{\mathcal{P}}$ is a function which associates to any input instance $I \in \mathcal{I}_{\mathcal{P}}$ the set of *feasible solutions* of I, *i.e.*, the set of solutions satisfying all the given constraints;
- $\mathcal{V}_{\mathcal{P}}$ is a function defined for pairs (I, s) such that $I \in \mathcal{I}_{\mathcal{P}}$ and $s \in \mathcal{F}_{\mathcal{P}}(I)$, and $\mathcal{V}_{\mathcal{P}}(I, s)$ provides a positive real number for each pair (I, s) which is the value of the feasible solution s;
- $\mathcal{G}_{\mathcal{P}} \in \{\min, \max\}$ specifies whether the problem is a minimization or a maximization problem.

Thus, the goal of an optimization problem is to find an *optimal solution* out of the set of feasible solutions. Formally, given an instance $I \in \mathcal{I}_{\mathcal{P}}$, an *optimal solution* of I is a feasible solution $s^* \in \mathcal{F}_{\mathcal{P}}(I)$ such that for all $s \in \mathcal{F}_{\mathcal{P}}(I)$, $\mathcal{V}_{\mathcal{P}}(I, s^*) \leq \mathcal{V}_{\mathcal{P}}(I, s)$ if $\mathcal{G}_{\mathcal{P}} = \min$, or $\mathcal{V}_{\mathcal{P}}(I, s^*) \geq \mathcal{V}_{\mathcal{P}}(I, s)$ if $\mathcal{G}_{\mathcal{P}} = \max$.

An optimization problem is called *combinatorial* if the variables are discrete and the set of feasible solutions is finite, or possibly countably infinite [66].

In order to solve a combinatorial optimization problem, the most straightforward approach would be to enumerate all possible feasible solutions and then determine which of them is an optimal one. This approach can be referred to as *brute-force search*. However, for many combinatorial optimization problems, the search space grows exponentially with the size of the problem, thus making the brute-force search impractical. Actually, there are numerous combinatorial optimization problems which are *computationally intractable*, that is, no *efficient*, *i.e.*, polynomial-time, algorithms can solve them in practice, for example, those known as **NP**-*hard* problems. Under the overwhelming consensus **P**≠**NP**, **NP**-hard problems do not admit efficient exact algorithms. Thus, it becomes very important to design a polynomial-time *approximation algorithm* for an **NP**-hard problem, which can find a solution provably close to the actual optimal solution.

In this thesis, all the problems we discuss are **NP**-hard combinatorial optimization problems that can be modeled by *graphs*. In this chapter, we will first introduce some basic graph terminologies in Section 1.1.1 and some definitions of complexity classes related to this thesis in Sections 1.1.2 and 1.1.3; then we introduce in Section 1.2 the main techniques that will be used in the design and analysis of approximation algorithms in this thesis; lastly in Section 1.3, we give a brief introduction on all the problems that will be discussed in the following chapters, along with our main contributions.

1.1 Preliminaries

1.1.1 Some graph terminologies

In this section, we introduce some basic graph terminologies, with most of the definitions adopted from [27, 58].

A *graph* $G = (V, E)$ consists of a finite set V of *vertices* and a finite set E of *edges*, with each edge in E corresponding to a pair of two vertices in V. We use the notation (u, v) for an edge associated with the vertices $u, v \in V$. Then, if $u \neq v$, in an *undirected graph*, each edge of E is an unordered pair of vertices of V, that is, (u, v) and (v, u) present the same edge in an undirected graph; in a *directed graph* (also called *digraph*), each edge is an ordered pair of vertices of V, that is, (u, v) and (v, u) are two distinct directed edges in a directed graph. An *orientation* of an undirected graph G is a digraph obtained by *orienting*, *i.e.*, choosing a direction for, each edge of G.

If $e = (u, v)$ is an edge in a graph $G = (V, E)$, we say u and v are *adjacent*, u and v are *neighbors* to each other, u and v are the two *endpoints* of e, u and v are both *incident with* e, and e is *incident on* u and v. For a vertex v, the *neighborhood* of v is the set of all the vertices adjacent to v, which is denoted as $N(v)$; the *closed neighborhood* of v is the union of $N(v)$ and $\{v\}$ itself, which is denoted as $N[v]$. The *degree* of a vertex v is the total number of vertices in $N(v)$, and we say v is a degree-$N(v)$ vertex. In a directed graph, we say $e = (u, v)$ *leaves* u and *enters* v, the *out-degree* of v is the number of edges leaving it, and the *in-degree* of v is the number of edges entering it.

An edge is called a *loop* if its two endpoints are the same. Two or more edges are called *multi-edges* if they are incident on the same pair of vertices. A graph is called *simple* if there are no loops or multi-edges in it. In this thesis, every graph is simple and undirected, unless stated otherwise.

We say a graph $G' = (V', E')$ is a *subgraph* of $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. Given a set $V' \subseteq V$, $G' = (V', E')$ is an *induced* subgraph of G if $E' = \{(u, v) \in E \mid u, v \in V'\}$, and we say G' is the subgraph of G *induced by* V'.

In a graph $G = (V, E)$, a *path* is an alternating sequence of vertices and edges v_1, e_1, v_2, $\ldots$, v_k, e_k, v_{k+1} such that $k \geq 0$ and $e_i = (v_i, v_{i+1}) \in E$ for $i = 1, 2, \ldots, k$. It is also called a v_1-v_{k+1}-*path* (or a path *from* v_1 *to* v_{k+1}). A path is *simple* if all vertices in the path are distinct. A *cycle* is a v_1-v_{k+1}-path, with $v_{k+1} = v_1$, and the cycle is *simple* if all vertices in the cycle are distinct.

A graph $G = (V, E)$ is called *connected* if for any two vertices $u, v \in V$, there is a u-v-path in G; otherwise G is *disconnected*. The maximal connected subgraphs of a graph are called its *connected components*.

An *independent set* in G is a subset of pairwise non-adjacent vertices $I \subseteq V$; a *matching* in G is a set of pairwise non-adjacent edges $M \subseteq E$, *i.e.*, the endpoints of all the edges in M are different. Consider a matching M in a graph G. We say a vertex v is *covered* by M if v is an endpoint of some edge $e \in M$; otherwise v is *exposed* by M. If all the vertices are covered by M, then M is called a *perfect matching* of G.

Here are the definitions of some special graphs which will be helpful in presenting this thesis. A *complete graph* is a graph in which every two vertices are adjacent. A *bipartite graph* is a graph $G = (V_1, V_2, E)$, where V_1 and V_2 are two disjoint sets of vertices and both of them are independent sets in G, *i.e.*, $E \subseteq \{(u, v) \mid u \in V_1, v \in V_2\}$. When $E = \{(u, v) \mid u \in V_1, v \in V_2\}$, with $|V_1| = n_1$, $|V_2| = n_2$, then G is called a *complete bipartite graph* and is denoted by K_{n_1,n_2}. A *forest* is a graph without a cycle as a subgraph, and a *tree* is a connected forest. A degree-1 vertex in a graph is called a *leaf*.

1.1.2 Notations of some basic complexity classes

In this section, we introduce some basic complexity classes, with most of the definitions adopted from [2, 3].

The study of *computational complexity* focuses on issues of *computational efficiency*, which is to quantify the amount of computational resources required to solve a given task. With appropriate encoding, discrete objects can usually be represented by strings of bits. A basic computational task is *computing a function*, whose inputs and outputs are both restricted to *finite strings of bits*, *i.e.*, $f : \{0,1\}^* \to \{0,1\}^*$.

A problem $\mathcal{P}$ is called a *decision problem* if the set $\mathcal{I}_{\mathcal{P}}$ of all the instances of $\mathcal{P}$ can be partitioned into a set $\mathcal{Y}_{\mathcal{P}}$ of YES instances and a set $\mathcal{N}_{\mathcal{P}}$ of NO instances, and for any instance $I \in \mathcal{I}_{\mathcal{P}}$, the problem asks to verify whether $I \in \mathcal{Y}_{\mathcal{P}}$. Formally, given a function f whose output is a single bit, *i.e.*, in $\{0,1\}$, the subset $L_f := \{x : f(x) = 1\}$ of $\{0,1\}^*$ is called a *language* or *decision problem*. For any decision problem $\mathcal{P}$, its *complementary problem* $\mathcal{P}^C$ is the decision problem with $\mathcal{I}_{\mathcal{P}^C} = \mathcal{I}_{\mathcal{P}}$, $\mathcal{Y}_{\mathcal{P}^C} = \mathcal{N}_{\mathcal{P}}$, and $\mathcal{N}_{\mathcal{P}^C} = \mathcal{Y}_{\mathcal{P}}$.

The computational efficiency of an algorithm is typically measured as the number of basic operations it performs as a function of its input length. Here are the five basic asymptotic notations which are to describe the running time of an algorithm.

- $f(n) \in O(g(n))$ if $\exists c > 0, \exists n_0 \in \mathbb{N}$, s.t. $\forall n \geq n_0$, $f(n) \leq c \cdot g(n)$;
- $f(n) \in \Omega(g(n))$ if $\exists c > 0, \exists n_0 \in \mathbb{N}$, s.t. $\forall n \geq n_0$, $f(n) \geq c \cdot g(n)$;
- $f(n) \in \Theta(g(n))$ if $\exists c_0, c_1 > 0, \exists n_0 \in \mathbb{N}$, s.t. $\forall n \geq n_0$, $c_0 \cdot g(n) \leq f(n) \leq c_1 \cdot g(n)$;
- $f(n) \in o(g(n))$ if $\forall c > 0, \exists n_0 \in \mathbb{N}$, s.t. $\forall n \geq n_0$, $f(n) < c \cdot g(n)$;
- $f(n) \in \omega(g(n))$ if $\forall c > 0, \exists n_0 \in \mathbb{N}$, s.t. $\forall n \geq n_0$, $f(n) > c \cdot g(n)$.

There is a simple mathematical model that suffices for studying computational efficiency, called the *Turing machine (TM)*, which can be defined as follows, according to [2].

Definition 1.2. A Turing machine (TM) M is described by a tuple (Γ, Q, δ) containing:

- A finite set Γ of the symbols that M's tapes can contain, which is called the alphabet of M. Assume that Γ contains a designated "blank" symbol, a designated "start" symbol, and the numbers 0 and 1.
- A finite set Q of possible states M's register can be in. Assume that Q contains a designated start state and a designated halting state.

- A function $\delta : Q \times \Gamma^k \to Q \times \Gamma^{k-1} \times \{\text{Left, Stay, Right}\}$, $k \geq 2$, describing the rules M uses in performing each step, which is called the *transition function* of M.

Let $f : \{0,1\}^* \to \{0,1\}^*$ be some function and M be a TM. *M computes f* if and only if for every input $x \in \{0,1\}^*$, M halts with $f(x)$ written on its output tape. *M computes f in $T(n)$-time* if its computation on every input of length n requires at most $T(n)$ steps.

A *complexity class* is a set of functions that can be computed within given resource bounds. We introduce as follows some basic complexity classes related to this thesis (the definitions are mainly based on the book [2]).

The class **P**, where **P** stands for "polynomial", is the set of decision problems that can be solved by a TM in polynomial time. The TM defined above is more precisely called the *deterministic* TM since for any input x, the machine's computation can proceed in exactly one way. The class **P** can also be defined more formally based on the definition of the class **DTIME**, where **D** refers to "deterministic", as follows.

Definition 1.3. Let $T : \mathbb{N} \to \mathbb{N}$ be some function. A language $L \subseteq \{0,1\}^*$ with input length n is in $\mathbf{DTIME}(T(n))$ if and only if there is a TM that runs in time $c \cdot T(n)$ for some constant $c > 0$ and computes the function $f_L : \{0,1\}^* \to \{0,1\}$, where $f_L(x) = 1 \Leftrightarrow x \in L$.

Definition 1.4. $\mathbf{P} = \cup_{c \geq 1} \mathbf{DTIME}(n^c)$.

The class **NP**, which stands for "nondeterministic polynomial", is the set of decision problems that can be verified by a deterministic TM in polynomial time. It can also be defined more formally using a variant of TM called *nondeterministic Turing machine (NDTM)*. The only difference between an NDTM and a standard TM is that an NDTM has two transition functions δ_0 and δ_1, and a special state denoted as q_{accept}. In each step, an NDTM M makes an arbitrary choice as to which of its two transition functions to apply. For every input x, $M(x) = 1$ if there exists some sequence of these choices that would make M reach q_{accept} on input x; otherwise, if every sequence of choices makes M halt without reaching q_{accept}, then $M(x) = 0$. *M runs in $T(n)$ time* if for every input $x \in \{0,1\}^*$ of length n and every sequence of nondeterministic choices, M reaches either the halting state or q_{accept} within $T(n)$ steps. Then, the class **NTIME**, where **N** refers to "nondeterministic", can be defined as follows.

Definition 1.5. For every function $T : \mathbb{N} \to \mathbb{N}$ and $L \subseteq \{0,1\}^*$, we say that $L \in \mathbf{NTIME}(T(n))$ if there is a constant $c > 0$ and a $c \cdot T(n)$-time NDTM M such that for every $x \in \{0,1\}^*$, $x \in L \Leftrightarrow M(x) = 1$.

Definition 1.6. **NP** $= \cup_{c \in \mathbb{N}}$**NTIME**$(n^c)$.

A language $L \subseteq \{0,1\}^*$ is *polynomial-time reducible to* a language $L' \subseteq \{0,1\}^*$, denoted by $L \leq_p L'$, if there is a polynomial-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ such that for every $x \in \{0,1\}^*$, $x \in L$ if and only if $f(x) \in L'$.

Definition 1.7. A language $L' \subseteq \{0,1\}^*$ is **NP**-*hard* if $L \leq_p L'$ for every $L \in$ **NP**. L' is **NP**-*complete* if L' is **NP**-hard and $L' \in$ **NP**.

1.1.3 Approximation algorithms

The problems we discuss in this thesis are all **NP**-*optimization* (**NPO**) problems. An optimization problem $\mathcal{P} = (\mathcal{I}, \mathcal{F}, \mathcal{V}, \mathcal{G})$ is called an **NPO** problem [3] if the following holds:

- the set of instances $\mathcal{I}$ is recognizable in polynomial time;
- there exists a polynomial ϕ such that, given an instance $I \in \mathcal{I}$, for any $s \in \mathcal{F}(I)$, $|s| \leq \phi(|I|)$, and additionally, for any I and s, it is decidable in polynomial time whether $s \in \mathcal{F}(I)$;
- the function $\mathcal{V}$ is computable in polynomial time.

These implies the corresponding decision problems are in **NP** [3]. Thus, the class **NPO** can also be defined as the class of optimization problems whose decision versions are in **NP**.

In order to approach **NP**-hard problems, one way is to design a polynomial time algorithm which can find a solution provably close to the actual optimal solution. Such a polynomial time algorithm is called an *approximation algorithm*, which can be formally defined as follows.

Definition 1.8 ([75])**.** An α-*approximation algorithm* for an optimization problem is a polynomial-time algorithm that for all instances of the problem produces a solution whose value is within a factor of α of the value of an optimal solution.

For an α-approximation algorithm, we call α the *performance ratio* or *approximation ratio* of the algorithm.

Definition 1.9 ([75])**.** A *polynomial-time approximation scheme* (**PTAS**) is a family of algorithms $\{A_\epsilon\}$, where there is an algorithm for each $\epsilon > 0$, such that A_ϵ is a $(1 + \epsilon)$-approximation algorithm (for minimization problems) or a $(1 - \epsilon)$-approximation algorithm (for maximization problems).

The running time of the algorithm A_ϵ is allowed to depend arbitrarily on $1/\epsilon$, which could be exponential in $1/\epsilon$ or worse.

Definition 1.10 ([75])**.** A *fully polynomial-time approximation scheme* (**FPTAS**) is an approximation scheme such that its running time is bounded by a polynomial in n and $1/\epsilon$, where n is the input size.

Definition 1.11 ([3])**.** The class **APX** is the class of **NPO** problems that allow polynomial-time approximation algorithms with a approximation ratio bounded by a constant.

A **PTAS** *reduction* from A to B is an approximation-preserving reduction from A to B such that there exists three polynomial-time computable functions f, g, δ, for any $\epsilon > 0$, if the solution y to $f(x)$ (an instance of problem B) is a $(1 + \delta(\epsilon))$-approximation for B, then the corresponding solution $g(x, y, \epsilon)$ to x (an instance of problem A) is a $(1 + \epsilon)$-approximation for A.

Definition 1.12. A problem is **APX**-*hard* if there exists a **PTAS** reduction from any problem in **APX** to the given problem. A problem is **APX**-*complete* if it is **APX**-hard and it is in **APX**.

According to Definition 1.8, we have the approximation ratio $\alpha > 1$ for minimization problems and $\alpha < 1$ for maximization problems. However, in the literature, an approximation ratio of an algorithm for a minimization (maximization, respectively) problem may also be expressed by $1/\alpha$, which is a value less than (greater than, respectively) 1. In this thesis, to express the approximation ratio of an algorithm for each problem, we will follow the convention in most of the literature on it, which can either be less than or greater than 1.

1.2 Techniques for approximation algorithms design and analysis

There are different techniques on the design and analysis of approximation algorithms. We focus on the *randomized rounding* and *local search* approaches for designing, and the *amortization* scheme for analyzing.

1.2.1 Randomized rounding

A *linear program* (LP) consists of a vector **x** of n real variables, a linear *objective function* in **x**, and some linear *constraints* on **x**. Given two constant vectors $\mathbf{c} \in \mathbb{R}^n, \mathbf{b} \in \mathbb{R}^m$ and a constant matrix $A = (a_{ij}) \in \mathbb{R}^{m\times n}$, an LP can be formulated as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} c_i x_i \\ \text{subject to} \quad & \sum_{i=1}^{n} a_{ij} x_i \geq b_j, \quad j = 1, 2, \ldots, m && (1.1) \\ & x_i \geq 0, \quad i = 1, 2, \ldots, n && (1.2) \end{aligned}$$

The goal is to find a setting of the variables that minimizes the objective function $\sum_{i=1}^{n} c_i x_i$ and satisfies all the constraints in (1.1) and (1.2).

If there are constraints that require each variable of the vector **x** to be an integer, then it is called an *integer linear program* (ILP).

Any vector of n real numbers assigned to **x** such that all of the constraints are satisfied is called a *feasible* solution. A feasible solution **x** which minimizes the objective function is an *optimal* solution, and the corresponding value of $\sum_{i=1}^{n} c_i x_i$ is the *optimal objective value*. An LP or ILP is *infeasible* if there does not exist any feasible solution to it. Note that an LP or ILP may also be *unbounded* if the optimal solution to it does not lead to a finite optimal objective value.

Given an ILP, if we replace the integral constraint $x_i \in \{d_1^i, d_2^i, \ldots, d_{\ell_i}^i\}$ on each variable x_i, where $d_1^i < d_2^i < \ldots < d_{\ell_i}^i$ and they are all integers, by a continuous constraint $x_i \in [d_1^i, d_{\ell_i}^i]$, we obtain an LP, which is called a *relaxation* of the original ILP. Then, every feasible solution to the original ILP is also a feasible solution to its LP relaxation.

Consider an ILP for an optimization problem and its LP relaxation. Let OPT be the optimal objective value for the original ILP, OPT_f be the optimal objective value for the LP relaxation. The *integrality gap* of an ILP/LP is the worst-case ratio between OPT and OPT_f, which is $\sup_I\{\mathrm{OPT}(I)/\mathrm{OPT}_f(I)\}$ for a minimization problem or $\inf_I\{\mathrm{OPT}(I)/\mathrm{OPT}_f(I)\}$ for a maximization problem, for every possible instance I. An integrality gap of $\rho > 1$ ($\rho < 1$, respectively) indicates that based on this ILP/LP formulation, the approximation ratio α for solving the corresponding minimization (maximization, respectively) problem has a lower bound (an upper bound, respectively) of ρ.

The techniques for designing approximation algorithms based on LP relaxation include the primal-dual method, the deterministic LP rounding, the randomized LP rounding, *etc*... For more details, including the definitions and applications of these LP based techniques, one can refer to [74, 75]. We mainly aim to design *randomized rounding* algorithms each consisting of the following four steps:

Step 1. Formulate an ILP for the optimization problem;

Step 2. Provide an LP relaxation for the ILP;

Step 3. Solve the LP relaxation and obtain an optimal fractional solution $\mathbf{x}^*$;

Step 4. Round the fractional solution $\mathbf{x}^*$ to an integral solution $\mathbf{x}$ to the original ILP.

The third step of solving an LP can be done in polynomial time [47, 52, 53].

Let SOL be the objective value of the solution we obtained by a randomized rounding algorithm. Then, for a minimization problem, we have $\mathrm{OPT}_f \leq \mathrm{OPT} \leq \mathrm{SOL}$, resulting in an approximation ratio $\alpha = \mathrm{SOL}/\mathrm{OPT} \leq \mathrm{SOL}/\mathrm{OPT}_f$; for a maximization problem, we have $\mathrm{OPT}_f \geq \mathrm{OPT} \geq \mathrm{SOL}$, resulting in an approximation ratio $\alpha = \mathrm{SOL}/\mathrm{OPT} \geq \mathrm{SOL}/\mathrm{OPT}_f$.

1.2.2 Local search

To solve a combinatorial optimization problem with a *local search* technique, we design an algorithm consisting of the following steps:

Step 1. We start with an arbitrary feasible solution;

Step 2. Check a specified neighborhood of the current solution, and if a local modification can lead to another solution which can improve the objective value of the current solution, we update the solution;

Step 3. Repeat step 2 till a *specified condition* is satisfied.

In step 3, the specified condition is for the local search algorithm to eventually terminate in polynomial time. We usually have the following three ways for specifying such a condition:

1. Directly specify a bound for the number of iterations;
2. Define the neighborhood of a solution appropriately such that we can find a *local optimal* solution in polynomial time, *i.e.*, no further improvements can be made within the defined search space;
3. Compare the objective value of the current solution with the one in the previous iteration; and set a condition for the algorithm to terminate when the difference between these two objective values is small enough.

Among the above three ways, we mainly study and follow the second one.

Let SOL be the objective value of the solution obtained by a local search algorithm, and OPT be the objective value of a *global optimal* solution to the problem. The *locality gap* of a local search algorithm is the worst-case ratio between SOL and OPT, which is $\sup_I\{\mathrm{SOL}(I)/\mathrm{OPT}(I)\}$ for a minimization problem or $\inf_I\{\mathrm{SOL}(I)/\mathrm{OPT}(I)\}$ for a maximization problem, for every possible instance I. A locality gap of $\rho > 1$ ($\rho < 1$, respectively) indicates that based on this local search algorithm, the approximation ratio α for solving the corresponding minimization (maximization, respectively) problem has a lower bound (an upper bound, respectively) of ρ.

1.2.3 Amortized analysis

Amortized analysis can be considered as a way to determine the average cost of a sequence of objects, among which some objects might be much more expensive than the others.

We can also analyze the performance ratio of an algorithm through an *amortization* scheme, usually for solving an optimization problem whose goal is to minimize or maximize the *cardinality* of a feasible solution.

Consider an optimization problem of selecting a minimum (or maximum) cardinality set of elements satisfying some given constraints. Let S be a feasible solution obtained by our algorithm, with $\mathrm{SOL} := |S|$ being the cardinality of S, and let S^* be an optimal solution, with $\mathrm{OPT} := |S^*|$ being the cardinality of S^*. The general idea of amortized analysis could start with assigning one token to each element of S^*, then distribute all OPT tokens to the elements of S; or the other way around, *i.e.*, start with assigning one token to each element of S, then distribute all SOL tokens to the elements of S^*.

If we start by assigning one token to each element of S^*, then we have a total of OPT tokens to be distributed all to the elements of S. We need to find a distribution scheme such that on average every element of S can receive at least $c < 1$ token (at most $c > 1$ tokens, respectively), for some constant c, if it is a minimization (maximization, respectively) problem. Thus, for a minimization problem, we get an approximation ratio of $\alpha = \text{SOL}/\text{OPT} \leq 1/c$; for a maximization problem, we get an approximation ratio of $\alpha = \text{SOL}/\text{OPT} \geq 1/c$.

If we start by assigning one token to each element of S, then we have a total of SOL tokens to be distributed all to the elements of S^*. We need to find a distribution scheme such that on average every element of S^* can receive at most $c' > 1$ tokens (at least $c' < 1$ token, respectively), for some constant c', if it is a minimization (maximization, respectively) problem. Thus, for a minimization problem, we get an approximation ratio of $\alpha = \text{SOL}/\text{OPT} \leq c'$; for a maximization problem, we get an approximation ratio of $\alpha = \text{SOL}/\text{OPT} \geq c'$.

The major step is to find an appropriate distribution scheme. We need to determine the relationships between an optimal solution and any feasible solution obtained by our algorithm. If our algorithm is a local search, then we can always find in polynomial time a local optimal solution in a specified small neighborhood; by discovering some good properties of the problem, we would be able to find for any constant number of elements a lower or upper bound of tokens they can receive, then the average token or tokens one element can receive becomes the bound of the performance ratio of our algorithm.

1.3 Problems discussed and thesis contributions

1.3.1 The vertex happiness problems

In Chapter 2, we study the vertex *happiness* problems introduced by Zhang and Li [78] recently, which were inspired by the study on *homophyly* [36] law governing the structures of large scale networks, stating that edges in a network tend to connect nodes with the same or similar attributes. In a network where the homophyly law holds but some vertices have unknown attributes, one may consider the natural question of how to assign (or predict) attributes so that the homophyly law is followed to the greatest degree. Following this idea, and identifying attributes with colors, Zhang and Li [78] introduced the following two interesting maximization problems in terms of vertex coloring. For simplicity, they focused on the case that each vertex can have only one color.

Maximum Happy Vertices (MHV): Given a graph $G = (V, E)$ with a weight $w(v) \geq 0$ for each vertex $v \in V$, a color set $C = \{1, 2, \ldots, k\}$, and a partial vertex coloring function $c : V \mapsto C$, *i.e.*, c assigns colors to a part of the vertices in V, the goal is to color all the uncolored vertices such that the total weight of *happy* vertices is maximized. A vertex is *happy* if it shares the same color with all its neighbors; otherwise, it is *unhappy*.

Maximum Happy Edges (MHE): Given a graph $G = (V, E)$ with a weight $w(e) \geq 0$ for each edge $e \in E$, a color set $C = \{1, 2, \ldots, k\}$, and a partial vertex coloring function $c : V \mapsto C$, the goal is to color all the uncolored vertices such that the total weight of *happy* edges is maximized. An edge is *happy* if its two endpoints share the same color; otherwise, it is *unhappy*.

We observe that, if no vertex is pre-colored i, for any i, then this color i can be removed without affecting the optimum; we therefore assume without loss of generality that every color is used in the given partial vertex coloring function c.

Note that these two "vertex-coloring" problems are in fact labeling problems, and we use "color" and "label" interchangeably. They are different from the classic *graph coloring* (Coloring) problem [48], in which a feasible vertex coloring scheme must assign different colors to any adjacent vertices.

Zhang and Li [78] proved that both the MHV and MHE problems are already **NP**-hard even if the color number k is fixed. More precisely, when $k = 2$, MHV and MHE are polynomial time solvable; when $k \geq 3$, MHV and MHE are **NP**-hard.

In Section 2.2, we present a non-uniform algorithm $\mathcal{A}$ for the MHV problem which is the combination of a simple randomized version of the greedy algorithm presented in [78] and a randomized LP-rounding algorithm presented in [77]. We show that the MHV problem on general graphs can be approximated within $1/\big(\Delta + 1/g(\Delta)\big)$ by algorithm $\mathcal{A}$, where $g(\Delta) = (\sqrt{\Delta} + \sqrt{\Delta + 1})^2 \Delta > 41.7$; for *not-too-large* k, that is, when $3 \leq k \leq h(\Delta)$, the approximation ratio can be improved to $\big(1 + (\Delta - 1)/h(\Delta)\big)/\Delta$, where $h(\Delta) \geq 219.8$. This improves the previous best approximation ratio of $1/(\Delta + 1)$ [77] for MHV.

We also consider the complement of MHV, the *minimum unhappy vertices* (MUHV) problem, which is to minimize the total weight of unhappy vertices. In order to design better approximation algorithms, we introduce in Section 2.3 the following *submodular multi-labeling* (Sub-ML) and *supermodular multi-labeling* (Sup-ML) problems, which covers the MUHV and MHV problems as a special case, respectively.

Submodular Multi-Labeling (Sub-ML): Given a ground set V, a non-negative submodular set function $f : 2^V \longrightarrow \mathbb{R}_+$, with $f(\emptyset) = 0$, a set of labels $L = \{1, 2, \ldots, k\}$, and a partial labeling function $\ell : V \mapsto L$ which pre-assigns each label i to all the elements of a non-empty subset $T_i \subset V$, the goal of the Sub-ML problem is to find a partition $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ of the ground set V to minimize $f(\mathcal{S}) := \sum_{i=1}^{k} f(S_i)$, where the part S_i is the subset of elements assigned with the label i.

Supermodular Multi-Labeling (Sup-ML): Given a ground set V, a non-negative supermodular set function $f : 2^V \longrightarrow \mathbb{R}_+$, with $f(\emptyset) = 0$, a set of labels $L = \{1, 2, \ldots, k\}$, and a partial labeling function $\ell : V \mapsto L$ which pre-assigns each label i to all the elements of a non-empty subset $T_i \subset V$, the goal of the Sup-ML problem is to find a partition $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ of the ground set V to maximize $f(\mathcal{S}) := \sum_{i=1}^{k} f(S_i)$, where the part S_i is the subset of elements assigned with the label i.

In Sections 2.4 and 2.5, we show that MUHV is a special case of Sub-ML, while MHV is a special case of Sup-ML, by re-writing their objective functions as set functions. Then, by showing that the Sub-ML and Sup-ML problems can be approximated within a factor of $(2 - 2/k)$ and $2/k$, respectively, we conclude that the MUHV and MHV problems can also be approximated within a factor of $(2 - 2/k)$ and $2/k$, respectively. The $2/k$-approximation algorithm for the MHV problem further improves the previous best approximation ratio of $1/k$ [78]. Together with our first approximation for MHV [79], we improve the approximation ratio for MHV from $\max\{1/k, 1/(\Delta + 1)\}$ [77, 78] to $\max\{2/k, 1/\big(\Delta + 1/g(\Delta)\big)\}$.

On the inapproximability, we show in Section 2.4 that the MUHV problem is approximation-equivalent to the *hypergraph multiway cut* (Hyp-MC) problem [65], thus MUHV is Unique Games-hard to achieve $(2-\frac{2}{k}-\epsilon)$-approximation, for any $\epsilon > 0$. In Section 2.5, we prove that the MHV problem is Unique Games-hard to approximate within a factor of $\Omega(\log^2 k/k)$, by showing an approximation preserving reduction from the *maximum independent set* (MIS) problem [43]. These two hardness results also give evidence that it is Unique Games-hard to approximate the Sub-ML and Sup-ML problems within a factor of $(2 - \frac{2}{k} - \epsilon)$ and $\Omega(\log^2 k/k)$, respectively.

The results presented in Section 2.2 contributed as Section 3 of the paper [79] which was published by *Algorithmica*; [79] also contains some results presented in [77]. I contributed most of the work shown in Sections 2.3-2.5, and they were summarized and presented in the paper [72] which is a submission currently still under review.

1.3.2 The maximum duo-preservation string mapping problem

In Chapter 3, we study the *maximum duo-preservation string mapping* (MAX-DUO) problem [21], which is the complement of the *minimum common string partition* (MCSP) problem, a well-studied string comparison problem in computer science, with applications in fields such as data compression and bioinformatics. In both data compression and bioinformatics, string (or sequence) comparison is a routine work. For the similarity between two strings, a commonly used measure is the *edit distance*, which is the minimum number of operations required to transform one string into the other. At the finest scale, the edit operations involve a single character of a string, including insertion, deletion, and substitution. When comparing two long strings such as the whole genomes of multiple species, long range operations become more interesting, leading to the genome rearrangement problems [22, 69]. In particular, consider a transportation operation to cut out a substring and insert it back to another position in the string. The problem of partitioning one string into a minimum number of substrings such that a reshuffle of them becomes the other string is then referred to as the MCSP problem. MCSP was first formally introduced by Goldstein *et al.* [46] as follows.

Minimum Common String Partition (MCSP): Consider two length-n strings $A = (a_1, a_2, \ldots, a_n)$ and $B = (b_1, b_2, \ldots, b_n)$ over some alphabet Σ, such that B is a permutation of A. A *partition* of A, denoted as $\mathcal{P}_A$, is a multi-set of substrings whose concatenation in a certain order becomes A. The number of substrings in $\mathcal{P}_A$ is the *cardinality* of $\mathcal{P}_A$. The MCSP problem asks for a minimum cardinality partition $\mathcal{P}_A$ of A that is also a partition of B. k**-MCSP** is the restricted version of MCSP when every letter of the alphabet Σ occurs at most k times in each of the two given strings.

The MCSP problem is **NP**-hard and **APX**-hard even when $k = 2$ [46]. The current best result is an $O(\log n \log^* n)$-approximation [28] for the general MCSP and an $O(k)$-approximation [57] for k-MCSP.

The complement of MCSP, referred to as the *maximum duo-preservation string mapping* (MPSM) problem by Chen *et al.* [21] can be defined as follows, while we call this problem as MAX-DUO instead (mostly because the acronym MPSM looks too similar to the other acronyms).

Maximum Duo-preservation String Mapping (Max-Duo): Given a string, an ordered pair of consecutive letters is called a *duo* [46]; a length-ℓ substring in a partition *preserves* $\ell - 1$ duos of the given string. The Max-Duo problem is to maximize the number of duos preserved in the common partition. ***k*-Max-Duo** is the restricted version of Max-Duo when every letter of the alphabet Σ occurs at most k times in each of the two given strings.

Boria *et al.* [10] proved that 2-Max-Duo is **APX**-hard, similar to 2-MCSP [46], via a linear reduction from MIS on cubic graphs. The Max-Duo problem can actually be cast as a special case of the well-known *maximum independent set* (MIS) problem [43]; in particular, Boria *et al.* [10] showed that an instance of k-Max-Duo translates to a graph with the maximum degree $\Delta \leq 6(k-1)$. It follows that the state-of-the-art $\big((\Delta + 3)/5 + \epsilon\big)$-approximation algorithm for MIS [7], for any $\epsilon > 0$, is a $\big((6k - 3)/5 + \epsilon\big)$-approximation algorithm for k-Max-Duo. Especially, 2-Max-Duo and 3-Max-Duo can be approximated within a ratio of $1.8 + \epsilon$ and $3 + \epsilon$, respectively, for any $\epsilon > 0$. The previous best result on general Max-Duo is a 3.25-approximation presented by Brubach [12].

We remark again that in order to be consistent with the results presented in the literature, all the approximation ratios for the maximization problems Max-Duo and k-Max-Duo in this section are actually $1/\alpha$, instead of α as defined in Definition 1.8. To keep consistency, we also use $1/\rho$ to express a locality gap if the approximation ratio is expressed by $1/\alpha$.

In Section 3.3, by studying the local optimal properties of the 2-Max-Duo problem, we present a vertex-degree reduction technique and show that 2-Max-Duo can be approximated arbitrarily close to 1.4. This improves the previous best results of $1.8 + \epsilon$, for $\epsilon > 0$, by directly applying the state-of-the-art $\big((\Delta + 3)/5 + \epsilon\big)$-approximation algorithm for MIS [10], for any $\epsilon > 0$, where Δ is the maximum vertex degree of the input graph.

In Section 3.4, we present an improved local search approximation algorithm for Max-Duo and showed that its performance ratio is no greater than $35/12 < 2.917$, which beats the previous best 3.25-approximation for Max-Duo [12], while the current best approximation ratio for Max-Duo is $2 + \epsilon$ [33], for any $\epsilon > 0$. The performance analysis of our 2.917-approximation algorithm is done through a complex yet interesting amortization.

This result presented in Section 3.3 formed the paper [70] which was presented on the 28*th International Symposium on Algorithms and Computation (ISAAC 2017)* and later submitted as the paper [25] which is currently still under review; I contributed most of the work to this result. The results presented in Section 3.4 were summarized and presented in the paper [71]; together with my supervisor Dr. Lin and the other collaborators, we desinged the algorithm and proved the performance based on many discussions.

1.3.3 The path partition problem

In Chapter 4, we investigate the *k-path partition* (k-PP) problem, which is motivated by the data integrity of communication in wireless sensor networks and several other applications and was first considered by Yan et al. [76]. One can consider a *broadcasting* problem in data communication networks. Given some information, by modeling a data communication network with a graph, *broadcasting* is to transmit the information from some vertices to all the other vertices in the network only through paths, that is, one vertex can only transmit the information to its adjacent vertices through the edge connecting them. It requires one unit of time to transmit the information from one vertex to another through an edge. The goal is to select the minimum number of vertices such that the information can be transmitted from those selected vertices to all the other vertices within a fixed number of time units. This is an application of the k-PP problem, which can be formally defined as follows.

k-Path Partition (k-PP): Given a simple graph $G = (V, E)$, the *order* of a simple path in G is the number of vertices on the path and it is called a *k-path* if its order is k. The k-PP problem asks to find a minimum collection of vertex-disjoint paths of order at most k such that every vertex is on some path in the collection.

Clearly, the 2-PP problem is exactly the Maximum Matching problem, which is solvable in $O(m\sqrt{n}\log(n^2/m)/\log n)$-time [45]. For $k \geq 3$, k-PP is **NP**-hard [76]. To the best of our knowledge, there is no approximation algorithm with proven performance for the general k-PP problem, except the trivial k-approximation using all 1-paths.

The k-PP problem is closely related to the *k-set cover* (k-SC) problem defined as follows.

k-Set Cover (k-SC): Given a universe $U = \{x_1, x_2, \ldots, x_n\}$ of n elements and a collection of subsets $C = \{S_1, S_2, \ldots, S_m\}$ of U, with every $S_i \in C$ has size at most k, the goal is to find a minimum sub-collection of C that *covers* all the elements of U.

The k-SC problem is a variant of the well-known *set cover* (SC) problem, which is one of the first proven **NP**-hard problems [43]. k-SC is **APX**-complete and admits an $(H_k - \frac{1}{2})$-approximation (for $k \geq 3$) [35] and an $(H_k - \frac{196}{390})$-approximation (for $k \geq 4$) [61].

For the k-PP problem with $k = 3$, Monnot and Toulouse [64] proposed a $3/2$-approximation, based on two maximum matchings. In Section 4.2, we present a local search algorithm Approx which first applies Algorithm A to computes a 3-path partition with the least 1-paths, and then applies Algorithm B to iteratively perform several replacement operations to reduce the total number of 2- and 3-paths. We point out that ALGORITHM A is already a $k/2$-approximation for the general k-PP problem. in Section 4.3, we show by an amortization scheme that APPROX is a $4/3$-approximation. The ratio of $4/3$ coincidentally meets the current best approximation ratio for the 3-SC problem.

The result of the $k/2$-approximation presented in Section 4.2.1 contributed as the first half of the paper [24] which was published by the *Journal of Combinatorial Optimization* (JOCO); this result was contributed mainly by discussions with my supervisor Dr. Lin and the other collaborators. The local search algorithm and the performance analysis presented in Sections 4.2.2 and 4.3 were summarized and presented in the paper [23] which is a submission currently still under review; I contributed most of the work to the design and anlysis of this $4/3$-approximation, based on some discussions with Dr. Lin and the other collaborators.

Chapter 2

The Vertex Happiness Problems[1]

2.1 Introduction

In this chapter, we investigate the vertex *happiness* problem recently introduced by Zhang and Li [78], which was inspired by the study on *homophyly* [36, Chapter 4] law stating that in a (large scale) network, the nodes have a tendency of connecting with nodes that share the similar attributes with them. Consider a network in which only a part of the nodes are assigned with some attributes, and assume the homophyly law holds, it would be common to ask how to assign attributes to the rest of the nodes so that the homophyly law could be followed to the maximum extent. By identifying attributes with colors, Zhang and Li [78] introduced the *maximum happy vertices* (MHV) problem and the *maximum happy edges* (MHE), which can be defined formally as follows. In both problems, each vertex can only be assigned with one color.

Maximum Happy Vertices (MHV): Given a graph $G = (V, E)$ with a non-negative weight $w(v)$ for each vertex $v \in V$, a color set $C = \{1, 2, \ldots, k\}$, and a partial vertex coloring function $c : V \mapsto C$, *i.e.*, c assigns colors only to a part of the vertices in V, the goal is to color all the uncolored vertices such that the total weight of *happy* vertices is maximized. A vertex is *happy* if it shares the same color with all its neighbors in the coloring scheme; otherwise, it is *unhappy*. (See Figure 2.1 for an instance along with a coloring scheme for the MHV problem.)

Maximum Happy Edges (MHE): Given a graph $G = (V, E)$ with a non-negative weight $w(e)$ for each edge $e \in E$, a color set $C = \{1, 2, \ldots, k\}$, and a partial vertex coloring function $c : V \mapsto C$, *i.e.*, c assigns colors only to a part of the vertices in V, the goal is to color all the uncolored vertices such that the total weight of *happy* edges is maximized. An edge

[1]This chapter is based on two papers [72, 79]. [79] is a work with Zhang, Jiang, Li, Lin, and Miyano, "Improved approximation algorithms for the maximum happy vertices and edges problems", which was published by Algorithmica; [72] is a work with Chen, Zhang, and Goebel, "Approximation algorithms for the vertex happiness", which is a submission under review, while there is an old version available publicly at arXiv [73], covering most of the results in [72].

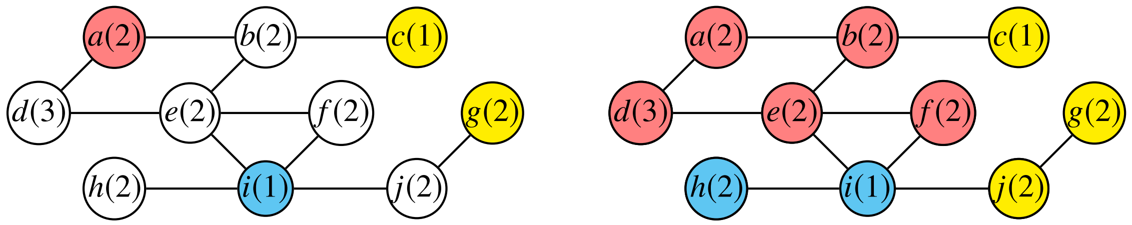


FIGURE 2.1: An instance of MHV on the left and a coloring scheme on the right, where the integer in the parenthesis on each vertex is its weight. In this coloring scheme, the total weight of happy vertices is $w(a) + w(d) + w(h) + w(g) = 9$.

is *happy* if its two endpoints share the same color in the coloring scheme; otherwise, it is *unhappy*.

We only study the MHV problem and its complement, the *minimum unhappy vertice* (MUHV) problem, which is to minimize the total weight of unhappy vertices in the given graph. In both the MHV and MUHV problems, there could be multiple vertices in the given graph pre-colored the same color. When only one vertex is pre-colored by the partial vertex coloring function c for each $i \in C$, we denote these two problems as the *restricted-MHV* and the *restricted-MUHV* problems, respectively. For MUHV, there is a polynomial time reduction from the general MUHV problem to the restricted-MUHV problem, by creating a vertex for each $i \in C$ with a weight large enough, pre-coloring it with the color i, connecting it to all the vertices pre-colored i, and uncoloring those vertices (see a detailed proof in Section 2.4). This reduction implies that the restricted-MUHV problem and the general MUHV problem are approximation-equivalent.

We remark that the vertex-coloring problems we study here are in fact labeling problems, and we use "color" and "label" interchangeably; they are different from the classic *graph coloring* (COLORING) problem [48], in which a feasible vertex coloring scheme must assign different colors to any adjacent vertices. We also note that, if no vertex is pre-colored i, for any i, then this color i can be removed without affecting the optimum; we therefore assume without loss of generality that every color is used in the given partial vertex coloring function c.

Given a graph $G = (V, E)$ with the vertex set V and the edge set E, for any subset $X \subseteq V$, define the *boundary* of X, denoted as $\partial(X)$, to be the subset of vertices of X each having at least one neighbor outside of X. Let $\iota(X) = X - \partial(X)$, which is called the *interior* of X. In a coloring scheme, let S_i denote the subset of all the vertices colored i; then every vertex of $\partial(X)$ is unhappy while all vertices of $\iota(S_i)$ are happy. We extend the vertex weight function to subsets of vertices, that is, $w(X) := \sum_{v \in X} w(v)$ for any $X \subseteq V$; and we define the set function $f_b(\cdot)$ as

$$f_b(X) \coloneqq w(\partial(X)),\ \forall X \subseteq V. \tag{2.1}$$

A vertex coloring scheme one-to-one corresponds to a partition $\mathcal{S} = \{S_1, S_2, \dots, S_k\}$ of the vertex set V, where each part S_i contains all the vertices colored i. This way, the MUHV problem can be cast as finding a partition $\mathcal{S}$ such that $f_b(\mathcal{S}) \coloneqq \sum_{i=1}^{k} f_b(S_i)$ is minimized.

It is not hard to validate (see a detailed proof in Section 2.4) that the boundary $\partial(\cdot)$ of a vertex subset in the given graph $G = (V, E)$ has the following properties for any two subsets $X, Y \subseteq V$:

(i) $\partial(\emptyset) = \emptyset$;

(ii) $\partial(X \cap Y) \subseteq \partial(X) \cup \partial(Y)$;

(iii) $\partial(X \cup Y) \subseteq \partial(X) \cup \partial(Y)$; and

(iv) $\partial(X \cap Y) \cap \partial(X \cup Y) \subseteq \partial(X) \cap \partial(Y)$.

Therefore, the set function $f_b : 2^V \to \mathbb{R}$ defined in Eq. (2.1) satisfies $f_b(X) + f_b(Y) \geq f_b(X \cap Y) + f_b(X \cup Y)$, for any two subsets $X, Y \subseteq V$ (see a detailed proof in Section 2.4). That is, $f_b(\cdot)$ is a *submodular* [63] function on the set V. This way, the MUHV problem can be cast as a special case of the following *submodular multi-labeling* (Sub-ML) problem:

Given a ground set V, a non-negative submodular set function $f : 2^V \to \mathbb{R}_+$, with $f(\emptyset) = 0$, a set of labels $L = \{1, 2, \dots, k\}$, and a partial labeling function $\ell : V \mapsto L$ which pre-assigns each label i to all the elements of a non-empty subset $T_i \subset V$, the goal of the Sub-ML problem is to find a partition $\mathcal{S} = \{S_1, S_2, \dots, S_k\}$ of the ground set V to minimize $f(\mathcal{S}) = \sum_{i=1}^{k} f(S_i)$, where the part S_i is the subset of elements assigned with the label i.

Conversely, given the graph $G = (V, E)$, we define another set function $f_p(\cdot)$ as

$$f_p(X) \coloneqq w(\iota(X)),\ \forall X \subseteq V. \tag{2.2}$$

Then $f_p(X) = w(X) - f_b(X)$ for any subset $X \subseteq V$ and consequently $f_p(\cdot)$ is a *supermodular* [63] function on the set V. Thus, the MHV problem can be cast as finding a partition $\mathcal{S} = \{S_1, S_2, \dots, S_k\}$ of the vertex set V such that $f_p(\mathcal{S}) = \sum_{i=1}^{k} f_p(S_i)$ is maximized, where each part S_i contains all the vertices colored i; it can also be cast as a special case of the *supermodular multi-labeling* (Sup-ML) problem that can be analogously defined.

2.1.1 Related work

Classification problems have been formulated as cuts, partition, labeling, or coloring, and have been widely studied for a very long time.

Zhang and Li [78] proved that both the MHV and MHE problems are already **NP**-hard even if the color number k is fixed. More precisely, when $k = 2$, MHV and MHE are polynomial time solvable; when $k \geq 3$, MHV and MHE are **NP**-hard.

For the unit weight version of MHV, Zhang and Li [78] presented two approximation algorithms. One algorithm is based on a greedy approach, whose approximation ratio is $1/k$, which is actually a $1/k$-approximation for the general MHV problem (the weighted version); the other algorithm is based on a subset-growth technique, whose approximation ratio is $\Omega(1/\Delta^3)$, where Δ is the maximum vertex degree of the input graph. Later, Zhang *et al.* [77] presented an improved algorithm with an approximation ratio of $1/(\Delta + 1)$ based on an LP relaxation (LP-MHV) shown as follows.

Let $y_j^i := y^i(v_j)$ indicate whether vertex v_j is colored i, x_j^i indicate whether the vertex v_j is happy by color i, x_j indicate whether the vertex v_j is happy, and $B(v_j) = N[v_j]$ be the closed neighborhood of the vertex v_j.

$$\begin{aligned}
\text{maximize} \quad & \sum_{j=1}^{n} w_j x_j && && \text{(LP-MHV)}\\
\text{subject to} \quad & \sum_{i=1}^{k} y_j^i = 1, && \forall v_j \in V && (2.3)\\
& y_j^i = 1, && \forall v_j \in V,\ \forall i \in C \text{ s.t. } c(v_j) = i && (2.4)\\
& x_j^i = \min_{v_\ell \in B(v_j)} \{y_\ell^i\}, && \forall v_j \in V,\ \forall i \in C && (2.5)\\
& x_j = \sum_{i=1}^{k} x_j^i, && \forall v_j \in V && (2.6)\\
& x_j,\ x_j^i,\ y_j^i \geq 0, && \forall v_j \in V,\ \forall i \in C && (2.7)
\end{aligned}$$

Note that the set of constraints 2.5 is equivalent to the linear constraints $x_j^i \leq y_\ell^i, \forall v_j \in V, \forall i \in C, v_\ell \in B(v_j)$.

Based on this LP relaxation (LP-MHV), Zhang *et al.* [77] presented a randomized rounding algorithm using the rounding scheme proposed by Kleinberg and Tardos [55].

In summary, the previous best approximation ratio for the MHV problem is $\max\{1/k, 1/(\Delta + 1)\}$ [77, 78], where Δ is the maximum vertex degree of the input graph. For the complementary MUHV problem, to the best of our knowledge, it hasn't been studied in the literature.

Recall that the MHV and the MUHV problems are a special case of the Sup-ML and the Sub-ML problems, respectively. We again remind the readers that in an instance of these multi-labeling problems, each label is pre-assigned to at least one element and to multiple elements in general. A restricted version of the Sub-ML problem, when each label is pre-assigned to exactly one element, is the *submodular multiway partition* (Sub-MP) problem [80], which has received a lot of studies. The restricted-MUHV problem is a special case of the Sub-MP problem.

The Sub-MP problem was first studied by Zhao *et al.* [80], who presented a $(k - 1)$-approximation algorithm. Years later, Chekuri and Ene [18] proposed a convex relaxation for Sub-MP by using the Lovász extension, and they presented a 2-approximation based on this relaxation. This was further improved to a $(2-2/k)$-approximation shortly after by Ene *et al.* [37], which immediately gives a $(2 - 2/k)$-approximation for the restricted-MUHV and the general MUHV problems. On the inapproximability, Ene *et al.* [37] proved that any $(2-2/k-\epsilon)$-approximation for Sub-MP requires exponentially many value queries for any $\epsilon > 0$, or otherwise it implies $\mathbf{NP} = \mathbf{RP}$.

It is important to note that although the restricted-MUHV problem and the general MUHV problem are approximation-equivalent, we cannot simply conclude that the Sub-MP problem and the Sub-ML problem are also approximation-equivalent based on similar reduction proofs. The difference between Sub-MP and Sub-ML depends heavily on how the set function $f(\cdot)$ is defined; a change to the ground set V could alter the optimal solution value and a feasible solution value differently.

The Sub-MP problem includes many well studied cut problems including the classic (edge-weighted) *multiway cut* (MC) problem [30], the *node-weighted multiway cut* (Node-MC) problem [44] and the *hypergraph multiway cut* (Hyp-MC) problem [65], all defined in the following, as special cases.

Multiway Cut (MC): Given a graph $G = (V, E)$ with a non-negative weight $w(e)$ for each edge $e \in E$ and a set $T = \{t_1, t_2, \dots, t_k\} \subseteq V$ of k distinct terminals, the goal is to remove a minimum weight set of edges $F \subseteq E$ such that no two distinct terminals in T are connected in $(V, E - F)$.

Node-weighted Multiway Cut (Node-MC): Given a graph $G = (V, E)$ with a non-negative weight $w(v)$ for each vertex $v \in V$ and a set $T = \{t_1, t_2, \ldots, t_k\} \subseteq V$ of k distinct terminals, the goal is to remove a minimum weight set of vertices $V' \subseteq V$ such that no two distinct terminals in T are connected in the subgraph of G induced by $V - V'$.

Hypergraph Multiway Cut (Hyp-MC): Given a hypergraph $H = (V_H, E_H)$ with a non-negative weight $w(e)$ for each hyperedge $e \in E_H$ and a set of k terminals $T = \{t_1, t_2, \ldots, t_k\} \subseteq V$, the Hyp-MC problem asks to remove a minimum-weight set of hyperedges so that every two terminals are disconnected.

The classic MC problem is **NP**-hard for $k \geq 3$ even if all edges have unit weight [30]. Dahlhaus *et al.* [30] also showed that MC is **APX**-hard and gave a first combinatorial algorithm which achieves a $(2 - 2/k)$-approximation. After that, there have been many approximation algorithms designed and analyzed [13, 17, 29, 42, 51, 68] for MC, most of which are based on the *CKR relaxation* presented by Călinescu *et al.* [17] shown as follows.

Let $\mathbf{e}_i = (0, \ldots, 0, 1, 0, \ldots, 0)$ be the vector with 1 in the i-th coordinate and zeros elsewhere, and Δ_k be the *k-dimensional simplex*, *i.e.*, $\Delta_k = \{x \in \mathbb{R}^k \mid \sum_{i=1}^{k} x_i = 1\}$. The CKR relaxation can be formulated as

$$\begin{aligned} \text{maximize} \quad & \sum_{e=(u,v)\in E} c_e \cdot \|x_u - x_v\|_1 && \text{(LP-CKR)} \\ \text{subject to} \quad & x_{t_i} = \mathbf{e}_i, \qquad i = 1, 2, \ldots, k \\ & x_u \in \Delta_k, \quad \forall u \in V \end{aligned}$$

where $\|x_u - x_v\|_1 = \sum_{i=1}^{k} |x_u^i - x_v^i|$.

Based on the CKR relaxation, Călinescu *et al.* [17] presented a randomized rounding algorithm for MC with an approximation ratio of $(3/2 - 1/k)$, which was further improved to 1.3438 as k goes to infinity by Karger *et al.* [51]. For $k = 3$ specifically, Cunnigham and Tang [29] and Karger *et al.* [51] independently presented a 12/11-approximation and showed that it is the best approximation achievable using the CKR relaxation for $k = 3$ by giving an integrality gap example of ratio $12/11 - \epsilon$ for any $\epsilon > 0$. Later, Buchbinder *et al.* [13] introduced a new rounding scheme for the CKR relaxation and presented an elegant 4/3-approximation algorithm for general k and further improved it to 1.3239. The current best approximation ratio for MC is 1.2965 [68] which is difficult and only verified by computer, still based on the same CKR relaxation; while recently, Buchbinder *et al.* [14]

designed a much simpler algorithm which yields an approximation ratio of 1.2969, which roughly matches the 1.2965 approximation guarantee in [68]. On the negative side, a lower bound of $8/\big(7+1/(k-1)\big)$ on the integrality gap of (LP-CKR) has been proved by Freund and Karloff [42], which was improved to $6/\big(5+1/(k-1)\big)$ recently by Angelidakis *et al.* [1].

The Hyp-MC problem and the Node-MC problem are actually approximation-equivalent, and they both admit a $(2-2/k)$-approximation [44, 65]; on the negative side, they are proven more difficult to approximate, that it is Unique Games-hard to achieve a $(2-2/k-\epsilon)$-approximation, for any $\epsilon > 0$ [37].

The complement of the Sub-MP problem, called the *supermodular multiway partition* (Sup-MP) problem, can be defined similarly. The restricted-MHV problem is then a special case of Sup-MP, which also includes the *multiway uncut* (MUC) problem [59] defined in the following, as a special case, where the k terminals in the input graph can be considered as k elements each being pre-assigned with a distinct label.

Multiway Uncut (MUC): the complement of MC, whose goal is to find a partition $\{V_1, V_2, \ldots, V_k\}$ of V such that for each i, t_i is contained in V_i and the total weight of edges not cut by the partition is maximized.

The MUC problem seems only studied by Langberg *et al.* [59], who presented a 0.8535-approximation based on an LP relaxation with a randomized rounding algorithm also using the rounding technique proposed by Kleinberg and Tardos [55].

When generalizing the MUC problem to pre-assign multiple terminals in a part of the vertex partition, it becomes the MHE problem, which is a special case of the Sup-ML problem. Zhang and Li [78] proposed a 1/2-approximation algorithm for the unit weight version of MHE based on a combinatorial partitioning strategy; by using an extended LP relaxation of the one presented for the MUC and adopting the rounding technique proposed by Kleinberg and Tardos [55], Zhang *et al.* [77] later proved that MHE can be approximated within $1/2 + (\sqrt{2}/4)h_1(k) \geq 0.8535$, where $h_1(k) \geq 1$ is a function of k. This is the previous best approximation ratio for the MHE problem (also for the MUC problem) on general graphs.

More broadly, the multi-labeling problems can be viewed as special cases of the *cost allocation* [19] problem, in which k different non-negative set functions are given for evaluating the k parts of the partition separately; they are also closely related to the *optimal allocation* problem [32, 39, 40, 54, 60] in combinatorial auctions, where no elements are necessarily pre-assigned a label but the set function (called utility function) is assumed monotone in general.

2.1.2 Our contributions

Our target problems are the MHV and the MUHV problems, and we aim to design improved approximation algorithms for them and to prove the hardness results in approximability.

First, we study the MHV problem. By combining a simple randomized version of the greedy $1/k$-approximation algorithm for MHV presented in [78] and the randomized $1/(\Delta + 1)$-approximation algorithm for MHV presented in [77], we give a non-uniform algorithm, and with deeper analysis, we show an approximation ratio of $1/\big(\Delta + 1/g(\Delta)\big)$, where Δ is the maximum vertex degree of the input graph and $g(\Delta) = (\sqrt{\Delta} + \sqrt{\Delta + 1})^2\Delta > 4\Delta^2$. This improves the previous best approximation ratio of $\max\{1/k, 1/(\Delta + 1)\}$ [77, 78].

Next, we show that the convex relaxation on the Lovász extension for the Sub-MP problem [18] can be extended for the Sub-ML problem; therefore the same approximation algorithm works for Sub-ML with a performance ratio of $(2 - 2/k)$. Analogously, we present the concave relaxation on the Lovász extension for the Sup-ML problem, thus showing that Sup-ML can be approximated within a factor of $2/k$. Therefore, the MUHV problem can be approximated within a factor of $(2 - 2/k)$ and the MHV problem can be approximated within a factor of $2/k$; the $2/k$-approximation for MHV improves the previous best ratio of $\max\{1/k, 1/\big(\Delta + 1/g(\Delta)\big)\}$, where Δ is the maximum vertex degree of the input graph and $g(\Delta) = (\sqrt{\Delta} + \sqrt{\Delta + 1})^2\Delta > 4\Delta^2$.

For the MUHV problem, the $(2 - 2/k)$-approximation can also be obtained due to its approximation-equivalent to the restricted-MUHV problem, which is a special case of Sub-MP. We also prove that the MUHV problem is approximation-equivalent to the Hyp-MC problem [65], thus MUHV is Unique Games-hard to approximate within a factor of $(2 - 2/k - \epsilon)$, for any $\epsilon > 0$. This hardness result gives another evidence that it is Unique Games-hard to achieve a $(2 - 2/k - \epsilon)$-approximation for the general Sub-ML problem, for any $\epsilon > 0$.

For the MHV problem, we show that the LP relaxation for the MHV problem presented in [77], called LP-MHV, is equivalent to the concave relaxation for the Sup-ML problem based on the Lovász extension to the set function $f_p(\cdot)$ defined in Eq. (2.2). We then prove an upper bound of $2/k$ on the integrality gap of LP-MHV, and conclude that the $2/k$-approximation is the best possible based on LP-MHV. On the inapproximability of MHV, we prove that it is Unique Games-hard to approximate within a factor of $\Omega(\log^2 k/k)$, by showing an approximation preserving reduction from the maximum independent set problem [43]. This hardness result also gives another evidence that it is Unique Games-hard to achieve an $\Omega(\log^2 k/k)$-approximation for the general Sup-ML problem.

2.1.3 Organization

The remainder of this chapter is organized as follows. In Section 2.2, we present a non-uniform algorithm for MHV, which is a combination of a simple randomized version of the greedy $1/k$-approximation [78] and the $1/(\Delta+1)$-approximation [77]; with deeper analysis, we show that it is a $1/\big(\Delta + 1/g(\Delta)\big)$-approximation, where Δ is the maximum vertex degree of the input graph and $g(\Delta) = (\sqrt{\Delta} + \sqrt{\Delta+1})^2\Delta > 4\Delta^2$. In Section 2.3, we introduce some basic notions such as the Lovász extension to a set function; we then present the relaxation based on the Lovász extension for the Sub-ML problem and a similar relaxation for the Sup-ML problem. We also present the approximation algorithm using the same randomized rounding technique for the Sub-MP problem in [37], and conclude that it is also a $(2 - 2/k)$-approximation for the Sub-ML problem and it is a $2/k$-approximation for the Sup-ML problem. In Section 2.4, we study the MUHV problem which admits a $(2 - 2/k)$-approximation, and further show that it is approximation-equivalent to the hypergraph multiway cut problem, thus MUHV is Unique Games-hard to approximate within a factor of $(2-2/k-\epsilon)$, for any $\epsilon > 0$. In Section 2.5, we study the MHV problem, by firstly introducing the LP relaxation formulated in [77], then showing its equivalence to the relaxation based on the Lovász extension to the set function $f(\cdot)$ defined in Eq. (2.1), and proving an upper bound of $2/k$ on the integrality gap; lastly, we prove an inapproximability result for MHV that it is Unique Games-hard to achieve an $\Omega(\log^2 k/k)$-approximation. We conclude this chapter in Section 2.6, along with some possible future work.

2.2 A $1/\big(\Delta + 1/g(\Delta)\big)$-approximation for MHV

Consider the LP relaxation (LP-MHV) presented by Zhang *et al.* [77] for the MHV problem on a general graph $G = (V, E)$. Let Δ be the maximum vertex degree of graph G, then $|B(v)| \leq \Delta + 1$. [2] We may safely assume $k \geq \Delta + 1$, since otherwise, the $1/k$-approximation would be better than our $1/\big(\Delta + 1/g(\Delta)\big)$-approximation.

Denote Algorithm $\mathcal{P}$ as the simple randomized version of the greedy $1/k$-approximation algorithm for MHV presented in [78] (see Figure 2.7 for a high-level description), and

[2]The problem with $\Delta \leq 2$ is trivial, so in the following analysis, we will assume $\Delta \geq 3$.

ALGORITHM $\mathcal{R}$ as the randomized $1/(\Delta + 1)$-approximation algorithm for MHV presented in [77] (see Figure 2.3 for a high-level description).

```
ALGORITHM $\mathcal{P}$
1: Pick a color $i \in \{1, 2, \ldots, k\}$ uniformly at random.
2: Color all the uncolored vertices in $i$.
```

FIGURE 2.2: A high-level description of ALGORITHM $\mathcal{P}$ for the MHV problem.

```
ALGORITHM $\mathcal{R}$
1: Solve (LP-MHV) to obtain an optimal solution $(x, y)$.
2: while there exists some uncolored vertex do
3:     Pick a color $i \in \{1, 2, \ldots, k\}$ uniformly at random.
4:     Pick a parameter $\rho \in [0, 1]$ uniformly at random.
5:     For each uncolored vertex $v_j$, if $y_j^i \geq \rho$, then color $v_j$ in $i$.
6: end while
```

FIGURE 2.3: A high-level description of ALGORITHM $\mathcal{R}$ for the MHV problem.

Our final algorithm for MHV is a randomized non-uniform algorithm, denoted as ALGORITHM $\mathcal{A}$, shown in Figure 2.4.

```
ALGORITHM $\mathcal{A}$
1: With probability $\lambda$ run ALGORITHM $\mathcal{R}$, and with probability $1 - p$ run ALGORITHM $\mathcal{P}$.
2: Return the coloring found (by either $\mathcal{R}$ or $\mathcal{P}$) in step 1.
```

FIGURE 2.4: The non-uniform ALGORITHM $\mathcal{A}$ for the MHV problem.

If we can make a good balance between ALGORITHM $\mathcal{P}$ and ALGORITHM $\mathcal{R}$ in step 1 of ALGORITHM $\mathcal{A}$ by carefully choosing the probability λ, we may get a ratio for MHV better than $1/(\Delta + 1)$. We adopt the same analysis scheme for ALGORITHM $\mathcal{R}$ from [77] which was motivated by the analysis ideas in [55, 59].

In ALGORITHM $\mathcal{R}$, each execution of steps 3 to 5 is called a *round*. A *ball* $B(v)$ is called a *blank* ball if it contains no colored vertices, and it is called a *monochrome* ball if it contains only one color. At the beginning of ALGORITHM $\mathcal{R}$, let $\mathcal{B}_0$ be the set of vertices whose $B(v)$'s are blank balls, and $\mathcal{B}_1$ be the set of vertices whose $B(v)$'s are monochrome balls. Then, only vertices in $\mathcal{B}_0 \cup \mathcal{B}_1$ can become happy.

For $\ell \in \{0,1\}$, let P_ℓ, R_ℓ, and A_ℓ be the total weight of happy vertices in $\mathcal{B}_\ell$ found by Algorithm $\mathcal{P}$, $\mathcal{R}$, and $\mathcal{A}$, respectively. We have

$$\begin{aligned} \mathrm{E}[A_\ell] &= (1-\lambda)\mathrm{E}[P_\ell] + \lambda \mathrm{E}[R_\ell] \\ &= (1-\lambda)\sum_{v_j \in \mathcal{B}_\ell} w_j \Pr[v_j \text{ is happy in } \mathcal{P}] + \lambda \sum_{v_j \in \mathcal{B}_\ell} w_j \Pr[v_j \text{ is happy in } \mathcal{R}], \end{aligned} \tag{2.8}$$

Let SOL be the total weight of happy vertices found by Algorithm $\mathcal{A}$. Then we have

$$\mathrm{E}[\mathrm{SOL}] = \mathrm{E}[A_0] + \mathrm{E}[A_1]. \tag{2.9}$$

2.2.1 Probability that a vertex in $\mathcal{B}_0$ becomes happy

For a vertex $v \in \mathcal{B}_0$, by Algorithm $\mathcal{P}$, we get

$$\Pr[v \text{ is happy in } \mathcal{P}] = 1. \tag{2.10}$$

By Algorithm $\mathcal{R}$, define the following events for $v \in \mathcal{B}_0$. $N^0_{<r}$: all vertices in $B(v)$ are not colored before the r-th round, N^0_r: all vertices in $B(v)$ are not colored in the r-th round, A^0_r: all vertices in $B(v)$ are colored in the r-th round, and E^0_r: there exists a vertex in $B(v)$ that is colored in the r-th round. The following Lemma 2.1 has already been proved in [77].

Lemma 2.1 ([77])**.** *Let v_j be a vertex in $\mathcal{B}_0$, then* $\Pr[A^0_r | N^0_{<r}] \geq \frac{x_j}{k}$.

Lemma 2.2. *Let v_j be a vertex in $\mathcal{B}_0$, then* $\Pr[E^0_r | N^0_{<r}] \leq \frac{\Delta+1-\Delta x_j}{k}$.

Proof. By step 5 of Algorithm $\mathcal{R}$, we have

$$\Pr[E^0_r | N^0_{<r}] = \sum_i \frac{1}{k} \max_{v_\ell \in B(v_j)} \{y^i_\ell\}. \tag{2.11}$$

Note that $1 \leq |B(v_j)| \leq \Delta + 1$. By simple calculation, we have

$$\max_{v_\ell \in B(v_j)} \{y^i_\ell\} \leq \sum_{v_\ell \in B(v_j)} y^i_\ell - (|B(v_j)| - 1) \min_{v_\ell \in B(v_j)} \{y^i_\ell\},$$

which implies

$$\begin{aligned}
\sum_i \max_{v_\ell \in B(v_j)} \{y_\ell^i\} &\le \sum_i \sum_{v_\ell \in B(v_j)} y_\ell^i - (|B(v_j)| - 1) \sum_i \min_{v_\ell \in B(v_j)} \{y_\ell^i\} \\
&= \sum_{v_\ell \in B(v_j)} \sum_i y_\ell^i - (|B(v_j)| - 1) \sum_i \min_{v_\ell \in B(v_j)} \{y_\ell^i\} \\
&= |B(v_j)| - (|B(v_j)| - 1)x_j = |B(v_j)|(1 - x_j) + x_j \\
&\le (\Delta + 1)(1 - x_j) + x_j = \Delta + 1 - \Delta x_j, \qquad (2.12)
\end{aligned}$$

where the second inequality holds due to the constraints in (LP-MHV).

Therefore, by (2.11) and (2.12), we have

$$\Pr[E_r^0 | N_{<r}^0] \le \frac{\Delta + 1 - \Delta x_j}{k}.$$

□

Lemma 2.3. *Let v_j be a vertex in $\mathcal{B}_0$, then* $\Pr[v \text{ is happy in } \mathcal{R}] \ge \dfrac{x_j}{\Delta + 1 - \Delta x_j}$.

Proof. First we note that

$$\begin{aligned}
\Pr[v_j \text{ is happy in } \mathcal{R}] &\ge \sum_{r=1}^{\infty} \Pr[N_{<r}^0] \cdot \Pr[A_r^0 | N_{<r}^0] \\
&= \sum_{r=1}^{\infty} \left(\prod_{t=1}^{r-1} \left(1 - \Pr[E_t^0 | N_{<t}^0]\right) \right) \cdot \Pr[A_r^0 | N_{<r}^0].
\end{aligned}$$

Then, by Lemma 2.1 and Lemma 2.2, we have

$$\begin{aligned}
\Pr[v_j \text{ is happy in } \mathcal{R}] &\ge \sum_{r=1}^{\infty} \left(1 - \frac{\Delta + 1 - \Delta x_j}{k}\right)^{r-1} \cdot \frac{x_j}{k} \\
&= \frac{x_j}{\Delta + 1 - \Delta x_j}.
\end{aligned}$$

□

2.2.2 Probability that a vertex in $\mathcal{B}_1$ becomes happy

For a vertex $v \in \mathcal{B}_1$, with $B(v)$ pre-colored i^*, by ALGORITHM $\mathcal{P}$, we get

$$\Pr[v \text{ is happy in } \mathcal{P}] = \frac{1}{k}. \tag{2.13}$$

By ALGORITHM $\mathcal{R}$, define the following events for $v \in \mathcal{B}_1$ similarly as before. $N^1_{<r}$: all vertices in $B(v)$ are not colored before the r-th round, N^1_r: all vertices in $B(v)$ are not colored in the r-th round, A^1_r: all vertices in $B(v)$ are colored in i^* in the r-th round, and E^1_r: there exists a vertex in $B(v)$ that is colored in the r-th round. The following Lemma 2.4 has already been proved in [77].

Lemma 2.4. *Let v_j be a vertex in $\mathcal{B}_1$, then* $\Pr[A^1_r | N^1_{<r}] \geq \frac{x_j}{k}$.

Lemma 2.5. *Let v_j be a vertex in $\mathcal{B}_1$, then* $\Pr[E^1_r | N^1_{<r}] \leq \frac{\Delta-(\Delta-1)x_j}{k}$.

Proof. Let $B^\circ(v_j)$ be the set of uncolored vertices in $B(v_j)$. Then, we have

$$\Pr[E^1_r | N^1_{<r}] = \sum_i \frac{1}{k} \max_{v_\ell \in B^\circ(v_j)} \{y^i_\ell\}. \tag{2.14}$$

Note that $1 \leq |B^\circ(v_j)| \leq \Delta$ (when $|B^\circ(v_j)| = 0$, v is already happy). By simple calculation, we have

$$\max_{v_\ell \in B^\circ(v_j)} \{y^i_\ell\} \leq \sum_{v_\ell \in B^\circ(v_j)} y^i_\ell - (|B^\circ(v_j)| - 1) \min_{v_\ell \in B^\circ(v_j)} \{y^i_\ell\},$$

which implies

$$\begin{aligned}
\sum_i \max_{v_\ell \in B^\circ(v_j)} \{y^i_\ell\} &\leq |B^\circ(v_j)| - (|B^\circ(v_j)| - 1) \sum_i \min_{v_\ell \in B^\circ(v_j)} \{y^i_\ell\} \\
&\leq |B^\circ(v_j)| - (|B^\circ(v_j)| - 1) \sum_i \min_{v_\ell \in B^\circ(v_j)} \{y^i_\ell\} \\
&= |B^\circ(v_j)| - (|B^\circ(v_j)| - 1)x_j = |B^\circ(v_j)|(1 - x_j) + x_j \\
&\leq \Delta(1 - x_j) + x_j = \Delta - (\Delta - 1)x_j,
\end{aligned} \tag{2.15}$$

where the second inequality holds since $\min_{v_\ell \in B^\circ(v_j)}\{y^i_\ell\} \geq \min_{v_\ell \in B(v_j)}\{y^i_\ell\}$.

Therefore, by (2.14) and (2.15), we have

$$\Pr[E^1_r | N^1_{<r}] \leq \frac{\Delta - (\Delta - 1)x_j}{k}.$$

□

Lemma 2.6. *Let v_j be a vertex in $\mathcal{B}_1$, then* $\Pr[v \text{ is happy in } \mathcal{R}] \geq \dfrac{x_j}{\Delta - (\Delta - 1)x_j}$.

Proof. Same as Lemma 2.3, we have

$$\begin{aligned}\Pr[v_j \text{ is happy in } \mathcal{R}] &\geq \sum_{r=1}^{\infty} \Pr[N^1_{<r}] \cdot \Pr[A^1_r | N^1_{<r}] \\ &= \sum_{r=1}^{\infty} \left(\prod_{t=1}^{r-1} \left(1 - \Pr[E^1_t | N^1_{<t}]\right) \right) \cdot \Pr[A^1_r | N^1_{<r}].\end{aligned}$$

Therefore, by Lemma 2.4 and Lemma 2.5, we get

$$\begin{aligned}\Pr[v_j \text{ is happy in } \mathcal{R}] &\geq \sum_{r=1}^{\infty} \left(1 - \frac{\Delta - (\Delta - 1)x_j}{k}\right)^{r-1} \cdot \frac{x_j}{k} \\ &= \frac{x_j}{\Delta - (\Delta - 1)x_j}.\end{aligned}$$

□

2.2.3 The approximation ratio analysis

Theorem 2.7. *The MHV problem on general graphs can be approximated within $\frac{1}{\Delta + 1/g(\Delta)}$ by* Algorithm $\mathcal{A}$*, where $g(\Delta) = (\sqrt{\Delta} + \sqrt{\Delta + 1})^2 \Delta > 41.7$. For not-too-large k ($3 \leq k \leq h(\Delta)$), the approximation ratio can be improved to $\frac{1 + (\Delta - 1)/h(\Delta)}{\Delta}$, where $h(\Delta) \geq 219.8$.*

Proof. By equalities (2.8), (2.10), and Lemma 2.3, we have

$$\begin{aligned}\mathrm{E}[A_0] &\geq (1 - \lambda) \sum_{v_j \in \mathcal{B}_0} w_j + \lambda \sum_{v_j \in \mathcal{B}_0} w_j \frac{x_j}{\Delta + 1 - \Delta x_j} \\ &= \sum_{v_j \in \mathcal{B}_0} \left(\frac{1 - \lambda}{x_j} + \frac{\lambda}{\Delta + 1 - \Delta x_j} \right) w_j x_j.\end{aligned}$$

Note that there are three items multiplied together in the above summation. We consider the first item and let $f_0(x) = \frac{1-\lambda}{x} + \frac{\lambda}{\Delta + 1 - \Delta x}$. Since $f_0(x)$ is a convex function of $x \in [0, 1]$, solving

$f_0'(x) = 0$, we get $x_0 = \frac{(\Delta+1)\sqrt{1-\lambda}}{\sqrt{\Delta\lambda}+\Delta\sqrt{1-\lambda}}$. We must guarantee that $x_0 \le 1$, which is equivalent to $\lambda \ge \frac{1}{\Delta+1}$, so we get that

$$f_0(x) \ge \begin{cases} f_0(x_0) = \frac{\Delta(1-\lambda)+\lambda+2\sqrt{\Delta\lambda(1-\lambda)}}{\Delta+1}, & \text{when } \frac{1}{\Delta+1} \le \lambda \le 1, \\ f_0(1) = 1, & \text{when } 0 \le \lambda \le \frac{1}{\Delta+1}. \end{cases}$$

By equalities (2.8), (2.13), and Lemma 2.6, we have

$$\begin{aligned} \mathrm{E}[A_1] &\ge (1-\lambda)\sum_{v_j\in\mathcal{B}_1} w_j\frac{1}{k} + \lambda\sum_{v_j\in\mathcal{B}_1} w_j\frac{x_j}{\Delta-(\Delta-1)x_j} \\ &= \sum_{v_j\in\mathcal{B}_1}\left(\frac{1-\lambda}{kx_j}+\frac{\lambda}{\Delta-(\Delta-1)x_j}\right)w_jx_j. \end{aligned}$$

Let $f_1(x) = \frac{1-\lambda}{kx} + \frac{\lambda}{\Delta-(\Delta-1)x}$. Since $f_1(x)$ is a convex function of $x \in [0,1]$, solving $f_1'(x) = 0$, we get $x_1 = \frac{\Delta\sqrt{1-\lambda}}{\sqrt{k(\Delta-1)\lambda}+(\Delta-1)\sqrt{1-\lambda}}$. As before, we must guarantee that $x_1 \le 1$. Therefore, we finally get that

$$f_1(x) \ge \begin{cases} f_1(x_1) = \frac{(\Delta-1)(1-\lambda)+k\lambda+2\sqrt{k(\Delta-1)\lambda(1-\lambda)}}{k\Delta}, & \text{when } \frac{1}{k(\Delta-1)+1} \le \lambda \le 1, \\ f_1(1) = \frac{(k-1)\lambda+1}{k}, & \text{when } 0 \le \lambda \le \frac{1}{k(\Delta-1)+1}. \end{cases}$$

Let OPT be the optimal total weight of happy vertices, then by (2.9), we have

$$\mathrm{E}[\mathrm{SOL}] = \mathrm{E}[A_0] + \mathrm{E}[A_1] \ge \min\{f_0(x), f_1(x)\}\cdot \mathrm{OPT}.$$

Denote

$$a_0(\lambda) = f_0(x_0) = \frac{\Delta(1-\lambda)+\lambda+2\sqrt{\Delta\lambda(1-\lambda)}}{\Delta+1}$$

and

$$a_1(\lambda) = f_1(x_1) = \frac{(\Delta-1)(1-\lambda)+k\lambda+2\sqrt{k(\Delta-1)\lambda(1-\lambda)}}{k\Delta}$$

Then the expected approximation ratio of Algorithm $\mathcal{A}$ is $\alpha(\lambda) = \min\{f_0(x), f_1(x)\}$:

$$\alpha(\lambda) = \begin{cases} \min\left\{1, \frac{(k-1)\lambda+1}{k}\right\} = \frac{(k-1)\lambda+1}{k}, & \text{when } 0 \le \lambda \le \frac{1}{k(\Delta-1)+1}, \\ \min\{1, a_1(\lambda)\} = a_1(\lambda), & \text{when } \frac{1}{k(\Delta-1)+1} \le \lambda \le \frac{1}{\Delta+1}, \\ \min\{a_0(\lambda), a_1(\lambda)\}, & \text{when } \frac{1}{\Delta+1} \le \lambda \le 1. \end{cases}$$

To obtain the best value for α through setting the probability λ, we consider the following three cases.

Case 1. $0 \le \lambda \le \frac{1}{k(\Delta-1)+1}$.

In this case, $\alpha(\lambda) = \frac{(k-1)\lambda+1}{k}$ is monotonically increasing in λ, and we have the maximum value

$$\alpha(\lambda_0) = \frac{\Delta}{k(\Delta-1)+1} \text{ at } \lambda_0 = \frac{1}{k(\Delta-1)+1}.$$

Case 2. $\frac{1}{k(\Delta-1)+1} \le \lambda \le \frac{1}{\Delta+1}$.

In this case, $a_1(\lambda)$ is monotonically increasing in λ, and we have the maximum value

$$\alpha(\lambda_1) = a_1(\lambda_1) \text{ at } \lambda_1 = \frac{1}{\Delta+1}.$$

Case 3. $\frac{1}{\Delta+1} \le \lambda \le 1$.

In this case, both $a_0(\lambda)$ and $a_1(\lambda)$ are concave functions of λ, and we have the maximum value for $a_0(\lambda)$ at λ_1 and the maximum value for $a_1(\lambda)$ at $\lambda_2 = \frac{k}{\Delta+k-1}$, with $\lambda_2 > \lambda_1$. We want to determine which one of $a_0(\lambda_2)$ and $a_1(\lambda_2)$ is larger by solving $a_0(\lambda) = a_1(\lambda)$, and we obtain

$$\lambda^* = \frac{K^2}{K^2+k},$$

where $K = (\sqrt{\Delta} + \sqrt{\Delta+1})(\Delta\sqrt{k} - \sqrt{\Delta^2-1})$.

That is, $a_0(\lambda_2) \ge a_1(\lambda_2)$ if and only if $\lambda_2 \le \lambda^*$, which is determined by the detailed values of Δ and k. Furthermore, by solving $\lambda^* = \lambda_2$ we represent k as a function of Δ, $h(\Delta)$, and we find that $h(\Delta)$ is a monotonically increasing function of Δ, with $h(\Delta) \ge h(3) = 219.8$.

To conclude, $a_0(\lambda_2) \ge a_1(\lambda_2)$ if and only if $\lambda_2 \le \lambda^*$, which holds if and only if $k \le h(\Delta)$. And if so, we have $\max_{\frac{1}{\Delta+1} \le \lambda \le 1} \min\{a_0(\lambda), a_1(\lambda)\} = a_1(\lambda_2)$ (see

Figure 2.5 for an illustration), and the maximum value of $\alpha(\lambda)$ we can obtain is

$$\alpha(\lambda_2) = a_1(\lambda_2) = \frac{\Delta + k - 1}{\Delta k};$$

otherwise, we have $\max_{\frac{1}{\Delta+1} \le \lambda \le 1} \min\{a_0(\lambda), a_1(\lambda)\} = a_0(\lambda^*) = a_1(\lambda^*)$ (see Figure 2.6 for an illustration), and the maximum value of $\alpha(\lambda)$ we can obtain is

$$\alpha(\lambda^*) = a_0(\lambda^*) = a_1(\lambda^*) = \frac{\left(K + \sqrt{\Delta k}\right)^2}{(\Delta + 1)(K^2 + k)}.$$

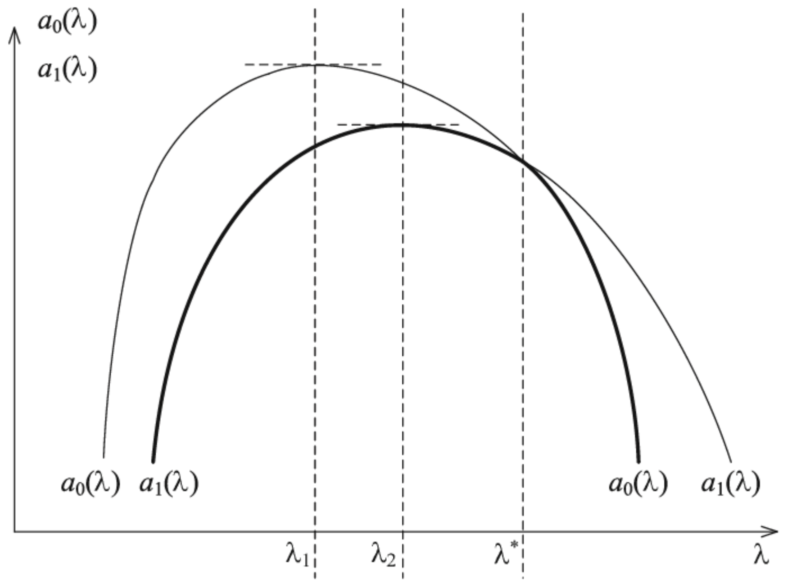


FIGURE 2.5: An illustration of $a_0(\lambda)$ and $a_1(\lambda)$ with $\lambda_2 \le \lambda^*$. The thick line denotes the function $\min\{a_0(\lambda), a_1(\lambda)\}$.

Recall that our goal is to select a probability λ to achieve the maximum value for $\alpha(\lambda)$, which is one of the three $\alpha(\lambda_0)$, $\alpha(\lambda_2)$, and $\alpha(\lambda^*)$ (since $\alpha(\lambda_1) < \alpha(\lambda_0)$). Based on the above argument, when $3 \le k \le h(\Delta)$ (that is, k is *not-too-large*), the approximation ratio we can achieve is $\max\{\alpha(\lambda_0), \alpha(\lambda_1), \alpha(\lambda_2)\} = \alpha(\lambda_2)$. So, we should set $\lambda = \lambda_2$ and the achieved approximation ratio is

$$\alpha(\lambda_2) = a_1(\lambda_2) = \frac{\Delta + k - 1}{\Delta k} \ge \frac{1 + (\Delta - 1)/h(\Delta)}{\Delta} > \frac{1}{\Delta}.$$

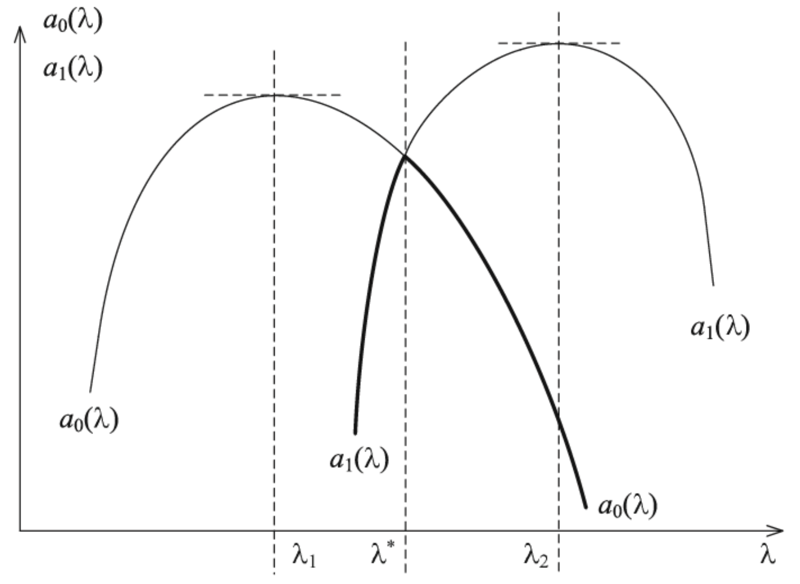


FIGURE 2.6: An illustration of $a_0(\lambda)$ and $a_1(\lambda)$ with $\lambda^* < \lambda_2$. The thick line denotes the function $\min\{a_0(\lambda), a_1(\lambda)\}$.

When $k \geq h(\Delta)$, the approximation ratio we can achieve is $\max\{\alpha(\lambda_0), \alpha(\lambda_1), \alpha(\lambda^*)\} = \alpha(\lambda^*)$. So, we should set $\lambda = \lambda^*$ and the achieved approximation ratio is

$$\alpha(\lambda^*) = a_0(\lambda^*) = a_1(\lambda^*) = \frac{\left(K + \sqrt{\Delta k}\right)^2}{(\Delta + 1)(K^2 + k)} = \frac{1}{\Delta + 1/g(\Delta)} > \frac{1}{\Delta + 1},$$

where $g(\Delta) = (\sqrt{\Delta} + \sqrt{\Delta + 1})^2 \Delta \geq g(3) > 41.7$.

Note that $\frac{1+(\Delta-1)/h(\Delta)}{\Delta} > \frac{1}{\Delta+1/g(\Delta)}$. Therefore, ALGORITHM $\mathcal{A}$ is a $\frac{1}{\Delta+1/g(\Delta)}$-approximation algorithm for the MHV problem, where k is the number of colors, Δ is the maximum vertex degree, and $g(\Delta) = (\sqrt{\Delta} + \sqrt{\Delta + 1})^2 \Delta$. For not-too-large k, the approximation ratio can be improved to $\frac{1+(\Delta-1)/h(\Delta)}{\Delta}$, which is greater than $\frac{1}{\Delta}$.

□

ALGORITHM $\mathcal{P}$ can be derandomized by trying every color, which is exactly the greedy $1/k$-approximation algorithm [78], and ALGORITHM $\mathcal{R}$ can also be derandomized in polynomial time [77] by trying all possible colors and all possible ρ (each value of y^i_j would be a possible selection for ρ and the number of different y^i_j values is in $O(kn)$). Therefore, the results in Theorem 2.7 is actually a deterministic result.

We remark that the $1/k$-approximation algorithm is also an algorithm that can round any optimal solution of (LP-MHV), with an objective value of OPT_f, to an integer solution with an objective value of $\mathrm{SOL} \geq \mathrm{OPT}_f/k$, making $1/k$ a lower bound on the integrality gap of (LP-MHV). Therefore, the approximation ratio $1/(\Delta + 1/g(\Delta))$ we obtained is also a lower bound on the integrality gap of (LP-MHV).

2.3 The Sub-ML and the Sup-ML problems

Given a ground set $V = \{v_1, v_2, \ldots, v_n\}$, $y_j := y(v_j)$ is a real variable that maps the element v_j to the closed unit interval $[0,1]$. For any non-negative set function $f : 2^V \to \mathbb{R}_+$, its Lovász extension [63, 74] is a function $\hat{f} : [0,1]^V \to \mathbb{R}_+$ such that

$$\hat{f}(\mathbf{y}) = \sum_{j=1}^{n-1} (y_{\pi_j} - y_{\pi_{j+1}}) f(\{v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_j}\}), \tag{2.16}$$

where $\mathbf{y} = (y_1, y_2, \ldots, y_n) \in [0,1]^V$ and π is a permutation on $\{1, 2, \ldots, n\}$ such that $1 = y_{\pi_1} \geq y_{\pi_2} \geq \ldots \geq y_{\pi_n} = 0$. It has been proved by Lovász [63] that a set function is submodular (supermodular, respectively) if and only if its Lovász extension is convex (concave, respectively).

In the context of the Sub-ML problem with $f(\cdot)$ being the non-negative submodular set function and $T_i \subset V$ being the non-empty subset of elements pre-labeled i, $i \in L = \{1, 2, \ldots, k\}$, we define a binary variable $y^i_j := y^i(v_j)$ for each pair of an element v_j and a label i, such that $y^i_j = 1$ if and only if the element v_j is labeled i. Next, y^i_j is relaxed to be a real variable in the closed unit interval $[0,1]$. For each i, let $\mathbf{y}_i = (y^i_1, y^i_2, \ldots, y^i_n) \in [0,1]^V$; let $\hat{f} : [0,1]^V \to \mathbb{R}_+$ is the Lovász extension of $f(\cdot)$ as defined in Eq. (2.16). A relaxation based on the Lovász extension for Sub-ML can be written as follows:

$$\text{minimize} \quad \sum_{i=1}^{k} \hat{f}(\mathbf{y}_i) \tag{CP-Sub-ML}$$

$$\text{subject to} \quad \sum_{i=1}^{k} y^i_j = 1, \quad \forall v_j \in V \tag{2.17}$$

$$y^i_j = 1, \quad \forall v_j \in T_i,\ i \in L \tag{2.18}$$

$$y^i_j \geq 0, \quad \forall v_j \in V,\ i \in L \tag{2.19}$$

The submodularity of the function $f(\cdot)$ implies that (CP-Sub-ML) is a *convex program*. Chekuri and Ene [20] have shown that this relaxation (CP-Sub-ML) can be solved exactly in polynomial time.

In fact, such a relaxation based on the Lovász extension was proposed by Chekuri and Ene [18] for the SUB-MP problem, which is a special case of the SUB-ML problem in which $|T_i| = 1$ for every label i. We extend this relaxation for the SUB-ML problem with only one change that in the set of constraints (2.17) $y_j^i = 1$ holds for multiple elements v_j. We remark that one cannot reduce the SUB-ML problem to SUB-MP by cruelly contracting all the elements pre-labeled with the same label into a single element, which suggests incorrectly that all these pre-labeled elements were identical.

The following approximation algorithm $\mathcal{RR}$ first solves the convex program (CP-Sub-ML), followed by a randomized rounding scheme, which was applied to solve the SUB-MP problem in [37], to obtain a feasible solution to the SUB-ML problem. Ene *et al.* [37] showed that $\mathcal{RR}$ is a $(2 - 2/k)$-approximation for SUB-MP. The algorithm uses a uniformly random variable θ in the interval $(\frac{1}{2}, 1]$, and defines the following $k + 3$ sets:

$$\begin{aligned} S_i(\theta) &= \{v_j \mid y_j^i > \theta\}, \text{ for each } i \in L, \\ S(\theta) &= \textstyle\bigcup_{i=1}^{k} S_i(\theta), \\ R(\theta) &= V - S(\theta), \\ Q(\theta) &= R(1-\theta). \end{aligned} \tag{2.20}$$

```
ALGORITHM $\mathcal{RR}$
1: Solve (CP-Sub-ML) to obtain an optimal fractional solution $\{y_j^i \mid v_j \in V, i \in L\}$.
2: Pick a parameter $\theta \in (\frac{1}{2}, 1]$ uniformly at random.
3: Assign all elements of $S_i(\theta)$ the label $i$, for each $i \in L$.
4: Pick a label $i'$ from $L$ uniformly at random, assign all elements of $R(\theta)$ the label $i'$.
```

FIGURE 2.7: A high-level description of ALGORITHM $\mathcal{P}$ for the MHV problem.

The performance analysis for the algorithm $\mathcal{RR}$ on the SUB-MP problem presented in [37] does not need the fact that $|T_i| = 1$ for every label i. Therefore, the same analysis can also prove the following theorem.

Theorem 2.8. *Algorithm $\mathcal{RR}$ is a $(2 - 2/k)$-approximation for the SUB-ML problem.*

For the SUP-ML problem with $f(\cdot)$ being a non-negative supmodular set function, a relaxation based on the Lovász extension can be written just by replacing the objective "minimize" in (CP-Sub-ML) to “maximize”, and we denote this relaxation as (CP-Sup-ML). (CP-Sup-ML) is a *concave program* which can also be solved in polynomial time similarly to the methods of solving (CP-Sub-ML). Using an analogous argument as the proof of Theorem 2.8, we can have the following corollary on the SUP-ML problem (we omit the proofs here because the only differences in proving this corollary and Theorem 2.8 lie on a couple of inequalities of which the sign of less than or equal to should be greater than or equal to instead).

Corollary 2.9. *Algorithm $\mathcal{RR}$ is a $2/k$-approximation for SUP-ML.*

2.4 The approximability of the MUHV problem

Recall that the MUHV problem can be cast as finding a partition $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ of the vertex set V such that $f_b(\mathcal{S}) = \sum_{i=1}^{k} f_b(S_i)$ is minimized, where the set function $f_b(\cdot)$ is defined in Eq. (2.1) and S_i is the subset of vertices colored i, for each i.

First, we prove the following two lemmas.

Lemma 2.10. *Given a graph $G = (V, E)$, the boundary $\partial(\cdot)$ of a vertex subset has the following properties for any two subsets $X, Y \subseteq V$:*

(i) $\partial(\emptyset) = \emptyset$;

(ii) $\partial(X \cap Y) \subseteq \partial(X) \cup \partial(Y)$;

(iii) $\partial(X \cup Y) \subseteq \partial(X) \cup \partial(Y)$; *and*

(iv) $\partial(X \cap Y) \cap \partial(X \cup Y) \subseteq \partial(X) \cap \partial(Y)$.

Proof. Recall that for any $X \subseteq V$, $\partial(X)$ is the subset of vertices of X each having at least one neighbor outside of X. It follows that $\partial(\emptyset) = \emptyset$.

Next, for any $v \in \partial(X \cap Y)$, $v \in X \cap Y$ and v has a neighbor $u \notin X \cap Y$. That is, u is either outside of X or outside of Y. If u is outside of X, then $v \in \partial(X)$; otherwise, $v \in \partial(Y)$. Therefore, $\partial(X \cap Y) \subseteq \partial(X) \cup \partial(Y)$.

For any $v \in \partial(X \cup Y)$, $v \in X \cup Y$ and v has a neighbor $u \notin X \cup Y$. If $v \in X$, then $v \in \partial(X)$; otherwise, $v \in \partial(Y)$. Therefore, $\partial(X \cup Y) \subseteq \partial(X) \cup \partial(Y)$.

Lastly, from the last paragraph, if $v \in \partial(X \cap Y) \cap \partial(X \cup Y)$, then $v \in X \cap Y$ and v has a neighbor $u \notin X \cup Y$. These imply that $v \in \partial(X)$ and $v \in \partial(Y)$, *i.e.*, $v \in \partial(X) \cap \partial(Y)$. Therefore, $\partial(X \cap Y) \cap \partial(X \cup Y) \subseteq \partial(X) \cap \partial(Y)$. □

Lemma 2.11. *Given a graph $G = (V, E)$, the set function $f_b(X) := w(\partial(X))$ defined in Eq. (2.1) satisfies $f_b(X) + f_b(Y) \geq f_b(X \cap Y) + f_b(X \cup Y)$, for any two subsets $X, Y \subseteq V$.*

Proof. According to Lemma 2.10, the boundary $\partial(\cdot)$ satisfies

(ii) $\partial(X \cap Y) \subseteq \partial(X) \cup \partial(Y)$ and

(iii) $\partial(X \cup Y) \subseteq \partial(X) \cup \partial(Y)$.

Therefore, $\partial(X \cap Y) \cup \partial(X \cup Y) \subseteq \partial(X) \cup \partial(Y)$ also holds. Furthermore, the boundary $\partial(\cdot)$ also satisfies

(iv) $\partial(X \cap Y) \cap \partial(X \cup Y) \subseteq \partial(X) \cap \partial(Y)$.

We thus conclude that

$$w(\partial(X \cap Y) \cup \partial(X \cup Y)) + w(\partial(X \cap Y) \cap \partial(X \cup Y)) \leq w(\partial(X) \cup \partial(Y)) + w(\partial(X) \cap \partial(Y)),$$

which is exactly

$$f_b(X) + f_b(Y) \geq f_b(X \cap Y) + f_b(X \cup Y).$$

□

Lemma 2.11 implies the submodularity of the function $f_b(\cdot)$ defined in Eq. (2.1), thus we have:

Lemma 2.12. *The set function $f_b(\cdot)$ defined in Eq. (2.1) is submodular.*

Therefore, the MUHV problem is a special case of the Sub-ML problem, and the following theorem immediately follows according to Theorem 2.8.

Theorem 2.13. *There exists a $(2 - 2/k)$-approximation algorithm for the MUHV problem.*

On the other hand, we can prove that the general MUHV problem and the restricted-MUHV problem are approximation-equivalent by showing the following Lemma 2.14. Then, Theorem 2.13 also follows the fact that the restricted-MUHV problem can be cast as a special case of the Sub-MP problem, which admits a $(2 - 2/k)$-approximation [37].

Lemma 2.14. *If the restricted-MUHV problem admits a ρ-approximation algorithm, then the general MUHV problem also admits a ρ-approximation algorithm.*

Proof. We prove this lemma by constructing a polynomial time reduction from the general MUHV problem to the restricted-MUHV problem.

Given an instance $\mathcal{I} = (G = (V, E), w(\cdot), C = \{1, 2, \ldots, k\}, c)$ of the general MUHV problem, we construct an instance of $\mathcal{I}' = (G' = (V', E'), w'(\cdot), C = \{1, 2, \ldots, k\}, c')$ of restricted-MUHV as follows:

- for each color $i \in C$, we create a vertex t_i and connect t_i to all the vertices $v \in V$ with $c(v) = i$;
- let $V' = V \cup T$, where $T = \{t_1, t_2, \ldots, t_k\}$, and $E' = E \cup \bigcup_{i=1}^{k} \{(t_i, v) | c(v) = i\}$;
- for each $v \in V$, let $w'(v) = w(v)$; for each $t_i \in T$, let $w(t_i) = W = \rho \cdot w(V) + 1$;
- let $C = \{1, 2, \ldots, k\}$ still be the color set, and the partial coloring function c' only pre-colors the vertices in T with $c'(t_i) = i$, for $i \in C$.

Let OPT($\mathcal{I}$) be the total weight of the optimal set of unhappy vertices in G; let OPT($\mathcal{I}'$) be the total weight of the optimal set of unhappy vertices in G'.

For any coloring function c^* that completes the given partial coloring function c for G, we can apply the same function c^* to color all the uncolored vertices in G'. Then for each $t_i \in T$, $c^*(v) = i$ for any $v \in N(t_i)$, so t_i must be happy. Thus, for any vertex in G, its happiness must be identical to the corresponding vertex in G', and they share the same weight. Therefore, under this coloring scheme, the set of unhappy vertices in G' has the same weight as the set of unhappy vertices in G. This also indicates that $\text{OPT}(\mathcal{I}') \leq \text{OPT}(\mathcal{I}) \leq w(V)$.

If the restricted-MUHV admits a ρ-approximation algorithm, then we can always find in polynomial time a coloring function c'^* that colors all the uncolored vertices in G', which makes $R \subseteq V'$ the set of unhappy vertices in G' and $w(R) \leq \rho \cdot \text{OPT}(\mathcal{I}') \leq \rho \cdot w(V)$. Under this coloring scheme, we must have $t_i \notin R$ for every $t_i \in T$, that is, vertices in T must be all happy. Assume for the sake of contradiction that t_i is unhappy for some $t_i \in T$, then we have $w(R) \geq W = \rho \cdot w(V) + 1 > \rho \cdot w(V)$, a contradiction. Then, by applying the same function c'^* to color all the corresponding uncolored vertices in G, every vertex in G has the same happiness as the corresponding vertex in G', and they share the same weight. Therefore, under this coloring scheme, the corresponding set R in G is also the set of unhappy vertices in G.

In summary, the general MUHV problem is polynomial-time reducible to the restricted-MUHV problem, and if there exists a ρ-approximation for the restricted-MUHV problem, then the general MUHV problem also admits a ρ-approximation algorithm. $\square$

Next, we prove that the restricted-MUHV problem and the HYP-MC problem are approximation-equivalent, thus MUHV and HYP-MC are also approximation-equivalent.

Lemma 2.15. *There is an approximation preserving reduction from the restricted-MUHV problem to the HYP-MC problem.*

Proof. Given an instance $\mathcal{I} = (G = (V, E), w(\cdot), C = \{1, 2, \ldots, k\}, c)$ of the restricted-MUHV problem, we construct an instance $\mathcal{I}' = (H = (V, E_H), w'(\cdot), T = \{t_1, t_2, \ldots, t_k\})$ of the HYP-MC problem as follows:

- let the vertex set be V; for each $i \in C$, let v_i which is pre-colored i be a terminal t_i; let $T = \{t_1, t_2, \ldots, t_k\}$ be the terminal set;
- for each $v \in V$, we create a hyperedge $e_v = N[v]$ and add it to the hyperedge set E_H, where $N[v] = \{v\} \cup N(v)$ is the set of all the neighbors of v in G along with v itself;
- for each hyperedge $e_v \in E_H$, let $w'(e_v) = w(v)$.

Each vertex in G corresponds one-to-one to a hyperedge in H and shares the same weight.

Consider a simple path P connecting two terminals t_i and t_j in the constructed hypergraph H. Every two consecutive vertices on P must belong to a common hyperedge, thus the path P corresponds one-to-one to a simple path connecting the two vertices t_i and t_j in G, which we also denote as P without any ambiguity. For any coloring function c^* that completes the given partial coloring function c for G, we have $c^*(t_i) = i$ for each $i = \{1, 2, \ldots, k\}$. It follows that any simple path P connecting t_i and t_j must contain at least one vertex $v \in V$ such that its preceding vertex or its succeeding vertex is colored differently from v itself. The vertex v is thus unhappy under the coloring scheme c^*. Then in the hypergraph H, removing the corresponding hyperedge e_v breaks the path P, thus disconnecting t_i and t_j via the path P. Therefore, removing all the hyperedges whose corresponding vertices in the graph G are unhappy disconnects all pairs of terminals in H, and the total weight of the removed hyperedges is equivalent to the total weight of the unhappy vertices in G.

Conversely, consider a subset E_H^* of hyperedges in the hypergraph $H = (V_H, E_H)$ whose removal disconnects all pairs of terminals. Let V^i and E_H^i denote the subsets of vertices and hyperedges in the connected component of the remainder hypergraph $(V, E_H - E_H^*)$ that contains the terminal t_i, for each $i = 1, 2, \ldots, k$. We complete the partial coloring function c by coloring all vertices of the corresponding vertex set V^i in G with the color i, for $i = 1, 2, \ldots, k$, and coloring all the other remaining vertices of V with the color 1. Clearly, all the vertices corresponding to the hyperedges of $E_H - E_H^*$ are happy. Thus, the total weight of unhappy vertices under this coloring scheme is no more than $w(E_H^*) := \sum_{e \in E_H^*} w(e)$.

In summary, the restricted-MUHV problem is polynomial-time reducible to the Hyp-MC problem, and our reduction preserves the value of any feasible solution and consequently preserves the approximation ratio. □

We note that due to the $(2 - \frac{2}{k})$-approximation for the Hyp-MC problem [44, 65], Theorem 2.13 can also be proved according to Lemma 2.15.

Lemma 2.16. *There is an approximation preserving reduction from the Hyp-MC problem to the restricted-MUHV problem.*

Proof. Given an instance $\mathcal{I} = (H = (V_H, E_H), w(\cdot), T = \{t_1, t_2, \ldots, t_k\})$ of the Hyp-MC problem, we construct an instance $\mathcal{I}' = (G = (V, E), w'(\cdot), C = \{1, 2, \ldots, k\}, c)$ of the restricted-MUHV problem as follows:

- for each hyperedge $e \in E_H$, we create a vertex v_e; let the vertex set be $V = V_H \cup V_E$, where $V_E = \{v_e \mid e \in E_H\}$; call $T = \{t_1, t_2, \ldots, t_k\} \subseteq V$ the terminal set;
- for each vertex $v \in V_H$, let $w'(v) = 0$; for each vertex $v_e \in V_E$, let $w'(v_e) = w(e)$;
- for each vertex $v_e \in V_E$, it is adjacent to every vertex of e; let the edge set be $E = \{\{v_e, v\} \mid e \in E_H, v \in e\}$;
- let the color set be $C = \{1, 2, \ldots, k\}$ and let the partial coloring function $c : V \mapsto C$ pre-color the terminal t_i with i, for each $i \in C$.

We note that the graph G is actually bipartite, and the two parts of vertices are V_H and V_E.

Consider a simple path P connecting two terminals t_i and t_j in the hypergraph H. Every two consecutive vertices on P must belong to a common hyperedge, thus the path P corresponds one-to-one to a simple path connecting the two vertices t_i and t_j in G, which we also denote as P without any ambiguity. For any coloring function c^* that completes the given partial coloring function c for G, we have $c^*(t_i) = i$ for each $i = \{1, 2, \ldots, k\}$. It follows that any simple path P connecting t_i and t_j must contain at least one vertex $v_e \in V_E$ such that its preceding vertex and its succeeding vertex, both in V_H, are colored differently. The vertex v_e is thus unhappy under the coloring scheme c^*. Then in the hypergraph H, removing the corresponding hyperedge e breaks the path P, thus disconnecting t_i and t_j via the path P. Therefore, removing all the hyperedges whose corresponding vertices in the graph G are unhappy disconnects all pairs of terminals, and the total weight of the removed hyperedges is equivalent to the total weight of the unhappy vertices in G.

Conversely, consider a subset E_H^* of hyperedges in the hypergraph $H = (V_H, E_H)$ whose removal disconnects all pairs of terminals. Let V_H^i and E_H^i denote the subsets of vertices and hyperedges in the connected component of the remainder hypergraph $(V_H, E_H - E_H^*)$ that contains the terminal t_i, for each $i = 1, 2, \dots, k$. Denote the vertex subsets in the constructed graph $G = (V, E)$ corresponding to V_H^i and E_H^i as V_H^i and V_E^i, respectively, for $i = 1, 2, \dots, k$. We complete the partial coloring function c by coloring all vertices of $V_H^i \cup V_E^i$ with the color i, for $i = 1, 2, \dots, k$, and coloring all the other remaining vertices of V with the color 1. Clearly, all the vertices of $\{v_e \mid e \in E_H - E_H^*\}$ are happy; due to every vertex of V_H has weight 0 (such that we may ignore its happiness), we conclude that the total weight of unhappy vertices under this coloring scheme is no more than $w(E_H^*) := \sum_{e \in E_H^*} w(e)$.

In summary, the Hyp-MC problem is polynomial-time reducible to the restricted-MUHV problem, and our reduction preserves the value of any feasible solution and consequently preserves the approximation ratio. □

Ene *et al.* [37] proved that achieving a $(2 - 2/k - \epsilon)$-approximation for Hyp-MC is **NP**-hard, for any $\epsilon > 0$, assuming the Unique Games Conjecture. According to Lemma 2.16, we have Theorem 2.17; due to MUHV being a special case of Sub-ML, Corollary 2.18 immediately follows.

Theorem 2.17. *No $(2 - 2/k - \epsilon)$-approximation algorithm for the restricted-MUHV or the general MUHV problem exists, for any $\epsilon > 0$, assuming the Unique Games Conjecture.*

Corollary 2.18. *No $(2 - 2/k - \epsilon)$-approximation algorithm for the Sub-ML problem exists, for any $\epsilon > 0$, assuming the Unique Games Conjecture.*

2.5 Improved approximation results for the MHV problem

2.5.1 A $2/k$-approximation for MHV

Recall that the MHV problem can be cast as finding a partition $\mathcal{S} = \{S_1, S_2, \dots, S_k\}$ of the vertex set V such that $f_p(\mathcal{S}) = \sum_{i=1}^{k} f_p(S_i)$ is maximized, where the set function $f_p(\cdot)$ is defined in Eq. (2.2) and S_i is the subset of vertices colored i, for each i. The following lemma can be proved analogously to Lemma 2.12.

Lemma 2.19. *The set function $f_p(\cdot)$ defined in Eq. (2.2) is supermodular.*

Therefore, the MHV problem is a special case of the Sup-ML problem, and the following theorem immediately follows according to Corollary 2.9.

Theorem 2.20. *Algorithm $\mathcal{RR}$ is a $2/k$-approximation for the MHV problem, which is a special case of the Sup-ML problem.*

Recall the LP relaxation (LP-MHV) for the MHV problem on a given graph $G = (V, E)$. For each color i, since there is at least one vertex pre-colored i and at least one vertex pre-colored another color (due to $k \geq 2$), we let $\mathbf{y}_i = (y^i_1, y^i_2, \ldots, y^i_n)$ and π be the permutation for $\mathbf{y}_i$ such that $1 = y^i_{\pi_1} \geq y^i_{\pi_2} \geq \ldots \geq y^i_{\pi_n} = 0$. In the concave relaxation (CP-Sup-ML) based on the Lovász extension for Sup-ML, when we set the supermodular set function f_p as in Eq. (2.2), the objective function of (CP-Sup-ML) becomes

$$\begin{aligned}\sum_{i=1}^{k} \hat{f}_p(\mathbf{y}_i) &= \sum_{i=1}^{k}\sum_{j=1}^{n-1}\left(y^i_{\pi_j} - y^i_{\pi_{j+1}}\right) f_p(\{v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_j}\}) \\ &= \sum_{i=1}^{k}\sum_{j=1}^{n-1}\left(y^i_{\pi_j} - y^i_{\pi_{j+1}}\right) \sum_{v_\ell \in \iota(\{v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_j}\})} w_\ell. \end{aligned} \tag{2.21}$$

For each vertex $v_p \in V$, let v_q denote its neighbor that appears the last in the permutation $(v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_n})$. Assume $p = \pi_{j_1}$ and $q = \pi_{j_2}$. Clearly, $v_p \in \iota(\{v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_j}\})$ if and only if $p, q \in \{\pi_1, \pi_2, \ldots, \pi_j\}$, that is, we must have $j_1, j_2 \leq j$. It follows that for the vertex $v_p \in V$, the coefficient of w_p in Eq. (2.21) is

$$\sum_{i=1}^{k}\sum_{j=\max\{j_1,j_2\}}^{n}\left(y^i_{\pi_j} - y^i_{\pi_{j+1}}\right) = \sum_{i=1}^{k} x^i_p = x_p,$$

where the last two equalities hold due to Constraints (2.5, 2.6) of (LP-MHV). This shows that by setting the supermodular set function g as defined in Eq. (2.2), (CP-Sup-ML) is the same as (LP-MHV). Therefore, we have the following theorem.

Theorem 2.21. *The LP relaxation for the MHV problem* (LP-MHV) *is the same as the relaxation based on the Lovász extension for the Sup-ML problem* CP-Sub-ML, *when the MHV problem is cast into the Sup-ML problem.*

Theorem 2.22. *The integrality gap of* (LP-MHV) *has an upper bound of $\frac{2}{k}$.*

Proof. We prove this theorem by constructing an instance $\mathcal{I} = (G = (V, E), w(\cdot), C = \{1, 2, \dots, k\}, c)$ of the MHV problem.

- Let $T = \{t_1, t_2, \dots, t_k\}$ be a set of k pre-colored vertices, called *terminals*; all terminals have the same weight $w_t \geq 0$, and the terminal t_i is pre-colored i, *i.e.* $c(t_i) = i$.
- Associated with each pair of distinct terminals t_i and t_j, $i < j$, there is a vertex $b_{\{ij\}}$. Let $V_b = \{b_{\{ij\}} \mid i < j\}$, then $|V_b| = \binom{k}{2}$; all vertices of V_b have the same weight $w_b \geq 0$, and none of them is pre-colored.
- The vertex set $V = T \cup V_b$; the edge set $E = \{\{t_i, b_{\{ij\}}\}, \{t_j, b_{\{ij\}}\} \mid i < j\}$. Clearly, $|V| = k + \binom{k}{2}$ and $|E| = 2\binom{k}{2}$.

Let c^* denote a coloring function that completes the given partial coloring function c, that is, c^* assigns a color for each vertex and it assigns the color i to the terminal t_i, for each $i \in C$. Then,

- all vertices of V_b must be unhappy, since the vertex $b_{\{ij\}}$ is adjacent to two terminals t_i and t_j colored with distinct colors;
- the terminal t_i is adjacent to $k - 1$ vertices $\{b_{\{ij\}} \mid j \neq i\}$, while the vertex $b_{\{ij\}}$ is adjacent to the terminals t_i and t_j; it follows that if t_i is happy, then all vertices of $\{b_{\{ij\}} \mid j \neq i\}$ are colored i, subsequently none of the other terminals can be happy; in other words, at most one of the k terminals can be happy, regardless of what the coloring function c^* is.

Let OPT(MHV) denote the value of an optimal solution to the constructed instance $\mathcal{I}$; we obtain:

$$\text{OPT(MHV)} \leq w_t. \tag{2.22}$$

Consider the following fractional feasible solution to the instance $\mathcal{I}$ in the LP relaxation (LP-MHV),

- for each terminal $t_i \in T$, $y^i(t_i) = 1$ and $y^j(t_i) = 0$ for all $j \neq i$;
- for each vertex $b_{\{ij\}} \in V_b$, $y^i(b_{\{ij\}}) = y^j(b_{\{ij\}}) = \frac{1}{2}$ and $y^\ell(b_{\{ij\}}) = 0$ for all $\ell \neq i, j$;
- for each terminal $t_i \in T$, we set $x^i(t_i) = y^i(b_{\{ij\}}) = \frac{1}{2}$, $x^j(t_i) = 0$ for all $j \neq i$, and $x(t_i) = \sum_{\ell=1}^{k} x^\ell(t_i) = \frac{1}{2}$;
- for each vertex $b_{\{ij\}} \in V_b$, we set $x^\ell(b_{\{ij\}}) = 0$ for all $\ell \in C$, and $x(b_{\{ij\}}) = 0$.

Let OPT(LP-MHV) denote the optimum of the instance $\mathcal{I}$ in the LP relaxation (LP-MHV). It is greater than or equal to the value of the above fractional feasible solution, that is,

$$\text{OPT(LP-MHV)} \geq \frac{1}{2} k w_t. \tag{2.23}$$

Combining Eqs. (2.22) and (2.23), it gives an upper bound on the integrality gap of the LP relaxation (LP-MHV):

$$\frac{\text{OPT(MHV)}}{\text{OPT(LP-MHV)}} \leq \frac{1}{\frac{1}{2}k} = \frac{2}{k}.$$

□

Theorems 2.20 and 2.22 together imply that the $\frac{2}{k}$-approximation algorithm $\mathcal{RR}$ for the MHV problem is the best possible based on the LP relaxation (LP-MHV), and furthermore,

Corollary 2.23. *The $\frac{2}{k}$-approximation algorithm $\mathcal{RR}$ for the* Sup-ML *problem is the best possible based on the concave relaxation on the Lovász extension (CP-Sup-ML).*

2.5.2 A hardness result for MHV

In this section, we show a hardness result on approximating the MHV problem by a reduction from the *maximum independent set* (MIS) problem, in which we are given a graph $G = (V, E)$ with a non-negative weight $w(v)$ for each vertex $v \in V$, the goal is to find a maximum-weight independent set $I \subseteq V$. We also note that if there is a connected component in G which is exactly a clique, then the maximum-weight vertex in the clique is an optimal solution to the MIS problem on that connected component. Thus, we assume without loss of generality that the input graph of the MIS problem does not contain any connected component being exactly a clique.

We observe that any graph G with maximum degree of $\Delta \geq 3$ can also be viewed as a Δ-partite graph if G contains no clique of size $\Delta + 1$, by solving the classic Coloring problem, which is to color all the vertices in the given graph such that no two adjacent vertices have the same color. In other words, the Coloring problem asks to partition the vertex set into subsets of independent sets, and the number of the subsets of independent sets is then equivalent to the number of colors required to color all the vertices. Due to Brooks' theorem [11], along with a simplified proof presented by Lovász [62], one can solve the Coloring problem on G by using at most Δ colors in polynomial time.

Given an instance $\mathcal{I} = (G = (V, E), w(\cdot))$ of MIS, where G is a k-partite graph, with $k \geq 3$ and $V_1, V_2, \ldots, V_k$ being the k parts of the vertex set V, we construct an instance $\mathcal{I}' = (G' = (V', E'), w'(\cdot), C = \{1, 2, \ldots, k\}, c)$ of MHV as follows:

- for each edge $e = (u, v) \in E$, we break it into two edges (u, z_e) and (v, z_e) by creating an additional vertex z_e;
- let $V' = V \cup X$, where $X = \{z_e | e \in E\}$, and $E' = \{(u, z_e), (v, z_e) | e = (u, v) \in E\}$;
- for each vertex $v \in V$, let $w'(v) = w(v)$; for each vertex $z_e \in X$, let $w'(z_e) = 0$;
- let the color set be $C = \{1, 2, \ldots, k\}$ and let the partial coloring function $c : V' \mapsto C$ pre-color each vertex $v_i \in V_i$ with i, for $i = 1, 2, \ldots, k$.

We note that in the graph G', only the vertices in X are uncolored; all the neighbors of any vertex in V are in X; each vertex $z_e \in X$ has exactly two neighbors u and v which correspond to the two endpoints of $e = (u, v) \in E$, and $c(u) \neq c(v)$ since G is k-partite, thus z_e must be unhappy.

Consider an independent set $I \subseteq V$ of G. For any two vertices $u, v \in I$ in the graph G', they do not share any neighbor, *i.e.*, $N(u) \cap N(v) = \emptyset$. Assume for the sake of contradiction that there exists some $x \in N(u) \cap N(v)$, then $x \in X$ and $N(x) = \{u, v\}$, indicating that $(u, v) \in E$ in graph G, which contradicts to I being an independent set of G. Then in graph G', for any $v \in I$, we color every vertex in $N(v)$ with $c(v)$; for any z_e of the remaining uncolored vertices in X, with $N(z_e) = \{v_i, v_j\}$, where $c(v_i) = i$ and $c(v_j) = j$, we color z_e with any color in $C - \{i, j\}$. This is a feasible coloring scheme for G', which makes all the vertices in I happy and all the vertices in $V' - I$ unhappy in G'. Since in the constructed instance, the weights of all vertices in V are unchanged, the total weight of I is also unchanged.

Conversely, consider a feasible coloring scheme for G', which makes all the vertices in $S \subseteq V'$ happy and the remaining vertices unhappy. Then, $S \subseteq V$ and for any two vertices $u, v \in S$, either $u, v \in V_i$ for some $i \in C$ or $u \in V_i$ and $v \in V_j$ for two distinct $i, j \in C$. In both cases, we can conclude $(u, v) \notin E$ in G. The first case is straightforward, for the second case, assume for the sake of contradiction that $(u, v) \in E$ in G, then u and v cannot be both happy in G' since $c(u) \neq c(v)$ and they share a common neighbor. Therefore, S is also an independent set in G. Still, since in the constructed instance, the weights of all vertices in V are unchanged, the total weight of S is also unchanged.

Therefore, any feasible solution to the given instance $\mathcal{I}$ of MIS corresponds one-to-one to a feasible solution to the constructed instance $\mathcal{I}'$ of MHV, and the two solutions have exactly the same value.

In summary, the MIS problem is polynomial-time reducible to the MHV problem, and our reduction preserves the value of any feasible solution and consequently preserves the approximation ratio. Austrin [4] proved that MIS is Unique Games-hard to approximate within a factor of $\Omega(\log^2 \Delta/\Delta)$, where Δ is the maximum vertex degree of the input graph, thus we have Theorem 2.24. Due to MHV being a special case of Sup-ML, Corollary 2.25 immediately follows.

Theorem 2.24. *The MHV problem is Unique Games-hard to approximate within a factor of* $\Omega(\log^2 k/k)$.

Corollary 2.25. *The Sup-ML problem is Unique Games-hard to approximate within a factor of* $\Omega(\log^2 k/k)$.

2.6 Concluding remarks and possible future work

In this chapter, we investigated the MHV problem and its complement, the MUHV problem. First, we presented a non-uniform algorithm for MHV with an approximation ratio of $1/\big(\Delta + 1/g(\Delta)\big)$, where Δ is the maximum vertex degree of the input graph and $g(\Delta) = (\sqrt{\Delta} + \sqrt{\Delta + 1})^2\Delta > 4\Delta^2$. This improves the previous best approximation ratio of $\max\{1/k, 1/(\Delta + 1)\}$ [77, 78].

Next, we showed that the MHV and MUHV problems are a special case of the supermodular and submodular multi-labeling (Sup-ML and Sub-ML) problems, respectively, by re-writing the objective functions as set functions. We showed that the convex relaxation on the Lovász extension, presented by Chekuri and Ene for the submodular multi-partitioning (Sub-MP) problem [18], can be extended for the Sub-ML problem, thereby proving that the Sub-ML (Sup-ML, respectively) can be approximated within a factor of $2 - 2/k$ ($2/k$, respectively). These general results imply that the MHV and the MUHV problems can also be approximated within a factor of $2/k$ and $2 - 2/k$, respectively, using the same approximation algorithms. The $2/k$-approximation algorithm for MHV further improves the previous best approximation ratio to $\max\{2/k, 1/\big(\Delta + 1/g(\Delta)\big)\}$.

For the MHV problem, we also showed that the LP relaxation presented by Zhang *et al.* [77] is the same as the concave relaxation on the Lovász extension for the Sup-ML problem; we then prove an upper bound of $2/k$ on the integrality gap of the LP relaxation. These suggest that the $2/k$-approximation algorithm is the best possible based on the LP relaxation; thus the $2/k$-approximation algorithm is also the best possible based on the concave relaxation on the

Lovász extension for the Sup-ML problem. Further, we proved that it is Unique Games-hard to approximate the MHV problem within a factor of $\Omega(\log^2 k/k)$, by a reduction from MIS, which also gives another evidence that the general Sup-ML problem is Unique Games-hard to approximate within a factor of $\Omega(\log^2 k/k)$.

For the MUHV problem, we showed that it is approximation-equivalent to the Hyp-MC problem, thus it is Unique Games-hard to achieve a $(2-2/k-\epsilon)$-approximation for MUHV, for any $\epsilon > 0$. This hardness result also gives another evidence that it is Unique Games-hard to achieve a $(2-2/k-\epsilon)$-approximation for the general Sub-ML problem, for any $\epsilon > 0$.

In summary, for both the MUHV problem and the generalized Sub-ML problems, we have closed the gaps between the upper and lower bounds on the approximability, which are both $(2-2/k)$; for the MHV problem and the generalized Sup-ML problems, we showed a lower bound of $2/k$ and an upper bound of $\Omega(\log^2 k/k)$ on the approximability for both of them. A possible future work would be to see if there is a better approximation algorithm for the MHV problem.

Chapter 3

The Maximum Duo-preservation String Mapping Problem[1]

3.1 Introduction

The *minimum common string partition* (MCSP) problem is a well-studied problem in computer science, with applications in the fields such as text compression and bioinformatics.

In both text compression and bioinformatics, string (or sequence) comparison is a routine work. For the similarity between two strings, a commonly used measure is the *edit distance*, which is the minimum number of operations required to transform one string into the other. At the finest scale, the edit operations involve a single character of a string, including insertion, deletion, and substitution. When comparing two long strings such as the whole genomes of multiple species, long range operations become more interesting, leading to the genome rearrangement problems [22, 69]. In particular, a transportation operation is to cut out a substring and insert it back to another position in the string. The problem of partitioning one string into a minimum number of substrings such that a reshuffle of them becomes the other string is then referred to as the MCSP problem. MCSP was first formally introduced by Goldstein *et al.* [46] as follows.

Minimum Common String Partition (MCSP): Consider two length-n strings $A = (a_1, a_2, \ldots, a_n)$ and $B = (b_1, b_2, \ldots, b_n)$ over some alphabet Σ, such that B is a permutation of A. A *partition* of A, denoted as $\mathcal{P}_A$, is a multi-set of substrings whose concatenation in a certain order becomes A. The number of substrings in $\mathcal{P}_A$ is the *cardinality* of $\mathcal{P}_A$. The MCSP problem asks for a minimum cardinality partition $\mathcal{P}_A$ of A that is also a partition of B.

[1]This chapter is based on the papers [25, 70, 71]. [70] is a work with Chen, Lin, Liu, Luo, and Zhang, "A $(1.4 + \epsilon)$-approximation algorithm for the 2-max-duo problem", which was published by the conference of the 28th International Symposium on Algorithms and Computation (ISAAC 2017), and later submitted as a journal version [25] which is under review; [71] is a work with Chen, Luo, and Lin, "A local search 2.917-approximation algorithm for duo-preservation string mapping", which is available publicly at arXiv.

k**-MCSP**: The restricted version of MCSP when every letter of the alphabet Σ occurs at most k times in each of the two given strings.

The MCSP problem is **NP**-hard and **APX**-hard even when $k = 2$ [46]. Several approximation algorithms [22, 26, 28, 46, 56, 57] have been presented since 2004. The current best result is an $O(\log n \log^* n)$-approximation [28] for the general MCSP and an $O(k)$-approximation [57] for k-MCSP. On the other hand, MCSP is proved to be *fixed parameter tractable* (FPT) [15, 16, 31, 49], with respect to k and/or to the cardinality of the optimal partition, which are/is considered as fixed rather than part of the input.

The complement of MCSP, referred to as the *maximum duo-preservation string mapping* (MPSM) problem by Chen *et al.* [21] can be defined as follows, while we call this problem as Max-Duo instead (mostly because the acronym MPSM looks too similar to the other acronyms).

Maximum Duo-preservation String Mapping (Max-Duo): The input still consists of two length-n strings $A = (a_1, a_2, \ldots, a_n)$ and $B = (b_1, b_2, \ldots, b_n)$ over some alphabet Σ, such that B is a permutation of A. Given a string, an ordered pair of consecutive letters is called a *duo* [46]; a length-ℓ substring in a partition *preserves* $\ell - 1$ duos of the given string. The Max-Duo problem is to maximize the number of duos preserved in the common partition of A and B.

k**-Max-Duo**: The restricted version of Max-Duo when every letter of the alphabet Σ occurs at most k times in each of the two given strings.

The Max-Duo problem has also been proved to be FPT by Beretta *et al.* [5, 6], with respect to the maximum number of preserved duos.

We next give a graphical view on a common partition of the two given strings $A = (a_1, a_2, \ldots, a_n)$ and $B = (b_1, b_2, \ldots, b_n)$. Construct a bipartite graph $G_0 = (A, B, E_0)$, where the vertices of A (B, respectively) are $a_1, a_2, \ldots, a_n$ in order ($b_1, b_2, \ldots, b_n$ in order, respectively) and there is an edge between a_i and b_j if they are the same letter. A common partition $\mathcal{P}$ of the strings A and B one-to-one corresponds to a perfect matching M in the graph G_0 (see Figure 3.1 for an example), and the number of duos preserved by the partition is exactly the number of pairs of *parallel* edges in the matching; if both $(a_i, b_j), (a_{i+1}, b_{j+1}) \in E$, then they form a pair of parallel edges.

Along with Max-Duo, Chen *et al.* [21] introduced the *constrained maximum induced subgraph* (CMIS) problem, in which one is given an m-partite graph $G = (V_1, V_2, \ldots, V_m, E)$,

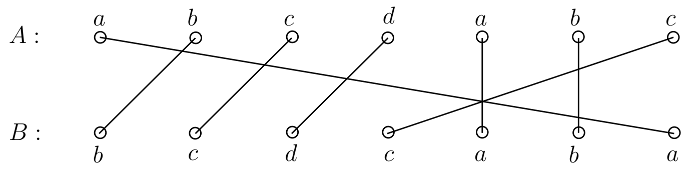


FIGURE 3.1: An instance of MAX-DUO with two strings $A = (a, b, c, d, a, b, c)$ and $B = (b, c, d, c, a, b, a)$, and a common partition $\{a, bcd, ab, c\}$ that preserves three duos (b, c), (c, d) and (a, b), corresponding to the perfect matching shown in the figure.

with each V_i having n_i^2 vertices arranged in an $n_i \times n_i$ matrix, and the goal is to select n_i vertices of each V_i in different rows and different columns such that the induced subgraph contains the maximum number of edges. The restricted version of CMIS when $n_i \leq k$ for all i is denoted as k-CMIS. For an instance of the MAX-DUO problem, one can first set m to be the number of distinct letters in the string A, set n_i to be the number of occurrences of the i-th distinct letter, and the (s, t)-vertex in the $n_i \times n_i$ matrix "means" mapping the s-th occurrence of the i-th distinct letter in the string A to its t-th occurrence in the string B; and then set an edge connecting a vertex of V_i and a vertex of V_j if the two vertices together preserve a duo. This way, the MAX-DUO problem becomes a special case of the CMIS problem, and furthermore the k-MAX-DUO is a special case of the k-CMIS. Chen *et al.* [21] presented a k^2-approximation for k-CMIS and a 2-approximation for 2-CMIS, based on linear programming and a randomized rounding. These imply that k-MAX-DUO can also be approximated within a ratio of k^2 and 2-MAX-DUO can be approximated within a ratio of 2.

Continuing on the graphical view as shown in Figure 3.1 on a common partition of the two given strings A and B, we can construct another graph $H = (V, F)$ in which every vertex of V corresponds to a pair of parallel edges in the bipartite graph $G_0 = (A, B, E_0)$, and two vertices of V are adjacent if the two pairs of parallel edges of E cannot co-exist in any perfect matching of G_0 (called *conflicting*, which can be determined in constant time, see Section 3.2). This way, a set of duos that can be preserved by some perfect matching of G_0 (called *compatible*, see Section 3.2) one-to-one corresponds to an independent set of H [10, 46]. Therefore, the MAX-DUO problem can be cast as a special case of the well-known *maximum independent set* (MIS) problem [43]; in particular, Boria *et al.* [10] showed that an instance of k-MAX-DUO translates to a graph with the maximum degree $\Delta \leq 6(k-1)$. It follows that the state-of-the-art $\big((\Delta + 3)/5 + \epsilon\big)$-approximation algorithm for MIS [7], for any $\epsilon > 0$, is a $\big((6k-3)/5 + \epsilon\big)$-approximation algorithm for k-MAX-DUO. Especially, 2-MAX-DUO and 3-MAX-DUO can be approximated within a ratio of $1.8 + \epsilon$ and $3 + \epsilon$, respectively, for any $\epsilon > 0$. Boria *et al.* [10] proved that 2-MAX-DUO is **APX**-hard, similar to 2-MCSP [46], via a linear reduction from MIS on cubic graphs, and since MIS on cubic graphs is **NP**-hard to approximate within 1.00719 [8], Boria *et al.* [10] showed that 2-Max-Duo is **NP**-hard to approximate within 1.00042. Besides, Boria *et al.* [10] claimed that 2-Max-Duo can be approximated within $1.6 + \epsilon$, for any $\epsilon > 0$. In this chapter, we also study the 2-Max-Duo problem; using the above reduction to the MIS problem, we present a vertex-degree reduction scheme and design an improved $(1.4 + \epsilon)$-approximation, for any $\epsilon > 0$.

In Section 3.2, we will construct another bipartite graph for an instance of the Max-Duo problem, and thus cast Max-Duo as a special case of the *maximum compatible bipartite matching* (MCBM) problem. Such a reduction was first shown by Boria *et al.* [10], who presented a 4-approximation for the MCBM problem, implying that Max-Duo can also be approximated within a ratio of 4. Boria *et al.* [9] also used this reduction, with the word *consecutive* in place of *compatible*, to present a local search 3.5-approximation for the MCBM problem. In the meantime, Brubach [12] presented a 3.25-approximation for the Max-Duo based on a novel combinatorial triplet matching.

The basic idea in the local search 3.5-approximation for the MCBM problem by Boria *et al.* [9] is to swap one edge of the current matching out for two compatible edges, thus to increase the size of the matching till a local optimum is reached. The performance ratio 3.5 is shown to be tight. We extend this idea to allow swapping five edges of the current matching out for six compatible edges. By observing that any matching can be partitioned into a subset of *singleton edges* and a subset of *parallel edges* (to be defined in Section 3.2), we also allow a new operation of swapping five edges of the current matching out for five compatible edges if the number of singleton edges is strictly decreased (or equivalently, the number of parallel edges is strictly increased). Through a complex yet interesting amortized analysis, we prove that our local search algorithm has an approximation ratio of at most $35/12 < 2.917$, for both the MCBM and the Max-Duo problems, which improves the previous best 3.25-approximation algorithm and breaks the barrier of 3.

We remark that immediately after the first version of our 2.917-approximation result was made publicly available on arXiv [71], Deduk *et al.* posted an article on arXiv [34], later appearing in CPM 2017 [33], in which they proposed an $n^{O(1/\epsilon)}$-time $(2 + \epsilon)$-approximation algorithm for the general Max-Duo problem, for any $\epsilon > 0$. Both results exceed the previously the best $\big((6k - 3)/5 + \epsilon\big)$-approximation algorithm for k-Max-Duo, when $k \geq 3$. The $(2 + \epsilon)$-approximation algorithm is also designed for the MCBM problem, and it is the current best approximation. In more details, given an $\epsilon > 0$, by setting $t = \lceil \frac{4}{\epsilon} \rceil + 1$, the algorithm first greedily finds a set of compatible *streaks* of size greater than or equal to t, where a streak is a maximal set of consecutive parallel edges; it then applies a local search for swapping $t-1$ edges of the current matching out for t compatible edges. One thus sees that both our 2.917-approximation and this $(2+\epsilon)$-approximation are based on two design ideas, one is to iteratively swap some edges out of the current matching for strictly more compatible edges to increase the size of the matching, and the other is to select into the matching as many consecutive parallel edges as possible. Interestingly, the performance analysis for the $(2+\epsilon)$-approximation is also done by an amortization scheme, though differently (called *credit distribution scheme* in [33]). From the approximation ratio perspective, our 2.917-approximation is superseded by the $(2+\epsilon)$-approximation; nevertheless, we believe that the design ideas in our algorithm and the amortized performance analysis can provide additional insights into the Max-Duo problem, and perhaps become helpful for further improvement.

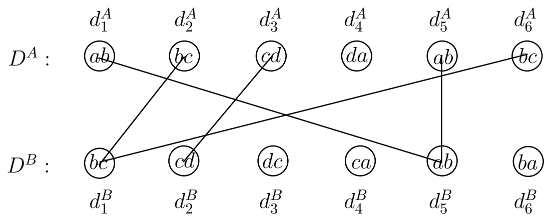
The rest of this chapter is organized as follows. In Section 3.2, We provide some preliminaries, including the formal description of the MCBM problem and the terminologies, several important structural properties of the graph constructed from the two given strings, and notations to be used throughout this chapter. In Section 3.3, we study the 2-Max-Duo problem. We present the vertex-degree reduction scheme in Section 3.3.1. The new approximation algorithm, denoted as Approx, is presented in Section 3.3.2, where we show that it is a $(1.4+\epsilon)$-approximation for 2-Max-Duo. In Section 3.3.3, we review the reduction from MIS on cubic graphs to 2-Max-Duo and make a conclusion that would be helpful for further improving the approximation result for 2-Max-Duo. In Section 3.4, we study the general Max-Duo problem. Our local search algorithm is presented in Section 3.4.1. In Section 3.4.2, we analyze the approximation ratio of our algorithm through amortization. In Section 3.4.3, we show a lower bound of $13/6 > 2.166$ on the locality gap of our algorithm for the MCBM problem, and a lower bound of $5/3 > 1.666$ on the locality gap of our algorithm for the Max-Duo problem. We conclude this chapter in Section 3.5, along with some possible future work.

3.2 Preliminaries

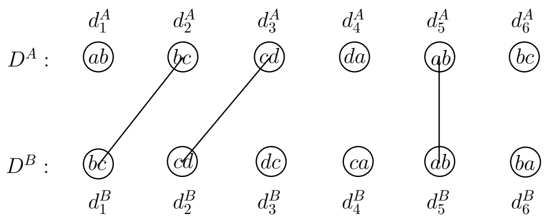
Consider an instance of the Max-Duo problem with two length-n strings $A=(a_1,a_2,\ldots,a_n)$ and $B=(b_1,b_2,\ldots,b_n)$ such that B is a permutation of A. Recall that we can view the instance as a bipartite graph $G_0=(A,B,E_0)$, where the vertices in A and B are $a_1,a_2,\ldots,a_n$ in order and $b_1,b_2,\ldots,b_n$ in order, respectively, and there is an edge between $a_i\in A$ and $b_j\in B$ if they are the same letter, denoted as $e_{i,j}$. We use $d_i^A=(a_i,a_{i+1})$ and

$d_i^B = (b_i, b_{i+1})$ to denote the i-th duo of A and B, respectively, for $i = 1, 2, \ldots, n-1$; and $D^A = \{d_1^A, d_2^A, \ldots, d_{n-1}^A\}$ and $D^B = \{d_1^B, d_2^B, \ldots, d_{n-1}^B\}$. We construct a bipartite graph $G = (D^A, D^B, E)$, where there is an edge $e_{i,j}$ connecting d_i^A and d_j^B if $a_i = b_j$ and $a_{i+1} = b_{j+1}$, suggesting that the duo d_i^A is preserved if the edge $e_{i,j} = (d_i^A, d_j^B)$ is selected into the solution matching. (See Figure 3.2a for the bipartite graph constructed from the two strings shown in Figure 3.1.) Note that selecting the edge $e_{i,j}$ rules out all the other edges incident on d_i^A and all the other edges incident on d_j^B, and some more edges described in the next paragraph.

Formally, the two edges $e_{i,j}$ and $e_{i,j'}$ with $j \neq j'$ are called *adjacent*, and they are *conflicting* since they cannot be both selected into a feasible solution matching. Similarly, two adjacent edges $e_{i,j}$ and $e_{i',j}$ with $i \neq i'$ are conflicting. The two edges $e_{i,j}$ and $e_{i+1,j+1}$ are called *parallel*; while the two edges $e_{i,j}$ and $e_{i+1,j'}$ with $j' \neq j, j+1$ are called *neighboring*. Two neighboring edges are conflicting too since they cannot be both selected. Similarly, the two edges $e_{i,j}$ and $e_{i',j+1}$ with $i' \neq i, i+1$ are neighboring and conflicting. Any two unconflicting edges are said *compatible* to each other, and a *compatible* set of edges contains edges that are pairwise compatible, which is consequently a feasible solution matching (called a *compatible matching*). (See Figure 3.2b for a compatible matching found in the bipartite graph in Figure 3.2a.) The goal of the *maximum compatible bipartite matching* (MCBM) problem is to find a maximum cardinality compatible matching in the bipartite graph $G = (D^A, D^B, E)$.


d_1^A d_2^A d_3^A d_4^A d_5^A d_6^A
D^A :
ab bc cd da ab bc
D^B :
bc cd dc ca db ba
d_1^B d_2^B d_3^B d_4^B d_5^B d_6^B

(A) The constructed bipartite graph.


d_1^A d_2^A d_3^A d_4^A d_5^A d_6^A
D^A :
ab bc cd da ab bc
D^B :
bc cd dc ca db ba
d_1^B d_2^B d_3^B d_4^B d_5^B d_6^B

(B) A compatible matching in the graph.

FIGURE 3.2: A bipartite graph $G = (D^A, D^B, E)$ constructed from the two strings $A = (a, b, c, d, a, b, c)$ and $B = (b, c, d, c, a, b, a)$, and a compatible matching in G containing three edges $e_{2,1}, e_{3,2}, e_{5,5}$.

Clearly, the bipartite graph $G = (D^A, D^B, E)$ in the MCBM problem does not have to be constructed out of two given strings in the MAX-DUO problem, and therefore MAX-DUO is a special case of MCBM. Nevertheless, when restricted to MAX-DUO, the cardinality of a compatible matching is exactly the number of duos preserved by the matching. An edge in a compatible matching M is called *singleton* if it is not parallel to any other edge in the matching. This way, the matching M is partitioned into two parts: $s(M)$ containing all the singleton edges and $p(M)$ containing all the parallel edges. A series of pairs of parallel edges $e_{i,j}, e_{i+1,j+1}, \ldots, e_{i+p,j+p}$, for some $p \geq 2$, is referred to as *consecutive parallel edges*.

Observation 3.1. *Any edge $e_{i,j} \in E$ can be conflicting with at most 6 edges that are pairwise compatible, which are $e_{i,j'}$, $e_{i-1,j''-1}$, $e_{i+1,j'''+1}$, $e_{i',j}$, $e_{i''-1,j-1}$, $e_{i'''+1,j+1}$ incident on $d^A_{i-1}, d^A_i, d^A_{i+1}, d^B_{j-1}$, d^B_j, d^B_{j+1}, respectively, where none of i', i'', i''' can be i and none of j', j'', j''' can be j.*

We remark that in Observation 3.1 by "at most", some of the six edges could be void, that is, non-existent in E; also, when $e_{i,j'}$ and $e_{i-1,j''-1}$ are both present, then they have to be parallel suggesting that $j' = j''$ (the same applies to $e_{i,j'}$ and $e_{i+1,j'''+1}$, $e_{i',j}$ and $e_{i''-1,j-1}$, $e_{i',j}$ and $e_{i'''+1,j+1}$).

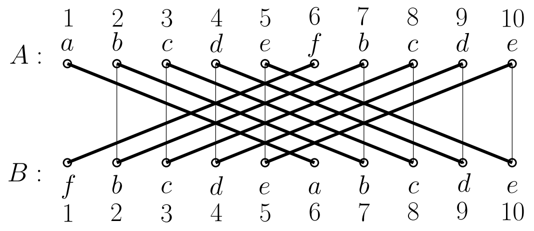
The fact that two pairs of parallel edges are conflicting if they cannot co-exist in any perfect matching of G_0 also motivates the following reduction from the k-Max-Duo problem to the MIS problem: From the bipartite graph $G_0 = (A, B, E_0)$, we construct another graph $H = (V, F)$ in which a vertex $v_{i,j}$ of V corresponds to the pair of parallel edges $(e_{i,j}, e_{i+1,j+1})$ of E_0; two vertices of V are *conflicting* if and only if the two corresponding pairs of parallel edges are conflicting, and two conflicting vertices of V are adjacent in H. One can see that a set of duos of A that can be preserved all together, a set of pairwise non-conflicting pairs of parallel edges of E_0, and an independent set in H, are equivalent to each other. See Figure 3.3b for an example of the graph $H = (V, F)$ constructed from the bipartite graph G_0 shown in Figure 3.3a. We note that $|V| \leq k(n-1)$ and thus H can be constructed in $O(k^2n^2)$ time from the instance of the k-Max-Duo problem.

In the graph H, for any $v \in V$, we use $N(v)$ to denote the set of its neighbors, that is, the vertices adjacent to v. The two ordered letters in the duo corresponding to the vertex v is referred to as the *letter content* of v. For example, in Figure 3.3b, the letter content of $v_{1,6}$ is "ab" and the letter content of $v_{6,1}$ is "fb".

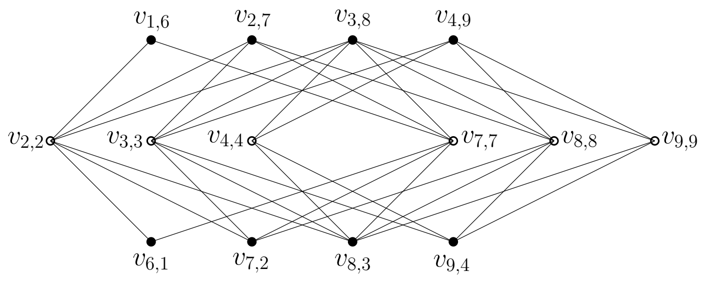
Recall from the construction that there is an edge $e_{i,j}$ in the graph $G_0 = (A, B, E_0)$ if $a_i = b_j$, and there is a vertex $v_{i,j}$ in the graph $H = (V, F)$ if the parallel edges $e_{i,j}$ and $e_{i+1,j+1}$ are in $G_0 = (A, B, E_0)$.

Lemma 3.1. *The graph $H = (V, F)$ has the following properties.*

1. *If $v_{i,j}$, $v_{i+2,j+2} \in V$, then $v_{i+1,j+1} \in V$.*
2. *Given any subset of vertices $V' \subset V$, let $E'_0 = \{e_{i,j} | v_{i,j} \in V'\}$, $A' = \{a_i | e_{i,j} \in E'_0\}$, and $B' = \{b_j | e_{i,j} \in E'_0\}$. If the subgraph $G'_0 = (A', B', E'_0)$ in H is connected, then all*



(A) The bipartite graph $G_0 = (A, B, E_0)$, where the ten edges in bold form a perfect matching.



(B) The instance graph $H = (V, F)$ of MIS, where the eight filled vertices form an independent set.

FIGURE 3.3: An instance of the k-MAX-DUO problem with $A = (a, b, c, d, e, f, b, c, d, e)$ and $B = (f, b, c, d, e, a, b, c, d, e)$. Figure 3.3a is the graphical view as a bipartite graph $G_0 = (A, B, E_0)$, where a perfect matching consisting of the ten bold edges form into eight pairs of parallel edges, corresponding to the eight preserved duos $(a, b), (b, c), (c, d), (d, e), (f, b), (b, c), (c, d)$ and (d, e). Figure 3.3b shows the instance graph $H = (V, F)$ of MIS constructed from H, where the independent set $\{v_{1,6}, v_{2,7}, v_{3,8}, v_{4,9}, v_{6,1}, v_{7,2}, v_{8,3}, v_{9,4}\}$ corresponds to the eight pairs of parallel edges shown in Figure 3.3a, and consequently also corresponds to the eight preserved duos. In this instance, we have $k = 2$. Any maximum independent set of H must contain some of the degree-6 vertices, invalidating the $(1.6 + \epsilon)$-approximation for 2-MAX-DUO proposed in [10].

the vertices of V' have the same letter content; and consequently for any two vertices $v_{i,j}, v_{h,\ell} \in V'$, we have both $v_{h,j}, v_{i,\ell} \in V$.

3. *For any $v_{i,j} \in V$, we have*

$$N(v_{i,j}) = \bigcup_{p=-1,0,1} \{v_{i'+p,j+p} \in V \mid i' \neq i\} \cup \bigcup_{p=-1,0,1} \{v_{i+p,j'+p} \in V \mid j' \neq j\}. \tag{3.1}$$

Proof. By definition, $v_{i,j} \in V$ if and only if $e_{i,j}, e_{i+1,j+1} \in E_0$.

1. If also $v_{i+2,j+2} \in V$, that is, $e_{i+2,j+2}$, $e_{i+3,j+3} \in E_0$, then $e_{i+1,j+1}, e_{i+2,j+2} \in E_0$ leading to $v_{i+1,j+1} \in V$.

2. Note that an edge $e_{i,j} \in E_0$ if and only if the two vertices a_i and b_j are the same letter, and clearly each connected component in H is complete bipartite and all the vertices are the same letter. It follows that if the induced subgraph $G'_0 = (A', B', E'_0)$ in G_0 is connected, then all its vertices are the same letter; furthermore, all the duos starting with these vertices have the same letter content; and therefore for any two vertices $v_{i,j}, v_{h,\ell} \in V'$, both $v_{h,j}, v_{i,\ell} \in V$.

3. For any vertex $v_{i,j}$, or equivalently the pair of parallel edges $(e_{i,j}, e_{i+1,j+1})$ in E_0, which are incident on four vertices $a_i, a_{i+1}, b_j, b_{j+1}$, a conflicting pair of parallel edges can be one of the six kinds: to share exactly one of the four vertices $a_i, a_{i+1}, b_j, b_{j+1}$, to share exactly two vertices a_i and a_{i+1}, and to share exactly two vertices b_j and b_{j+1}. The sets of these six kinds of conflicting pairs are as described in the lemma, for example, $\{v_{i'-1,j-1} \in V \mid i' \neq i\}$ is the set of conflicting pairs each sharing only the vertex b_j with the pair $v_{i,j}$.

□

From Lemma 3.1 and its proof, we see that for any vertex of V there are at most $k-1$ conflicting vertices of each kind (corresponding to a set in Equation 3.1). We thus have the following corollary.

Corollary 3.2. *The maximum degree of the vertices in $H = (V, F)$ is $\Delta \leq 6(k-1)$.*

3.3 On approximating the 2-Max-Duo problem

3.3.1 Properties for the graph H when $k = 2$

First, from Corollary 3.2 we have $\Delta \leq 6$. Berman and Fujito [7] have presented an approximation algorithm with a performance ratio arbitrarily close to $(\Delta + 3)/5$ for the MIS problem, on graphs with maximum degree Δ. This immediately implies a $(1.8 + \epsilon)$-approximation for 2-Max-Duo. Our goal is to reduce the maximum degree of the graph $H = (V, F)$ to achieve a better approximation algorithm. To this purpose, we examine all the degree-6 and degree-5 vertices in the graph H, and show a scheme to safely remove them from consideration when computing an independent set. This gives rise to a new graph H_2 with maximum degree at most 4, leading to a desired $(1.4 + \epsilon)$-approximation for 2-Max-Duo.

We remark that, in our scheme we first remove the degree-6 vertices from H to compute an independent set, and later we add half of these degree-6 vertices to the computed independent set to become the final solution. Contrary to the claim that there always exists a maximum independent set in H containing no degree-6 vertices [10, Lemma 1], the instance in Chapter 3.3 shows that any maximum independent set for the instance must contain some degree-6 vertices, thus invalidating the $(1.6 + \epsilon)$-approximation for 2-Max-Duo proposed in [10].

In more details, the instance of 2-Max-Duo, illustrated in Chapter 3.3, consists of two length-10 strings $A = (a, b, c, d, e, f, b, c, d, e)$ and $B = (f, b, c, d, e, a, b, c, d, e)$. The bipartite graph $G_0 = (A, B, E_0)$ is shown in Chapter 3.3a and the instance graph $H = (V, F)$ of the MIS problem is shown in Chapter 3.3b. In the graph H, we have six degree-6 vertices: $v_{2,2}, v_{7,7}, v_{3,3}, v_{3,8}, v_{8,3}$ and $v_{8,8}$. One can check that $\{v_{1,6}, v_{2,7}, v_{3,8}, v_{4,9}, v_{6,1}, v_{7,2}, v_{8,3}, v_{9,4}\}$ is an independent set in H, of size 8. On the other hand, if none of these degree-6 vertices is included in an independent set, then because the four vertices $v_{4,4}, v_{4,9}, v_{9,4}, v_{9,9}$ form a square implying that at most two of them can be included in the independent set, the independent set would be of size at most 6, and thus can never be a maximum independent set in H.

Consider a duo (a_i, a_{i+1}) of the string A and for ease of presentation assume its letter content is "*ab*". If no duo of the string B has the same letter content "*ab*", then this duo of the string A can never be preserved; in fact this duo does not even become (a part of) a vertex of V of the graph H. If there is exactly one duo (b_j, b_{j+1}) of the string B having the same letter content "*ab*", then these two duos make up a vertex $v_{i,j} \in V$, and from Lemma 3.1 we know that the degree of the vertex $v_{i,j} \in V$ is at most 5, since there is no such vertex $v_{i,j'}$ with $j' \neq j$ sharing exactly the two letters a_i and a_{i+1} with $v_{i,j}$. Therefore, if the degree of the vertex $v_{i,j} \in V$ is six, then there must be two duos of the string A and two duos of the string B having the same letter content "*ab*". Assume the other duo of the string A and the other duo of the string B having the same letter content "*ab*" are $(a_{i'}, a_{i'+1})$ and $(b_{j'}, b_{j'+1})$, respectively. Then all four vertices $v_{i,j}, v_{i,j'}, v_{i',j}, v_{i',j'}$ exist in V. We call the subgraph of H induced by these four vertices a *square*, and denote it as $S(i, i'; j, j') = (V(i, i'; j, j'), F(i, i'; j, j'))$, where $V(i, i'; j, j') = \{v_{i,j}, v_{i,j'}, v_{i',j}, v_{i',j'}\}$ and $F(i, i'; j, j') = \{(v_{i,j}, v_{i,j'}), (v_{i,j}, v_{i',j}), (v_{i',j'}, v_{i,j'}), (v_{i',j'}, v_{i',j})\}$ due to their conflicting relationships. One clearly sees that every square has a unique letter content, which is the letter content of its four member vertices.

In Figure 3.3b, there are three squares $S(2, 7; 2, 7)$, $S(3, 8; 3, 8)$ and $S(4, 9; 4, 9)$, with their letter contents "*bc*", "*cd*" and "*de*", respectively. The above argument says that every degree-6 vertex of V must belong to a square, but the converse is not necessarily true, for example, all vertices of the square $S(4, 9; 4, 9)$ have degree 4. We next characterize several properties of a square.

The following lemma is a direct consequence of how the graph H is constructed and the fact that $k = 2$.

Lemma 3.3. *In the graph $H = (V, F)$ constructed from an instance of 2-Max-Duo,*

1. *for each index i, there are at most two distinct j and j' such that $v_{i,j}, v_{i,j'} \in V$;*
2. *if $v_{i,j}, v_{i,j'} \in V$ where $j' \neq j$, and $v_{i+1,j''+1} \in V$ (or symmetrically, $v_{i-1,j''-1} \in V$), then either $j'' = j$ or $j'' = j'$.*

Lemma 3.4. *For any square $S(i, i'; j, j')$ in the graph $H = (V, F)$, $N(v_{i,j}) = N(v_{i',j'})$, $N(v_{i,j'}) = N(v_{i',j})$, and $N(v_{i,j}) \cap N(v_{i,j'}) = \emptyset$. (Together, these imply that every vertex of V is adjacent to either none or exactly two of the four member vertices of a square.)*

Proof. Consider the two vertices $v_{i,j}$ and $v_{i',j'}$, which have common neighbors $v_{i,j'}$ and $v_{i',j}$ in the square.

Note that $v_{i,j'}$ and $v_{i,j}$ share both the letters a_i and a_{i+1}. If there is a vertex adjacent to $v_{i,j}$ by sharing a_{i+1} but not a_i, then this vertex is $v_{i+1,j''+1}$ with $j'' \neq j$, and thus it has to be $v_{i+1,j'+1}$ (by Lemma 3.3). We consider two subcases: If $i + 1 = i' - 1$, then $j' + 1 = j - 1$ due to $k = 2$. Thus, this vertex $v_{i+1,j'+1}$ actually shares a_{i+1} and b_j with $v_{i,j}$; also, it shares $a_{i'}$ and $b_{j'+1}$ with $v_{i',j'}$; and therefore it is adjacent to $v_{i',j'}$ too, but not adjacent to $v_{i,j'}$ or $v_{i',j}$. If $i + 1 \neq i' - 1$, then this vertex $v_{i+1,j'+1}$ shares only a_{j+1} with the vertex $v_{i,j}$; also it shares only $b_{j'+1}$ with $v_{i',j'}$; and therefore it is adjacent to $v_{i',j'}$ too, but not adjacent to $v_{i,j'}$ or $v_{i',j}$.

The other three symmetric cases can be discussed exactly the same and the lemma is proved. □

Corollary 3.5. *In the graph $H = (V, F)$, the degree-6 vertices can be partitioned into pairs, where each pair of degree-6 vertices belong to a square in H and they are adjacent to the same six other vertices, two inside the square and four outside of the square.*

Proof. We have seen that every degree-6 vertex in the graph H must be in a square. The above Lemma 3.4 states that the four vertices of a square $S(i, i'; j, j')$ can be partitioned into two pairs, $\{v_{i,j}, v_{i',j'}\}$ and $\{v_{i,j'}, v_{i',j}\}$, and the two vertices inside each pair are non-adjacent to each other and have the same neighbors. In particular, if the vertex $v_{i,j}$ in the square $S(i, i'; j, j')$ has degree 6, then Lemma 3.1 states that it is adjacent to the six vertices $v_{i-1,j'-1}, v_{i,j'}, v_{i+1,j'+1}, v_{i'-1,j-1}, v_{i',j}, v_{i'+1,j+1}$ (see an illustration in Figure 3.4). □

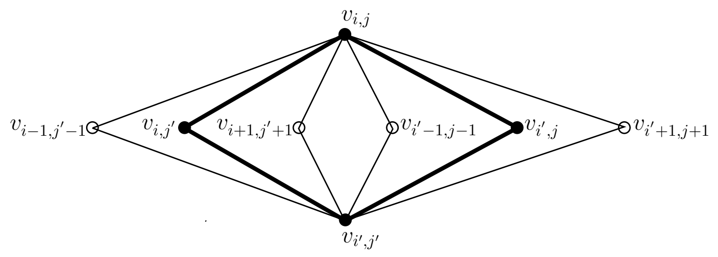


FIGURE 3.4: The square $S(i,i';j,j')$ shown in bold lines. The two non-adjacent vertices $v_{i,j}$ and $v_{i',j'}$ of the square form a pair stated in Corollary 3.5; they have 6 common neighbors, of which two inside the square and four outside of the square.

Corollary 3.6. *If there is no square in the graph $H = (V, F)$, then every degree-5 vertex is adjacent to a degree-1 vertex.*

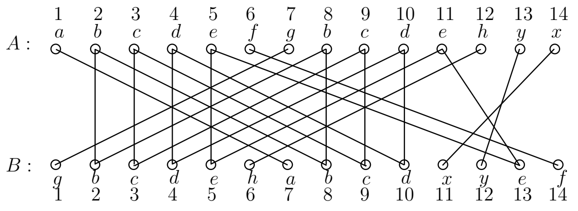
Proof. Assume the vertex $v_{i,j}$ has degree 5. Due to the non-existence of any square in the graph H and Lemma 3.1, either there is no vertex sharing exactly the two letters a_i and a_{i+1} with $v_{i,j}$, or there is no vertex sharing exactly the two letters b_j and b_{j+1} with $v_{i,j}$. We assume without loss of generality that there is no vertex sharing exactly the two letters a_i and a_{i+1} with $v_{i,j}$, and furthermore assume $v_{i',j}$, $i' \neq i$, is the vertex sharing exactly the two letters b_j and b_{j+1} with $v_{i,j}$.

It follows that $N(v_{i,j}) = \{v_{i-1,j''-1}, v_{i+1,j'''+1}, v_{i'-1,j-1}, v_{i',j}, v_{i'+1,j+1}\}$, for some $j'' \neq j$ and $j''' \neq j$. Due to $k = 2$, this implies that $a_{i-1} \neq b_{j-1} = a_{i'-1}$ and $a_{i+2} \neq b_{j+2} = a_{i'+2}$. Therefore, there is no vertex of V sharing exactly the letter $a_{i'}$ ($a_{i'+1}, b_j, b_{j+1}$, respectively) with the vertex $v_{i',j}$, neither a vertex of V sharing exactly the two letters $a_{i'}$ and $a_{i'+1}$ with the vertex $v_{i',j}$. That is, the vertex $v_{i',j}$ is adjacent to only $v_{i,j}$ in the graph H. □

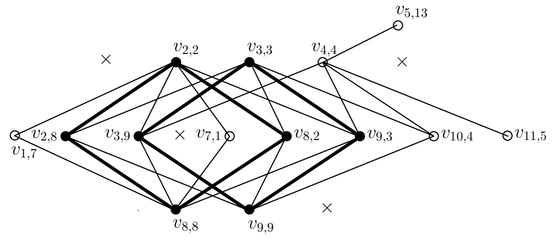
We say the two vertices $v_{i,j}$ and $v_{i+1,j+1}$ of V are *consecutive*; and we say the two squares $S(i,i';j,j')$ and $S(i+1,i'+1;j+1,j'+1)$ in H are *consecutive*. Clearly, two consecutive squares contain four pairs of consecutive vertices. The following Lemma 3.7 summarizes the fact that when two consecutive vertices belong to two different squares, then these two squares are also consecutive (and thus contain the other three pairs of consecutive vertices).

Lemma 3.7. *In the graph H, if there are two consecutive vertices $v_{i,j}$ and $v_{i+1,j+1}$ belonging to two different squares $S(i_1,i'_1;j_1,j'_1)$ and $S(i_2,i'_2;j_2,j'_2)$ respectively, then $i_2 = i_1 + 1, i'_2 = i'_1 + 1, j_2 = j_1 + 1, j'_2 = j'_1 + 1$, i.e., these two squares are consecutive.*

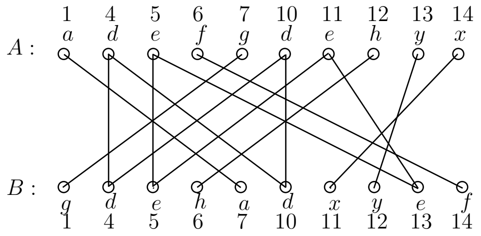
Proof. This is a direct result of the fact that no two distinct squares have any member vertex in common. □



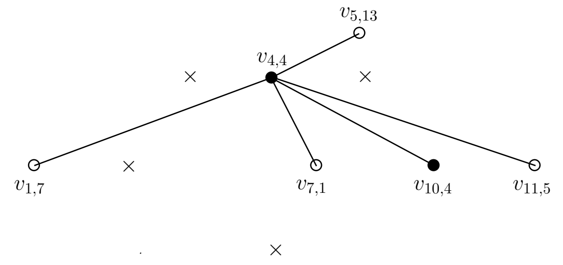
(A) The bipartite graph $G_0 = (A, B, G_0)$.



(B) The instance graph $H = (V, F)$.



(C) The bipartite graph $G'_0 = (A', B', E'_0)$ after removal of $\mathcal{S}^2(2, 8; 2, 8)$.



(D) The updated instance graph $H' = (V', F')$ after removal of $\mathcal{S}^2(2, 8; 2, 8)$.

Figure 3.5: An instance of the 2-Max-Duo problem with $A = (a, b, c, d, e, f, g, b, c, d, e, h, y, x)$ and $B = (g, b, c, d, e, h, a, b, c, d, x, y, e, f)$. The bipartite graph $G_0 = (A, B, E_0)$ is shown in Figure 3.5a and the instance graph $H = (V, F)$ of the MIS problem is shown in Figure 3.5b. There is a maximal series of 2 squares $\mathcal{S}^2(2, 8; 2, 8)$ in the graph G, with the four substrings "*bcd*". The bipartite graph $G'_0 = (A', B', E'_0)$ is shown in Figure 3.5c and the graph $H' = (V', F')$ is shown in Figure 3.5d, on $A' = (a, d, e, f, g, d, e, h, y, x)$ and $B' = (g, d, e, h, a, d, x, y, e, f)$. Applying the vertex contracting process on H also gives the graph H'.

A series of p consecutive squares $\{S(i + q, i' + q; j + q, j' + q), q = 0, 1, \ldots, p - 1\}$ in the graph H, where $p \geq 1$, is *maximal* if none of the square $S(i - 1, i' - 1; j - 1, j' - 1)$ and the square $S(i + p, i' + p; j + p, j' + p)$ exists in the graph H. Note that the non-existence of the square $S(i - 1, i' - 1; j - 1, j' - 1)$ in H does not rule out the existence of some of the four vertices $v_{i-1,j-1}, v_{i'-1,j'-1}, v_{i-1,j'-1}, v_{i'-1,j-1}$ in V; in fact by Lemma 3.1 there can be as many as two of these four vertices existing in V (however, more than two would imply the existence of the square). Similarly, there can be as many as two of the four vertices $v_{i+p,j+p}, v_{i'+p,j'+p}, v_{i+p,j'+p}, v_{i'+p,j+p}$ existing in V. In the sequel, a maximal series of p consecutive squares starting with $S(i, i'; j, j')$ is denoted as $\mathcal{S}^p(i, i'; j, j')$, where $p \geq 1$. See for an example in Figure 3.5b where there is a maximal series of 2 consecutive squares $\mathcal{S}^2(2, 8; 2, 8)$, where the instance of the 2-Max-Duo is expanded slightly from the instance shown in Figure 3.3.

Lemma 3.8. *Suppose $\mathcal{S}^p(i, i'; j, j')$, where $p \geq 1$, exists in the graph H. Then,*

1. *the two substrings* $(a_i, a_{i+1}, \ldots, a_{i+p})$ *and* $(a_{i'}, a_{i'+1}, \ldots, a_{i'+p})$ *of the string* A *and the two substrings* $(b_j, b_{j+1}, \ldots, b_{j+p})$ *and* $(b_{j'}, b_{j'+1}, \ldots, b_{j'+p})$ *of the string* B *are identical and do not overlap;*
2. *if a maximum independent set of* H *contains less than* $2p$ *vertices from* $\mathcal{S}^p(i, i'; j, j')$*, then it must contain either the four vertices* $v_{i-1,j-1}, v_{i'-1,j'-1}, v_{i'+p,j+p}, v_{i+p,j'+p}$ *or the four vertices* $v_{i'-1,j-1}, v_{i-1,j'-1}, v_{i+p,j+p}, v_{i'+p,j'+p}$*.*

Proof. By the definition of the square $S(i+q, i'+q; j+q, j'+q)$, we have $a_{i+q} = a_{i'+q}$ and $a_{i+q+1} = a_{i'+q+1}$; we thus conclude that the two substrings $(a_i, a_{i+1}, \ldots, a_{i+p})$ and $(a_{i'}, a_{i'+1}, \ldots, a_{i'+p})$ are identical. In Figure 3.5b, for $\mathcal{S}^2(2, 8; 2, 8)$ the two substrings are "*bcd*". If these two substrings overlapped, then there would be three occurrences of at least one letter, contradicting the fact that $k = 2$. This proves the first item.

Note that the square $S(i-1, i'-1; j-1, j'-1)$ does not exist in the graph H, and thus at most two of its four vertices (which are $v_{i-1,j-1}, v_{i'-1,j-1}, v_{i-1,j'-1}$ and $v_{i'-1,j'-1}$) exist in V. We claim that if no vertex of the square $S(i, i'; j, j')$ is in I^*, then there are exactly two of the four vertices $v_{i-1,j-1}, v_{i'-1,j-1}, v_{i-1,j'-1}$ and $v_{i'-1,j'-1}$ exist in V and they both are in I^*. Suppose otherwise there is at most one of the four vertices in I^*, say $v_{i-1,j-1}$; we may increase the size of I^* by removing $v_{i-1,j-1}$ while adding either the two vertices $v_{i,j}$ and $v_{i',j'}$ or the two vertices $v_{i',j}$ and $v_{i,j'}$ (depending on which vertices of the square $S(i+1, i'+1; j+1, j'+1)$ are in I^*), a contradiction.

Assume next that a vertex of the square $S(i, i'; j, j')$ is in I^*, say $v_{i,j}$; then due to maximality of I^* and Lemma 3.4 both $v_{i,j}$ and $v_{i',j'}$ are in I^*. We claim and prove similarly as in the last paragraph that if no vertex of the square $S(i+1, i'+1; j+1, j'+1)$ is in I^*, then there are exactly two of the four vertices $v_{i-1,j-1}, v_{i'-1,j-1}, v_{i-1,j'-1}$ and $v_{i'-1,j'-1}$ exist in V and they both are in I^*. If there is a vertex of the square $S(i+1, i'+1; j+1, j'+1)$ in I^*, then it must be one of $v_{i+1,j+1}$ and $v_{i'+1,j'+1}$; and due to maximality and Lemma 3.4 both $v_{i+1,j+1}$ and $v_{i'+1,j'+1}$ are in I^*. And so on; repeatedly applying this argument, we claim and prove similarly that if no vertex of the square $S(i+p-1, i'+p-1; j+p-1, j'+p-1)$ is in I^*, then there are exactly two of the four vertices $v_{i-1,j-1}, v_{i'-1,j-1}, v_{i-1,j'-1}$ and $v_{i'-1,j'-1}$ exist in V and they both are in I^*. If there is a vertex of the square $S(i+p-1, i'+p-1; j+p-1, j'+p-1)$ in I^*, then it must be one of $v_{i+p-1,j+p-1}$ and $v_{i'+p-1,j'+p-1}$; and due to maximality and Lemma 3.4 both $v_{i+p-1,j+p-1}$ and $v_{i'+p-1,j'+p-1}$ are in I^*.

To summarize, we proved in the above two paragraphs that if I^* contains less than $2p$ vertices from $\mathcal{S}^p(i, i'; j, j')$, then there are exactly two of the four vertices $v_{i-1,j-1}, v_{i'-1,j-1}, v_{i-1,j'-1}$

and $v_{i'-1,j'-1}$ exist in V and they both are in I^*; and these two vertices are either $v_{i-1,j-1}$ and $v_{i'-1,j'-1}$ or $v_{i'-1,j-1}$ and $v_{i-1,j'-1}$. Symmetrically, there are exactly two of the four vertices $v_{i+p,j+p}, v_{i'+p,j+p}, v_{i+p,j'+p}$ and $v_{i'+p,j'+p}$ exist in V and they both are in I^*; and these two vertices are either $v_{i+p,j+p}$ and $v_{i'+p,j'+p}$ or $v_{i'+p,j+p}$ and $v_{i+p,j'+p}$. Clearly from the above, when the combination is $v_{i-1,j-1}$ and $v_{i'-1,j'-1}$ versus $v_{i+p,j+p}$ and $v_{i'+p,j'+p}$, we may increase the size of I^* to contain exactly $2p$ vertices from $\mathcal{S}^p(i, i'; j, j')$ without affecting any vertex outside of $\mathcal{S}^p(i, i'; j, j')$, a contradiction. Therefore, the only possible combinations are $v_{i-1,j-1}$ and $v_{i'-1,j'-1}$ versus $v_{i'+p,j+p}$ and $v_{i+p,j'+p}$, and $v_{i'-1,j-1}$ and $v_{i-1,j'-1}$ versus $v_{i+p,j+p}$ and $v_{i'+p,j'+p}$. This proves the second item of the lemma. □

Suppose $\mathcal{S}^p(i, i'; j, j')$, where $p \geq 1$, exists in the graph H. Let A' denote the string obtained from A by removing the two substrings $(a_i, a_{i+1}, \ldots, a_{i+p-1})$ and $(a_{i'}, a_{i'+1}, \ldots, a_{i'+p-1})$ and concatenating the remainder together, and B' denote the string obtained from B by removing the two substrings $(b_j, b_{j+1}, \ldots, b_{j+p-1})$ and $(b_{j'}, b_{j'+1}, \ldots, b_{j'+p-1})$ and concatenating the remainder. Let the graph $H' = (V', F')$ denote the instance graph of the MIS problem constructed from the two strings A' and B'. See for an example H' in Figure 3.5d, where there is a maximal series of 2 consecutive squares $\mathcal{S}^2(2, 8; 2, 8)$ in the graph H.

Corollary 3.9. *Suppose $\mathcal{S}^p(i, i'; j, j')$, where $p \geq 1$, exists in the graph H. Then, the union of a maximum independent set in the graph $H' = (V', F')$ and certain $2p$ vertices from $\mathcal{S}^p(i, i'; j, j')$ becomes a maximum independent set in the graph $H = (V, F)$, where these certain $2p$ vertices are $v_{i,j}, v_{i+1,j+1}, \ldots, v_{i+p-1,j+p-1}$ and $v_{i',j'}, v_{i'+1,j'+1}, \ldots, v_{i'+p-1,j'+p-1}$ if $v_{i-1,j-1}$ or $v_{i+p,j+p}$ is in the maximum independent set in H', or they are $v_{i',j}, v_{i'+1,j+1}, \ldots,$ $v_{i'+p-1,j+p-1}$ and $v_{i,j'}, v_{i+1,j'+1}, \ldots, v_{i+p-1,j'+p-1}$ if $v_{i'-1,j-1}$ or $v_{i'+p,j+p}$ is in the maximum independent set in H'.*

Proof. Consider the construction of the graph $H' = (V', F')$ from the two strings A' and B'. Equivalently, starting with the graph $H = (V, F)$, if we contract the p vertices $v_{i,j}, v_{i+1,j+1}, \ldots, v_{i+p-1,j+p-1}$ into the vertex $v_{i+p,j+p}$ if it exists or otherwise into a void vertex, contract the p vertices $v_{i',j'}, v_{i'+1,j'+1}, \ldots, v_{i'+p-1,j'+p-1}$ into the vertex $v_{i'+p,j'+p}$ if it exists or otherwise into a void vertex, contract the p vertices $v_{i',j}, v_{i'+1,j+1}, \ldots, v_{i'+p-1,j+p-1}$ into the vertex $v_{i'+p,j+p}$ if it exists or otherwise into a void vertex, and contract the p vertices $v_{i,j'}, v_{i+1,j'+1}, \ldots, v_{i+p-1,j'+p-1}$ into the vertex $v_{i+p,j'+p}$ if it exists or otherwise into a void vertex, then we obtain a graph that is exactly H'. In the graph H', the vertices $v_{i-1,j-1}$ and $v_{i'+p,j+p}$, if both exist in V, become adjacent to each other; so are the vertices $v_{i'-1,j-1}$ and $v_{i+p,j+p}$, if both exist in V. It follows that the maximum independent set in the graph $H' = (V', F')$ does not contain both vertices $v_{i-1,j-1}$ and $v_{i'+p,j+p}$, or both vertices $v_{i'-1,j-1}$ and

$v_{i+p,j+p}$. Therefore, starting with the maximum independent set in the graph $H' = (V', F')$, we can add exactly $2p$ vertices from $\mathcal{S}^p(i, i'; j, j')$ to form an independent set in H, of which the maximality can be proved by a simple contradiction.

We remark that in the extreme case where none of the vertices of $S(i-1, i'-1; j-1, j'-1)$ and none of the vertices of $S(i+p, i'+p; j+p, j'+p)$ are in the maximum independent set in H', we may add either of the two sets of $2p$ vertices from $\mathcal{S}^p(i, i'; j, j')$ to form a maximum independent set in H. □

Iteratively applying the above string shrinkage process, or equivalently the vertex contracting process, associated with the elimination of a maximal series of consecutive squares. In $O(n)$ iterations, we achieve the final graph containing no squares, which we denote as $H_1 = (V_1, F_1)$.

3.3.2 An approximation algorithm for 2-Max-Duo

A high-level description of the approximation algorithm, denoted as Approx, for the 2-Max-Duo problem is depicted in Figure 3.6.

```
Algorithm Approx
 1: Construct the graph $H = (V, F)$ from two input strings $A$ and $B$;
 2: while (there is a square in the graph) do
 3:     find a maximal series of squares;
 4:     locate the four identical substrings of $A$ and $B$ as in Lemma 3.8;
 5:     remove the corresponding substrings and accordingly update the graph;
 6: end while
 7: denote the resultant graph as $H_1 = (V_1, F_1)$;
 8: set $L_1$ to contain all degree-0 and degree-1 vertices of $H_1$;
 9: set $N[L_1]$ to be the closed neighborhood of $L_1$ in $H_1$, i.e. $N[L_1] = L_1 \cup N(L_1)$;
10: set $H_2 = H_1[V_1 - N[L_1]]$, the subgraph of $H_1$ induced by $V_1 - N[L_1]$;
11: compute an independent set $I_2$ in $H_2$ by the $((\Delta + 3)/5 + \epsilon)$-approximation in [7];
12: set $I_1 = I_2 \cup L_1$, an independent set in $H_1$;
13: return an independent set $I$ in $H$ using $I_1$ and Corollary 3.9.
```

Figure 3.6: A high-level description of the approximation algorithm for 2-Max-Duo.

In more details, given an instance of the 2-Max-Duo problem with two length-n strings A and B, the first step of our algorithm is to construct the graph $H = (V, F)$, which is done in $O(n^2)$ time. In the second step (Lines 2–7 in Figure 3.6), it iteratively applies the vertex contracting process presented in Section 3.3.1 at the existence of a maximal series of consecutive squares, and at the end it achieves the final graph $H_1 = (V_1, F_1)$ which does not contain any square. This second step can be done in $O(n^2)$ time too since each iteration of vertex contracting process is done in $O(n)$ time and there are $O(n)$ iterations. In the third step (Lines 8–10 in Figure 3.6), let L_1 denote the set of singletons (degree-0 vertices) and leaves (degree-1 vertices) in the graph H_1; our algorithm removes all the vertices of L_1 and their neighbors from the graph H_1 to obtain the remainder graph $H_2 = (V_2, F_2)$. This step can be done in $O(n^2)$ time too due to $|V_1| \leq |V| \leq 2n$, and the resultant graph H_2 has maximum degree $\Delta \leq 4$ by Corollaries 3.5 and 3.6. (See for an example illustrated in Figure 3.7a.) In the fourth step (Lines 11–12 in Figure 3.6), our algorithm calls the state-of-the-art approximation algorithm for the MIS problem [7] on the graph H_2 to obtain an independent set I_2 in H_2; and returns $I_1 = L_1 \cup I_2$ as an independent set in the graph H_1. The running time of this step is dominated by the running time of the state-of-the-art approximation algorithm for the MIS problem, which is a high polynomial in n and $1/\epsilon$. In the last step (Line 13 in Figure 3.6), using the independent set I_1 in H_1, our algorithm adds $2p$ vertices from each maximal series of p consecutive squares according to Corollary 3.9, to produce an independent set I in the graph H. (For an illustrated example see Figure 3.7b.) The last step can be done in $O(n)$ time.

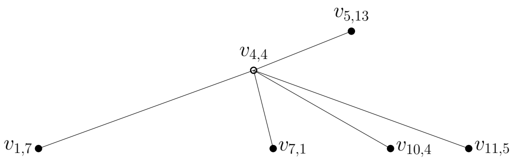
The state-of-the-art approximation algorithm for the MIS problem on a graph with maximum degree Δ has a performance ratio of $(\Delta + 3)/5 + \epsilon$, for any $\epsilon > 0$ [7].

Lemma 3.10. *In the graph $H_1 = (V_1, F_1)$, let OPT_1 denote the cardinality of a maximum independent set in F_1, and let SOL_1 denote the cardinality of the independent set I_1 returned by the algorithm* Approx*. Then, $OPT_1 \leq (1.4 + \epsilon)SOL_1$, for any $\epsilon > 0$.*

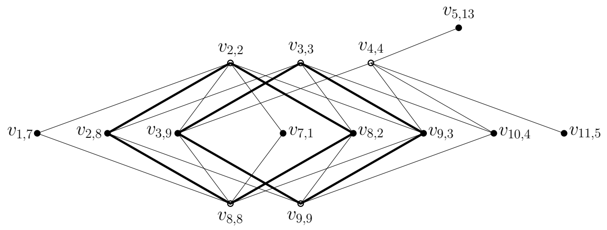
Proof. Let L_1 denote the set of singletons (degree-0 vertices) and leaves (degree-1 vertices) in the graph H_1; our algorithm Approx removes all the vertices of L_1 and their neighbors from the graph H_1 to obtain the remainder graph $H_2 = (V_2, F_2)$. The graph H_2 has maximum degree $\Delta \leq 4$ by Corollaries 3.5 and 3.6. Let OPT_2 denote the cardinality of a maximum independent set in H_2, and let SOL_2 denote the cardinality of the independent set I_2 returned by the state-of-the-art approximation algorithm for the MIS problem. We have $\mathrm{OPT}_1 = |L_1| + \mathrm{OPT}_2$ and $\mathrm{OPT}_2 \leq (1.4 + \epsilon)\mathrm{SOL}_2$, for any $\epsilon > 0$. Therefore,

$$\mathrm{OPT}_1 \leq |L_1| + (1.4 + \epsilon)\mathrm{SOL}_2 \leq (1.4 + \epsilon)(|L_1| + \mathrm{SOL}_2) = (1.4 + \epsilon)\mathrm{SOL}_1.$$

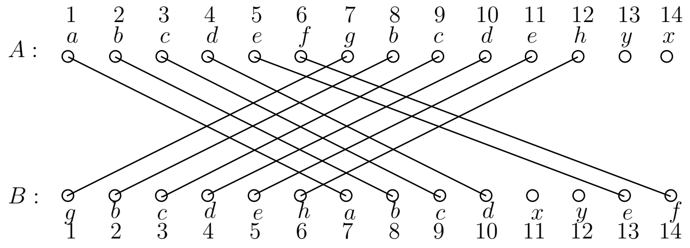
This proves the lemma. □



(A) The independent set $I_1 = \{v_{1,7}, v_{7,1}, v_{10,4}, v_{11,5}, v_{5,13}\}$ in H_1, consisting of all the five leaves of $H_1 = H'$ shown in Figure 3.5d.



(B) Using I_1, since $v_{10,4} \in I_1$, the four vertices $v_{2,8}, v_{3,9}, v_{8,2}, v_{9,3}$ are added to form an independent set I in the original graph H shown in Figure 3.5b.



(C) The parallel edges of G_0 corresponding to the independent set I shown in Figure 3.7b, also correspond to the 9 preserved duos $(a, b), (b, c), (c, d), (e, f), (g, b), (b, c), (c, d), (d, e), (e, h)$ for the instance shown in Figure 3.5a.

FIGURE 3.7: Illustration of the execution of our algorithm APPROX on the instance shown in Figure 3.5. The independent set I_1 in the graph H_1 is shown in Figure 3.7a in filled circles, for which we did not apply the state-of-the-art approximation algorithm for the MIS problem. The independent set I in the graph H is shown in Figure 3.7b in filled circles, according to Corollary 3.9 the four vertices $v_{2,8}, v_{3,9}, v_{8,2}, v_{9,3}$ are added due to $v_{10,4} \in I_1$. The parallel edges of G_0 corresponding to the vertices of I are shown in Figure 3.7c, representing a feasible solution to the 2-MAX-DUO instance shown in Figure 3.5.

Theorem 3.11. *The* 2-*MAX-DUO problem can be approximated within a ratio arbitrarily close to* 1.4*, by a linear reduction to the MIS problem.*

Proof. We prove by induction. At the presence of maximal series of p consecutive squares, we perform the vertex contracting process iteratively. In each iteration to handle one maximal series of p consecutive squares, let H and H' denote the graph before and after the contracting step, respectively. Let OPT′ denote the cardinality of a maximum independent set in H', and let SOL′ denote the cardinality of the independent set I' returned by the algorithm APPROX. Given any $\epsilon > 0$, from Lemma 3.10, we may assume that $\text{OPT}' \leq (1.4 + \epsilon)\text{SOL}'$.

Let OPT denote the cardinality of a maximum independent set in H, and let SOL denote the cardinality of the independent set returned by the algorithm APPROX, which adds $2p$ vertices from the maximal series of p consecutive squares to the independent set I' in H', according to Corollary 3.9, to produce an independent set I in the graph H. Lemma 3.8 states that $\mathrm{OPT} = \mathrm{OPT}' + 2p$. Therefore,

$$\mathrm{OPT} = \mathrm{OPT}' + 2p \leq (1.4 + \epsilon)\mathrm{SOL}' + 2p \leq (1.4 + \epsilon)(\mathrm{SOL}' + 2p) = (1.4 + \epsilon)\mathrm{SOL}.$$

This proves that for the original graph $H = (V, F)$ we also have $\mathrm{OPT} \leq (1.4 + \epsilon)\mathrm{SOL}$ accordingly. That is, the worst-case performance ratio of our algorithm Approx is $1.4 + \epsilon$, for any $\epsilon > 0$. The time complexity of the algorithm Approx has been determined to be polynomial at the beginning of the section, and it is dominated by the time complexity of the state-of-the-art approximation algorithm for the MIS problem. The theorem is thus proved. □

3.3.3 Review of the reduction from MIS on cubic graphs

In the above approximation algorithm Approx for 2-Max-Duo, we apply a vertex-degree reduction scheme on the constructed instance graph of the MIS problem, to remove all the degree-6 vertices and all the degree-5 vertices. This scheme essentially reduces the 2-Max-Duo problem to computing a maximum independent set in a graph of maximum degree $\Delta \leq 4$. One might wonder whether all the degree-4 vertices can be similarly removed.

Goldstein *et al.* [46] proved that the 2-MCSP problem is **APX**-hard via a linear reduction from *MIS on cubic graphs* (Cubic-MIS), and Boria *et al.* [10] showed that the same reduction could also be applied to prove that 2-Max-Duo is also APX-hard. In this section, we review this **APX**-hardness reduction from Cubic-MIS to 2-Max-Duo, to point out that it is unlikely possible to further reduce the maximum degree Δ from 4 to 3 by removing all the degree-4 vertices.

Given a cubic graph $H' = (V', F')$ as an input for Cubic-MIS, an instance of 2-Max-Duo can be created in the following steps.

1. For each $u \in V'$, define a small 2-Max-Duo instance $I_u = (A_u, B_u)$ as shown in Figure 3.8, where both A_u and B_u are length-28 strings with seven main substrings, and each pair of two consecutive main substrings are separated by a substring of two distinct letters (x_u^i, y_u^i) in A_u and by (y_u^i, x_u^i) in B_u, respectively, for $i = 1, \ldots, 6$. There 12 letters x_u^i's and y_u^i's are distinct, each appears only once in A_u and represented by dot in Figure 3.8.

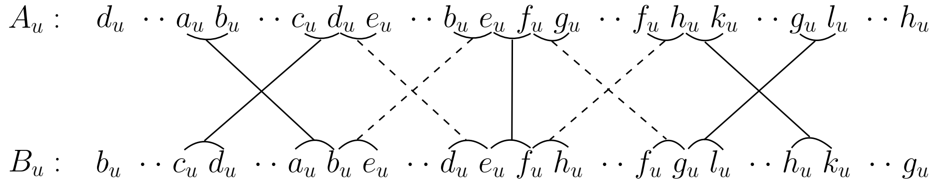

A_u : $d_u \cdot\cdot a_u b_u \cdot\cdot c_u d_u e_u \cdot\cdot b_u e_u f_u g_u \cdot\cdot f_u h_u k_u \cdot\cdot g_u l_u \cdot\cdot h_u$
B_u : $b_u \cdot\cdot c_u d_u \cdot\cdot a_u b_u e_u \cdot\cdot d_u e_u f_u h_u \cdot\cdot f_u g_u l_u \cdot\cdot h_u k_u \cdot\cdot g_u$

FIGURE 3.8: The instance $I_u = (A_u, B_u)$ defined for each vertex $u \in V'$. Each two dots between a pair of substrings represent a substring of two distinct letters (x_u^i, y_u^i) in A_u and (y_u^i, x_u^i) in B_u, respectively, for $i = 1, \ldots, 6$. All these 12 letters x_u^i, y_u^i are distinct and also distinct from all the other letters in A_u and B_u. Each solid or dashed line connects a pair of common duos in A_u and B_u. The set of five duos connecting by solid lines is a unique optimal solution to I_u.

One can easily check that there are nine common duos between A_u and B_u, and the set of five duos connected by solid lines in Figure 3.8 is a unique optimal solution to the instance I_u. Equivalently, this constructs a gadget subgraph of the MIS problem, as shown in Figure 3.9, in which there are nine vertices one-to-one corresponding to the nine common duos and two vertices are adjacent if and only if they are conflicting. The vertex subset $\{a_u b_u, c_u d_u, e_u f_u, g_u l_u, h_u k_u\}$ is the unique maximum independent set in this subgraph.

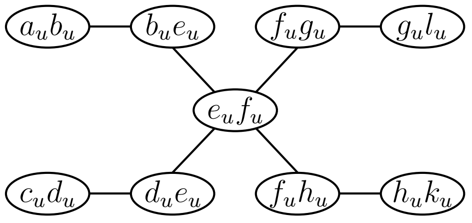
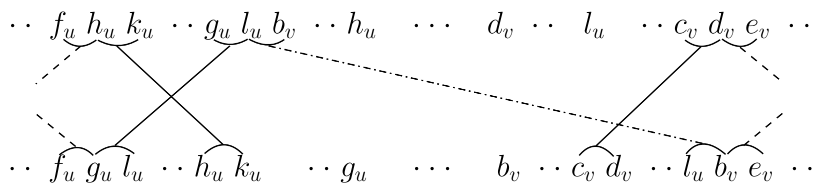


FIGURE 3.9: The gadget subgraph associated with the instance $I_u = (A_u, B_u)$, in which there are nine vertices corresponding to the nine common duos between A_u and B_u.

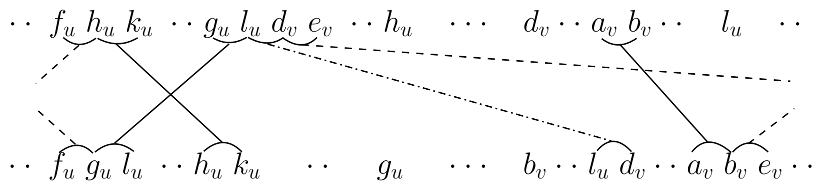
2. Orient each edge in F' such that each vertex in V' has an in-degree of at most two and an out-degree of at most two. This can be done by partitioning H' into a set of edge-disjoint cycles and a forest, followed by orienting the edges of a cycle to form a directed cycle, and rooting a tree at a leaf and then orient the edges away from the root.

3. Let $A_{H'} = \bigcup_{u \in V'} A_u$, $B_{H'} = \bigcup_{u \in V'} B_u$, $I_{H'} = (A_{H'}, B_{H'})$. For each $\overrightarrow{(u, v)} \in F'$, modify instances I_u and I_v such that an optimal solution to $I_{H'}$ coincides with at most one of the optimal solutions to I_u and I_v. To this purpose, either the common duo $a_v b_v$ is revised into $l_u b_v$ ($k_u b_v$, respectively) to be in conflict with only the common duo $g_u l_u$ ($h_u k_u$, respectively); the common duo $c_v d_v$ is revised into $l_u d_v$ ($k_u d_v$, respectively) to be in conflict with only the common duo $g_u l_u$ ($h_u k_u$, respectively). These four options

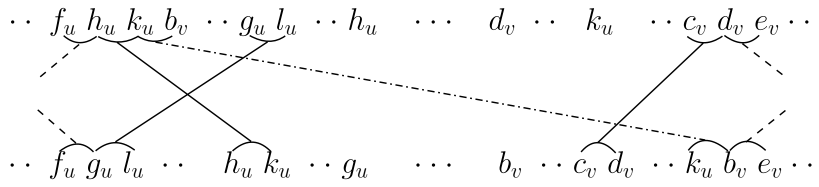
of modification are shown in Figure 3.10. Since every vertex of V' has at most two incoming edges and at most two outgoing edges, the revision process for the directed edge $(u, v) \in F'$ can be independently done with respect to all the other edges of F'.


$\cdots f_u\ h_u\ k_u \cdots g_u\ l_u\ b_v \cdots h_u \cdots d_v \cdots l_u \cdots c_v\ d_v\ e_v \cdots$
$\cdots f_u\ g_u\ l_u \cdots h_u\ k_u \cdots g_u \cdots b_v \cdots c_v\ d_v \cdots l_u\ b_v\ e_v \cdots$

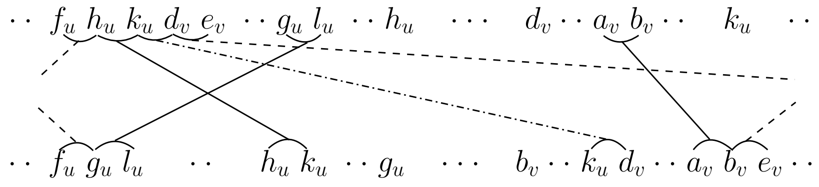
(A) The common duo $a_v b_v$ is revised into duo $l_u b_v$ to be in conflict with only the common duo $g_u l_u$.


$\cdots f_u\ h_u\ k_u \cdots g_u\ l_u\ d_v\ e_v \cdots h_u \cdots d_v \cdots a_v\ b_v \cdots l_u \cdots$
$\cdots f_u\ g_u\ l_u \cdots h_u\ k_u \cdots g_u \cdots b_v \cdots l_u\ d_v \cdots a_v\ b_v\ e_v \cdots$

(B) The common duo $c_v d_v$ is revised into duo $l_u d_v$ to be in conflict with only the common duo $g_u l_u$.


$\cdots f_u\ h_u\ k_u\ b_v \cdots g_u\ l_u \cdots h_u \cdots d_v \cdots k_u \cdots c_v\ d_v\ e_v \cdots$
$\cdots f_u\ g_u\ l_u \cdots h_u\ k_u \cdots g_u \cdots b_v \cdots c_v\ d_v \cdots k_u\ b_v\ e_v \cdots$

(C) The common duo $a_v b_v$ is revised into duo $k_u b_v$ to be in conflict with only the common duo $h_u k_u$.

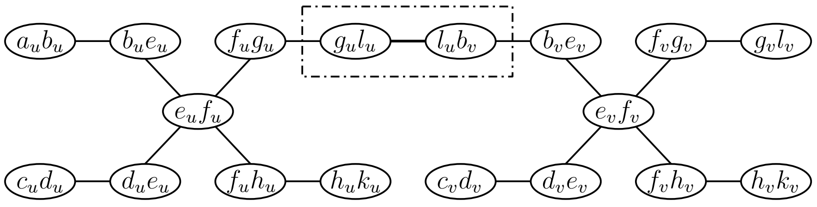

$\cdots f_u\ h_u\ k_u\ d_v\ e_v \cdots g_u\ l_u \cdots h_u \cdots d_v \cdots a_v\ b_v \cdots k_u \cdots$
$\cdots f_u\ g_u\ l_u \cdots h_u\ k_u \cdots g_u \cdots b_v \cdots k_u\ d_v \cdots a_v\ b_v\ e_v \cdots$

(D) The common duo $c_v d_v$ is revised into duo $k_u d_v$ to be in conflict with only the common duo $h_u k_u$.

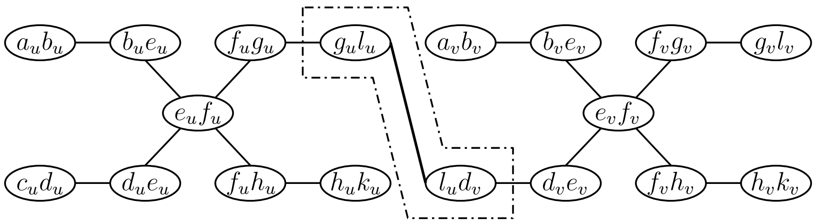
FIGURE 3.10: Four options of modifying instances I_u and I_v by only modifying the right side of A_u, B_u and the left side of A_v, B_v such that an optimal solution to $I_{H'}$ coincides with at most one of the optimal solutions to I_u and I_v.

According to the proofs in [10, 46], one can check that there exists an independent set of size α in H' if and only if there are $4|V'| + \alpha$ duos can be preserved in $I_{H'}$. The above common duo modification process for each directed edge $(u, v) \in F'$ is equivalent to joining the two gadget subgraphs for the vertices $u, v \in V'$ by connecting one of $g_u l_u$ and $h_u k_u$ to one of $a_v b_v$ and $c_v d_v$, but additionally revising the letter content of the common duo of I_v.

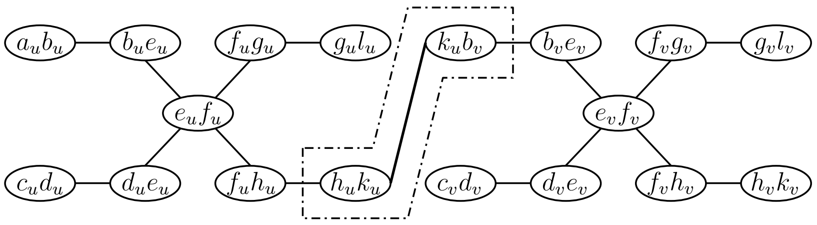
Corresponding to the four options of modification shown in Figure 3.10, the two gadget subgraphs are joined as shown in Figure 3.11, respectively.



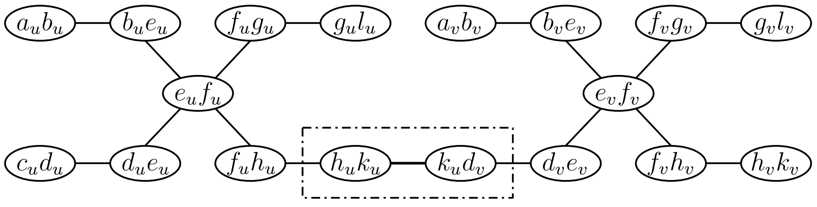
(A) When the modification is done as in Figure 3.10a, the vertex l_ub_v connects the two gadget subgraphs.



(B) When the modification is done as in Figure 3.10b, the vertex l_ud_v connects the two gadget subgraphs.



(C) When the modification is done as in Figure 3.10c, the vertex k_ub_v connects the two gadget subgraphs.



(D) When the modification is done as in Figure 3.10d, the vertex k_ud_v connects the two gadget subgraphs.

FIGURE 3.11: Four different configurations for joining the two gadget subgraphs for the vertices $u, v \in V'$, in each of which a common duo is revised for the directed edge $(u, v) \in E'$.

Since each directed edge $(u, v) \in F'$ gives rise to exactly one of the four possible configurations shown in Figure 3.11, we conclude from H' being cubic that exactly three of the four degree-1 vertices $\{a_ub_u, c_ud_u, g_ul_u, h_uk_u\}$ in the gadget subgraph for the vertex $u \in V'$

increase their degree to 2. It follows that however the edge orientation scheme is, all the vertices in the final graph H have degrees 1, 2, or 4.

Therefore, it is impossible to determine in polynomial time which subset of all the degree-4 vertices is in the maximum independent set of G. It would be interesting to investigate whether the maximum degree can be further reduced to 3, but not by determining in polynomial time which subset of all the degree-4 vertices is in the maximum independent set.

3.4 On approximating the general Max-Duo problem

3.4.1 A local search algorithm $\mathcal{LS}$

Given a bipartite graph $G = (D^A, D^B, E)$, the 3.5-approximation algorithm presented by Boria *et al.* [9] starts with an arbitrary maximal compatible matching, iteratively seeks swapping one edge in the current matching out for two compatible edges, and terminates when the expansion by such swapping is impossible.

Our local search algorithm is an extension of the above algorithm, to iteratively apply two different swapping operations to increase the size of the matching and to decrease the number of singleton edges in the matching, respectively. We present the algorithm in details in the following. Note that we also start with an arbitrary maximal compatible matching, which by Observation 3.1 can be obtained in $O(n^2)$-time, where n is the number of vertices in one side of the bipartite graph (or more precisely, $|D^A| = |D^B| = n - 1$).

Let M denote the current compatible matching in hand. For any edge $e_{i,j} \in M$, let $C(e_{i,j})$ be the set of all the edges of E conflicting with $e_{i,j}$; then ($q = -1, 0, +1$ in the following set unions)

$$C(e_{i,j}) = \bigcup_{q=-1}^{+1} \left\{ e_{i+q,j'+q} \in E \mid j' \neq j \right\} \cup \bigcup_{q=-1}^{+1} \left\{ e_{i'+q,j+q} \in E \mid i' \neq i \right\}. \tag{3.2}$$

Clearly, $|C(e_{i,j})| \leq 6(n-1)$. Recall that $|E| \in O(n^2)$. We have the following observation, which essentially narrows down the candidate edges for swapping with the edge $e_{i,j}$.

Observation 3.2. *For a maximal compatible matching M and an edge $e_{i,j} \in M$, the edges compatible with all the edges of $M - \{e_{i,j}\}$ must be in $C(e_{i,j}) \cup \{e_{i,j}\}$.*

We next describe the two different swapping operations. Both of them apply to a maximal compatible matching M. One operation is to replace 5 edges of M by 6 edges, denoted as REPLACE-5-BY-6, thus to increase the size of the matching; and the other operation is to replace 5 edges of M by 5 edges with the resulted matching having strictly less singleton edges, denoted as REDUCE-5-BY-5. Note that in each iteration, the operation REDUCE-5-BY-5 applies only when the operation REPLACE-5-BY-6 fails to expand the current matching M.

3.4.1.1 Operation REPLACE-5-BY-6

The operation REPLACE-5-BY-6 seeks to expand the current maximal compatible matching M by swapping five edges of M out for six compatible edges. It does so by scanning all size-5 subsets of M and terminates at a successful expansion. If no such expansion is possible, it also terminates but without making any change to the matching M.

Let $X = \{e_1, e_2, \ldots, e_5\}$ be a subset of M (in the special case where $|M| \leq 5$, we seek for a compatible matching of size $|M| + 1$ directly by an exhaustive search). The operation composes a set $E' = X \cup C(X)$, where $C(X)$ contains all the edges each conflicting with an edge of X but compatible with (all the edges of) $M - X$; it then checks every size-6 subset X' of E' for compatibility and, if affirmative, swaps X out for X' to expand M.

Recall that $|M| < n$. The number of size-5 subsets of M is $O(n^5)$. For each size-5 subset X, composing the set E' takes $O(n^2)$ time and $|E'| < 30n$. It follows that the number of size-6 subsets of E' is $O(n^6)$. Lastly, checking the compatibility of each size-6 subset X' takes $O(1)$ time. Therefore, the time complexity of the operation REPLACE-5-BY-6 is $O(n^{11})$.

In Section 3.4.1.4, we show that the six edges of the target subset X', if exists, must be incident on six out of a set of at most 30 vertices determined by the five edges of X (at most 15 vertices of D^A and at most 15 vertices of D^B, see Observation 3.1). This gives rise to no more than $\binom{30}{6}$ size-6 vertex sets. For each such size-6 vertex set Z, let Z_c denote the subset of vertices each is incident with at most $3 \times 5 - 2 = 13$ edges of E', and let $Z_n = Z - Z_c$. Choosing one edge of E' incident on a vertex of Z_c, we form an edge subset of size $|Z_c|$, and we can prove that if this edge subset is compatible then it can be extended to the target subset X' in $O(n)$ time, through a linear scan on the vertex set D^A and a linear scan on the vertex set D^B. Note that there are at most $13^{|Z_c|} \leq 13^6$ possibilities to check and therefore the target size-6 subset X', if exists, can be found in $O(n)$ time. This way, the time complexity of the operation REPLACE-5-BY-6 is improved to $O(n^6)$.

In the $(2 + \epsilon)$-approximation algorithm by Deduk *et al.* [33], the second phase of local optimization is to iteratively swap a subset of $t - 1$ edges out of the current compatible matching for a subset of t edges (*i.e.*, Replace-$(t - 1)$-By-t in our notation), and the authors show that there would be $O(n^2)$ iterations each of which takes $O(n^{4t})$ time, where $t = \lceil \frac{4}{\epsilon} \rceil + 1$. Using our time analysis (for technical details see ...), one sees that there are only $O(n)$ iterations and each iteration takes only $O(n^t)$ time. Therefore, we are able to lower the time complexity of the $(2 + \epsilon)$-approximation significantly from $O(n^{\frac{16}{\epsilon}+4})$ to $O(n^{\frac{4}{\epsilon}+2})$, though both of them are in $n^{O(1/\epsilon)}$.

3.4.1.2 Operation Reduce-5-by-5

From Equation 3.2, one sees that given a maximal compatible matching M, a pair of parallel edges of M are expected to conflict much less edges outside of M than two singleton edges of M do. This hints that for two compatible matchings of the same cardinality, the one with more parallel edges more likely can be expanded, and motivates the new operation Reduce-5-by-5.

When the operation Replace-5-By-6 fails to expand the current maximal compatible matching M, the operation Reduce-5-by-5 seeks to decrease the number of singleton edges in M, by swapping five edges of M out for five compatible edges. Similarly, it does so by scanning all size-5 subsets of M, and terminates at a successful reduction. If no such reduction is possible, it also terminates but without making any change to the matching M.

Recall that M is partitioned into $p(M)$ and $s(M)$, containing all the parallel edges and all the singleton edges, respectively. Let $X = \{e_1, e_2, \ldots, e_5\}$ be a subset of M (in the special case where $|M| \leq 5$, we seek for a compatible matching of the same size but containing strictly less singleton edges directly by an exhaustive search). The operation composes a set $E' = X \cup C(X)$, where $C(X)$ contains all the edges each conflicting with an edge of X but compatible with $M - X$; it then checks every size-5 subset X' of E' for compatibility and subsequently checks whether $|s(M - X \cup X')| < |s(M)|$, if both affirmative, swaps X out for X' to reduce the number of singleton edges in M.

For the time complexity of the operation Reduce-5-by-5, similarly we recall that $|M| < n$. Partitioning M into $p(M)$ and $s(M)$ takes $O(n^2)$ time. There are $O(n^5)$ size-5 subsets of M. For each such size-5 subset X, composing the set E' takes $O(n^2)$ time and $|E'| < 30n$. It follows that the number of size-5 subsets of E' is $O(n^5)$. Lastly, checking the compatibility of each size-5 subset X' takes $O(1)$ time and counting the singleton edges of $M - X \cup X'$

can be done in $O(n)$ time. Therefore, the time complexity of the operation Reduce-5-By-5 is $O(n^{11})$ too.

In Section 3.4.1.4, we similarly show in technical details that the time complexity of the operation Reduce-5-By-5 can be improved to $O(n^6)$.

3.4.1.3 The local search algorithm $\mathcal{LS}$

Our local search algorithm is iterative. The compatible matching M is initialized to $\emptyset$.

At the beginning of each iteration, we greedily expand the current compatible matching M to the maximal, by adding one edge at a time. Next, with the current maximal compatible matching M, the operation Replace-5-By-6 is applied to expand M. If successful, the iteration ends. Otherwise, M is not modified by the operation Replace-5-By-6 and the operation Reduce-5-By-5 is applied to reduce the number of singleton edges in M. If successful, the iteration ends; otherwise the entire algorithm terminates and returns the current M as the solution.

Clearly, the step of greedy expansion takes $O(n^2)$ time. The running time of the rest of the iteration is $O(n^6)$, which is dominant.

Note that every iteration, except the last, either increases the cardinality of the compatible matching or decreases the number of the singleton edges in the compatible matching. Also, since one would not remove only the middle edge of a set of three parallel edges, removing (adding, respectively) an edge from (to, respectively) the matching would increase the number of the singleton edges in the compatible matching by at most one. Therefore, the operation Replace-5-By-6 would increase the number of the singleton edges in the compatible matching by at most 11. We thus conclude that there are only $O(n)$ iterations in the entire algorithm, which we denote as $\mathcal{LS}$. It follows that the time complexity of the algorithm $\mathcal{LS}$ is $O(n^7)$. We state the above in the following theorem.

Theorem 3.12. *The time complexity of the local search algorithm $\mathcal{LS}$ for the MCBM problem is $O(n^7)$, where n is the number of vertices in one side of the bipartite graph.*

3.4.1.4 A better time complexity analysis for $\mathcal{LS}$

Now we show in technical details how the time complexity of the operations Replace-5-By-6 and Reduce-5-By-5 can be improved to $O(n^6)$, leading to a total time complexity of

$O(n^7)$ for the algorithm $\mathcal{LS}$. We in fact show that the time complexity of both operations REPLACE-ρ-BY-$(\rho+1)$ and REDUCE-ρ-BY-ρ is $O(n^{\rho+1})$, for any integer $\rho \geq 1$.

Given a maximal compatible matching M of $G = (V, E)$ and a subset of edges $X \subseteq M$, with $|X| = \rho$ being a constant, we define some notations as follows:

- $V_0(X) := \bigcup_{e_{i,j} \in X} \{d^A_{i-1}, d^A_i, d^A_{i+1}, d^B_{j-1}, d^B_j, d^B_{j+1}\}$, then any edge in E conflicting with at least one edge of X must be incident on some vertex in $V_0(X)$;
- $C(X) := \bigcup_{e_{i,j} \in X} C(e_{i,j})$;
- $C'(X) \subseteq C(X)$ contains all the edges each compatible with all the edges of $M - X$;
- $G' = (V', E')$ is a subgraph of G with $E' = X \cup C'(X)$ and V' is the union of all the endpoints of edges in E', then all the candidate edges that can be added to $M - X$ to form another compatible matching of G must be from E';
- $V(X) := V_0(X) \cap V'$, then any edge in E' must be incident on some vertex in $V(X)$.

Observe that the sizes of $V_0(X)$ and $V(X)$ are both in $O(1)$, the size of E' is then in $O(n)$. Since M is maximal, E' is just the subset of all the edges each incident on some vertex in $V_0(X)$ while not conflicting with any edge of $M - X$. E' can be found in $O(n)$ time as follows. For each vertex in $V_0(X)$, say d^A_i, we check if there is an edge in $M - X$ incident on d^A_{i-1} or d^A_{i+1} (which takes $O(1)$ time). If there is an edge $e_{i-1,j-1} \in M - X$ (or $e_{i+1,j+1} \in M - X$) and $e_{i,j} \in E$, then we add $e_{i,j}$ to E' (if $e_{i,j} \in E$) and move on to the next vertex in $V_0(X)$; otherwise we add all the edges incident on d^A_i to E'. In total, it takes $O(n)$ time to obtain E'.

For any subset of vertices $U' \subseteq V$, we say a subset F' of $|U'|$ edges is *incident on* U' if each edge of F' is incident on exactly one vertex of U'. In order to find a size-ρ subset of edges $X' \subseteq E'$, with $|X| \leq \rho \leq |E'|$, to obtain another compatible matching $M' = (M - X) \cup X'$ of G, we can traverse through all subsets U's of ρ vertices from $V(X)$, and check if there is a subset of pairwise compatible edges in E' incident on U.

Consider a subset $U \subseteq V(X)$ of ρ vertices, we define

- $U^A := U \cap D^A$, $U^B := U \cap D^B$, then $\rho = |U^A| + |U^B|$;
- $U^A_{i,h} := \{d^A_i, d^A_{i+1}, \ldots, d^A_{i+h-1}\} \subseteq U^A$ (or $U^B_{j,\ell} := \{d^B_j, d^B_{j+1}, \ldots, d^B_{j+\ell-1}\} \subseteq U^B$, resp.) is a subset of maximal subset of consecutive vertices in U^A (or U^B, resp.);
- $E'_{i,j,h} := \{e_{i,j}, e_{i+1,j+1}, \ldots, e_{i+h-1,j+h-1}\} \subseteq E'$ is a size-h subset of consecutive parallel edges;

- $V^A_{E'} := \{d^A_i | e_{i,j} \in E'\}$, $V^B_{E'} := \{d^B_j | e_{i,j} \in E'\}$, and the vertices in $V^A_{E'}$ and $V^B_{E'}$ are kept as $d^A_{i_1}, d^A_{i_2}, \ldots$ and $d^B_{j_1}, d^B_{j_2}, \ldots$, with $i_1, i_2, \ldots$ and $j_1, j_2, \ldots$ in ascending order, respectively.

For the purpose of finding a compatible matching $X' \subseteq E'$ incident on U, we can safely remove from E' all the edges with one endpoint in U^A and the other endpoint in U^B. Observe that for any $U^A_{i,h}$ (or $U^B_{j,\ell}$, resp.), the only possible subset of pairwise compatible edges incident on $U^A_{i,h}$ (or $U^B_{j,\ell}$, resp.) is a subset of consecutive parallel edges $E'_{i,j',h}$ for some j' (or $E'_{i',j,\ell}$ for some i', resp.). Thus, X' may exist in E' only if there is at least one subset of consecutive parallel edges in E' incident on $U^A_{i,h}$ and $U^B_{j,\ell}$, for $\forall U^A_{i,h} \subseteq U^A$ and $\forall U^B_{j,\ell} \subseteq U^B$, respectively. For simplicity, we also call a maximal subset of consecutive parallel edges in X' as a *streak* of X'. Then, any X' can be partitioned into two subsets X'_1 and X'_2, where X'_1 is the union of streaks of X' each incident on a single $U^A_{i,h}$ or a single $U^B_{j,\ell}$ and X'_2 is the union of streaks of X' each incident on a combination of at least one $U^A_{i,h}$ and at least one $U^B_{j,\ell}$. Correspondingly, U can also be partitioned into two subsets U_1 and U_2 such that X'_1 is incident on U_1 and X'_2 is incident on U_2.

Consider any pair of $U^A_{i,h}$ and $U^B_{j,\ell}$, we observe that there are two possible cases in which $E'_{i,j',h} \cup E'_{i',j,\ell}$ form a size-$(h+\ell)$ subset of consecutive parallel edges incident on $U^A_{i,h} \cup U^B_{j,\ell}$:

- $i' = i + h$ and $j' = j - h$: define $X'_{i,h;j,\ell} = E'_{i,j',h} \cup E'_{i',j,\ell} = E'_{i,j-h,h+\ell}$;
- $i' = i - \ell$ and $j' = j + \ell$: define $X'_{j,\ell;i,h} = E'_{i',j,\ell} \cup E'_{i,j',h} = E'_{i-\ell,j,\ell+h}$.

Define $Y = \{X'_{i,h;j,\ell}, X'_{j,\ell;i,h} \subseteq E' | U^A_{i,h} \subseteq U^A, U^B_{j,\ell} \subseteq U^B\}$, then X'_2 must be the union of all the $X'_{\cdot,\cdot;\cdot,\cdot}$'s in some subset of Y. Since the size of Y is a constant due to $|U|$ being a constant, we can find all possible X'_2 by traversing through all subsets of Y in $O(1)$ time. Then X'_1 will be a subset of pairwise compatible edges incident on U_1 in $E'_1 = E' - \left(X'_2 \cup C(X'_2)\right)$.

Define $U^A_{1adj} = \bigcup_{d^A_i \in U^A_1} \{d^A_{i-1}, d^A_{i+1}\} \cap V^A_{E'_1}$ and $U^B_{1adj} = \bigcup_{d^B_j \in U^B_1} \{d^B_{j-1}, d^B_{j+1}\} \cap V^B_{E'_1}$. For the purpose of finding a compatible matching $X'_1 \subseteq E'_1$ incident on U_1, we can safely remove from E'_1 all the edges $e_{i,j}$ such that $d^A_i \in U^A_1$ and $d^B_j \in U^B_{1adj}$ OR $d^A_i \in U^A_{1adj}$ and $d^B_j \in U^B_1$. Then in the remaining edges in E'_1, any edge incident on a vertex in U^A_1 is compatible with any edge incident on a vertex of U^B_1. Thus, if there is a subset X'_{1A} of pairwise compatible edges incident on U^A_1 and a subset X'_{1B} of pairwise compatible edges incident on U^B_1, then $X'_{1A} \cup X'_{1B}$ must also be a compatible matching in E'_1.

For each $U^A_{i,h} \subseteq U^A_1$ and $U^B_{j,\ell} \subseteq U^B_1$., we first find all the subsets of consecutive parallel edges $E'_{i,j',h}$ incident on $U^A_{i,h}$ and $E'_{i',j,\ell}$ incident on $U^B_{j,\ell}$, respectively. Let $E^* = \bigcup_{\forall U^A_{i,h}, U^B_{j,h} \subseteq U} E'_{i,j,h}$.

Then, the edges of a compatible matching $X'_1 \subseteq E'_1$ must be all from E^*. Define $deg(u)$ as the number of edges in E^* incident on u for each $u \in U_1$. X' may exist only if $deg(u) \geq 1$ for each $u \in U_1$. We will determine a constant c, and partition U_1 into two subsets $U_{\geq c} = \{u \in U_1 | deg(u) \geq c\}$ and $U_{<c} = U_1 - U_{\geq c}$. For the vertices in $U_{<c}$, we can traverse through all $O(1)$ combinations of edges incident on $U_{<c}$ to find possible subset of pairwise compatible edges, say X'_0. Let $E^*_1 = E^* - \big(X'_0 \cup C(X'_0)\big)$. From the following lemma which will be proved later, we can conclude that we can always find a compatible matching X'_A incident on $U^A_{\geq c}$ (X'_B incident on $U^B_{\geq c}$, resp.) in linear time, if $deg(u) \geq 2|U^A_{\geq c}| - 1$ ($deg(u) \geq 2|U^B_{\geq c}| - 1$, resp.) in E^*_1, for $\forall u \in U^A_{\geq c}$ ($\forall u \in U^B_{\geq c}$, resp.). Then $X'_A \cup X'_B$ is a compatible matching in E^*_1. Together with X'_0, $X'_1 = X'_0 \cup X'_A \cup X'_B$ will be a compatible matching in E'_1. See function FindCompatibleEdgesAtU for a high-level description of the algorithm of finding an X'_1 in E'_1.

Lemma 3.13. *Given a subgraph $G^* = (V^A, V^B, E^*)$ of G, a subset of vertices $U^A \subseteq V^A$, with $|U^A|$ being a constant, we can always find a compatible matching incident on U^A in linear time, if $deg(u) \geq 2|U^A| - 1$ in G^* for $\forall u \in U^A$.*

Because $|U_1|$ is constant, each step of performing function FindCompatibleEdgesAtU on E'_1, U_1 can either be done in $O(1)$ time or in $O(n)$ time. Since the two functions FindCompatibleEdgesAtU^A and FindCompatibleEdgesAtU^B can both be done in linear time, function FindCompatibleEdgesAtU can be done in $O(n)$ time.

Now let us determine c. Consider a vertex $u \in U^A_{\geq c}$. For the $|U^A_{<c}|$ edges incident on $U^A_{<c}$ added into X'_0, there are at most $3|U^A_{<c}|$ edges incident on d^A_i which are conflicting with them and have been removed from E^*. Recall that in E^*, any edge incident on a vertex in U^A_1 is compatible with any edge incident on a vertex of U^B_1. Therefore, together with Lemma 3.1, in order to always find a size-$|U^A_{\geq c}|$ compatible matching from E^*_1, we must have $deg(u) \geq 2|U^A_{\geq c}| - 1 + 3|U^A_{<c}| = |U^A_{<c}| + 2|U_1| - 1$. Since E^*_1 is not empty only if $|U_{<c}| \leq |U_1| - 1$, we can set $c = 3|U_1| - 2$.

Proof. (of Lemma 3.13)
Check each vertex $d^B_j \in V^B_{E^*}$ from the smallest j to the largest j. We arbitrarily select an edge $e_{i,j} \in E^*$, with $d^A_i \in U^A$, and find the maximal subset of consecutive vertices $U^A_{i,h} \subseteq U^A$, then add the corresponding h consecutive parallel edges $e_{i,j}, e_{i+1,j+1}, \ldots, e_{i+h-1,j+h-1}$ to the solution matching, say X'_A. We move to the next vertex $d^B_{j'} \in V^B_{E^*}$, if $j' = j + h$, that is, $d^B_{j'}$ is next to d^B_{j+h-1}, then any edge $e_{i',j'}$ with $d^A_{i'} \in U^A$ must be conflicting with an edge in the current X'_A. While any edge $e_{i',j'}$ with $j' \geq j + h + 1$ and $d^A_{i'} \in U^A - \{d^A_i, \ldots, d^A_{i+h-1}\}$ must

```
FindCompatibleEdgesAtU($E', U = U^A \cup U^B$)
 1: $E' \leftarrow E' -$ {edges $e_{i,j}$ such that $d_i^A \in U^A$ and $d_j^B \in U^B_{adj}$ OR $d_i^A \in U^A_{adj}$ and $d_j^B \in U^B$}
 2: for each maximal subset of consecutive vertices $U^A_{i,h} \subseteq U^A$ (or $U^B_{j,\ell} \subseteq U^B$, resp.) do
 3:     find all subsets of consecutive parallel edges $E'_{i,j',h}$ (or $E'_{i',j,\ell}$, resp.) in $E'$ incident
        on $U^A_{i,h}$ (or $U^B_{j,\ell}$, resp.)
 4: end for
 5: $E^* \leftarrow \bigcup_{\forall U^A_{i,h}, U^B_{j,h} \subseteq U} E'_{i,j,h}$
 6: $deg(u) \leftarrow$ the number of edges in $E^*$ incident on $u$ for each $u \in U$
 7: $X' \leftarrow \emptyset$
 8: if $deg(u) \geq 1$ for all $u \in U$ then
 9:     $c \leftarrow 3|U| - 2$
10:     $U_{<c} \leftarrow$ the subset of vertices in $U$ with $deg(u) < c$ for $u \in U_{<c}$
11:     $U_{\geq c} \leftarrow U - U_{<c}$
12:     if $|U_{<c}| = 0$ then
13:         $X'_A \leftarrow$ FindCompatibleEdgesAt$U^A$($E^*, U$)                          ▹ $O(n)$
14:         $X'_B \leftarrow$ FindCompatibleEdgesAt$U^B$($E^*, U$)                          ▹ $O(n)$
15:         $X' \leftarrow X'_A \cup X'_B$
16:     else
17:         for each subset $X'_0 \subseteq E^*$ of $|U_{<c}|$ edges in incident on $U_{<c}$ do ▹ $O(1)$ iterations
18:             if the edges in $X'_0$ are pairwise compatible then             ▹ checked in $O(1)$
19:                 $E^*_1 \leftarrow E^* - \left(X'_0 \cup C(X'_0)\right)$
20:                 $X'_A \leftarrow$ FindCompatibleEdgesAt$U^A$($E^*_1, U_{\geq c}$)              ▹ $O(n)$
21:                 $X'_B \leftarrow$ FindCompatibleEdgesAt$U^B$($E^*_1, U_{\geq c}$)              ▹ $O(n)$
22:                 $X' \leftarrow X'_0 \cup X'_A \cup X'_B$
23:             end if
24:         end for
25:     end if
26: end if
27: return $X'$
```

Figure 3.12: A high-level description of the algorithm of finding an X'_1 in E'_1.

be compatible with all the edges in the current X'_A. Thus, we can consider the vertex $d^B_{j'}$ with the smallest $j' \geq j + h + 1$, and repeat the procedure until $|X'_A| = |U^A|$.

We observe that every time we come to a vertex $d^B_j \in V^B_{E^*}$ and select an edge incident on d^B_j to X'_A, there is at most one vertex, just the one next to d^B_j, cannot be considered to have an edge compatible to the current X'_A. Therefore, if $deg(u) \geq 2|U^A| - 1$ for each vertex $u \in U^A$, *i.e.*, there are at least $2|U^A| - 1$ edges in E^* incident on u, then we can always find a size-$|U^A|$ compatible matching in G^*.

See function FindCompatibleEdgesAtU^A for a high-level description of the algorithm of finding a compatible matching of G^* incident on U^A. Since $|V^B_{E^*}| \leq |V^B|$ is in $O(n)$, there are $O(n)$ iterations, and each iteration can be done in $O(1)$ time, thus X'_A can be found in linear time. □

```
FindCompatibleEdgesAtU^A($E^*, U = U^A \cup U^B$)
 1: select any edge $e_{i,j_1} \in E^*$ with $d_i^A \in U^A$ and $d_{j_1}^B \in V_{E^*}^B - U_{adj}^B$
 2: find the maximal subset of consecutive vertices $U_{i,h}^A \subseteq U^A$
 3: $X' \leftarrow \{e_{i,j_1}, e_{i+1,j_1+1}, \ldots, e_{i+h-1,j_1+h-1}\}$
 4: $U_r^A \leftarrow U^A - \{d_i^A, \ldots, d_{i+h-1}^A\}$
 5: $\ell \leftarrow h + 1$
 6: while $d_{j_\ell}^B \in V_{E^*}^B$ do                              ▷ $O(n)$ iterations
 7:     if $j_\ell \geq j_h + 2$ and $d_{j_\ell}^B \notin U_{adj}^B$ then
 8:         select any edge $e_{i',j_\ell} \in E^*$ with $d_{i'}^A \in U_r^A$
 9:         find the maximal subset of consecutive vertices $U_{i,h}^A \subseteq U_r^A$
10:         $X' \leftarrow X' \cup \{e_{i',j_\ell}, e_{i'+1,j_\ell+1}, \ldots, e_{i'+h-1,j_\ell+h-1}\}$
11:         $U_r^A \leftarrow U_r^A - \{d_{i'}^A, \ldots, d_{i'+h-1}^A\}$
12:         $\ell \leftarrow \ell + h$
13:     else
14:         $\ell \leftarrow \ell + 1$
15:     end if
16: end while
17: return $X'$
```

Figure 3.13: A high-level description of the algorithm of finding a compatible matching of G^* incident on U^A.

The function FindCompatibleEdgesAtU^B is the same as FindCompatibleEdgesAtU^A except that every superscript "^{A}" is replaced by "^{B}" and every superscript "^{B}" is replaced by "^{A}".

See operation Replace-ρ-by-(ρ+1) for a high-level description of the algorithm of expanding the current maximal compatible matching by swapping ρ edges out for $\rho + 1$ compatible edges.
In operation Replace-ρ-by-($\rho + 1$), there are $O(n^\rho)$ iterations. Line 5 can be done in $O(n)$ time: all the maximal subsets of consecutive vertices can be determined in $O(1)$ time, and for each maximal subset of consecutive vertices, let d_i^A (or d_j^B) be the vertex in U_ℓ with i being the smallest subscript and $q = |U_\ell|$, then we only need to check if there are consecutive parallel edges $e_{i+1,j+1}, \ldots e_{i+q-1,j+q-1} \in E'$ for each edge $e_{i,j}$ incident on d_i^A. The other steps can also be done in $O(n)$ time, thus the time complexity of operation Replace-ρ-by-($\rho + 1$) is $O(n^{\rho+1})$.

The operation Reduce-ρ-by-ρ is almost the same as the operation Replace-ρ-by-($\rho + 1$) except the following:

- in line 3, the size of U is ρ instead of $\rho + 1$;
- before each of the lines 17, 29, and 35, we need to check if $(M - X) \cup X'$ has strictly less singleton edges, and if it has, then we swap X out by X'.

In order to compare the number of singleton edges in M and $M' = (M - X) \cup X'$, we only need to find the edges incident on vertices of $V_0(X) \cup V_0(X')$ in M and M', and compare those two subsets of edges to see if the number of singleton edges from M' is strictly less. Since the size of $V_0(X) \cup V_0(X')$ must be constant and whether an edge in a singleton or not can also be checked in $O(1)$ time, this additional condition can be checked in $O(1)$ time. Therefore, the time complexity of the operation Reduce-ρ-by-ρ is also $O(n^{\rho+1})$.

Observe that in the operation Replace-ρ-by-$(\rho + 1)$, for each subset of ρ edges in M, we traverse through all possible combinations of $\rho + 1$ edges in $M - X$ except for the vertices with relatively large degrees, which are the cases when we are trying to find a compatible matching X'_1 incident on U_1 in E'_1. Recall that X'_1 is a union of streaks of X' each incident on a single $U^A_{i,h}$ or a single $U^B_{j,\ell}$. Thus, when all the vertices in $U^A_{i,h}$ or $U^B_{j,\ell}$ have large degrees, no edge in X'_1 can be parallel with an edge from $M - X$; no matter which sequence of consecutive parallel edges incident on $U^A_{i,h}$ or $U^B_{j,\ell}$ is selected to X'_1, the numbers of singleton edges in X' are always the same. Therefore, the operation Reduce-ρ-by-ρ will not miss any possible combination of X' which will lead to an improved compatible matching $M' = (M - X) \cup X'$ with strictly less singleton edges.

Together with the first step of greedy expansion which takes $O(n^2)$ time, when we set $\rho = 5$, the time complexity of algorithm $\mathcal{LS}$ is $O(n^7)$.

3.4.2 Approximation ratio analysis for the algorithm $\mathcal{LS}$

We analyze the performance ratio of the algorithm $\mathcal{LS}$ through *amortization*. The main result is to prove that the algorithm $\mathcal{LS}$ is a 35/12-approximation for the MCBM problem, and thus it is also a 35/12-approximation for the Max-Duo problem.

3.4.2.1 The amortization scheme

Let M^* be the optimal compatible matching to the MCBM problem and OPT $= |M^*|$, and M be the maximal compatible matching returned by the algorithm $\mathcal{LS}$ and SOL $= |M|$. We partition M into $s(M)$ and $p(M)$. (In the sequel, notations with a superscript * are associated with M^*; notations without a superscript are associated with M. In general, the subscript of a vertex of D^A has an i or h, and the subscript of a vertex of D^B has a j or ℓ.)

In the amortization scheme, we assign one token to each edge $e^* \in M^*$, and thus the total amount of tokens is OPT. The edge e^* will be conflicting to a number of edges of M (including the case where e^* is in M, then e^* is conflicting to itself only); it then splits the token evenly and distributes a fraction to every conflicting edge of M. To the end, the total amount of tokens received by all the edges of M is exactly OPT. Our main task is to estimate an upper bound (which is expected to be $35/12$) on the amount of tokens received by an edge of M, thereby to give a lower bound on SOL.

Formally, we define the function $\tau(e \leftarrow e^*) \geq 0$ to be the amount of token $e^* \in M^*$ gives to $e \in M$. For the edge $e^* \in M^*$, let $C(e^*) \subseteq M$ be the subset of edges of M conflicting with e^*, and for the edge $e \in M$, let $C^*(e) \subseteq M^*$ be the subset of edges of M^* conflicting with e. From the maximality, we know that both $|C(e^*)|, |C^*(e)| \geq 1$, for any e^*, e. Then, $\tau(e \leftarrow e^*) = \frac{1}{|C(e^*)|}$, if $e \in C(e^*)$; or otherwise $\tau(e \leftarrow e^*) = 0$. The total amount of tokens $e \in M$ receives is denoted as

$$\omega(e) := \sum_{e^* \in C^*(e)} \frac{1}{|C(e^*)|}, \forall e \in M. \tag{3.3}$$

And we have $\text{OPT} = \sum_{e \in M} \omega(e) \leq \max_{e \in M} \omega(e) \cdot \text{SOL}$.

Therefore, the quantity $\max_{e \in M} \omega(e)$ is an upper bound on the performance ratio of the algorithm $\mathcal{LS}$. We thus aim to estimate $\max_{e \in M} \omega(e)$. In the following, we will see that $\max_{e \in M} \omega(e) = 10/3$, which is larger than our target ratio $35/12$. We then switch to enumerate all possible cases where an edge e has $\omega(e) \geq 3$ and amortize some fraction of its token to certain provably existing edges e' with $\omega(e') < 3$. In other words, we will estimate the average value of $\omega(\cdot)$ for all the edges of M, denoted as $\overline{\omega(e)}$, and prove an upper bound (which is shown to be $35/12$) on $\overline{\omega(e)}$ that is also an upper bound on the performance ratio of the algorithm $\mathcal{LS}$.

To this purpose, we may assume without loss of generality that $M \cap M^* = \emptyset$ since their $\omega(\cdot)$'s are all 1. According to Observation 3.1 in Section 3.2, we have $|C^*(e)| \leq 6$ and $|C(e^*)| \leq 6$ for any $e \in M$ and $e^* \in M^*$. Consider an arbitrary edge $e_{i,j} \in M$, we have

$$C^*(e_{i,j}) = \{e^*_{i-1,j''-1}, e^*_{i,j'}, e^*_{i+1,j'''+1}, e^*_{i''-1,j-1}, e^*_{i',j}, e^*_{i'''+1,j+1}\},$$

where $e^*_{i,j'}$ ($e^*_{i-1,j''-1}$, $e^*_{i+1,j'''+1}$, respectively) denotes the edge of M^* incident on d^A_i (d^A_{i-1}, d^A_{i+1}, respectively), if it exists, or otherwise it is a void edge; $e^*_{i',j}$ ($e^*_{i''-1,j-1}$, $e^*_{i'''+1,j+1}$, respectively) denotes the edge of M^* incident on d^B_j (d^B_{j-1}, d^B_{j+1}, respectively), if it exists, or otherwise it is a void edge; and none of i', i'', i''' can be i and none of j', j'', j''' can be j. (It is important to point out that $C^*(e_{i,j})$ does not necessarily contain 6 edges, due to the possible void edges.) We partition $C^*(e_{i,j})$ into two parts $C^*(e_{i,\cdot})$ and $C^*(e_{\cdot,j})$:

$$C^*(e_{i,\cdot}) = \{e^*_{i-1,j''-1}, e^*_{i,j'}, e^*_{i+1,j'''+1}\} \text{ and } C^*(e_{\cdot,j}) = \{e^*_{i''-1,j-1}, e^*_{i',j}, e^*_{i'''+1,j+1}\}.$$

(Again, each of $C^*(e_{i,\cdot})$ and $C^*(e_{\cdot,j})$ does not necessarily contain 3 edges, due to the possible void edges.) We extend the function notation to let $\tau(e_{i,j} \leftarrow C^*(e_{i,j}))$ be the multi-set of the $\tau(e_{i,j} \leftarrow e^*)$ values, where $e^* \in C^*(e_{i,j})$, that is,

$$\begin{aligned}
\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) &= \left\{ \frac{1}{|C(e^*)|} \;\middle|\; e^* \in C^*(e_{i,\cdot}) \right\},\\
\tau(e_{i,j} \leftarrow C^*(e_{\cdot,j})) &= \left\{ \frac{1}{|C(e^*)|} \;\middle|\; e^* \in C^*(e_{\cdot,j}) \right\},\\
\tau(e_{i,j} \leftarrow C^*(e_{i,j})) &= \tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) \cup \tau(e_{i,j} \leftarrow C^*(e_{\cdot,j})).
\end{aligned}$$

Then $\omega(e_{i,j})$ is the sum of all the (at most six) values in the set $\tau(e_{i,j} \leftarrow C^*(e_{i,j}))$; each of these values can be any of $1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \frac{1}{6}$, since $1 \le |C(e^*)| \le 6$ for any $e^* \in C^*(e_{i,j})$. We need the following three more subsets of M, all of which are associated with $e_{i,j} \in M$.

$$\begin{aligned}
C(C^*(e_{i,\cdot})) &= \bigcup_{e^* \in C^*(e_{i,\cdot})} C(e^*),\\
C(C^*(e_{\cdot,j})) &= \bigcup_{e^* \in C^*(e_{\cdot,j})} C(e^*),\\
C(C^*(e_{i,j})) &= C(C^*(e_{i,\cdot})) \cup C(C^*(e_{\cdot,j})).
\end{aligned}$$

3.4.2.2 Value combinations of $\tau(e_{i,j} \leftarrow C^*(e_{i,j}))$ with $\omega(e_{i,j}) \ge 3$

Note that the operation Replace-5-by-6 actually executes swapping p edges of the current compatible matching out for $p+1$ compatible edges to expand the matching, for $p = 1, 2, 3, 4, 5$. Therefore, for any edge $e_{i,j} \in M$, we can never have two edges $e^*_{i_1,j_1}, e^*_{i_2,j_2} \in C^*(e_{i,j})$ such that $|C(e^*_{i_1,j_1})| = |C(e^*_{i_2,j_2})| = 1$, that is, both of them conflict with only the edge $e_{i,j}$ in M. Thus we immediately have the following lemma, which has also been observed in [9].

Lemma 3.14. [9] *For any edge $e_{i,j} \in M$, there is at most one edge $e^*_{i_1,j_1} \in C^*(e_{i,j})$ such that $|C(e^*_{i_1,j_1})| = 1$.*

Lemma 3.15. *For any edge $e_{i,j} \in M$, and for any pair of parallel edges $e^*_{i_1,j_1}, e^*_{i_1+1,j_1+1} \in C^*(e_{i,j})$, $||C(e^*_{i_1,j_1})| - |C(e^*_{i_1+1,j_1+1})|| \le 2$.*

Proof. Since the edges of $C(e^*_{i_1,j_1}) \cup C(e^*_{i_1+1,j_1+1}) \subseteq M$ are pairwise compatible, we have $C(e^*_{i_1,j_1}) - C(e^*_{i_1+1,j_1+1}) \subseteq \{e_{i_1-1,\diamond}, e_{\diamond,j_1-1}\}$ and $C(e^*_{i_1+1,j_1+1}) - C(e^*_{i_1,j_1}) \subseteq \{e_{i_1+2,\diamond}, e_{\diamond,j_1+2}\}$, where $e_{i_1-1,\diamond}$ ($e_{\diamond,j_1-1}, e_{i_1+2,\diamond}, e_{\diamond,j_1+2}$, respectively) denotes the edge of M incident on $d^A_{i_1-1}$ ($d^B_{j_1-1}, d^A_{i_1+2}, d^B_{j_1+2}$, respectively), if it exists, or otherwise it is a void edge. Thus, $|C(e^*_{i_1,j_1}) - C(e^*_{i_1+1,j_1+1})| \le 2$ and $|C(e^*_{i_1+1,j_1+1}) - C(e^*_{i_1,j_1})| \le 2$, which together imply $||C(e^*_{i_1,j_1})| - |C(e^*_{i_1+1,j_1+1})|| \le 2$. □

Lemma 3.16. *Suppose $|C^*(e_{i,\cdot})| = 3$, then $C^*(e_{i,\cdot}) = \{e^*_{i-1,j'-1}, e^*_{i,j'}, e^*_{i+1,j'+1}\}$ for some $j' \neq j$. In this case we can never have $|C(e^*_{i-1,j'-1})| = |C(e^*_{i,j'})| = |C(e^*_{i+1,j'+1})| = 2$, if one of the following three conditions holds:*

1. *there is an edge $e^*_{i_1,j_1} \in C^*(e_{\cdot,j})$ such that $|C(e^*_{i_1,j_1})| = 1$;*
2. *$|C(C^*(e_{\cdot,j}))| \le |C^*(e_{\cdot,j})|$;*
3. *there is at least one singleton edge of M in $C(C^*(e_{i,\cdot}))$.*

Proof. (of Lemma 3.16) Recall that $C^*(e_{i,\cdot}) = \{e^*_{i-1,j''-1}, e^*_{i,j''}, e^*_{i+1,j'''+1}\}$ for some $j', j'', j'''(\neq j)$. When all these three edges of M^* exist, they are consecutive parallel edges, that is, $j' = j'' = j'''$ and thus $C^*(e_{i,\cdot}) = \{e^*_{i-1,j'-1}, e^*_{i,j'}, e^*_{i+1,j'+1}\}$ for some $j' \neq j$. This proves the first half of the lemma.

Next, assume $|C(e^*_{i-1,j'-1})| = |C(e^*_{i,j'})| = |C(e^*_{i+1,j'+1})| = 2$, and we will show none of the three conditions holds.

Since $e_{i,j} \in C(e^*_{i-1,j'-1}) \cap C(e^*_{i,j'}) \cap C(e^*_{i+1,j'+1})$, each of $C(e^*_{i-1,j'-1}), C(e^*_{i,j'}), C(e^*_{i+1,j'+1})$ contains exactly one edge other than $e_{i,j}$. Observe that any edge of M conflicting with $e^*_{i,j'}$ must be conflicting with either $e^*_{i-1,j'-1}$ or $e^*_{i+1,j'+1}$. We conclude that either $C(e^*_{i-1,j'-1}) = C(e^*_{i,j'})$ or $C(e^*_{i+1,j'+1}) = C(e^*_{i,j'})$, implying that $2 \le |C(C^*(e_{i,\cdot}))| \le 3$.

If $|C(C^*(e_{i,\cdot}))| = 2$, then the algorithm $\mathcal{LS}$ would have replaced these two edges of $C(C^*(e_{i,\cdot}))$ by the three edges of $C^*(e_{i,\cdot})$, contradicting to the fact that M is the solution by $\mathcal{LS}$. Therefore, $|C(C^*(e_{i,\cdot}))| = 3$.

If the first condition holds, the algorithm $\mathcal{LS}$ would have replaced the three edges of $C(C^*(e_{i,\cdot}))$ by the edge $e^*_{i_1,j_1}$ and the three edges of $C^*(e_{i,\cdot})$ to expand M, again a contradiction.

If the second condition holds, we have $|C(C^*(e_{i,j}))| \le |C(C^*(e_{i,\cdot}))| + |C(C^*(e_{\cdot,j}))| - 1 \le 2 + |C^*(e_{\cdot,j})| < |C^*(e_{i,j})| \le 6$. Then, the algorithm $\mathcal{LS}$ would have replaced all the edges of $C(C^*(e_{i,j}))$ by all the edges of $C^*(e_{i,j})$ to expand M, also a contradiction.

When there is at least one singleton edge of M in $C(C^*(e_{i,\cdot}))$, we distinguish two cases where $e_{i,j}$ is singleton or not. If $e_{i,j}$ is not a singleton, then we may assume the edge $e_{i+1,j+1} \in M$ and thus $e_{i+1,j+1} \in C(C^*(e_{i,\cdot}))$ too; it follows from $|C(e^*_{i-1,j'-1})| = |C(e^*_{i,j'})| = |C(e^*_{i+1,j'+1})| = 2$ that these two edges form an isolated pair of parallel edges in M. In this case, the algorithm $\mathcal{LS}$ would have replaced the three edges in $C(C^*(e_{i,\cdot}))$ by the three parallel edges of $C^*(e_{i,\cdot})$ to decrease the number of singleton edges by at least one, a contradiction. If $e_{i,j}$ is a singleton, then the other edge conflicting with $e^*_{i,j'}$ must also be a singleton. The algorithm $\mathcal{LS}$ would still have replaced the three edges in $C(C^*(e_{i,\cdot}))$ by the three parallel edges of $C^*(e_{i,\cdot})$ to decrease the number of singleton edges by at least one, again a contradiction.

In summary, we conclude that none of the three conditions would hold. This proves the second half of the lemma. □

For an edge $e_{i,j} \in M$ with $\omega(e_{i,j}) \ge 3$, we can now characterize the multi-set $\tau(e_{i,j} \leftarrow C^*(e_{i,j}))$ of six values, in which an entry of 0 represents a void edge in $C^*(e_{i,j})$. We arrange these six values in a non-increasing order. Using the above three Lemmas 3.14–3.16, we have the following conclusion:

Lemma 3.17. *For an edge $e_{i,j} \in M$ with $\omega(e_{i,j}) \ge 3$, there are 8 possible value combinations of $\tau(e_{i,j} \leftarrow C^*(e_{i,j}))$, which are* $\left\{1, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{3}\right\}$, $\left\{1, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}\right\}$, $\left\{1, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{3}, \frac{1}{3}\right\}$, $\left\{1, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}\right\}$, $\left\{1, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{3}, \frac{1}{5}\right\}$, $\left\{1, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right\}$, $\left\{1, \frac{1}{2}, \frac{1}{2}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right\}$, *and* $\left\{1, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 0\right\}$. *These combinations give rise to* $\omega(e_{i,j}) = \frac{10}{3}, \frac{13}{4}, \frac{19}{6}, \frac{37}{12}, \frac{91}{30}, 3, 3$ *and* 3 *respectively.*

We remark that in Lemma 3.17, $|C^*(e_{i,j})| = 6$ except for the last combination where $|C^*(e_{i,j})| = 5$. Also, we see that $\max_{e \in M} \omega(e) \le 10/3$, implying that the algorithm $\mathcal{LS}$ is a 10/3-approximation. (This is worse than the previous 3.25-approximation though.)

3.4.2.3 Ordered value combinations of $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot}))$ with $\omega(e_{i,j}) \ge 3$

To find a good upper bound on the average value of $\omega(\cdot)$ for all the edges of M, we consider all the possible combinations of edges in $C(C^*(e_{i,\cdot}))$ with $\omega(e_{i,j}) \ge 3$. The goal is to show that every edge of $C(C^*(e_{i,\cdot}))$ other than $e_{i,j}$ has its $\omega(\cdot) \leq 3$, and there must be some *accompanying* edges with $\omega(\cdot) \leq 2.5$. This way, we are able to "*move*" a fraction of token received by the edge $e_{i,j}$ to these *accompanying* edges.

We discuss the possible ordered value combinations of $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot}))$ in this section. The discussion holds for $\tau(e_{i,j} \leftarrow C^*(e_{\cdot,j}))$ too.

We use the following vectors to represent the *ordered values* of $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot}))$ and $\tau(e_{i,j} \leftarrow C^*(e_{\cdot,j}))$, respectively:

$$\big(\tau(e_{i,j} \leftarrow e^*_{i-1,j''-1}), \tau(e_{i,j} \leftarrow e^*_{i,j'}), \tau(e_{i,j} \leftarrow e^*_{i+1,j'''+1})\big),$$

$$\big(\tau(e_{i,j} \leftarrow e^*_{i''-1,j-1}), \tau(e_{i,j} \leftarrow e^*_{i',j}), \tau(e_{i,j} \leftarrow e^*_{i'''+1,j+1})\big).$$

Using the first condition of Lemma 3.16, we can rule out $\left\{\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right\}$ for $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot}))$. From the 8 possible value combinations of $\tau(e_{i,j} \leftarrow C^*(e_{i,j}))$ stated in Lemma 3.17, by Lemma 3.15 we can identify in total 12 possible value combinations of $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot}))$ with $\omega(e_{i,j}) \geq 3$, stated in the following lemma.

Lemma 3.18. *For an edge $e_{i,j} \in M$ with $\omega(e_{i,j}) \geq 3$, there are* 12 *possible value combinations of $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot}))$, which are $\left\{1, \frac{1}{2}, \frac{1}{2}\right\}$, $\left\{1, \frac{1}{2}, \frac{1}{3}\right\}$, $\left\{1, \frac{1}{2}, \frac{1}{4}\right\}$, $\left\{1, \frac{1}{3}, \frac{1}{3}\right\}$, $\left\{\frac{1}{2}, \frac{1}{2}, \frac{1}{3}\right\}$, $\left\{\frac{1}{2}, \frac{1}{2}, \frac{1}{4}\right\}$, $\left\{\frac{1}{2}, \frac{1}{3}, \frac{1}{3}\right\}$, $\left\{\frac{1}{2}, \frac{1}{3}, \frac{1}{4}\right\}$, $\left\{\frac{1}{2}, \frac{1}{3}, \frac{1}{5}\right\}$, $\left\{\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right\}$, $\left\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right\}$, and $\left\{\frac{1}{2}, \frac{1}{2}, 0\right\}$.*

Lemma 3.19. *Suppose $|C^*(e_{i,\cdot})| = 3$ and $C^*(e_{i,\cdot}) = \{e^*_{i-1,j'-1}, e^*_{i,j'}, e^*_{i+1,j'+1}\}$ for some $j' \neq j$. We have*

$$C(e^*_{i,j'}) \subseteq C(e^*_{i-1,j'-1}) \cup C(e^*_{i+1,j'+1}), \tag{3.4}$$

$$|C(C^*(e_{i,\cdot}))| \leq |C(e^*_{i-1,j'-1})| + |C(e^*_{i+1,j'+1})| - 1, \tag{3.5}$$

$$|C(C^*(e_{i,\cdot}))| \geq \max \begin{cases} 3, \\ |C(e^*_{i-1,j'-1})| + |C(e^*_{i+1,j'+1})| - 2, \\ |C(e^*_{i-1,j'-1})| + |C(e^*_{i+1,j'+1})| - |C(e^*_{i,j'})|. \end{cases} \tag{3.6}$$

Proof. Observe that any edge of M conflicting with $e^*_{i,j'}$ must also conflict with either $e^*_{i-1,j'-1}$ or $e^*_{i+1,j'+1}$. We have $C(e^*_{i,j'}) \subseteq C(e^*_{i-1,j'-1}) \cup C(e^*_{i+1,j'+1})$, which proves the inequality (3.4) and also indicates that $C(C^*(e_{i,\cdot})) = C(e^*_{i-1,j'-1}) \cup C(e^*_{i+1,j'+1})$. Since $e_{i,j} \in C(e^*_{i-1,j'-1}) \cap C(e^*_{i,j'}) \cap C(e^*_{i+1,j'+1}) \subseteq \{e_{i,j}, e_{\diamond,j'}\}$, where $e_{\diamond,j'}$ is a possible edge of M incident on $d^B_{j'}$, we have

$$|C(e^*_{i-1,j'-1})| + |C(e^*_{i+1,j'+1})| - 2 \leq |C(C^*(e_{i,\cdot}))| \leq |C(e^*_{i-1,j'-1})| + |C(e^*_{i+1,j'+1})| - 1.$$

This proves the inequality (3.5) and the second inequality in (3.6).

Also observe that any edge of M conflicting with both $e^*_{i-1,j'-1}$ and $e^*_{i+1,j'+1}$ must conflict with $e^*_{i,j'}$ too. We have $C(e^*_{i-1,j'-1}) \cap C(e^*_{i+1,j'+1}) \subseteq C(e^*_{i,j'})$. Therefore,

$$|C(C^*(e_{i,\cdot}))| \geq |C(e^*_{i-1,j'-1})| + |C(e^*_{i+1,j'+1})| - |C(e^*_{i,j'})|,$$

proving the last inequality in (3.6). $|C(C^*(e_{i,\cdot}))| \geq 3$ can be proven by a simple contradiction, since otherwise the algorithm $\mathcal{LS}$ would replace all the edges of $C(C^*(e_{i,\cdot}))$ by the three edges of $C^*(e_{i,\cdot})$ to expand M. □

Lemma 3.20. *Suppose $|C^*(e_{i,\cdot})| = 3$ and $C^*(e_{i,\cdot}) = \{e^*_{i-1,j'-1},\ e^*_{i,j'},\ e^*_{i+1,j'+1}\}$ for some $j' \neq j$, and there is an edge $e^*_{i_3,j_3} \in C^*(e_{\cdot,j})$ such that $|C(e^*_{i_3,j_3})| = 1$. For any two edges $e^*_{i_1,j_1}, e^*_{i_2,j_2} \in C^*(e_{i,\cdot})$, if $|C(e^*_{i_1,j_1})| = |C(e^*_{i_2,j_2})| = 2$, then the following two statements hold:*

1. *$e^*_{i_1,j_1}$ and $e^*_{i_2,j_2}$ are parallel, that is, either $i_2 = i_1+1, j_2 = j_1+1$ or $i_2 = i_1-1, j_2 = j_1-1$.*
2. *$C(e^*_{i_1,j_1}) \cap C(e^*_{i_2,j_2}) = \{e_{i,j}\}$.*

Proof. Using $|C(e^*_{i_3,j_3})| = 1$, we know from Lemma 3.14 that $|C(e^*_{i-1,j'-1})| \geq 2$, $|C(e^*_{i,j'})| \geq 2$ and $|C(e^*_{i+1,j'+1})| \geq 2$.

To prove the first statement, we suppose to the contrary that $i_1 = i-1$ and $i_2 = i+1$, and thus $|C(e^*_{i-1,j'-1})| = |C(e^*_{i+1,j'+1})| = 2$. From the inequality (3.5) of Lemma 3.19, we have $|C(e^*_{i,j'})| \leq |C(C^*(e_{i,\cdot}))| \leq 3$. Since Lemma 3.16 has ruled out the possibility of $|C(e^*_{i,j'})| = 2$, we have $|C(e^*_{i,j'})| = |C(C^*(e_{i,\cdot}))| = 3$. However, the algorithm $\mathcal{LS}$ would replace the three edges of $C(C^*(e_{i,\cdot}))$ by $e^*_{i_3,j_3}$ and the three edges of $C^*(e_{i,\cdot})$ to expand M, a contradiction.

Based on the first statement, we assume without loss of generality that $|C(e^*_{i,j'})| = |C(e^*_{i-1,j'-1})| = 2$. Note that $e_{i,j} \in C(e^*_{i,j'}) \cap C(e^*_{i-1,j'-1})$. If $C(e^*_{i,j'}) = C(e^*_{i-1,j'-1})$, then the algorithm $\mathcal{LS}$ would replace the two edges of $C(e^*_{i,j'})$ by the three edges $e^*_{i,j'}, e^*_{i-1,j'-1}, e^*_{i_3,j_3}$ to expand M, a contradiction. Therefore, $C(e^*_{i,j'}) \neq C(e^*_{i-1,j'-1})$, which implies the second statement $C(e^*_{i_1,j_1}) \cap C(e^*_{i_2,j_2}) = \{e_{i,j}\}$. □

Note that each value combination $\{\tau_1, \tau_2, \tau_3\}$ of $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot}))$ in Lemma 3.18 gives rise to $3! = 6$ different ordered value combinations. Due to symmetry, we consider only three of them: (τ_2, τ_1, τ_3), (τ_1, τ_2, τ_3), and (τ_1, τ_3, τ_2), in the following to determine whether or not they can be possible ordered value combinations for $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot}))$.

1. $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left\{1, \frac{1}{2}, \frac{1}{2}\right\}$.
   The case of $\left(1, \frac{1}{2}, \frac{1}{2}\right)$ can be ruled out by the inequalities (3.5) and (3.6) of Lemma 3.19. Then the only possible case left is $\left(\frac{1}{2}, 1, \frac{1}{2}\right)$.
2. $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left\{1, \frac{1}{2}, \frac{1}{3}\right\}$.
   The case of $\left(1, \frac{1}{3}, \frac{1}{2}\right)$ can immediately be ruled out by the inequality (3.5) of Lemma 3.19. Then the two possible cases left are $\left(\frac{1}{2}, 1, \frac{1}{3}\right)$ and $\left(1, \frac{1}{2}, \frac{1}{3}\right)$.
3. $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left\{1, \frac{1}{2}, \frac{1}{4}\right\}$.
   Both cases of $\left(\frac{1}{2}, 1, \frac{1}{4}\right)$ and $\left(1, \frac{1}{4}, \frac{1}{2}\right)$ can immediately be ruled out by Lemma 3.15. Then the only possible case left is $\left(1, \frac{1}{2}, \frac{1}{4}\right)$.
4. $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left\{1, \frac{1}{3}, \frac{1}{3}\right\}$.
   Consider the case of $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(1, \frac{1}{3}, \frac{1}{3}\right)$. In this case, we have $C(C^*(e_{i,\cdot})) = C(e^*_{i,j'}) = C(e^*_{i+1,j'+1})$ with $|C(C^*(e_{i,\cdot}))| = 3$, indicating that one of the three edges in $C(C^*(e_{i,\cdot}))$ must be a singleton edge of M and there is no edge in $M - C(C^*(e_{i,\cdot}))$ parallel with any edge in $C(C^*(e_{i,\cdot}))$. However, the algorithm $\mathcal{LS}$ would replace the three edges of $C(C^*(e_{i,\cdot}))$ by the three parallel edges of $C^*(e_{i,\cdot})$ to reduce the singleton edges in M, a contradiction.
   Thus the only possible case left is $\left(\frac{1}{3}, 1, \frac{1}{3}\right)$.
5. $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left\{\frac{1}{2}, \frac{1}{2}, \frac{1}{3}\right\}$.
   The case of $\left(\frac{1}{2}, \frac{1}{3}, \frac{1}{2}\right)$ can immediately be ruled out by Lemma 3.20. Then the only possible case left is $\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{3}\right)$.
6. $\tau(C^*(e_{i,\cdot})) = \left\{\frac{1}{2}, \frac{1}{2}, \frac{1}{4}\right\}$.
   The case of $\left(\frac{1}{2}, \frac{1}{4}, \frac{1}{2}\right)$ can immediately be ruled out by the inequality (3.5) of Lemma 3.19. Then the only possible case left is $\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{4}\right)$.
7. $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left\{\frac{1}{2}, \frac{1}{3}, \frac{1}{3}\right\}$.
   Both cases of $\left(\frac{1}{3}, \frac{1}{2}, \frac{1}{3}\right)$ and $\left(\frac{1}{2}, \frac{1}{3}, \frac{1}{3}\right)$ are possible.
8. $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left\{\frac{1}{2}, \frac{1}{3}, \frac{1}{4}\right\}$.
   All three cases of $\left(\frac{1}{3}, \frac{1}{2}, \frac{1}{4}\right)$, $\left(\frac{1}{2}, \frac{1}{3}, \frac{1}{4}\right)$, and $\left(\frac{1}{2}, \frac{1}{4}, \frac{1}{3}\right)$ are possible.
9. $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left\{\frac{1}{2}, \frac{1}{3}, \frac{1}{5}\right\}$.
   Both cases of $\left(\frac{1}{3}, \frac{1}{2}, \frac{1}{5}\right)$ and $\left(\frac{1}{2}, \frac{1}{5}, \frac{1}{3}\right)$ can immediately be ruled out by Lemma 3.15. Then the only possible case left is $\left(\frac{1}{2}, \frac{1}{3}, \frac{1}{5}\right)$.
10. $\tau(C^*(e_{i,\cdot})) = \left\{\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right\}$.
    Both cases of $\left(\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right)$ and $\left(\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\right)$ are possible.
11. $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right\}$.
    The only case $\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$ is possible.

12. $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left\{\frac{1}{2}, \frac{1}{2}, 0\right\}$.
    Both cases of $\left(\frac{1}{2}, \frac{1}{2}, 0\right)$ and $\left(\frac{1}{2}, 0, \frac{1}{2}\right)$ are possible.

We summarize the above discussion in the following lemma:

Lemma 3.21. *For an edge $e_{i,j} \in M$ with $\omega(e_{i,j}) \geq 3$, there are* 18 *possible ordered value combinations of $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot}))$, which are* $\left(\frac{1}{2}, 1, \frac{1}{2}\right)$, $\left(\frac{1}{2}, 1, \frac{1}{3}\right)$, $\left(1, \frac{1}{2}, \frac{1}{3}\right)$, $\left(1, \frac{1}{2}, \frac{1}{4}\right)$, $\left(\frac{1}{3}, 1, \frac{1}{3}\right)$, $\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{3}\right)$, $\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{4}\right)$, $\left(\frac{1}{3}, \frac{1}{2}, \frac{1}{3}\right)$, $\left(\frac{1}{2}, \frac{1}{3}, \frac{1}{3}\right)$, $\left(\frac{1}{3}, \frac{1}{2}, \frac{1}{4}\right)$, $\left(\frac{1}{2}, \frac{1}{3}, \frac{1}{4}\right)$, $\left(\frac{1}{2}, \frac{1}{4}, \frac{1}{3}\right)$, $\left(\frac{1}{2}, \frac{1}{3}, \frac{1}{5}\right)$, $\left(\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right)$, $\left(\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\right)$, $\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$, $\left(\frac{1}{2}, \frac{1}{2}, 0\right)$, *and* $\left(\frac{1}{2}, 0, \frac{1}{2}\right)$.

3.4.2.4 Edge combinations of $C(C^*(e_{i,j}))$ with $\omega(e_{i,j}) \geq 3$

We examine all possible combinations of the edges in $C(C^*(e_{i,\cdot}))$ with $\omega(e_{i,j}) \geq 3$. We distinguish two cases where $e_{i,j} \in p(M)$ and $e_{i,j} \in s(M)$, respectively. In fact, as shown in the following section, the edge $e_{i,j}$ cannot be a parallel edge in M.

$e_{i,j}$ cannot be a parallel edge of M Recall that the number of singleton edges of the maximal compatible matching M cannot be further reduced by the algorithm $\mathcal{LS}$ using the operation REDUCE-5-BY-5.

We assume to the contrary that $e_{i,j} \in p(M)$, and assume that $e_{i+1,j+1} \in p(M)$ too.

From $|C^*(e_{i,j})| \geq 5$ in Lemma 3.17, we consider $|C^*(e_{i,\cdot})| = 3$ and suppose that $C^*(e_{i,\cdot}) = \{e^*_{i-1,j'-1}, e^*_{i,j'}, e^*_{i+1,j'+1}\}$ for some $j' \neq j$.

Clearly, $|C(e^*_{i,j'})| \geq 2$ and $|C(e^*_{i+1,j'+1})| \geq 2$ since both contain the edges $e_{i,j}$ and $e_{i+1,j+1}$. It follows that the middle value in the ordered value combination of $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot}))$ must be $\leq \frac{1}{2}$. This rules out three of the 18 possible ordered value combinations stated in Lemma 3.21, each having a 1 in the middle, which are $\left(\frac{1}{2}, 1, \frac{1}{2}\right)$, $\left(\frac{1}{2}, 1, \frac{1}{3}\right)$, $\left(\frac{1}{3}, 1, \frac{1}{3}\right)$. Furthermore, since $\left(\frac{1}{2}, 1, \frac{1}{2}\right)$ is the only one resulted from the (unordered) value combination $\left\{1, \frac{1}{2}, \frac{1}{2}\right\}$, we conclude that it is impossible to have $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left\{1, \frac{1}{2}, \frac{1}{2}\right\}$. For the same reason, it is impossible to have $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left\{1, \frac{1}{3}, \frac{1}{3}\right\}$.

When $|C^*(e_{\cdot,j})| = 3$, the argument in the last paragraph applies to $C^*(e_{\cdot,j})$ too.

Consider the 8 possible value combinations of $\tau(e_{i,j} \leftarrow C^*(e_{i,j}))$ such that $\omega(e_{i,j}) \geq 3$, in Lemma 3.17. We observe that $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) \in \left\{\left\{\frac{1}{2}, \frac{1}{3}, \frac{1}{4}\right\}, \left\{\frac{1}{2}, \frac{1}{3}, \frac{1}{5}\right\}, \left\{\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right\}, \left\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right\}, \left\{\frac{1}{2}, \frac{1}{2}, 0\right\}\right\}$ only if $\tau(e_{i,j} \leftarrow C^*(e_{\cdot,j})) = \left\{1, \frac{1}{2}, \frac{1}{2}\right\}$, which is impossible to happen. We thus conclude that only 5 out of the 12 value combinations of $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot}))$ in Lemma 3.18 remain possible, which are $\{1, \frac{1}{2}, \frac{1}{3}\}$, $\{1, \frac{1}{2}, \frac{1}{4}\}$, $\{\frac{1}{2}, \frac{1}{2}, \frac{1}{3}\}$, $\{\frac{1}{2}, \frac{1}{2}, \frac{1}{4}\}$, and $\{\frac{1}{2}, \frac{1}{3}, \frac{1}{3}\}$. These give 6 possible ordered value combinations of $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot}))$, which are, $(1, \frac{1}{2}, \frac{1}{3})$, $(1, \frac{1}{2}, \frac{1}{4})$, $(\frac{1}{2}, \frac{1}{2}, \frac{1}{3})$, $(\frac{1}{2}, \frac{1}{2}, \frac{1}{4})$, $(\frac{1}{3}, \frac{1}{2}, \frac{1}{3})$, and $(\frac{1}{2}, \frac{1}{3}, \frac{1}{3})$.

In the rest of this section, we have $|C^*(e_{i,j})| = 6$, $C^*(e_{i,\cdot}) = \{e^*_{i-1,j'-1}, e^*_{i,j'}, e^*_{i+1,j'+1}\}$ for some $j' \neq j$, and $C^*(e_{\cdot,j}) = \{e^*_{i'-1,j-1}, e^*_{i',j}, e^*_{i'+1,j+1}\}$ for some $i' \neq i$.

Lemma 3.22. *For the pair of parallel edges $e_{i,j}, e_{i+1,j+1} \in p(M)$, $C^*(e_{i,j}) \cap C^*(e_{i+1,j+1}) = \{e^*_{i,j'}, e^*_{i+1,j'+1}, e^*_{i',j}, e^*_{i'+1,j+1}\}$. If $|C(e^*_{i-1,j'-1})| = 1$, then there is at most one edge $e^*_{i_1,j_1} \in C^*(e_{i,j}) \cap C^*(e_{i+1,j+1})$ such that $|C(e^*_{i_1,j_1})| = 2$.*

Proof. The first half of the lemma is trivial. For the second half, we note that $C(e^*_{i-1,j'-1}) = \{e_{i,j}\}$; if there is another edge $e^*_{i_2,j_2} \in C^*(e_{i,j}) \cap C^*(e_{i+1,j+1})$ such that $|C(e^*_{i_2,j_2})| = 2$, that is, $C(e^*_{i_1,j_1}) = C(e^*_{i_2,j_2}) = \{e_{i,j}, e_{i+1,j+1}\}$, then the algorithm $\mathcal{LS}$ would replace the two edges $e_{i,j}$ and $e_{i+1,j+1}$ by the three edges $e^*_{i-1,j'-1}, e^*_{i_1,j_1}, e^*_{i_2,j_2}$ to expand M, a contradiction. $\square$

Lemma 3.22 states that when $e_{i,j}$ is a parallel edge of M, the value combination of $\tau(e_{i,j} \leftarrow C^*(e_{i,j}))$ contains at most two $\frac{1}{2}$'s, besides the value 1. Among the 8 possible value combinations of $\tau(e_{i,j} \leftarrow C^*(e_{i,j}))$ such that $\omega(e_{i,j}) \geq 3$, in Lemma 3.17, the only one with two $\frac{1}{2}$'s is $\{1, \frac{1}{2}, \frac{1}{2}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}\}$. This leaves only two possible ordered value combinations of $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot}))$, which are, $(1, \frac{1}{2}, \frac{1}{3})$ and $(\frac{1}{2}, \frac{1}{3}, \frac{1}{3})$.

Assume $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = (1, \frac{1}{2}, \frac{1}{3})$ and $\tau(e_{i,j} \leftarrow C^*(e_{\cdot,j})) = (\frac{1}{2}, \frac{1}{3}, \frac{1}{3})$ (or the other way around). By the inequalities (3.5) and (3.6) in Lemma 3.19, we have $|C(C^*(e_{i,\cdot}))| = 3$ and $3 \leq |C(C^*(e_{\cdot,j}))| \leq 4$. Since $\{e_{i,j}, e_{i+1,j+1}\} \subseteq C(C^*(e_{i,\cdot})) \cap C(C^*(e_{\cdot,j}))$, we have $4 \leq |C(C^*(e_{i,j}))| \leq 5$. Thus, the algorithm $\mathcal{LS}$ would replace all the edges of $C(C^*(e_{i,j}))$ by the six edges of $C^*(e_{i,j})$ to expand M. This contradiction leaves no ordered value combination of $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot}))$. We thus have proved the following lemma:

Lemma 3.23. *When the edge $e_{i,j}$ is a parallel edge of M, there is no value combination of $\tau(e_{i,j} \leftarrow C^*(e_{i,j}))$ such that $\omega(e_{i,j}) \geq 3$.*

$e_{i,j}$ is a singleton edge of M With $e_{i,j} \in s(M)$, we discuss each of the 18 possible ordered value combinations of $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot}))$ listed in Lemma 3.21.

Consider an edge $e_{h,\ell} \in C(C^*(e_{i,\cdot}))$, $e_{h,\ell} \neq e_{i,j}$. Note that $e_{h,\ell}$ might be parallel with an edge in $M - C(C^*(e_{i,\cdot}))$. We define $\mathcal{N}_p(e_{h,\ell})$ to be the subset of the maximal consecutive parallel (to $e_{h,\ell}$) edges in $M - C(C^*(e_{i,.}))$. Therefore, $\mathcal{N}_p(e_{h,\ell})$ will be either $\{e_{h+1,\ell+1}, \ldots, e_{h+q,\ell+q}\}$ or $\{e_{h-1,\ell-1}, \ldots, e_{h-q,\ell-q}\}$, for some $q \geq 0$ (when $q = 0$, this set is empty). Let

$$\mathcal{N}_p(C(C^*(e_{i,.}))) = \bigcup_{e_{h,\ell} \in C(C^*(e_{i,.}))} \mathcal{N}_p(e_{h,\ell}),$$

and

$$\mathcal{N}_p[C(C^*(e_{i,.}))] = \mathcal{N}_p(C(C^*(e_{i,.}))) \cup C(C^*(e_{i,.})).$$

Recall that, in general, the subscript of a vertex of D^A has an i or h, and the subscript of a vertex of D^B has a j or ℓ. In the sequel, for simplicity, we use e_h and e_ℓ (e^*_h and e^*_ℓ, respectively) to denote the edges of M (M^*, respectively) incident on the vertices d^A_h and d^B_ℓ, respectively, if they exist, or otherwise they are void edges.

We next discuss all possible configurations of the edges of $C^*(e_{i,.})$ and $C(C^*(e_{i,.}))$ in figures, associated with each of the 18 ordered value combinations of $\tau(e_{i,j} \leftarrow C^*(e_{i,.}))$ listed in Lemma 3.21. We adopt the following scheme for graphically presenting a configuration: In each figure (for example, Figure 3.15), the edge $e_{i,j}$ is in the bold solid line; the edges in vertical bold dashed lines are in $C^*(e_{i,.})$ (for example, $e^*_{i,j'}$); the edges in thin solid lines are edges in $C(C^*(e_{i,.}))$ (for example, e_{i+2}); and the edges in thin dashed lines are edges in $\mathcal{N}_p(C(C^*(e_{i,.}))$ (for example, e_{i+3}); the vertices in filled circles are surely not incident with any edge of M (for example, $i - 2$); the vertices in hollow circles have uncertain incidence in M (for example, $j' - 2$).

We remind the readers that if there is no entry 1 in a value combination of $\tau(e_{i,j} \leftarrow C^*(e_{i,.}))$, then there must be an entry 1 in the corresponding value combination of $\tau(e_{i,j} \leftarrow C^*(e_{.,j}))$, that is, there is an edge $e^*_{i_1,j_1} \in C^*(e_{.,j}))$ such that $|C(e^*_{i_1,j_1})| = 1$.

1. $\tau(e_{i,j} \leftarrow C^*(e_{i,.})) = \left(\frac{1}{2}, 1, \frac{1}{2}\right)$: According to the inequalities (3.5) and (3.6) of Lemma 3.19, we have $|C(C^*(e_{i,.}))| = 3$. There is exactly one edge of M incident on either of $i + 2$ and $j' + 2$ but not both. We assume $e_{i+2} \in M$. If e_{i+2} is a singleton edge of M or $|\mathcal{N}_p(e_{i+2})| \geq 2$, then the algorithm $\mathcal{LS}$ would replace $e_{i,j}$ and e_{i+2} by the two parallel edges $e^*_{i,j'}$ and $e^*_{i+1,j'+1}$ to reduce the singleton edges, a contradiction. Therefore, we have $e_{i+3} \in M$ but no edge of M is incident on $i + 4$. The incidence at $i - 2$ and $j' - 2$ and further to the left can be symmetrically discussed. In this sense, there is only one possible edge combination of $C(C^*(e_{i,.}))$, as shown in Figure 3.15 with $e_{i+2}, e_{j'-2} \in M$, where the corresponding configuration of $\mathcal{N}_p[C(C^*(e_{i,.}))]$ is also shown.

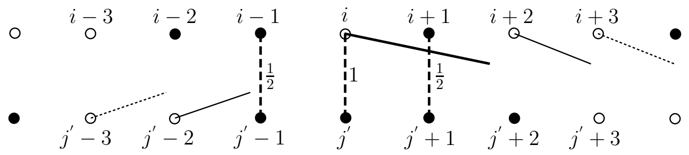


FIGURE 3.15: The only possible configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{2}, 1, \frac{1}{2}\right)$. We have $|C(C^*(e_{i,\cdot}))| = 3$, and $|\mathcal{N}_p(e_{i+2})| = |\mathcal{N}_p(e_{j'-2})| = 1$ in this configuration. It also represents the other three symmetric configurations where $|\mathcal{N}_p(e_{i+2})| = |\mathcal{N}_p(e_{i-2})| = 1$, $|\mathcal{N}_p(e_{j'+2})| = |\mathcal{N}_p(e_{j'-2})| = 1$, and $|\mathcal{N}_p(e_{j'+2})| = |\mathcal{N}_p(e_{i-2})| = 1$, respectively. (Recall that the edge $e_{i,j}$ is in bold solid line, the edges in vertical bold dashed lines are in $C^*(e_{i,\cdot})$, the edges in thin solid lines are in $C(C^*(e_{i,\cdot}))$, and the edges in thin dashed lines are in $\mathcal{N}_p(C(C^*(e_{i,\cdot}))$; the vertices in filled circles are surely incident with no edges of M and the vertices in hollow circles have uncertain incidence in M.)

2. $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{2}, 1, \frac{1}{3}\right)$: We have $C(e^*_{i,j'}) \subset C(e^*_{i-1,j'-1})$, and thus $|C(e^*_{i,j'}) \cup C(e^*_{i-1,j'-1})| = 2$ and $|C(C^*(e_{i,\cdot}))| = 4$. There is exactly one edge of M incident on either of $i-2$ and $j'-2$ but not both. We assume $e_{j'-2} \in M$. If $e_{j'-2}$ is a singleton edge of M or $|\mathcal{N}_p(e_{j'-2})| \geq 2$, then the algorithm $\mathcal{LS}$ would replace $e_{i,j}$ and $e_{j'-2}$ by the two parallel edges $e^*_{i,j'}$ and $e^*_{i-1,j'-1}$ to reduce the singleton edges, a contradiction. Therefore, we have $e_{j'-3} \in M$ but no edge of M is incident on $j'-4$. In this sense, there is only one possible edge combination of $C(C^*(e_{i,\cdot}))$, as shown in Figure 3.16 with $e_{j'-2} \in M$, where the corresponding configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ is also shown.

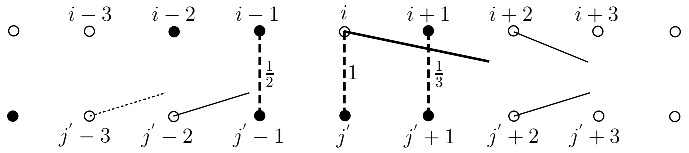


FIGURE 3.16: The only possible configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{2}, 1, \frac{1}{3}\right)$. We have $|C(C^*(e_{i,\cdot}))| = 4$ and $|\mathcal{N}_p(e_{j'-2})| = 1$ in this configuration. It also represents the symmetric configuration where $|\mathcal{N}_p(e_{i-2})| = 1$.

3. $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(1, \frac{1}{2}, \frac{1}{3}\right)$: According to the inequalities (3.5) and (3.6) of Lemma 3.19, we have $|C(C^*(e_{i,\cdot}))| = 3$. Since $e_{i,j}$ is a singleton edge of M, $e_{j'+1} \in M$; and either $e_{i+2} \in M$ or $e_{j'+2} \in M$ but no both. If $e_{i+2} \in M$, then $e_{j'+1}$ is a singleton edge of M, and thus the algorithm $\mathcal{LS}$ would replace $e_{i,j}$ and $e_{j'+1}$ by the two parallel edges $e^*_{i,j'}$ and $e^*_{i-1,j'-1}$ to reduce the singleton edges, a contradiction. Therefore, $e_{j'+2} \in M$. Similarly, if $\mathcal{N}_p(e_{j'+2}) = \emptyset$ or $|\mathcal{N}_p(e_{j'+2})| \geq 2$, then the algorithm $\mathcal{LS}$ would replace the three edges of $C(C^*(e_{i,\cdot}))$ by the three parallel edges of $C^*(e_{i,\cdot})$ to reduce the singleton edges, a contradiction. This leaves the only possible configuration

with $|\mathcal{N}_p(e_{j'+2})| = 1$, as shown in Figure 3.17, where the corresponding configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ is also shown.

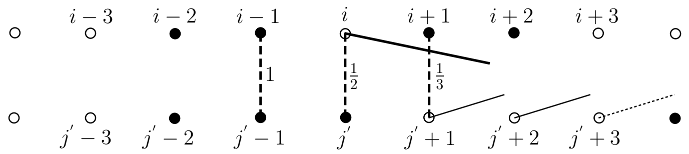


FIGURE 3.17: The only possible configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(1, \frac{1}{2}, \frac{1}{3}\right)$. We have $|C(C^*(e_{i,\cdot}))| = 3$ and $|\mathcal{N}_p(e_{j'+2})| = 1$ in this configuration.

4. $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(1, \frac{1}{2}, \frac{1}{4}\right)$: We have $e_{j'} \notin M$ and $|C(C^*(e_{i,\cdot}))| = 4$. Therefore, $e_{i+2}, e_{j'+1}, e_{j'+2} \in M$. If $|\mathcal{N}_p(e_{j'+2})| \geq 1$, then the algorithm $\mathcal{LS}$ would replace $e_{i,j}$ and $e_{j'+1}$ by the two parallel edges $e^*_{i,j'}$ and $e^*_{i-1,j'-1}$ to reduce the singleton edges, a contradiction. Therefore, $\mathcal{N}_p(e_{j'+2}) = \emptyset$, that is, $e_{j'+3} \notin M$. There is only one possible edge combination of $C(C^*(e_{i,\cdot}))$, as shown in Figure 3.18, where the corresponding configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ is also shown.

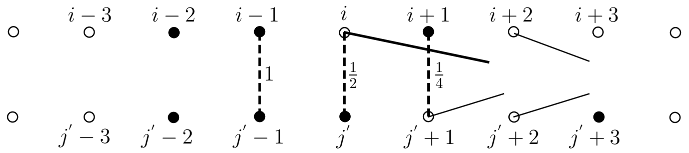


FIGURE 3.18: The only possible configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(1, \frac{1}{2}, \frac{1}{4}\right)$. We have $|C(C^*(e_{i,\cdot}))| = 4$ and $\mathcal{N}_p(e_{j'+2}) = \emptyset$ in this configuration.

5. $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{3}, 1, \frac{1}{3}\right)$: According to the inequalities (3.5) and (3.6) of Lemma 3.19, we have $|C(C^*(e_{i,\cdot}))| = 5$. There is only one possible edge combination of $C(C^*(e_{i,\cdot}))$, which is shown in Figure 3.19, where any configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ is possible.

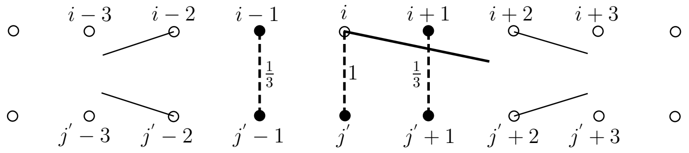
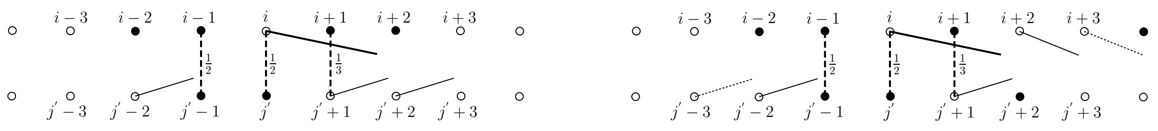


FIGURE 3.19: The only possible configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{3}, 1, \frac{1}{3}\right)$, where $|C(C^*(e_{i,\cdot}))| = 5$.

6. $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{3}\right)$: According to Lemma 3.20, we have $C(e^*_{i,j'}) \cap C(e^*_{i-1,j'-1}) = \{e_{i,j}\}$; thus $e_{j'+1} \in M$, either $e_{i'-2} \in M$ or $e_{j'-2} \in M$ but no both, either $e_{i+2} \in M$

or $e_{j'+2} \in M$ but no both, and $|C(C^*(e_{i,\cdot}))| = 4$. We assume $e_{j'-2} \in M$ ($e_{i'-2} \in M$ is discussed the same). When $e_{i+2} \in M$, $e_{j'+1}$ is a singleton edge of M. If $e_{j'-2}$ is also a singleton edge of M, then the algorithm $\mathcal{LS}$ would replace the four edges in $C(C^*(e_{i,\cdot}))$ by the three parallel edges in $C^*(e_{i,\cdot})$ and $e^*_{i_1,j_1}$ to reduce the singleton edges, a contradiction. Therefore in this case we have $|\mathcal{N}_p(e_{j'-2})| \ge 1$, that is, $e_{j'-3} \in M$. Similarly, if e_{i+2} is a singleton edge of M or $|\mathcal{N}_p(e_{i+2})| \ge 2$, then the algorithm $\mathcal{LS}$ would replace the three edges $e_{i,j}, e_{j'+1}, e_{i+2}$ by the two parallel edges $e^*_{i,j'}, e^*_{i+1,j'+1}$ and $e^*_{i_1,j_1}$ to reduce the singleton edges, a contradiction. That is, $e_{i+3} \in M$ but $e_{i+4} \notin M$. This edge combination of $C(C^*(e_{i,\cdot}))$ is shown in Figure 3.20b, where the corresponding configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ is also shown.

When $e_{j'+2} \in M$, for the same reason, if $|\mathcal{N}_p(e_{j'+2})| \ne 1$ then $e_{j'-2}$ must not be a singleton edge of M. This edge combination of $C(C^*(e_{i,\cdot}))$ is shown in Figure 3.20a, where the corresponding configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ is also shown.



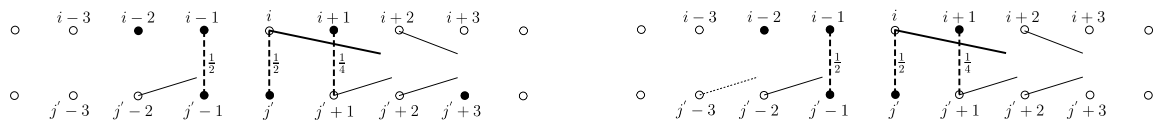
(A) If $|\mathcal{N}_p(e_{j'+2})| \ne 1$ then $|\mathcal{N}_p(e_{j'-2})| \ge 1$.

(B) $|\mathcal{N}_p(e_{i+2})| = 1$ and $|\mathcal{N}_p(e_{j'-2})| \ge 1$.

FIGURE 3.20: The two possible configurations of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{3}\right)$. We have $|C(C^*(e_{i,\cdot}))| = 4$. They also represent the symmetric case where $e_{i'-2} \in M$ instead of $e_{j'-2} \in M$.

7. $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{4}\right)$: According to Lemma 3.20, we have $C(e^*_{i,j'}) \cap C(e^*_{i-1,j'-1}) = \{e_{i,j}\}$; thus $e_{j'+1} \in M$, either $e_{i'-2} \in M$ or $e_{j'-2} \in M$ but no both, $e_{i+2}, e_{j'+2} \in M$, and $|C(C^*(e_{i,\cdot}))| = 5$. We assume $e_{j'-2} \in M$ ($e_{i'-2} \in M$ is discussed the same). If $e_{j'-2}$ is a singleton edge of M and $|\mathcal{N}_p(e_{j'+2})| \ge 1$, then the algorithm $\mathcal{LS}$ would replace the three edges $e_{i,j}$, $e_{j'-2}$, and $e_{j'+1}$ by $e^*_{i_1,j_1}$ and the two parallel edges $e^*_{i,j'}$ and $e^*_{i-1,j'-1}$ to reduce the singleton edges, a contradiction. Therefore, $|\mathcal{N}_p(e_{j'-2})| \ge 1$ (shown in Figure 3.21b) or $\mathcal{N}_p(e_{j'+2}) = \emptyset$ (shown in Figure 3.21a). These two edge combinations of $C(C^*(e_{i,\cdot}))$ are shown in Figure 3.21a and Figure 3.21b, respectively, where the corresponding configurations of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ are also shown.

   Between the two configurations shown in Figure 3.21a and Figure 3.21b, we notice that for every edge $e \in C(C^*(e_{i,\cdot})) - \{e_{i,j}\}$, the largest possible value for $\omega(e)$ in Figure 3.21a is at least as large as in Figure 3.21b. Since we are interested in the worst-case analysis, we say Figure 3.21b is *shadowed* by Figure 3.21a and we will consider Figure 3.21a only in the sequel.



(A) $\mathcal{N}_p(e_{j'+2}) = \emptyset$.

(B) $|\mathcal{N}_p(e_{j'-2})| \geq 1$.

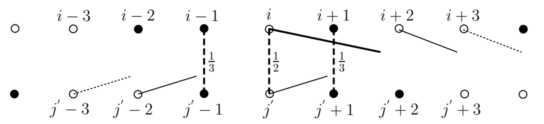
FIGURE 3.21: The two possible configurations of $\mathcal{N}_p[C(C^*(e_{i,.}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,.})) = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{4}\right)$. They are associated with the only possible edge combination of $C(C^*(e_{i,.}))$ with $|C(C^*(e_{i,.}))| = 5$, which also represents the symmetric case where $e_{i'-2} \in M$ instead of $e_{j'-2} \in M$. The first configuration shadows the second one.

8. $\tau(e_{i,j} \leftarrow C^*(e_{i,.})) = \left(\frac{1}{3}, \frac{1}{2}, \frac{1}{3}\right)$: According to the inequalities (3.5) and (3.6) of Lemma 3.19, we have $4 \leq |C(C^*(e_{i,.}))| \leq 5$. Since $i-1$ and $i+1$ are symmetric with respect to i, we only discuss one of them. We have either $e_{j'} \in M$ or $e_{j'-1} \in M$, but not both.

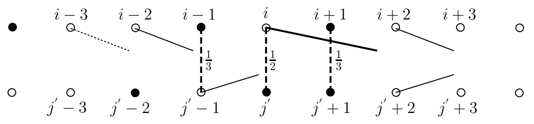
   When $e_{j'} \in M$, then either $e_{i-2} \in M$ or $e_{j'-2} \in M$, but not both. We assume $e_{j'-2} \in M$. Similarly, either $e_{i+2} \in M$ or $e_{j'+2} \in M$, but not both. We assume $e_{i+2} \in M$. If e_{i+2} is a singleton edge of M or $|\mathcal{N}_p(e_{i+2})| \geq 2$, then the algorithm $\mathcal{LS}$ would replace the three edges $e_{i,j}$, $e_{j'}$, e_{i+2} by $e^*_{i_1,j_1}$ and the two parallel edges $e^*_{i,j'}$ and $e^*_{i+1,j'+1}$ to reduce the singleton edges, a contradiction. Therefore, $|\mathcal{N}_p(e_{i+2})| = 1$; for the same reason, $|\mathcal{N}_p(e_{j'-2})| = 1$. This edge combination of $C(C^*(e_{i,.}))$ is shown in Figure 3.22a, where the corresponding configuration of $\mathcal{N}_p[C(C^*(e_{i,.}))]$ is also shown.

   When $e_{j'-1} \in M$, then still either $e_{i-2} \in M$ or $e_{j'-2} \in M$, but not both. On the other side, $e_{i+2} \in M$ and $e_{j'+2} \in M$. When $e_{i-2} \in M$, $e_{j'-1}$ is a singleton edge of M; and therefore $|\mathcal{N}_p(e_{i'-2})| = 1$. This edge combination of $C(C^*(e_{i,.}))$ is shown in Figure 3.22b, where the corresponding configuration of $\mathcal{N}_p[C(C^*(e_{i,.}))]$ is also shown.

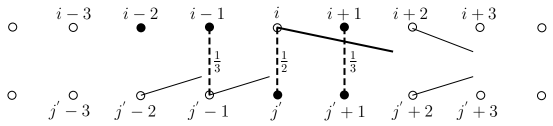
   When $e_{j'-2} \in M$, the edge combination of $C(C^*(e_{i,.}))$ is shown in Figure 3.22c, where the corresponding configuration of $\mathcal{N}_p[C(C^*(e_{i,.}))]$ is also shown.



(A) $|C(C^*(e_{i,.}))| = 4$ and $|\mathcal{N}_p(e_{i+2})| = |\mathcal{N}_p(e_{j'-2})| = 1$.



(B) $|C(C^*(e_{i,.}))| = 5$ and $|\mathcal{N}_p(e_{i-2})| = 1$.



(C) $|C(C^*(e_{i,.}))| = 5$.

FIGURE 3.22: The three possible configurations of $\mathcal{N}_p[C(C^*(e_{i,.}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,.})) = \left(\frac{1}{3}, \frac{1}{2}, \frac{1}{3}\right)$, associated with three possible edge combinations of $C(C^*(e_{i,.}))$ with $|C(C^*(e_{i,.}))| = 4, 5, 5$, respectively. The configuration in Figure 3.22a also represents the symmetric case where $e_{i-2} \in M$ instead of $e_{j'-2} \in M$ and/or $e_{j'+2} \in M$ instead of $e_{i+2} \in M$.

9. $\tau(e_{i,j} \leftarrow C^*(e_{i,.})) = \left(\frac{1}{2}, \frac{1}{3}, \frac{1}{3}\right)$: According to the inequalities (3.5) and (3.6) of Lemma 3.19, we have $3 \leq |C(C^*(e_{i,.}))| \leq 4$. If $|C(C^*(e_{i,.}))| = 3$, then the algorithm $\mathcal{LS}$ would replace the three edges of $C(C^*(e_{i,.}))$ by $e^*_{i_1,j_1}$ and the three parallel edges in $C^*(e_{i,.})$ to expand M, a contradiction. Therefore, $|C(C^*(e_{i,.}))| = 4$. From $e_{j'-1}, e_{j'+1} \in M$, we know that either $e_{i+2} \in M$ or $e_{j'+2} \in M$ but not both. If $e_{i+2} \in M$, then all three edges $e_{j'-1}, e_{i,j}, e_{j'+1}$ are singleton edges of M, and the algorithm $\mathcal{LS}$ would replace the four edges of $C(C^*(e_{i,.}))$ by $e^*_{i_1,j_1}$ and the three parallel edges of $C^*(e_{i,.})$ to reduce the singleton edges, a contradiction. Therefore, $e_{j'+2} \in M$, which then implies $|\mathcal{N}_p(e_{j'+2})| = 1$. This only edge combination of $C(C^*(e_{i,.}))$ is shown in Figure 3.23, where the corresponding configuration of $\mathcal{N}_p[C(C^*(e_{i,.}))]$ is also shown.

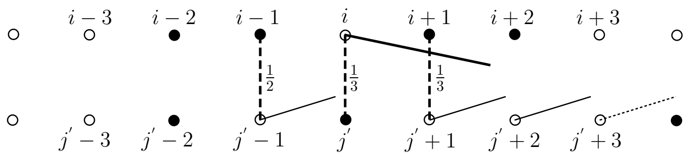
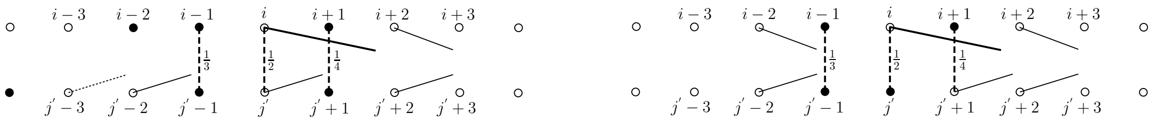


FIGURE 3.23: The only possible configuration of $\mathcal{N}_p[C(C^*(e_{i,.}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,.})) = \left(\frac{1}{2}, \frac{1}{3}, \frac{1}{3}\right)$. We have $|C(C^*(e_{i,.}))| = 4$ and $|\mathcal{N}_p(e_{j'+2})| = 1$.

10. $\tau(e_{i,j} \leftarrow C^*(e_{i,.})) = \left(\frac{1}{3}, \frac{1}{2}, \frac{1}{4}\right)$: According to the inequalities (3.5) and (3.6) of Lemma 3.19, we have $5 \leq |C(C^*(e_{i,.}))| \leq 6$. Note that either $e_{j'} \in M$ or $e_{j'+1} \in M$ but not both.

When $e_{j'} \in M$, we have two symmetric cases where $e_{i-2} \in M$ and $e_{j'-2} \in M$, respectively; and we assume $e_{j'-2} \in M$. We conclude that $e_{j'-2}$ must not be a singleton edge of M or $|\mathcal{N}_p(e_{j'-2})| \geq 2$. This edge combination of $C(C^*(e_{i,\cdot}))$ is shown in Figure 3.24a, where the corresponding configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ is also shown.

When $e_{j'+1} \in M$, both $e_{i-2} \in M$ and $e_{j'-2} \in M$. This edge combination of $C(C^*(e_{i,\cdot}))$ is shown in Figure 3.24b, where the corresponding configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ is also shown.



(A) $|C(C^*(e_{i,\cdot}))| = 5$ and $|\mathcal{N}_p(e_{j'-2})| = 1$.

(B) $|C(C^*(e_{i,\cdot}))| = 6$.

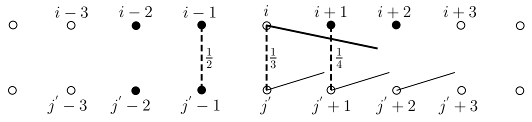
FIGURE 3.24: The two possible configurations of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{3}, \frac{1}{2}, \frac{1}{4}\right)$, associated with two possible edge combinations of $C(C^*(e_{i,\cdot}))$ with $|C(C^*(e_{i,\cdot}))| = 5, 6$, respectively. The configuration in Figure 3.24a also represents the symmetric case where $e_{i-2} \in M$ instead of $e_{j'-2} \in M$.

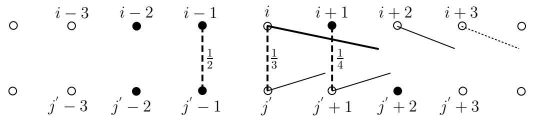
11. $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{2}, \frac{1}{3}, \frac{1}{4}\right)$: According to the inequalities (3.5) and (3.6) of Lemma 3.19, we have $4 \leq |C(C^*(e_{i,\cdot}))| \leq 5$. Note that either $e_{j'-1} \notin M$ or $e_{j'} \notin M$ but not both, and $e_{j'+1} \in M$.

    When $e_{j'-1} \notin M$, then either $e_{i+2} \in M$ or $e_{j'+2} \in M$ but not both. When $e_{j'+2} \in M$, the edge combination of $C(C^*(e_{i,\cdot}))$ is shown in Figure 3.25a, where the corresponding configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ is also shown.

    When $e_{i+2} \in M$, we conclude that e_{i+2} should not be a singleton edge of M; the edge combination of $C(C^*(e_{i,\cdot}))$ is shown in Figure 3.25b, where the corresponding configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ is also shown.

    When $e_{j'} \notin M$, then both $e_{i+2} \in M$ and $e_{j'+2} \in M$; we conclude that $\mathcal{N}_p(e_{j'+2}) = \emptyset$. This edge combination of $C(C^*(e_{i,\cdot}))$ is shown in Figure 3.25c, where the corresponding configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ is also shown.


(A) $|C(C^*(e_{i,\cdot}))| = 4$.


(B) $|C(C^*(e_{i,\cdot}))| = 4$ and $|\mathcal{N}_p(e_{i+2})| \geq 1$.

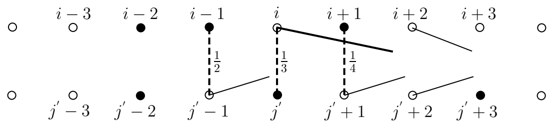

(C) $|C(C^*(e_{i,\cdot}))| = 5$ and $\mathcal{N}_p(e_{j'+2}) = \emptyset$.

FIGURE 3.25: The three possible configurations of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{2}, \frac{1}{3}, \frac{1}{4}\right)$, associated with three possible edge combinations of $C(C^*(e_{i,\cdot}))$ with $|C(C^*(e_{i,\cdot}))| = 4, 4, 5$, respectively.

12. $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{2}, \frac{1}{4}, \frac{1}{3}\right)$: This ordered value combination is impossible due to the edge $e_{i,j}$ being a singleton edge of M.
13. $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{2}, \frac{1}{3}, \frac{1}{5}\right)$: We have $e_{j'}, e_{j'+1}, e_{j'+2}, e_{i+2} \in M$, giving rise to $|C(C^*(e_{i,\cdot}))| = 5$. This only edge combination of $C(C^*(e_{i,\cdot}))$ is shown in Figure 3.26, where the corresponding configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ is also shown.

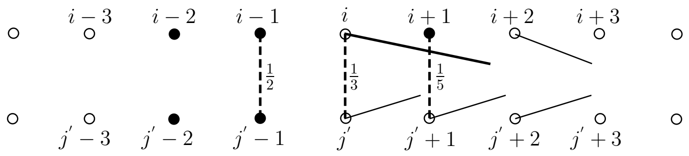


FIGURE 3.26: The only possible configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{2}, \frac{1}{3}, \frac{1}{5}\right)$, where $|C(C^*(e_{i,\cdot}))| = 5$.

14. $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right)$: This ordered value combination is impossible due to the edge $e_{i,j}$ being a singleton edge of M.
15. $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\right)$: Since the edge $e_{j'}$ has to be in M, we have both $e_{i-2} \in M$ and $e_{j'-2} \in M$, and both $e_{i+2} \in M$ and $e_{j'+2} \in M$, giving rise to $|C(C^*(e_{i,\cdot}))| = 6$. This only edge combination of $C(C^*(e_{i,\cdot}))$ is shown in Figure 3.27, where the corresponding configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ is also shown.

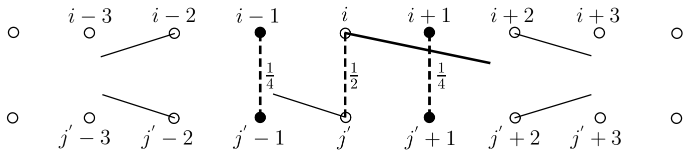
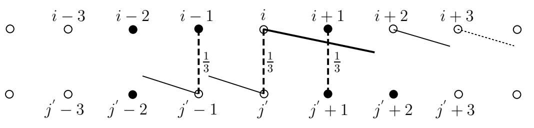


FIGURE 3.27: The only possible configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\right)$, where $|C(C^*(e_{i,\cdot}))| = 6$.

16. $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$: According to the inequalities (3.5) and (3.6) of Lemma 3.19, we have $4 \leq |C(C^*(e_{i,\cdot}))| \leq 5$. Note that exactly one of the three vertices $j'-1, j, j'+1$ is not incident with any edge of M, we thus consider two cases where $e_{j'} \notin M$ and $e_{j'+1} \notin M$ ($e_{j'-1} \notin M$ is symmetric to $e_{j'+1} \in M$), respectively.

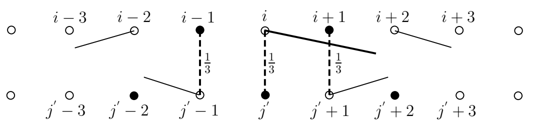
    When $e_{j'+1} \notin M$, then either $e_{i+2} \in M$ or $e_{j'+2} \in M$ but not both, while $e_{i-2} \notin M$ and $e_{j'-2} \notin M$. We assume $e_{i+2} \in M$, which implies e_{i+2} should not be a singleton edge of M. This edge combination of $C(C^*(e_{i,\cdot}))$ is shown in Figure 3.28a, where the corresponding configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ is also shown.

    When $e_{j'} \notin M$, then either $e_{i+2} \in M$ or $e_{j'+2} \in M$ but not both, and either $e_{i-2} \in M$ or $e_{j'-2} \in M$ but not both. Four different combinations of their memberships of M give rise to 0, 1, 2 singleton edges between $e_{j'-1}$ and $e_{j'+1}$. These three edge combinations of $C(C^*(e_{i,\cdot}))$ are shown in Figure 3.28b, Figure 3.28c, Figure 3.28d, respectively, where the corresponding configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ is also shown.



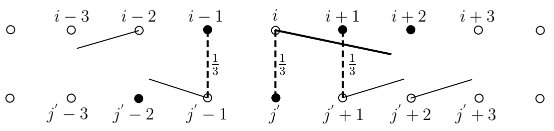
$i-3$ $i-2$ $i-1$ i $i+1$ $i+2$ $i+3$
$\frac{1}{3}$ $\frac{1}{3}$ $\frac{1}{3}$
$j'-3$ $j'-2$ $j'-1$ j' $j'+1$ $j'+2$ $j'+3$

(A) $|C(C^*(e_{i,\cdot}))| = 4$ and $|\mathcal{N}_p(e_{i+2})| \geq 1$.



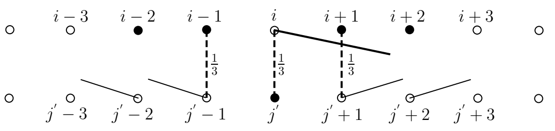
$i-3$ $i-2$ $i-1$ i $i+1$ $i+2$ $i+3$
$\frac{1}{3}$ $\frac{1}{3}$ $\frac{1}{3}$
$j'-3$ $j'-2$ $j'-1$ j' $j'+1$ $j'+2$ $j'+3$

(B) $|C(C^*(e_{i,\cdot}))| = 5$.



$i-3$ $i-2$ $i-1$ i $i+1$ $i+2$ $i+3$
$\frac{1}{3}$ $\frac{1}{3}$ $\frac{1}{3}$
$j'-3$ $j'-2$ $j'-1$ j' $j'+1$ $j'+2$ $j'+3$

(C) $|C(C^*(e_{i,\cdot}))| = 5$.

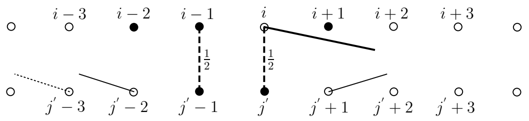


$i-3$ $i-2$ $i-1$ i $i+1$ $i+2$ $i+3$
$\frac{1}{3}$ $\frac{1}{3}$ $\frac{1}{3}$
$j'-3$ $j'-2$ $j'-1$ j' $j'+1$ $j'+2$ $j'+3$

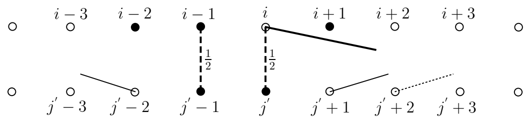
(D) $|C(C^*(e_{i,\cdot}))| = 5$.

FIGURE 3.28: The four possible configurations of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$, associated with four possible edge combinations of $C(C^*(e_{i,\cdot}))$ with $|C(C^*(e_{i,\cdot}))| = 4, 5, 5, 5$, respectively. Figure 3.28a also represents the case where $e_{j'+2} \in M$ instead of $e_{i+2} \in M$; Figure 3.28c also represents the case where $e_{j'-2}, e_{i+2} \in M$ instead of $e_{i-2}, e_{j'+2} \in M$.

17. $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{2}, \frac{1}{2}, 0\right)$: We have $2 \leq |C(C^*(e_{i,\cdot}))| \leq 3$. If $|C(C^*(e_{i,\cdot}))| = 2$, then the algorithm $\mathcal{LS}$ would replace the two edges of $C(C^*(e_{i,\cdot}))$ by $e^*_{i_1,j_1}$ and the two edges of $C^*(e_{i,\cdot})$ to expand M, a contradiction. Therefore, $|C(C^*(e_{i,\cdot}))| = 3$, and furthermore $e_{j'+1} \in M$, and either $e_{i-2} \in M$ or $e_{j'-2} \in M$ but not both. Due to symmetry we assume $e_{j'-2} \in M$. We conclude that at most one of $e_{j'+1}$ and $e_{j'-2}$ can be a singleton edge of M. The edge combination of $C(C^*(e_{i,\cdot}))$ when $e_{j'-2}$ is not a singleton edge is shown in Figure 3.29a, and the edge combination of $C(C^*(e_{i,\cdot}))$ when $e_{j'+1}$ is not a singleton edge is shown in Figure 3.29b, where the corresponding configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ is also shown, respectively.



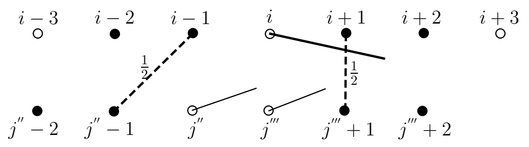
(A) $|C(C^*(e_{i,\cdot}))| = 3$ and $|\mathcal{N}_p(e_{j'-2})| \geq 1$.



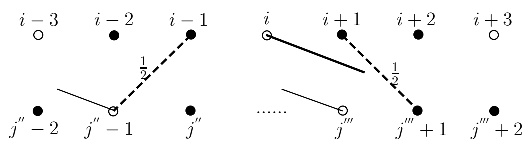
(B) $|C(C^*(e_{i,\cdot}))| = 3$ and $|\mathcal{N}_p(e_{j'+1})| \geq 1$.

FIGURE 3.29: The two possible configurations of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = (\frac{1}{2}, \frac{1}{2}, 0)$, associated with the only possible edge combinations of $C(C^*(e_{i,\cdot}))$ with $|C(C^*(e_{i,\cdot}))| = 3$. Each of them also represents the case where $e_{i-2} \in M$ instead of $e_{j'-2} \in M$.

18. $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = (\frac{1}{2}, 0, \frac{1}{2})$: We denote the two edges of $C^*(e_{i,\cdot})$ as $e_{i-1,j''-1}$ and $e_{i+1,j'''+1}$, respectively; clearly, $|(j''-1)-(j'''+1)| \geq 2$. We have $2 \leq |C(C^*(e_{i,\cdot}))| \leq 3$. The same as in the last case, we have $|C(C^*(e_{i,\cdot}))| = 3$, and furthermore exactly one of $i-2, j''-2, j''-1, j''$ is incident with an edge of M, and exactly one of $i+2, j''', j'''+1, j'''+2$ is incident with an edge of M. Among these 16 edge combinations of $C(C^*(e_{i,\cdot}))$, in one of them the two edges of $C(C^*(e_{i,\cdot}))$ could be parallel to each other, as shown in Figure 3.30a (this happens when $j''' = j''+1$, and $e_{j''}, e_{j'''} \in M$), where the corresponding configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ is also shown; one of the other 15 is shown in Figure 3.30b ($j''' > j''+1$, $e_{j''-1}, e_{j'''} \in M$), where the corresponding configuration of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ is also shown.



(A) $|C(C^*(e_{i,\cdot}))| = 3$ and $j''' = j''+1$.



(B) $|C(C^*(e_{i,\cdot}))| = 3$ and $j''' > j''+1$.

FIGURE 3.30: The two possible configurations of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ when $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = (\frac{1}{2}, 0, \frac{1}{2})$, associated with the two possible edge combinations of $C(C^*(e_{i,\cdot}))$ with $|C(C^*(e_{i,\cdot}))| = 3$. The second configuration represents the other 15 symmetric cases exactly one of $i-2, j''-2, j''-1, j''$ is incident with an edge of M and exactly one of $i+2, j''', j'''+1, j'''+2$ is incident with an edge of M, but the two edges are not parallel to each other.

Therefore, we have a total of 27 configurations of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ associated with all the possible edge combinations of $C(C^*(e_{i,\cdot}))$, up to symmetry, for further discussion.

Lemma 3.24. *When $e_{i,j}$ is a singleton edge of M with $\omega(e_{i,j}) \geq 3$, there is at least one parallel edge of M in $C(C^*(e_{i,j}))$ for each of the* 8 *possible value combinations of $\tau(e_{i,j} \leftarrow C^*(e_{i,j}))$.*

Proof. From Lemma 3.17, for each of the 8 possible value combinations of $\tau(e_{i,j} \leftarrow C^*(e_{i,j}))$, there is an entry 1 in the ordered value combination of $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot}))$ or

$\tau(e_{i,j} \leftarrow C^*(e_{\cdot,j}))$. There are only 5 possible such ordered value combinations, which are $\left(\frac{1}{2}, 1, \frac{1}{2}\right)$, $\left(\frac{1}{2}, 1, \frac{1}{3}\right)$, $\left(1, \frac{1}{2}, \frac{1}{3}\right)$, $\left(1, \frac{1}{2}, \frac{1}{4}\right)$, and $\left(\frac{1}{3}, 1, \frac{1}{3}\right)$. The above Figures 3.15–3.18 show that for the first 4 ordered value combinations, there is at least one parallel edge of M in $C(C^*(e_{i,j}))$.

If $\left(\frac{1}{3}, 1, \frac{1}{3}\right)$ is the ordered value combination for $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot}))$, then $\tau(e_{i,j} \leftarrow C^*(e_{\cdot,j}))$ has an ordered value combination either $\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{3}\right)$ or $\left(\frac{1}{2}, \frac{1}{3}, \frac{1}{2}\right)$. The above Figures 3.20 and 3.23 show that there is at least one parallel edge of M in $C(C^*(e_{\cdot,j}))$. This proves the lemma. □

3.4.2.5 An upper bound on $\omega(e)$ for $e \in C(C^*(e_{i,j})) - \{e_{i,j}\}$

From Lemma 3.23, in the sequel we always consider the case $e_{i,j}$ is a singleton edge of M with $\omega(e_{i,j}) \geq 3$.

We walk through all the 27 configurations of $\mathcal{N}_p[C(C^*(e_{i,\cdot}))]$ to determine an upper bound on $\omega(e)$, for any $e \in C(C^*(e_{i,\cdot})) - \{e_{i,j}\}$.

Lemma 3.25. *For any edge $e \in C(C^*(e_{i,\cdot})) - \{e_{i,j}\}$, $|C(e^*_{h,\ell})| \geq 2$ for all edges $e^*_{h,\ell} \in C^*(e)$, if any one of the following five conditions holds:*

1. $|C(C^*(e_{i,\cdot}))| = |C^*(e_{i,\cdot})| = 3$.
2. $|C(C^*(e_{i,\cdot}))| = |C^*(e_{i,\cdot})| + 1 = 4$ *and there is an edge* $e^*_{i_1,j_1} \in C^*(e_{\cdot,j})$ *such that* $|C(e^*_{i_1,j_1})| = 1$.
3. $e \in C(e^*_{i_2,j_2})$ *for some* $e^*_{i_2,j_2} \in C^*(e_{i,\cdot})$ *with* $|C(e^*_{i_2,j_2})| = 2$, *and there is an edge* $e^*_{i_1,j_1} \in C^*(e_{\cdot,j})$ *such that* $|C(e^*_{i_1,j_1})| = 1$.
4. $e \in C(e^*_{i_1,j_1}) \cup C(e^*_{i_2,j_2})$ *for some* $e^*_{i_1,j_1}, e^*_{i_2,j_2} \in C^*(e_{i,\cdot})$ *with* $|C(e^*_{i_1,j_1}) \cup C(e^*_{i_2,j_2})| = 2$.
5. $e \in C(e^*_{i_2,j_2}) \cup C(e^*_{i_3,j_3})$ *for some* $e^*_{i_2,j_2}, e^*_{i_3,j_3} \in C^*(e_{i,\cdot})$ *with* $|C(e^*_{i_2,j_2}) \cup C(e^*_{i_3,j_3})| = 3$, *and there is an edge* $e^*_{i_1,j_1} \in C^*(e_{\cdot,j})$ *such that* $|C(e^*_{i_1,j_1})| = 1$.

And consequently, $\omega(e) \leq \frac{17}{6}$.

Proof. We prove by contradiction, and thus assume that there is an edge $e^*_{h,\ell} \in C^*(e)$ such that $|C(e^*_{h,\ell})| = 1$.

If the first condition holds, then the algorithm $\mathcal{LS}$ would replace the three edges of $C(C^*(e_{i,\cdot}))$ by $e^*_{h,\ell}$ and the three parallel edges of $C^*(e_{i,\cdot})$ to expand M, a contradiction.

If the second condition holds, then the algorithm $\mathcal{LS}$ would replace the four edges of $C(C^*(e_{i,.}))$ by $e^*_{i_1,j_1}$, $e^*_{h,\ell}$, and the three parallel edges in $C^*(e_{i,.})$ to expand M, a contradiction.

If the third condition holds, then the algorithm $\mathcal{LS}$ would replace the two edges of $C(e^*_{i_2,j_2})$ by $e^*_{i_2,j_2}$, $e^*_{i_1,j_1}$, $e^*_{h,\ell}$ to expand M, a contradiction.

If the fourth condition holds, then the algorithm $\mathcal{LS}$ would replace the two edges of $C(e^*_{i_1,j_1}) \cup C(e^*_{i_2,j_2})$ by $e^*_{i_1,j_1}$, $e^*_{i_2,j_2}$, $e^*_{h,\ell}$ to expand M, a contradiction.

If the fifth condition holds, then the algorithm $\mathcal{LS}$ would replace the three edges of $C(e^*_{i_2,j_2}) \cup C(e^*_{i_3,j_3})$ by $e^*_{i_2,j_2}$, $e^*_{i_3,j_3}$, $e^*_{i_1,j_1}$, $e^*_{h,\ell}$ to expand M, a contradiction.

Therefore, we proved that $|C(e^*_{h,\ell})| \geq 2$ for all edges $e^*_{h,\ell} \in C^*(e)$. It then follows from Lemma 3.16 that $\omega(e) \leq 5 \times \frac{1}{2} + \frac{1}{3} = \frac{17}{6}$. □

Lemma 3.26. *For each edge $e \in C(C^*(e_{i,.})) - \{e_{i,j}\}$ in Figures 3.15, 3.17, 3.20a, 3.20b, 3.22a, 3.23, 3.25a, 3.25b, 3.28a, 3.29a, 3.29b, 3.30a, 3.30b, $e_{j'-2}$ in Figure 3.21a, $e_{j'-1}$ in Figure 3.22b, $e_{j'}$ in Figure 3.24a, $e_{j'-1}$ in Figure 3.25c, and $e_{j'}$ in Figure 3.27, its total amount of tokens is $\omega(e) \leq \frac{17}{6}$.*

Proof. At least one of the five conditions in Lemma 3.25 applies to each of these edges. For example, in Figure 3.15, for the edge $e_{j'-2}$, the fourth condition of Lemma 3.25 holds by setting $(i_1, j_1) := (i-1, j'-1)$ and $(i_2, j_2) := (i, j')$; for the edge e_{i+2}, the fourth condition of Lemma 3.25 holds by setting $(i_1, j_1) := (i, j')$ and $(i_2, j_2) := (i+1, j'+1)$. □

Lemma 3.27. *For both the edges $e_{i+q}, e_{j'+q} \in C(C^*(e_{i,.}))$ shown in Figures 3.16, 3.18, 3.19, 3.21a, 3.22b, 3.22c, 3.24b, 3.25c, for some $q = 2$ or -2, the total amount of tokens for each of them is at most $\frac{35}{12}$.*

Proof. Consider the edge e_{i+q}. If it is a parallel edge of M, then it simply cannot fit into any of the 27 configurations shown in Figures 3.15–3.30, in which the edge $e_{i,j}$ is a singleton edge of M. (By "fitting into" it means the edge e_{i+q} takes up the role of the edge $e_{i,j}$ in the configuration.) If e_{i+q} is a singleton edge of M, we show next that due to the existence of the paired edge $e_{j'+q} \in M$, e_{i+q} cannot fit into any of the 27 configurations shown in Figures 3.15–3.30 either. This is done by using the edge combinations of $C(C^*(i, \mathbf{.}))$ and the existence of certain edges in $\mathcal{N}_p(C(C^*(e_{i,.})))$.

In more details, we first see that e_{i+q} can only possibly fit into 7 of the 27 configurations shown in Figures 3.22a, 3.24a, 3.25a, 3.25b, 3.26, 3.27, 3.28a, due to the existence of the edge $e_{j'+q} \in M$. Next, if it were fit in any of them, then in the fitted configuration there is an edge $e_{i-2} \in M$ but none of the five edges $e_{i-3}, e_{j'-3}, e_{j'-2}, e_{i-1}, e_{j'-1}$ can be in M. This last requirement rules out Figure 3.22a due to $e_{j'-3}, e_{i+3} \in \mathcal{N}_p(C(C^*(e_{i,\cdot})))$; it rules out Figure 3.24a due to $e_{j'-3} \in \mathcal{N}_p(C(C^*(e_{i,\cdot})))$; it rules out Figures 3.25a, 3.25b, 3.26 due to $e_{j'+1} \in C(C^*(e_{i,\cdot}))$ but none of $e_{i-2}, e_{j'-2}$ is in $C(C^*(e_{i,\cdot}))$; it rules out Figure 3.27 due to $e_{i-2}, e_{j'-2}, e_{i+2}, e_{j'+2} \in C(C^*(e_{i,\cdot}))$; and it rules out Figure 3.28a due to $e_{j'-1} \in C(C^*(e_{i,\cdot}))$ and $e_{i+3} \in \mathcal{N}_p(C(C^*(e_{i,\cdot})))$.

Therefore, $\omega(e_{i+q}) < 3$.

Using at most six values from $\{1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \frac{1}{6}\}$, the sum closest but less than 3 is $1 + \frac{1}{2} + \frac{1}{2} + \frac{1}{2} + \frac{1}{4} + \frac{1}{5} = \frac{59}{20}$. In order for $\tau(e_{i,j} \leftarrow C^*(e_{i,j}))$ to have a value combination $\left\{1, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \frac{1}{5}\right\}$, Lemma 3.15 says that the value combinations for $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot}))$ and $\tau(e_{i,j} \leftarrow C^*(e_{\cdot,j}))$ are $\left\{1, \frac{1}{2}, \frac{1}{2}\right\}$ and $\left\{\frac{1}{2}, \frac{1}{4}, \frac{1}{5}\right\}$. Furthermore, Lemmas 3.15 and 3.19 together state that the subsequent ordered value combinations are $\left(\frac{1}{2}, 1, \frac{1}{2}\right)$ and $\left(\frac{1}{2}, \frac{1}{4}, \frac{1}{5}\right)$. However, $\left(\frac{1}{2}, 1, \frac{1}{2}\right)$ requires $e_{i,j}$ to be a singleton edge of M, while $\left(\frac{1}{2}, \frac{1}{4}, \frac{1}{5}\right)$ implies $e_{i,j}$ is a parallel edge of M, a contradiction.

The second closest sum to 3 is $\frac{35}{12}$, that is the sum of the value combinations $\left\{1, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \frac{1}{6}\right\}$ (which can be ruled out similarly as in the last paragraph) and $\left\{1, \frac{1}{2}, \frac{1}{2}, \frac{1}{3}, \frac{1}{3}, \frac{1}{4}\right\}$. Therefore, $\omega(e_{i+q}) \leq \frac{35}{12}$. □

Lemma 3.28. *For each edge $e \in C(C^*(e_{i,j})) - \{e_{i,j}\}$ with $\omega(e_{i,j}) \geq 3$, we have $\omega(e) \leq \frac{35}{12}$, except for the following two cases where we have $\omega(e_{i,j}) = 3$:*

1. *in the configuration shown in Figure 3.28b, it is possible to have either $\omega(e_{j'-1}) = 3$ (when $|\mathcal{N}_p(e_{i-2})| \geq 1$) or $\omega(e_{j'+1}) = 3$ (when $|\mathcal{N}_p(e_{i+2})| \geq 1$), but not both;*
2. *in the configuration shown in Figure 3.28c, it is possible to have either $\omega(e_{j'-1}) = 3$ or $\omega(e_{i-2}) = 3$ (when $|\mathcal{N}_p(e_{j'-3})| \geq 1$), but not both.*

Proof. Recall that Lemma 3.26 settled all the edges of $C(C^*(e_{i,j})) - \{e_{i,j}\}$ in Figures 3.15, 3.17, 3.20a, 3.20b, 3.22a, 3.23, 3.25a, 3.25b, 3.28a, 3.29a, 3.29b, 3.30a, 3.30b, $e_{j'-2}$ in Figure 3.21a, $e_{j'-1}$ in Figure 3.22b, $e_{j'}$ in Figure 3.24a, $e_{j'-1}$ in Figure 3.25c, and $e_{j'}$ in Figure 3.27; Lemma 3.27 settled all the paired edges $e_{i+q}, e_{j'+q} \in C(C^*(e_{i,\cdot}))$ in Figures 3.16, 3.18, 3.19, 3.21a, 3.22b, 3.22c, 3.24b, 3.25c, for some $q = 2$ or -2, and all the edges known to be parallel, including $e_{j'-2}$ in Figure 3.16, $e_{j'+1}$ in Figure 3.18, $e_{j'+1}$ in Figure 3.21a, e_{i-2} in Figure 3.22b, $e_{j'-1}$ in Figure 3.22c, $e_{j'-2}$ in Figure 3.24a, $e_{j'+1}$ in

Figure 3.24b, $e_{j'+1}$ in Figure 3.25c, $e_{j'}$, $e_{j'+1}$, $e_{j'+2}$ in Figure 3.26, $e_{j'+1}$, $e_{j'+2}$ in Figure 3.28c, and $e_{j'-2}$, $e_{j'-1}$, $e_{j'+1}$, $e_{j'+2}$ in Figure 3.28d.

We therefore are left to prove the lemma for the edges not known to be parallel in Figures 3.24a, 3.26, 3.27, 3.28b, 3.28c. We deal with them separately in the following.

1. The edges e_{i+2}, $e_{j'+2}$ in Figure 3.24a and the edges e_{i-2}, $e_{j'-2}$, e_{i+2}, $e_{j'+2}$ in Figure 3.27, which can be settled the same.

   Consider the edge e_{i+2}, which can potentially fit into the configuration in Figure 3.24a or Figure 3.27. In either case, there is an edge $e^*_{i_1,j_1} \in C^*(e_{\cdot,j})$ such that $|C(e^*_{i_1,j_1})| = 1$ and there is an edge $e^*_{h_1,\ell_1} \in C^*(e_{i+2})$ such that $|C(e^*_{h_1,\ell_1})| = 1$. Then the algorithm $\mathcal{LS}$ would replace the four edges $e_{i,j}$, $e_{j'}$, e_{i+2}, $e_{j'+2}$ by the five edges $e^*_{i,j'}$, $e^*_{i+1,j'+1}$, $e^*_{i+2,j'+2}$, $e^*_{i_1,j_1}$, $e^*_{h_1,\ell_1}$ to expand M, a contradiction. In summary, e_{i+2} cannot fit into any of the 27 configurations shown in Figures 3.15–3.30 and thus $\omega(e_{i+2}) \leq \frac{35}{12}$.

2. The edge e_{i+2} in Figure 3.26.

   If e_{i+2} is to fit in, then it can fit only into the configuration in Figure 3.26. This suggests that $C(C^*(e_{i+2,\cdot})) = C(C^*(e_{i,\cdot}))$. Since there is an edge $e^*_{i_1,j_1} \in C^*(e_{\cdot,j})$ such that $|C(e^*_{i_1,j_1})| = 1$ and there is an edge $e^*_{h_1,\ell_1} \in C^*(e_{i+2})$ such that $|C(e^*_{h_1,\ell_1})| = 1$, the algorithm $\mathcal{LS}$ would replace the five edges of $C(C^*(e_{i,\cdot}))$ by any six edges from $\{e^*_{i-1,j'-1}, e^*_{i,j'}, e^*_{i+1,j'+1}, e^*_{i+2,j'+2}, e^*_{i+3,j'+3}, e^*_{i_1,j_1}, e^*_{h_1,\ell_1}\}$ to expand M, a contradiction. In summary, e_{i+2} cannot fit into any of the 27 configurations shown in Figures 3.15–3.30 and thus $\omega(e_{i+2}) \leq \frac{35}{12}$.

3. The edges e_{i-2} and e_{i+2} in Figure 3.28b, which can be settled the same.

   Consider the edge e_{i-2}, which can potentially fit into the configuration in Figure 3.28b or Figure 3.28c. In either case, all the four edges e_{i-2}, $e_{j'-1}$, $e_{i,j}$, $e_{j'+1}$ are singleton edges of M, and there is an edge $e^*_{i_1,j_1} \in C^*(e_{\cdot,j})$ such that $|C(e^*_{i_1,j_1})| = 1$ and there is an edge $e^*_{h_1,\ell_1} \in C^*(e_{i-2})$ such that $|C(e^*_{h_1,\ell_1})| = 1$. Then the algorithm $\mathcal{LS}$ would replace these four singleton edges of M by the edges $e^*_{i_1,j_1}$, $e^*_{h_1,\ell_1}$ and the two parallel edges $e^*_{i-1,j'-1}$, $e^*_{i,j'}$ to reduce the singleton edges of M, a contradiction. In summary, e_{i-2} cannot fit into any of the 27 configurations shown in Figures 3.15–3.30 and thus $\omega(e_{i-2}) \leq \frac{35}{12}$.

4. The edges $e_{j'-1}$ and $e_{j'+1}$ in Figure 3.28b, which can be settled the same.

   Consider the edge $e_{j'-1}$, which can potentially fit into the configuration in Figure 3.28b or Figure 3.28c. If $e_{j'-1}$ fits into the configuration in Figure 3.28b, then the same as in the last case the algorithm $\mathcal{LS}$ would be able to reduce the singleton edges of M, a contradiction. If $e_{j'-1}$ fits into the configuration in Figure 3.28c, then the edge e_{i-2} is a parallel edge of M. From $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$, we conclude that $\omega(e_{i,j}) \leq 3$, and consequently $\omega(e_{i,j}) = 3$ and $\omega(e_{j'-1}) = 3$.

It is easy to see that we cannot have both $\omega(e_{j'-1}) = \omega(e_{j'+1}) = 3$, since otherwise the algorithm $\mathcal{LS}$ would be able to expand M by swapping out the five edges of $C(C^*(e_{i,\cdot}))$, a contradiction.

In summary, we have either $\omega(e_{j'-1}) \leq \frac{35}{12}$ or $\omega(e_{j'-1}) = 3$, the latter of which implies $|\mathcal{N}_p(e_{i-2})| \geq 1$ and it is the first case stated in the lemma.

5. The edge e_{i-2} in Figure 3.28c.

   If e_{i-2} is to fit in, then it can fit only into the configuration in Figure 3.28b or Figure 3.28c. If e_{i-2} fits into the configuration in Figure 3.28b, then all the four edges $e_{i,j}$, $e_{j'-1}$, e_{i-2}, $e_{j'-3}$ are singleton edges of M. Since there is an edge $e^*_{i_1,j_1} \in C^*(e_{\cdot,j})$ such that $|C(e^*_{i_1,j_1})| = 1$ and there is an edge $e^*_{h_1,\ell_1} \in C^*(e_{i-2})$ such that $|C(e^*_{h_1,\ell_1})| = 1$, the algorithm $\mathcal{LS}$ would replace these four singleton edges of M by the edges $e^*_{i_1,j_1}$, $e^*_{h_1,\ell_1}$ and the two parallel edges $e^*_{i-1,j'-1}$, $e^*_{i-2,j'-2}$ to reduce the singleton edges, a contradiction. If e_{i-2} fits into the configuration in Figure 3.28c, then the edge $e_{j'-3}$ is a parallel edge of M. From $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$, we conclude that $\omega(e_{i,j}) \leq 3$, and consequently $\omega(e_{i,j}) = 3$ and $\omega(e_{i-2}) = 3$. In summary, we have either $\omega(e_{i-2}) \leq \frac{35}{12}$ or $\omega(e_{i-2}) = 3$, the latter of which implies $|\mathcal{N}_p(e_{j'-3})| \geq 1$.

6. The edge $e_{j'-1}$ in Figure 3.28c.

   If $e_{j'-1}$ is to fit in, then it can fit only into the configuration in Figure 3.28b or Figure 3.28c. If $e_{j'-1}$ fits into the configuration in Figure 3.28b, then the edge e_{i-2} is a singleton edge of M. If $e_{j'-1}$ fits into the configuration in Figure 3.28c, then the edge e_{i-2} is a parallel edge of M. From $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$, we conclude that $\omega(e_{i,j}) \leq 3$, and consequently $\omega(e_{i,j}) = 3$ and $\omega(e_{j'-1}) = 3$. In summary, we have either $\omega(e_{j'-1}) \leq \frac{35}{12}$ or $\omega(e_{j'-1}) = 3$.

It is also easy to see that we cannot have both $\omega(e_{j'-1}) = \omega(e_{i-2}) = 3$ in the last two items, since otherwise the algorithm $\mathcal{LS}$ would be able to expand M by swapping out the five edges of $C(C^*(e_{i,\cdot}))$, a contradiction. This is the second case stated in the lemma. We have proved the lemma. □

3.4.2.6 An upper bound on $\omega(e)$ for $e \in C(C^*(e_{i,j}))$ known to be parallel

In this section, we provide a better upper bound on the total amount of tokens received by an edge of $C(C^*(e_{i,j}))$ that is *known* to be parallel, for example, in Figure 3.16 the edge $e_{j'-2}$ is known parallel but the edge e_{i+2} is not. Also, from Lemma 3.28, in Figure 3.28b it is possible to have $\omega(e_{j'-1}) = 3$ when $|\mathcal{N}_p(e_{i-2})| \geq 1$; we therefore consider the edge e_{i-2} to be parallel too. For the same reason, we consider the edge e_{i+2} in Figure 3.28b to be parallel.

Lemma 3.29. *For each parallel edge $e \in C(C^*(e_{i,\cdot}))$ with $\omega(e_{i,j}) \geq 3$, we have $|C(e^*_{h,\ell})| \geq 2$ for all $e^*_{h,\ell} \in C^*(e)$, except for the following edges:*

1. *the edge $e_{j'+2}$ in Figures 3.18, 3.21a, 3.24b, 3.25c, 3.26,*
2. *the edges e_{i-2}, e_{i+2} in Figure 3.28b,*
3. *the edges $e_{j'+1}, e_{j'+2}$ in Figure 3.28c, and*
4. *the edges $e_{j'-1}, e_{j'-2}, e_{j'+1}, e_{j'+2}$ in Figure 3.28d.*

Proof. At least one of the five conditions in Lemma 3.25 applies to each of these edges. For example, in Figure 3.16, for the edge $e_{j'-2}$, the fourth condition of Lemma 3.25 holds by setting $(i_1, j_1) := (i-1, j'-1)$ and $(i_2, j_2) := (i, j')$. □

Among all the 27 configurations in Figures 3.15–3.30, we have the following two observations.

Observation 3.3. *If the edge e_{i-1} (or the edge $e_{j'-1}$) is a known parallel edge of M in $C(C^*(e_{i,\cdot}))$, and $e^*_{i-1,j'-1} \in M^*$, then $|C(e^*_{i-1,j'-1})| \geq 3$; if the edge $e_{j'}$ is a known parallel edge of M in $C(C^*(e_{i,\cdot}))$, and $e^*_{i,j'} \in M^*$, then $|C(e^*_{i,j'})| \geq 3$; if the edge e_{i+1} (or the edge $e_{j'+1}$) is a known parallel edge of M in $C(C^*(e_{i,\cdot}))$, and $e^*_{i+1,j'+1} \in M^*$, then $|C(e^*_{i+1,j'+1})| \geq 3$.*

Observation 3.4. *When $e^*_{i,j'} \in C^*(e_{i,\cdot})$, if the edge e_{i+p} (or $e_{j'+p}$, respectively) is a known parallel edge of M in $C(C^*(e_{i,\cdot}))$ for some $p = -2, 2$, then $|C(e^*_{i+p})| \geq 2$ (or $|C(e^*_{j'+p})| \geq 2$, respectively).*

Based on Lemmas 3.23, 3.25, and Observations 3.3 and 3.4, we can prove the following two lemmas.

Lemma 3.30. *For any pair of parallel edges $e_{h,\ell}, e_{h+1,\ell+1} \in C(C^*(e_{i,\cdot}))$, and an edge $e^* \in C^*(e_{h,\ell}) \cap C^*(e_{h+1,\ell+1})$, we have $|C(e^*)| \geq 3$ if one of the following three conditions holds.*

1. $|C(C^*(e_{i,\cdot}))| = |C^*(e_{i,\cdot})| = 3$.
2. $|C(C^*(e_{i,\cdot}))| = |C^*(e_{i,\cdot})| + 1 = 4$ *and there is an edge* $e^*_{i_1,j_1} \in C^*(e_{\cdot,j})$ *such that* $|C(e^*_{i_1,j_1})| = 1$.
3. $e_{h,\ell}, e_{h+1,\ell+1} \in C(e^*_{i_2,j_2}) \cup C(e^*_{i_3,j_3})$ *for some* $e^*_{i_2,j_2}, e^*_{i_3,j_3} \in C^*(e_{i,\cdot})$ *with* $|C(e^*_{i_2,j_2}) \cup C(e^*_{i_3,j_3})| = 3$*, and there is an edge* $e^*_{i_1,j_1} \in C^*(e_{\cdot,j})$ *such that* $|C(e^*_{i_1,j_1})| = 1$.

Proof. We prove by contradiction, and thus assume that there is an edge $e^* \in C^*(e_{h,\ell}) \cap C^*(e_{h+1,\ell+1})$ such that $C(e^*) = \{e_{h,\ell}, e_{h+1,\ell+1}\}$.

If the first condition holds, then it follows from Observation 3.3 that $e^* \notin C^*(e_{i,\cdot})$. In this case, the algorithm $\mathcal{LS}$ would replace the three edges of $C(C^*(e_{i,\cdot}))$ by e^* and the three edges of $C^*(e_{i,\cdot})$ to expand M, a contradiction.

If the second condition holds, then again it follows from Observation 3.3 that $e^* \notin C^*(e_{i,\cdot})$. Also, the edge $e^*_{i_1,j_1}$ is distinct from e^*. In this case, the algorithm $\mathcal{LS}$ would replace the four edges in $C(C^*(e_{i,\cdot}))$ by $e^*_{i_1,j_1}$, e^*, and the three edges of $C^*(e_{i,\cdot})$ to expand M, a contradiction.

If the third condition holds, then by Observation 3.3 the edge e^* is distinct from $e^*_{i_2,j_2}, e^*_{i_3,j_3}$, and the edge $e^*_{i_1,j_1}$ is distinct from e^*. In this case, the algorithm $\mathcal{LS}$ would replace the three edges of $C(e^*_{i_2,j_2}) \cup C(e^*_{i_3,j_3})$ by $e^*_{i_2,j_2}, e^*_{i_3,j_3}, e^*_{i_1,j_1}, e^*$ to expand M, a contradiction. □

Lemma 3.31. *For any pair of parallel edges* $e_{h,\ell}, e_{h+1,\ell+1}$ *where* $e_{h,\ell} \in C(C^*(e_{i,\cdot}))$*, there is at most one edge* $e^* \in C^*(e_{h,\ell}) \cap C^*(e_{h+1,\ell+1})$ *such that* $|C(e^*)| = 2$*, if one of the following six conditions holds.*

1. $e_{h+1,\ell+1} \notin C(C^*(e_{i,\cdot}))$ *and* $|C(C^*(e_{i,\cdot}))| = |C^*(e_{i,\cdot})| = 3$.
2. $e_{h,\ell} \in C(e^*_{i_1,j_1}) \cup C(e^*_{i_2,j_2})$, $e_{h+1,\ell+1} \notin C(e^*_{i_1,j_1}) \cup C(e^*_{i_2,j_2})$ *for some* $e^*_{i_1,j_1}, e^*_{i_2,j_2} \in C^*(e_{i,\cdot})$ *with* $|C(e^*_{i_1,j_1}) \cup C(e^*_{i_2,j_2})| = 2$.
3. $e_{h+1,\ell+1} \notin C(C^*(e_{i,\cdot}))$, $|C(C^*(e_{i,\cdot}))| = |C^*(e_{i,\cdot})| + 1 = 4$*, and there is an edge* $e^*_{i_1,j_1} \in C^*(e_{\cdot,j})$ *such that* $|C(e^*_{i_1,j_1})| = 1$.
4. $e_{h,\ell} \in C(e^*_{i_2,j_2}) \cup C(e^*_{i_3,j_3})$, $e_{h+1,\ell+1} \notin C(e^*_{i_2,j_2}) \cup C(e^*_{i_3,j_3})$ *for some* $e^*_{i_2,j_2}, e^*_{i_3,j_3} \in C^*(e_{i,\cdot})$, *with* $|C(e^*_{i_2,j_2}) \cup C(e^*_{i_3,j_3})| = 3$*, and there is an edge* $e^*_{i_1,j_1} \in C^*(e_{\cdot,j})$ *such that* $|C(e^*_{i_1,j_1})| = 1$.
5. $e_{h,\ell} \in C(e^*_{i_2,j_2})$, $e_{h+1,\ell+1} \notin C(e^*_{i_2,j_2})$ *for some* $e^*_{i_2,j_2} \in C^*(e_{i,\cdot})$ *with* $|C(e^*_{i_2,j_2})| = 2$*, and there is an edge* $e^*_{i_1,j_1} \in C^*(e_{\cdot,j})$ *such that* $|C(e^*_{i_1,j_1})| = 1$.
6. $e_{h+1,\ell+1} \in C(C^*(e_{i,\cdot}))$, $|C(C^*(e_{i,\cdot}))| = |C^*(e_{i,\cdot})| + 2 = 5$*, and there is an edge* $e^*_{i_1,j_1} \in C^*(e_{\cdot,j})$ *such that* $|C(e^*_{i_1,j_1})| = 1$.

Proof. We prove by contradiction, and thus assume that there are two edges $e^*_{h_1,\ell_1}, e^*_{h_2,\ell_2} \in C^*(e_{h,\ell}) \cap C^*(e_{h+1,\ell+1})$ such that $C(e^*_{h_1,\ell_1}) = C(e^*_{h_2,\ell_2}) = \{e_{h,\ell}, e_{h+1,\ell+1}\}$.

If the first condition holds, then due to $e_{h+1,\ell+1} \notin C(C^*(e_{i,\cdot}))$, none of $e^*_{h_1,\ell_1}, e^*_{h_2,\ell_2}$ is in $C^*(e_{i,\cdot})$. In this case, the algorithm $\mathcal{LS}$ would replace the edge $e_{h+1,\ell+1}$ and the three edges of $C(C^*(e_{i,\cdot}))$ by $e^*_{h_1,\ell_1}, e^*_{h_2,\ell_2}$ and the three edges of $C^*(e_{i,\cdot})$ to expand M, a contradiction.

The other five conditions can be similarly proved by this kind of contradiction. □

Using Lemmas 3.30 and 3.31, we can prove a better upper bound on $\omega(e)$ for those edges stated in Lemma 3.29. This better bound is $\frac{5}{2}$, a reduction from $\frac{17}{6}$ stated in Lemma 3.25.

Lemma 3.32. *For any parallel edge $e \in C(C^*(e_{i,\cdot}))$ discussed in Lemma 3.29, its total amount of tokens received $\omega(e)$ can be better bounded, in particular, $\omega(e) \leq \frac{5}{2}$.*

Proof. We enumerate through all these edges in the following:

1. In Figure 3.15, we have $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot})) = \left(\frac{1}{2}, 1, \frac{1}{2}\right)$. For the edge $e_{j'-2}$, it is parallel to $e_{j'-3} \notin C(C^*(e_{i,\cdot}))$. By the condition 1 of Lemma 3.31 and Lemma 3.29, $\omega(e_{j'-2}) \leq 3\left(\frac{1}{2} + \frac{1}{3}\right) = \frac{5}{2} = 2.5$. The same argument applies to the edge e_{i+2}.

   In the rest of the proof, we point out only the condition used in the argument to avoid repetition.

2. In Figure 3.16, $\omega(e_{j'-2}) \leq 3\left(\frac{1}{2} + \frac{1}{3}\right) = \frac{5}{2} = 2.5$, due to the condition 2 of Lemma 3.31.

3. In Figure 3.17, $\omega(e_{j'+1}), \omega(e_{j'+2}) \leq 2\left(\frac{1}{2} + 2 \times \frac{1}{3}\right) = \frac{7}{3} \approx 2.333$, due to the condition 1 of Lemma 3.30.

4. In Figure 3.18, $\omega(e_{j'+1}) \leq \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{3}\right) + \left(2 \times \frac{1}{2} + \frac{1}{3}\right) = \frac{29}{12} \approx 2.417$, due to the condition 2 of Lemma 3.30.

5. In Figure 3.20a, $\omega(e_{j'+1}), \omega(e_{j'+2}) \leq 2\left(\frac{1}{2} + 2 \times \frac{1}{3}\right) = \frac{7}{3} \approx 2.333$, due to the condition 2 of Lemma 3.30.

6. In Figure 3.20b, $\omega(e_{j'-2}) \leq 3\left(\frac{1}{2} + \frac{1}{3}\right) = \frac{5}{2} = 2.5$, due to the condition 3 of Lemma 3.31;

   $\omega(e_{i+2}) \leq 4 \times \frac{1}{3} + 2 \times \frac{1}{2} = \frac{7}{3} \approx 2.333$, due to the condition 3 of Lemma 3.31.

7. In Figure 3.21a, $\omega(e_{j'+1}) \leq \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{3}\right) + \left(2 \times \frac{1}{2} + \frac{1}{3}\right) = \frac{29}{12} \approx 2.417$, due to the condition 4 of Lemma 3.31.

8. In Figure 3.22a, $\omega(e_{j'-2}), \omega(e_{i+2}) \leq 4 \times \frac{1}{3} + 2 \times \frac{1}{2} = \frac{7}{3} \approx 2.333$, due to the condition 3 of Lemma 3.31.

9. In Figure 3.22b, $\omega(e_{j'-1}), \omega(e_{j'-2}) \leq 2\left(\frac{1}{2} + 2 \times \frac{1}{3}\right) = \frac{7}{3} \approx 2.333$, due to the condition 3 of Lemma 3.30.
10. In Figure 3.22c, $\omega(e_{i-2}) \leq 2\left(\frac{1}{2} + 2 \times \frac{1}{3}\right) = \frac{7}{3} \approx 2.333$, due to the condition 4 of Lemma 3.31.
11. In Figure 3.23, $\omega(e_{j'+1}), \omega(e_{j'+2}) \leq 5 \times \frac{1}{3} + \frac{1}{2} = \frac{13}{6} \approx 2.167$, due to the condition 2 of Lemma 3.30.
12. In Figure 3.24a, $\omega(e_{j'-2}) \leq 4 \times \frac{1}{3} + 2 \times \frac{1}{2} = \frac{7}{3} \approx 2.333$, due to the condition 4 of Lemma 3.31.
13. In Figure 3.24b, $\omega(e_{j'+1}) \leq \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{3}\right) + \left(2 \times \frac{1}{2} + \frac{1}{3}\right) = \frac{29}{12} \approx 2.417$, due to the condition 5 of Lemma 3.31.
14. In Figure 3.25a, $\omega(e_{j'}), \omega(e_{j'+2}) \leq \left(\frac{1}{2} + \frac{1}{3} + \frac{1}{4}\right) + \left(\frac{1}{2} + 2 \times \frac{1}{3}\right) = \frac{9}{4} = 2.25$, due to the condition 2 of Lemma 3.30;

    $\omega(e_{j'+1}) \leq \left(\frac{1}{3} + \frac{1}{4} + \frac{1}{3}\right) + \left(\frac{1}{2} + 2 \times \frac{1}{3}\right) = \frac{25}{12} \approx 2.083$, due to the condition 2 of Lemma 3.30.
15. In Figure 3.25b, $\omega(e_{j'}), \omega(e_{i+2}) \leq \left(\frac{1}{2} + \frac{1}{3} + \frac{1}{4}\right) + \left(\frac{1}{2} + 2 \times \frac{1}{3}\right) = \frac{9}{4} = 2.25$, due to the condition 2 of Lemma 3.30;

    $\omega(e_{j'+1}) \leq \left(\frac{1}{3} + \frac{1}{4} + \frac{1}{3}\right) + \left(\frac{1}{2} + 2 \times \frac{1}{3}\right) = \frac{25}{12} \approx 2.083$, due to the condition 2 of Lemma 3.30.
16. In Figure 3.25c, $\omega(e_{j'+1}) \leq \left(\frac{1}{2} + \frac{1}{3} + \frac{1}{4}\right) + \left(\frac{1}{2} + 2 \times \frac{1}{3}\right) = \frac{9}{4} = 2.25$, due to the condition 4 of Lemma 3.31.
17. In Figure 3.26, $\omega(e_{j'}) \leq \left(\frac{1}{2} + \frac{1}{3} + \frac{1}{5}\right) + \left(\frac{1}{2} + 2 \times \frac{1}{3}\right) = \frac{11}{5} = 2.2$, due to the condition 3 of Lemma 3.30;

    $\omega(e_{j'+1}) \leq \left(2 \times \frac{1}{3} + \frac{1}{5}\right) + \left(\frac{1}{2} + 2 \times \frac{1}{3}\right) = \frac{61}{30} \approx 2.033$, due to the condition 3 of Lemma 3.30.
18. In Figure 3.28a, $\omega(e_{j'}) \leq 5 \times \frac{1}{3} + \frac{1}{2} = \frac{13}{6} \approx 2.167$, due to the condition 2 of Lemma 3.30;

    $\omega(e_{j'-1}) \leq 2\left(\frac{1}{2} + 2 \times \frac{1}{3}\right) = \frac{7}{3} \approx 2.333$, due to the condition 2 of Lemma 3.30;

    $\omega(e_{i+2}) \leq 4 \times \frac{1}{3} + 2 \times \frac{1}{2} = \frac{7}{3} \approx 2.333$, due to the condition 3 of Lemma 3.31.
19. In Figure 3.29a, $\omega(e_{j'-2}) \leq 3 \times \frac{1}{2} + 3 \times \frac{1}{3} = \frac{5}{2} = 2.5$, due to the condition 4 of Lemma 3.31.
20. In Figure 3.29b, $\omega(e_{j'+1}) \leq 2 \times \frac{1}{2} + 3 \times \frac{1}{3} = 2$, due to the condition 4 of Lemma 3.31 and no edge of M^* incident on $j' + 1$.

21. In Figure 3.30a, $\omega(e_{j''}), \omega(e_{j'''}) \le 1 + 3 \times \frac{1}{2} = \frac{5}{2} = 2.5$, simply due to no edge of M^* incident on j'' and $e_{j'''}$ where $j''' = j'' + 1$.

Note the maximum value among the above is $\frac{5}{2} = 2.5$. The lemma is proved. □

The next lemma is on the parallel edges excluded from Lemma 3.32.

Lemma 3.33. *For each of following parallel edge $e \in C(C^*(e_{i,\cdot}))$ with $\omega(e_{i,j}) \ge 3$, we have*

1. *for the edge $e_{j'+2}$ in Figures 3.18, 3.21a, 3.24b, $\omega(e_{j'+2}) \le \frac{29}{12}$;*
2. *for the edge $e_{j'+2}$ in Figures 3.25c, 3.26, $\omega(e_{j'+2}) \le \frac{35}{12}$;*
3. *for the edges e_{i-2}, e_{i+2} in Figure 3.28b, either $\omega(e_{i-2}), \omega(e_{i+2}) \le \frac{35}{12}$, or $\omega(e_{i-2}) \le \frac{13}{6}$ when $\omega(e_{j'-1}) = 3$, or $\omega(e_{i+2}) \le \frac{13}{6}$ when $\omega(e_{j'+1}) = 3$;*
4. *for the edges $e_{j'+1}, e_{j'+2}$ in Figure 3.28c, $\omega(e_{j'+1}) \le \frac{13}{6}$ and $\omega(e_{j'+2}) \le \frac{7}{3}$;*
5. *for the edges $e_{j'-1}, e_{j'-2}, e_{j'+1}, e_{j'+2}$ in Figure 3.28d, $\omega(e) \le \frac{35}{12}$.*

Proof. We first note that in items 2) and 5) we do not succeed in getting a better bound, and thus quote the existing bounds. More specifically, for the edge $e_{j'+2}$ in Figure 3.25c, $\omega(e_{j'+2}) \le \frac{35}{12}$ is from Lemma 3.27; for the others, $\omega(e) \le \frac{35}{12}$ is from Lemma 3.28.

In the rest of the proof, we let $e^*_{i_1,j_1}$ denote the edge of $C^*(e_{i,j})$ such that $|C(e^*_{i_1,j_1})| = 1$.

1. The edge $e_{j'+2}$ in Figures 3.18, 3.21a, 3.24b.

   One sees that for the edge $e_{j'+2}$ in Figure 3.24b, its $\omega(e_{j'+2})$ is larger (or worst) when $\mathcal{N}_p(e_{j'+2}) = \emptyset$ than when $\mathcal{N}_p(e_{j'+2}) \ne \emptyset$. We therefore consider the worse case when $\mathcal{N}_p(e_{j'+2}) = \emptyset$; this way, all three edges can be discussed exactly the same (ignoring the incidence information of $i-2$ and $j'-2$ in M).

   Assume the edge $e_{j'+2}$ is incident on h, i.e., $e_{h,j'+2} := e_{j'+2}$. We consider the case where $|C^*(e_{h,j'+2})| \ge 5$, as otherwise $\omega(e_{j'+2}) \le 1 + \frac{1}{2} + \frac{1}{2} + \frac{1}{4} < \frac{29}{12}$. When there is an edge $e^*_{h_1,\ell_1} \in C^*(e_{h,j'+2})$ such that $|C(e^*_{h_1,\ell_1})| = 1$, then either $h_1 = h+1$ or $\ell_1 = j'+3$. If there is an edge e^*_h of M^* incident on h, then $|C(e^*_h)| \ge 3$, since otherwise the algorithm $\mathcal{LS}$ would be able to expand M by swapping the four edges $e_{i,j}, e_{h-1,j'+1}, e_{h,j'+2}, e_{i+2}$ of $C(C^*(e_{i,\cdot}))$ by the five edges $e^*_h, e^*_{h_1,\ell_1}, e^*_{i_1,j_1}, e^*_{i,j'}, e^*_{i+1,j'+1}$; for the same reason, if there is an edge e^*_{h-1} of M^* incident on $h-1$, then $|C(e^*_{h-1})| \ge 3$. These together say that the value combination of $\tau(e_{h,j'+2} \leftarrow C^*(e_{h,\cdot}))$ is impossible to have two values $\ge \frac{1}{2}$. Therefore, if $|C^*(e_{h,j'+2})| = 5$, we have $\omega(e_{h,j'+2}) \le 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{3} + \frac{1}{4} = \frac{29}{12}$, due to Lemmas 3.14, 3.16, and $|C(e^*_{i+1,j'+1})| = 4$ and $|C(e^*_{i+2,j'+2})| \ge 3$. When there is no edge $e^* \in C^*(e_{h,j'+2})$ such that $|C(e^*)| = 1$, if $|C^*(e_{h,j'+2})| = 5$, then we have $\omega(e_{h,j'+2}) \le 4 \times \frac{1}{2} + \frac{1}{4} < \frac{29}{12}$, due to $|C(e^*_{i+1,j'+1})| = 4$.

We next consider $|C^*(e_{h,j'+2})| = 6$ and $C^*(e_{h,j'+2}) = \{e^*_{i+1,j'+1}, e^*_{i+2,j'+2}, e^*_{i+3,j'+3}, e^*_{h-1,\ell-1}, e^*_{h,\ell}, e^*_{h+1,\ell+1}\}$, for some ℓ. If there is an edge of $C^*(e_{h,j'+2})$ conflicts only one edge of M, then this edge has to be $e^*_{h+1,\ell+1}$. Then we have $|C(e^*_{i+2,j'+2})| \ge 4$, since otherwise the algorithm $\mathcal{LS}$ would replace the four edges $e_{i,j}$, e_{i+2}, $e_{h-1,j'+1}$, $e_{h,j'+2}$ by the five edges $e^*_{i_1,j_1}, e^*_{i,j'}, e^*_{i+1,j'+1}, e^*_{i+2,j'+2}, e^*_{h+1,\ell+1}$ to expand M, a contradiction. For a similar reason, we have $|C(e^*_{h,\ell})| \ge 3$ and then $|C(e^*_{h-1,\ell-1})| \ge 4$. It follows that $|C(e^*_{i+3,j'+3})| \ge 3$ and $|C(e^*_{h,\ell})| = 3$. Therefore, we have $\omega(e_{h,j'+2}) \le \left(2 \times \frac{1}{4} + \frac{1}{3}\right) + \left(1 + \frac{1}{3} + \frac{1}{4}\right) = \frac{29}{12}$. When there is no edge of $C^*(e_{h,j'+2})$ conflicts only one edge of M, since we cannot have both $|C(e^*_{h,\ell})| = |C(e^*_{h-1,\ell-1})| = 2$, we have $\omega(e_{h,j'+2}) \le \left(\frac{1}{4} + \frac{1}{3} + \frac{1}{2}\right) + \left(2 \times \frac{1}{2} + \frac{1}{3}\right) = \frac{29}{12}$.

Note that in the above proof we did not use the incidence information of $i-2$ and $j'-2$ in M. In summary, we have $\omega(e_{j'+2}) \le \frac{29}{12} \approx 2.417$ in Figures 3.18, 3.21a, 3.24b.

3. The edges e_{i-2} and e_{i+2} in Figure 3.28b, which can be discussed exactly the same.

   Recall that there is an edge $e^*_{i_1,j_1} \in C^*(e_{\cdot,j})$ such that $|C(e^*_{i_1,j_1})| = 1$.

   From Lemma 3.28, we know that when $\omega(e_{j'-1}) = 3$, e_{i-2} has to be a parallel edge of M; if e_{i-2} is a singleton then $\omega(e_{i-2}) \le \frac{35}{12}$, and if $\omega(e_{j'-1}) < 3$, then $\omega(e_{j'-1}) \le \frac{35}{12}$. We consider in the following $\omega(e_{j'-1}) = 3$.

   Assume the edge $e_{j'-1}$ is incident on h', i.e., $e_{h',j'-1} := e_{j'-1}$. Thus we have $|C(e^*_{i-2,j'-2})| = 3$ (that is, no edge of M incident on $j'-3$), $|C(C^*(e_{\cdot,j'-1}))| = 5$, and there is an edge $e^*_{h'_1,j'_1} \in C^*(e_{h',\cdot})$ such that $|C(e^*_{h'_1,j'_1})| = 1$.

   Assume the edge e_{i-2} is incident on ℓ, i.e., $e_{i-2,\ell} := e_{i-2}$. We observe first that if there is an edge e^*_ℓ of M^* incident on ℓ, then $|C(e^*_\ell)| \ge 3$, since otherwise the algorithm $\mathcal{LS}$ would be able to expand M by swapping the five edges of $C(C^*(e_{\cdot,j'-1}))$ by six edges including e^*_ℓ; for the same reason, if there is an edge $e^*_{\ell-1}$ of M^* incident on $\ell-1$, then $|C(e^*_{\ell-1})| \ge 3$; if there is an edge $e^*_{i-3} = e^*_{i-3,j'-3}$ of M^* incident on $i-3$, then $|C(e^*_{i-3})| \ge 3$. This says that the value combination of $\tau(e_{i-2,\ell} \leftarrow C^*(e_{i-2,\ell}))$ is impossible to have two values $\ge \frac{1}{2}$.

   If there is an edge of $C^*(e_{i-2,\ell})$ conflicting only one edge of M, then this edge has to be $e^*_{\ell+1}$. In this case, the algorithm $\mathcal{LS}$ would replace the five edges of $C(C^*(e_{\cdot,j'-1}))$ by the edges $e^*_{\ell+1}, e^*_{h'_1,j'_1}, e^*_{i_1,j_1}$ and the three edges of $C^*(e_{\cdot,j'-1})$ to expand M, a contradiction. Therefore, there is no edge of $C^*(e_{i-2,\ell})$ conflicting only one edge of M. It follows that if $|C^*(e_{i-2,\ell})| \le 5$, we have $\omega(e_{i-2,\ell}) \le \frac{1}{2} + 4 \times \frac{1}{3} = \frac{11}{6}$.

We next assume $C^*(e_{i-2,\ell}) = \{e^*_{i-1,j'-1}, e^*_{i-2,j'-2}, e^*_{i-3,j'-3}, e^*_{h-1,\ell-1}, e^*_{h,\ell}, e^*_{h+1,\ell+1}\}$, for some h. Note that every one of $e^*_{i-3,j'-3}, e^*_{h-1,\ell-1}, e^*_{h,\ell}$ conflicts both edges $e_{i-3,\ell-1}$ and $e_{i-2,\ell}$. If there is one of them conflicting only these two edges of M, then the five edges of $C(C^*(e_{\cdot,j'-1}))$ can be replaced by six edges to expand M. It thus follows that all three $|C(e^*_{i-3,j'-3})|$, $|C(e^*_{h-1,\ell-1})|$, $|C(e^*_{h,\ell})| \geq 3$; and subsequently $\omega(e_{i-2,\ell}) \leq 5 \times \frac{1}{3} + \frac{1}{2} = \frac{13}{6} \approx 2.167$.

In summary, we have for e_{i-2} in Figure 3.28b: if $\omega(e_{j'-1}) \leq \frac{35}{12}$ then $\omega(e_{i-2}) \leq \frac{35}{12}$; if $\omega(e_{j'-1}) = 3$ then $\omega(e_{i-2}) \leq \frac{13}{6}$. Similarly, we have for e_{i+2} in Figure 3.28b: if $\omega(e_{j'+1}) \leq \frac{35}{12}$ then $\omega(e_{i+2}) \leq \frac{35}{12}$; if $\omega(e_{j'+1}) = 3$ then $\omega(e_{i+2}) \leq \frac{13}{6}$.

4.1. The edge $e_{j'+1}$ in Figure 3.28c.

Recall that there is an edge $e^*_{i_1,j_1} \in C^*(e_{\cdot,j})$ such that $|C(e^*_{i_1,j_1})| = 1$.

From Lemma 3.28, we know that if $\omega(e_{j'-1}) < 3$, then $\omega(e_{j'-1}) \leq \frac{35}{12}$. We consider in the following $\omega(e_{j'-1}) = 3$. Assume the edge $e_{j'-1}$ is incident on h', i.e., $e_{h',j'-1} := e_{j'-1}$. From $\omega(e_{j'-1}) = 3$, there is an edge $e^*_{h'_1,j'_1} \in C^*(e_{h',\cdot})$ such that $|C(e^*_{h'_1,j'_1})| = 1$.

Assume the edge $e_{j'+1}$ is incident on h, i.e., $e_{h,j'+1} := e_{j'+1}$. We observe first that if there is an edge e^*_h of M^* incident on h, then $|C(e^*_h)| \geq 3$, since otherwise the algorithm $\mathcal{LS}$ would be able to expand M by swapping the five edges of $C(C^*(e_{i,\cdot}))$ by six edges including e^*_h; for the same reason, if there is an edge e^*_{h+1} of M^* incident on $h+1$, then $|C(e^*_{h+1})| \geq 3$. This says that the value combination of $\tau(e_{h,j'+2} \leftarrow C^*(e_{h,\cdot}))$ is impossible to have two values $\geq \frac{1}{2}$.

If there is an edge of $C^*(e_{h,j'+1})$ conflicting only one edge of M, then this edge has to be e^*_{h-1}. In this case, the algorithm $\mathcal{LS}$ would replace the five edges of $C(C^*(e_{i,\cdot}))$ by the edges $e^*_{h-1}, e^*_{h'_1,j'_1}, e^*_{i_1,j_1}$ and the three edges of $C^*(e_{i,\cdot})$ to expand M, a contradiction. Therefore, there is no edge of $C^*(e_{h,j'+1})$ conflicting only one edge of M. It follows that if $|C^*(e_{h,j'+1})| \leq 5$, we have $\omega(e_{h,j'+1}) \leq \frac{1}{2} + \frac{1}{3} + \frac{1}{2} + \frac{1}{3} + \frac{1}{3} = 2$.

We next assume $C^*(e_{h,j'+1}) = \{e^*_{h-1,\ell-1}, e^*_{h,\ell}, e^*_{h+1,\ell+1}, e^*_{i,j'}, e^*_{i+1,j'+1}, e^*_{i+2,j'+2}\}$, for some ℓ. Note that every one of $e^*_{i+2,j'+2}, e^*_{h,\ell}, e^*_{h+1,\ell+1}$ conflicts both edges $e_{h,j'+1}$ and $e_{h+1,j'+2}$. If there is one of them conflicting only these two edges of M, then the five edges of $C(C^*(e_{i,\cdot}))$ can be replaced by six edges to expand M. It thus follows that all three $|C(e^*_{i+2,j'+2})|$, $|C(e^*_{h,\ell})|$, $|C(e^*_{h+1,\ell+1})| \geq 3$; and subsequently $\omega(e_{h,j'+1}) \leq 5 \times \frac{1}{3} + \frac{1}{2} = \frac{13}{6} \approx 2.167$.

From Lemma 3.28, we also know that if $\omega(e_{i-2}) < 3$, then $\omega(e_{i-2}) \leq \frac{35}{12}$. We consider $\omega(e_{i-2}) = 3$, which implies that there is an edge $e^*_{h'_1,j'_1} \in C^*(e_{i-2})$ such that $|C(e^*_{h'_1,j'_1})| = 1$. It follows by the same argument as in the above that $\omega(e_{j'+1}) \leq \frac{13}{6}$.

In summary, we have for $e_{j'+1}$ in Figure 3.28c: if both $\omega(e_{j'-1}), \omega(e_{i-2}) \leq \frac{35}{12}$ then $\omega(e_{j'+1}) \leq \frac{35}{12}$ too; otherwise, $\omega(e_{j'+1}) \leq \frac{13}{6}$.

4.2. The edge $e_{j'+2}$ in Figure 3.28c.

The argument here is the same as in the last item (4.1) to consider $\omega(e_{j'-1}) = 3$.

Assume the edge $e_{j'+2}$ is incident on h, i.e., $e_{h,j'+2} := e_{j'+2}$.

We observe first, for the same reasons, that if there is an edge e^*_h of M^* incident on h, then $|C(e^*_h)| \geq 3$; if there is an edge e^*_{h+1} of M^* incident on $h+1$, then $|C(e^*_{h+1})| \geq 3$; there is no edge of $C^*(e_{h,j'+2})$ conflicting only one edge of M; if there is an edge $e^*_{j'+2}$ of M^* incident on $j'+2$, then $|C(e^*_{j'+2})| \geq 2$; if there is an edge $e^*_{j'+3}$ of M^* incident on $j'+3$, then $|C(e^*_{j'+3})| \geq 2$. These together say that the value combination of $\tau(e_{h,j'+2} \leftarrow C^*(e_{h,j'+2}))$ is impossible to have a value 1, and it is impossible to have three values $\geq \frac{1}{2}$. It follows that $\omega(e_{h,j'+2}) \leq 2 \times \frac{1}{2} + 4 \times \frac{1}{3} = \frac{7}{3} \approx 2.333$.

In summary, we have for $e_{j'+2}$ in Figure 3.28c: if both $\omega(e_{j'-1}), \omega(e_{i-2}) \leq \frac{35}{12}$ then $\omega(e_{j'+2}) \leq \frac{35}{12}$ too; otherwise, $\omega(e_{j'+1}) \leq \frac{7}{3}$.

This finishes the proof of the lemma. □

Lemma 3.18 states the 12 possible value combinations of $\tau(e_{i,j} \leftarrow C^*(e_{i,\cdot}))$ with $\omega(e_{i,j}) \geq 3$, which are $\{1, \frac{1}{2}, \frac{1}{2}\}$, $\{1, \frac{1}{2}, \frac{1}{3}\}$, $\{1, \frac{1}{2}, \frac{1}{4}\}$, $\{1, \frac{1}{3}, \frac{1}{3}\}$, $\{\frac{1}{2}, \frac{1}{2}, \frac{1}{3}\}$, $\{\frac{1}{2}, \frac{1}{2}, \frac{1}{4}\}$, $\{\frac{1}{2}, \frac{1}{3}, \frac{1}{3}\}$, $\{\frac{1}{2}, \frac{1}{3}, \frac{1}{4}\}$, $\{\frac{1}{2}, \frac{1}{3}, \frac{1}{5}\}$, $\{\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\}$, $\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\}$, and $\{\frac{1}{2}, \frac{1}{2}, 0\}$. We next count the minimum number of known-to-be parallel edges of M in $C(C^*(e_{i,\cdot}))$ and use Lemmas 3.32 and 3.33 to upper bound their $\omega(\cdot)$ values respectively, for each combination.

1. $\{1, \frac{1}{2}, \frac{1}{2}\}$: there are at least 2 parallel edges of M, each with $\omega(\cdot) \leq \frac{5}{2} = 2.5$ (by Lemma 3.32);
2. $\{1, \frac{1}{2}, \frac{1}{3}\}$: there is at least 1 parallel edge of M, with $\omega(\cdot) \leq \frac{5}{2} = 2.5$ (by Lemma 3.32);
3. $\{1, \frac{1}{2}, \frac{1}{4}\}$: there are at least 2 parallel edges of M, each with $\omega(\cdot) \leq \frac{29}{12} \approx 2.417$ (by Lemmas 3.32, 3.33);
4. $\{1, \frac{1}{3}, \frac{1}{3}\}$: no parallel edge;
5. $\{\frac{1}{2}, \frac{1}{2}, \frac{1}{3}\}$: there are at least 2 parallel edges of M, each with $\omega(\cdot) \leq \frac{5}{2} = 2.5$ (by Lemma 3.32);
6. $\{\frac{1}{2}, \frac{1}{2}, \frac{1}{4}\}$: there are at least 2 parallel edges of M, each with $\omega(\cdot) \leq \frac{29}{12} \approx 2.417$ (by Lemmas 3.32, 3.33);

7. $\left\{\frac{1}{2},\frac{1}{3},\frac{1}{3}\right\}$: there is at least 1 parallel edge of M, with $\omega(\cdot) \le \frac{7}{3} \approx 2.333$ (by Lemma 3.32);
8. $\left\{\frac{1}{2},\frac{1}{3},\frac{1}{4}\right\}$: there are three possible cases,
   (a) there is at least 1 parallel edge of M with $\omega(\cdot) \le \frac{7}{3} \approx 2.333$ (by Lemma 3.32), or
   (b) there are at least 2 parallel edges of M, each with $\omega(\cdot) \le \frac{29}{12} \approx 2.417$ (by Lemmas 3.32, 3.33), or
   (c) there are at least 2 parallel edges of M, one with $\omega(\cdot) \le \frac{9}{4} = 2.25$ and the other with $\omega(\cdot) \le \frac{35}{12} \approx 2.917$ (by Lemmas 3.32, 3.33);
9. $\left\{\frac{1}{2},\frac{1}{3},\frac{1}{5}\right\}$: there are at least 2 parallel edges of M, each with $\omega(\cdot) \le \frac{11}{5} = 2.2$ (by Lemma 3.32);
10. $\left\{\frac{1}{2},\frac{1}{4},\frac{1}{4}\right\}$: no parallel edge;
11. $\left\{\frac{1}{3},\frac{1}{3},\frac{1}{3}\right\}$: there are five possible cases,
    (a) there is no singleton edge other than $e_{i,j}$ with $\omega(\cdot) \ge 3$, no parallel edge;
    (b) there is no singleton edge other than $e_{i,j}$ with $\omega(\cdot) \ge 3$, but there is at least 1 parallel edge of M with $\omega(\cdot) \le \frac{35}{12} \approx 2.917$ (by Lemmas 3.32, 3.33),
    (c) there is no singleton edge other than $e_{i,j}$ with $\omega(\cdot) \ge 3$, but there are at least 2 parallel edges of M, each with $\omega(\cdot) \le \frac{35}{12} \approx 2.917$ (by Lemmas 3.32, 3.33),
    (d) there is one singleton edge other than $e_{i,j}$ with $\omega(\cdot) = 3$, accompanied by at least 1 parallel edge of M with $\omega(\cdot) \le \frac{13}{6} \approx 2.167$ (by Lemma 3.33),
    (e) there is one singleton edge other than $e_{i,j}$ with $\omega(\cdot) = 3$, accompanied by at least 2 parallel edges of M, one with $\omega(\cdot) \le \frac{13}{6} \approx 2.167$ and the other with $\omega(\cdot) \le \frac{7}{3} \approx 2.333$ (by Lemma 3.33);
12. $\left\{\frac{1}{2},\frac{1}{2},0\right\}$: no parallel edge.

Lemma 3.17 states the 8 possible value combinations of $\tau(e_{i,j} \leftarrow C^*(e_{i,j}))$ with $\omega(e_{i,j}) \ge 3$, which are $\left\{1,\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{3}\right\}$, $\left\{1,\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{4}\right\}$, $\left\{1,\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{3},\frac{1}{3}\right\}$, $\left\{1,\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{3},\frac{1}{4}\right\}$, $\left\{1,\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{3},\frac{1}{5}\right\}$, $\left\{1,\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{4},\frac{1}{4}\right\}$, $\left\{1,\frac{1}{2},\frac{1}{2},\frac{1}{3},\frac{1}{3},\frac{1}{3}\right\}$, and $\left\{1,\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2},0\right\}$. These combinations give rise to $\omega(e_{i,j}) = \frac{10}{3}, \frac{13}{4}, \frac{19}{6}, \frac{37}{12}, \frac{91}{30}$, 3, 3 and 3 respectively. Based on the above list, we conclude the minimum number of known-to-be parallel edges of M in $C(C^*(e_{i,j}))$ for each combination, using the cut-off upper bound 2.5 on their $\omega(\cdot)$ values, as follows.

1. $\left\{1,\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{3}\right\}$ ($\omega(e_{i,j}) = \frac{10}{3}$): there are at least 4 parallel edges of M;
2. $\left\{1,\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{4}\right\}$ ($\omega(e_{i,j}) = \frac{13}{4}$): there are at least 4 parallel edges of M;
3. $\left\{1,\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{3},\frac{1}{3}\right\}$ ($\omega(e_{i,j}) = \frac{19}{6}$): there are at least 3 parallel edges of M;

4. $\left\{1, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}\right\}$ ($\omega(e_{i,j}) = \frac{37}{12}$): there are at least 3 parallel edges of M;
5. $\left\{1, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{3}, \frac{1}{5}\right\}$ ($\omega(e_{i,j}) = \frac{91}{30}$): there are at least 4 parallel edges of M;
6. $\left\{1, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right\}$ ($\omega(e_{i,j}) = 3$): there are at least 2 parallel edges of M;
7. $\left\{1, \frac{1}{2}, \frac{1}{2}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right\}$ ($\omega(e_{i,j}) = 3$): there are two possible cases,
   (a) there is no singleton edge other than $e_{i,j}$ with $\omega(\cdot) \geq 3$, but there are at least 2 parallel edges of M;
   (b) there is one singleton edge other than $e_{i,j}$ with $\omega(\cdot) = 3$, accompanied by at least 3 parallel edges of M;
8. $\left\{1, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 0\right\}$ ($\omega(e_{i,j}) = 3$): there are at least 2 parallel edges of M.

We conclude this section with the following lemma.

Lemma 3.34. *Every edge $e_{i,j} \in M$ with $\omega(e_{i,j}) \geq 3$ must be a singleton, and $\omega(e_{i,j}) \in \left\{\frac{10}{3}, \frac{13}{4}, \frac{19}{6}, \frac{37}{12}, \frac{91}{30}, 3\right\}$. Furthermore,*

1. *the existence of an edge $e_{i,j} \in M$ with $\omega(e_{i,j}) = \frac{10}{3}$ or $\frac{13}{4}$ or $\frac{91}{30}$ is accompanied with at least 4 parallel edges of M each with $\omega(\cdot) \leq 2.5$;*
2. *the existence of an edge $e_{i,j} \in M$ with $\omega(e_{i,j}) = \frac{19}{6}$ or $\frac{37}{12}$ is accompanied with at least 3 parallel edges of M each with $\omega(\cdot) \leq 2.5$;*
3. *the existence of an edge $e_{i,j} \in M$ with $\omega(e_{i,j}) = 3$ is accompanied with at least 1.5 parallel edges of M each with $\omega(\cdot) \leq 2.5$.*

Each of these accompanying parallel edges must belong to either $C(C^*(e_{i,\cdot}))$ with $|C^*(e_{i,\cdot})| = 3$, or $C(C^*(e_{\cdot,j}))$ with $|C^*(e_{\cdot,j})| = 3$, for some $e_{i,j}$.

3.4.2.7 An upper bound on the average value of $\omega(e)$

Let $M_{\geq 3}$ be the subset of all the edges of M with $\omega(\cdot) \geq 3$, and let $n_s = |M_{\geq 3}|$. Let P denote the subset of all the accompanying parallel edges of M determined in Lemma 3.34, and let $n_p = |P|$. From Lemma 3.23, every edge of $M_{\geq 3}$ is a singleton, and thus $M_{\geq 3} \cap P = \emptyset$.

Lemma 3.35. *Each edge of P belongs to $C(C^*(e_{i,j}))$ for at most four distinct edges $e_{i,j} \in M_{\geq 3}$.*

Proof. Consider an edge $e_{h,\ell} \in P$ and assume that the edge $e_{h+1,\ell+1}$ is also in M.

Consider the vertex d^B_ℓ on which $e_{h,\ell}$ is incident; let $e^*_{\ell-2}, e^*_{\ell-1}, e^*_\ell, e^*_{\ell+1}, e^*_{\ell+2}$ be the edge of M^* incident on the vertex $d^B_{\ell-2}$, $d^B_{\ell-1}$, d^B_ℓ, $d^B_{\ell+1}$, $d^B_{\ell+2}$, respectively, if such an edge exists. Clearly, for any edge $e_{i,j} \in M_{\geq 3}$, if $C^*(e_{i,j})$ does not contain any of the five edges $e^*_{\ell-2}, e^*_{\ell-1}, e^*_{\ell}, e^*_{\ell+1}, e^*_{\ell+2}$, then $e_{h,\ell} \notin C(C^*(e_{i,j}))$ (unless $e_{h,\ell} \in C(C^*(e_{i,j}))$ through the symmetric discussion using the vertex d^A_h). We distinguish two cases where $C^*(e_{i,\cdot})$ contains one of the five edges and $C^*(e_{\cdot,j})$ contains one of the five edges, respectively.

When $C^*(e_{i,\cdot})$ contains one of the five edges $e^*_{\ell-2}, e^*_{\ell-1}, e^*_{\ell}, e^*_{\ell+1}, e^*_{\ell+2}$, we see from all the 27 configurations of $C(C^*(e_{i,\cdot}))$ and Lemma 3.28 that neither of the edges $e^*_{\ell}, e^*_{\ell+1}$, if exists, can be incident on the vertex d^A_i. It follows that the vertex d^A_i is an end of one of the three edges $e^*_{\ell-2}, e^*_{\ell-1}, e^*_{\ell+2}$.

When $C^*(e_{\cdot,j})$ contains one of the five edges $e^*_{\ell-2}, e^*_{\ell-1}, e^*_{\ell}, e^*_{\ell+1}, e^*_{\ell+2}$, we know that $j = \ell - 2$ due to the fact that the edge $e_{i,j}$ is a singleton edge of M.

Since no two edges of $M_{\geq 3}$ are adjacent to a common edge of M^*, we conclude that there are at most three distinct edges $e_{i,j} \in M_{\geq 3}$ such that $e_{h,\ell} \in C(C^*(e_{i,j}))$ through the vertex d^B_ℓ. Furthermore, if there are such three distinct edges, then one is incident on $v^B_{\ell-2}$, one is adjacent to $e^*_{\ell-1}$ (but not incident on $v^B_{\ell-1}$), and the other is adjacent to $e^*_{\ell+2}$ (but not incident on $v^B_{\ell+2}$); the five edges $e^*_{\ell-2}, e^*_{\ell-1}, e^*_{\ell}, e^*_{\ell+1}, e^*_{\ell+2}$ all exist, so do the extra two edges $e^*_{\ell-3}$ and $e^*_{\ell+3}$, and these seven edges of M^* are consecutively parallel.

The three edges $e^*_{\ell-1}, e^*_{\ell}, e^*_{\ell+1}$ of M^* are conflicting with only the three edges of $M_{\geq 3}$ and the two parallel edges $e_{h,\ell}, e_{h+1,\ell+1}$ of M; and for each of these three edges of $M_{\geq 3}$, there is another distinct edge of M^* conflicting with only this edge of $M_{\geq 3}$. In other words, there are six edges of M^* conflicting with only the three edges of $M_{\geq 3}$ and the two parallel edges $e_{h,\ell}, e_{h+1,\ell+1}$ of M, a contradiction as the algorithm $\mathcal{LS}$ would swap them to expand M.

This proves that there are at most two distinct edges $e_{i,j} \in M_{\geq 3}$ such that $e_{h,\ell} \in C(C^*(e_{i,j}))$ through the vertex d^B_ℓ. Symmetrically, we can prove that there are at most two distinct edges $e_{i,j} \in M_{\geq 3}$ such that $e_{h,\ell} \in C(C^*(e_{i,j}))$ through the vertex d^A_h. Therefore, there are at most four distinct edges $e_{i,j} \in M_{\geq 3}$ such that $e_{h,\ell} \in C(C^*(e_{i,j}))$. □

Using Lemma 3.34, assume there is a fraction of xn_s edges of $M_{\geq 3}$ each accompanied with 4 parallel edges of P; there is a fraction of yn_s edges of $M_{\geq 3}$ each accompanied with 3 parallel edges of P; and there is a fraction of $(1 - x - y)n_s$ edges of $M_{\geq 3}$ each accompanied with 1.5 parallel edges of P, where $x \geq 0, y \geq 0, 1 - x - y \geq 0$. From Lemma 3.35, we have

$$4n_p \geq 4xn_s + 3yn_s + 1.5(1 - x - y)n_s = (1.5 + 2.5x + 1.5y)n_s,$$

which gives

$$\frac{n_p}{n_s} \geq \frac{1.5 + 2.5x + 1.5y}{4}, \tag{3.7}$$

and the average amount of tokens for all the edges of $M_{\geq 3} \cup P$ is, using Equation 3.7,

$$\overline{\omega(e)} \leq \frac{2.5n_p + \frac{10}{3}xn_s + \frac{19}{6}yn_s + 3(1 - x - y)n_s}{n_p + n_s} \leq \frac{5}{2} + \frac{12 + 8x}{33 + 15x} \leq \frac{35}{12}. \tag{3.8}$$

Since every other edge of M has its $\omega(\cdot) \leq \frac{35}{12}$ too (see Lemma 3.28, proved in Section 3.4.2.5). Therefore, the average amount of tokens for all the edges of M is no greater than $\frac{35}{12}$. We have thus proved the following theorem.

Theorem 3.36. *The algorithm $\mathcal{LS}$ is an $O(n^{13})$-time $\frac{35}{12}$-approximation for both the* MCBM *and the* Max-Duo *problems.*

3.4.3 Lower bounds on the locality gap for the algorithm $\mathcal{LS}$

In this section, we present two instances of the MCBM and Max-Duo problems, respectively, to show that the approximation ratio of the algorithm $\mathcal{LS}$ has a lower bound of $\frac{13}{6} > 2.166$ for MCBM and a lower bound of $\frac{5}{3} > 1.666$ for Max-Duo.

3.4.3.1 An instance of MCBM

For the MCBM problem, consider the bipartite graph $G = (V^A, V^B, E)$ shown in Figure 3.31, where $V^A = \{1, 2, \ldots, 26\}$, $V^B = \{1', 2', \ldots, 26'\}$, and E is the set of all the edges in solid and dashed lines. One can see that the set of 26 consecutive parallel edges (in dashed lines) is an optimal solution M^* to the MCBM problem on G. Let M be the maximal compatible matching shown as solid lines in Figure 3.31, and assume it is the starting matching for the algorithm $\mathcal{LS}$ on G.

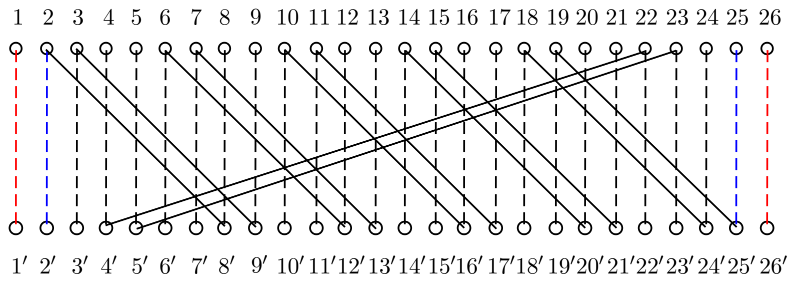


FIGURE 3.31: A bipartite graph $G = (V^A, V^B, E)$, where $V^A = \{1, 2, \ldots, 26\}$, $V^B = \{1', 2', \ldots, 26'\}$, and $E = M \cup M^*$. M consists of the 12 edges in solid lines and it is a maximal compatible matching in G; M^* consists of the 26 edges in dashed lines and it is an optimal compatible matching to the MCBM problem on G.

Lemma 3.37. *M cannot be further improved by the algorithm $\mathcal{LS}$ due to the following reasons:*

1. *M is a maximal compatible matching in G;*
2. *all the edges of M are parallel edges;*
3. *for any $e \in M$, there is at most one edge of M^* compatible with all the edges in $M - \{e\}$;*
4. *for any 2 edges $e_1, e_2 \in M$, there are at most two edges of M^* compatible with all the edges in $M - \{e_1, e_2\}$;*
5. *for any 3 edges $e_1, e_2, e_3 \in M$, there are at most three edges in M^* compatible with all the edges in $M - \{e_1, e_2, e_3\}$;*
6. *for any 4 edges $e_1, \ldots, e_4 \in M$, there are at most four edges in M^* compatible with all the edges in $M - \{e_1, \ldots, e_4\}$;*
7. *for any 5 edges $e_1, \ldots, e_5 \in M$, there are at most five edges in M^* compatible with all the edges in $M - \{e_1, \ldots, e_5\}$.*

Proof. The first two items are trivial. We note that the second item implies that M cannot be further improved by the algorithm using the operation REDUCE-5-BY-5. We next show that M cannot be further improved by the algorithm using the operation REPLACE-5-BY-6.

We examine whether some edges of M can be swapped out for more edges from M^* by REPLACE-5-BY-6. For ease of presentation, we partition M^* into 3 subsets $M^{*1} = \{(1, 1'), (26, 26')\}$, $M^{*2} = \{(2, 2'), (25, 25')\}$, and $M^{*3} = \{(3, 3'), (4, 4'), \ldots, (24, 24')\}$ (in Figure 3.31, their edges are colored red, blue, black, respectively). We have the following observations.

Observation 3.5. *To swap for an edge of M^{*1}, a unique edge of M has to be swap out. In details,* $(2, 8')$ *needs to be swapped out for* $(1, 1')$*, and* $(19, 25')$ *needs to be swapped out for* $(26, 26')$.

Observation 3.6. *To swap for an edge of M^{*2}, a unique pair of parallel edges of M has to be swap out. In details,* $(2, 8')$ *and* $(3, 9')$ *need to be swapped out for* $(2, 2')$*, and* $(19, 25')$ *and* $(18, 24')$ *need to be swapped out for* $(25, 25')$.

Observation 3.7. *To swap for an edge of M^{*3}, a unique triplet of edges of M has to be swap out. In details, when $i = 3$ $(mod\ 4)$, $e_{i-1}, e_i, e_{i+1'}$ need to be swapped out for (i, i'); when $i = 0$ $(mod\ 4)$, $e_{i-1}, e_{i'}, e_{i+1'}$ need to be swapped out for (i, i'); when $i = 1$ $(mod\ 4)$, $e_{i-1'}, e_{i'}, e_{i+1}$ need to be swapped out for (i, i'); when $i = 2$ $(mod\ 4)$, $e_{i-1'}, e_i, e_{i+1}$ need to be swapped out for (i, i').*

The Observations 3.5, 3.6, 3.7 prove trivially the items 3–5 of the lemma.

To prove the item 6, we next see what a subset of four edges of M can do. If these four edges are able to swap in one edge of M^{*3}, then by Observation 3.7 this edge of M^{*3} actually requires three out of the four edges. Note that these particular three edges are not able to swap for any other edge of M^{*3}. If they contain a unique pair of parallel edges of M in Observation 3.6, then they can swap in three edges one from each of M^{*1}, M^{*2}, M^{*3}, and the fourth edge either forms with two of them to form another triplet to swap in an other edge of M^{*3}, or it is able to swap in the other edge of M^{*1}. If these particular three edges do not contain a unique pair of parallel edges of M in Observation 3.6, then they can swap in only the edge of M^{*3}, and the fourth edge can either form with one of them to form a unique pair of parallel edges of M in Observation 3.6 to swap in two edges one from each of M^{*1}, M^{*2}, and/or form with two of them to form another triplet to swap in an other edge of M^{*3}, or it is able to swap in the other edge of M^{*1}. Therefore, these four edges can swap in the best case four edges $(1, 1'), (2, 2'), (3, 3')$ (or $(26, 26'), (25, 25'), (24, 24')$, respectively) and another edge of $M^{*1} \cup M^{*3}$. If these four edges are not able to swap in any edge of M^{*3}, then they can swap in the best case four edges of $M^{*1} \cup M^{*2}$.

To prove the item 7, we next see what a subset of five edges of M can do. If these five edges are able to swap in two edges of M^{*3}, then by Observation 3.7 these two edges of M^{*3} actually require at least four out of the five edges. Note that these particular four edges are only able to swap for four edges of M^*, including the above two edges of M^{*3} and the other two edges must be either $(1, 1'), (2, 2')$ or $(26, 26'), (25, 25')$. Then the fifth edge either forms with two of these particular four edges to form another triplet to swap in an other edge of

M^{*3}, or it is able to swap in the other edge of M^{*1}. If these five edges are not able to swap in at least two edges of M^{*3}, then they can swap in the best case one edge of M^{*3} and four edges of $M^{*1} \cup M^{*2}$. □

Theorem 3.38. *There is a lower bound of* $\frac{13}{6} > 2.166$ *on the locality gap of the algorithm* $\mathcal{LS}$ *for the MCBM problem.*

Proof. By Lemma 3.37, if the matching M is fed as the starting matching to the algorithm $\mathcal{LS}$, then the algorithm terminates without modifying it. Note that we have $|M| = 12$ and $|M^*| = 26$, we conclude that the algorithm can not do better than $\frac{13}{6}$ in the worst case, which proves Table 3.38. □

Remark 3.39. Using our amortized analysis, let $C(e^*)$ be the subset of edges of M conflicting with the edge $e^* \in M^*$. Then in the above instance, we have $|C(e^*)| = 1, 2, 3$ for $e^* \in M^{*1}, M^{*2}, M^{*3}$, respectively. The maximum total amount of tokens received by the edges of M is achieved at $(2, 8')$, where $\omega((2, 8')) = 1 + \frac{1}{2} + 4 \times \frac{1}{3} = \frac{17}{6} \approx 2.833$. This maximum is quite close to the approximation ratio $\frac{35}{12} \approx 2.917$ of the algorithm $\mathcal{LS}$, which is also the maximum possible $\omega(\cdot)$ value for the parallel edges of M. (Recall that $\frac{35}{12} = 1 + 2 \times \frac{1}{2} + 2 \times \frac{1}{3} + \frac{1}{4}$.)

Remark 3.40. We may have variants of the algorithm $\mathcal{LS}$ by substituting the operation Replace-5-By-6 with the similarly defined operation Replace-ρ-By-$(\rho + 1)$, for any $\rho \geq 1$ (with or without the operation Reduce-5-By-6).

For $\rho = 1, 2, 3, 4$, we can construct similar instances to show the corresponding lower bounds on the locality gap for the variants on the MCBM problem.

- $\rho = 4$.

  We can construct a similar instance $G = (V^A, V^B, E)$, except that $|V^A| = |V^B| = 22$ and $|M| = 10$. The performance ratio of the algorithm is $\frac{11}{5} = 2.2$.

- $\rho = 3$.

  We can construct two different instances for $G = (V^A, V^B, E)$. One is similar to the previous two, except that $|V^A| = |V^B| = 18$ and $|M| = 8$. In the other we have $|V^A| = |V^B| = 9$ and $|M| = 4$, such that the four edges of M are consecutively parallel. The performance ratio of the algorithm on both instances is $\frac{9}{4} = 2.25$.

- $\rho = 2$.

  We can construct an instance similar to the second instance when $\rho = 3$, which is a graph $G = (V^A, V^B, E)$ with $|V^A| = |V^B| = 8$ and $|M| = 3$, such that the three edges of M are consecutively parallel. The performance ratio of the algorithm is $\frac{8}{3} \approx 2.667$.

- $\rho = 1$.

  We can construct a similar graph $G = (V^A, V^B, E)$ with $|V^A| = |V^B| = 7$ and $|M| = 2$, such that the two edges of M are parallel. The performance ratio of the algorithm is $\frac{7}{2} = 3.5$. This is essentially the 3.5-approximation by Boria *et al.* [9], and the instance shows that the performance ratio is tight.

3.4.3.2 An instance of Max-Duo

For the Max-Duo problem, consider an instance with two identical length-11 strings $A = B = (a, b, c, d, e, f, b, c, d, e, g)$. We construct the corresponding bipartite graph $G = (V^A, V^B, E)$ (shown in Figure 3.32), where $V^A = \{1, 2, \ldots, 10\}$ and $V^B = \{1', 2', \ldots, 10'\}$. Since $A = B$, each pair of duos represented by the vertices i and i' are the same, for $i = 1, 2, \ldots, 10$. Thus it is easy to see that there is an optimal solution M^* to MCBM on G, which consists of all the 10 edges in dashed lines shown in Figure 3.32. Let M be the compatible matching consisting of the six edges in solid lines in Figure 3.32.

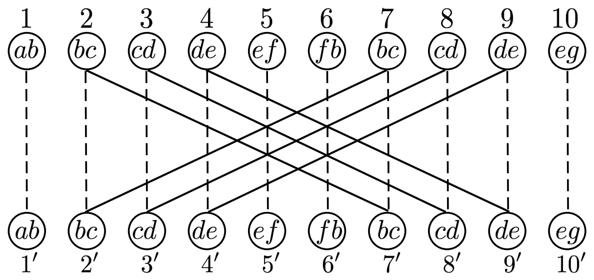


Figure 3.32: The corresponding bipartite graph $G = (V^A, V^B, E)$ constructed from two identical strings $A = B = (a, b, c, d, e, f, b, c, d, e, g)$, where $V^A = \{1, 2, \ldots, 10\}$, $V^B = \{1', 2', \ldots, 10'\}$, and $E = M \cup M^*$. M consists of the six edges in solid lines and it is a compatible matching in G; M^* consists of the ten edges in dashed lines and it is an optimal compatible matching to the MCBM problem on G.

Lemma 3.41. *M cannot be further improved by the algorithm $\mathcal{LS}$ due to the following reasons:*

1. *M is a local maximal compatible matching;*
2. *all edges of M are parallel edges;*
3. *for any $e \in M$, there is no edge in M^* compatible with all the 5 edges in $M - \{e\}$;*
4. *for any 2 edges $e_1, e_2 \in M$, there are at most 2 edges in M^* compatible with all the 4 edges in $M - \{e_1, e_2\}$;*

5. *for any* 3 *edges* $e_1, e_2, e_3 \in M$*, there are at most* 2 *edges in* M^* *compatible with all the* 3 *edges in* $M - \{e_1, e_2, e_3\}$*;*
6. *for any* 4 *edges* $e_1, \ldots, e_4 \in M$*, there are at most* 4 *edges in* M^* *compatible with both of the* 2 *edges in* $M - \{e_1, \ldots, e_4\}$*;*
7. *for any* 5 *edges* $e_1, \ldots, e_5 \in M$*, there are* 4 *edges in* M^* *compatible with the only edge in* $M - \{e_1, \ldots, e_5\}$.

Proof. The first three items are trivial. We note that the second item implies that M cannot be further improved by the algorithm using the operation Reduce-5-By-5. We partition M into three subsets $M^2 = \{(2, 7'), (7, 2')\}$, $M^3 = \{(3, 8'), (8, 3')\}$, and $M^4 = \{(4, 9'), (9, 4')\}$.

For any $e \in M$, we see that $|C^*(e)| = 6$, implying there are only 4 edges of M^* compatible with e; this proves the 7th observation.

For the two edges e_1, e_2 in the same part of the partition, we see that $C^*(e_1) = C^*(e_2)$, implying that if one of them is in M, then none of the six edges of $C^*(e_1)$ can be compatible with it. (This proves again the item 3.) Also, we see that the edges $(1, 1'), (6, 6')$ are not compatible with the edges and only these edges in M^2; and the edges $(10, 10'), (5, 5')$ are not compatible with the edges and only these edges in M^4.

To prove the item 4, we see that when the two edges $e_1, e_2 \in M$ are not in the same part, then from the last paragraph there is no edge in M^* compatible with all the 4 edges in $M - \{e_1, e_2\}$; when the two edges $e_1, e_2 \in M$ are in the same part, then either there are two edges in M^* compatible with all the 4 edges in $M - \{e_1, e_2\}$, if this part is M^2 or M^4, or otherwise there is no edge in M^* compatible with all the 4 edges in $M - \{e_1, e_2\}$.

For the item 5, any three edges $e_1, e_2, e_3 \in M$ cannot take up two separate parts, and therefore there are at most 2 edges in M^* compatible with all the 3 edges in $M - \{e_1, e_2, e_3\}$.

For any 4 edges $e_1, \ldots, e_4 \in M$, if they take up two parts, then there are exactly 4 edges of M^* compatible with the 2 edges in $M - \{e_1, \ldots, e_4\}$, which belong to the same part; if they do not take up two parts, then there are at most 2 edges of M^* compatible with the 2 edges in $M - \{e_1, \ldots, e_4\}$. This proves the item 6, and completes the proof of the lemma. □

Theorem 3.42. *There is a lower bound of* $\frac{5}{3} > 1.666$ *on the locality gap of the algorithm* $\mathcal{LS}$ *for the* Max-Duo *problem.*

Proof. By Lemma 3.41, if the matching M is fed as the starting matching to the algorithm $\mathcal{LS}$, then the algorithm terminates without modifying it. Note that we have $|M| = 6$ and

$|M^*| = 10$, we conclude that the algorithm can not do better than $\frac{5}{3}$ in the worst case, which proves Chapter 3.42. □

3.5 Concluding remarks and possible future work

In this chapter, we examined the MAX-DUO problem, the complement of the well studied MCSP problem.

In Section 3.3, based on an existing linear reduction to the MIS problem [10, 46], we presented a vertex-degree reduction technique for the 2-MAX-DUO to reduce the maximum degree of the constructed instance graph to 4. Along the way, we uncovered many interesting structural properties of the constructed instance graph. This degree reduction enabled us to adopt the state-of-the-art approximation algorithm for the MIS problem on low degree graphs [7] to achieve a $(1.4 + \epsilon)$-approximation for 2-MAX-DUO, for any $\epsilon > 0$.

It is worth mentioning that our vertex-degree reduction technique can be applied for k-MAX-DUO with $k \geq 3$. In fact, we had worked out the details for $k = 3$, to reduce the maximum degree of the constructed instance graph from 12 to 10, leading to a $(2.6 + \epsilon)$-approximation for 3-MAX-DUO, for any $\epsilon > 0$. Nevertheless, the $(2.6 + \epsilon)$-approximation is superseded by the $(2 + \epsilon)$-approximation for the general MAX-DUO [33].

In Section 3.4, motivated by an earlier local search algorithm, we presented an improved algorithm $\mathcal{LS}$ for a more general MCBM problem, that uses one operation to increase the cardinality of the solution and another novel operation to reduce the singleton edges in the solution. The algorithm is iterative and has a time complexity $O(n^7)$, where n is the length of the input strings. Through an amortized analysis, we were able to show that the proposed algorithm $\mathcal{LS}$ has an approximation ratio of at most $35/12 < 2.917$. Our result improves the previous best 3.25-approximation for both problems, thus breaking the barrier of 3; but there is a better $(2+\epsilon)$-approximation [33] which appears about the same time as ours. The $(2+\epsilon)$-approximation is based on the same two design ideas while applies them in a different order to first greedily select as many large-size consecutive parallel edges as possible, followed by swapping procedure for increasing the matching size; its performance analysis is also done by an amortization scheme, though different. We believe both approximation algorithms and both performance analyses are interesting, and they together will provide better insights into the MAX-DUO problem, eventually leading to further improved approximation algorithms.

We also showed that there is a lower bound of $13/6 > 2.166$ and $5/3 > 1.666$ on the locality gap of the algorithm $\mathcal{LS}$ for the MCBM and the Max-Duo problems, respectively. Our construction method could be used to design instances to show a lower bound of 2 for the $(2+\epsilon)$-approximation too.

For possible future work, it would be worthwhile to further investigate the 2-Max-Duo problem to see whether the maximum degree can be further reduced to 3, by examining the structural properties associated with the degree-4 vertices to see if all the degree-4 vertices can be converted to vertices with maximum degree of 3. On the other hand, it is also interesting to examine whether a better-than-1.4 approximation algorithm can be designed directly for the MIS problem on those degree-4 graphs obtained at the end of the vertex contracting process. For the general Max-Duo problem, one direction would be to find a technique other than the local search approach which may result in an approximation guarantee of 2 or less.

```
Replace-$\rho$-by-$(\rho + 1)(G, M)$
 1: for each subset $X$ of $M$ with $|X| = \rho$ do                ▷ $O(n^\rho)$ combinations of such $X$
 2:     find $V_0(X)$, $E' = X \cup C'(X)$, and $V(X)$                 ▷ $E'$ can be found in $O(n)$ time
 3:     for each subset $U$ of $V(X)$ with $|U| = \rho + 1$ do          ▷ $O(1)$ combinations of such $U$
 4:         $E' \leftarrow E' -$ {edges with one endpoint in $U^A$ and the other in $U^B$}
 5:         find all the maximal subsets of consecutive vertices in $U^A$ and $U^B$          ▷ $O(n)$
 6:         if there is at least one subset of consecutive parallel edges incident on $U^A_{i,h}$ and $U^B_{j,\ell}$, for
    $\forall U^A_{i,h} \subseteq U^A$ and $\forall U^B_{j,\ell} \subseteq U^B$, respectively then
 7:             for every pair of $U^A_{i,h}$ and $U^B_{j,\ell}$ do                     ▷ $O(1)$ pairs in total
 8:                 find all possible subsets of consecutive parallel edges in $E'$ incident on $U^A_{i,h} \cup U^B_{j,\ell}$
    ▷ at most two possible subsets
 9:                 $X'_{i,h;j,\ell} \leftarrow E'_{i,j',h} \cup E'_{i',j,\ell}$ if $i' = i + h$ and $j' = j - h$
10:                 $X'_{j,\ell;i,h} \leftarrow E'_{i',j,\ell} \cup E'_{i,j',h}$ if $i' = i - \ell$ and $j' = j + \ell$
11:             end for
12:             $Y \leftarrow \{X'_{i,h;j,\ell}, X'_{j,\ell;i,h} | U^A_{i,h} \subseteq U^A, U^B_{j,\ell} \subseteq U^B\}$                ▷ $|Y|$ is in $O(1)$
13:             for each subset $Y' \subseteq Y$ do                  ▷ $2^{|Y|}$ subsets, which is in $O(1)$
14:                 if $Y' = \emptyset$ then
15:                     $X' \leftarrow$ FindCompatibleEdgesAtU$(E', U)$                         ▷ $O(n)$
16:                     if $X' \neq \emptyset$ then
17:                         $M \leftarrow (M - X) \cup X'$
18:                         break
19:                     end if
20:                 else
21:                     $X'_2 \leftarrow$ the union of all $X'_{\cdot,\cdot;\cdot,\cdot}$ in $Y'$                    ▷ $|X'_2|$ is in $O(1)$
22:                     if the edges in $X'_2$ are pairwise compatible then           ▷ checked in $O(1)$
23:                         $E'_1 \leftarrow E' - \left(X'_2 \cup C(X'_2)\right)$
24:                         $V'_1 \leftarrow$ the union of all the endpoints of edges in $E'_1$
25:                         $U_1 \leftarrow U \cap V'_1$
26:                         $X'_1 \leftarrow$ FindCompatibleEdgesAtU$(E'_1, U_1)$                       ▷ $O(n)$
27:                         if $X'_1 \neq \emptyset$ then
28:                             $X' \leftarrow X'_1 \cup X'_2$
29:                             $M \leftarrow (M - X) \cup X'$
30:                             break
31:                         end if
32:                     else
33:                         $X' \leftarrow$ FindCompatibleEdgesAtU$(E', U)$                        ▷ $O(n)$
34:                         if $X' \neq \emptyset$ then
35:                             $M \leftarrow (M - X) \cup X'$
36:                             break
37:                         end if
38:                     end if
39:                 end if
40:             end for
41:         end if
42:     end for
43: end for
44: return $M$
```

Figure 3.14: A high-level description of the algorithm of expanding the current maximal compatible matching by swapping ρ edges out for $\rho + 1$ compatible edges.

Chapter 4

The Path Partition Problem[1]

4.1 Introduction

Motivated by the data integrity of communication in wireless sensor networks and several other applications, the *k-path partition* (k-PP) problem was first considered by Yan et al. [76]. One can consider a broadcasting problem in data communication networks. Given some information, by modeling a data communication network with a graph, broadcasting is to transmit the information from some vertices to all the other vertices in the network only through paths, that is, one vertex can only transmit the information to its adjacent vertices through the edge connecting them. It requires one unit of time to transmit the information from one vertex to another through an edge. The goal is to select the minimum number of vertices such that the information can be transmitted from those selected vertices to all the other vertices within a fixed number of time units. This is an application of the k-PP problem, which can be formally defined as follows. Given a graph $G = (V, E)$, the *order* of a simple path in G is the number of vertices on the path and it is called a *k-path* if its order is k. (The *length* of a k-path is $k - 1$, the number of edges thereon.) The k-PP problem is to find a minimum collection of vertex-disjoint paths of order at most k such that every vertex is on some path in the collection.

Clearly, the 2-PP problem is exactly the Maximum Matching problem, which is solvable in $O(m\sqrt{n}\log(n^2/m)/\log n)$-time [45], where $n = |V|$ and $m = |E|$. For $k \geq 3$, k-PP is **NP**-hard [43]. (See Figure 4.1 for an instance with a solution for the 3-PP problem.)

We point out the key phrase "at most k" in the definition, that ensures the existence of a feasible solution for any given graph; on the other hand, if one asks for a path partition in which every path has an order exactly k, the problem is called *P_k-partitioning* and is

[1]This chapter is based on two papers [23, 24]. [24] is a work with Chen, Goebel, Lin, Su, and Zhang, "An improved approximation algorithm for the minimum 3-path partition problem" which is published by the *Journal of Combinatorial Optimization* (JOCO); [23] is a work with Chen, Goebel, Lin, Liu, Su, Tong, and Zhang "A local search 4/3-approximation algorithm for the minimum 3-path partition problem" which is a submission under review.

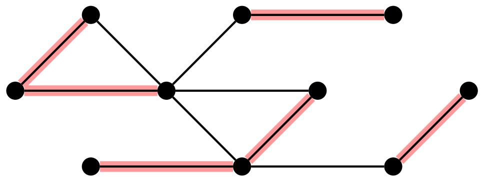


FIGURE 4.1: An instance of 3-PP. The two 2-paths and the two 3-paths represented by edges with red backgroud is a solution to 3-PP for this instance.

also NP-complete for any fixed constant $k \geq 3$ [43], even on bipartite graphs of maximum degree three [64]. To the best of our knowledge, there is no approximation algorithm with proven performance for the general k-PP problem, except the trivial k-approximation using all 1-paths. For 3-PP, Monnot and Toulouse [64] proposed a $3/2$-approximation, based on two maximum matchings.

The k-PP problem is a generalization to the PATH COVER problem [41] (also called PATH PARTITION), which is to find a minimum collection of vertex-disjoint paths which together cover all the vertices in G. PATH COVER contains the HAMILTONIAN PATH problem [43] as a special case, and thus it is **NP**-hard and it is outside **APX**.

The k-PP problem is also closely related to the well-known *set cover* SC problem. Given a collection of subsets $C = \{S_1, S_2, \ldots, S_m\}$ of a finite ground set $U = \{x_1, x_2, \ldots, x_n\}$, an element $x_i \in S_j$ is said to be *covered* by the subset S_j, and a *set cover* is a collection of subsets which together cover all the elements of the ground set U. The SC problem asks to find a minimum set cover. SC is one of the first proven NP-hard problems [43], and is also one of the most studied optimization problems for the approximability [50] and inapproximability [38, 67, 74]. The variant of SC in which every given subset has size at most k is called k-SC, which is APX-complete and admits a $4/3$-approximation for $k = 3$ [35] and an $(H_k - \frac{196}{390})$-approximation for $k \geq 4$ [61].

To see the connection between k-PP and k-SC, we may take the vertex set V of the given graph as the universe, and an ℓ-path with $\ell \leq k$ as a subset, then the k-PP problem is the same as asking for a *special* minimum set cover in which the subsets are *mutually disjoint*. However, existing approximations for k-SC do not readily apply to k-PP. This is because in a feasible set cover, an element of the ground set U could be covered by multiple subsets, while in k-PP, every vertex is on exactly one path in a feasible solution. If one wants to enforce the mutual disjointness requirement in the SC problem, then they can expand C to include all the proper subsets of each given subset $S_j \in C$. But in an instance graph of k-PP, not every subset of vertices on a path is traceable, and so such an expanding technique does not apply. In summary, k-PP and k-SC share some similarities, but none contains the other as a special case.

For the 3-PP problem, the previous best result is a $3/2$-approximation; for the 3-SC problem, the currently best result is the 20+ years old $4/3$-approximation. In this chapter, we propose a $4/3$-approximation for 3-PP, which coincidently meets the current best approximation result for 3-SC.

We briefly review the previous $3/2$-approximation for 3-PP by Monnot and Toulouse [64]. It first computes a maximum matching M^* in the given G, then computes another maximum matching between the edges of M^* and the vertices exposed by M^*, which essentially maximizes the number of 3-paths each is formed by attaching an isolated vertex to an edge of M^*. It returns the achieved 3-path partition deduced from these two matchings and the remaining isolated vertices. The running time of this algorithm is in $O(nm)$, where $n = |V|$ and $m = |E|$.

Our $4/3$-approximation algorithm for the 3-PP problem first computes a 3-path partition with the least 1-paths in $O(nm)$ time, and then it applies a local search scheme to repeatedly search for an *expected collection* of 2- and 3-paths and replace it by a strictly smaller *replacement collection* of new 2- and 3-paths. It is worth pointing out that the number of 1-paths in the 3-path partition computed in the first step is actually the minimum among all k-path partitions, for any $k \geq 3$. Thus this first step itself is already a $k/2$-approximation for the k-PP problem. We prove its performance through a more complicated amortized analysis.

The rest of this chapter is organized as follows. In Section 4.2, we present the local search scheme, including the algorithm to compute a 3-path partition with the least 1-paths, and the expected collections of 2- and 3-paths, along with the replacement collection of new 2- and 3-paths. The performance analysis is presented in Section 4.3, where we also show that the ratio 4/3 is tight for our algorithm. We conclude the chapter in Section 4.4, along with some possible future work.

4.2 A local search approximation algorithm

Our $4/3$-approximation algorithm for the 3-PP problem first computes a 3-path partition with the least 1-paths, then with this 3-path partition, it repeatedly finds a certain collection of 2- and 3-paths (called an *expected collection*) and replaces it by another collection of one less new 2- and 3-paths (called a *replacement collection*). We present in Section 4.2.1 the $O(nm)$-time algorithm that computes a 3-path partition with the least 1-paths. The expected collections of 2- and 3-paths and the corresponding replacement collections are presented in Section 4.3.2. The complete algorithm, denoted as Approx, is summarized in Section 4.3.3.

4.2.1 Computing a 3-path partition with the least 1-paths

In a 3-path partition, a 1-path contains only one vertex and in the sequel it is often referred to as a singleton of the 3-path partition.

In this section, we present an algorithm called Algorithm A for computing a 3-path partition with the least 1-paths, and show that Algorithm A runs in $O(nm)$ time in the given graph $G = (V, E)$, where $n = |V|$ and $m = |E|$. We explain in the following the three steps in Algorithm A, the first two of which constitute exactly the $3/2$-approximation by Monnot and Toulouse [64]. A high-level description of Algorithm A is depicted in Figure 4.3.

4.2.1.1 Step 1: computing a maximum matching

Recall that the running time of computing a maximum matching of the graph $G = (V, E)$ is in $O(m\sqrt{n}\log(n^2/m)/\log n)$-time [45]. In the first step, we apply an $O(m\sqrt{n}\log(n^2/m)/\log n)$-time algorithm to find a maximum matching in G, denoted as M^*; let V_0 denote the subset of vertices exposed by M^*. If $V_0 = \emptyset$, then we have achieved a 3-path partition without (and thus the least) 1-paths, in which a 2-path one-to-one corresponds to an edge of M^*. In the sequel we assume V_0 is non-empty. The following two lemmas are trivial due to the edge maximality of M^*.

Lemma 4.1. *In the graph $G = (V, E)$, all the vertices of V_0 are pairwise non-adjacent to each other; for any edge $(u, v) \in M^*$, if u is adjacent to a vertex $x \in V_0$ and v is adjacent to a vertex $y \in V_0$, then $x = y$.*

Lemma 4.2. *In any 3-path partition for the graph $G = (V, E)$, the total number of 2-paths and 3-paths is at most $|M^*|$.*

Proof. Clearly, if there were more than $|M^*|$ vertex disjoint 2-paths and 3-paths in the graph G, then selecting one edge per such path gives rise to a matching of size greater than $|M^*|$, contradicting the maximality of M^*. □

4.2.1.2 Step 2: computing a second maximum matching

In the second step, we construct a bipartite graph $G' = (M^*, V_0, E')$ as follows:

1. each edge $e = (u, v) \in M^*$ is "shrunk" into a vertex denoted as e and the part containing all these vertices is denoted as M^*;
2. each vertex of V_0 remains as a vertex and the part containing these vertices is still denoted as V_0;
3. the vertices of M^* (V_0, respectively) are non-adjacent to each other;
4. a vertex $e = (u, v) \in M^*$ and a vertex $v_0 \in V_0$ are adjacent in G' if and only if either $(u, v_0) \in E$ or $(v, v_0) \in E$ or both, and the set of edges in G' is denoted as E'.

The graph G' can be constructed in $O(m)$-time, where $m = |E|$ is the number of edges in the graph $G = (V, E)$. We then apply an $O(m\sqrt{n}\log(n^2/m)/\log n)$-time algorithm to find a maximum matching in G', denoted as M_1. For each edge $((u, v), v_0) \in M_1$, we select the edge (u, v_0) if $(u, v_0) \in E$ or otherwise the edge (v, v_0) into the edge set M_2, which is a matching in the graph $G = (V, E)$. The following lemma is trivial due to the construction of M_1 and M_2.

Lemma 4.3. *In the graph $G = (V, E)$, the subgraph $Q = (V, M^* \cup M_2)$ is a collection of vertex disjoint 1-paths, 2-paths, and 3-paths; moreover, the total number of 2-paths and 3-paths is $|M^*|$.*

Let $E^* = M^* \cup M_2$ and $Q = (V, E^*)$, which is the *starting* 3-path partition. Note that the above two steps constitute the 3/2-approximation by Monnot and Toulouse [64], for which the ratio 3/2 is claimed tight. In other words, our Algorithm A builds on the 3/2-approximation and uses its output 3-path partition Q as the starting point. In the next subsection, we present the third step of Algorithm A to iteratively update both the edge set E^* and the 3-path partition Q, to maintain the total number of 2-paths and 3-paths in Q and to minimize the number of 1-paths in Q. Therefore, the 3-path partition produced by Algorithm A is at least as good as the solution by the 3/2-approximation.

4.2.1.3 Step 3: reducing 1-paths to the minimum

Let Q_1 (Q_2, Q_3, respectively) denote the collection of 1-paths (2-paths, 3-paths, respectively) in Q. The third step is iterative, and in every iteration we try to eliminate one singleton while maintaining the total number of 2-paths and 3-paths to be $|M^*|$. That is, we have an invariant that the total number of 2-paths and 3-paths in the 3-path partition Q is $|M^*|$.

Clearly, if $Q_1 = \emptyset$, then we are done with the third step. We thus assume Q_1 is non-empty. For ease of presentation, a vertex that is an ending vertex of a 2-path or a 3-path in the current 3-path partition Q is called an *endpoint*; a vertex that is the middle vertex of a 3-path in Q is called a *midpoint*.

Consider a singleton (i.e., 1-path) v_0 in Q. Due to Lemma 4.2, we conclude that v_0 cannot be adjacent to any endpoint of a 3-path, or any other singleton in the graph G. Therefore, if v_0 is adjacent to a vertex w_1 in G, then w_1 has to be the midpoint of some 3-path $P_1 \in Q_3$.

In the case where w_1 is the midpoint of some 3-path $P_1 \in Q_3$: u_1-w_1-v_1. We claim that if the vertex u_1 is adjacent to a vertex u_2 in the graph G, then u_2 is neither a singleton or an endpoint of another 3-path.

We prove this claim by contradiction. First, u_2 cannot be a singleton due to its role the same as v_0 (due to Lemma 4.2). Next, assume u_2 is an endpoint of a 3-path $P_2 \neq P_1$ and P_2 is u_2-w_2-v_2; then we may remove the edges (w_1, u_1) and (w_1, u_2) while adding the edge (u_1, u_2) to E^*, resulting in $(|M^*| + 1)$ 2-paths and 3-paths in total and thus contradicting Lemma 4.2. This proves the claim.

It follows from the above claim that either u_2 is an endpoint of a 2-path or u_2 is the midpoint of another 3-path. (That is, u_2 now takes up the role of w_1.)

Case 1. In the case when u_2 is an endpoint of some 2-path u_2-v_2, denoted as $P_2 \in Q_2$. We remove the edge (w_1, u_1) while adding the edges (v_0, w_1) and (u_1, u_2) to E^*, resulting in two new 3-paths v_0-w_1-v_1 and u_1-u_2-v_2 while destroying the two paths P_1 and P_2 (that is, u_1-w_1-v_1 and u_2-v_2). This process eliminates the singleton v_0 and maintains in total $|M^*|$ 2-paths and 3-paths, and we say that the *alternating* path v_0-w_1-u_1-u_2 *saves* the singleton v_0. We subsequently update the edge set E^* and the 3-path partition Q, and end the iteration.

Case 2. In the general setting, in the graph G, v_0 is adjacent to the midpoint w_1 of a 3-path P_1, and for $j = 1, 2, \ldots, i-1$, one endpoint u_j of P_j is adjacent to the midpoint w_{j+1} of another 3-path P_{j+1}, and lastly one endpoint u_i of P_i is adjacent to an endpoint u_{i+1} of a 2-path P_{i+1} (see Figure 4.2 for an illustration). Then we may delete the edges $\{(w_j, u_j) \mid j = 1, 2, \ldots, i\}$ from E^* while adding the edges (v_0, w_1), $\{(u_j, w_{j+1}) \mid j = 1, 2, \ldots, i-1\}$, and (u_i, u_{i+1}) to E^* to obtain $(i+1)$ 3-paths from the collection of one singleton, i 3-paths, and one 2-path. This process eliminates the singleton v_0 and maintains in total $|M^*|$ 2-paths and 3-paths, and we say the *alternating* path v_0-w_1-u_1-w_2-u_2-...-w_i-u_i-u_{i+1} *saves* the singleton v_0. We remark that a length-$(2i + 1)$ alternating path connects a singleton to a 2-path, through a series of i 3-paths, where $i \geq 0$ (see Figure 4.2). We subsequently update the edge set E^* and the 3-path partition Q, and end the iteration.

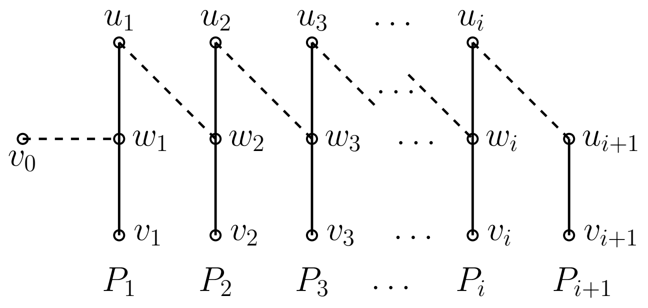


FIGURE 4.2: An alternating path v_0-w_1-u_1-w_2-u_2-...-w_i-u_i-u_{i+1} that saves the singleton v_0, where the first i paths are 3-paths and the last one is a 2-path. In the figure, solid edges are in the edge set E^* and dashed edges are outside of E^*.

Lemma 4.4. *Given an edge set E^* in the graph $G = (V, E)$, the associated 3-path partition Q containing $|M^*|$ 2-paths and 3-paths, and a singleton v_0 therein, finding a simple alternating path to save v_0, if exists, can be done in $O(m)$ time, where $m = |E|$.*

Proof. Firstly, if an alternating path is not simple, then a cycle that forms a subpath is also alternating and has an even length, and thus the cycle can be removed resulting in a shorter alternating path. Repeating this process if necessary, at the end we achieve a simple alternating path. Therefore, we can limit the search for a simple alternating path.

We construct a digraph G'' by creating the following four kinds of directed edges:

1. all those edges incident on v_0, each oriented out of v_0;
2. all those edges of the 3-paths, each oriented from the midpoint and to the endpoint;
3. all those edges each connecting an endpoint of a 3-path to the midpoint of another 3-path, oriented from the endpoint and to the midpoint;
4. all those edges each connecting an endpoint of a 3-path to an endpoint of a 2-path, oriented out of the endpoint of the 3-path.

Then, the edges on all possible alternating paths that save v_0 must be on G'' formed by the above four kinds of directed edges. If follows that by a BFS (*breadth-first search*) traversal starting from v_0 on G'', if an endpoint of a 2-path can be reached then we achieve a simple alternating path; otherwise, we conclude that no alternating path saving the singleton v_0 exists. Both construction of the digraph and the BFS traversal take $O(m)$ time. This proves the lemma. □

Using Lemma 4.4, the third step of the algorithm is to iteratively find a simple alternating path to save a singleton; it terminates when no alternating path is found. We still use Q to denote the 3-path partition at termination. A high-level description of our ALGORITHM A is provided in Figure 4.3.

```
ALGORITHM A on $G = (V, E)$:
Step 1.  1.1. compute a maximum matching $M^*$ in $G$;
         1.2. determine the subset $V_0$ of vertices in $G$ which are exposed by $M^*$;
Step 2.  2.1. construct the bipartite graph $G' = (M^*, V_0, E')$;
         2.2. compute a maximum matching $M_1$ in $G'$;
         2.3. determine the edge set $M_2$ associated with $M_1$;
         2.4. initialize $E^* = M^* \cup M_2$ and the associated 3-path cover $Q$;
Step 3.  3.1. repeatedly find an alternating path to save a singleton in $Q$,
              till no alternating path is found;
         3.2. return the resulting 3-path partition $Q$.
```

FIGURE 4.3: A high-level description of ALGORITHM A for computing a 3-path partition in the graph $G = (V, E)$ with the least singletons.

4.2.1.4 The main theorem

We prove in the next theorem that the 3-path partition produced by ALGORITHM A contains the minimum number of singletons.

Theorem 4.5. *ALGORITHM A is an $O(nm)$-time algorithm for computing a 3-path partition in the graph $G = (V, E)$ with the least 1-paths.*

Proof. Recall that at the end of the second step, the achieved starting 3-path partition contains $|M^*|$ 2-paths and 3-paths; in the third step, in each iteration where an alternating path is found to save a singleton of the current 3-path partition, we swap the edges on the alternating path inside the edge set with the edges outside of the edge set to *move* from the current 3-path partition to another 3-path partition which contains still $|M^*|$ 2-paths and 3-paths (that is, an invariant) but one less singleton.

Denote Q_o and Q_o^* as the 3-path partition produced by Algorithm A for the original input graph $G = (V, E)$ and some 3-path partition of G with the least 1-paths, respectively. Let V' be the set of all the vertices on the paths of $Q_o \cap Q_o^*$, and G' be the subgraph of G induced by $V - V'$. Let $Q = Q_o - Q_o^* and Q^* = Q_o^* - Q_o$. We prove the theorem by showing that the numbers of singletons in the corresponding two 3-path partitions Q and Q^* of the induced subgraph G' must be equal.

Let Q_1 (Q_2, Q_3, respectively) denote the collection of 1-paths (2-paths, 3-paths, respectively) in the 3-path partition Q produced by Algorithm A (the associated edge set is E^*), and let Q_1^* denote the collection of 1-paths in Q^*. Our assumption is $|Q_1| > |Q_1^*|$. Since $Q \cap Q^* = \emptyset$, we have

$$Q_1 \cap Q_1^* = \emptyset. \tag{4.1}$$

i.e., a singleton $v_0 \in Q_1$ is not a singleton in Q_1^*, and thus v_0 is on some path of $Q_2^* \cup Q_3^*$.

Suppose the edge (v_0, w_1) is on some path of $Q_2^* \cup Q_3^*$. Let us examine where the vertex w_1 could be in the computed 3-path partition Q. Recall that Q contains in total $|M^*|$ 2-paths and 3-paths. Due to Lemma 4.2, w_1 cannot be a singleton or an endpoint of a 3-path. From the non-existence of an alternating path at the end of the third step of Algorithm A, w_1 cannot be an endpoint a 2-path either. Therefore, w_1 has to be the midpoint of some 3-path u_1-w_1-v_1, denoted as $P_1 \in Q$. (We refer the reader to Figure 4.2 for an illustration, taking that the solid edges are in Q while the dashed edges are in Q^*.)

Now we examine where the endpoints u_1 and v_1 of the path P_1 could be in Q^*. Apparently not both of them are adjacent to w_1 in Q^*, or otherwise the degree of w_1 in Q^* would be at least three. Assume without loss of generality u_1 is not adjacent to w_1 in Q^*. If u_1 is a singleton in Q^*, then we may add the edge (u_1, w_1) to Q^* and remove the edge (v_0, w_1) from Q^* to obtain another 3-path partition $Q^{*\prime}$ in which u_1 is no longer a singleton but v_0 becomes a singleton. That is, $Q^{*\prime}$ is also an optimal solution and shares a singleton v_0 with Q, contradicting Equation 4.1. This proves that u_1 is not a singleton in Q^* and consequently, exactly the same as v_0, it is on some path of $Q_2^* \cup Q_3^*$. (Again, we refer the reader to Figure 4.2 for an illustration.)

Suppose the edge (u_1, w_2) is on some path of $Q_2^* \cup Q_3^*$. We next examine where the vertex w_2 could be in the computed 3-path partition Q. Due to Lemma 4.2, w_2 cannot be a singleton or an endpoint of a new 3-path (other than P_1). From the non-existence of an alternating path at the end of the third step, w_2 cannot be an endpoint a 2-path either. Therefore, w_2 either collides into v_1 or it has to be the midpoint of some new 3-path (other than P_1). If $w_2 = v_1$, then we may remove the edges (u_1, w_1) and (w_1, v_1) from E^* and add the edges (v_0, w_1) and (u_1, v_1) to E^* to obtain another 3-path partition that contains $(|M^*| + 1)$ 2-paths and 3-paths, contradicting Lemma 4.2. It follows that w_2 has to be the midpoint of some new 3-path u_2-w_2-v_2, denoted as P_2. (Again, we refer the reader to Figure 4.2 for an illustration.)

Now we recursively examine where the endpoints u_2 and v_2 of the path P_2 could be in Q^*. Apparently not both of them are adjacent to w_2 in Q^*, or otherwise the degree of w_2 in Q^* would be at least three. Assume without loss of generality u_2 is not adjacent to w_2 in Q^*. If u_2 is a singleton in Q^*, then we may use the alternating path v_0-w_1-u_1-w_2-u_2 to obtain another optimal 3-path partition $Q^{*\prime}$ that violates Equation 4.1. This proves that u_2 is not a singleton in Q^* and consequently, exactly the same as v_0 and u_1, it is on some path of $Q_2^* \cup Q_3^*$.

Suppose the edge (u_2, w_3) is on some path of $Q_2^* \cup Q_3^*$. We may repeat the above argument for w_2 to prove that w_3 has to be the midpoint of some new 3-path u_3-w_3-v_3, denoted as $P_3 \in Q$, resulting in the same configuration as shown in Figure 4.2. From the path P_3, repeating the same argument we will discover a new distinct path $P_4 \in Q$. Repeatedly, we will discover an infinitely many distinct 3-paths in Q, contradicting the fact that the graph G is finite. Such a contradiction proves that the 3-path partition Q produced by ALGORITHM A has the same number of, and thus the least, singletons as Q^*.

For the running time, since in each iteration of the third step we may "*glue*" all singletons as one for finding an alternating path. If no alternating path is found, then the algorithm terminates; otherwise one can easily check which singletons are the root of the alternating path and pick to save one of them, and the iteration ends. It follows that there could be $O(n)$ iterations and each iteration needs $O(m)$ time, and thus the total running time for the third step is $O(nm)$. Since the first two steps take $O(m\sqrt{n} \log(n^2/m)/\log n)$ time, the overall running time of ALGORITHM A is in $O(nm)$. This finishes the proof of the theorem. □

We observe that the number of 1-paths in the 3-path partition produced by ALGORITHM A is actually the minimum among all k-path partitions, for any $k \geq 3$. Thus, we have the following corollary.

Corollary 4.6. *ALGORITHM A is an $O(nm)$-time algorithm for computing a k-path partition in the graph $G = (V, E)$ with the least 1-paths, thus it is a $k/2$-approximation for the k-PP problem, for any $k \geq 3$.*

4.2.2 Local operations and their priorities

With the 3-path partition Q produced by ALGORITHM A, we design four local operations to improve Q. Throughout the local search, the 3-path partitions are maintained to have the least 1-paths. Our four local operations are designed so not to touch the 1-paths and thus the final 3-path partition still contains the least 1-paths. Each operation transfers three 2-paths to two 3-paths with the aid of a few other 2- or 3-paths. These operations are associated with different priorities, that is, one operation applies only when all the other operations of higher priorities (labeled by smaller numbers) fail to apply to the current 3-path partition. This local search is iterative, and every iteration ends after executing a designed local operation which strictly reduces the number of paths in the partition by exactly one. It terminates when none of the designed local operations applies.

Definition 4.7. With respect to the current 3-path partition Q, a local OPERATION i_1-i_2-BY-j_1-j_2, where $j_1 = i_1 - 3$ and $j_2 = i_2 + 2$, replaces a collection of i_1 2-paths and i_2 3-paths of Q (called an *expected collection*) by a collection of j_1 2-paths and j_2 3-paths on the same subset of $2i_1 + 3i_2$ vertices (called a *replacement collection*).

We present in the rest of this section all the replacement operations to perform on the 3-path partition with the least 1-paths.

4.2.2.1 OPERATION 3-0-BY-0-2, highest priority 1

When three 2-paths of Q can be connected into a 6-path in the graph G (see Fig. 4.4 for an illustration), they form into an expected collection. By removing the middle edge on the 6-path, we achieve two 3-paths on the same six vertices and they form the replacement collection. In the example illustrated in Fig. 4.4, with the two dashed edges in E but outside of Q, OPERATION 3-0-BY-0-2 replaces the three 2-paths (solid black edges) by two new 3-paths (edges with red backgroud).

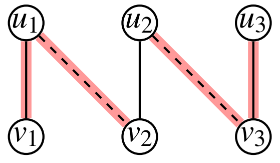


FIGURE 4.4: The configuration of the expected collection for OPERATION 3-0-BY-0-2, which has the highest priority 1, where solid black edges are in Q, dashed edges are in E but outside of Q, and the edges with red backgroud form the replacement collection.

An Operation 3-0-By-0-2 does not need the assistance of any 3-path of Q. In each of the following operations, we need the aid of a couple of other 2- or/and 3-paths to transfer three 2-paths to two 3-paths.

4.2.2.2 Operation 3-1-By-0-3, priority 2

Consider an expected collection of three 2-paths $P_1 = u_1$-v_1, $P_2 = u_2$-v_2, $P_3 = u_3$-v_3, and a 3-path $P_4 = u$-w-v in Q. Note that an Operation 3-1-By-0-3 applies only when Operation 3-0-By-0-2 fails to apply to the current Q, thus P_1, P_2, P_3 cannot be connected into a 6-path. We only determine the following two classes of configurations for the expected collection in an Operation 3-1-By-0-3.

In the first class, which has priority 2.1, u, w, v are adjacent to an endpoint of P_1, P_2, P_3 in G, respectively (see Fig. 4.5 for an illustration). The operation breaks the 3-path u-w-v into three singletons and connects each of them to the respective 2-path to form the replacement collection of three new 3-paths. In the example illustrated in Fig. 4.5, Operation 3-1-By-0-3 replaces the expected collection by three new 3-paths represented by edges highlighted in red.

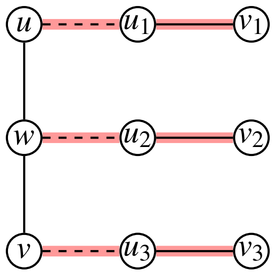


Figure 4.5: The first class of configuration of the expected collection for Operation 3-1-By-0-3, which has priority 2.1, where solid black edges are in Q, dashed edges are in E but outside of Q, and the edges highlighted in red form the replacement collection.

In the second class, which has priority 2.2, two of the three 2-paths, say P_1 and P_2, are adjacent and thus they can be replaced by a new 3-path and a singleton. We determine two configurations in this class (see Fig. 4.6 for illustrations). In the first configuration, the singleton is adjacent to the midpoint w and P_3 is adjacent to one of u and v; in the second configuration, the singleton and P_3 are adjacent to u and v, respectively. For an expected collection of any of the two configurations, the operation replaces it by three new 3-paths.

In the example illustrated in Fig. 4.6a, the singleton is u_1 and P_3 is adjacent to u. Operation 3-1-By-0-3 replaces the expected collection by three new 3-paths represented by edges highlighted in red.In the example illustrated in Fig. 4.6b, the singleton is v_2 and $P_3 = u_3$-v_3 is adjacent to u. Operation 3-1-By-0-3 replaces the expected collection by three new 3-paths represented by edges highlighted in red.

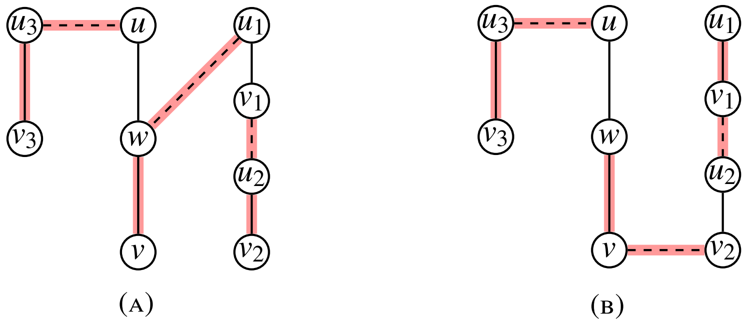


Figure 4.6: The second class of configurations of the expected collection in Operation 3-1-By-0-3, which has priority 2.2, where solid black edges are in Q, dashed edges are in E but outside of Q, and the edges highlighted in red form the replacement collection.

4.2.2.3 Operation 4-1-By-1-3, priority 3

Consider an expected collection of four 2-paths $P_1 = u_1$-v_1, $P_2 = u_2$-v_2, $P_3 = u_3$-v_3, $P_4 = u_4$-v_4, and a 3-path $P_5 = u$-w-v in Q. Note that an Operation 4-1-By-1-3 applies only when Operation 3-0-By-0-2 and Operation 3-1-By-0-3 both fail to apply to the current Q. Thus, we only consider the cases when the four 2-paths can be separated into two pairs, each of which are adjacent in the graph G, and we can replace them by two new 3-paths while leaving two singletons which are adjacent to a common vertex on P_5.

In the configuration for the expected collection in an Operation 4-1-By-1-3, the two singletons must be adjacent to a common endpoint, say u, of P_5 (see Fig. 4.7 for an illustration), then they can be replaced by a new 2-path v-w and a new 3-path. Overall, the operation replaces the expected collection by three new 3-paths and a new 2-path. In the example illustrated in Fig. 4.7, the two singletons are u_1 and u_3, and they are both adjacent to u. Operation 4-1-By-1-3 replaces the expected collection by three new 3-paths and a new 2-path represented by edges highlighted in red.

4.2.2.4 Operation 4-2-By-1-4, lowest priority 4

Consider an expected collection of four 2-paths $P_1 = u_1$-v_1, $P_2 = u_2$-v_2, $P_3 = u_3$-v_3, $P_4 = u_4$-v_4, and two 3-paths $P_5 = u$-w-v, $P_6 = u'$-w'-v' in Q. The four 2-paths can be

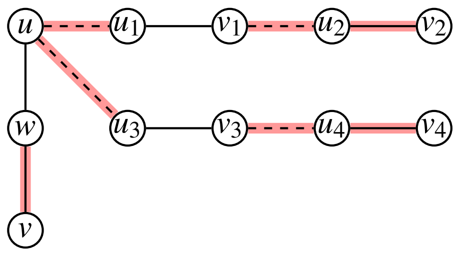


FIGURE 4.7: The configuration of the expected collection for OPERATION 4-1-BY-1-3, which has priority 3, where solid black edges are in Q, dashed edges are in E but outside of Q, and the edges highlighted in red form the replacement collection.

separated into two pairs, each of which are adjacent in the graph G, thus we can replace them by two new 3-paths while leaving two singletons, which are adjacent to P_5 and P_6, respectively (see Fig. 4.8 for illustrations). We determine three classes of configurations for the expected collection in this operation, for which the replacement collection consists of four new 3-paths and a new 2-path.

In the first class, the two singletons are adjacent to P_5 and P_6 at endpoints, say u and u', respectively; additionally, one of the five edges (u, v'), (v, u'), (w, v'), (v, w'), (v, v') is in E (see Fig. 4.8a for an illustration). In the example illustrated in the Fig. 4.8a, if $(u, v') \in E$, then OPERATION 4-1-BY-1-3 replaces the expected collection by four new 3-paths and a new 2-path represented by edges highlighted in red.

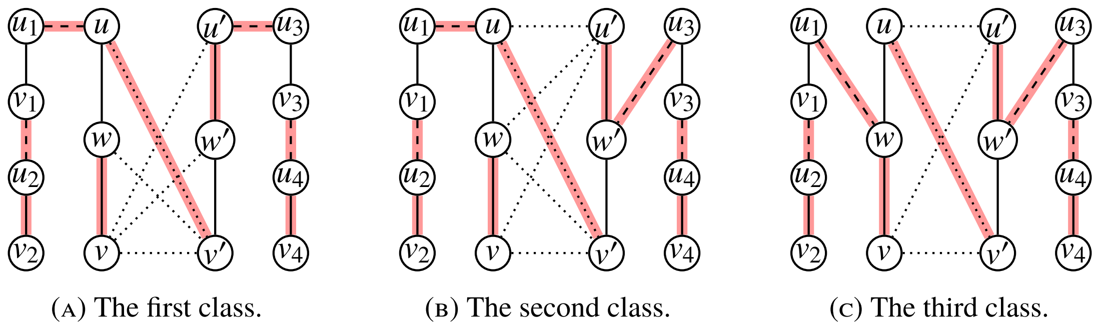


FIGURE 4.8: The three classes of configurations of the expected collections for an OPERATION 4-2-BY-1-4, where solid black edges are in Q, dashed edges are in E but outside of Q, and the edges highlighted in red form a possible replacement collection. In every class, each dotted edge between $P_5 = u\text{-}w\text{-}v$ and $P_6 = u'\text{-}w'\text{-}v'$ corresponds to one configuration.

In the second class, one singleton is adjacent to an endpoint of a 3-path, say u on P_5, and the other singleton is adjacent to the midpoint w' of P_6; additionally, one of the six edges (u, u'), (u, v'), (w, u'), (w, v'), (v, u'), (v, v'), is in E (see Fig. 4.8b for an illustration). In the example illustrated in Fig. 4.8b, if $(u, u') \in E$, then OPERATION 4-1-BY-1-3 replaces the expected collection by four new 3-paths and a new 2-path represented by edges highlighted in red.

In the third class, the two singletons are adjacent to the midpoints of the two 3-paths, w and w', respectively; additionally, one of the four edges (u, u'), (u, v'), (v, u'), (v, v') is in E (see Fig. 4.8c for an illustration). In the example illustrated in Fig. 4.8c, if $(u, u') \in E$, then Operation 4-1-By-1-3 replaces the expected collection by four new 3-paths and a new 2-path represented by edges highlighted in red.

4.2.3 The complete local search algorithm Approx

The first three steps of our local search algorithm Approx is to run Algorithm A to achieve a 3-path partition Q with the least 1-paths. The fourth step is iterative, and in each iteration the algorithm tries to apply one of the four local operations, from the highest priority to the lowest, by finding a corresponding expected collection and determining the subsequent replacement collection. When no expected collection can be found, the fourth step terminates. We denote the last two steps of our algorithm Approx as Algorithm B. The final 3-path partition Q is the output solution. A high-level description of the complete algorithm Approx is illustrated in Figure 4.9. While leaving the performance analysis for Approx to the next section, we give the running time analysis below.

```
Algorithm Approx on $G = (V, E)$:

Step 1-3. Algorithm A:
    compute a 3-path partition $Q$ with the least 1-paths in $G$;
Step 4-5. Algorithm B:
    Step 4. Iteratively perform:
    4.1. if Operation 3-0-By-0-2 applies, update $Q$ and break;
    4.2. if Operation 3-1-By-0-3 with priority 2.1 applies, update $Q$ and break;
    4.3. if Operation 3-1-By-0-3 with priority 2.2 applies, update $Q$ and break;
    4.4. if Operation 4-1-By-1-3 applies, update $Q$ and break;
    4.5. if Operation 4-2-By-1-4 applies, update $Q$ and break;
    Step 5. Return $Q$.
```

Figure 4.9: A high-level description of the local search algorithm Approx, where each "break" is meant to go to the next iteration of the loop.

We know that Algorithm A runs in $O(nm)$ time (Theorem 4.5), where $n = |V|$ and $m = |E|$. Note that there are $O(n)$ 2-paths and $O(n)$ 3-paths in Q at the beginning of each iteration of

Step 4, and therefore there are $O(n^6)$ *original* candidate collections to be examined, since a candidate collection has a maximum size of 6. When a local operation applies, an iteration ends and the 3-path partition Q reduces its size by 1, while introducing at most 5 new 2- and 3-paths. These new 2- and 3-paths give rise to $O(n^5)$ *new* candidate collections to be examined in the subsequent iterations. Since there are at most n iterations in Step 4, we conclude that the total number of original and new candidate collections examined in Step 4 is $O(n^6)$. Determining whether a candidate collection is an expected collection, and if so, deciding the corresponding replacement collection, can be done in $O(1)$ time. We thus prove that the overall running time of Algorithm B is $O(n^6)$, and consequently prove the following theorem.

Theorem 4.8. *The running time of the algorithm Approx is in $O(n^6)$.*

4.3 Analysis of the approximation ratio 4/3

In this section, we show that our local search algorithm Approx is a 4/3-approximation for 3-PP. The performance analysis is done through amortization.

The 3-path partition produced by the algorithm Approx is denoted as Q; let Q_i denote the sub-collection of i-paths in Q, for $i = 1, 2, 3$, respectively. Let Q^* be an optimal 3-path partition, *i.e.*, it achieves the minimum total number of paths, and let Q_i^* denote the sub-collection of i-paths in Q^*, for $i = 1, 2, 3$, respectively. Since our Q contains the least 1-paths among all 3-path partitions for G, we have

$$|Q_1| \le |Q_1^*|. \tag{4.2}$$

Since both Q and Q^* cover all the vertices of V, we have

$$|Q_1| + 2|Q_2| + 3|Q_3| = n = |Q_1^*| + 2|Q_2^*| + 3|Q_3^*|. \tag{4.3}$$

Next, we prove the following inequality which gives an upper bound on $|Q_2|$, through an amortized analysis:

$$|Q_2| \le |Q_1^*| + 2|Q_2^*| + |Q_3^*|. \tag{4.4}$$

Combining Eqs. (4.2, 4.3, 4.4), it follows that

$$3|Q_1| + 3|Q_2| + 3|Q_3| \le 4|Q_1^*| + 4|Q_2^*| + 4|Q_3^*|, \tag{4.5}$$

that is, $|Q| \le \frac{4}{3}|Q^*|$, and consequently the following theorem holds.

Theorem 4.9. *The algorithm* Approx *is an $O(n^6)$-time $4/3$-approximation for the 3-PP problem, and the performance ratio $4/3$ is tight for* Approx.

In the amortized analysis, each 2-path of Q_2 has one token (*i.e.*, $|Q_2|$ tokens in total) to be distributed to the paths of Q^*. The upper bound in Eq. (4.4) will immediately follow if we prove the following lemma.

Lemma 4.10. *There is a distribution scheme in which*

1. *every 1-path of Q_1^* receives at most 1 token;*
2. *every 2-path of Q_2^* receives at most 2 tokens;*
3. *every 3-path of Q_3^* receives at most 1 token.*

In the rest of the section we present the distribution scheme that satisfies the three requirements stated in Lemma 4.10.

Denote $E(Q_2)$, $E(Q_3)$, $E(Q_2^*)$, $E(Q_3^*)$ as the set of all the edges on the paths of Q_2, Q_3, Q_2^*, Q_3^*, respectively, and $E(Q^*) = E(Q_2^*) \cup E(Q_3^*)$. In the subgraph of $G\big(V, E(Q_2) \cup E(Q^*)\big)$, only the midpoint of a 3-path of Q_3^* may have degree 3, *i.e.*, incident with two edges of $E(Q^*)$ and an edge of $E(Q_2)$, while all the other vertices have degree at most 2 since each is incident with at most one edge of $E(Q_2)$ and at most one edge of $E(Q^*)$.

Our distribution scheme consists of two phases. We define two functions $\tau_1(P)$ and $\tau_2(P)$ to denote the fractional amount of token received by a path $P \in Q^*$ in Phase 1 and Phase 2, respectively; we also define the function $\tau(P) = \tau_1(P) + \tau_2(P)$ to denote the total amount of token received by the path $P \in Q^*$ at the end of our distribution process. Then, we have $\sum_{P \in Q^*} \tau(P) = |Q_2|$.

4.3.1 Distribution process Phase 1

In Phase 1, we distribute all the $|Q_2|$ tokens to the paths of Q^* (*i.e.*, $\sum_{P \in Q^*} \tau_1(P) = |Q_2|$) such that a path $P \in Q^*$ receives some token from a 2-path u-$v \in Q_2$ only if u or v is (or both are) on P, and the following three requirements are satisfied:

1. $\tau_1(P_i) \le 1$ for $\forall P_i \in Q_1^*$;

2. $\tau_1(P_j) \le 2$ for $\forall P_j \in Q_2^*$;
3. $\tau_1(P_\ell) \le 3/2$ for $\forall P_\ell \in Q_3^*$.

In this phase, the one token held by each 2-path of Q_2 is breakable but can only be broken into two halves. Thus for every path $P \in Q^*$, $\tau_1(P)$ is a multiple of $1/2$.

For each 2-path u-$v \in Q_2$, at most one of u and v can be a singleton of Q^*. If $P_1 = v \in Q_1^*$, then the whole 1 token of the path u-v is distributed to v, that is, $\tau_1(v) = 1$ (see Fig. 4.10a for an illustration). This way, we have $\tau_1(P) \le 1$ for $\forall P \in Q_1^*$.

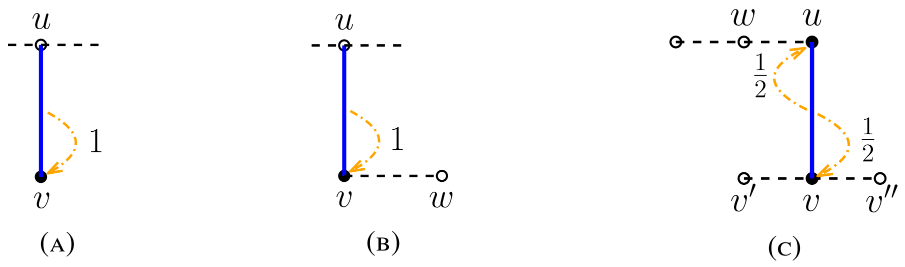


FIGURE 4.10: Illustrations of the token distribution scheme in Phase 1, where solid edges are in $E(Q_2)$ and dashed edges are in $E(Q^*)$. In Fig. 4.10c, u or v can be either an endpoint or the midpoint of the corresponding 3-path of Q_3^*.

For a 2-path u-$v \in Q_2$, we consider the cases when both u and v are incident with an edge of $E(Q^*)$. If one of u and v, say v, is incident with an edge of $E(Q_2^*)$, that is, v is on a 2-path $P_1 = v$-$w \in Q_2^*$, then the 1 token of the path u-v is given to the path $P_1 \in Q_2^*$ (see Fig. 4.10b for an illustration). Note that if u is also on a 2-path $P_2 \in Q_2^*$ and $P_2 \ne P_1$, then the path P_2 receives no token from the path u-v. The choice of which one of the two vertices u and v comes first does not matter. This way, we have $\tau_1(P) \le 2$ for $\forall P \in Q_2^*$ since the 2-path $P_1 \in Q_2^*$ might receive another token from a 2-path of Q_2 incident on w.

Next, we consider the cases for a 2-path u-$v \in Q_2$ in which each of u and v is incident with an edge of $E(Q_3^*)$. Consider a 3-path $P_1 \in Q_3^*$: v'-v-v''. We distinguish two cases for a vertex of P_1 to determine the amount of token received by P_1 (see Fig. 4.10c for an illustration). In the first case, either the vertex, say v', is not on any path of Q_2 or it is on a path of Q_2 with 0 token left, then P_1 receives no token *through vertex* v'. In the second case, the vertex, say v (the following argument applies the same to the other two vertices v' and v''), is on a path u-$v \in Q_2$ holding 1 token, and consequently u must be on a 3-path $P_2 \in Q_3^*$, then the 1 token of u-v is broken into two halves, with $1/2$ token distributed to P_1 through vertex v and the other $1/2$ token distributed to P_2 through vertex u. This way, we have $\tau_1(P) \le 3/2$ for $\forall P \in Q_3^*$ since the 3-path $P_1 \in Q_3^*$ might receive another $1/2$ token through each of v' and v''.

4.3.2 Distribution process Phase 2

In Phase 2, we will transfer the extra $1/2$ token from every 3-path $P \in Q_3^*$ with $\tau_1(P) = 3/2$ to some other paths of Q^* in order to satisfy the three requirements of Lemma 4.10. In this phase, each $1/2$ token can be broken into two quarters, thus for a path $P \in Q^*$, $\tau_2(P)$ is a multiple of $1/4$.

Consider a 3-path $P_1 = v''\text{-}v'\text{-}v \in Q_3^*$. We observe that if $\tau_1(P_1) = 3/2$, then each of v, v', and v'' must be incident with an edge of $E(Q_2)$, the other endpoint of which must also be on a 3-path of Q_3^*. One of the three vertices, say v, on an edge $(u, v) \in E(Q_2)$, must have its corresponding u outside of P_1. Denote P_2 as the 3-path of Q_3^* where u is on. Let w be a vertex adjacent to u on P_2, *i.e.*, (u, w) is an edge on P_2. (See Fig. 4.11 for an illustration.) We can verify the following claim.

Claim 4.11. *w must be on a 3-path of Q_3, being either an endpoint or the midpoint.*

Proof. See Fig. 4.11 for an illustration. Firstly, w cannot collide into any of u', u'' since otherwise the three 2-paths $u\text{-}v$, $u'\text{-}v'$, $u''\text{-}v''$ could be replaced due to Operation 3-0-By-0-2. Then, suppose w is on a 2-path $w\text{-}x$ of Q_2, then the three 2-paths $u\text{-}v$, $u'\text{-}v'$, $w\text{-}x$ could be replaced due to Operation 3-0-By-0-2. Lastly, suppose w is a singleton of Q_1, then w and the 2-path $u\text{-}v$ could be merged to a 3-path so that Q is not a partition with the least 1-paths, a contradiction. Thus, w cannot be a singleton of Q_1 or on any 2-path of Q_2, and the claim is proved. □

We thus conclude that $\tau_1(P_2) \leq 1$, and we have the following lemma.

Lemma 4.12. *For any 3-path $P_1 \in Q_3^*$ with $\tau_1(P_1) = 3/2$, there must be another 3-path $P_2 \in Q_3^*$ with $\tau_1(P_2) \leq 1$ such that*

1. *$u\text{-}v$ is a 2-path of Q_2, where v is on P_1 and u is on P_2, and*
2. *any vertex adjacent to u on P_2 must be on a 3-path P_3 of Q_3.*

The first step of Phase 2 is to transfer this extra $1/2$ token back from P_1 to the 2-path $u\text{-}v$ through vertex v (see Fig. 4.11 for an illustration). Thus, we have $\tau_2(P_1) = -1/2$ and $\tau(P_1) = 3/2 - 1/2 = 1$.

Using Lemma 4.12 and all its notations, let x_1 and y_1 be the other two vertices on P_3 ($P_3 = w\text{-}x_1\text{-}y_1$ or $P_3 = x_1\text{-}x\text{-}y_1$). Denote $P_4 \in Q^*$ ($P_5 \in Q^*$, respectively) as the path where

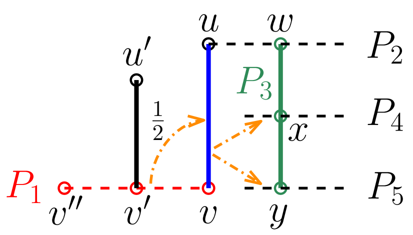


FIGURE 4.11: An illustration of a 3-path $P_1 = v\text{-}v'\text{-}v'' \in Q_3^*$ with $\tau_1(P_1) = 3/2$, where $u\text{-}v$, $u'\text{-}v' \in E(Q_2)$, $P_3 \in Q_3$, with w being either the midpoint or an endpoint of P_3, and $P_2 \in E(Q_3^*)$ is represented by dashed edges, on which w is adjacent to u.

x_1 (y_1, respectively) is on. Next, we will transfer the $1/2$ token from $u\text{-}v$ to the paths P_4 or/and P_5 through some *pipe* or *pipes*.

We define a *pipe* $r \longrightarrow s \longrightarrow t$, where r is an endpoint of a 2-path of Q_2 which receives $1/2$ token in the first step of Phase 2, (r, s) is an edge on a 3-path $P' \in Q_3^*$ with $\tau_1(P') \leq 1$ ($P' = P_2$ here), s and t are both on a 3-path of Q_3 (P_3 here), and t is a vertex on our destination path of Q^* (P_4 or P_5 here) which will receive token from the 2-path of Q_2. r and t are called the *head* and *tail* of the pipe, respectively. For example, in Fig. 4.12a, there are four possible pipes $u \longrightarrow w \longrightarrow x_1$, $u \longrightarrow w \longrightarrow y_1$, $u'' \longrightarrow w \longrightarrow x_1$, and $u'' \longrightarrow w \longrightarrow y_1$. We distinguish the cases, on which of Q_1^*, Q_2^*, Q_3^* the two paths P_4 and P_5 belong to, to determine how they receive more token through some pipe or pipes.

Recall that u can be either an endpoint or the midpoint of P_2. We distinguish the following cases with u being an endpoint of P_2 (the cases for u being the midpoint can be discussed the same), that is, $P_2 = u\text{-}w\text{-}u''$, depending on which of Q_1^*, Q_2^*, Q_3^* the two paths P_4 and P_5 belong to, to determine the upper bounds on $\tau(P_4)$ and $\tau(P_5)$.

Case 1. At least one of P_4 and P_5 is a singleton of Q_1^*, say $P_4 = x_1 \in Q_1^*$ (see Fig. 4.12 for illustrations). In this case, we have $\tau_1(P_4) = 0$, so we transfer the $1/2$ token from $u\text{-}v$ to P_4 through pipe $u \longrightarrow w \longrightarrow x_1$. We observe that if P_5 is also a 3-path of Q_3^*, with (y_1, y_2) being an edge on P_5, then $y_2 \longrightarrow y_1 \longrightarrow x_1$ is a candidate pipe through which P_4 could receive another $1/2$ token. We distinguish the following two sub-cases based on whether w is an endpoint or the midpoint of P_3 to determine all the possible pipes through each of which could P_4 receive $1/2$ token.

Sub-case 1.1. w is the midpoint of $P_3 = x_1\text{-}w\text{-}y_1$ (see Fig. 4.12a for an illustration). If $P_5 \in Q_3^*$, with (y_1, y_2) being an edge on P_5, then y_2 cannot be on a 2-path of Q_2 (suppose y_2 is on a 2-path $P'' \in Q_2$, then the three 2-paths $u\text{-}v$, $u'\text{-}v'$, P'', and the 3-path P_3 could be replaced due to OPERATION 3-1-BY-0-3). Therefore, only through pipe $u'' \longrightarrow w \longrightarrow$

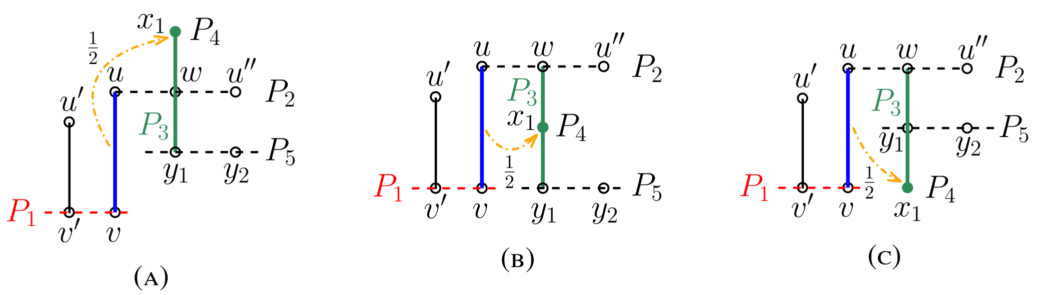


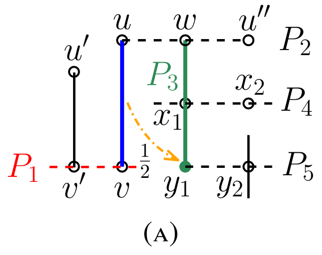
FIGURE 4.12: The cases when P_4 is a singleton of Q_1^*, where solid edges are in $E(Q_2)$ or $E(Q_3)$ and dashed edges are in $E(Q^*)$. x_1 is the tail of the pipe through which P_4 could receive 1/2 token from the 2-path of u-v.

x_1 could P_4 receive another 1/2 token. Thus, $\tau_2(P_4) \le 1/2 \times 2 = 1$, implying $\tau(P_4) \le 0 + 1 = 1$.

Sub-case 1.2. w is an endpoint of P_3, *i.e.*, either $P_3 = w$-x_1-y_1 (see Fig. 4.12b for an illustration) or $P_3 = w$-y_1-x_1 (see Fig. 4.12c for an illustration). In each sub-case, u'' cannot be the head of any pipe (*i.e.*, there does not exist a path u''-v''-v'''-u''', where u''-v'', v'''-$u''' \in Q_2$ and v''-$v''' \in Q_2^*$, since otherwise, the four 2-paths u-v, u'-v', u''-v'', v'''-u''', and the 3-path P_3 could be replaced due to OPERATION 4-1-BY-1-3). If $P_5 \in Q_3^*$, with (y_1, y_2) being an edge on P_5, then y_2 in Fig. 4.12b cannot be on a 2-path of Q_2 (suppose y_2 is on a 2-path $P'' \in Q_2$, then the three 2-paths u-v, u'-v', P'', and the 3-path P_3 could be replaced due to OPERATION 3-1-BY-0-3); y_2 in Fig. 4.12c cannot be the head of any pipe (*i.e.*, there does not exist a path y_2-z-z'-y', where y_2-z, z'-$y' \in Q_2$ and z-$z' \in Q_2^*$, since otherwise, the three 2-paths u-v, y_2-z, z'-y', and the 3-path P_3 could be replaced due to OPERATION 3-1-BY-0-3). Therefore, through no other pipe could P_4 receive any other token in either sub-case. Thus, $\tau_2(P_4) \le 1/2$, implying $\tau(P_4) \le 0 + 1/2 = 1/2$.

Case 2. Both P_4 and P_5 are paths of $Q_2^* \cup Q_3^*$ (see Fig. 4.13 for illustrations). We distinguish two sub-cases based on whether w is an endpoint or the midpoint of P_3 to determine how to transfer the 1/2 token from u-v to P_4 or P_5 or both.

Sub-case 2.1. w is an endpoint of $P_3 = w$-x_1-y_1, with y_1 on P_5 (see Fig. 4.13a for an illustration). In this sub-case, we transfer the 1/2 token from u-v to P_5 through pipe $u \to w \to y_1$. Similar to the sub-case shown in Fig. 4.12b, if (y_1, y_2) is an edge on P_5, then y_2 cannot be on a 2-path of Q_2 due to OPERATION 3-1-BY-0-3. Thus, through no other pipe



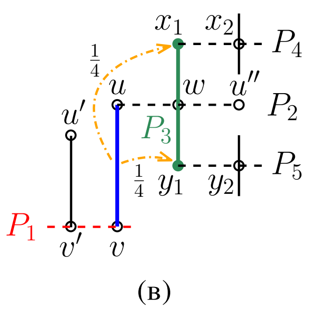


FIGURE 4.13: The cases when both P_4 and P_5 are in $Q_2^* \cup Q_3^*$, where solid edges are in $E(Q_2)$ or $E(Q_3)$ and dashed edges are in $E(Q^*)$. In Fig. 4.13a, y_1 is the tail of the pipe through which P_5 receives $1/2$ token from the 2-path u-v; in Fig. 4.13b, x_1 is the tail of the pipe through which P_4 receives $1/4$ token from the 2-path u-v and y_1 is the tail of the pipe through which P_5 receives $1/4$ token from the 2-path u-v.

with tail y_1 could P_5 receive any other token. Therefore, P_5 could receive at most $1/2$ token through pipes with tail y_1.

Sub-case 2.2. w is the midpoint of $P_3 = x_1$-w-y_1 (see Fig. 4.13b). In this sub-case, we break the $1/2$ token holding by u-v into two quarters, with $1/4$ transferred to P_4 through pipe $u \to w \to x_1$ and the other $1/4$ transferred to P_5 through pipe $u \to w \to y_1$. Similar to the sub-case shown in Fig. 4.12a, if (x_1, x_2) is an edge on P_4 (or (y_1, y_2) is an edge on P_5, respectively), then x_2 (or y_2, respectively) cannot be on a 2-path of Q_2 due to OPERATION 3-1-BY-0-3. Thus, only through pipe $u'' \to w \to x_1$ could P_4 receive another $1/4$ token and only through pipe $u'' \to w \to y_1$ could P_5 receive another $1/4$ token. Therefore, P_4 (P_5, respectively) could receive at most $1/2$ token through pipes with tail x_1 (y_1, respectively).

Now we discuss if P_4 in Fig. 4.13b and P_5 in Fig. 4.13a and Fig. 4.13b could receive more token through pipes with vertices other than x_1 and y_1 being the tail, respectively. Let (x_1, x_2) and (y_1, y_2) be edges on P_4 and P_5, respectively. We first prove the following two claims.

Claim 4.13. *y_2 in both Fig. 4.13a and Fig. 4.13b, and x_2 in Fig. 4.13b must each be on a 3-path of Q_3.*

Proof. Firstly, we have already proved in the discussions for sub-cases 2.1 and 2.2 that y_2 in both Fig. 4.13a, Fig. 4.13b, and x_2 in Fig. 4.13b cannot be on a 2-path of Q_2. Suppose x_2 in Fig. 4.13b is a singleton of Q_1, then the 3-path P_3 and the edge (x_1, x_2) could be reconnected into two 2-paths, implying Q not a partition with the least 1-paths, a contradiction. This argument also applies to y_2 in Fig. 4.13a and Fig. 4.13b. Thus, the claim is proved. □

Claim 4.13 implies that for P_5 in Fig. 4.13a or 4.13b (P_4 in Fig. 4.13b, respectively), we have $\tau_1(P_5) \le 1/2$ ($\tau_1(P_4) \le 1/2$, respectively).

Claim 4.14. *Any of y_2 in Fig. 4.13a or Fig. 4.13b, or x_2 in Fig. 4.13b cannot be the tail of a pipe.*

Proof. We only prove that y_2 in Fig. 4.13a cannot be the tail of a pipe, then the same argument will also apply to y_2 in Fig. 4.13b and x_2 in Fig. 4.13b. Suppose y_2 in Fig. 4.13a is the tail of a pipe, say $z_1 \to w' \to y_2$. That is, y_2 and w' are on the same 3-path, say P_3', of Q_3; there is a path w'-z_1-z_2-z_3-z_4, where w'-z_1, z_2-$z_3 \in Q_2^*$ and z_1-z_2, z_3-$z_4 \in Q_2$. (See Fig. 4.14 for an illustration.) Then the four 2-paths u-v, u'-v', z_1-z_2, z_3-z_4, and the two 3-paths P_3 and P_3' could be replaced due to Operation 4-2-By-1-4. Thus, the claim is proved. □

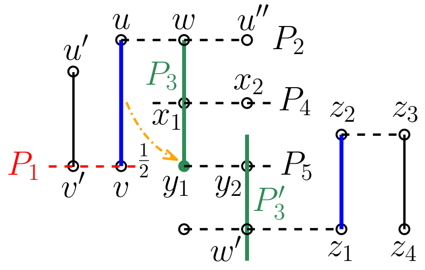


Figure 4.14: An illustration of y_2 in Fig. 4.13a being the tail of a pipe, say $z_1 \to w' \to y_2$, which could never happen due to Operation 4-2-By-1-4.

Claim 4.14 implies that through no pipe with tail y_2 in both Fig. 4.13a and 4.13b (with tail x_2 in Fig. 4.13b, respectively) could P_5 (P_4, respectively) receive any other token. That is, if P_5 (P_4, respectively) is a 2-path or a 3-path with y_1 (x_1, respectively) being the midpoint, then it could receive token only through pipes with tail y_1 (x_1, respectively), thus we have $\tau_2(P_5) \le 1/2$ ($\tau_2(P_4) \le 1/2$, respectively).

Next, we discuss the cases when P_5 in Fig. 4.13a is a 3-path, with y_1 being an endpoint (the following argument also applies to the cases when P_5 in Fig. 4.13b is a 3-path, with y_1 being an endpoint, and the cases when P_4 in Fig. 4.13b is a 3-path with x_1 being an endpoint). Let $P_5 = y_1$-y_2-y_3 (see Fig. 4.15 for an illustration). According to Claim 4.14, P_5 could only receive token through pipes with tail y_1 or y_3. We distinguish the following three cases based on whether y_3 is on a path of Q_1, or Q_2, or Q_3.

- If y_3 is a singleton of Q_1, then we have $\tau_1(P_5) = 0$, thus with the $1/2$ token received through pipe $u \longrightarrow w \longrightarrow y_1$, we have $\tau_2(P_5) \leq 1/2$, implying $\tau(P_5) \leq 1/2$.
- If y_3 is on a 2-path of Q_2, then we have $\tau_1(P_5) \leq 1/2$, thus with the $1/2$ token received through pipe $u \longrightarrow w \longrightarrow y_1$, we have $\tau_2(P_5) \leq 1/2$, implying $\tau(P_5) \leq 1$.
- If y_3 is on a 3-path of Q_3, then we have $\tau_1(P_5) = 0$. y_3 could either be the tail of one pipe as y_1 in sub-case 2.1 (Fig. 4.13a), or be the tail of at most two pipes as x_1 or y_1 in sub-case 2.2 (Fig. 4.13b). For any of these sub-cases, P_5 could receive at most $1/2$ token through pipes with tail y_3. Thus, with the $1/2$ token received through pipe $u \longrightarrow w \longrightarrow y_1$, we have $\tau_2(P_5) \leq 1/2 + 1/2 = 1$, implying $\tau(P_5) \leq 0 + 1 = 1$.

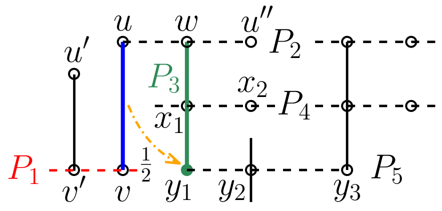


FIGURE 4.15: An illustration of y_1 being an endpoint of P_5 in Fig. 4.13a, where $P_5 = y_1$-y_2-y_3, solid edges are in $E(Q_2)$ or $E(Q_3)$ and dashed edges are in $E(Q^*)$. y_3 could be on a path of Q_1, Q_2, or Q_3.

From the above two cases, we conclude that for any $P \in \{P_4, P_5\}$, if $\tau_2(P) > 0$, then we have $\tau_1(P) \leq 1/2$ and $\tau_2(P) \leq 1$, and it falls into one of the following four scenarios:

1. If w is an endpoint of P_3 and $\tau_1(P) = 0$, then there are at most two pipes through each of which could P receive $1/2$ token. That is, $\tau_2(P) \leq 1/2 \times 2 = 1$, implying $\tau(P) \leq 0 + 1 = 1$.
2. If w is an endpoint of P_3 and $\tau_1(P) = 1/2$, then only through one pipe could P receive the $1/2$ token. That is, $\tau_2(P) \leq 1/2$, implying $\tau(P) \leq 1/2 + 1/2 = 1$.
3. If w is the midpoint of P_3 and $\tau_1(P) = 0$, then there are at most four pipes through each of which could P receive $1/4$ token. That is, $\tau_2(P) \leq 1/4 \times 4 = 1$, implying $\tau(P) \leq 0 + 1 = 1$.
4. If w is the midpoint of P_3 and $\tau_1(P) = 1/2$, then there are at most two pipes through each of which could P receive $1/4$ token. That is, $\tau_2(P) \leq 1/4 \times 2 = 1/2$, implying $\tau(P) \leq 1/2 + 1/2 = 1$.

In summary, for any $P_1 \in Q^*$ with $\tau_1(P_1) = 3/2$, we have $\tau_2(P_1) = -1/2$; for any $P \in Q^*$ with $\tau_2(P) > 0$, we have $\tau_1(P) = 0$ if $\tau_2(P) \leq 1$, or $\tau_1(P) \leq 1/2$ if $\tau_2(P) \leq 1/2$. Therefore, at the end of Phase 2, we have

1. $\tau(P_i) \leq 1$ for $\forall P_i \in Q_1^*$,
2. $\tau(P_j) \leq 2$ for $\forall P_j \in Q_2^*$,
3. $\tau(P_\ell) \leq 1$ for $\forall P_\ell \in Q_3^*$.

This proves Lemma 4.10.

4.3.3 A tight instance of algorithm Approx

Figure 4.16 illustrates a tight instance, in which our solution 3-path partition Q contains nine 2-paths and three 3-paths (solid edges) and an optimal 3-path partition Q^* contains nine 3-paths (dashed edges). Each 3-path of Q^* receives 1 token from the 2-paths in Q in our distribution process. This instance shows that the performance ratio of 4/3 is tight for Approx.

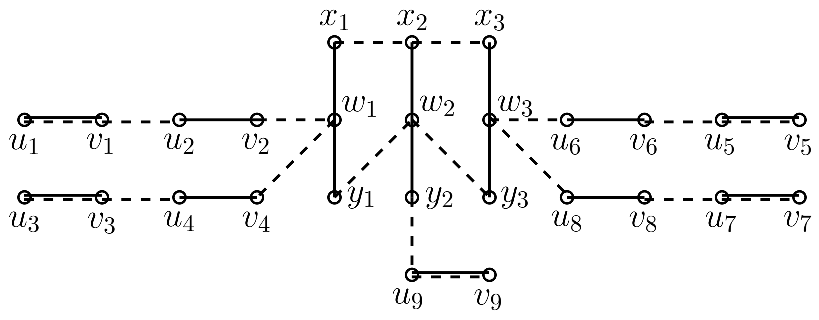


Figure 4.16: A tight instance of 27 vertices, where solid edges represent a 3-path partition Q produced by Approx and dashed edges represent an optimal 3-path partition Q^*. The edges (u_i, v_i), $i = 1, 3, 5, 7, 9$, are in $E(Q_2) \cap E(Q^*)$, shown in both solid and dashed edges. In our distribution process, each of the nine 3-paths in Q^* receives 1 token from the 2-paths in Q.

4.4 Concluding remarks and possible future work

In this chapter, we studied the 3-PP problem and designed a 4/3-approximation algorithm Approx for 3-PP. Approx contains two algorithms A and B, with Algorithm A computing a 3-path partition Q with the least 1-paths in $O(nm)$-time first, and then Algorithm B reduces the total number of paths in Q by repeatedly finding a certain collection of 2- and 3-paths and replacing it with a smaller size collection of new 2- and 3-paths in $O(n^6)$ time, thus the overall running time of Approx is $O(n^6)$. Algorithm A is already a $k/2$-approximation for the general k-PP problem. The performance ratio $4/3$ of Approx is proved by an amortization scheme, using the structure properties of the 3-path partition returned by Approx, matching the current best approximation ratio for the 3-SC problem. In addition, we also showed that the performance ratio $4/3$ is tight for our algorithm Approx by giving an instance in Figure 4.16.

Since the k-PP and the k-SC problems are very closely related, it would be interesting to see if there is a better than $4/3$-approximation for either of them by investigating better properties on both the special cases of $k = 3$. Also, due to the non-existence of any approximation algorithm for the general k-PP problem with proven performance except a trivial k-approximation, it would also be worthwhile to design an algorithm for k-PP with an approximation ratio better than k.

Chapter 5

Conclusions and Future Work

In this thesis, we concentrated on the design and analysis of approximation algorithms for the following interesting **NP**-hard combinatorial optimization problems:

- maximum happy vertices (MHV) and minimum unhappy vertices (MUHV),
- maximum duo-preservation string mapping (Max-Duo),
- k-path partition (k-PP).

For the MHV problem, we presented a lower bound of $\max\{2/k, 1/\big(\Delta + 1/g(\Delta)\big)\}$ and an upper bound of $\Omega(\log^2 k/k)$ on the approximability; for the MUHV problem, we closed the gap between the upper and lower bounds on the approximability, which are both $(2 - 2/k)$. The main methods used in obtaining these results are some randomized rounding techniques based on linear programing relaxation and some polynomial time reduction proofs from and to some related problems.

For the Max-Duo problem, a lower bound on its approximability is 1.00042 [10], and we proposed a $(1.4 + \epsilon)$-approximation algorithm for 2-Max-Duo and a 2.917-approximation algorithm for general Max-Duo, while the current best result is a $(2+\epsilon)$-approximation [33].

For the k-PP problem, we proposed a $4/3$-approximation algorithm for $k = 3$, including a $k/2$-approximation for $k \geq 3$.

All the approximability results presented in this thesis maintain to be the current best results, except for the 2.917-approximation for general Max-Duo. The main techniques used in obtaining the results for these problems are local search and amortized analysis.

Given all the approximability results presented in this thesis, one direction of possible future work is trying to close the gaps between the lower and upper bounds on the approximability of the problems we have investigated. On the positive side, we may consider designing new algorithms with possibly better approximation guarantees; on the negative side, we may try to improve the inapproximability bounds. Specifically, based on our approximation results on special cases of the Max-Duo and k-PP problems, we want to design improved approximation algorithms for the general Max-Duo and k-PP problems.

In addition, all the approximation results for Max-Duo and k-PP are based on the idea of trying to find the local optimum solutions, and our observations on some good properties of the local structures finally lead to these approximation results. Both the performance guarantees of the local search algorithms for Max-Duo and 3-PP are proved by amortized analysis, which seems to be a very interesting and effective technique in proving the performance ratio of some certain kind of algorithms (especially local search). Thus, another direction of possible future work would be to explore more applications for local search and amortization.

At last, I want to remark that during my Ph.D. studies, I have also made contributions to the following publications which are not included in this thesis.

1. L. Liu and Y. Chen and J. Dong and R. Goebel and G. Lin and Y. Luo and G. Ni and B. Su and **Y. Xu** and A. Zhang. Approximation algorithms for the three-machine proportionate mixed-shop scheduling. Submission under review.
2. W. Luo, B. Su, **Y. Xu**, and G. Lin. An approximation framework for bounded facility location problems. In *The 24th International Computing and Combinatorics Conference (COCOON 2018)*, volume 10976 of *LNCS*, pages 353–364, 2018.
3. W. Luo and **Y. Xu** and B. Gu and W. Tong and R. Goebel and G. Lin. Algorithms for Communication Scheduling in Data Gathering Network with Data Compression. *Algorithmica*, 80(11):3158–3176, 2018.
4. W. Luo, **Y. Xu**, W. Tong, and G. Lin. Single machine scheduling with job-dependent machine deterioration. In *Proceedings of the 27th International Symposium on Algorithms and Computation (ISAAC 2016)*, volume 64 of *LIPIcs*, pages 55:1–55:13, 2016.
5. S. Zhai and P. Zhang and D. Zhu and W. Tong and **Y. Xu** and G. Lin. An approximation algorithm for genome sorting by reversals to recover all adjacencies. *Journal of Combinatorial Optimization*, 2018.

Bibliography

[1] H. Angelidakis, Y. Makarychev, and P. Manurangsi. An improved integrality gap for the călinescu-karloff-rabani relaxation for multiway cut. In *Proceedings of the 19th Integer Programming and Combinatorial Optimization (IPCO 2017)*, pages 39–50, 2017.

[2] S. Arora and B. Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[3] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and approximation: Combinatorial optimization problems and their approximability properties*. Springer Science & Business Media, 2012.

[4] P. Austrin, S. Khot, and M. Safra. Inapproximability of vertex cover and independent set in bounded degree graphs. In *24th Annual IEEE Conference on Computational Complexity*, pages 74–80, 2009.

[5] S. Beretta, M. Castelli, and R. Dondi. Corrigendum to "Parameterized tractability of the maximum-duo preservation string mapping problem" [646(2016), 16–25]. *Theoretical Computer Science*, 653:108–110, 2016.

[6] S. Beretta, M. Castelli, and R. Dondi. Parameterized tractability of the maximum-duo preservation string mapping problem. *Theoretical Computer Science*, 646:16–25, 2016.

[7] P. Berman and T. Fujito. On approximation properties of the independent set problem for low degree graphs. *Theory of Computing Systems*, 32:115–132, 1999.

[8] P. Berman and M. Karpinski. On some tighter inapproximability results. In *Proceedings of the of 26th International Colloquium on Automata, Languages and Programming (ICALP'99)*, pages 200–209, 1999.

[9] N. Boria, G. Cabodi, P. Camurati, M. Palena, P. Pasini, and S. Quer. A 7/2-approximation algorithm for the maximum duo-preservation string mapping problem. In *Proceedings of the 27th Annual Symposium on Combinatorial Pattern Matching (CPM 2016)*, volume 54 of *LIPIcs*, pages 11:1–11:8, 2016.

[10] N. Boria, A. Kurpisz, S. Leppänen, and M. Mastrolilli. Improved approximation for the maximum duo-preservation string mapping problem. In *Proceedings of the 14th International Workshop on Algorithms in Bioinformatics (WABI 2014)*, volume 8701 of *LNBI*, pages 14–25, 2014.

[11] R. L. Brooks. On colouring the nodes of a network. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 37, pages 194–197, 1941.

[12] B. Brubach. Further improvement in approximating the maximum duo-preservation string mapping problem. In *Proceedings of the 16th International Workshop on Algorithms in Bioinformatics (WABI 2016)*, volume 9838 of *LNBI*, pages 52–64, 2016.

[13] N. Buchbinder, J. S. Naor, and R. Schwartz. Simplex partitioning via exponential clocks and the multiway cut problem. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC 2013)*, pages 535–544. ACM, 2013.

[14] N. Buchbinder, R. Schwartz, and B. Weizman. Simplex transformations and the multiway cut problem. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)*, pages 2400–2410, 2017.

[15] L. Bulteau, G. Fertin, C. Komusiewicz, and I. Rusu. A fixed-parameter algorithm for minimum common string partition with few duplications. In *Proceedings of the 13th International Workshop on Algorithms in Bioinformatics (WABI 2013)*, volume 8126 of *LNBI*, pages 244–258, 2013.

[16] L. Bulteau and C. Komusiewicz. Minimum common string partition parameterized by partition size is fixed-parameter tractable. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'14)*, pages 102–121, 2014.

[17] G. Călinescu, H. Karloff, and Y. Rabani. An improved approximation algorithm for multiway cut. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing (STOC 1998)*, pages 48–52. ACM, 1998.

[18] C. Chekuri and A. Ene. Approximation algorithms for submodular multiway partition. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 807–816. IEEE, 2011.

[19] C. Chekuri and A. Ene. Submodular cost allocation problem and applications. In *Automata, Languages and Programming*, pages 354–366. Springer, 2011.

[20] C. Chekuri and A. Ene. Submodular cost allocation problem and applications. *arXiv*, 1105.2040, 2011.

[21] W. Chen, Z. Chen, N. F. Samatova, L. Peng, J. Wang, and M. Tang. Solving the maximum duo-preservation string mapping problem with linear programming. *Theoretical Computer Science*, 530:1–11, 2014.

[22] X. Chen, J. Zheng, Z. Fu, P. Nan, Y. Zhong, S. Lonardi, and T. Jiang. Assignment of orthologous genes via genome rearrangement. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2:302–315, 2005.

[23] Y. Chen, R. Goebel, G. Lin, L. Liu, B. Su, W. Tong, **Y. Xu**, and A. Zhang. A local search $4/3$-approximation algorithm for the minimum 3-path partition problem. Submission under review.

[24] Y. Chen, R. Goebel, G. Lin, B. Su, **Y. Xu**, and A. Zhang. An improved approximation algorithm for the minimum 3-path partition problem. *Journal of Combinatorial Optimization (JOCO)*, pages 1–15, 2019.

[25] Y. Chen, G. Lin, T. Liu, T. Luo, B. Su, **Y. Xu**, and P. Zhang. A $(1.4+\epsilon)$-approximation algorithm for the 2-max-duo problem. Submission under review.

[26] M. Chrobak, P. Kolman, and J. Sgall. The greedy algorithm for the minimum common string partition problem. In *Proceedings of the 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2004) and the 8th International Workshop on Randomization and Computation (RANDOM 2004)*, volume 3122 of *LNCS*, pages 84–95, 2004.

[27] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT press Cambridge, 2001.

[28] G. Cormode and S. Muthukrishnan. The string edit distance matching problem with moves. *ACM Transactions on Algorithms*, 3:2:1–2:19, 2007.

[29] W. H. Cunningham and L. Tang. *Optimal 3-terminal cuts and linear programming*. Springer, 1999.

[30] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM Journal on Computing*, 23:864–894, 1994.

[31] P. Damaschke. Minimum common string partition parameterized. In *Proceedings of the 8th International Workshop on Algorithms in Bioinformatics (WABI 2008)*, volume 5251 of *LNBI*, pages 87–98, 2008.

[32] S. Dobzinski and M. Schapira. An improved approximation algorithm for combinatorial auctions with submodular bidders. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithm*, pages 1064–1073. Society for Industrial and Applied Mathematics, 2006.

[33] B. Dudek, P. Gawrychowski, and P. Ostropolski-Nalewaja. A family of approximation algorithms for the maximum duo-preservation string mapping problem. In *Proceedings of the 28th Annual Symposium on Combinatorial Pattern Matching (CPM 2017)*, volume 78 of *LIPIcs*, 2017.

[34] B. Dudek, P. Gawrychowski, and P. Ostropolski-Nalewaja. A family of approximation algorithms for the maximum duo-preservation string mapping problem. *arXiv*, 1702.02405, 2017.

[35] R. Duh and M. Fürer. Approximation of k-set cover by semi-local optimization. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC'97, pages 256–264, 1997.

[36] D. Easley and J. Kleinberg. *Networks, Crowds, and Markets: Reasoning about a highly connected world.* Cambridge University Press, 2010.

[37] A. Ene, J. Vondrák, and Y. Wu. Local distribution and the symmetry gap: Approximability of multiway partitioning problems. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2013)*, pages 306–325. SIAM, 2013.

[38] U. Feige. A threshold of for approximating set cover. *Journal of the ACM*, 45:634–652, 1998.

[39] U. Feige. On maximizing welfare when utility functions are subadditive. *SIAM Journal on Computing*, 39:122–142, 2009.

[40] U. Feige and J. Vondrak. The allocation problem with submodular utility functions. In *In Proc. of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2006.

[41] D. S. Franzblau and A. Raychaudhuri. Optimal hamiltonian completions and path covers for trees, and a reduction to maximum flow. *The ANZIAM Journal*, 44:193–204, 2002.

[42] A. Freund and H. Karloff. A lower bound of $8/(7 + \frac{1}{k-1})$ on the integrality ratio of the Călinescu-Karloff-Rabani relaxation for multiway cut. *Information Processing Letters*, 75:43–50, 2000.

[43] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman and Company, 1979.

[44] N. Garg, V. V. Vazirani, and M. Yannakakis. Multiway cuts in node weighted graphs. *Journal of Algorithms*, 50:49–61, 2004.

[45] A. V. Goldberg and A. V. Karzanov. Maximum skew-symmetric flows and matchings. *Mathematical Programming*, 100(3):537–568, 2004.

[46] A. Goldstein, P. Kolman, and J. Zheng. Minimum common string partition problem: Hardness and approximations. In *Proceedings of the 15th International Symposium on Algorithms and Computation (ISAAC 2004)*, volume 3341 of *LNCS*, pages 484–495, 2004.

[47] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981.

[48] F. Harary. *Graph Theory*. Addison-Wesley, Reading, MA, 1969.

[49] H. Jiang, B. Zhu, D. Zhu, and H. Zhu. Minimum common string partition revisited. *Journal of Combinatorial Optimization*, 23:519–527, 2012.

[50] D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

[51] D. R. Karger, P. Klein, C. Stein, M. Thorup, and N. E. Young. Rounding algorithms for a geometric embedding of minimum multiway cut. *Mathematics of Operations Research*, 29:436–461, 2004.

[52] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing (STOC 1984)*, pages 302–311. ACM, 1984.

[53] L. Khachiian. Polynomial algorithm in linear programming. In *Akademiia Nauk SSSR, Doklady*, volume 244, pages 1093–1096, 1979.

[54] S. Khot, R. J. Lipton, E. Markakis, and A. Mehta. Inapproximability results for combinatorial auctions with submodular utility functions. *Algorithmica*, 52:3–18, 2008.

[55] J. Kleinberg and E. Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields. *Journal of the ACM*, 49:616–639, 2002.

[56] P. Kolman and T. Waleń. Reversal distance for strings with duplicates: Linear time approximation using hitting set. In *Proceedings of the 4th International Workshop on Approximation and Online Algorithms (WAOA 2006)*, volume 4368 of *LNCS*, pages 279–289, 2006.

[57] P. Kolman and T. Waleń. Approximating reversal distance for strings with bounded number of duplicates. *Discrete Applied Mathematics*, 155:327–336, 2007.

[58] B. Korte and J. Vygen. *Combinatorial optimization*, volume 2. Springer, 2012.

[59] M. Langberg, Y. Rabani, and C. Swamy. Approximation algorithms for graph homomorphism problems. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques: 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2006) and 10th International Workshop on Randomization and Computation (RANDOM 2006)*, pages 176–187. Springer, 2006.

[60] B. Lehmann, D. Lehmann, and N. Nisan. Combinatorial auctions with decreasing marginal utilities. In *Proceedings of the 3rd ACM conference on Electronic Commerce*, pages 18–28. ACM, 2001.

[61] A. Levin. Approximating the unweighted k-set cover problem: Greedy meets local search. In *Proceedings of the 4th International Workshop on Approximation and Online Algorithms (WAOA 2006)*, LNCS 4368, pages 290–301, 2006.

[62] L. Lovász. Three short proofs in graph theory. *Journal of Combinatorial Theory, Series B*, 19(3):269–271, 1975.

[63] L. Lovász. *Submodular functions and convexity*, pages 235–257. Springer, 1983.

[64] J. Monnot and S. Toulouse. The path partition problem and related problems in bipartite graphs. *Operations Research Letters*, 35:677–684, 2007.

[65] K. Okumoto, T. Fukunaga, and H. Nagamochi. Divide-and-conquer algorithms for partitioning hypergraphs and submodular systems. *Algorithmica*, 62:787–806, 2012.

[66] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[67] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and sub-constant error-probability PCP characterization of NP. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC)*, pages 475–484, 1997.

[68] A. Sharma and J. Vondrák. Multiway cut, pairwise realizable distributions, and descending thresholds. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC 2014)*, pages 724–733. ACM, 2014.

[69] K. M. Swenson, M. Marron, J. V. Earnest-DeYoung, and B. M. Moret. Approximating the true evolutionary distance between two genomes. *Journal of Experimental Algorithmics (JEA)*, 12:3–5, 2008.

[70] **Y. Xu**, Y. Chen, G. Lin, T. Liu, T. Luo, and P. Zhang. A $(1.4 + \epsilon)$-approximation algorithm for the 2-max-duo problem. In *Proceedings of the 28th International Symposium on Algorithms and Computation (ISAAC 2017)*, volume 92 of *LIPIcs*, pages 66:1–66:12, 2017.

[71] **Y. Xu**, Y. Chen, T. Luo, and G. Lin. A local search 2.917-approximation algorithm for duo-preservation string mapping. *arXiv*, 1702.01877, 2017.

[72] **Y. Xu**, Y. Chen, P. Zhang, and R. Goebel. Approximation algorithms for vertex happiness. Submission under review.

[73] **Y. Xu**, P. Zhang, R. Goebel, and G. Lin. Approximation algorithms for the vertex happiness. *arXiv*, 1606.03185v2, 2017.

[74] V. V. Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.

[75] D. P. Williamson and D. B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.

[76] J.-H. Yan, G. J. Chang, S. M. Hedetniemi, and S. T. Hedetniemi. k-path partitions in trees. *Discrete Applied Mathematics*, 78:227–233, 1997.

[77] P. Zhang, T. Jiang, and A. Li. Improved approximation algorithms for the maximum happy vertices and edges problems. In *21st International Computing and Combinatorics Conference (COCOON 2015)*, pages 159–170. Springer, 2015.

[78] P. Zhang and A. Li. Algorithmic aspects of homophyly of networks. *Theoretical Computer Science*, 593:117–131, 2015.

[79] P. Zhang, **Y. Xu**, T. Jiang, A. Li, G. Lin, and E. Miyano. Improved approximation algorithms for the maximum happy vertices and edges problems. *Algorithmica*, 80(5):1412–1438, 2017.

[80] L. Zhao, H. Nagamochi, and T. Ibaraki. Greedy splitting algorithms for approximating multiway partition problems. *Mathematical Programming*, 102:167–183, 2005.