Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Αλγόριθμοι και Πολυπλοκότητα: Χρωματισμοί γράφων και υπεργράφων

Διδακτορική διατριβή

Παναγιώτης Χείλαρης

Αθήνα, 2007

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Αλγόριθμοι και Πολυπλοκότητα: Χρωματισμοί γράφων και υπεργράφων

Διδακτορική διατριβή

του

Παναγιώτη Χείλαρη

Τριμελής συμβουλευτική επιτροπή:	Επιβλέπων:	Ε. Ζάχος
	Μέλη:	Τ. Σελλής
		Γ. Κολέτσος

Εγκρίθηκε από την επταμελή εξεταστική επιτροπή την ...

Ε. Ζάχος   Τ. Σελλής   Γ. Κολέτσος

Α. Δελής   Α. Παγουρτζής   Φ. Αφράτη   Ν. Παπασπύρου

Αθήνα, 2007

Παναγιώτης Χείλαρης

στην οικογένειά μου

Περίληψη. Μελετάμε διάφορους χρωματισμούς κορυφών για γράφους και υπεργράφους. Επικεντρωνόμαστε στους χρωματισμούς χωρίς συγκρούσεις. Οι χρωματισμοί χωρίς συγκρούσεις έχουν εφαρμογές στην αποδοτική ανάθεση συχνοτήτων σε κυψελωτά δίκτυα. Δείχνουμε πώς σχετίζονται οι χρωματισμοί χωρίς συγκρούσεις ως προς μονοπάτια γράφων με άλλα προβλήματα χρωματισμού γράφων, όπως: διατεταγμένοι χρωματισμοί (ή αλλιώς κατάταξη κορυφών), χρωματισμοί χωρίς τετράγωνα (συμβολοσειρές), και κλασικοί χρωματισμοί γράφων. Αποδεικνύουμε ιδιότητες των παραπάνω χρωματισμών καθώς και άνω και κάτω φράγματα για τα χρώματα που απαιτούνται για διάφορες κλάσεις γράφων (αλυσίδες, δακτύλιοι, δένδρα, πλέγματα).

Αναλύουμε διάφορες εκδοχές του χρωματισμού χωρίς συγκρούσεις για γράφους αλυσίδες και γράφους δακτυλίους ως προς μονοπάτια. Στην περίπτωση των αλυσίδων το πρόβλημα είναι επίσης γνωστό ως χρωματισμός χωρίς συγκρούσεις ως προς διαστήματα. Για μία άμεση (online) εκδοχή του προβλήματος όπου οι κορυφές του γράφου εμφανίζονται μία μία και ο αλγόριθμος πρέπει να δεσμευτεί στο χρώμα της κορυφής προτού δει τις θέσεις των μελλοντικών κορυφών, αναλύουμε τον άπληστο αλγόριθμο πρώτου ταιριάσματος και τον άπληστο αλγόριθμο μοναδικού μεγίστου. Αποδεικνύουμε ακριβώς πόσα χρώματα χρησιμοποιεί ο άπληστος αλγόριθμος πρώτου ταιριάσματος στην χειρότερη περίπτωση και αναλύουμε ειδικές εισόδους για τον άπληστο αλγόριθμο μοναδικού μεγίστου. Επίσης, συσχετίζουμε χρωματισμούς μοναδικού μεγίστου όπου το χρώμα «1» εμφανίζεται i φορές με μεταθέσεις με $i-1$ κοιλάδες. Ένα ουσιώδες μέρος της ανάλυσής μας βασίζεται στον αριθμό των χρωμάτων που εμφανίζονται μόνον μία φορά (στον εκάστοτε χρωματισμό). Τέλος, ορίζουμε δύο νέα μοντέλα άμεσων αλγορίθμων χρωματισμών χωρίς συγκρούσεις ως προς διαστήματα: ένα αιτιοκρατικό στο οποίο ο αλγόριθμός γνωρίζει τις απόλυτες τελικές θέσεις των προς χρωματισμό κορυφών, και ένα πιθανοκρατικό στο οποίο ο αντίπαλος του αλγορίθμου πρέπει να δεσμευτεί στην είσοδο του πριν ο αλγόριθμος αρχίσει να χρωματίζει. Για αυτά τα δύο μοντέλα δίνουμε αλγορίθμους που χρησιμοποιούν λογαριθμικό αριθμό χρωμάτων.

Abstract. We study several vertex coloring problems for graphs and hypergraphs. We concentrate on conflict-free coloring. The conflict-free coloring problem has applications in efficient frequency assignment in cellular networks. We show the relation of conflict-free coloring paths of graphs with other graph coloring problems, like ordered coloring (also known as vertex ranking), squarefree coloring, and traditional graph coloring. We prove properties of the above colorings and upper and lower bounds for the colors needed for several graph classes (chains, rings, trees, grids).

We analyze several versions of conflict-free coloring chain and ring graphs with respect to paths. For chain graphs, the problem is also known as conflict-free coloring with respect to intervals. For an online version of the problem where vertices of the graph appear one by one, and an algorithm has to commit on the color of a vertex before seeing the positions of the future vertices, we analyze the first-fit greedy and the unique maximum greedy algorithms. We prove tights bounds for the colors used by the first-fit greedy algorithm and we analyze special input sequences for the unique maximum greedy algorithm. We also relate unique maximum colorings with i occurrences of color '1' with permutations with $i-1$ valleys. An essential part of the analysis is based on counting the number of uniquely occurring colors. Finally, we define two new online algorithm models for conflict-free coloring with respect to intervals: one is deterministic and the algorithm has knowledge of the absolute final positions of the vertices to be colored, and the other is probabilistic and the adversary has to commit to the input before the algorithm starts coloring. For these two models, we provide algorithms that use a logarithmic number of colors.

Πρόλογος

Στην παρούσα εργασία μελετάμε διάφορες παραλλαγές του προβλήματος χρωματισμού γράφων και υπεργράφων. Επικεντρωνόμαστε στους χρωματισμούς χωρίς συγκρούσεις (conflict-free coloring).

Στο κεφάλαιο 1, ορίζουμε το πρόβλημα χρωματισμού χωρίς συγκρούσεις σε υπεργράφους, άλλα σχετικά προβλήματα, και αναφέρουμε τις εφαρμογές τους (π.χ., ανάθεση συχνοτήτων σε κυψελωτά δίκτυα, αποδοτική παράλληλη επίλυση αραιών γραμμικών συστημάτων). Επίσης, κάνουμε μία επισκόπηση της υπάρχουσας βιβλιογραφίας.

Στο κεφάλαιο 2, ορίζουμε το πρόβλημα χρωματισμού γράφου χωρίς συγκρούσεις ως προς μονοπάτια και δείχνουμε πώς μπορεί να οριστεί με την βοήθεια του χρωματισμού χωρίς συγκρούσεις σε υπεργράφους. Αποδεικνύουμε κάποιες ιδιότητες αυτού του χρωματισμού. Επίσης, συσχετίζουμε τον χρωματισμό χωρίς συγκρούσεις ως προς μονοπάτια με τον διατεταγμένο χρωματισμό (ordered coloring, γνωστός και ως vertex ranking). Παρουσιάζουμε το πρόβλημα χρωματισμού χωρίς συγκρούσεις ως προς διαστήματα και δείχνουμε την ισοδυναμία του με τον χρωματισμό χωρίς συγκρούσεις ως προς μονοπάτια για γράφους αλυσίδες. Βελτιώνουμε τα γνωστά από την βιβλιογραφία άνω και κάτω φράγματα για τον διατεταγμένο χρωματικό αριθμό του $m \times m$ γράφου πλέγματος, κάτι που έχει συνέπειες και για τον χρωματικό αριθμό χωρίς συγκρούσεις ως προς μονοπάτια. Θεωρούμε χρωματισμούς γράφων πλεγμάτων ως διδιάστατους πίνακες φυσικών αριθμών και χαλαρώνοντας τους περιορισμούς που προκύπτουν από τον χρωματισμό χωρίς συγκρούσεις ως προς μονοπάτια, βρίσκουμε χρωματισμούς που χρησιμοποιούν λογαριθμικό αριθμό χρωμάτων για μία ευρεία κατηγορία περιορισμών.

Στο κεφάλαιο 3, θεωρούμε μία δυναμική εκδοχή του προβλήματος, στην οποία έχει νόημα να θεωρήσουμε ότι η είσοδος εμφανίζεται στον αλγόριθμο με άμεσο (online) τρόπο, δηλαδή οι προς χρωματισμό κορυφές εμφανίζονται μία μία, ο αλγόριθμος πρέπει να δεσμευτεί στο χρώμα της κάθε εμφανιζόμενης κορυφής, χωρίς γνώση των μελλοντικών κορυφών, και δεν μπορεί να αλλάξει το χρώμα παλαιών κορυφών. Επικεντρωνόμαστε στον χρωματισμό χωρίς συγκρούσεις ως προς διαστήματα. Αναλύουμε ακριβώς την συμπεριφορά χειρότερης περίπτωσης του άπληστου αλγορίθμου πρώτου ταιριάσματος (δηλαδή ταιριάζουμε άνω και κάτω φράγματα), λύνοντας ένα ανοιχτό πρόβλημα στην βιβλιογραφία. Δίνουμε διάφορα αποτελέσματα για τον άπληστο αλγόριθμο μοναδικού μεγίστου και μελετάμε την συμπεριφορά του σε διάφορες κλάσεις εισόδων. Μεταξύ άλλων, αποδεικνύουμε μία αντιστοιχία ανάμεσα στους χρωματισμούς μοναδικού μεγίστου όπου το χρώμα «1» εμφανίζεται i φορές με μεταθέσεις με $i-1$ κοιλάδες. Μελετάμε το πλήθος των χρωμάτων που εμφανίζονται ακριβώς μία φορά σε έναν χρωματισμό που προκύπτει από τον άπληστο αλγόριθμο πρώτου ταιριάσματος ή τον άπληστο αλγόριθμο μοναδικού μεγίστου. Για τον οικονομικότερο σε πλήθος χρωμάτων αιτιοκρατικό αλγόριθμο της βιβλιογραφίας δίνουμε μία είσοδο που τον αναγκάζει να χρησιμοποιήσει τα περισσότερα δυνατά χρώματα: $\Omega(\log^2 n)$. Ορίζουμε το πρόβλημα χρωματισμού χωρίς συγκρούσεις ως προς κυκλικά τόξα· αποδεικνύουμε ακριβώς πόσα χρώματα χρειάζονται για την στατική εκδοχή του προβλήματος και μελετάμε πώς συμπεριφέρονται οι άπληστοι αλγόριθμοι στην άμεση εκδοχή του προβλήματος.

Στο κεφάλαιο 4, παρουσιάζουμε έναν άμεσο αιτιοκρατικό αλγόριθμο για ένα μοντέλο απολύτων θέσεων των εμφανιζόμενων σημείων, ο οποίος χρησιμοποιεί το πολύ $2\lg n$ χρώματα. Επίσης, παρουσιάζουμε έναν πιθανοκρατικό αλγόριθμο που χρωματίζει εναντίον ενός αντιπάλου που πρέπει να δεσμευτεί στην είσοδο του προτού αποκαλύψει την πρώτη κορυφή και χρησιμοποιεί το πολύ $1 + \log_{3/2} n$ χρώματα.

Στον επίλογο, διατυπώνουμε κάποια συμπεράσματα και παρουσιάζουμε τα σπουδαιότερα, κατά την κρίση μας, ανοιχτά προβλήματα στην περιοχή των χρωματισμών χωρίς συγκρούσεις.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντά μου Στάθη Ζάχο. Έχει εύρος και βάθος γνώσης στην περιοχή της Πληροφορικής, αλλά και σε πολλά άλλα γνωστικά πεδία. Η αφοσίωση του στην διδασκαλία και στην διάδοση αυτής της γνώσης είναι υποδειγματική. Οι κατευθύνσεις που μου έδωσε για την έρευνά μου και η συνεργασία μας έχουν διαμορφώσει το περιεχόμενο της παρούσας εργασίας. Θα ήθελα επίσης να ευχαριστήσω τον Amotz Bar-Noy, ο οποίος μου ανέφερε το πρόβλημα χρωματισμών χωρίς συγκρούσεις για διαστήματα. Η συνεργασία μαζί του υπήρξε χρήσιμη εμπειρία για μένα και ήταν πολύ γενναιόδωρος μαζί μου. Ένας άλλος που θα ήθελα να ευχαριστήσω είναι ο Shakhar Smorodinsky, ο οποίος με τίμησε με την συνεργασία του. Θα ήθελα επίσης να ευχαριστήσω τον Géza Tóth για μερικές πολύ ωραίες ιδέες στα προβλήματα που μελέτησα στην παρούσα εργασία. Θα ήθελα να ευχαριστήσω τον Ernst Specker (είναι παππούς μου κατά την ακαδημαϊκή έννοια) για την συνεισφορά του σε ωραίες ιδέες και νεολογισμούς, σχετικά με προβλήματα που μελετήθηκαν στην παρούσα εργασία. Θα ήθελα επίσης να ευχαριστήσω τον Μιχάλη Λαμπή για την βοήθειά του και για την συνεισφορά του σε μερικά προβλήματα σε αυτήν την εργασία. Έχω ωφεληθεί από συζητήσεις πάνω στα προβλήματα της εργασίας με τον János Pach και τον David Peleg και επομένως θα ήθελα να τους ευχαριστήσω. Θα ήθελα να ευχαριστήσω τα άλλα δύο μέλη της τριμελούς επιτροπής μου, Τίμο Σελλή και Γιώργο Κολέτσο, για την στήριξη τους και επειδή ήταν διαθέσιμοι όποτε τους χρειάστηκα. Θα ήθελα ιδιαιτέρως να ευχαριστήσω τον Αλέξη Δελή (στην επταμελή επιτροπή μου), ο οποίος μου έδωσε εξαιρετικές συμβουλές για την δέουσα συμπεριφορά στο πεδίο της Πληροφορικής, αλλά και για την ζωή γενικότερα. Η επταμελής μου επιτροπή αποτελείται επίσης από τους Φώτω Αφράτη, Άρη Παγουρτζή, και Νίκο Παπασπύρου, οι οποίοι με στήριξαν σημαντικά και είχα χρήσιμες συζητήσεις μαζί τους. Επίσης, θα ήθελα να ευχαριστήσω τον Γιώργο Κόλλιο, ο οποίος με συμβούλεψε και με υποστήριξε κατά το τελικό στάδιο της συγγραφής της διατριβής. Θα ήθελα να ευχαριστήσω τον Πέτρο Ποτίκα για την βοήθεια και την στήριξη του, ειδικά κατά τα τελευταία στάδια της συγγραφής της παρούσας εργασίας· μεταξύ άλλων διάβασε προσχέδιες εκδόσεις και προσέφερε πολλές διορθώσεις. Τέλος, θα ήθελα να ευχαριστήσω όλα τα μέλη του Co.Re.Lab (Computation and Reasoning Lab of the National Technical University of Athens), ειδικά την Κατερίνα Ποτίκα, τον Βαγγέλη Μπαμπά, και την Γεωργία Καούρη για πολλές χρήσιμες συζητήσεις και επειδή με βοήθησαν να πολεμήσω την γραφειοκρατία. Ειδικά για το τελευταίο, θα ήθελα να ευχαριστήσω ιδιαιτέρως την Ελένη Κασσεσιάν.

Algorithms and Complexity: Graph and hypergraph colorings

Panos Hilaris

Contents

Contents **15**

List of Figures **19**

List of Tables **21**

Prologue **23**
Acknowledgements 24

1 Introduction **25**
1.1 Conflict-free coloring 25
1.2 Ordered coloring 28

2 Conflict-free coloring paths of a graph **31**
2.1 Introduction 31
2.2 Relation with other problems 32
2.2.1 Ordered coloring 32
2.2.2 Squarefree colorings 34
2.3 Conflict-free coloring graphs 37
2.3.1 Chain 37
2.3.2 Ring 37
2.3.3 Tree 37
2.3.4 Grid 38
2.4 Conflict-free coloring games 38
2.5 Conflict-free coloring for intervals 42
2.5.1 A lower bound for the number of colors needed 43
2.5.2 A conflict-free coloring that achieves the lower bound . 43
2.6 Improving the bounds for grids 46
2.7 Upper bound for grids 46
2.8 Special subgraphs in the grid 46
2.8.1 The rhombus 46

2.8.2 The triangle 47
2.8.3 A $8m/3$ upper bound 48
2.8.4 A $18m/7$ upper bound 49
2.9 Lower bound for grids 49
2.10 Neochromatica 50
2.11 Arrays and meander paths 50
2.12 Arrays, subarrays, and thin subarrays 51
2.12.1 Conflict-free coloring with respect to subarrays 51
2.12.2 Conflict-free coloring with respect to thin subarrays . . 51
2.13 Multidimensional arrays 52
2.13.1 Conflict-free coloring with respect to subarrays 52
2.13.2 Conflict-free coloring with respect to thin subarrays . . 52

3 Greedy algorithms for online conflict-free coloring 53
3.1 Points appearing dynamically: more constraints 53
3.2 Insertion sequences and permutations 53
3.3 Separation of static and dynamic versions of the problem . . . 54
3.4 First-fit greedy algorithm 55
3.5 Some observations related to the greedy algorithm 56
3.6 A sequence of bad insertion sequences 58
3.7 A simpler sequence of bad insertion sequences 60
3.7.1 An upper bound on the use of colors 62
3.8 Description of unique max algorithm 63
3.9 Running UM for small insertion sequences 63
3.10 Signed insertion sequences 63
3.11 A lower bound on the use of colors by UM 65
3.12 All-seeing colorings 66
3.12.1 Smallest all-seeing colorings 66
3.13 Colorings with a specific number of occurrences of a color . . . 69
3.13.1 A connection of $N_1(n, i)$ with permutations with $i - 1$ valleys 69
3.14 The decrement operation on colorings 70
3.15 Insertions to the left or right of a point 70
3.15.1 Use of colors by UM colorings with one '1' 71
3.15.2 lr-sequences and the greedy algorithm 71
3.16 Relative insertion sequences 75
3.16.1 1-local relative insertions 75
3.17 Color use of different sequences in UM 75
3.18 Color use of different lr-sequences in UM 76
3.19 Analysis for first-fit greedy and UM 84
3.20 Statistics for uniquely occurring colors 85

3.21 The leveled algorithm 87
3.22 A bad insertion sequence for the leveled algorithm 87
3.23 Use of colors by the leveled algorithm for small lengths 88
3.24 Unique occurrences of colors 91
3.25 The static case for rings 91
3.26 The greedy algorithm for ring coloring 92
3.26.1 Use of colors by greedy for small lengths 92
3.26.2 Unique occurrences of colors 94
3.27 The UM algorithm for ring coloring 94
3.27.1 Color use in UM 94

4 Logarithmic online algorithms **97**
4.1 Description of the $2\lg n$ algorithm 97
4.2 Correctness of the $2\lg n$ algorithm 100
4.3 Slight improvements over the $2\lg n$ algorithm 101
4.4 Description of the oblivious adversary algorithm 102
4.5 Analysis of the oblivious adversary algorithm 103

Bibliography **107**

List of Figures

1.1 Points on a line and intervals 27

2.1 The connected-subset-CF-game 39
2.2 The path-CF-game 39
2.3 An optimal static conflict-free coloring algorithm with respect to intervals 44
2.4 A tree for $n = 15$ and its inorder traversal 44
2.5 A tree for $n = 8$ and its inorder traversal 45
2.6 The rhombus 47
2.7 The rhombus separation 47
2.8 The wide triangle 47
2.9 The wide triangle separation 48
2.10 An $8m/3$ upper bound 48
2.11 An $18m/7$ upper bound 49
2.12 A 3×3 grid 50
2.13 A conflict-free coloring of the 3×3 grid with respect to paths 50
2.14 A conflict-free coloring of a 3×3 array with respect to meander paths 51

3.1 The bad sequence coloring by leveled ($n = 15$) 88
3.2 The bad sequence coloring by FF ($n = 15$) 88

4.1 The $2 \lg n$ algorithm 99
4.2 The recursion in a coloring 100

List of Tables

3.1 Best and worst use of colors for greedy 57
3.2 First sequence using most colors for its length for greedy . . . 57
3.3 Sequences using numbers of colors for greedy 57
3.4 Sequences ratio using numbers of colors for greedy 64
3.5 Best and worst use of colors for UM 64
3.6 First sequence using most colors for its length for UM 64
3.7 Sequences using numbers of colors for UM 65
3.8 Sequences ratio using numbers of colors for UM 65
3.9 Length of shortest k-beas colorings 67
3.10 Length of shortest k-leas colorings 67
3.11 First lr-sequence using most colors for its length (UM) 72
3.12 Coloring of lr-sequence using most colors for its length (UM) . 73
3.13 Number of lr-sequences that use k colors (UM) 74
3.14 Ratio of lr-sequences using k colors (UM) up to 16 74
3.15 Ratio of lr-sequences using k colors (UM) over 16 78
3.16 First 1-local sequence using most colors for its length (UM) . . 79
3.17 Coloring of first 1-local sequence using most colors for its length (UM) 80
3.18 First 1-local sequence using most colors for its length (greedy) 81
3.19 Coloring of first 1-local sequence using most colors for its length (greedy) 82
3.20 Minimum length sequence that uses k colors (UM) 83
3.21 Minimum length lr-sequence that uses k colors (UM) 83
3.22 Number of unique occurrences for greedy algorithm 86
3.23 Number of unique occurrences for UM algorithm 86
3.24 Best and worst use of colors for level 89
3.25 First sequence using most colors for its length for level 89
3.26 Sequences using numbers of colors for level 90
3.27 Sequences ratio using numbers of colors for level 90
3.28 Uniquely occurring colors for level 91
3.29 Uniquely occurring colors for level ratio 91

3.30 Best and worst use of colors for first-fit greedy 93
3.31 First sequence using most colors for its length for greedy . . . 93
3.32 Sequences using numbers of colors for first-fit greedy 93
3.33 Sequences ratio using numbers of colors for first-fit greedy . . 95
3.34 Uniquely occurring colors for greedy 95
3.35 Uniquely occurring colors for greedy ratio 95
3.36 Best and worst use of colors for UM 96

4.1 Colorings that use less than or equal to 2 colors 100

Prologue

In this work, aspects of conflict-free coloring are studied.

In chapter 1, we state the conflict-free coloring problem, some related problems, and we mention applications. We also review the literature.

In chapter 2, we state the more general version of the conflict-free coloring problem, as a coloring of graphs, and hypergraphs, and we prove some basic properties of such colorings. We also show the connections with other kinds of colorings of graphs and hypergraphs. We describe the problem of conflict-free coloring for intervals and consider the static case. We give better bounds for the ordered chromatic number of the $m \times m$ grid. We look at coloring of grids as arrays of numbers and by weakening the constraints related to conflict-free coloring with respect to paths, we get logarithmic colorings for a wide range of constraints.

In chapter 3, we consider a dynamic version of the problem, where it is relevant to consider the input revealed in an online way. We analyze the behavior of a fully greedy first-fit algorithm, that performs very bad in the worst case. We give some results related to the behavior of the unique max greedy algorithm (from [27]). We consider the number of uniquely occurring colors in full greedy and unique max algorithms. We make some remarks about the leveled algorithm (presented in [27]). We introduce the conflict-free problem for intervals in rings, and see how greedy and unique max algorithms perform in that case.

In chapter 4, we present a logarithmic online deterministic algorithm in a model where the absolute positions of points are given (about $2 \lg n$ colors used, at most). We also present a randomized expected logarithmic algorithm that colors always correctly against an oblivious adversary that has to commit on a permutation before revealing the first point (uses $O(\log n)$ colors).

Finally, in the epilogue, we give some concluding remarks, and collect the most important problems left open on conflict-free coloring.

Acknowledgements

I would like to thank my advisor Stathis Zachos. He has breadth and depth of knowledge in Computer Science, but also in many other fields. His dedication to teaching and disseminating this knowledge is exemplary. The directions he has given me for research and our collaboration have shaped the contents of this thesis. I would also like to thank Amotz Bar-Noy for originally introducing me to the conflict-free coloring problem for intervals. Working with him has been a very beneficial experience for me and he has been very generous to me. Another coauthor I would like to thank is Shakhar Smorodinsky, who has honored me with his collaboration. I would like to also thank Géza Tóth for very nice ideas on the problems studied in this work. I would like to thank Ernst Specker (he is a grandfather to me in the academic sense) for his contribution of nice ideas and neologisms, related to the problems studied in this work. I would also like to thank Michael Lampis for his help and for his contributions in some of the problems of this thesis. I have been benefited by discussions on the problems with János Pach and David Peleg, so I would also like to thank them. I would also like to thank the other members of my three-member committee, Timos Sellis and Giorgos Koletsos, for their support and the fact that they were there whenever I needed them. I would especially like to thank Alex Delis (in my seven-member committee), who has been a constant source of great advice for matters related to conduct in the Computer Science field, and for life in general. My seven-member committee is also comprised of Foto Afrati, Aris Pagourtzis, and Nikos Papaspyrou, who have offered great support and I had many helpful discussions with them. I would also like to thank George Kollios, who gave advice and supported me during the last stage of my work. I would like to thank Petros Potikas for his help and support during the last stage of writing this thesis. Among other things, he read draft versions and he has offered many corrections. Last, but not least, I would like to thank all the people at Co.Re.Lab (Computation and Reasoning Lab of the National Technical University of Athens), especially Katerina Potika, Evangelos Bampas, and Georgia Kaouri, for many helpful discussions and for helping me fighting the bureaucracy. Especially for fighting the bureucracy, I would like to thank Eleni Kassesian.

Chapter 1

Introduction

In this chapter we introduce some coloring problems in graphs and hypergraphs that can model frequency assignment problems in cellular networks. We also mention some related coloring problems and their applications.

1.1 Conflict-free coloring

A *vertex coloring* of a graph $G = (V, E)$ is a function $C\colon V \to \mathbb{N}^+$ such that for every edge $\{v_1, v_2\} \in E$: $C(v_1) \neq C(v_2)$. A *hypergraph* $G = (V, E)$ is a generalization of a graph for which *hyperedges* can be arbitrary-sized non-empty subsets of V. There are several ways to define vertex coloring in hypergraphs: On one extreme, it is required that for every hyperedge, not all colors are the same (there are at least two colors); on the other extreme, it is required that for every edge, no color is repeated (all the colors are different). In between these two extremes, there is another possible generalization: A vertex coloring C of hypergraph G is called *conflict-free* if in every hyperedge e there is a vertex whose color is unique among all other colors in the hyperedge. Formally:

$$\forall e \in E\colon \exists v \in e\colon \forall v' \in e\colon v' \neq v \to C(v') \neq C(v).$$

Conflict-free coloring models frequency assignment for cellular networks. A cellular network consists of two kinds of nodes: *base stations* and *mobile agents*. Base stations have fixed positions and provide the backbone of the network; they are modeled by vertices in V. Mobile agents are the clients of the network and they are served by base stations. This is done as follows: Every base station has a fixed frequency; this is modeled by the coloring C, i.e., colors represent frequencies. If an agent wants to establish a link with a base station it has to tune itself to this base station's frequency. Since agents are mobile, they can be in the range of many different base stations. The range of communication of every agent is modeled by a hyperedge $e \in E$, which is the set of base stations that can communicate with the agent. To avoid interference, the system must assign frequencies to base stations in the following way: For any range, there must be a base station in the range with a frequency that is not reused by some other base station in the range. This is modeled by the conflict-free property. One can solve the problem by assigning n different frequencies to the n base stations. However, using many frequencies is expensive, and therefore, a scheme that reuses frequencies, where possible, is preferable.

The study of conflict-free colorings was originated in the work of Even et al. [26] and Smorodinsky [50]. In addition to the practical motivation described above, this new coloring model has drawn much attention of researchers through its own theoretical interest and such colorings have been the focus of several recent papers (see, e.g., [26, 50, 44, 29, 27, 33, 23, 51, 5, 8]). Other references for frequency assignment in cellular and wireless networks include [1]. NP-completeness and hardness of approximation results can be found in [18]. There are also related problems in optical fibre networks [2, 3, 41, 24, 42, 39, 45, 30, 40].

Suppose we are given a set of n base stations, also referred to as *antennas*. Assume, for simplicity, that the area covered by a single antenna is given as a disk in the plane. Namely, the location of each antenna and its radius of transmission is fixed and is given (the transmission radii of the antennas are not necessarily equal). Even et al. [26] showed that one can find an assignment of frequencies to the antennas with a total of at most $O(\log n)$ frequencies such that each antenna is assigned one of the frequencies and the resulting assignment is free of conflicts, in the preceding sense. Furthermore, it was shown that this bound is worst-case optimal. Let $\mathcal{R}$ be a set of regions in the plane. For a point $p \in \cup_{r\in\mathcal{R}} r$, let $r(p) = \{r \in \mathcal{R} \mid p \in r\}$. Let $H(\mathcal{R})$ denote the hypergraph $(\mathcal{R}, \{r(p) \mid p \in \cup_{r\in\mathcal{R}}\})$. We say that $H(\mathcal{R})$ is the hypergraph *induced* by $\mathcal{R}$. Thus, Even et al. [26] showed that any hypergraph induced by a family $\mathcal{R}$ of n discs in the plane admits a CF-coloring with only $O(\log n)$ colors and that this bound is tight in the worst case. Furthermore, such a coloring can be found in deterministic polynomial time[1]. The results of [26] were further extended in [29] by combining more involved probabilistic and geometric ideas. The main result of [29] is a general randomized algorithm which CF-colors any set of n "simple" regions (not necessarily convex)

[1]In [26] it is shown that finding the minimum number of colors needed to CF-color a given collection of discs is NP-hard even when all discs are congruent, and an $O(\log n)$ approximation-ratio algorithm is provided.

whose union has “low” complexity, using a “small” number of colors.

For conflict-free coloring of n points with respect to (closed) disks, in [44], Pach and Tóth prove a lower bound of $\Omega(\log n)$ colors. They also generalize the result to homothetic copies of any convex body. Har-Peled and Smorodinsky in [29], for conflict-free coloring of n points with respect to axis-parallel rectangles, give a coloring using $O(\sqrt{n})$ colors. In [44], this is improved to $O(\sqrt{n \log\log n / \log n})$. In [5], Alon and Smorodinsky consider coloring a collection of n disks in which each disk intersects at most k others such that for each point p in the union of all disks there is at least one disk in the collection containing p whose color differs from that of all other member of the collection that contain p (this is the dual problem of coloring points with respect to ranges). The proof uses the probabilistic method [6], and especially the Lovász Local Lemma.

In [51], Smorodinsky studies ‘traditional’ coloring of hypergraphs[2] that are induced by simple Jordan curves. He applies the above results to conflict-free coloring of regions with near linear union complexity (using a polylogarithmic number of colors), and axis-parallel rectangles (using $O(\log^2 n)$ colors).

In [23], Elbassioni and Mustafa consider an interesting variation of the problem of conflict-free coloring points with respect to axis-parallel rectangles: They prove that given any set of n points on the plane, one can add $O(n^{1-\varepsilon})$ new points, so that all points can be conflict-free colored with $\tilde{O}(n^{\frac{3}{8}(1+\varepsilon)})$ colors.

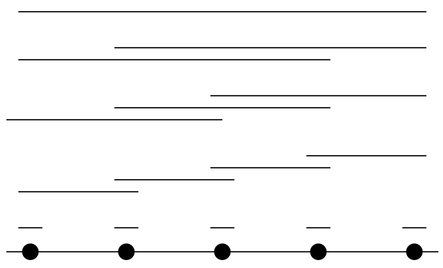


Figure 1.1: Points on a line and intervals

The paper [27] considered the special case of the problem where the hypergraph is defined as follows: Vertices are identified by points that lie on a line and E consists of all subsets of V defined by intervals intersecting at least one vertex. A line with n points has $n(n+1)/2$ such subsets (for every $i \in \{1, \ldots, n\}$, there are $n - i + 1$ different subsets containing i points). For

[2]This is one of the vertex colorings that generalize graph vertex coloring, mentioned in the start of this thesis: Every hyperedge is not monochromatic.

$n = 5$, these subsets are shown in figure 1.1. We call these subsets intervals because for our problem, two intervals are equivalent if they contain the same vertices. We represent colorings by listing the colors of points from left to right in a *string*. For example, for the points in figure 1.1 ($n = 5$), 12312 is a conflict-free coloring, whereas 12123 is not.

Conflict-free coloring for intervals is important because it can model assignment of frequencies in networks where the agents' movement is approximately unidimensional, e.g., the cellular network that covers a single long road and has to serve agents that move along this road. Also, conflict-free coloring for intervals plays a role in conflict-free coloring for more complicated range spaces (see [26]).

The problem becomes more interesting when the vertices are given online by an adversary. Namely, at every given time step $t \in \{1, \ldots, n\}$, a new vertex $v_t \in V$ is given and the algorithm must assign v_t a color such that the coloring is a conflict-free coloring of the hypergraph that is induced by the vertices $V_t = \{v_1, \ldots, v_t\}$. Once v_t is assigned a color, that color cannot be changed in the future. This is an online setting, so the algorithm has no knowledge of how vertices will be given in the future. For this version of the problem, in the case of intervals, Fiat et al. [27] provide several algorithms. Their randomized algorithm uses $O(\log n \log\log n)$ colors with high probability. Their deterministic algorithm uses $O(\log^2 n)$ colors in the worst case. That algorithm requires $\Omega(log^2 n)$ colors on some inputs.

1.2 Ordered coloring

We define a restriction of a conflict-free coloring for hypergraphs. A vertex coloring C of hypergraph G is called *ordered* if in every hyperedge e there is a vertex whose color is unique *and maximum* among all other colors in the hyperedge. Formally:

$$\forall e \in E \colon \exists v \in e \colon \forall v' \in e \colon v' \neq v \rightarrow C(v') < C(v).$$

If the hyperedges are induced by all (simple) paths of a graph, then we have the notion of an *ordered coloring* or a *vertex ranking* of a graph. See the next chapter for more details. Vertex ranking problems in the simple graph setting have many applications. One application is in the filed of VLSI design [35, 49]. Another application is in the field of parallel Cholesky factorization of matrices [22, 36, 28, 37]. In general graphs, finding the exact ordered chromatic number of a graph is NP-complete [38]. Approximability results are given in [12]. The vertex ranking problem is also interesting for the Operations Research community, because it also has applications in planning efficient assembly of products in manufacturing systems [31]. In general, it seems the vertex ranking problem can model situations where interrelated tasks have to be accomplished fast in parallel, with some constraints (assembly from parts, parallel query optimization in databases, etc.)

Chapter 2

Conflict-free coloring paths of a graph

2.1 Introduction

Given is a graph G, with vertex set $V(G)$ and edge set $E(G)$. The aim is to color the vertices of the graph such that for each path p in the graph, there is a vertex v in p whose color is not used by any other vertex in p. This is called a *conflict-free* coloring (CF coloring) of graph G with respect to paths. It is a minimization problem, i.e., the goal is to find such a coloring that uses as few colors as possible. Formally:

Definition 1. A k-CF-coloring is a function $C\colon V(G) \to \{1, \dots, k\}$ such that:

$$\forall \text{path } p \in G\colon \exists v \in p\colon \forall v' \in p\colon v' \neq v \to C(v') \neq C(v).$$

The *conflict-free chromatic number* of a graph G, denoted by $\chi_{\text{cf}}(G)$, is the minimum k for which G has a k-CF-coloring.

Since the above coloring involves sets of vertices included in a path, one can ask the same question in terms of hypergraphs. Details follow.

A *hypergraph* $H = (V, E)$ is a generalization of a graph for which *hyperedges* can be arbitrary-sized non-empty subsets of V.

Definition 2. A vertex coloring C of hypergraph $H = (V, E)$ is called *conflict-free* if in every hyperedge e there is a vertex whose color is unique among all other colors in the hyperedge. Formally, $\forall e \in E\colon \exists v \in e\colon \forall v' \in e\colon v' \neq v \to C(v') \neq C(v)$.

Proposition 3. *Given a graph G with vertex set V, define the following hypergraph:*

$$H_G = (V, \{\mathit{vert}(p) \mid p \in \mathit{paths}(G)\}),$$

where vert(p) is the set of vertices of path p. Then, a conflict-free *coloring of graph G with respect to paths is a conflict-free coloring of H_G and vice versa.*

2.2 Relation with other problems

2.2.1 Ordered coloring

A closely related problem is *ordered coloring* [34] or *vertex ranking* [31]. Ordered coloring is like conflict-free coloring, but we have the following additional constraint: the unique color in a path must also be the maximum color in the path. Formally, we define:

Definition 4. A *unique maximum* CF coloring is a CF coloring in which the maximum color in every path p is also a unique color in path p.

We remark that the definition given above is not what is typical in the bibliography [34]. Instead the following definition is more typical:

Definition 5. An *ordered k*-coloring of a graph G is a function $C : V(G) \to \{1, \dots, k\}$ such that for every pair of distinct vertices v, v', and every path p from v to v', if $C(v) = C(v')$, there is an internal vertex v'' of p such that $C(v) < C(v'')$. The *ordered chromatic number* of a graph G, denoted by $\chi_o(G)$, is the minimum k for which G has an ordered k-coloring.

We prove that the two definitions are equivalent:

Proposition 6. *C is a unique maximum CF coloring if and only if C is an ordered coloring.*

Proof. If C is a unique maximum CF-coloring, then for any two same color vertices v, v', every (v, v')-path p has a unique maximum color, greater than $C(v)$, which appears in some internal vertex of p.

If C is an ordered coloring, then consider any path p in G. The maximum color in p has to occur exactly in one vertex. If it occurs in two vertices v, v' of p then there is a (v, v')-path contained in p which has an internal vertex with a greater color; a contradiction to the maximality of $C(v)$ in p. □

Every ordered coloring is also a CF-coloring and thus $\chi_{cf}(G) \leq \chi_o(G)$.

In ordered colorings, an even stronger property is true:

Proposition 7. *In any ordered coloring C of G, in every connected subset S of vertices of G, the maximum color appearing in S, i.e.,* $\max\{C(v) \mid v \in S\}$, *is unique in S.*

Proof. By contradiction; if there are two different vertices x, y in S with the maximum color, then there is a (x, y)-path in S, for which there is no internal vertex with higher color. $\square$

Now, we define yet another coloring: A k-connected-set-CF-coloring is a function $C\colon V(G) \to \{1, \ldots, k\}$ such that for every connected subset of vertices S, there is a vertex $v \in S$, with a unique color in S. This is similar to CF-coloring, but the uniqueness property is taken with respect to all connected sets, instead of only paths. The *connected set conflict-free chromatic number* of a graph G, denoted by $\chi_{\text{cf}}^{\text{cs}}(G)$, is the minimum k for which G has a k-connected-set-CF-coloring.

A CS-CF-coloring of G is not necessarily an ordered coloring, although an ordered coloring is CS-CF. However, we can prove that $\chi_{\text{o}}(G) = \chi_{\text{cf}}^{\text{cs}}(G)$.

Lemma 8. *In a CS-CF-coloring of a connected graph G, let U denote the vertices with unique colors in all G. Then $G - U$ is disconnected or empty.*

Proof. By contradiction. If $G - U$ is connected, then $V - U$ is a connected subset of vertices of G which has no unique color. $\square$

Proposition 9. *For every graph G, $\chi_{\text{cf}}^{\text{cs}}(G) \geq \chi_{\text{o}}(G)$.*

Proof. By induction on $\chi_{\text{cf}}^{\text{cs}}(G)$. For the base, if $\chi_{\text{cf}}^{\text{cs}}(G) = 1$, then $\chi_{\text{o}}(G) = 1$ (these are exactly the graphs with single vertex connected components). For the inductive step, given an optimal CS-CF-coloring of G, we are going to convert it to a $\chi_{\text{cf}}^{\text{cs}}(G)$-ordered coloring of G. For every connected component of G we have $\chi_{\text{cf}}^{\text{cs}}(G') \leq \chi_{\text{cf}}^{\text{cs}}(G)$, so we only need to take care of connected components for which $\chi_{\text{cf}}^{\text{cs}}(G') = \chi_{\text{cf}}^{\text{cs}}(G)$ (the others are taken care of by the inductive hypothesis): In such a connected component G', consider the set U of unique colors in $V(G')$. Rename the colors of U, so that they are the maximum colors of U. After the renaming of colors, every path that intersects with U has already the property of the unique maximum color. If $G - U$ is empty, $U = V(G')$ and the coloring of G' is already ordered (all colors are unique). Otherwise, by lemma 8, $G' - U$ is disconnected. Each connected component of $G' - U$ is CF-CS colored with at most $\chi_{\text{cf}}^{\text{cs}}(G) - |U|$ colors, so by the inductive hypothesis each connected component of $G' - U$ can be colored with an ordered coloring using no more than $\chi_{\text{cf}}^{\text{cs}}(G) - |U|$ color. Thus, we have an ordered coloring of G' with at most $\chi_{\text{cf}}^{\text{cs}}(G)$ colors. $\square$

The ordered chromatic number is monotone with respect to minors. A graph X is a *minor* of Y, denoted as $X \preccurlyeq Y$, if there is a subgraph G of Y, and a sequence $G_0, \ldots, G_k$, with $G_0 = G$ and $G_k = X$, such that $G_i = G_{i-1}/e_{i-1}$, where $e_{i-1} \in E(G_{i-1})$ (i.e., edge e_{i-1} is *contracted* in G_{i-1}), for $i \in \{1, \ldots, k\}$.

Proposition 10. *If $X \preccurlyeq Y$, then $\chi_{\mathrm{o}}(X) \leq \chi_{\mathrm{o}}(Y)$.*

Proof. If G is a subgraph of Y, and C an ordered coloring of Y, the restriction of C to $V(G)$ is an ordered coloring of G.

Given G and an ordered coloring C of G, then the following is an ordered coloring of G/xy: For v different from x, y, use the same color as in C. For the vertex v_{xy} that arises from the contraction of edge xy, use $\max\{C(x), C(y)\}$. For every path p of G/xy, either $v_{xy} \notin p$, in which case p is also a path in G, and thus it contains a maximum unique color, or $p = p_1 v_{xy} p_2$ (with p_1, p_2 possibly empty paths), in which case path $p_1 x y p_2$ has a maximum unique color, and thus also p has a maximum unique color. □

It is not known whether the conflict-free chromatic number is monotone under minors. Even if it is monotone, a recoloring proof like the above would be a lot more elaborate. However, it is easy to prove the weaker assertion, for subgraphs:

Proposition 11. *If $X \subseteq Y$, then $\chi_{\mathrm{cf}}(X) \leq \chi_{\mathrm{cf}}(Y)$.*

Proof. Graph X contains a subset of the paths of Y, so the restriction of an optimal coloring of $V(Y)$ to $V(X)$ is a CF-coloring for X. □

2.2.2 Squarefree colorings

Another related problem is obtained by looking at colorings of paths as strings. We impose the following restriction: Every coloring of a path, when viewed as a string, shall not contain a repetition. Formally, a string $w \in (\mathbb{N}^+)^*$ is called *squarefree* if there is no substring of w of the form $x^2 = xx$, where x is a nonempty string. Given a coloring C of the vertices of a graph, for every path $p = v_1 \dots v_\ell$, we define the color string of p to be $C(v_1) \dots C(v_\ell)$.

Remark 12. It is not necessary to also consider the reverse path that gives rise to the string $C(v_\ell) \dots C(v_1)$ since a string is squarefree if and only if its reverse is squarefree.

Definition 13. A coloring $C\colon V(G) \to \{1, \dots, k\}$ is a squarefree k-coloring if for every path in the graph its color string is squarefree.

Every CF-coloring is squarefree and thus $\chi_{\mathrm{sf}}(G) \leq \chi_{\mathrm{cf}}(G)$.

We have the following relation between colorings:

$$\begin{tikzcd} C \\ SF \arrow[u] \\ CF \arrow[u] \\ OC \arrow[u] \end{tikzcd}$$

where C is the class of 'traditional' vertex coloring of graphs.

The above is a proper hierarchy as can be exhibited by the following colorings of the chain P_6:

$$\begin{gathered} 121212 \\ 123132 \\ 121312 \\ 313213 \end{gathered}$$

which are 'traditional', squarefree, ordered, and conflict-free, respectively.

In terms of chromatic numbers:

Proposition 14. *$\chi(G) \leq \chi_{\mathrm{sf}}(G) \leq \chi_{\mathrm{cf}}(G) \leq \chi_{\mathrm{o}}(G)$.*

The problem of squarefree coloring looks a lot easier than CF coloring and ordered coloring: For example, a seminal result by Thue shows that 3 colors suffice to color any chain [52]. More precisely:

$$\chi_{\mathrm{sf}}(P_n) = \begin{cases} n, & \text{if } n \leq 2 \\ 3, & \text{otherwise} \end{cases}$$

A coloring of a chain can be seen as a string, over an alphabet of possible colors. The proof of Thue relies on squarefreeness preserving morphisms. A morphism can be described by where it maps each letter of the alphabet. The following is a squarefreeness preserving morphism on the three letter alphabet $\{a, b, c\}$:

$$a \rightarrow abcab, \quad b \rightarrow acabcb, \quad c \rightarrow acbcacb$$

Starting with the word a, and by repetitive applications, the above morphism gives arbitrarily long squarefree words on three letters:

$$abcabacabcbacbcacbabcabacabcb\ldots$$

The above result is in contrast with the coloring defined in the start of this section, because, as we will see, for chains both ordered coloring and conflict-free coloring need $\Omega(\log n)$ colors.

Another, more recent result is by Currie [20], on the squarefree chromatic number of rings:

$$\chi_{\mathrm{sf}}(C_n) = \begin{cases} 4, & \text{if } n \in \{5, 7, 9, 10, 14, 17\} \\ 3, & \text{otherwise} \end{cases}$$

As we will see, for rings both ordered coloring and conflict-free coloring need $\Omega(\log n)$ colors.

The above ring result can be also interpreted as follows: *For every ring there is a* subdivision *of it which is squarefree colorable using 3 colors* (see [13, 21] for the definition of subdivision).

A similar result was proved for trees [15], namely that every tree has a sufficiently large subdivision which can be squarefree colored with 3 colors.

For general graphs the following result is known (see [11]):

Theorem 15. *Every graph has a subdivision which can be squarefree colored with 4 colors.*

It is an open problem whether this 4 can be made a 3. If yes, it would be a striking generalization of Thue's result.

Cubefree colorings

Another related class of colorings consists of *cubefree* colorings, where color strings of paths can not contain a x^3 substring, for x non-empty.

It is known ([52] and implicit in [46]) that 2 colors suffice to color any chain. A cubefreeness preserving morphism, on a two letter alphabet $\{a, b\}$, is quite simple:

$$a \rightarrow ba, \quad b \rightarrow ba$$

A cubefree word starts like:

$$abbabaabbaab \ldots$$

Cubefree colorings can also be put in the above hierarchy over squarefree colorings but they are not comparable with traditional colorings.

Squarefree, cubefree, and related colorings have been studied extensively for strings. A good starting point for the interested reader is the book by Allouche and Shallit [4].

2.3 Conflict-free coloring graphs

We study conflict-free coloring some graphs with (relatively) few edges, and with respect to all paths.

2.3.1 Chain

Conflict-free coloring of a chain is more well known as conflict-free coloring with respect to intervals [27]. Exactly, $1 + \lfloor \lg n \rfloor$ colors are needed: For $n = 2^k - 1$, the coloring C^k is defined recursively as follows: $C^k = 1$ and $C^{k+1} = C^k \circ (k+1) \circ C^k$ (where $\circ$ is the concatenation operator for strings). Coloring C^k is conflict-free, and for n with $n < 2^k - 1$ the prefix of length n of C^k is conflict-free. This had been proved in [26]. This is also an ordered coloring.

2.3.2 Ring

To conflict-free color a *ring*, we use the above conflict-free coloring of a chain. We pick an arbitrary vertex v and color it with a unique color (not to be reused anywhere else in the coloring). The remaining vertices form a chain that we color with the method described above. This method colors a ring of n vertices with $2 + \lfloor \lg(n-1) \rfloor$ colors. For example, if $n = 8$, the coloring is

$$41213121$$

where '4' is the first unique color used for v. It is not difficult to see that the coloring is conflict-free: All paths that include v are conflict-free colored, and the remaining graph $G - v$ is a chain of $n - 1$ nodes, so paths of $G - v$ are also conflict-free colored. For further analysis, and a proof that $2 + \lfloor \lg(n-1) \rfloor$ is tight, check section 3.25.

2.3.3 Tree

For a *tree* graph, we use the idea of a 1/2-separator [32, 25]. A 1/2-separator is a vertex which, when removed, leaves connected components whose size is bounded by $n/2$. The method to color a tree is as follows: Find a 1/2-separator, color it with a unique color. Then color recursively the connected components, after the removal of the 1/2-separator. Thus, $\chi_{\mathrm{cf}}(T) \leq 1 + \lfloor \lg n \rfloor$ for a tree with n vertices. See also [34].

If a maximum color is used for every separator, the above coloring is an ordered coloring. Moreover, one can find optimal ordered colorings of trees [31].

It is unknown whether one can do better by using conflict-free colorings that are not ordered colorings.

2.3.4 Grid

A grid of size $m \times m$, i.e, with $n = m^2$ vertices can be colored with an ordered coloring with at most $4m$ colors: The idea is to use unique maximum colors for the row closest to the middle and column closest to the middle (that is less than $2m$ colors), and then color recursively in the 4 subgrids with size at most $\lfloor m/2 \rfloor \times \lfloor m/2 \rfloor$ each. A slight variation gives a coloring with at most $3m$ color: Use m unique maximum colors for the row closest to the middle, and then use about $m/2$ more unique colors for the part of the middle column over the middle row, and the same $m/2$ colors for the middle column under the middle row; then use recursion in the 4 subgrids with size at most $\lfloor m/2 \rfloor \times \lfloor m/2 \rfloor$ each.

There is also a lower bound of $\chi_o(G_m) \geq m$ (see [34]). Another proof is given here:

Proposition 16. *If G_m is the $m \times m$ grid, $\chi_o(G_m) \geq m$.*

Proof. By induction. Base: For $m = 1$, it is true, as $\chi_o(K_1) = 1$. For the inductive step, consider a Hamilton path p of G_m, with $m > 1$. If G_m is ordered colored, then there is a vertex v with a unique color in p (and thus in G). So, for some v, $\chi_o(G_m) = 1 + \chi_o(G_m - v)$. However, for every v, $G_{m-1} \preccurlyeq G_m - v$ (easy proof). Therefore, from proposition 10, $\chi_o(G) \geq 1 + \chi_o(G_{m-1})$ and from the inductive hypothesis, $\chi_o(G) \geq 1 + m - 1 = m$. □

A lower bound for conflict-free coloring is more elaborate to prove and is given in section 2.4.

More tight bounds on the ordered chromatic number for the grid are given in a following section and [7]. Variations of the problem that give rise to logarithmic colorings are given in a following section and [16].

2.4 Conflict-free coloring games

We define a game, given in figure 2.1, played by two players, on a graph G.

We also define the game, given in figure 2.2.

Every pair of consecutive moves, one from player 1 and then one from player 2, constitutes a *round* of the game.

The above games are obviously finite, because $|V(G_i)| > |V(G_{i+1})|$.

Connected-subset-CF-game(G):

```
$i \leftarrow 0$
$G_1 \leftarrow G$
while $G_i$ is not empty do:
    $i \leftarrow i + 1$
    Player 1 chooses a connected subset $S_i$ in $G_i$
    Player 2 chooses a vertex $v_i$ in $S_i$
    $G_{i+1} \leftarrow G_i[S_i] - v_i$
```

Figure 2.1: The connected-subset-CF-game

Path-CF-game(G):

```
$i \leftarrow 0$
$G_1 \leftarrow G$
while $G_i$ is not empty do:
    $i \leftarrow i + 1$
    Player 1 chooses a path $p_i$ in $G_i$
    Player 2 chooses a vertex $v_i$ in $p_i$
    $G_{i+1} \leftarrow G_i[V(p_i)] - v_i$
```

Figure 2.2: The path-CF-game

Proposition 17. *In the CS-CF-game, there is a strategy for player 2, so that the game stops in at most $\chi_o(G)$ rounds.*

Proof. By induction on $\chi_o(G)$: If $\chi_o(G) = 0$, i.e., the graph is empty, the game is played for zero rounds. If $\chi_o(G) = k > 0$, then in round 1 some connected set S_1 is chosen by Player 1. The strategy of player 2 is to find an optimal ordered coloring C of G and choose a vertex v_1 in S_1 that has a unique color in S_1. Then, $G_2 = G[S_1] - v_1 \subset G_1$ and the restriction of C to $S_1 - v_1$ is an ordered coloring that is using at most $k - 1$ colors. Thus, $\chi_o(G_2) \leq k - 1$, and by the inductive hypothesis Player 2 has a strategy so that the game stops after at most $k - 1$ more rounds. Therefore, the game is played in total for at most $1 + k - 1 = k$ rounds. □

A similar proposition is true for the path-CF-game:

Proposition 18. *In the path-CF-game, there is a strategy for player 2, so that the game stops in at most $\chi_{cf}(G)$ rounds.*

Proof. By induction on $\chi_{cf}(G)$: If $\chi_{cf}(G) = 0$, i.e., the graph is empty, the game is played for zero rounds. If $\chi_{cf}(G) = k > 0$, then in round 1 some path p_1 is chosen by Player 1. The strategy of player 2 is to find an optimal CF-coloring C of G and choose a vertex v_1 in p_1 that has a unique color in p_1. Then, $G_2 = G[V(p_1)] - v_1 \subset G_1$ and the restriction of C to $V(p_1) - v_1$ is a conflict-free coloring that is using at most $k-1$ colors. Thus, $\chi_{cf}(G_2) \leq k-1$, and by the inductive hypothesis Player 2 has a strategy so that the game stops after at most $k-1$ more rounds. Therefore, the game is played in total for at most $1 + k - 1 = k$ rounds. □

For the CS-CF-game, there is also a corresponding strategy for player 1. We first prove the following lemma:

Lemma 19. *For every $v \in V(G)$, $\chi_o(G - v) \geq \chi_o(G) - 1$*

Proof. Assume for the sake of contradiction that there exists a $v \in V(G)$ for which $\chi_o(G - v) < \chi_o(G) - 1$. Then an optimal coloring of $G - v$ can be extended to a coloring of G, where v has a new unique color. Therefore there is a coloring of G that uses less than $\chi_o(G) - 1 + 1 = \chi_o(G)$ colors; a contradiction. □

Proposition 20. *In the CS-CF-game, there is a strategy for player 1, so that the game goes on for at least $\chi_o(G)$ rounds.*

Proof. By induction on $\chi_o(G)$: If $\chi_o(G) = 0$, i.e., the graph is empty, the game is played for zero rounds. If $\chi_o(G) = k > 0$, the strategy of Player 1 is to choose an S_1 such that $\chi_o(G[S_1]) = k$; there is always such a connected subset, because otherwise one could color all connected components of G with less than k colors (a contradiction to $\chi_o(G) = k$). For every choice of v_1 by Player 2, by lemma 19, $\chi_o(G_2) \geq k - 1$, and thus, by the inductive hypothesis Player 1 has a strategy so that the game goes on for at least $k-1$ more rounds. Therefore, the game is played in total for at least $1+k-1 = k$ rounds. □

A proposition analogous to 20 for the path-CF-game is not known.

Now, for a class of graphs we will reduce the problem of CF-coloring with respect to paths, to CF-coloring with respect to connected sets. We remind the definition of a minor: A graph X is a *minor* of Y, denoted as $X \preccurlyeq Y$, if there is a subgraph G of Y, and a sequence $G_0, \ldots, G_k$, with $G_0 = G$ and $G_k = X$, such that $G_i = G_{i-1}/e_{i-1}$, where $e_{i-1} \in E(G_{i-1})$ (i.e., edge e_{i-1} is *contracted* in G_{i-1}), for $i \in \{1, \ldots, k\}$. In that case, we write $G = MX$, and (see [21]) there is a partition $\{V_x \mid x \in V(X)\}$ of $V(G)$ into connected subsets (in G), such that for any two vertices x, y of X, there is a V_x-V_y edge in G if and only if $xy \in E(X)$. Each V_x is called a *branch set.*

Theorem 21. *If $G = MX$, with branch sets $\{V_x \mid x \in V(X)\}$, and for each connected subset S of X, there is a path in G that covers all vertices in $\bigcup_{x \in S} V_x$, then $\chi_o(X) \leq \chi_{cf}(G)$.*

Proof. Idea: Player 1 plays the path-CF-game on G, by using a strategy in the CS-CF-game on X. □

The main theorem of this section is:

Theorem 22. $\chi_{cf}(G_m) \geq \lfloor m/2 \rfloor$.

Proof. It is enough to prove it for m even, because then, if m is odd, $G_m \supseteq G_{m-1}$ and from proposition 11, $\chi_{cf}(G_m) \geq \chi_{cf}(G_{m-1}) \geq (m-1)/2 = \lfloor m/2 \rfloor$.

For even m, $G_m = MG_{m/2}$. In order to define the branch sets we need, we will have to give names to the vertices of the two graphs:

$$V(G_m) = \{v_{i,j} \mid (i,j) \in \{1, \ldots, m\}^2\}$$

and

$$V(G_{m/2}) = \{u_{i,j} \mid (i,j) \in \{1, \ldots, m/2\}^2\}.$$

The edge sets of the two graphs are:

$$E(G_m) = \{v_{i,j}v_{i',j'} \mid |i-i'| + |j-j'| = 1\}$$

and

$$E(G_{m/2}) = \{u_{i,j}u_{i',j'} \mid |i-i'| + |j-j'| = 1\}.$$

The branch sets we define are, for every $u \in V(G_{m/2})$:

$$V_{u_{i,j}} = \{v_{2i-1,2j-1}, v_{2i-1,2j}, v_{2i,2j-1}, v_{2i,2j}\},$$

i.e., they induce $m^2/4$ 2×2 subgrids in G_m.

Now, it can be proved that for every connected set S in $G_{m/2}$, $\bigcup_{u \in S} V_u$ is covered by a path in G_m: Take a spanning tree of S, and prove by induction on the size of the tree that $\bigcup_{u \in S} V_u$ is covered by a path.

Therefore, by theorem 21, $\chi_{\text{cf}}(G_m) \geq \chi_{\text{o}}(G_{m/2}) \geq m/2$ (the last inequality by proposition 16). □

2.5 Conflict-free coloring for intervals

An interval containing n points is given (points are linearly ordered in the interval). Colors are positive integers. A coloring is an assignment of colors to the points of the interval. The coloring is represented by an array of positive integers $A[1..n]$, where for each point i, $A[i]$ is the assigned color. A *conflict-free* coloring is an assignment c of colors to the points, such that for every subinterval, there is a color that appears exactly once, i.e., for all i, j, with $1 \leq i \leq j \leq n$, the subarray $A[i..j]$ contains a color that appears exactly once in the subarray.

We try to find a conflict-free coloring using as few colors as possible. By convention if we need l different colors, we use the smallest possible positive numbers for colors: $\{1, 2, \dots, l\}$.

In the static version of the problem, the number of points, n, is given and we must come up with a conflict-free coloring using as few colors as possible. Let $C(n)$ be the minimum number of colors for conflict-free coloring an interval of n points. $C(n)$ is a non-decreasing function.

2.5.1 A lower bound for the number of colors needed

In order to find a lower bound for the number of colors needed for n points, first, we have to observe that in any conflict-free colored interval $[1..n]$, there is a color that appears exactly once. If that color is assigned to point k, then the subintervals $[1..k-1]$ and $[k+1..n]$ use one less color than the whole interval. We can also assume that one of the two subintervals uses only colors that appear in the other interval, because any interval that spans points in both $[1..k-1]$ and $[k+1..n]$ will also span point k and thus have $c(k)$ as its uniquely appearing color. If we also consider the non-decreasing property of the function $C(n)$, we can concentrate on the conflict-free coloring of the interval of maximum length among $[1..k-1]$ and $[k+1..n]$. The length of this interval is at least $\lfloor n/2 \rfloor$.

From the above, we have the following recurrence for $C(n)$:

$$\begin{aligned} &C(n) \geq 1 + C(\lfloor n/2 \rfloor) \\ &C(1) = 1 \end{aligned}$$

The solution of the recurrence relation gives:

$$C(n) \geq 1 + \lfloor \lg n \rfloor$$

2.5.2 A conflict-free coloring that achieves the lower bound

An offline coloring algorithm that achieves this lower bound goes like this:

Starting at point 1, color with color 1 every 2 points
Starting at point 2, color with color 2 every 4 points
...
Starting at point 2^{i-1}, color with color i every 2^i points
...
and keep doing this until you have colored all points.

Some C-like pseudocode for the above is given in figure 2.3.

Color i is used only if $n \geq 2^{i-1}$, so in fact $1 + \lfloor \lg n \rfloor$ colors are used by the algorithm. But is the coloring conflict-free?

We can describe more easily the coloring produced by the algorithm by using a binary tree of n nodes. If $n = 2^k - 1$, then the tree is full, and it is as follows: The levels of the tree are numbered from the bottom, to the top (root). The lowest level is level 1. Nodes at level i are labeled with number (color) i. The root is at level $1 + \lfloor \lg n \rfloor$, so it also has label $1 + \lfloor \lg n \rfloor$.

```
int first, step, color;
int point;
step = 1; color = 0;
while (first < n) {
    first = step;
    step = 2 * step;
    color = color + 1;
    for (point = first; point <= n; point = point + step) {
        A[point] = color;
    }
}
```

Figure 2.3: An optimal static conflict-free coloring algorithm with respect to intervals

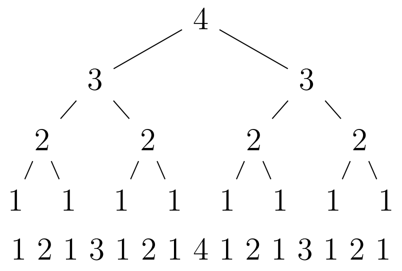
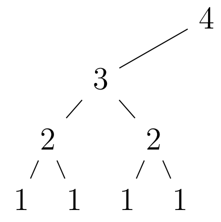


Figure 2.4: A tree for $n = 15$ and its inorder traversal

The conflict free coloring arises from an inorder traversal of the tree. See figure 2.4 for a tree for $n = 15$ and its inorder traversal.

If $n \neq 2^k - 1$, the tree is missing its rightmost[1] nodes and the description with levels is not that elegant in all cases. Nice examples in this case are trees with $n = 2^k$ (see figure 2.5).



1 2 1 3 1 2 1 4

Figure 2.5: A tree for $n = 8$ and its inorder traversal

In any case, we can prove formally, without using the tree representation that the coloring produced by the algorithm is conflict-free.

First, we prove the following fact for the coloring produced by the algorithm:

Fact 23. *For any color i that is repeated in some interval, there is a color i' in the coloring contained in that interval such that $i' > i$.*

Proof. Take two appearances of the same color i in the interval, at points p_1 and p_2, such that there is no point colored with i between p_1 and p_2. This means that $p_1 = 2^{i-1} + k \cdot 2^i$ and $p_2 = 2^{i-1} + (k+1) \cdot 2^i$, for some $k \geq 0$ (points consecutively colored with i are 2^i apart), so p_1 and p_2 have $2^i - 1$ points in between. Of these:

$$\begin{aligned} 2^{i-1} &\text{ have color } 1 \\ 2^{i-2} &\text{ have color } 2 \\ &\cdots \\ 2^{i-(i-1)} &\text{ have color } i-1 \end{aligned}$$

which sums up to $2^i - 2$ points. Therefore there is a point with a color i', such that $i' > i$. $\square$

Now, it is easy to prove the claim:

[1]Here 'rightmost' means rightmost with respect to the usual drawing of the tree, not rightmost innermost.

Claim 24. *The coloring produced by the algorithm is conflict-free.*

Proof. Assume there is an interval for which there is no uniquely appearing color. Take the maximum appearing color i in the interval and consider two appearances of i, such that there is no other appearance of color i in between them. By fact 23, there is a color i', such that $i' > i$ appearing in the interval, but this is a contradiction, because i is the maximum color in the interval. □

2.6 Improving the bounds for grids

In this chapter we tighten the upper and lower bounds for the exact ordered chromatic number of the $m \times m$ grid (G_m).

In the proofs we give below we partition the grid with the help of separators. All results are in the order of m, so without further mention we do not include terms logarithmic on m. These terms might be introduced by constant additive terms in a recursive bound. We are also omitting, in most cases floors and ceilings, because we are interested in asymptotic behavior. In that sense, a result like, for example, $\chi_{\mathrm{o}}(G_m) \leq 2.67m$ should be read as a bound of $2.67m \pm o(m)$.

2.7 Upper bound for grids

In order to improve the basic upper bound of $3m$ from a previous section, we need to find more intricate separators, that will be colored with unique colors. The idea is to use separators along diagonals in the grid. We will also need to find efficient colorings of some subgraphs that are left after we remove diagonal-like separators.

2.8 Special subgraphs in the grid

We exhibit colorings of subgraphs of the grid that allow us to improve the upper bound for $\chi_{\mathrm{o}}(G_m)$.

2.8.1 The rhombus

The rhombus R_x is the subgraph of the grid shown in figure 2.6. It has height x.

We have the following upper bound:

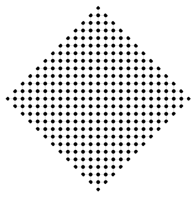


Figure 2.6: The rhombus

Proposition 25. $\chi_{\mathrm{o}}(R_x) \leq 3x/2$.

Proof. Use a diagonal separator to cut the rhombus in half ($x/2$ unique colors are used), then cut also the remaining parts in half with a diagonal separator ($x/4$ unique colors per part). This is shown in figure 2.7.

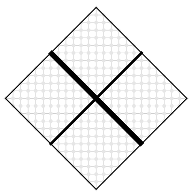


Figure 2.7: The rhombus separation

Therefore, we have the following recursive formula:

$$\chi_{\mathrm{o}}(R_x) \leq x/2 + x/4 + \chi_{\mathrm{o}}(R_{\lfloor x/2 \rfloor}).$$

Its solution gives: $\chi_{\mathrm{o}}(R_x) \leq 3x/2$. □

2.8.2 The triangle

The triangle T_x is the subgraph of the grid shown in figure 2.8. Its long side has length x.

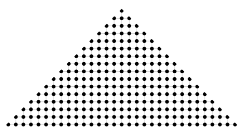


Figure 2.8: The wide triangle

We have the following upper bound:

Proposition 26. $\chi_{\mathrm{o}}(T_x) \leq 7x/6$.

Proof. See figure 2.9. Use a separator diagonally, parallel to the diagonal sides of the 'triangle' T_x, with $2x/6$ unique colors. In the two remaining parts, separate diagonally by using separators parallel to the other diagonal side of the 'triangle' T_x; each of those separators uses $x/6$ unique colors. With one more use of $x/6$ unique colors, we end up with connected components that are subgraphs of the rhombus $R_{2x/6}$.

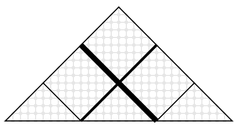


Figure 2.9: The wide triangle separation

Therefore, we have the following formula:

$$\chi_{\mathrm{o}}(T_x) \leq 2x/6 + x/6 + x/6 + \chi_{\mathrm{o}}(R_{\lfloor 2x/6 \rfloor}).$$

and since by proposition 25, $\chi_{\mathrm{o}}(R_x) \leq 3x/2$, we have $\chi_{\mathrm{o}}(T_x) \leq 7x/6$. □

2.8.3 A $8m/3$ upper bound

In figure 2.10, we show how a grid has to be partitioned with the help of separators to achieve a $8m/3$ upper bound.

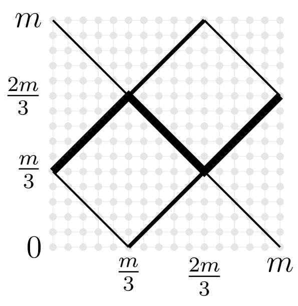


Figure 2.10: An $8m/3$ upper bound

The separators use m, $m/3$, and $m/3$ colors. Then, we have rhombi of height $2m/3$ that remain and, by proposition 25, can be colored with m colors. In total, we have $8m/3$ colors:

Proposition 27. *$\chi_{\mathrm{o}}(G_m) \leq 8m/3 \approx 2.6667m$.*

2.8.4 A $18m/7$ upper bound

In figure 2.11, we show how a grid has to be partitioned with the help of separators to achieve a $18m/7$ upper bound.

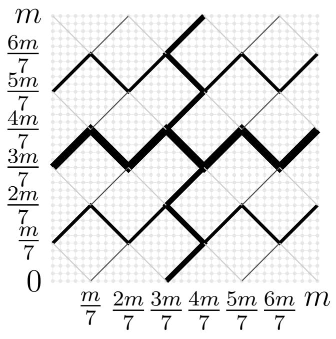


Figure 2.11: An $18m/7$ upper bound

The separators use m, $3m/7$, $3m/7$, $m/7$, and $m/7$ colors. Then, we have rhombi of height $2m/7$ that remain and, by proposition 25, each rhombus can be colored with $3m/7$ colors. In total, we have $18m/7$ colors:

Proposition 28. *$\chi_o(G_m) \leq 18m/7 \approx 2.5714m$.*

2.9 Lower bound for grids

In this section we sketch a proof of the following:

Theorem 29. *$\chi_o(G_m) \geq 1.33m$*

In order to prove a better lower bound than the one given in proposition 16, we need to find big grid minors after the removal of points with unique colors in the graph. It is only necessary to consider what grid minors we can find after removing separators. In fact, one can only consider *minimal separators.*

Definition 30. A separator S is minimal if for every $v \in S$, $S - v$ is not a separator.

From that point on, we consider all possible cases of minimal separators. The first categorization is: (a) central separators, and (b) side touching separators.

A central separator is one that does not touch the extreme points of the grid. It can be proved that one can remove the subgrid enclosing the central separator, and then find a big enough grid minor to achieve the $4m/3$ bound.

For side touching separators, one can find a big enough grid subgraph in one of the four corners of the grid, which is not touched by the separator.

2.10 Neochromatica

In the following sections, we look at a conflict-free coloring of a grid from a different point of view. It can be seen as a two-dimensional array of entries that represent the colors. We interpret the $m \times m$ grid conflict-free coloring problem in this context. Then, we relax the constraints of the problem in two ways to get colorings that use $\Theta(\log m)$ colors, whereas the original problem needs $\Theta(m)$ colors. We generalize to multidimensional arrays. Ernst Specker calls these new family of colorings *neochromatica*.

2.11 Arrays and meander paths

Consider the $m \times m$ grid with a conflict-free coloring with respect to paths. The paths defined in the graph are simple, in the sense that they are not self intersecting, and in the standard drawing of a graph (see figure 2.12) they always go along the horizontal or the vertical direction. Thus they look like *meanders*.

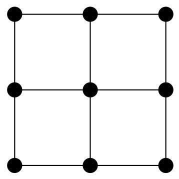


Figure 2.12: A 3×3 grid

It is not very convenient to place colors on a grid drawn like the one in figure 2.12. See figure 2.13.

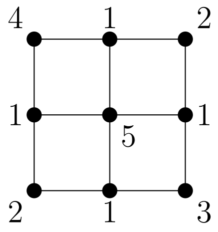


Figure 2.13: A conflict-free coloring of the 3×3 grid with respect to paths

Instead, it is more convenient to fill the colors in a two-dimensional array, of size m in each dimension. For example, see figure 2.14.

4	1	2
1	5	1
2	1	3

Figure 2.14: A conflict-free coloring of a 3×3 array with respect to meander paths

2.12 Arrays, subarrays, and thin subarrays

We are going to relax some of the constraints of conflict-free coloring with respect to meander paths, that forces linear use of colors with respect to m, to achieve logarithmic colorings with respect to m.

We do it in the following two ways:

- In every *subarray*, there must be a unique color.
- In every *thin* subarray, i.e., a subarray which has length 1 in one of the two dimensions, there must be a unique color.

We are going to use extensively the standard conflict-free coloring of the chain. If points are numbered 1 through n, from left to right on a chain, then the i-th point's color is denoted by $C(i)$, i.e., $C(1) = 1$, $C(2) = 2$, $C(3) = 1$, $C(4) = 3$, and so on.

2.12.1 Conflict-free coloring with respect to subarrays

The point (i_1, i_2) is colored as

$$C(i_1, i_2) = C(i_1) + C(i_2) - 1$$

The coloring uses asymptotically $2 \lg m$ colors.

2.12.2 Conflict-free coloring with respect to thin subarrays

The point (i_1, i_2) is colored as

$$C(i_1, i_2) = (C(i_1) + C(i_2) - 1) \bmod_1 \lceil \lg(m+1) \rceil$$

where $\bmod_1$ is the modulo operator, but returning $\lceil \lg(m+1) \rceil$ instead of 0 (i.e., its minimum output value is 1).

The coloring uses asymptotically $\lg m$ colors.

2.13 Multidimensional arrays

One can generalize the results of the previous section to multidimensional grids or arrays. A grid in d dimensions, in which each side has length m contains m^d vertices. A multidimensional grid can also be viewed as a multidimensional array. One can conflict-free color with respect to subarrays, or with respect to thin arrays (arrays which have length different than one in at most one dimension). Each point (or cell) of the grid (or array) is denoted by its d coordinates: $(i_1, \ldots, i_d)$. Each coordinate ranges from 1 to m.

2.13.1 Conflict-free coloring with respect to subarrays

The point $(i_1, \ldots, i_d)$ of the d-dimensional grid is colored as follows:

$$C(i_1, \ldots, i_k) = \sum_{k=1}^{d} C(i_k) - (d-1)$$

The above coloring uses asymptotically $d \lg m$ colors.

The same problem has been solved in [29]: A d-dimensional grid is colored with respect to subsets of points defined by axis parallel rectangular boxes.

2.13.2 Conflict-free coloring with respect to thin subarrays

The point $(i_1, \ldots, i_d)$ of the d-dimensional grid is colored as follows:

$$C(i_1, \ldots, i_k) = (\sum_{k=1}^{d} C(i_k) - (d-1)) \bmod{}_1 \lceil \lg(m+1) \rceil$$

where $\bmod{}_1$ is the modulo operator, but returning $\lceil \lg(m+1) \rceil$ instead of 0 (i.e., its minimum output value is 1).

It is interesting that the above coloring uses asymptotically only $\lg m$ colors, i.e., the number of colors used does not depend on the dimension d. Another interesting fact is that the coloring is very far from satisfying the unique max property. It is an open problem, whether one can use $O(\log m)$ colors with this additional stronger constraint.

Chapter 3

Greedy algorithms for online conflict-free coloring

3.1 Points appearing dynamically: more constraints

In the online dynamic version of the problem, points are inserted in succession, between any existing points, and a color must be assigned to each point, so that the coloring is conflict-free *at all times*.

3.2 Insertion sequences and permutations

The sequence of inserting points can be described by the position in which each new point is inserted. If $i-1$ points have already been inserted, the i-th point can go in any of i positions described by an integer in the range $[0..i-1]$: 0 means put the new point in the start of the sequence (before any other point), and $k > 0$ means put the new point immediately after the k-th existing point.

An insertion sequence of length n is represented by an array $s[1..n]$, where $0 \leq s[i] \leq i-1$. We consider insertion sequences of the same length n ordered lexicographically. The first and last elements in that order are:

$$\begin{aligned} s_n^{\text{first}} &= [0, 0, 0, \ldots, 0] \\ s_n^{\text{last}} &= [0, 1, 2, \ldots, n-1] \end{aligned}$$

We also call insertion sequences, described as above, *relative positions* insertion sequences.

There are $n!$ possible insertion sequences of length n. It is no coincidence that there is a natural correspondence with the permutations of length n. A permutation is represented by an array $p[1..n]$, whose elements are of course a permutation of $\{1, \dots, n\}$. If you read the permutation from the left to the right you see the position at which the inserted point will end up in the final interval, i.e., the i-th inserted point will end up at position $p[i]$ of the final interval. There are functions for converting between insertion sequences and permutations, under this interpretation. If an input is given in the form of a permutation, described as above, we say that the input is given in *absolute positions*.

In the dynamic *offline* setting, the input can be given in either absolute or relative positions, because the two representations are easily convertible to each other if the *whole* sequence is known. For example, the insertion sequence $\sigma = 00121$ corresponds to the permutation $\pi = 51342$, which means the first point inserted is at the 5^{th} absolute position (rightmost), the second point inserted is at the 1^{st} absolute position (leftmost), and so on.

3.3 Separation of static and dynamic versions of the problem

The dynamic version of the problem is more difficult than the static version of the problem, because there might be more constraints to be satisfied. Even for $n = 3$, we have the static coloring 121, but the insertion sequence $\sigma = 011$ ($\pi = 132$ in absolute positions) needs three different colors to be colored dynamically. In [8], we exhibit sequences that separate the two models by more than one color. Similar separation results can be proved: (a) between the dynamic offline and the absolute positions model, and (b) between the absolute positions and the relative positions model. Details are omitted for these two cases.

In order to separate the offline dynamic model from the online models, we need to exhibit two insertion sequences of absolute positions that have a common prefix, but the optimal offline algorithm colors essentially differently the sequences, up to that prefix. Two such sequences of absolute positions are: $\pi = 123654$ and $\pi' = 123465$, which share the common prefix 123. Two optimal colorings for the above sequences are: $C = 123121$ and $C' = 121312$, respectively, and it is easy to see that for an optimal coloring to happen, for π the algorithm has to use a new color for the third point, whereas for π' the algorithm has to reuse a previously used color. This means that there is no deterministic online algorithm which achieves the optimal coloring for both π and π'. Similarly, in order to separate the absolute position from the relative position online model, we need to exhibit two insertion sequences of relative positions that have a common prefix, but for which the respective absolute positions inputs have different prefixes, so knowledge of absolute positions might allow us to color the sequences better. Two possible sequences of relative positions are: $\sigma = 0122456$ and $\sigma' = 0123456$ with common prefix 012. The corresponding absolute positions sequences are: $\pi = 1243567$ and $\pi' = 1234567$, i.e., they differ in the third position. In both cases the optimal coloring is of the form 1213121 and it only happens if for σ the algorithm uses a new color for the third point, whereas for σ' the algorithm reuses a previously used color. This means that there is no deterministic online algorithm on relative positions that can achieve as good color use as a deterministic online algorithm on absolute positions, because the relative positions algorithm can not differentiate between the two sequences early enough.

More involved separation results are given in [8, 9].

3.4 First-fit greedy algorithm

The first-fit greedy algorithm does the following:

> For each new point inserted, try to color it with the minimum color possible, such that the conflict-free property is maintained.

For example, the first-fit greedy algorithm colors insertion sequence $\sigma = 010322$ ($\pi = 251643$ in absolute positions) as follows:

$$\begin{array}{c} [.1....], \\ [.1..2.], \\ [21..2.], \\ [21..23], \\ [21.323], \\ [214323]. \end{array}$$

The analysis of the first-fit greedy algorithm was left as an open problem in [27].

Implementation

For each new point inserted, you only need to check the intervals in which this point lies and give the new point the minimum color that makes all those intervals have the uniquely appearing color property.

The above idea is helpful in proofs too. To prove that a coloring is conflict-free after an insertion of a point in an already conflict-free color sequence, you have just to check those new intervals.

3.5 Some observations related to the greedy algorithm

The greedy algorithm applied to the sequence s_n^{last} gives the same coloring as the optimal offline algorithm:

$$\begin{aligned}
0 &\to 1\\
01 &\to 12\\
012 &\to 121\\
0123 &\to 1213\\
01234 &\to 12131\\
012345 &\to 121312\\
0123456 &\to 1213121\\
01234567 &\to 12131214
\end{aligned}$$

You get a similar coloring for the sequence s_n^{first}:

$$\begin{aligned}
0 &\to 1\\
00 &\to 21\\
000 &\to 121\\
0000 &\to 3121\\
00000 &\to 13121\\
000000 &\to 213121\\
0000000 &\to 1213121\\
00000000 &\to 41213121
\end{aligned}$$

In table 3.1 we show the best and worst use of colors for all sequences of the same length, for lengths up to 12. (Best is always $1 + \lfloor \lg n \rfloor$.)

In table 3.2 we show the first appearing insertion sequence (in the lexicographic order) which uses the maximum color for length from 5 to 12.

In table 3.3 we show for each length how many sequences use some number of colors (we also give the ratio in table 3.4).

n	best	worst
1	1	1
2	2	2
3	2	3
4	3	3
5	3	4
6	3	4
7	3	5
8	4	5
9	4	6
10	4	6
11	4	7
12	4	7
13	4	8?

Table 3.1: Best and worst use of colors for greedy

n	sequence	coloring
5	0 0 0 0 1	3 4 1 2 1
6	0 0 0 0 0 2	1 3 4 1 2 1
7	0 0 0 0 1 2 3	3 4 3 5 1 2 1
8	0 0 0 0 0 0 6 6	2 1 3 1 2 1 5 4
9	0 0 0 0 1 3 6 7 7	3 4 1 3 2 1 2 6 5
10	0 0 0 0 0 0 6 5 8 8	2 1 3 1 2 4 1 4 6 5
11	0 0 0 0 1 3 6 7 6 9 9	3 4 1 3 2 1 5 2 5 7 6
12	0 0 0 0 0 0 6 5 8 7 10 10	2 1 3 1 2 4 1 5 4 5 7 6
13	0 0 0 0 1 3 6 7 6 9 8 11 11 ?	3 4 1 3 2 1 5 2 6 5 6 7 8 7

Table 3.2: First sequence using most colors for its length for greedy

n	$n!$	3	4	5	6	7	8
5	120	108	12				
6	720	376	344				
7	5040	752	4248	40			
8	40320		38092	2228			
9	362880		299924	62880	76		
10	3628800		2329592	1293848	5360		
11	39916800		17080328	22577320	259040	112	
12	479001600		110463696	358955060	9573588	9256	
13	6227020800		577315696	5356053620	293128704	522636	144

Table 3.3: Sequences using numbers of colors for greedy

3.6 A sequence of bad insertion sequences

We will give a sequence of bad insertion sequences, i.e., sequences that make the greedy algorithm use many colors. More precisely, for any odd $n \geq 9$ there is an insertion sequence b^n of length n that uses $m_n = \frac{n+3}{2}$ colors.

We use this notation: If s is an insertion sequence of length n, then for all i, j, such that $1 \leq i \leq j \leq n$, we denote the subsequence starting at point i and ending at point j as $s_{[i..j]}$. We also use '+' to denote concatenation of sequences. The coloring you get by the greedy algorithm on a sequence s is denoted by $c(s)$ ($c(s)$ is also a sequence, so we can use the same notation for color sequences).

The bad sequences are defined recursively:

$$\begin{aligned} b^9 &= [0,0,0,0,1,3,6,7,7], \\ b^{n+2} &= b^n_{[1..n-1]} + [n-3, n, n], \quad \text{for } n \geq 9. \end{aligned}$$

We denote the coloring (i.e., a color sequence) that the algorithm assigns to b^n with $c^n (= c(b^n))$.

Proposition 31. *For all odd $n \geq 9$:*

- *c^n uses $m_n = \frac{n+3}{2}$ colors.*
- *Colors m_n and $m_n - 1$ are not used in $c^n_{[1..n-2]}$.*
- *$c^n_{[n-1..n]}$ is $[m_n, m_n - 1]$.*
- *c^n_{n-2} and c^n_{n-4} are the same color (which is different from m_n, $m_n - 1$).*

Proof by induction. Base case ($n = 9$): True.

Inductive step: Assume the proposition is true for b^n. We will prove that it is true for b^{n+2}.

Take out the last element of b^n. You get sequence $s = b^n_{[1..n-1]}$ and coloring $c(s) = c^n_{[1..n-2]} + [m_n - 1]$, where the $c^n_{[1..n-2]}$ part uses only colors up to $m_n - 2$.

Do one more insertion at the sequence s after point $n-3$. You get sequence $s' = s + [n-3]$. Color it using the greedy algorithm to get the coloring $c' = c(s')$. We show that the color that the greedy algorithm assigns to this last insertion after point $n-3$ (this is point $n-2$) is $m_n - 1$, i.e., $c'_{n-2} = m_n - 1$:

- c' is conflict-free: We build on the conflict-free property of $c(s)$ and the fact that $c(s)$ does not contain $m_n - 1$ except in point $n-1$. We have to check only new intervals for the last point added to s'. From these we have that any interval $c'_{[i..n-2]}$, for $1 \leq i \leq n-2$, is conflict free. Also, any interval in $c'_{n-2..n} = [m_n - 1, c^n_{n-2}, m_n - 1]$ is conflict-free because c^n_{n-2} is less than $m_n - 1$ (by the induction hypothesis).

- The greedy algorithm can not give a color less than $m_n - 1$ to c'_{n-2} (proof by contradiction): $c^n_{[1..n-3]}$ and $c'_{[1..n-3]}$ are the same and in the run of the greedy algorithm for b^n: (a) $c^n_{[1..n-3]}$ is an intermediate coloring, (b) immediately after the appearance of $c^n_{[1..n-3]}$, point $n-2$ is colored with color $m_n - 1$ (because $b^n_{n-2} = n-3$ and by the induction hypothesis). If $c'_{n-2} < m_n - 1$ then the greedy algorithm would also choose this smaller color for the insertion sequence b^n, which is a contradiction.

Do one more insertion at the sequence s' after point n. You have sequence $s'' = s' + [n]$ and coloring $c'' = c(s'')$. The new color assigned is $c''_{n+1} = m_n$. This happens because the last color chosen for coloring b^n is also m_n: The new color can not be $m_n - 1$ or c^n_{n-2}, because $s'_{n-2..n}$ is $[m_n - 1, c^n_{n-2}, m_n - 1]$. It can also be no other color $c''_{n+1} < m_n$, because then the greedy algorithm could use that color at the last insertion for b^n (see lemma 32).

Do one more insertion at the sequence s'' after point n. You have sequence $b^{n+2} = s'' + [n]$ and coloring c^{n+2}. The new color assigned is $m_n + 1$, because from the previous argument it has to be at least m_n, but it has already an adjacent m_n colored point (the last point). Thus, the coloring sequence is:

$$c^{n-2} = c^n_{[1..n-3]} + [m_n - 1, c^n_{n-2}, m_n - 1, m_n + 1, m_n]$$

which has all the required properties. □

Lemma 32. *If the greedy algorithm gives a point inserted at the end of the color sequence:*

$$c(s') = c^n_{[1..n-5]} + [c^n_{n-2}, y, m_n - 1, c^n_{n-2}, m_n - 1]$$

the color x, i.e.:

$$c(s'') = c^n_{[1..n-5]} + [c^n_{n-2}, y, m_n - 1, c^n_{n-2}, m_n - 1, x]$$

then the greedy algorithm assigns the same color x to a point inserted between the last two points of the color sequence:

$$c(b^n_{[1..n-1]}) = c^n_{[1..n-5]} + [c^n_{n-2}, y, c^n_{n-2}, m_n - 1]$$

giving the color sequence:

$$c(b^n) = c^n_{[1..n-5]} + [c^n_{n-2}, y, c^n_{n-2}, x, m_n - 1]$$

Proof. x can not be any of c^n_{n-2}, $m_n - 1$, or y.

Now, in

$$c'' = c(s'') = c^n_{[1..n-5]} + [c^n_{n-2}, y, m_n - 1, c^n_{n-2}, m_n - 1, x],$$

any interval $c''_{[i..n+1]}$ has a point j in which a unique color c''_j appears. This color can be x, y, or one in $c''_{[i..n-5]} = c^n_{[i..n-5]}$. For the color sequence

$$c^n = c(b^n) = c^n_{[1..n-5]} + [c^n_{n-2}, y, c^n_{n-2}, x, m_n - 1],$$

in the interval $c^n_{[i..n-1]}$, for the same i, c''_j is also a uniquely appearing color (if it is in $c^n_{[i..n-5]}$ or y, then it is also appearing at the same point j; otherwise it is x). Intervals that contain the last point, have the unique color $m_n - 1$ at that last point. Therefore, c^n is a conflict-free coloring. □

3.7 A simpler sequence of bad insertion sequences

There is a very simple sequence of bad insertion sequences that achieves the same use of colors, that inserts only at the first three positions from the left. Since the only possible elements of the insertion sequence are 0, 1, 2, we use the more compact string notation for these bad sequences. The bad sequences are:

$$001, 00201, 0020201, 002020201, \ldots$$

or, more formally, using the w^i notation to denote concatenation of i copies of the same string w:

$$\{00(20)^i 1 \mid i \in \mathbb{N}\}$$

We will prove that $00(20)^i 1$, of length $2i + 3$, uses $i + 3$ colors, and we start with:

Lemma 33. *Insertion sequence* $00(20)^i$ *uses* $i + 2$ *colors and the first two colors in the coloring are* $i + 2$, $i + 1$. *The third and fourth colors, in case* $i > 0$ *are* i, $i + 1$.

Proof by induction. Base case ($i = 0$ and 1): True: 00 gives the coloring 21 and 0020 gives the coloring 3212.

Inductive step: Assume the proposition is true for $i > 0$. We will prove that it is true for $i + 1$.

The coloring for $i > 0$ is:

$$c^i = \quad i+2 \quad i+1 \quad i \quad i+1 \quad \ldots$$

The next insertion (at position 2) is between $i + 1$ and i. It has to get color $i + 2$, because if it would get another smaller color ($i + 1$ is also impossible), then this color could also be used as the color occurring in the first position of c^i. The coloring becomes:

$$i+2 \quad i+1 \quad i+2 \quad i \quad i+1 \quad \ldots$$

The next insertion (at position 0) can not get a color among $i+2$, $i+1$, i. It can not get any other already used color, because then this color could also be used as the color occurring in the first position of c^i. So, a new color has to be used and we get:

$$c^{i+1} = \quad i+3 \quad i+2 \quad i+1 \quad i+2 \quad \ldots$$

□

Lemma 34. *Insertion sequence* $00(20)^i1$ *uses* $i + 3$ *colors.*

Proof. We augment the coloring of $00(20)^i$ that we have from lemma 33, which is:

$$i+2 \quad i+1 \quad \ldots$$

The insertion (at position 1) between $i + 2$ and $i + 1$ has to get a new color, because any other color, except $i + 2$ which is in any case impossible, would be chosen by the greedy algorithm when coloring $00(20)^i$, in the last insertion. □

We define $C_G(n)$ as the maximum number of colors used among all insertion sequences of length n, when the greedy algorithm is used for coloring.

We have proved the following:

Proposition 35. *For* $n \geq 2$*:* $C_G(n) \geq \lceil n/2 \rceil + 1$.

3.7.1 An upper bound on the use of colors

In order to prove an upper bound on the use of colors, we consider uniquely occurring colors in a coloring.

Lemma 36. *In any greedy coloring there are at most three distinct colors with the following property: each of those colors occurs exactly once.*

Proof. Assume the three colors x, y, z that occur uniquely in the coloring:

$$\dots x \dots y \dots z \dots$$

W.l.o.g., a new point can be inserted:

- either between y and z, so it can get color x,
- or after z, so it can get color x or y.

In any case, no new color has to be introduced. □

Proposition 37. *For $n \geq 2$: $C_{\mathrm{G}}(n) \leq \lceil n/2 \rceil + 1$.*

Proof. By induction on the insertion of points.

Base: For $n = 2$ it is true.

Induction step: Assume it is true for $n \geq 2$. We will prove it for $n + 1$. We have the following two cases:

- If n is even, then a new color can be used and we have $C_{\mathrm{G}}(n+1) \leq C_{\mathrm{G}}(n) + 1 \leq \lceil n/2 \rceil + 1 + 1 = \lceil (n+1)/2 \rceil + 1$.
- If n is odd, then either less than $\lceil n/2 \rceil + 1$ colors are used (in which case, even a use of a new color for the newly inserted point leaves us with less than $\lceil (n+1)/2 \rceil + 1$ colors), or exactly $\lceil n/2 \rceil + 1$ colors are used. In that last case, the $\lceil n/2 \rceil + 1 = (n+3)/2$ used colors occur at $(n+3)/2$ points and the remaining $n - (n+3)/2 = (n-3)/2$ points must also be colored by those same colors, so we have at least $(n+3)/2 - (n-3)/2 = 3$ colors that occur exactly once, and by lemma 36 exactly 3 colors that occur exactly once. Also by lemma 36, the newly inserted point can not get a new color, so still $\lceil n/2 \rceil + 1 = \lceil (n+1)/2 \rceil + 1$ are used.

In any case, $C_{\mathrm{G}}(n+1) \leq \lceil (n+1)/2 \rceil + 1$. □

Corollary 38. *For $n \geq 2$: $C_{\mathrm{G}}(n) = \lceil n/2 \rceil + 1$.*

3.8 Description of unique max algorithm

The unique max algorithm does the following:

> For each new point inserted, try to color it with the minimum color possible, such that the following property is maintained: the maximum color in every interval is unique.

The algorithm was introduced in [27].

For each new point inserted, you only need to check the intervals in which this point lies and give the new point the minimum color that makes all those intervals have the unique maximum property. However there is a more efficient way to implement the algorithm, which is also described in [27].

3.9 Running UM for small insertion sequences

In table 3.5 we show the best and worst use of colors for all sequences of the same length, for lengths up to 14. (Best is always $1 + \lfloor \lg n \rfloor$.)

In table 3.6 we show the first appearing insertion sequence (in the lexicographic order) which uses the maximum color for length from 5 to 13.

In table 3.7 we show for each length how many sequences use some number of colors (we also give the ratio in table 3.8).

3.10 Signed insertion sequences

Sometimes we have a insertion sequence that needs to make a lot of insertions close to the right end of the coloring. In that case, it would be easier to specify the distance from the right end when making such insertions. For this reason, we introduce the notion of a *signed insertion sequence*, in which each insertion is a signed (positive or negative) integer: positive integers denote insertion from the left end of the coloring and negative integers from the right end. For example, an insertion $-j$ means insert j points from the end of the coloring.

We remark that insertions $+0$ and -0 are very different: the former means insert at before the start of the coloring and the latter means insert after the end of the coloring.

The algorithm to convert a signed insertion sequence s to an unsigned one, u, is:

n	$n!$	3	4	5	6	7	8
5	120	0.900000	0.100000				
6	720	0.522222	0.477778				
7	5040	0.149206	0.842857	0.007937			
8	40320		0.944742	0.055258			
9	362880		0.826510	0.173280	0.000209		
10	3628800		0.641973	0.356550	0.001477		
11	39916800		0.427898	0.565609	0.006489	0.000003	
12	479001600		0.230612	0.749382	0.019987	0.000019	
13	6227020800		0.092711	0.860131	0.047074	0.000084	0.000000

Table 3.4: Sequences ratio using numbers of colors for greedy

n	best	worst
1	1	1
2	2	2
3	2	3
4	3	4
5	3	4
6	3	5
7	3	5
8	4	6
9	4	6
10	4	7
11	4	7
12	4	8
13	4	8
14	4	9

Table 3.5: Best and worst use of colors for UM

n	sequence	coloring
5	0 0 0 0 1	3 4 1 2 1
6	0 0 0 0 3 1	3 5 1 2 4 1
7	0 0 0 0 0 4 2	1 3 5 1 2 4 1
8	0 0 0 0 1 2 5 3	3 4 3 6 1 2 5 1
9	0 0 0 0 0 0 1 6 4	2 4 1 3 6 1 2 5 1
10	0 0 0 0 1 2 6 6 5 3	3 4 3 7 1 2 5 1 6 5
11	0 0 0 0 0 0 1 7 7 6 4	2 4 1 3 7 1 2 5 1 6 5
12	0 0 0 0 4 4 4 1 4 2 3 4	3 6 7 6 8 1 2 3 1 4 5 4
13	0 0 0 0 0 0 2 2 1 9 9 8 6	2 5 1 3 4 3 8 1 2 6 1 7 6

Table 3.6: First sequence using most colors for its length for UM

n	$n!$	4	5	6	7	8	9
8	40320	18048	21660	612			
9	362880	88924	248592	25364			
10	3628800	421988	2577760	627160	1892		
11	39916800	1804664	25618312	12314436	179388		
12	479001600	6515832	248613404	215646600	8223844	1920	
13	6227020800	18647040	2390241068	3543308768	274441996	381928	
14	87178291200	37294080	23080273748	56291051128	7739387516	30284536	192

Table 3.7: Sequences using numbers of colors for UM

n	$n!$	4	5	6	7	8	9
8	40320	0.447619	0.537202	0.015179			
9	362880	0.245051	0.685053	0.069896			
10	3628800	0.116289	0.710362	0.172828	0.000521		
11	39916800	0.045211	0.641793	0.308503	0.004494		
12	479001600	0.013603	0.519024	0.450200	0.017169	0.000004	
13	6227020800	0.002995	0.383850	0.569022	0.044073	0.000061	
14	87178291200	0.000428	0.264748	0.645700	0.088777	0.000347	0.000000

Table 3.8: Sequences ratio using numbers of colors for UM

```
for $i \leftarrow 1$ to length($s$) do:
    if $s_i$ is positive signed then:
        $u_i \leftarrow |s_i|$
    else: // $s_i$ is negative signed
        $u_i \leftarrow i - 1 - |s_i|$
```

3.11 A lower bound on the use of colors by UM

In [27], for all k, a coloring C_k produced by the UM algorithm is exhibited:

$$C_k = [\underbrace{\ 1\ }_{D_1}, \underbrace{2, 1}_{D_2}, \underbrace{3, 2, 1}_{D_3}, \dots, \underbrace{k-1, k-2, \dots, 1}_{D_{k-1}}, \underbrace{k, k-1, \dots, 1}_{D_k}],$$

where D_k is the decreasing sequence $[k, \dots, 1]$.

This coloring uses k colors and has length $k(k+1)/2$.

The proof that UM produces such a coloring is by induction on k. For $k = 1$, 2, it is true. Now, to get C_{k-1} from C_k do the following $k+1$ insertions in the following order: insert between D_k and D_{k-1}, insert between D_{k-1} and D_{k-2}, …, insert between D_2 and D_1, insert at the start of the sequence, insert at the start of the sequence; the colors assigned by UM are $k+1$, k, …, 2, 1, respectively.

By symmetry, UM can also produce the coloring:

$$C_k^I = [\underbrace{1, 2, \ldots, k}_{I_k}, \underbrace{1, 2, \ldots, k-1}_{I_{k-1}}, \ldots, \underbrace{1, 2, 3}_{I_3}, \underbrace{1, 2}_{I_2}, \underbrace{\ 1\ }_{I_1}],$$

where I_k is the increasing sequence $[1, \ldots, k]$.

3.12 All-seeing colorings

Definition 39. A *left-end all-seeing* (leas) coloring is a coloring in which all colors are visible from the left end of the coloring.

Symmetrically, a *right-end all-seeing* coloring (reas) is a coloring in which all colors are visible from the right end of the coloring.

A coloring which is both leas and reas is called *both-ends all-seeing* (beas) coloring.

We remark that colorings C_k and C_k^I of section 3.11 are beas colorings.

Proposition 40. *In a beas coloring, the only color that occurs uniquely is the maximum color.*

Beas colorings are very bad in terms of avoiding new colors in subsequent insertions of the UM algorithm. In fact we can immediately, with three insertions, get three new colors: If B_k is a beas coloring which uses k colors, then it is possible to get the coloring:

$$[k+1] + B_k + [k+3, k+2]$$

Similarly, in a leas (or a reas) coloring, we can immediately insert two new colors. For example, if L_k is a leas coloring, which uses k colors, then it is possible to get the coloring:

$$[k+1, k+2] + L_k$$

This means, that a leas (or a reas) coloring contains at most two uniquely occurring colors.

3.12.1 Smallest all-seeing colorings

A k-beas coloring is a beas coloring in which the unique maximum color is k. We define similarly a k-leas coloring.

k	1	2	3	4	5
n	1	3	5	8	11

Table 3.9: Length of shortest k-beas colorings

In table 3.9 we show the length n of the shortest k-beas coloring, for small k.

Those shortest colorings are as follows: 1, 121, 12321, and then for $k = 4$:

12134321
12314321
12341321
12343121

and for $k = 5$:

12321454321
12324154321
12345142321
12345412321

In table 3.10 we show the length n of the shortest k-leas coloring, for small k.

k	1	2	3	4	5	6	7
n	1	2	4	6	8	11	14

Table 3.10: Length of shortest k-leas colorings

Those shortest colorings are as follows: 1, 12, and then for $k = 3$: 1213, 1231, 1232, and for $k = 4$:

121343
123142
123143
123214
123241
123412

and for $k = 5$:

12321454
12324154

12343152
12343512

and for $k = 6$:

12134321565
12134325165
12134541623
12134546123
12314321565
12314325165
12314541623
12314546123
12324541623
12324546123
12341321565
12341325165
12343121565
12343125165
12343152165
12343152615
12343215652
12343251652
12343512165
12343512615
12345142163
12345142613
12345412163
12345412613
12345416213

12345416231
12345416232
12345461213
12345461231
12345461232

3.13 Colorings with a specific number of occurrences of a color

How many insertion sequences of length n, if the UM algorithm is used, give a coloring in which color c occurs exactly i times? We denote this number by $N_c(n, i)$.

In the following, we give an analysis for $c = 1$ and we exhibit a connection with permutations with a specific number of valleys.

3.13.1 A connection of $N_1(n, i)$ with permutations with $i - 1$ valleys

As we have already seen, the insertion sequences of n points in the interval are in a 1-1 correspondence with the permutations of length n.

$N_1(n, i)$ is the number of insertion sequences of length n for which color 1 is repeated exactly i times, if the UM algorithm is used.

It is easy to find $N_1(n, 1)$. For $n = 1$, we have just $N_1(1, 1) = 1$. Now, the only way to maintain occurrence of color 1 exactly one time is to insert the next point to the left or to the right of the unique point colored with color 1, i.e., $N_1(n + 1, 1) = 2N_1(n, 1)$, which gives $N_1(n, 1) = 2^{n-1}$.

$N_1(n, 1)$ is the same as the number of valleyless permutations of length n. A permutation π is valleyless if the graph $\{(i, \pi_i) \mid 1 \leq i \leq n\}$ has no valley, i.e., there is no i such that π_i is less than a $\pi_{i'}$ and a $\pi_{i''}$, with $i' < i < i''$. It is easy to see that valleyless permutations are increasing, or decreasing, or increasing and then decreasing (in this last case, they have one peak). See [48] on how a similar counting argument gives the above result on $N_1(n, 1)$: The key observation is that in a valleyless permutation the smallest element (i.e., 1) appears either at the leftmost end or the rightmost end of the permutation.

In general $N_1(n, i)$ is the same as the number of permutations of length n with exactly $i-1$ valleys. In order to compute $N_1(n, i)$, we need $N_1(1, i) = 0$, for $i > 1$. Now for $n > 1$, $i > 1$, how we can get a coloring of length n with exactly i occurrences of 1? There are two ways:

- Start with a $n-1$ length coloring in which color 1 occurs exactly i times and insert a point to the left or to the right of these colored with 1 points. There are $2i$ such positions because points colored with 1 are not adjacent.

- Start with a $n-1$ length coloring in which color 1 appears exactly $i-1$ times and insert the new point at a position which is not adjacent to a point colored with 1. There are $n-2(i-1)$ such positions.

Thus, we get the recursive formula:

$$N_1(n,i) = 2i \cdot N_1(n-1,i) + (n-2(i-1)) \cdot N_1(n-1,i-1)$$

for $n > 1$, $i > 1$. A similar argument is used in [48], where, also, generating functions are provided.

Therefore there is a 1-1 correspondence between permutations of length n with exactly $i-1$ valleys and colorings by the UM algorithm of length n with exactly i occurrences of color 1.

3.14 The decrement operation on colorings

Consider the following operation on colorings: Delete all 1's in a coloring sequence and then decrease everything remaining by 1. We call this the *decrement* operation and denote it by dec.

If we apply the decrement operation to a coloring produced by the greedy algorithm, it is not always the case that we get a coloring that is conflict free: For example, $\operatorname{dec} 212 = 11$, which does not have the conflict free property.

However, if we apply dec to a coloring produced by the UM algorithm, we get a coloring which can be produced by the UM algorithm. In fact, if we have the insertion sequence that produced coloring C, we can (with the help of the coloring) transform it to a sequence producing $\operatorname{dec} C$. The proof is elementary but tedious.

3.15 Insertions to the left or right of a point

In this section, we consider insertion sequences which always insert immediately to the left or to the right of an existing, already colored, point. We restrict ourselves to the simple case where there is only one point colored with 1, and all remaining points are inserted adjacent to that point, by using the greedy or the UM algorithm.

An insertion sequence of length n that inserts always to the left or to the right of the '1' that appeared in the first insertion, can be represented by a binary sequence of length $n-1$. We call these sequences *lr-sequences* and we represent an insertion to the left by 'l' and an insertion to the right by 'r'.

For example, if the UM algorithm is used and we have the lr-sequence rll, we get the coloring 3412. For completeness, the empty lr-sequence ε gives coloring 1. In the case of the UM algorithm, the colorings with one '1' are exactly the ones we get with the lr-sequences (as we have seen, $N_1(n,1) = 2^{n-1}$). However, with the greedy algorithm one can get a single '1' in the coloring, although there are insertions not immediately adjacent to the existing 1, e.g., the insertion sequence 0100 gives coloring 3212.

3.15.1 Use of colors by UM colorings with one '1'

In table 3.11, we show for each of the provided lengths, the lexicographically first lr-sequence that gives the maximum use of colors for that length. In table 3.12, we show the corresponding coloring of the lexicographically first lr-sequence.

In table 3.13 we show for each length how many lr-sequences use some number of colors (we also give the ratio in tables 3.14 and 3.15).

3.15.2 lr-sequences and the greedy algorithm

In the greedy algorithm, if insertions are always made immediately adjacent to the first '1' that appeared, insertions to the left take colors do not depend on the colored points to the right of the '1', and vice versa. This is because the unique '1' plays the role of a separator between the left and the right part of the sequence. Thus, the coloring depends only on the number of insertions to the left and to the right. The coloring of points to the left is similar to the optimal coloring of the offline algorithm (see subsection 2.5.2): we already have the unique '1', so all other colors have to be shifted up by one color. We have a symmetric coloring for points inserted to the right. This means, that if we have i_l insertions to the left and i_r insertions to the right, we have $n = 1 + i_l + i_r$, and $2 + \lfloor \lg \max(i_l, i_r) \rfloor$ colors are used, where $i_l + i_r > 0$.

Therefore, the maximum use of colors happens when all points are inserted in one of the two directions and is $2 + \lfloor \lg(n-1) \rfloor$, where $n > 1$.

n	*lr*-sequence
1	ε
2	*l*
3	*ll*
4	*lrl*
5	*llll*
6	*lllrl*
7	*lllrll*
8	*lllrrrl*
9	*lllrllrl*
10	*lrrrlrlll*
11	*lllrrrllrl*
12	*lllllllrrrl*
13	*lllllllrrrll*
14	*lllrrrllrlrll*
15	*lllrrrllrlllll*
16	*lllllllrrrrrrrl*
17	*lllllllrrrllllrl*
18	*lllrrrllrlrllllll*
19	*lllrrrllrlllrlllll*
20	*lllllllrrrllrrrlrll*
21	*lllllllrrrllrlrlllll*
22	*lllrrrrrrrllrlrllllll*
23	*lllrrrrrrrllrlllrlllll*
24	*lllllllrrrllrrrlrllllll*
25	*lllllllrrrllrrrlllrlllll*
26	*lrrrlrllllllrrlllrlrrrrrl*
27	*llrllrlrrrrrrllrrrlrllllll*
28	*lllllllrrrrrrrllrrrlrllllll*
29	*lllllllrrrrrrrllrrrlllrlllll*
30	*lllllllrrrrrrrllllrllrlrlllll*
31	*lllllllrrrrrrrllllrllrlllrllll*
32	*lllllllrrrllrrrlrllllllllllllll*
33	*lllllllllrrrlrlllllrrlllrlrrrrrl*
34	*lllrrrllrlrlllllrlrlrllrlllrrrrrl*
35	*lllrrrllrlllrllllrlrlrllrlllrrrrrl*
36	*lllllllrrrrrrrllrrrlrllllllllllllll*
37	*lllllllrrrrrrrllrrrlllrlllllllllllll*

Table 3.11: First lr-sequence using most colors for its length (UM)

colors	coloring
1	1
2	2 1
3	2 3 1
4	2 4 1 3
4	2 3 2 4 1
5	2 3 2 5 1 4
5	2 3 2 5 2 1 4
6	2 3 2 6 1 4 5 4
6	2 3 2 5 2 6 1 3 4
7	2 5 6 5 7 1 2 3 4 3
7	2 3 2 6 2 7 1 3 4 5 4
7	2 3 2 4 2 3 2 7 1 5 6 5
7	2 3 2 4 2 3 2 7 2 1 5 6 5
8	2 3 2 6 2 7 6 8 1 2 3 4 5 4
8	2 3 2 6 2 7 2 6 2 8 1 3 4 5 4
8	2 3 2 4 2 3 2 8 1 5 6 5 7 5 6 5
8	2 3 2 4 2 3 2 7 2 3 2 8 1 4 5 6 5
9	2 3 2 6 2 7 6 8 6 7 6 9 1 2 3 4 5 4
9	2 3 2 6 2 7 2 6 8 6 7 6 9 1 2 3 4 5 4
9	2 3 2 4 2 3 2 7 2 8 7 9 1 2 3 4 3 5 6 5
9	2 3 2 4 2 3 2 7 2 4 8 4 7 4 9 1 2 3 5 6 5
10	2 3 2 7 2 8 7 9 7 8 7 10 1 2 3 4 5 4 6 4 5 4
10	2 3 2 7 2 8 2 7 9 7 8 7 10 1 2 3 4 5 4 6 4 5 4
10	2 3 2 4 2 3 2 7 2 8 7 9 7 8 7 10 1 2 3 4 3 5 6 5
10	2 3 2 4 2 3 2 7 2 8 2 7 9 7 8 7 10 1 2 3 4 3 5 6 5
11	2 5 6 5 7 5 6 5 3 4 3 2 11 1 8 9 8 10 8 9 2 8 2 3 4 3
11	2 3 4 3 2 8 2 9 8 10 8 9 8 11 1 2 3 4 3 5 6 5 7 5 6 5 2
11	2 3 2 4 2 3 2 8 2 9 8 10 8 9 8 11 1 2 3 4 3 5 6 5 7 5 6 5
11	2 3 2 4 2 3 2 8 2 9 2 8 10 8 9 8 11 1 2 3 4 3 5 6 5 7 5 6 5
11	2 3 2 4 2 3 2 8 2 3 2 9 2 8 10 8 9 8 11 1 2 3 4 5 6 5 7 5 6 5
11	2 3 2 4 2 3 2 8 2 3 2 9 2 8 2 10 8 9 8 11 1 2 3 4 5 6 5 7 5 6 5
11	2 3 2 4 2 3 2 7 2 8 7 9 7 8 7 10 7 8 7 9 7 8 7 11 1 2 3 4 3 5 6 5
11	2 3 2 4 2 3 2 5 2 6 5 7 5 6 5 3 4 3 2 11 1 8 9 8 10 8 9 2 8 2 3 4 3
12	2 3 2 6 2 7 6 8 6 7 6 2 4 5 4 2 3 2 12 1 9 10 9 11 9 10 2 3 9 2 3 4 5 4
12	2 3 2 6 2 7 2 6 8 6 7 6 2 4 5 4 2 3 2 12 1 9 10 9 11 9 10 2 3 9 2 3 4 5 4
12	2 3 2 4 2 3 2 8 2 9 8 10 8 9 8 11 8 9 8 10 8 9 8 12 1 2 3 4 3 5 6 5 7 5 6 5
12	2 3 2 4 2 3 2 8 2 9 2 8 10 8 9 8 11 8 9 8 10 8 9 8 12 1 2 3 4 3 5 6 5 7 5 6 5

Table 3.12: Coloring of lr-sequence using most colors for its length (UM)

n	2^{n-1}	4	5	6	7	8	9	10	11
5	16	16							
6	32	16	16						
7	64	12	52						
8	128	8	100	20					
9	256		164	92					
10	512		188	320	4				
11	1024		172	812	40				
12	2048		144	1636	268				
13	4096		104	2916	1076				
14	8192		60	4640	3468	24			
15	16384		32	6724	9428	200			
16	32768		16	9056	22632	1064			
17	65536			11264	49692	4580			
18	131072			12864	101384	16784	40		
19	262144			13568	194492	53792	292		
20	524288			13140	354144	155552	1452		
21	1048576			11624	616476	413892	6584		
22	2097152			9448	1030416	1030936	26328	24	
23	4194304			7080	1660584	2431040	95436	164	
24	8388608			4936	2586292	5476736	319796	848	
25	16777216			3280	3893344	11872068	1004524	4000	
26	33554432			2080	5665196	24878724	2991572	16856	4
27	67108864			1240	7962648	50585080	8495320	64544	32

Table 3.13: Number of lr-sequences that use k colors (UM)

n	2^{n-1}	4	5	6	7	8
5	16	1.000000				
6	32	0.500000	0.500000			
7	64	0.187500	0.812500			
8	128	0.062500	0.781250	0.156250		
9	256		0.640625	0.359375		
10	512		0.367188	0.625000	0.007812	
11	1024		0.167969	0.792969	0.039062	
12	2048		0.070312	0.798828	0.130859	
13	4096		0.025391	0.711914	0.262695	
14	8192		0.007324	0.566406	0.423340	0.002930
15	16384		0.001953	0.410400	0.575439	0.012207
16	32768		0.000488	0.276367	0.690674	0.032471

Table 3.14: Ratio of lr-sequences using k colors (UM) up to 16

3.16 Relative insertion sequences

Another way to describe insertion of points, is to specify where the new point is inserted, related to the last inserted point. If the i-th point is to be inserted, and the last point (i.e., the $i-1$-th) is at position p, where $1 \leq p \leq i-1$, then the allowed relative positions for insertion range from $-p$ to -1 and from $+1$ to $+(i-p)$. Negative relative insertions correspond to insertions to the left of the last inserted point, and positive relative insertions correspond to insertions to the right of the last insertion point. As is the case for *lr*-sequences, the empty relative insertion sequence ε corresponds to only one point inserted. For example, the relative insertion sequence $[+1, -2, -1, +3]$ corresponds to insertion sequence 01003.

3.16.1 1-local relative insertions

We will restrict the relative insertion sequences to insertions which are not far from the last inserted point. The most restricted case is the one in which we only allow insertions adjacent to the last inserted point, either to the left or to the right, i.e., we only allow a -1 or a $+1$ insertion. We denote a -1 insertion by L and a $+1$ insertion by R.

In table 3.16, we show for each of the provided lengths, the lexicographically first 1-local relative sequence that gives the maximum use of colors for that length, when the unique maximum algorithm is used. In table 3.17, the corresponding coloring is given.

In table 3.18, we show for each of the provided lengths, the lexicographically first 1-local relative sequence that gives the maximum use of colors for that length, when the full greedy algorithm is used. In table 3.19, the corresponding coloring is given.

3.17 Color use of different sequences in UM

In table 3.20, for several kinds of sequences, we give the minimum length sequence that achieves the use of k colors, when the UM algorithm is used. The abbreviations used for kinds of sequences are:

- any: all possible sequences
- lr: all *lr*-sequences
- 1-loc: all 1-local sequences

- alt: alternating sequence: *lrlrlrlrlrlr*...
- btw2: insert between the last two points inserted sequence: 0 1 1 2 2 3 3 4 4 ...
- 0^*: insert always at the left end (all-zero sequence)
- root: the C_k coloring sequence of section 3.11

We denote by $\ell_S(k)$ the length of the shortest sequence in set S that uses k colors. Then, we just have the following closed formulae:

$$\ell_{0^*}(k) = 2^{k-1} \qquad \ell_{\text{root}}(k) = (k^2 - k + 2)/2$$

3.18 Color use of different lr-sequences in UM

In this section, we try to describe structured *lr*-sequences that achieve high use of colors. In these *lr*-sequences, colorings that resemble the coloring produced by the 0^n insertion sequence play an important role. If all colors from 1 to $q-1$ are seen from a point, then insertion of $n_T(q,r) := 2^{r-q+1} - 1$ points ($r \geq q \geq 1$) at that same position gives a coloring subsequence that resembles the coloring of the $0^{n_T(q,r)}$ insertion sequence, where each color is shifted up by $q-1$. We denote such a subsequence by $\langle q..r..q\rangle$, and call it a *tree subcoloring*. For example, $\langle 3..5..3\rangle = 3435343$, corresponds to the coloring 1213121, produced by insertion sequence 0^7.

If we insert all points of an *lr*-sequence in one direction, we can get the coloring:

$$1 \ k \ \langle 2..k-1..2\rangle\,,$$

where $k \geq 3$, which achieves k colors with $2^{k-2} + 1$ insertions. We call these sequences $0 - ch$, because there is no change in the direction of inserting points.

Another approach is to build two tree subcolorings on the two sides of the '1', to finally get a coloring:

$$\langle 2..j..2\rangle \ 1 \ k \ \langle j+1..k..j+1\rangle\,,$$

where $j \geq 2$, $k \geq j+1$, which achieves k colors with $2^{j-1} + 2k - j - 1$ insertions. Number of insertions is minimized, if we take $j = \lfloor k/2 \rfloor$ (we try to minimize $2^x + 2^y$, when the sum $x+y$ is constant). We call these sequences 1-ch, because they have one change in the direction of insertion (just from left to right).

We are now going to give a coloring produced by 2 changes in the direction of insertion, that achieves k colors with less insertions:

$$2, \langle j+1..k-1..j+1 \rangle, k, 1, 2, \langle 3..j..3 \rangle,$$

where $j \geq 3$, $k \geq j+2$, which achieves k colors with $4 + n_T(3, j) + n_T(j+1, k-1)$ insertions. We minimize number of insertions by taking $j = \lceil k/2 \rceil$. We call these sequences 2-ch.

In table 3.21, for several kinds of lr-sequences, we give the minimum length sequence that achieves the use of k colors, when the UM algorithm is used.

n	2^{n-1}	6	7	8	9	10	11
17	65536	0.171875	0.758240	0.069885			
18	131072	0.098145	0.773499	0.128052	0.000305		
19	262144	0.051758	0.741928	0.205200	0.001114		
20	524288	0.025063	0.675476	0.296692	0.002769		
21	1048576	0.011086	0.587917	0.394718	0.006279		
22	2097152	0.004505	0.491341	0.491589	0.012554	0.000011	
23	4194304	0.001688	0.395914	0.579605	0.022754	0.000039	
24	8388608	0.000588	0.308310	0.652878	0.038123	0.000101	
25	16777216	0.000196	0.232061	0.707630	0.059874	0.000238	
26	33554432	0.000062	0.168836	0.741444	0.089156	0.000502	0.000000
27	67108864	0.000018	0.118653	0.753776	0.126590	0.000962	0.000000

Table 3.15: Ratio of lr-sequences using k colors (UM) over 16

n	1-local sequence	used
1	ε	1
2	L	2
3	LR	3
4	LLL	3
5	$LLLR$	4
6	$LLLRL$	4
7	$LLLRRR$	5
8	$LLLRLLR$	5
9	$LRRRLRLL$	6
10	$LLLRRRLLR$	6
11	$LLLLLLLRRR$	6
12	$LLLLLLLRRRL$	6
13	$LLLRRRLLRLRL$	7
14	$LLLRRRLLRLLLL$	7
15	$LLLLLLLRRRRRRR$	7
16	$LLLLLLLRRRLLLLR$	7
17	$LLLRRRLLRLRLLLLL$	8
18	$LLLRRRLLRLLLRLLLL$	8
19	$LLLLLLLRRRLLRRRLRL$	8
20	$LLLLLLLRRRLLRLRLLLL$	8
21	$LLLRRRRRRRLLRLRLLLLL$	9
22	$LLLRRRRRRRLLRLLLRLLLL$	9
23	$LLLLLLLRRRLLRRRLRLLLLL$	9
24	$LLLLLLLRRRLLRRRLLLRLLLL$	9
25	$LRRRLRLLLLLLRRLLLRLRRRRR$	10
26	$LLRLLRLRRRRRRLLRRRLRLLLLL$	10
27	$LLLLLLLRRRRRRRLLRRRLRLLLLL$	10
28	$LLLLLLLRRRRRRRLLRRRLLLRLLLL$	10
29	$LLLLLLLRRRRRRRLLLLRLLRLRLLLL$	10
30	$LLLLLLLRRRRRRRLLLLRLLRLLLRLLL$	10
31	$LLLLLLLRRRLLRRRLRLLLLLLLLLLLL$	10
32	$LLLLLLLLLRRRLRLLLLLRRLLLRLRRRRR$	10
33	$LLLRRRLLRLRLLLLLRLRLRLLRLLLRRRRR$	11
34	$LLLRRRLLRLLLRLLLLRLRLRLLRLLLRRRRR$	11
35	$LLLLLLLRRRRRRRLLRRRLRLLLLLLLLLLLLL$	11
36	$LLLLLLLRRRRRRRLLRRRLLLRLLLLLLLLLLLL$	11

Table 3.16: First 1-local sequence using most colors for its length (UM)

coloring
1
2 1
2 3 1
3 1 2 1
3 4 1 2 1
3 1 4 1 2 1
3 4 3 5 1 2 1
3 2 5 1 4 1 2 1
2 3 2 1 6 4 5 4 1
3 4 3 2 6 1 5 1 2 1
4 5 4 6 1 2 1 3 1 2 1
4 5 4 1 6 1 2 1 3 1 2 1
3 4 3 2 1 7 5 6 1 5 1 2 1
3 4 3 2 7 1 5 1 6 1 5 1 2 1
4 5 4 6 4 5 4 7 1 2 1 3 1 2 1
4 5 4 3 7 1 2 1 6 1 2 1 3 1 2 1
3 4 3 2 1 8 5 6 5 7 5 6 1 5 1 2 1
3 4 3 2 1 8 5 6 5 7 5 1 6 1 5 1 2 1
4 5 4 2 3 2 1 8 6 7 1 6 1 2 1 3 1 2 1
4 5 4 2 1 8 3 6 3 7 3 1 6 1 2 1 3 1 2 1
3 4 3 5 3 4 3 2 1 9 6 7 6 8 6 7 1 6 1 2 1
3 4 3 5 3 4 3 2 1 9 6 7 6 8 6 1 7 1 6 1 2 1
4 5 4 2 3 2 1 9 6 7 6 8 6 7 1 6 1 2 1 3 1 2 1
4 5 4 2 3 2 1 9 6 7 6 8 6 1 7 1 6 1 2 1 3 1 2 1
2 3 2 1 7 1 8 7 9 7 8 7 10 1 2 3 2 4 5 4 6 4 5 4 1
1 4 5 4 6 4 5 4 2 3 2 1 10 7 8 7 9 7 8 1 7 1 2 3 2 1
4 5 4 6 4 5 4 2 3 2 1 10 7 8 7 9 7 8 1 7 1 2 1 3 1 2 1
4 5 4 6 4 5 4 2 3 2 1 10 7 8 7 9 7 1 8 1 7 1 2 1 3 1 2 1
4 5 4 6 4 5 4 3 2 1 10 7 8 7 9 7 1 8 1 2 1 7 1 2 1 3 1 2 1
4 5 4 6 4 5 4 3 2 1 10 7 8 7 9 1 7 1 8 1 2 1 7 1 2 1 3 1 2 1
4 5 4 2 3 2 1 10 6 7 6 8 6 7 6 9 6 7 6 8 6 7 1 6 1 2 1 3 1 2 1
2 3 2 1 7 1 8 7 9 7 8 7 10 1 2 3 2 4 5 4 6 4 5 1 4 1 2 1 3 1 2 1
3 4 3 2 1 8 2 1 9 8 10 8 9 8 11 1 2 1 3 4 3 1 5 6 5 7 5 6 1 5 1 2 1
3 4 3 2 1 8 2 1 9 8 10 8 9 8 11 1 2 1 3 4 3 1 5 6 5 7 5 1 6 1 5 1 2 1
4 5 4 6 4 5 4 2 3 2 1 11 7 8 7 9 7 8 7 10 7 8 7 9 7 8 1 7 1 2 1 3 1 2 1
4 5 4 6 4 5 4 2 3 2 1 11 7 8 7 9 7 8 7 10 7 8 7 9 7 1 8 1 7 1 2 1 3 1 2 1

Table 3.17: Coloring of first 1-local sequence using most colors for its length (UM)

n	1-local sequence	used
1	ε	1
2	L	2
3	LR	3
4	LLL	3
5	$LLLR$	4
6	$LLLRL$	4
7	$LLLRRR$	5
8	$LLLRRRL$	5
9	$LLLLLLLR$	5
10	$LLLLLLLRL$	5
11	$LLLLLLLRRR$	6
12	$LLLLLLLRRRL$	6
13	$LLLLLLLRRRLL$	6
14	$LLLLLLLRRRLLL$	6
15	$LLLLLLLRRRRRRR$	7
16	$LLLLLLLRRRRRRRL$	7
17	$LLLLLLLRRRRRRRLL$	7
18	$LLLLLLLRRRRRRRLLL$	7
19	$LRLLLLLRLRLLRRRRRR$	8
20	$LLRLLLLLRLRLLRRRRRR$	8
21	$LLLLLRLLLRLRLLRRRRRR$	8
22	$LLLLLLRLLLRLRLLRRRRRR$	8
23	$LLLLLLLLLLLLLLLRRRRRRR$	8
24	$LLLLLLLLLLLLLLLRRRRRRRL$	8
25	$LLLRRRRRRLRRRLRRRRLLLLLL$	9
26	$LLLRRRRRRLLRLLRLRLRRLLLLL$	9
27	$LLLRLLRRLLLRLLLRLLLLRRRRRR$	9

Table 3.18: First 1-local sequence using most colors for its length (greedy)

coloring
1
2 1
2 3 1
3 1 2 1
3 4 1 2 1
3 1 4 1 2 1
3 4 3 5 1 2 1
3 4 3 1 5 1 2 1
4 5 1 2 1 3 1 2 1
4 1 5 1 2 1 3 1 2 1
4 5 4 6 1 2 1 3 1 2 1
4 5 4 1 6 1 2 1 3 1 2 1
4 5 4 2 1 6 1 2 1 3 1 2 1
4 5 4 1 2 1 6 1 2 1 3 1 2 1
4 5 4 6 4 5 4 7 1 2 1 3 1 2 1
4 5 4 6 4 5 4 1 7 1 2 1 3 1 2 1
4 5 4 6 4 5 4 2 1 7 1 2 1 3 1 2 1
4 5 4 6 4 5 4 1 2 1 7 1 2 1 3 1 2 1
2 1 5 6 5 7 5 6 5 8 2 1 2 3 1 4 1 3 1
1 2 5 6 5 7 5 6 5 8 1 2 1 3 2 4 2 3 2 1
2 1 5 6 5 7 5 6 5 8 2 1 2 3 1 4 1 3 1 2 1
1 2 5 6 5 7 5 6 5 8 1 2 1 3 2 4 2 1 3 1 2 1
5 6 5 7 5 6 5 8 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1
5 6 5 7 5 6 5 1 8 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1
3 4 3 5 3 4 1 2 1 3 1 2 1 9 6 7 6 8 6 7 6 3 1 2 1
3 4 3 5 3 4 2 3 1 2 1 9 6 7 6 8 6 7 2 1 6 1 3 1 2 1
3 2 3 1 6 7 6 8 6 7 6 9 2 3 2 1 2 3 2 4 1 5 1 4 1 2 1

Table 3.19: Coloring of first 1-local sequence using most colors for its length (greedy)

k	any	lr	1-loc	alt	btw2	0*	root
1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2
3	3	3	3	3	3	4	4
4	4	4	5	4	5	8	7
5	6	6	7	6	9	16	11
6	8	8	9	10	12	32	16
7	10	10	13	13	17	64	22
8	12	14	17	18	29	128	29
9	14	18	21	30	45	256	37
10	?	22	25	46	81	512	46
11		26	33	82	105	..	56
12		34	?	106	177		67
13		?		178	245		79
14				246	323		92

Table 3.20: Minimum length sequence that uses k colors (UM)

k	any lr	0-ch	1-ch	2-ch
1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	5	4	4
5	6	9	6	6
6	8	17	8	8
7	10	33	12	10
8	14	65	16	14
9	18	129	24	18
10	22	257	32	26
11	26	513	48	34
12	34	..	64	50
13	?		96	66
14			128	98

Table 3.21: Minimum length lr-sequence that uses k colors (UM)

3.19 Analysis for first-fit greedy and UM

We consider uniquely occurring colors in a coloring. We considered the case for the greedy algorithm in 3.7.1. In this section, we study the case for both the greedy and the unique maximum algorithm in more detail.

In both the greedy and the UM algorithm, bigger uniquely occurring colors appear later (during the insertion) than smaller uniquely occurring colors, i.e., if u, u' are uniquely occurring colors with $u < u'$, then u' appeared after u. This is obvious from the minimality of the chosen color at each insertion for both algorithms.

In the following, we are going to use u_1, u_2, u_3, u_4 for the uniquely occurring colors, where:

$$u_1 < u_2 < u_3 < u_4.$$

Let's consider relative positions of uniquely occurring colors in a coloring. If we have two uniquely occurring colors, then, for both the greedy and the UM algorithm, we have the following possibilities:

$$\ldots u_1 \ldots u_2 \ldots$$
$$\ldots u_2 \ldots u_1 \ldots$$

The two relative positions are in fact symmetric. We remark, that in the following, all relative positions come in symmetric pairs.

Now, for a third uniquely occurring color to appear, if the greedy algorithm is used, this happens only if we have an insertion *between* u_1 and u_2. In all other cases, a color can be reused, so:

$$\ldots u_1 \ldots u_3 \ldots u_2 \ldots$$
$$\ldots u_2 \ldots u_3 \ldots u_1 \ldots$$

For a third uniquely occurring color in the UM algorithm, the above relative positions are also possible, but additionally, there can be two more cases, in which u_3 sees both u_1 and u_2 at the same direction; this means that u_3 has to be closer to u_1 than u_2, because $u_1 < u_2$. Therefore, we have the previous two relative positions plus two new ones:

$$\ldots u_1 \ldots u_3 \ldots u_2 \ldots$$
$$\ldots u_2 \ldots u_3 \ldots u_1 \ldots$$
$$\ldots u_2 \ldots u_1 \ldots u_3 \ldots$$
$$\ldots u_3 \ldots u_1 \ldots u_2 \ldots$$

Note however, that neither in the greedy nor in the UM algorithm increasing colors appear in increasing point positions, i.e., we can not have a situation like:

$$\dots u_1 \dots u_2 \dots u_3 \dots$$

(or the symmetric one).

Now, we already have seen that four uniquely occurring colors are not possible in the greedy algorithm (see lemma 36). For the UM algorithm, however, this is possible to happen. It can only happen if in the last two cases of three uniquely appearing colors for UM, i.e.:

$$\dots u_2 \dots u_1 \dots u_3 \dots$$
$$\dots u_3 \dots u_1 \dots u_2 \dots$$

the new point sees all uniquely occurring colors. For each of the above relative positions with three uniquely occurring colors, we have two possibilities:

$$\dots u_2 \dots u_1 \dots u_4 \dots u_3 \dots$$
$$\dots u_2 \dots u_4 \dots u_1 \dots u_3 \dots$$
$$\dots u_3 \dots u_1 \dots u_4 \dots u_2 \dots$$
$$\dots u_3 \dots u_4 \dots u_1 \dots u_2 \dots$$

Now, by inspection of the above relative positions, it can be easily seen that there can be no fifth uniquely occurring color with the UM algorithm: Any newly inserted point must be left or right from u_4, but then this newly inserted point does not see some uniquely appearing color, because u_4 has uniquely occurring colors to both directions. Therefore, we have the following lemma:

Lemma 41. *In any coloring by the UM algorithm there are at most four distinct colors with the following property: each of those colors occurs exactly once.*

3.20 Statistics for uniquely occurring colors

In table 3.22 we show how many uniquely occurring colors we have for sequences of each length, for lengths up to 13, for the greedy algorithm.

In table 3.23 we show how many uniquely occurring colors we have for sequences of each length, for lengths up to 14, for the UM algorithm.

n	1 unique	2 unique	3 unique
1	1	0	0
2	0	2	0
3	4	0	2
4	0	24	0
5	88	20	12
6	376	288	56
7	1 468	3 484	88
8	17 348	21 860	1 112
9	235 060	107 788	20 032
10	2 221 544	1 182 520	224 736
11	19 604 800	18 455 648	1 856 352
12	187 378 584	276 847 216	14 775 800
13	2 391 648 240	3 658 809 800	176 562 760

Table 3.22: Number of unique occurrences for greedy algorithm

n	1 unique	2 unique	3 unique	4 unique
1	1	0	0	0
2	0	2	0	0
3	2	0	4	0
4	0	20	0	4
5	32	16	72	0
6	96	376	184	64
7	552	2 820	1 260	408
8	7 568	14 092	16 952	1 708
9	64 772	136 272	141 672	20 164
10	492 364	1 751 844	1 092 744	291 848
11	5 000 896	19 817 256	11 880 548	3 218 100
12	69 109 940	209 303 084	168 269 416	32 319 160
13	1 008 959 956	2 416 125 020	2 430 550 536	371 385 288
14	14 076 837 648	34 140 836 472	33 395 758 308	5 564 858 772

Table 3.23: Number of unique occurrences for UM algorithm

3.21 The leveled algorithm

The leveled algorithm (la) is the best deterministic algorithm known so far. It was introduced in [27]. It is proved there that it uses $O(\log^2 n)$ colors in the worst case. It is also mentioned there that there are insertion sequences that force la to use $\Theta(\log^2 n)$ colors, but no such sequence is given. Here, we make some observations about the algorithm and give some bad insertion sequence, that uses $\Theta(\log^2 n)$ colors; we also prove that this sequence can be colored with $O(\log n)$ colors if the first-fit greedy algorithm is used on it.

We sketch how the leveled algorithm works, and for more details, we refer the interested reader to [27]: The algorithm colors points in levels. In every level, points are colored independently by using a substring of the optimal static coloring (1213121. . .). Therefore the number of colors per level is logarithmic on the number of points per level. In order to maintain this logarithmic number of colors, each level is extended only if points appear to the left or to the right of all points in the level. If a point appears between two points in the level, the point is deferred for coloring in a higher level. It can be proved (with the help of a binomial tree argument; see [19]) that the number of levels is logarithmic on n. Therefore, the algorithm uses $O(\log^2 n)$ colors.

3.22 A bad insertion sequence for the leveled algorithm

We describe an insertion sequence (in fact a sequence of insertion sequences) that forces the leveled algorithm to use $\Omega(\log^2 n)$ colors.

We will describe an insertion sequence of size $n = 2^k - 1$ which forces the leveled algorithm to use $k(k+1)/2 = \Omega(\log^2 n)$ colors. We will describe it with the help of the tree form of the optimal static coloring. Consider the optimal static coloring C^k for $n = 2^k - 1$. Define the following insertion sequence: Request the points that are colored with color 1 from left to right, then request the points that are colored with 2 from left to right, and so on, until you request the point colored with k. Each color class in C^k is colored in the same level of the leveled algorithm. We have a logarithmic color of levels (with respect to n) and it can be proved that the total number of colors is $\Omega(\log^2 n)$.

The coloring, in a tree-like form, of the above bad sequence by the leveled algorithm, for $n = 15 = 2^4 - 1$ is shown in figure 3.1.

However, the above bad sequence can be colored with a logarithmic number of colors. For example, the first-fit greedy algorithm can be used. The

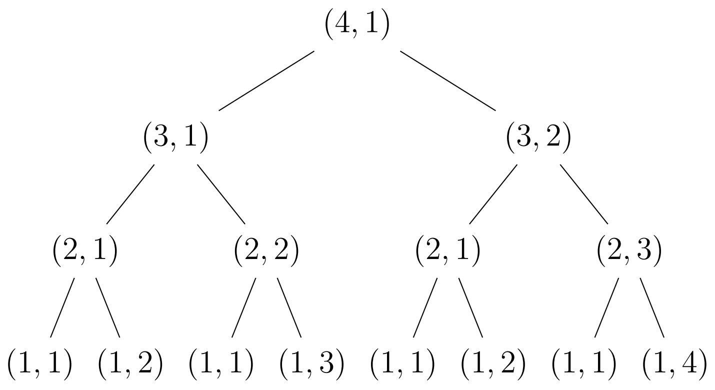


Figure 3.1: The bad sequence coloring by leveled ($n = 15$)

coloring, in a tree-like form, of the above bad sequence by the first-fit greedy algorithm, for $n = 15 = 2^4 - 1$, is shown in figure 3.2.

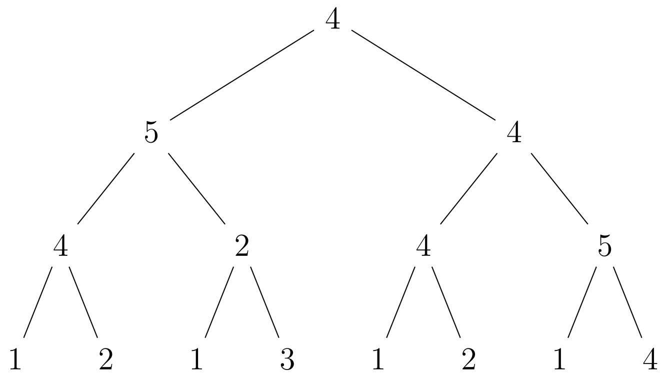


Figure 3.2: The bad sequence coloring by FF ($n = 15$)

3.23 Use of colors by the leveled algorithm for small lengths

In table 3.24 we show the best and worst use of colors for all sequences of the same length, for some small lengths.

In table 3.25 we show the first appearing insertion sequence (in the lexicographic order) which uses the maximum color for some lengths.

In table 3.26 we show for each length how many sequences use some number of colors (we also give the ratio in table 3.27).

n	best	worst
1	1	1
2	2	2
3	2	3
4	3	4
5	3	5
6	3	5
7	3	6
8	4	7
9	4	8
10	4	8
11	4	9
12	4	9
13	4	9

Table 3.24: Best and worst use of colors for level

n	sequence	coloring
3	0 0 1	2 3 1
4	0 0 2 1	2 4 1 3
5	0 0 2 1 3	2 4 1 5 3
6	0 0 0 0 1 3	3 4 1 5 2 1
7	0 0 0 0 1 4 3	3 4 1 6 2 5 1
8	0 0 0 0 4 1 4 3	3 5 1 7 2 6 1 4
9	0 0 0 0 4 2 1 6 5	3 6 1 5 2 8 1 7 4
10	0 0 0 0 0 5 2 1 6 5	1 6 3 5 1 8 2 7 1 4
11	0 0 0 0 0 5 3 1 7 3 7	1 6 3 8 1 5 2 9 1 7 4
12	0 0 0 0 0 0 6 3 1 7 3 7	2 6 1 8 3 5 1 9 2 7 1 4
13	0 0 0 0 0 0 0 0 3 1 7 3 7	4 6 1 8 2 5 1 9 3 7 1 2 1

Table 3.25: First sequence using most colors for its length for level

n	$n!$	3	4	5	6	7	8	9
4	24	20	4					
5	120	46	70	4				
6	720	84	484	152				
7	5040	84	2508	2292	156			
8	40320		10608	24792	4824	96		
9	362880		40142	221442	96584	4688	24	
10	3628800		150028	1794932	1536208	145696	1936	
11	39916800		561584	14080204	21578556	3593984	102056	416
12	479001600		1947888	109770416	286295216	76923664	4029312	35104
13	6227020800		5466716	851005228	3716580956	1521219420	130620924	2127556

Table 3.26: Sequences using numbers of colors for level

n	$n!$	3	4	5	6	7	8	9
4	24	0.833333	0.166667					
5	120	0.383333	0.583333	0.033333				
6	720	0.116667	0.672222	0.211111				
7	5040	0.016667	0.497619	0.454762	0.030952			
8	40320		0.263095	0.614881	0.119643	0.002381		
9	362880		0.110621	0.610235	0.266160	0.012919	0.000066	
10	3628800		0.041344	0.494635	0.423338	0.040150	0.000534	
11	39916800		0.014069	0.352739	0.540588	0.090037	0.002557	0.000010
12	479001600		0.004067	0.229165	0.597692	0.160592	0.008412	0.000073
13	6227020800		0.000878	0.136663	0.596847	0.244293	0.0209 76	0.000342

Table 3.27: Sequences ratio using numbers of colors for level

3.24 Unique occurrences of colors

In table 3.28 we show how many uniquely occurring colors we have for sequences of each length, for small lengths, for the greedy algorithm.

n	1	2	3	4	5	6	7	8
1	1							
2	0	2						
3	2	0	4					
4	0	20	0	4				
5	30	16	70	0	4			
6	84	288	196	152				
7	238	2166	1476	1004	156			
8	2700	10492	19400	5200	2432	96		
9	23446	89156	155452	75452	16366	2984	24	
10	213552	1003920	1221808	1012240	142544	33328	1408	
11	2696764	10287516	14186636	10011384	2468164	238512	27672	152
12	35334712	117409476	185619560	101229500	35977184	3149376	272688	9104
13	446095836	1655726022	2340827526	1270008064	449473540	62104044	2615864	169904

Table 3.28: Uniquely occurring colors for level

In table 3.29 we show the ratio of uniquely occurring colors we have for sequences of each length, for small lengths, for the greedy algorithm.

n	1	2	3	4	5	6	7	8
1	1.000000							
2	0.000000	1.000000						
3	0.333333	0.000000	0.666667					
4	0.000000	0.833333	0.000000	0.166667				
5	0.250000	0.133333	0.583333	0.000000	0.033333			
6	0.116667	0.400000	0.272222	0.211111				
7	0.047222	0.429762	0.292857	0.199206	0.030952			
8	0.066964	0.260218	0.481151	0.128968	0.060317	0.002381		
9	0.064611	0.245690	0.428384	0.207925	0.045100	0.008223	0.000066	
10	0.058849	0.276653	0.336698	0.278946	0.039281	0.009184	0.000388	
11	0.067560	0.257724	0.355405	0.250806	0.061833	0.005975	0.000693	0.000004
12	0.073767	0.245113	0.387513	0.211334	0.075109	0.006575	0.000569	0.000019
13	0.071639	0.265894	0.375915	0.203951	0.072181	0.009973	0.000420	0.000027

Table 3.29: Uniquely occurring colors for level ratio

3.25 The static case for rings

The following is true for any ring-coloring (not just for the ones produced by the greedy algorithm):

Proposition 42. *In every ring coloring of $n \geq 2$ points, there must be at least two uniquely occurring colors.*

Proof. If there is only one uniquely occurring color, given to some point p, consider the interval containing all points except p. Then this interval does not have the conflict-free property. □

In fact, for static ring CF-coloring, one can prove a bound similar to the $1 + \lfloor \lg n \rfloor$ for the intervals case. The main idea is to remove a vertex from the ring, color it with a unique color, and then color the remaining vertices with an optimal intervals coloring. This gives a coloring that uses $2 + \lfloor \lg (n-1) \rfloor$ colors.

The above result is tight:

Proposition 43. *A ring of size n can be colored with $2 + \lfloor \lg (n-1) \rfloor$ colors, but not less.*

Proof. Take out a unique colored vertex in the ring. The remaining $n-1$ vertices use $\lfloor \lg(n-1) \rfloor$ colors and constitute an intervals CF coloring, which is impossible since you need at least $1 + \lfloor \lg(n-1) \rfloor$ colors to color $n-1$ points with respect to intervals. □

3.26 The greedy algorithm for ring coloring

The greedy algorithm does the following: For each new point inserted, try to color it with the minimum color possible, such that the conflict-free property is maintained.

For each new point inserted, you only need to check the intervals in which this point lies and give the new point the minimum color that makes all those intervals have the uniquely occurring color property. This idea is helpful in proofs too. To prove that a coloring is conflict-free after an insertion of a point in an already conflict-free color sequence, you have just to check those new intervals.

3.26.1 Use of colors by greedy for small lengths

In table 3.30 we show the best and worst use of colors for all sequences of the same length, for some small lengths.

In table 3.31 we show the first appearing insertion sequence (in the lexicographic order) which uses the maximum color for some lengths.

In table 3.32 we show for each length how many sequences use some number of colors (we also give the ratio in table 3.33).

n	best	worst
1	1	1
2	2	2
3	3	3
4	3	3
5	4	4
6	4	4
7	4	5
8	4	5
9	5	6
10	5	6
11	5	7
12	5	7
13	?	?

Table 3.30: Best and worst use of colors for first-fit greedy

n	sequence	coloring
7	0 0 0 0 0 1 2	4 1 5 2 3 2 1
8	0 0 0 0 0 0 0 1	3 5 2 4 2 3 2 1
9	0 0 0 0 0 2 2 6 4	4 2 5 1 6 3 2 3 1
10	0 0 0 0 0 0 0 1 2 5	3 5 1 2 4 6 2 3 2 1
11	0 0 0 0 0 2 2 6 4 8 8	4 2 5 1 6 3 2 3 7 4 1
12	0 0 0 0 0 0 0 1 2 5 6 7	3 5 1 2 4 6 4 7 2 3 2 1
13?	?0 0 0 0 1 3 6 7 6 9 8 11 11 ?	?3 4 1 3 2 1 5 2 6 5 6 7 8 7

Table 3.31: First sequence using most colors for its length for greedy

n	$n!$	4	5	6	7	8
7	5040	3024	2016			
8	40320	6912	33408			
9	362880		351756	11124		
10	3628800		3165440	463360		
11	39916800		27364876	12544224	7700	
12	479001600		218942736	259484736	574128	

Table 3.32: Sequences using numbers of colors for first-fit greedy

3.26.2 Unique occurrences of colors

As is the case for the non-ring case, in the greedy algorithm there can not be more than three uniquely occurring colors. If there are three uniquely occurring colors, u_1, u_2, u_3, and a new point is inserted so that it is closer to u_1 and u_2 than u_3, then it can be legally colored with color u_3.

In table 3.34 we show how many uniquely occurring colors we have for sequences of each length, for small lengths, for the greedy algorithm.

In table 3.35 we show the ratio of uniquely occurring colors we have for sequences of each length, for small lengths, for the greedy algorithm.

3.27 The UM algorithm for ring coloring

The UM algorithm behaves very differently in the ring case, because now a lot more colors are visible, when inserting a new point; this leads in some cases in heavy color use.

3.27.1 Color use in UM

In table 3.36 we show the best and worst use of colors for some small lengths of insertion sequences.

n	$n!$	4	5	6	7	8
7	5040	0.600000	0.400000			
8	40320	0.171429	0.828571			
9	362880		0.969345	0.030655		
10	3628800		0.872310	0.127690		
11	39916800		0.685548	0.314259	0.000193	
12	479001600		0.457081	0.541720	0.001199	

Table 3.33: Sequences ratio using numbers of colors for first-fit greedy

n	1 unique	2 unique	3 unique
1	1	0	0
2	0	2	0
3	0	0	6
4	0	24	0
5	0	0	120
6	0	648	72
7	0	3 024	2 016
8	0	15 424	24 896
9	0	195 732	167 148
10	0	2 760 080	868 720
11	0	27 587 164	12 329 636
12	0	268 854 768	210 146 832

Table 3.34: Uniquely occurring colors for greedy

n	1 unique	2 unique	3 unique
1	1.000000	0.000000	0.000000
2	0.000000	1.000000	0.000000
3	0.000000	0.000000	1.000000
4	0.000000	1.000000	0.000000
5	0.000000	0.000000	1.000000
6	0.000000	0.900000	0.100000
7	0.000000	0.600000	0.400000
8	0.000000	0.382540	0.617460
9	0.000000	0.539385	0.460615
11	0.000000	0.691117	0.308883
12	0.000000	0.561282	0.438718

Table 3.35: Uniquely occurring colors for greedy ratio

n	best	worst
1	1	1
2	2	2
3	3	3
4	3	4
5	4	5
6	4	6
7	4	7
8	4	8
9	5	9
10	5	10
11	?	11
12	?	12
13	?	13

Table 3.36: Best and worst use of colors for UM

Chapter 4

Logarithmic online algorithms

4.1 Description of the $2\lg n$ algorithm

We present a recursive algorithm, that uses at most $2\lg n$ colors to color correctly online a sequence of *absolute* point positions (for the definition of input sequences of absolute positions, see section 3.2). We assume that the total number of points n is also given. See also [8, 9].

The core of the algorithm is a way to color any permutation π of absolute positions of points with $b = b(n)$ (we will give $b(n)$ soon) colors, such that the two maximum colors $u_b^L = b - 1$ and $u_b^R = b$ appear relatively close to the center of the coloring. Color $b-1$ appears to the left and color b appears to the right. Relatively close means that these unique colors both appear somewhere in the $n/2$ central points of the coloring. The value of $b(n)$ will be the smallest even integer such that $n \leq 2^{b/2}$, except if $n \leq 2$, when it is 2. Another way to put it is $b(n) = 2|\text{bin}(n-1)|$, where $|\text{bin}(x)|$ is the length of the binary representation of x:

$$|\text{bin}(x)| = \begin{cases} 1, & x = 0 \\ 1 + \lfloor \lg x \rfloor, & x > 0 \end{cases}$$

This achieves the approximately $2\lg n$ bound. The maximum number of points to color with b colors is $n^{\max}(b) = 2^{b/2}$.

More typically, given $l \leq n^{\max}(b(n))/2$, $r \leq n^{\max}(b(n))/2$, the following intervals are defined, in a $l + r = n$ length coloring:

- *leftmost interval*: from 1 to $\lfloor l/2 \rfloor$
- *left middle interval*: from $\lfloor l/2 \rfloor + 1$ to l
- *right middle interval*: from $l + 1$ to $l + \lceil r/2 \rceil$

- *rightmost interval*: from $l + \lceil r/2 \rceil + 1$ to $l + r$

(l and r can also be 0 or 1, in which case, some intervals are empty). The leftmost and the left middle intervals comprise the *left part* of the coloring and the right middle and rightmost intervals comprise the *right part* of the coloring. The size of the intervals is shown below:

$$\overbrace{\underbrace{\text{leftmost}}_{\lfloor l/2 \rfloor} \mid \underbrace{\text{left middle}}_{\lceil l/2 \rceil}}^{l} \mid \overbrace{\underbrace{\text{right middle}}_{\lceil r/2 \rceil} \mid \underbrace{\text{rightmost}}_{\lfloor r/2 \rfloor}}^{r}$$

We remark that there is a slight preference to the middle intervals (use of the ceiling, instead of the floor function). This will prove helpful later in the proof of correctness of the algorithm.

The algorithm colors any permutation π of length n, in a way such that at most b colors are used and if $l \geq 1$, then u_b^L occurs uniquely, somewhere in the left middle interval and if $r \geq 1$, then u_b^R occurs uniquely, somewhere in the right middle interval:

$$\overbrace{\underbrace{\ldots\ldots}_{\lfloor l/2 \rfloor} \mid \underbrace{\ldots u_b^L \ldots}_{\lceil l/2 \rceil}}^{l} \mid \overbrace{\underbrace{\ldots u_b^R \ldots}_{\lceil r/2 \rceil} \mid \underbrace{\ldots\ldots}_{\lfloor r/2 \rfloor}}^{r}$$

This is achieved by reserving color u_b^L for the first point that appears in the left middle interval and never reusing it, and by reserving u_b^R for the first point that appears in the right middle interval and never reusing it.

Initially, if the algorithm is given n points, in order to use $b = b(n)$ colors, it has to partition the points into two parts such that $l \leq n^{\max}(b)/2$ and $r \leq n^{\max}(b)/2$. There can be many ways to do it, but a standard way is to set $l = \lfloor n/2 \rfloor$ and $r = \lceil n/2 \rceil$, which tries to balance the sizes of the two parts, and if this is not possible (i.e., for odd n), it gives the slightest possible additional length to the right part.

The algorithm is shown in figure 4.1.

A crucial property of the algorithm is that the coloring of the two parts (left and right) are *independent* of each other. This is obvious in the way the main subroutine separates points of the left part from points of the right part. In each case, the color of each point is *only* dependent on the colors that have appeared in the respective (left or right) part.

The recursion (use of L, R subroutines) is shown in figure 4.2, where L_b means we use subroutine L with b colors (similarly for R_b).

```
coloring algorithm with $b$ colors:
    break the $n$ points in a left and a right part
      (of lengths $l = \lfloor n/2 \rfloor$, $r = \lceil n/2 \rceil$)
    for each point appearing online do
        if point is in left part
            color it with subroutine $L$ and $b$ colors
        if point is in right part
            color it with subroutine $R$ and $b$ colors

subroutine $L$ with $b$ colors:
    break the left part in a leftmost and a left middle interval
    if the point is in the leftmost interval
        color it with subroutine $L$ and $b - 2$ colors
    if the point is in the left middle interval
        if the left middle interval is empty
            give color $u_b^L = b - 1$ to the point
        else ($u_b^L$ has been used in this left middle interval)
            if point is to the left of $u_b^L$ in the left middle interval
                color it with subroutine $R$ and $b - 2$ colors
            if point is to the right of $u_b^L$ in the left middle interval
                color it with subroutine $L$ and $b - 2$ colors

subroutine $R$ with $b$ colors:
    break the right part in a right middle and a rightmost interval
    if the point is in the rightmost interval
        color it with subroutine $R$ and $b - 2$ colors
    if the point is in the right middle interval
        if the right middle interval is empty
            give color $u_b^R = b$ to the point
        else ($u_b^R$ has been used in this right middle interval)
            if point is to the left of $u_b^R$ in the right middle interval
                color it with subroutine $R$ and $b - 2$ colors
            if point is to the right of $u_b^R$ in the left middle interval
                color it with subroutine $L$ and $b - 2$ colors
```

Figure 4.1: The $2\lg n$ algorithm

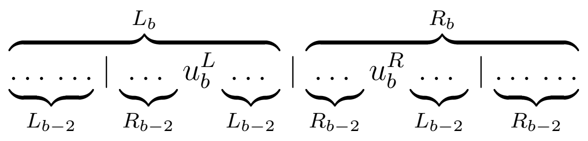


Figure 4.2: The recursion in a coloring

4.2 Correctness of the $2 \lg n$ algorithm

Since a coloring is just a concatenation of a part that is colored by an L_b algorithm, followed by a part that is colored by an R_b algorithm, it is enough to prove the following proposition.

Proposition 44. *If L_b is applied on any l points with $0 \leq l \leq n^{\max}(b)/2$, it gives a legal coloring C_b^L at all times (i.e., in the dynamic online sense).*

If R_b is applied on any r points with $0 \leq r \leq n^{\max}(b)/2$, it gives a legal coloring C_b^R at all times (i.e., in the dynamic online sense).

Moreover any partial C_b^L can be concatenated with any partial C_b^R to give a legal coloring $C_b^L \circ C_b^R$, at all times.

Proof. By induction on the number of colors used at most (i.e., b colors).

Base (b is at most 2): For $0 \leq l \leq 1$, $0 \leq r \leq 1$ the colorings are shown in table 4.1 and are correct.

$l \setminus r$	0	1
0	ε	2
1	1	1 2

Table 4.1: Colorings that use less than or equal to 2 colors

Inductive step: Assume the hypothesis is true for $b-2$ colors, then we will prove it is true for b colors. We have to consider four cases:

Case where none of u_b^L, u_b^R has appeared:

As long as no point in the two middle intervals has appeared, the coloring is done only in the leftmost and the rightmost interval, using colors up to $b-2$. By induction, the coloring up to that point is legal at all times, because each interval has size at most $n^{\max}(b)/4 = n^{\max}(b-2)$ (so $b-2$ colors suffice) and by the induction these colorings of the leftmost and rightmost interval can be concatenated to give a legal coloring.

Then, one point in the middle intervals is asked. W.l.o.g., assume it is in the left middle interval, so it gets color u_b^L. This color will remain unique, so for proving correctness, from now on, we have to consider only intervals not containing this u_b^L.

From now on, points in the left middle interval that are before the u_b^L-colored one are colored with subroutine R_{b-2} (see figure 4.2). The size of this interval where R_{b-2} is used is at most $n^{\max}(b)/4 = n^{\max}(b-2)$, so this interval is legally colored, by the inductive hyppothesis. Also, when combined with the leftmost interval which is colored by L_{b-2} and which also has size at most $n^{\max}(b)/4 = n^{\max}(b-2)$, those two intervals concatenated together, are always legally colored.

For points to the right of the u_b^L-colored one, as long as no point in the right middle interval has been requested, either the L_{b-2} subroutine is used for points in the left middle interval (see figure 4.2), or the R_{b-2} subroutine is used for points in the rightmost interval. These two intervals are both of length at most $n^{\max}(b)/4 = n^{\max}(b-2)$, so by induction, are always legally colored, and their concatenation is legally colored.

However, as soon as the first point in the right middle interval is requested, it gets color u_b^R and this color will remain unique. Again, this means that we only have to consider intervals that do not contain this u_b^R-colored point.

In the right middle interval and to the left of the u_b^R-colored point, the R_{b-2} subroutine is used, which, combined with the L_{b-2} colored interval to the right of u_b^L, by induction, gives a legal coloring. Also the right middle interval points to the right of the u_b^R-colored point, are colored with the L_{b-2} subroutine, which combined with the R_{b-2} colored rightmost interval, by induction, is always legally colored (check figure 4.2; all mentioned intervals are of length at most $n^{\max}(b)/4 = n^{max}(b-2)/2$). □

4.3 Slight improvements over the $2\lg n$ algorithm

The algorithm described above can color up to $2^{b/2}$ points if given b colors. Without any significant change to the algorithm, one can prove that if the algorithm is given b colors it also works correctly for all instances that have up to $2 \cdot 2^{b/2} - 2$ points (this is almost the double number of points).

One can also use low numbered colors for the central uniquely colored points and use colors based on the size of the subintervals that are to be colored in the recursion (i.e., not always $b-2$, but maybe less, because the subinterval can be small). This can also lead to some decrease of colors in some instances, but it does not give significant (if any) improvement in worst case instances.

4.4 Description of the oblivious adversary algorithm

We present a randomized algorithm that achieves expected logarithmic use of colors in coloring correctly online a sequence of insertions, which is chosen by an oblivious adversary. The adversary is oblivious in the following sense: The adversary has to commit on an insertion order of points (a permutation) before the algorithm starts execution, and can not change it during the course of the run of the algorithm; in other words, the adversary must choose the permutation without knowledge of the random choices of the algorithm. The algorithm does not need to know the absolute point positions in the final coloring neither the final number of points.

Recently, randomized algorithms that use $O(\log n)$ colors with high probability have been obtained ([17, 33, 10]). All of these algorithms assume the slightly weaker *oblivious* adversary model (see [14]), in which the adversary has to commit on a specific input sequence before revealing the first vertex to the algorithm without knowing the random bits that the algorithm is going to use and the *expected* number of colors is analyzed (this kind of analysis was introduced in [47]). The randomized model can be seen as a relaxation of the strict deterministic model: some power is taken from the adversary, or equivalently given to the algorithm, in order to use just a logarithmic number of colors. A similar algorithm to the one in [10] has been found independently by Olonetsky [43].

The algorithm is based on the generalized coloring algorithm for monotone range spaces described in [29]. The generalized algorithm finds in every step an independent set in the conflict graph of the problem, colors all points in the independent set with a new color, removes the colored points, recomputes the conflict graph, and iterates, until no points are left. Obviously, choosing a big independent set at each step means small color use at the end.

For conflict-free coloring for intervals, at every step, the conflict graph is three colorable: Each newly inserted point neighbors with at most two other points. Each color class gives an independent set. Our problem is that we must commit from the start on an independent set (we must color a point as soon as it appears in an online fashion). The size of an independent set can vary from 1 to $\lceil n/2 \rceil$.

Our algorithm uses any coloring strategy (as long as it gives a correct three coloring) and it chooses each of the three possible independent sets induced by the coloring with equal probability.

4.5 Analysis of the oblivious adversary algorithm

In this section, we fix a specific 3-coloring algorithm to be used at each stage of the algorithm. The coloring we get by applying the first-fit 3-coloring algorithm to permutation π is denoted by $\mathrm{ff}(\pi)$. We denote by a, b, c the colors used, in that priority, in a first fit algorithm.

Given a permutation π the use of colors by the randomized algorithm that uses first-fit to do the three coloring and chooses equiprobably among the colors of the three coloring is a random variable U_π. We intend to prove that for any permutation π, the expected value of the above variable, denoted by $\mathrm{E}[U_\pi]$ is bounded by $1 + \log_{3/2} n$, where $n = |\pi|$ is the length of the permutation.

We define random variable $U_\pi^+ = 1 + U_\pi$, to describe the use of one additional color in some non-empty round. By linearity of expectations: $\mathrm{E}[U_\pi^+] = 1 + \mathrm{E}[U_\pi]$.

If $|\pi| = 1$, then $\mathrm{ff}(\pi)$ uses one color, and $U_\pi = 1$. Of course, since the color of the independent set is randomly chosen it can be b or c, in which case that round of the coloring is empty, and thus does not contribute to the use of colors; if b or c is chosen, then the same permutation $\pi = 1$ is forwarded to the next round. Formally:

$$U_1 = \tfrac{1}{3} \cdot 1 + \tfrac{1}{3} U_1 + \tfrac{1}{3} U_1,$$

from which we get $U_1 = 1$.

If $|\pi| > 1$, then $\mathrm{ff}(\pi)$ uses either two (a and b) or three (a, b, and c) colors. We denote by $\pi \setminus q$ the permutation that is to be colored in the next round, if the independent set colored with color $q \in \{a, b, c\}$ is chosen in $\mathrm{ff}(\pi)$. If independent sets with color a, b, c have sizes x, y, z respectively, then:

- If $\mathrm{ff}(p)$ uses 3 colors, $|\pi \setminus a| = y + z$, $|\pi \setminus b| = x + z$, $|\pi \setminus c| = x + y$, and $|\pi| = x + y + z$. Thus $|\pi \setminus a| + |\pi \setminus b| + |\pi \setminus c| = 2|\pi|$.
- If $\mathrm{ff}(p)$ uses 2 colors, $|\pi \setminus a| = y$, $|\pi \setminus b| = x$, and $|\pi| = x + y$. Thus $|\pi \setminus a| + |\pi \setminus b| = |\pi|$. We also have $|\pi \setminus c| = 0$, and thus $\pi \setminus c = \pi$, but that is not a problem.

Since colors a, b, c are chosen equiprobably we have:

$$U_\pi = \tfrac{1}{3}(U_\pi^a + U_\pi^b + U_\pi^c)$$

where:

$$U_\pi^q = \begin{cases} U_{\pi-q}^+, & \mathrm{ff}(\pi) \text{ uses color } q \\ U_\pi, & \mathrm{ff}(\pi) \text{ does not use color } q \end{cases}$$

From the above, in case, where color c is not used in ff(p), we have:

$$U_\pi = \tfrac{1}{3}(U^+_{\pi\setminus a} + U^+_{\pi\setminus b} + U_\pi) \quad \text{or} \quad U_\pi = \tfrac{1}{2}(U^+_{\pi\setminus a} + U^+_{\pi\setminus b}).$$

Therefore, in general:

$$U_\pi = \begin{cases} \tfrac{1}{3}(U^+_{\pi\setminus a} + U^+_{\pi\setminus b} + U^+_{\pi\setminus c}), & \text{ff}(\pi) \text{ uses 3 colors} \\ \tfrac{1}{2}(U^+_{\pi\setminus a} + U^+_{\pi\setminus b}), & \text{ff}(\pi) \text{ uses 2 colors} \end{cases}$$

and by linearity of expectations:

$$\mathrm{E}[U_\pi] = \begin{cases} 1 + \tfrac{1}{3}(\mathrm{E}[U_{\pi\setminus a}] + \mathrm{E}[U_{\pi\setminus b}] + \mathrm{E}[U_{\pi\setminus c}]), & \text{ff}(\pi) \text{ uses 3 colors} \\ 1 + \tfrac{1}{2}(\mathrm{E}[U_{\pi\setminus a}] + \mathrm{E}[U_{\pi\setminus b}]), & \text{ff}(\pi) \text{ uses 2 colors} \end{cases}$$

Proposition 45. *If* $|\pi| = n$, $\mathrm{E}[U_\pi] \leq 1 + \log_{3/2} n$.

Proof. By induction. Base $(n = 1)$: $\mathrm{E}[U_1] = 1$. For $n > 1$, by the induction hypothesis, we consider the cases (a) when ff(p) uses 3 colors:

$$\begin{aligned} E[U_\pi] &\leq 1 + \tfrac{1}{3}(1 + \log_{3/2}|\pi \setminus a| + 1 + \log_{3/2}|\pi \setminus b| + 1 + \log_{3/2}|\pi \setminus c|) \\ &= 2 + \tfrac{1}{3}\log_{3/2}|\pi \setminus a||\pi \setminus b||\pi \setminus c| \\ &= 2 + \log_{3/2}\sqrt[3]{|\pi \setminus a||\pi \setminus b||\pi \setminus c|} \\ &\leq 2 + \log_{3/2}\frac{|\pi \setminus a| + |\pi \setminus b| + |\pi \setminus c|}{3} \\ &= 2 + \log_{3/2}\frac{2|\pi|}{3} = 1 + \log_{3/2} n, \end{aligned}$$

and (b) when ff(p) uses 2 colors:

$$\begin{aligned} E[U_\pi] &\leq 1 + \tfrac{1}{2}(1 + \log_{3/2}|\pi \setminus a| + 1 + \log_{3/2}|\pi \setminus b|) \\ &= 2 + \tfrac{1}{2}\log_{3/2}|\pi \setminus a||\pi \setminus b| \\ &= 2 + \log_{3/2}\sqrt{|\pi \setminus a||\pi \setminus b|} \\ &\leq 2 + \log_{3/2}\frac{|\pi \setminus a| + |\pi \setminus b|}{2} \\ &= 2 + \log_{3/2}\frac{|\pi|}{2} < 2 + \log_{3/2}\frac{|\pi|}{3/2} = 1 + \log_{3/2} n, \end{aligned}$$

where in both cases we used the geometric/arithmetic mean inequality:

$$(a_1 \cdots a_p)^{1/p} \leq (a_1 + \dots a_p)/p.$$

□

Epilogue

We investigated a special coloring problem (conflict-free coloring) for graphs and hypergraphs, that has applications in frequency assignment in cellular networks. We considered related problems, mainly ordered coloring, which have applications in parallel optimization of several tasks (like Cholesky factorization), circuit design, assembly of parts, and database query optimization.

There are still gaps between lower and upper bounds. The most important open problem is narrowing the gap between lower and upper bound in the relative positions model for conflict-free coloring with respect to intervals: $\Omega(\log n)$, and $O(\log^2 n)$, respectively, which are a logarithmic factor apart.

In the case of all-intervals, static uses $O(\log n)$ and the best known online deterministic algorithm $O(\log^2 n)$ colors, but this logarithmic factor 'jump' is not a result of the online model, because it occurs just between the absolute positions model and the fully online (relative positions) model.

In the dynamic online absolute positions setting, it is natural for the algorithm to know the total number n of points that will be inserted from the start. The triples algorithm exploits that knowledge to achieve $O(\log n)$ colorings. An open problem in the absolute positions model is to maintain an $O(\log k)$ coloring after the first k points have been inserted for all k with $0 < k \leq n$.

Another open problem is the worst case (or even the average case) performance of the Unique Max algorithm. No sharp asymptotic bounds are known. The best known upper bound we proved is linear, whereas the best known lower bound is $\Omega(\sqrt{n})$. In this thesis, we have provided an extensive analysis of the UM algorithm, that might shed some light on this open problem.

Bibliography

[1] Karen I. Aardal, Stan P.M. van Hoesel, Arie M.C.A Koster, Carlo Mannino, and Antonio Sassano. Models and solutions techniques for frequency assignment problems. Technical Report ZIB Report 01-40, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Germany, 2001.

[2] Alok Aggarwal, Amotz Bar-Noy, Don Coppersmith, Rajiv Ramaswami, Baruch Schieber, and Madhu Sudan. Efficient routing and scheduling algorithms for optical networks. In *SODA*, pages 412–423, 1994.

[3] Alok Aggarwal, Amotz Bar-Noy, Don Coppersmith, Rajiv Ramaswami, Baruch Schieber, and Madhu Sudan. Efficient routing in optical networks. *Journal of the ACM*, 43(6):973–1001, 1996.

[4] Jean-Paul Allouche and Jeffrey Shallit. *Automatic sequences: theory, applications, generalizations*. Cambridge University Press, 2003.

[5] Noga Alon and Shakhar Smorodinsky. Conflict-free colorings of shallow discs. In *Proceedings of the 22nd Annual ACM Symposium on Computational Geometry (SoCG)*, pages 41–43, 2006.

[6] Noga Alon and Joel Spencer. *The probabilistic method*. Wiley, 2nd edition, 2000.

[7] Amotz Bar-Noy, Panagiotis Cheilaris, Michael Lampis, and Stathis Zachos. Conflict-free coloring graphs and other related problems. Manuscript, 2006.

[8] Amotz Bar-Noy, Panagiotis Cheilaris, and Shakhar Smorodinsky. Conflict-free coloring for intervals: from offline to online. In *Proceedings of the 18th annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 128–137, 2006.

[9] Amotz Bar-Noy, Panagiotis Cheilaris, and Shakhar Smorodinsky. Deterministic conflict-free coloring for intervals: from offline to online. Under review, 2006.

[10] Amotz Bar-Noy, Panagiotis Cheilaris, and Shakhar Smorodinsky. Randomized online conflict-free coloring for hypergraphs. Manuscript, 2006.

[11] János Barát and David R. Wood. Notes on nonrepetitive graph colouring, 2005.

[12] Hans L. Bodlaender, John R. Gilbert, Hjálmtyr Hafsteinsson, and Ton Kloks. Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *Journal of Algorithms*, 18(2):238–255, 1995.

[13] Béla Bollobás. *Modern Graph Theory.* Springer, 1998.

[14] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis.* Cambridge University Press, 1998.

[15] Boštjan Brešar, Jarosław Grytczuk, Sandi Klavžar, Stanisław Niwczyk, and Iztok Peterin. Nonrepetitive colorings of trees, 2003.

[16] Panagiotis Cheilaris, Ersnt Specker, and Stathis Zachos. Neochromatica. Manuscript, 2006.

[17] Ke Chen. How to play a coloring game against a color-blind adversary. In *Proceedings of the 22nd Annual ACM Symposium on Computational Geometry (SoCG)*, pages 44–51, 2006.

[18] B. N. Clark, C. J. Colburn, and D. S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86:165–177, 1990.

[19] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Cliff Stein. *Introduction to Algorithms.* MIT Press, 2nd edition, 2001.

[20] James D. Currie. There are ternary circular square-free words of length n for $n \geq 18$. *Electronic Journal of Combinatorics*, 9(1), 2002.

[21] Reinhard Diestel. *Graph Theory.* Springer, 3rd edition, 2005.

[22] Iain S. Duff. Parallel implementation of multifrontal schemes. *Parallel Computing*, 3(3):193–204, 1986.

[23] Khaled Elbassioni and Nabil H. Mustafa. Conflict-free colorings of rectangles ranges. In *Proceedings of the 23rd International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 254–263, 2006.

[24] Thomas Erlebach, Aris Pagourtzis, Katerina Potika, and Stamatis Stefanakos. Resource allocation problems in multifiber WDM tree networks. In Hans L. Bodlaender, editor, *WG*, volume 2880 of *Lecture Notes in Computer Science*, pages 218–229. Springer, 2003.

[25] Thomas Erlebach, Aris Pagourtzis, Katerina Potika, and Stamatis Stefanakos. Resource allocation problems in multifiber WDM tree networks. In *Proceedings of 29th Workshop on Graph Theoretic Concepts in Computer Science (WG 2003), LNCS 2880*, pages 218–229, 2003.

[26] Guy Even, Zvi Lotker, Dana Ron, and Shakhar Smorodinsky. Conflict-free colorings of simple geometric regions with applications to frequency assignment in cellular networks. *SIAM Journal on Computing*, 33:94–136, 2003. Also in *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2002.

[27] Amos Fiat, Meital Levy, Jiří Matoušek, Elchanan Mossel, János Pach, Micha Sharir, Shakhar Smorodinsky, Uli Wagner, and Emo Welzl. Online conflict-free coloring for intervals. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 545–554, 2005.

[28] Alan George, Michael T. Heath, Esmond G. Ng, and Joseph W. H. Liu. Symbolic cholesky factorization on a local-memory multiprocessor. *Parallel Computing*, 5(1-2):85–95, 1987.

[29] Sariel Har-Peled and Shakhar Smorodinsky. Conflict-free coloring of points and simple regions in the plane. *Discrete and Computational Geometry*, 34:47–70, 2005.

[30] Stratis Ioannidis, Christos Nomikos, Aris Pagourtzis, and Stathis Zachos. Routing and wavelength assignment in generalized WDM tree networks of bounded degree. In Panayiotis Bozanis and Elias N. Houstis, editors, *Panhellenic Conference on Informatics*, volume 3746 of *Lecture Notes in Computer Science*, pages 57–67. Springer, 2005.

[31] Ananth V. Iyer, H. Ronald Ratliff, and Gopalakrishanan Vijayan. Optimal node ranking of trees. *Information Processing Letters*, 28:225–229, 1988.

[32] Camille Jordan. Sur les assemblages de lignes. *Journal für die Reine und Angewandte Mathematik*, 70:185–190, 1869.

[33] Haim Kaplan and Micha Sharir. Online CF coloring for halfplanes, congruent disks, and axis-parallel rectangles. Manuscript, 2004.

[34] Meir Katchalski, William McCuaig, and Suzanne Seager. Ordered colourings. *Discrete Mathematics*, 142:141–154, 1995.

[35] Charles E. Leiserson. Area-efficient graph layouts (for VLSI). In *FOCS*, pages 270–281. IEEE, 1980.

[36] Joseph W. H. Liu. Computational models and task scheduling for parallel sparse cholesky factorization. *Parallel Computing*, 3(4):327–342, 1986.

[37] Joseph W. H. Liu. Reordering sparse matrices for parallel elimination. *Parallel Computing*, 11(1):73–91, 1989.

[38] Donna Crystal Llewellyn, Craig A. Tovey, and Michael A. Trick. Erratum: Local optimization on graphs. *Discrete Applied Mathematics*, 46(1):93–94, 1993.

[39] Christos Nomikos, Aris Pagourtzis, Katerina Potika, and Stathis Zachos. Fiber cost reduction and wavelength minimization in multifiber WDM networks. In Nikolas Mitrou, Kimon P. Kontovasilis, George N. Rouskas, Ilias Iliadis, and Lazaros F. Merakos, editors, *NETWORKING*, volume 3042 of *Lecture Notes in Computer Science*, pages 150–161. Springer, 2004.

[40] Christos Nomikos, Aris Pagourtzis, Katerina Potika, and Stathis Zachos. Routing and wavelength assignment in multifiber WDM networks with non-uniform fiber cost. *Computer Networks*, 50(1):1–14, 2006.

[41] Christos Nomikos, Aris Pagourtzis, and Stathis Zachos. Routing and path multicoloring. *Inf. Process. Lett.*, 80(5):249–256, 2001.

[42] Christos Nomikos, Aris Pagourtzis, and Stathis Zachos. Satisfying a maximum number of pre-routed requests in all-optical rings. *Computer Networks*, 42(1):55–63, 2003.

[43] Svetlana Olonetsky. Personal communication, 2006.

[44] János Pach and Géza Tóth. Conflict free colorings. In *Discrete and Computational Geometry, The Goodman-Pollack Festschrift*, pages 665–671. Springer Verlag, 2003.

[45] Katerina Potika. Maximizing the number of connections in multifiber WDM chain, ring and star networks. In Raouf Boutaba, Kevin C. Almeroth, Ramón Puigjaner, Sherman X. Shen, and James P. Black, editors, *NETWORKING*, volume 3462 of *Lecture Notes in Computer Science*, pages 1465–1470. Springer, 2005.

[46] E. Prouhet. Mémoire sur quelques relations entre les puissances des nombres. *Comptes Rendus de l'Académie des Sciences, Paris, Série I*, 33:225, 1851.

[47] Prabhakar Raghavan and Marc Snir. Memory versus randomization in on-line algorithms. In *Proceedings of the 16th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 687–703, 1989.

[48] Robert G. Rieper and Melkamu Zeleke. Valleyless sequences. arXiv e-print archive, math.CO/0005180, 17th May 2000, http://arxiv.org/abs/math/0005180.

[49] Arunabha Sen, Haiyong Deng, and Sumanta Guha. On a graph partition problem with application to VLSI layout. *Information Processing Letters*, 43(2):87–94, 1992.

[50] Shakhar Smorodinsky. *Combinatorial Problems in Computational Geometry*. PhD thesis, School of Computer Science, Tel-Aviv University, 2003.

[51] Shakhar Smorodinsky. On the chromatic number of some geometric hypergraphs. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2006.

[52] Axel Thue. Über unendliche Zeichenreihen. *Norske vid. Selsk. Skr. Mat. Nat. Kl*, 7:1–22, 1906.