Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Εργαστηριο Λογικης και Επιστημης Υπολογιστων (Co.Re.Lab.)

Αλγόριθμοι Χωρίς Μνήμη για το Γενικευμένο Πρόβλημα των k-Εξυπηρετητών σε Ομοιόμορφους Μετρικούς Χώρους

Διπλωματικη Εργασια

του

Δημητρίου Ι. Χρήστου

Αθήνα, Ιούλιος 2020

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Λογικής και Επιστήμης Υπολογιστών (Co.Re.Lab.)

Αλγόριθμοι Χωρίς Μνήμη για το Γενικευμένο Πρόβλημα των k-Εξυπηρετητών σε Ομοιόμορφους Μετρικούς Χώρους

Διπλωματικη Εργασια

του

Δημητρίου Ι. Χρήστου

Επιβλέπων: Δημήτριος Φωτάκης
Αναπληρωτής Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 13η Ιουλίου 2020.

....	....	....
Δημήτριος Φωτάκης	Αριστείδης Παγουρτζής	Αντώνιος Συμβώνης
Αναπ. Καθηγητής Ε.Μ.Π.	Αναπ. Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2020

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Λογικής και Επιστήμης Υπολογιστών (Co.Re.Lab.)

Υπεύθυνη Δήλωση

Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας, και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης, βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου.

....
Δημήτριος Ι. Χρήστου
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Το γενικευμένο πρόβλημα των k-εξυπηρετητών είναι μία σημαντική γενίκευση του προβλήματος των k-εξυπηρετητών, το οποίο αποτελεί ένα από τα πιο θεμελιώδη προβλήματα της θεωρίας των online αλγορίθμων. Αν και το πρόβλημα των k-εξυπηρετητών έχει μελετηθεί και κατανοηθεί εκτενώς τις τελευταίες δεκαετίες, το γενικευμένο πρόβλημα των k-εξυπηρετητών έχει γίνει κατανοητό σε πολύ μικρότερο βαθμό, με τα περισσότερα σχετικά αποτελέσματα να αφορούν μόνο ειδικές περιπτώσεις μετρικών χώρων.

Στόχος της παρούσας διπλωματικής εργασίας είναι η μελέτη του γενικευμένου προβλήματος των k-εξυπηρετητών σε ομοιόμορφους μετρικούς χώρους. Παρακινούμενοι από την αντιστάθμιση ανάμεσα στον λόγο ανταγωνιστικότητας και στην υπολογιστική αποτελεσματικότητα των αλγορίθμων, μελετάμε την ισχύ των αλγορίθμων χωρίς μνήμη και δείχνουμε σφιχτά φράγματα της τάξης $\Theta(k!)$ για τον λόγο ανταγωνιστικότητάς τους. Συγκεκριμένα, δείχνουμε οτι ο *Αρμονικός Αλγόριθμος* πετυχαίνει αυτόν τον λόγο και αποδεικνύουμε αντίστοιχα κάτω φράγματα. Αυτό το αποτέλεσμα βελτιώνει το $\approx 2^{2^k}$ διπλά εκθετικό φράγμα των Chiplunkar και Vishwanathan για την πιο γενική περίπτωση των ομοιόμορφων μετρικών χώρων με διαφορετικά βάρη.

Λέξεις - Κλειδιά: Online Αλγόριθμοι, Γενικευμένο Πρόβλημα k-Εξυπηρετητών, Αλγόριθμοι Χωρίς Μνήμη

Abstract

The generalized k-server problem is a far-reaching extension of the k-server problem, one of the most fundamental problems in the theory of online algorithms. While the k-server problem has been extensively studied and understood over the last decades, the generalized k-server problem is much less understood and most results are restricted to special types of metric spaces.

In this thesis, we consider the generalized k-server problem on uniform metrics. Motivated by the trade-offs between the competitive ratio and computational efficiency, we study the power of memoryless algorithms and show tight bounds of $\Theta(k!)$ on their competitive ratio. In particular we show that the *Harmonic Algorithm* achieves this competitive ratio and provide matching lower bounds. This improves the $\approx 2^{2^k}$ doubly-exponential bound of Chiplunkar and Vishwanathan for the more general setting of uniform metrics with different weights.

Keywords: Online Algorithms, Generalized k-Server Problem, Memoryless Algorithms

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντα και καθηγητή μου κ. Δημήτρη Φωτάκη για την απίστευτη υποστήριξη και καθοδήγησή του σε όλη τη διάρκεια της προπτυχιακής μου πορείας, για τις ευκαιρίες που μου έδωσε και κυρίως για το ενδιαφέρον που μου έδειξε τόσο σε ακαδημαϊκό όσο και σε προσωπικό επίπεδο. Στη συνέχεια, θα ήθελα να ευχαριστήσω τον Γρηγόρη Κουμούτσο, ο οποίος επέβλεψε προσωπικά την εκπόνηση της παρούσας διπλωματικής εργασίας και υπήρξε καταπληκτικός δάσκαλος για εμένα τον τελευταίο ένα χρόνο. Χωρίς το ενδιαφέρον του, την καθοδήγησή του και τη συμβολή του, η παρούσα διπλωματική εργασία δεν θα είχε ολοκληρωθεί.

Θα ήθελα επίσης να ευχαριστήσω θερμά την ομάδα από το εργαστήριο LIP6 του Sorbonne University με την οποία είχα την χαρά να συνεργαστώ πέρυσι και φέτος στα πλαίσια της πρακτικής μου, τον Ευριπίδη Μπάμπη, τον Bruno Escoffier και τον Nguyen Kim Thang, για την καθοδήγησή τους και τη συμμετοχή τους στα πρώτα μου ερευνητικά βήματα.

Και κυρίως, θα ήθελα να ευχαριστήσω την οικογένεια μου, τους γονείς μου και τις αδερφές μου, για την ανιδιοτελή αγάπη που μου έχουν προσφέρει όλα αυτά τα χρόνια, τις θυσίες τους και την υποστήριξή τους σε όλες μου τις επιλογές.

Δημήτριος Χρήστου,
Αθήνα, Ιούλιος 2020

Acknowledgments

First of all, I would like to thank my supervisor and teacher Prof. Dimitris Fotakis for his massive support and guidance during the last five years, the opportunities he gave me and most importantly the interest he has shown in me, both academically and personally.

Next, I would like to express my gratitude towards Grigorios Koumoutsos, who personally supervised this thesis and was an excellent teacher to me over the last year. Without his guidance and his participation in this work, this thesis would not be completed.

I would also like to thank the group from the LIP6 lab in Sorbonne University, that supervised my internships last year and this year as well: Evripidis Bampis, Bruno Escoffier and Nguyen Kim Thang. Thank you for your guidance and being a huge part of my first steps in the field of academic research.

But most importantly, I would like to express my gratitude towards my family, my parents and my sisters, for their selfless love throughout all this years, their sacrifices and their unlimited support.

Dimitrios Christou,
Athens, July 2020

Εκτεταμένη Ελληνική Περίληψη

Το βασικό σκέλος της παρούσας διπλωματικής εργασίας έχει αποδωθεί στην αγγλική γλώσσα, κυρίως για λόγους προσβασιμότητας. Σε αυτό το κομμάτι της, συνοψίζουμε το περιεχόμενό της, δίνοντας έμφαση στους βασικούς ορισμούς, τις μεθοδολογίες και τα θεωρήματα, αλλά χωρίς τις μαθηματικές αποδείξεις. Η δομή της ενότητας αυτής είναι σε ένα προς ένα αντιστοίχηση με το (αγγλικό) περιεχόμενο της διπλωματικής εργασίας.

Εισαγωγή

Σε ένα κλασσικό πρόβλημα βελτιστοποίησης, μας δίνετε μία συγκεκριμένη είσοδος και ο στόχος μας είναι να βρούμε μία βέλτιστη λύση για την είσοδο αυτή. Ωστόσο, σε πολλές πραγματικές εφαρμογές, η υπόθεση ότι μας δίνετε κατευθείαν ολόκληρη η είσοδος δεν είναι ρεαλιστική. Τις περισσότερες φορές, χρειάζεται να λύνουμε προβλήματα βελτιστοποίησης παίρνοντας αποφάσεις χωρίς να έχουμε πλήρη γνώση της εισόδου. Αναφερόμαστε σε τέτοιου είδους προβλήματα με τον όρο online προβλήματα, και για να τα επιλύσουμε χρειάζεται να σχεδιάσουμε online αλγορίθμους οι οποίοι παίρνουν αποφάσεις χωρίς να γνωρίζουν το μέλλον. Ο βασικός μας στόχος είναι να σχεδιάσουμε online αλγορίθμους για τους οποίους έχουμε εγγυήσεις που αφορούν την απόδοσή τους σε σχέση με την βέλτιστη offline λύση η οποία έχει ολόκληρη την είσοδο στη διάθεσή της από την αρχή.

Το πρόβλημα των k-εξυπηρετητών (k-server problem) αποτελεί ένα από τα βασικότερα και πιο εκτενώς μελετημένα προβλήματα στην θεωρία των online αλγορίθμων. Σε αυτό το πρόβλημα, μας δίνεται ένας μετρικός χώρος με n σημεία και k κινούμενοι εξυπηρετητές που βρίσκονται σε κάποια από τα σημεία του μετρικού χώρου. Σε κάθε βήμα, λαμβάνουμε ένα αίτημα που αντιστοιχεί σε ένα από τα σημεία του μετρικού χώρου, και για να το εξυπηρετήσουμε πρέπει να μετακινήσουμε κάποιον από τους εξυπηρετητές σε αυτό το σημείο. Στόχος μας είναι να ελαχιστοποιήσουμε την συνολική απόσταση που διανύουν οι εξυπηρετητές.

Το πρόβλημα των k-εξυπηρετητών αποτελεί γενίκευση πολλών online προβλημάτων, με πιο σημαντικό το πρόβλημα των σελίδων (paging problem) το οποίο αποτελεί την ειδική περίπτωση όπου ο μετρικός χώρος είναι ομοιόμορφος. Το πρόβλημα των σελίδων έχει επιλυθεί πλήρως: το competitive ratio του είναι k για ντετερμινιστικούς αλγορίθμους και $H_k = \Theta(logk)$ για τυχαίους αλγορίθμους. Ωστόσο, σε γενικούς μετρικούς χώρους η κατανόησή μας για το πρόβλημα των k-εξυπηρετητών είναι ακόμα ελλιπής: οι Κουτσουπιάς και Παπαδημητρίου απέδειξαν την ύπαρξη ενός ντετερμινιστικού αλγορίθμου με competitive ratio

$2k-1$, ενώ γνωρίζουμε ότι το βέλτιστο εφικτό είναι τουλάχιστον k. Πολλοί πιστεύουν ότι πράγματι το σωστό competitive ratio για το πρόβλημα είναι k, και η θέση αυτή είναι γνωστή στη βιβλιογραφία ως η εικασία των k-εξυπηρετητών. Για τυχαίους αλγορίθμους, πιστεύετε ότι υπάρχει αλγόριθμος με competitive ratio $O(logk)$ αλλά παρά τις αμέτρητες προσπάθειες τα τελευταία χρόνια, η εικασία αυτή παραμένει ανοιχτή.

Σε μία προσπάθεια για καλύτερη κατανόηση του προβλήματος των k-εξυπηρετητών, έχουν προταθεί διάφορες γενικεύσεις του όπως για παράδειγμα το πρόβλημα των k-εξυπηρετητών με βάρη και το CNN πρόβλημα. Με στόχο να συνοψίσουν τέτοιες γενικεύσεις του προβλήματος κάτω από ένα ενιαίο πρόβλημα, οι Κουτσουπίας και Taylor πρότειναν το γενικευμένο πρόβλημα των k-εξυπηρετητών, όπου ο κάθε εξυπηρετητής s_i βρίσκεται σε ένα ξεχωριστό μετρικό χώρο M_i και κάθε αίτημα είναι της μορφής $(r_1, r_2, \ldots, r_k)$ με $r_i \in M_i$: για να εξυπηρετηθεί, πρέπει τουλάχιστον ένας εξυπηρετητής s_i να μεταβεί στο σημείο r_i του μετρικού χώρου M_i.

Το γενικευμένο πρόβλημα των k-εξυπηρετητών έχει μία πολύ πλουσιότερη δομή από το κλασσικό πρόβλημα των k-εξυπηρετητών, και για γενικούς μετρικούς χώρους δεν έχει βρεθεί ακόμα αλγόριθμος με competitive ratio που να εξαρτάται μόνο από το k. Γενικά, τα μόνα αποτελέσματα της βιβλιογραφίας αφορούν τις ακόλουθες δύο περιπτώσεις:

- Ομοιόμορφες Μετρικές: Όλοι οι μετρικοί χώροι $M_1, \ldots, M_k$ είναι ομοιόμορφοι, με την ίδια απόσταση (πχ 1) ανάμεσα σε δύο σημεία τους.
- Ομοιόμορφες Μετρικές με βάρη: Όλοι οι μετρικοί χώροι $M_1, \ldots, M_k$ είναι ομοιόμορφοι αλλά έχουν διαφορετικά βάρη: για να κουνήσεις τον εξυπηρετητή της μετρικής M_i το κόστος είναι w_i.

Αν και μοιάζουν παρόμοιες, οι παραπάνω ειδικές περιπτώσεις διαφέρουν σημαντικά μεταξύ τους. Για ντετερμινιστικούς αλγορίθμους, οι Bansal et.al. [11] ανέλυσαν έναν αλγόριθμο με competitive ratio $O(k2^k)$ σε ομοιόμορφους μετρικούς χώρους, ενώ στην περίπτωση των ομοιόμορφων μετρικών χώρων με βάρη κάθε ντετερμινιστκός αλγόριθμος έχει competitive ratio τουλάχιστο $2^{2^{k-4}}$ [10].

Ένα βασικό μειονέκτημα πολλών αλγορίθμων που πετυχαίνουν το καλύτερο competitive ratio για κάποια προβλήματα είναι ότι είναι υπολογιστικά μη-αποδοτικοί. Για παράδειγμα, ο Work Function Algorithm των Κουτσουπιάς και Παπαδημητρίου που πετυχαίνει το (σχεδόν βέλτιστο) $2k-1$ competitive ratio για το πρόβλημα των k-εξυπηρετητών, τρέχει πάνω σε όλες τις πιθανές διατάξεις των εξυπηρετητών, που είναι $\binom{n}{k}$.

Η παρατήρηση αυτή οδηγεί στην μελέτη της σχέσης ανάμεσα στην αποδοτικότητα και το competitive ratio ενός online αλγορίθμου. Ένα πρώτο βήμα σε αυτή την κατεύθυνση είναι η μελέτη αλγορίθμων χωρίς μνήμη, δηλαδή αλγορίθμων που παίρνουν αποφάσεις βασισμένοι μόνο στην τρέχουσα διάταξή τους και το τρέχον αίτημα που λαμβάνουν.

Η απόδοση των αλγορίθμων χωρίς μνήμη έχει μελετηθεί εκτενώς για το πρόβλημα των k-εξυπηρετητών [16, 55], με βασικότερο αλγόρθμο τον Αρμονικό Αλγόριθμο ο οποίος μετακινεί τον κάθε εξυπηρετητή με πιθανότητα αντιστρόφως ανάλογη της απόστασής του από το αίτημα. Εικάζεται ότι το competitive ratio του Αρμονικού Αλγορίθμου είναι $O(k^2)$, ωστόσο το πρόβλημα αυτό παραμένει ανοιχτό μέχρι και σήμερα.

Στα πλαίσια αυτής της διπλωματικής εργασίας, θα μελετήσουμε αλγόριθμους χωρίς μνήμη για το πρόβλημα των k-εξυπηρετητών σε ομοιόμορφους μετρικούς χώρους. Ένας τέτοιος αλγόριθμος λαμβάνει ένα αίτημα της μορφής $r = (r_1, \dots, r_k)$ και αποφασίζει σε ποιο μετρικό χώρο θα κουνηθεί σύμφωνα με μία κατανομή $p_1, p_2, \dots, p_k$, όπου p_i είναι η πιθανότητα να κουνήσει τον εξυπηρετητή του μετρικού χώρου M_i στο σημείο αιτήματος r_i. Σε αυτό το πλαίσιο, ο Αρμονικός Αλγόριθμος είναι ο αλγόριθμος που διαλέγει μία μετρική ομοιόμορφα, δηλαδή έχει πιθανότητες $p_i = \frac{1}{k}$ για κάθε μετρικό χώρο M_i.

Πρόσφατα, οι Chiplunkar και Vishnawathan [28] μελέτησαν αλγόριθμους χωρίς μνήμη για το γενικευμένο πρόβλημα των k-εξυπηρετητών σε ομοιόμορφους μετρικούς χώρους με βάρη, και πήραν διπλά εκθετικά φράγματα για το competitive ratio τέτοιων αλγορίθμων. Εφόσον το πρόβλημα με βάρη φαίνεται να είναι εκθετικά δυσκολότερο από το πρόβλημα χωρίς, είναι φυσικό να περιμένουμε βελτίωση αυτών των φραγμάτων όταν θα μελετήσουμε το πρόβλημα σε ομοιόμορφους χώρους.

Συνεισφορά. Σε αυτήν την πτυχιακή εργασία μελετάμε την δύναμη των αλγορίθμων χωρίς μνήμη για το γενικευμένο πρόβλημα των k-εξυπηρετητών σε ομοιόμορφους μετρικούς χώρους και υπολογίζουμε ακριβώς το competitive ratio αυτών των αλγορίθμων.

Αρχικά στρέφουμε την προσοχή μας στο μοντέλο του adaptive online αντιπάλου και αποδεικνύουμε τα ακόλουθα:

Θεώρημα 1. *Ο Αρμονικός Αλγόριθμος έχει competitive ratio $k \cdot \alpha_k$ ενάντια σε adaptive online αντιπάλους, όπου α_k είναι η λύση της αναδρομής $\alpha_k = 1 + (k-1)\alpha_{k-1}$ με $\alpha_1 = 1$.*

Θεώρημα 2. *Κάθε αλγόριθμος χωρίς μνήμη έχει competitive ratio τουλάχιστον $k \cdot \alpha_k$ σε μετρικούς χώρους με $n \geq 3$ σημεία ενάντια σε adaptive online αντιπάλους.*

Θεώρημα 3. *Κάθε αλγόριθμος χωρίς μνήμη έχει competitive ratio της τάξης $\Theta(2^k)$ σε μετρικούς χώρους με $n = 2$ σημεία ενάντια σε adaptive online αντιπάλους.*

Τα παραπάνω τρία θεωρήματα προσδιορίζουν πλήρως την δύναμη των αλγορίθμων χωρίς μνήμη στο μοντέλο του adaptive online αντιπάλου. Επίσης, από αυτά προκύπτει άμεσα ότι ο Αρμονικός Αλγόριθμος είναι ο βέλτιστος αλγόριθμος χωρίς μνήμη για το γενικευμένο πρόβλημα των k-εξυπηρετητών σε ομοιόμορφους μετρικούς χώρους.

Στη συνέχεια εστιάζουμε στο μοντέλο του oblivious αντιπάλου και αποδεικνύουμε τα ακόλουθα:

Θεώρημα 4. *Ο Αρμονικός Αλγόριθμος έχει competitive ratio $k\alpha_k$ σε μετρικούς χώρους με $n \geq 2k+1$ σημεία ενάντια σε oblivious αντιπάλους.*

Θεώρημα 5. *Ο Αρμονικός Αλγόριθμος έχει competitive ratio της τάξης $\Theta(2^k)$ σε μετρικούς χώρους με $n = 2$ σημεία ενάντια σε oblivious αντιπάλους.*

Ουσιαστικά, τα παραπάνω θεωρήματα δείχνουν ότι η απόδοση του Αρμονικού Αλγορίθμου δεν επηρεάζεται από το είδος του αντιπάλου.

Συνοψίζοντας, τα αποτελέσματά μας δείχνουν ότι σε ομοιόμορφους μετρικούς χώρους υπάρχει εκθετική βελτίωση σε σχέση με ομοιόμορφους μετρικούς χώρους με βάρη. Δείξαμε επίσης ότι παρά τα συνήθη αποτελέσματα για ομοιόμορφες εκδοχές online προβλημάτων, ένας αλγόριθμος χωρίς μνήμη για το γενικευμένο πρόβλημα των k-εξυπηρετητών σε ομοιόμορφους μετρικούς χώρους δεν μπορεί να πετύχει το (ντετερμινιστικό) competitive ratio $2^{O(k)}$.

Οργάνωση. Στο Κεφάλαιο 2, δίνουμε τους βασικούς ορισμούς στο πεδίο των online αλγορίθμων και αναφέρουμε κάποια από τα βασικότερα και πιο μελετημένα προβλήματα της περιοχής, δίνοντας έμφαση σε κάποιες από τις τεχνικές που χρησιμοποιούνται. Στη συνέχεια, στο Κεφάλαιο 3 δίνουμε το απαραίτητο υπόβαθρο στις αλυσίδες Markov που απαιτείται για την ανάλυση των αποτελεσμάτων μας.

Έχοντας τελειώσει με τα προαπαιτούμενα, προχωράμε στο Κεφάλαιο 4 όπου παρουσιάζουμε την πρωτότυπη δουλειά αυτής της πτυχιακής εργασίας και αποδεικνύουμε τα πέντε βασικά Θεωρήματα που αναφέραμε νωρίτερα. Τέλος, στο Κεφάλαιο 5 συνοψίζουμε τα αποτελέσματά μας και συζητάμε πιθανές κατευθύνσεις για έρευνα με στόχο την καλύτερη κατανόηση του γενικευμένου προβλήματος των k-εξυπηρετητών.

Online Αλγόριθμοι

Σε ένα online πρόβλημα βελτιστοποίησης, η είσοδος σ χωρίζεται σε μια ακολουθία από αιτήματα $\sigma = \sigma_1, \dots, \sigma_m$. Κάθε φορά που λαμβάνουμε ένα αίτημα σ_t, ένας online αλγόριθμος πρέπει να λάβει κάποιες αποφάσεις χωρίς να γνωρίζει τα μελλοντικά αιτήματα $\sigma_{t+1}, \dots, \sigma_m$. Ένας αλγόριθμος που διαβάζει όλη την είσοδο σ και μετά παράγει μία λύση καλείται offline αλγόριθμος.

Στα πλαίσια αυτής της διπλωματικής, εστιάζουμε σε online προβλήματα ελαχιστοποίησης P. Έστω I το σύνολο όλων των πιθανών εισόδων για ένα πρόβλημα P. Για κάθε είσοδο $\sigma \in I$, έστω $ALG(\sigma)$ το κόστος ενός online αλγορίθμου και $OPT(\sigma)$ το κόστος του βέλτιστου offline αλγορίθμου.

Ορισμός 1. *Θα λέμε ότι ένας ντετερμινιστικός online αλγόριθμος ALG για ένα πρόβλημα ελαχιστοποίησης P είναι c-competitive, αν υπάρχει σταθερά α τέτοια ώστε για κάθε είσοδο $\sigma \in I$ να ισχύει*

$$ALG(\sigma) \leq c \cdot OPT(\sigma) + \alpha.$$

Αν $\alpha = 0$ τότε λέμε ότι ο ALG είναι αυστηρά c-competitive. Το competitive ratio ενός αλγορίθμου ALG είναι το ελάχιστο c για το οποίο είναι c-competitive. Το competitive ratio του online προβλήματος P είναι το ελάχιστο c για το οποίο υπάρχει c-competitive αλγόριθμος για το P.

Εναλλακτικά, μπορούμε να σκεφτούμε ότι υπάρχει ένας ισχυρός αντίπαλος ο οποίος παρατηρεί τον αλγόριθμο και κατασκευάζει την είσοδο του προβλήματος με στόχο να μεγιστοποιήσει τον λόγο του κόστους του αλγορίθμου προς το κόστος με το οποίο λύνει αυτός την είσοδο. Αν ο αλγόριθμος είναι τυχαίος, τότε τίθεται το ζήτημα της πόσης πληροφορίας έχει στην διάθεσή του ο αντίπαλος όταν φτιάχνει την είσοδο. Έχουν προταθεί διάφορα μοντέλα αντιπάλων [14] για την ανάλυση τυχαίων online αλγορίθμων:

1. **Oblivious Αντίπαλος:** Σε αυτό το μοντέλο, ο αντίπαλος γνωρίζει την περιγραφή του αλγορίθμου αλλά όχι τις τυχαίες επιλογές του και επιλύει offline το πρόβλημα.

2. **Adaptive Online Αντίπαλος:** Σε αυτό το μοντέλο, ο αντίπαλος γνωρίζει την περιγραφή του αλγορίθμου και τις τυχαίες επιλογές του και επιλύει online το πρόβλημα.

3. **Adaptive Offline Αντίπαλος:** Σε αυτό το μοντέλο, ο αντίπαλος γνωρίζει την περιγραφή του αλγορίθμου και τις τυχαίες επιλογές του και επιλύει offline το πρόβλημα.

Τυπικά, ο oblivious αντίπαλος είναι ο ευκολότερος, έπειτα ο adaptive online και τέλος ο adaptive offline. Στην ανάλυση online αλγορίθμων συνήθως θεωρούμε το μοντέλο του adaptive online αντιπάλου. Σημειώνουμε ότι ο adaptive offline αντίπαλος είναι τόσο ισχυρός που αποδεικνύεται πως η τυχαιότητα δεν βοηθάει ενάντιά του. Σε κάθε μία από αυτές τις περιπτώσεις, ως 'κόστος' του αλγορίθμου θεωρούμε την αναμενόμενη τιμή της τυχαίας μεταβλητής που δίνει το κόστος του.

Μέθοδος Δυναμικού. Ένα από τα πιο συνήθη εργαλεία για την ανάλυση online αλγορίθμων είναι η μέθοδος της συνάρτησης δυναμικού. Μία συνάρτηση δυναμικού αντιστοιχεί έναν παραγματικό αριθμό σε κάθε πιθανό συνδιασμό διάταξης ανάμεσα στον αλγόριθμο και τον αντίπαλο. Διαισθητικά, βολεύει να σκεφτόμαστε αυτό το δυναμικό ως μια μετρική απόστασης ανάμεσα στον αλγόριθμο και τον αντίπαλο.

Η επιλογή ενός κατάλληλου δυναμικού αποτελεί πολύ αυθαίρετη διαδικασία και επαφίεται στην βαθιά κατανόηση της δομής του εκάστοτε προβλήματος. Η γενική ιδέα είναι πως αν επιλέξουμε ένα κατάλληλο (για το πρόβλημα) δυναμικό και αποδείξουμε ότι τηρεί κάποιες ιδιότητες, τότε μπορούμε εύκολα να δείξουμε άνω φράγματα για το competitive ratio του αλγορίθμου μας. Μία από τις βασικότερες τεχνικές της μεθόδου συνάρτησης δυναμικού (που χρησιμοποιούμε και εμείς σε αυτήν την εργασία) είναι η μέθοδος των εναλλασσόμενων κινήσεων. Σύμφωνα με αυτήν την μέθοδο, έστω ότι μια συνάρτηση δυναμικού που ικανοποιεί τις ακόλουθες ιδιότητες:

1. Αν σε ένα αίτημα μετακινείται μόνο ο αντίπαλος και πληρώνει x, τότε το δυναμικό αυξάνεται το πολύ κατά $c \cdot x$.

2. Αν σε ένα αίτημα μετακινείται μόνο ο αλγόριθμος και πληρώνει x, τότε το δυναμικό μειώνεται τουλάχιστον κατά x.

3. Το δυναμικό είναι φραγμένο από κάποιες αυθαίρετες τιμές.

Τότε, μπορούμε άμεσα να αποδείξουμε ότι ο αλγόριθμος είναι c-competitive ενάντια σε έναν adaptive online αντίπαλο.

Γνωστά Online Προβλήματα. Το πρόβλημα των σελίδων (paging problem) [71] αποτελεί ένα από τα πρώτα online προβλήματα που μελετήθηκαν στη βιβλιογραφία. Σε αυτό το πρόβλημα, έχουμε μία μνήμη cache μεγέθους k σελίδων και ένα σύνολο από n σελίδες. Σε κάθε αίτημα, μας ζητείται μία από τις n σελίδες και αν δεν βρίσκεται μέσα στην cache πρέπει να βγάλουμε κάποια άλλη σελίδα από την cache ώστε να την αντικαταστήσει. Στόχος μας είναι να ελαχιστοποιήσουμε τον συνολικό αριθμό εξώσεων από την μνήμη cache. Γενικά, το πρόβλημα των σελίδων έχει κατανοηθεί πλήρως, και γνωρίζουμε ότι για ντετερμινιστικούς αλγορίθμους το competitive ratio του προβλήματος είναι $O(k)$, ενώ για τυχαίους αλγορίθμους είναι $O(logk)$. Σημειώνεται ότι γνωρίζουμε αλγορίθμους που πετυχαίνουν ακριβώς αυτά τα φράγματα.

Αρχικά, κάθε online πρόβλημα αναλυόταν με εξειδικευμένες τεχνικές που δεν μπορούσαν να επεκταθούν σε άλλα προβλήματα. Σε μία προσπάθεια για ενοποίηση του πεδίου των online αλγορίθμων, οι Borodin et al. [17] πρότειναν το πρόβλημα των Metrical Task Systems (MTS). Σε αυτό το πρόβλημα έχουμε έναν μετρικό χώρο με N σημεία και σε κάθε αίτημα μας δίνεται ένα διάνυσμα κόστους πάνω στα σημεία αυτά. Στη συνέχεια πρέπει να αποφασίσουμε σε πιο σημείο θα ικανοποιήσουμε το αίτημα, πληρώνοντας το κόστος του σημείου αυτού συν την απόσταση για να πάμε σε αυτό το σημείο από το τρέχον σημείο μας. Στόχος μας είναι να ελαχιστοποιήσουμε το συνολικό μας κόστος. Γνωρίζουμε ότι για το MTS, το ντετερμινιστικό competitive ratio είναι $2N-1$ [17] και το competitive ratio τυχαίων αλγορίθμων είναι $O(log^2 NloglogN)$ [37] και $\Omega(logN)$ [17]. Συνήθως, ενδιαφερόμαστε για στιγμιότυπα του MTS με κάποια συγκεκριμένη δομή, και στόχος μας είναι να δούμε αν μπορούμε να πετύχουμε competitive ratio που δεν εξαρτάται από τον αριθμό των σημείων N.

Το πρόβλημα των k-εξυπηρετητών αποτελεί την πιο σημαντική ειδική περίπτωση του MTS. Σε αυτό το πρόβλημα, μας δίνεται ένας μετρικός χώρος με n σημεία και k κινούμενοι εξυπηρετητές που βρίσκονται σε κάποια από τα σημεία του μετρικού χώρου. Σε κάθε βήμα, λαμβάνουμε ένα αίτημα που αντιστοιχεί σε ένα από τα σημεία του μετρικού χώρου, και για να το εξυπηρετήσουμε πρέπει να μετακινήσουμε κάποιον από τους εξυπηρετητές σε αυτό το σημείο. Στόχος μας είναι να ελαχιστοποιήσουμε την συνολική απόσταση που διανύουν οι εξυπηρετητές. Είναι εύκολο να δούμε ότι το πρόβλημα των k-εξυπηρετητών αντιστοιχεί στο MTS με ένα σημείο για κάθε πιθανή διάταξη των k εξυπηρετητών ($N = n^k$), αποστάσεις ανάμεσα στα σημεία ίσες με την ελάχιστη απόσταση ανάμεσα στις διατάξεις και αιτήματα με τιμή 0 στις διατάξεις που ικανοποιούν το αίτημα και τιμή ∞ στις υπόλοιπες. Για ντετερμινιστικούς αλγορίθμους γνωρίζουμε ότι το competitive ratio είναι τουλάχιστον k και το πολύ $2k-1$. Για τυχαίους αλγόριθμους, γνωρίζουμε ότι είναι τουλάχιστον $O(logk)$ και το πολύ $O(log^6 k)$ [60]. Το πρόβλημα αυτό δεν έχει επιλυθεί ακόμα πλήρως και αποτελεί αντικείμενο τεράστιου ενδιαφέροντος στον χώρο των online αλγορίθμων.

Τέλος, το γενικευμένο πρόβλημα των k-εξυπηρετητών αποτελεί μία γενίκευση του προβλήματος των k-εξυπηρετητών και μία ειδική περίπτωση του MTS. Ο ορισμός του προβλήματος έχει ήδη δοθεί παραπάνω. Το πρόβλημα αυτό αποτελεί ένα από τα πιο δυσνόητα προβλήματα του πεδίου και ελάχιστα αποτελέσματα είναι γνωστά, μόνο για συγκεκριμένες περιπτώσεις. Σε ομοιόμορφους μετρικούς χώρους γνωρίζουμε ότι το competitive ratio είναι τουλάχιστον $2^k - 1$ και το πολύ $O(k2^k)$ για ντετερμινιστικούς αλγορίθμους και τουλάχιστον $O(k)$ και το πολύ $O(k^2 logk)$ για τυχαίους αλγορίθμους. Σε ότι αφορά αλγορίθμους χωρίς μνήμη για το πρόβλημα, το μόνο γνωστό αποτέλεσμα είναι ότι σε ομοιόμορφους μετρικούς χώρους με βάρη, το competitive ratio αλγορίθμων χωρίς μνήμη είναι διπλά εκθετικό. Το τελευταίο αποτέλεσμα οφείλεται στην δουλειά των Chiplunkar και Vishwanathan [28].

Τυχαίοι Περίπατοι

Ένα από τα βασικά μαθηματικά εργαλεία που θα χρησιμοποιήσουμε στην ανάλυσή μας είναι οι τυχαίοι περίπατοι και γενικά οι ιδιότητες των στοχαστικών ανελίξεων. Μία στοχαστική ανέλιξη είναι μια ακολουθία από τυχαίες μεταβλητές που εξελίσσεται με κάποιο τυχαίο αλλά προδιαγεγραμμένο τρόπο. Σε αυτήν την εργασία εστιάζουμε σε στοχαστικές ανελίξεις που εξελίσσονται σε διακριτό χρόνο (βήματα $\{0, 1, \dots\}$) και λαμβάνουν τιμές από ένα διακριτό και πεπερασμένο σύνολο $\mathcal{X}$. Συγκεκριμένα, μας ενδιαφέρουν οι αλυσίδες Markov, που ορίζονται ως:

Ορισμός 2. *Μία αλυσίδα Markov είναι μία στοχαστική ανέλιξη X με διακριτό χώρο καταστάσεων $\mathcal{X}$ και διακριτό σύνολο δεικτών $T = \{0, 1, \dots\}$ η οποία ικανοποιεί την ακόλουθη Μαρκοβιανή ιδιότητα:*

$$\mathbb{P}(X_t = s | X_0 = x_0, X_1 = x_1, \dots, X_{t-1} = x_{t-1}) = \mathbb{P}(X_t = s | X_{t-1} = x_{t-1})$$

για κάθε $t \geq 1$ και $s, x_0, \dots, x_{t-1} \in \mathcal{X}$.

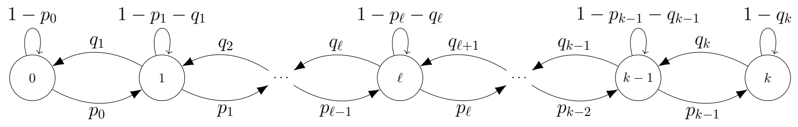
Ο πίνακας $P = [p_{ij}]$ όπου $p_{ij} = \mathbb{P}(X_1 = j | X_0 = i)$ καλείται πίνακας μετάβασης της αλυσίδας Markov και μαζί με την αρχική κατάσταση, την χαρακτηρίζει πλήρως. Οι αλυσίδες Markov έχουν πολλές καλές και χρήσιμες ιδιότητες, όπως για παράδειγμα την ύπαρξη μίας στάσιμης κατανομής στην οποία συγκλίνουν υπό κάποιες συνθήκες. Επίσης, χρησιμοποιούνται ως μοντέλα σε μία πληθώρα εφαρμογών.

Για τους σκοπούς τη εργασίας αυτής, θα μελετήσουμε μία συγκεκριμένη κατηγορία αλυσίδων Markov οι οποίες είναι γνωστές ως αλυσίδες Γεννήσεων-Θανάτων. Πρόκειται για μία κλάση αλυσίδων όπου οι μεταβάσεις από ένα σημείο περιορίζονται μόνο σε δύο 'γειτονικά' του σημεία, δηλαδή ο πίνακας μεταβάσεων είναι τριπλά διαγώνιος. Τυπικά, μία αλυσίδα Γεννήσεων-Θανάτων ορίζεται ως:

Ορισμός 3. *Μία αλυσίδα Markov με χώρο καταστάσεων $\mathcal{X} = \{0, 1, \dots, k\}$ για κάποιο $k \in \mathbb{N}$ ονομάζεται αλυσίδα Γεννήσεων-Θανάτων αν για τον πίνακα μεταβάσεών της $P = [P_{ij}]$ ισχύει:*

$$P_{ij} = \begin{cases} p_i & , j = i+1 \\ q_i & , j = i-1 \\ 1 - p_i - q_i & , j = i \\ 0 & , \text{διαφορετικά} \end{cases} \qquad \forall i, j \in \mathcal{X}$$

όπου $q_0 = 0$ και $p_k = 0$ για τα άκρα της αλυσίδας. Μία γραφική απεικόνιση δίνετε στο Σχήμα 1.



Σχήμα 1: Μία Αλυσίδα Γεννήσεων-Θανάτων

Αν επιπλέον $p_0 = 0$, τότε η κατάσταση $X = 0$ καλείται κατάσταση απορρόφησης με την έννοια πως αν η φτάσουμε σε αυτήν την κατάσταση δεν μπορούμε να φύγουμε. Σε αυτήν την περίπτωση, η ποσότητα που μας ενδιαφέρει να μελετήσουμε είναι ο μέσος χρόνος έκλειψης (EET), δηλαδή ο αναμενόμενος αριθμός βημάτων, ξεκινώντας από κάποια κατάσταση $\ell \in \{0, 1, \ldots, k\}$ μέχρι να φτάσουμε στην κατάσταση απορρόφησης $X = 0$. Συμβολίζουμε αυτήν την ποσότητα με $h(\ell)$. Μπορούμε να υπολογίσουμε την έκφραση αυτή σε σχέση με τις πιθανότητες p_i, q_i και να διατυπώσουμε το ακόλουθο θεώρημα:

Θεώρημα 6. *Ο μέσος χρόνος έκλειψης μίας αλυσίδας Γεννήσεων-Θανάτων με κατάσταση απορρόφησης $X = 0$ ξεκινώντας από κάποια αρχική κατάσταση $\ell \in \{1, \ldots, k\}$ είναι*

$$h(\ell) = \frac{1}{q_1} + \sum_{i=2}^{k} \frac{p_1 \cdots p_{i-1}}{q_1 \cdots q_i} + \sum_{i=1}^{\ell-1} \Big(\frac{q_1 \cdots q_i}{p_1 \cdots p_i} \sum_{j=i+1}^{k} \frac{p_1 \cdots p_{j-1}}{q_1 \cdots q_j} \Big)$$

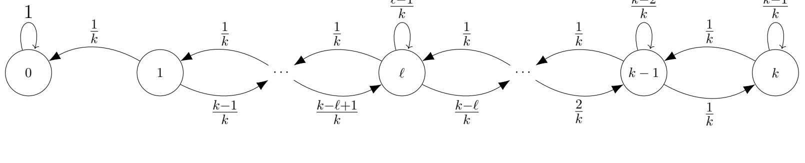
με $h(0) = 0$.

Χρησιμοποιώντας αυτό το Θεώρημα, θα ορίσουμε δύο ειδικές αλυσίδες Γεννήσεων-Θανάτων και θα υπολογίσουμε το EET τους. Αυτές οι αλυσίδες συνδέονται άμεσα με το competitive ratio του Αρμονικού Αλγορίθμου για το γενικευμένο πρόβλημα των k-εξυπηρετητών σε ομοιόμορφες μετρικές και για αυτόν τον λόγο τις ορίζουμε και τις μελετάμε.

Θεώρημα 7. *Ο μέσος χρόνος έκλειψης της Αρμονικής Αλυσίδας (Σχήμα 2) με κατάσταση απορρόφησης $X = 0$ ξεκινώντας από κάποια αρχική κατάσταση $\ell \in \{1, \ldots, k\}$ είναι*

$$h(\ell) = k \sum_{i=k-\ell+1}^{k} \alpha_i$$

με $h(0) = 0$, όπου α_ℓ είναι η λύση της αναδρομικής σχέσης $\alpha_\ell = 1 + (\ell - 1)\alpha_{\ell-1}$.

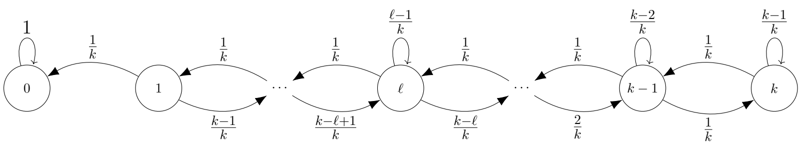


Σχήμα 2: Η Αρμονική Αλυσίδα

Θεώρημα 8. *Ο μέσος χρόνος έκλειψης της Δυαδικής Αλυσίδας (Σχήμα 3) με κατάσταση απορρόφησης $X = 0$ ξεκινώντας από κάποια αρχική κατάσταση $\ell \in \{1, \dots, k\}$ είναι*

$$h(\ell) = 2^k - 1 + \sum_{i=1}^{\ell-1} \frac{1}{\binom{k-1}{i}} (2^k - \sum_{j=0}^{i} \binom{k}{j})$$

με $h(0) = 0$.



Σχήμα 3: Η Δυαδική Αλυσίδα

Αλγόριθμοι χωρίς μνήμη για το γενικευμένο πρόβλημα των k-εξυπηρετηρών

Είμαστε πλέον σε θέση να διατυπώσουμε και να αποδείξουμε τα πρωτότυπα αποτελέσματα αυτής της διπλωματικής εργασίας. Όπως είπαμε, μελετάμε τη δύναμη των αλγορίθμων χωρίς μνήμη για το γενικευμένο πρόβλημα των k-εξυπηρετητών σε ομοιόμορφους χώρους. Για αυτό το πρόβλημα, ένας αλγόριθμος χωρίς μνήμη χαρακτηρίζεται πλήρως από μία κατανομή $p_1, \dots, p_k$ όπου p_i είναι η πιθανότητα με την οποία ο αλγόριθμος ικανοποιεί ένα αίτημα στην μετρική M_i.

Αρχικά δίνουμε ένα άνω φράγμα για το competitive ratio του Αρμονικού Αλγορίθμου, δηλαδή του αλγορίθμου που κουνιέται σε κάθε μετρική M_i με πιθανότητα $p_i = \frac{1}{k}$. Ουσιαστικά περιγράφουμε τον τρόπο με τον οποίο λειτουργεί αυτός ο αλγόριθμος και τον συνδέουμε με την Αρμονική Αλυσίδα. Έπειτα, ορίζουμε μία συνάρτηση δυναμικού για το πρόβλημα που ορίζεται ως το EET της Αρμονικής Αλυσίδας με αρχική κατάσταση το πλήθος των εξυπηρετητών που ο αλγόριθμος και ο αντίπαλος έχουν σε διαφορετικές θέσεις. Χρησιμοποιώντας την μέθοδο των Εναλλασσόμενων Κινήσεων, μπορούμε να δείξουμε ένα άνω φράγμα $k \cdot \alpha_k$ για τον Αρμονικό Αλγόριθμο και να αποδείξουμε το Θεώρημα 1.

Στη συνέχεια εργαζόμαστε για την απόδειξη του κάτω φράγματος του Θεωρήματος 2. Σε αυτήν την περίπτωση μελετάμε οποιονδήποτε αλγόριθμο χωρίς μνήμη και επομένως χαλάει η συμμετρία του προβλήματος. Ως συνέπεια, πρέπει να δουλέψουμε σε μία αλυσίδα Markov με 2^k καταστάσεις αντί για k. Η άμεση ανάλυση της αλυσίδας αυτής δεν είναι εφικτή για γενικές τιμές του k. Ωστόσο, εφαρμόζοντας τεχνικές παρόμοιες με αυτές των Chiplunkar και Vishwanathan [28] μπορούμε να πάρουμε ένα κάτω φράγμα, το οποίο αποδεικνύει πως (1) κάθε αλγόριθμος χωρίς μνήμη έχει competitive ratio τουλάχιστον $k \cdot \alpha_k$ και (2) μόνο ο Αρμονικός Αλγόριθμος πετυχαίνει αυτήν την τιμή, δηλαδή είναι ο βέλτιστος αλγόριθμος χωρίς μνήμη για το πρόβλημα.

Η τεχνική που εφαρμόσαμε για την απόδειξη του Θεωρήματος 2 απαιτούσε μετρικούς χώρους με $n \geq 3$ σημεία. Αυτό είναι απαραίτητο, αφού για $n = 2$ η δομή του προβλήματος απλοποιείται σημαντικά και τελικά το competitive ratio του Αρμονικού Αλγορίθμου βελτιώνεται σε $\Theta(2^k)$ από $\Theta(k!)$ (Θεώρημα 3). Για να αποδείξουμε το Θεώρημα αυτό, συνδέουμε την επίδοση του Αρμονικού Αλγορίθμου με το EET της Δυαδικής Αλυσίδας και χρησιμοποιώντας παρόμοιες τεχνικές με αυτές του Θεωρήματος 1, παίρνουμε το ζητούμενο.

Τέλος, μελετάμε την περίπτωση oblivious αντιπάλων, οι οποίοι δεν είχαν μελετηθεί για αλγορίθμους χωρίς μνήμη έως τώρα σε αυτό το πρόβλημα. Κατασκευάζοντας oblivious εισόδους χωρίς γνώση των επιλογών του Αρμονικού Αλγορίθμου, μπορούμε να δείξουμε ότι το competitive ratio του δεν αλλάζει σε σχέση με το μοντέλο του adaptive online αντιπάλου, αποδεικνύοντας τα Θεωρήματα 4 και 5.

Συμπεράσματα

Συνοψίζοντας, μελετήσαμε για πρώτη φορά την δύναμη των αλγορίθμων χωρίς μνήμη για το γενικευμένο πρόβλημα των k-εξυπηρετητών σε ομοιόμορφους μετρικούς χώρους. Ενάντια σε adaptive online αλγορίθμους, μπορέσαμε να δείξουμε ότι τα ‘σφιχτά’ διπλά εκθετικά φράγματα των Chiplunkar και Vishwanathan [28] βελτιώνονται σε φράγματα της τάξης του $k!$. Ο βέλτιστος αλγόριθμος χωρίς μνήμη που τα πετυχαίνει είναι ο Αρμονικός Αλγόριθμος. Επίσης, μελετήσαμε ξεχωριστά την περίπτωση των μετρικών χώρων με $n = 2$ σημεία και δείξαμε ότι εκεί το competitive ratio του προβλήματος είναι $\Theta(2^k)$. Τέλος, εξετάσαμε και το μοντέλο του oblivious αντιπάλου, στο οποίο δείξαμε ότι η επίδοση του Αρμονικού Αλγορίθμου δεν αλλάζει, υποδεικνύοντας ότι για το συγκεκριμένο μοντέλο ο adaptive online αντίπαλος δεν έχει περισσότερη δύναμη από τον oblivious.

Σε συνδιασμό με τα αποτελέσματα των Chiplunkar και Vishwanathan, η δουλειά αυτή χαρακτηρίζει πλήρως την δύναμη των αλγορίθμων χωρίς μνήμη σε ομοιόμορφους μετρικούς χώρους με ή χωρίς βάρη. Ένα επόμενο βήμα θα ήταν να εξετάσουμε πιο πολύπλοκους χώρους, όπως για παράδειγμα άστρα ή δέντρα. Ωστόσο, επισημαίνουμε ότι οι αλγόριθμοι χωρίς μνήμη έχουν μη-φραγμένο competitive ratio σε γενικούς μετρικούς χώρους ακόμα και για $k = 2$: αυτό έχει δειχθεί ανεξάρτητα από τους Chrobak και Sgall [32] και Koutsoupias και Taylor [58].

Contents

1 Introduction **1**

1.1 Contribution . . . 4

1.2 Organization . . . 6

2 Online Algorithms **9**

2.1 Competitive Analysis . . . 10

2.2 The Potential Method . . . 13

2.2.1 Amortized Cost . . . 14

2.2.2 Interleaving Moves . . . 14

2.3 Paging . . . 16

2.3.1 Deterministic Algorithms for Paging . . . 16

2.3.2 Randomized Algorithms for Paging . . . 17

2.4 Metrical Task Systems . . . 18

2.5 The k-Server Problem . . . 19

2.6 The Generalized k-Server Problem . . . 21

2.6.1 Results on the generalized k-server problem . . . 22

2.6.2 Results on memoryless algorithms . . . 23

3 Random Walks **29**

3.1 Markov Chains . . . 29

3.2 Random Walks on Birth-Death Chains . . . 31

3.2.1 Stationary Distribution of Birth-Death Chains . . . 32

3.2.2 Expected Extinction Time of Birth-Death Chains . . . 33

3.3 Random Walks on the Harmonic Chain . . . 36

3.4 Random Walks on the Binary Chain . . . 38

4 Memoryless Algorithms for the Generalized k-Server Problem **41**

4.1 Adaptive Upper Bound for the Harmonic Algorithm . . . 42

4.1.1 Definition of the Potential Function . . . 42

4.1.2 Bounding the Competitive Ratio . . . 44

4.2 General Adaptive Lower Bound . . . 46

4.2.1 Constructing the adversarial instance . . . 46

4.2.2 Proving the Lower Bound 48
4.3 Special Case: Two Point Metric Spaces 52
4.4 Oblivious Bounds 53
4.4.1 Oblivious Lower Bound for the Harmonic Algorithm 53
4.4.2 Special Case: Two Point Metrics 56

5 Concluding Remarks **59**

Bibliography **61**

Chapter 1

Introduction

In classical optimization, we are given a problem and a specific input, and the goal is to find the optimal solution for the given input. However, in many real life applications, the assumption that the whole input is available, is not realistic. Most of the times, we need to solve optimization problems, while taking decisions with incomplete information about the input. We call such problems online optimization problems, or simply *online problems*. To solve problems in online optimization, we need to design *online algorithms* which make decisions without knowledge of the future.

The primary objective of this field is the design of online algorithms whose performance has some specific guarantees compared to the performance of an optimal *offline algorithm* that knows the entire input before making any decisions. This is captured by the notion of *competitive ratio* used in competitive analysis.

The *k-server problem* is one of the most fundamental and extensively studied problems in the theory of online algorithms. In fact, the study of this problem over the last decades (that continues until this day) has led to the development of many techniques and a deeper understanding of online algorithms.

In this problem, we are given a metric space of n points and k mobile servers located at points of the metric space. At each step, a request arrives at a point of the metric space and must be served by moving a server there. The goal is to minimize the total distance travelled by the servers.

The k-server problem generalizes various online problems, most notably the *paging* (caching) *problem*, which corresponds to the k-server problem on uniform metric spaces. Paging, first studied in the seminal work of Sleator and Tarjan [71], is well-understood: the competitive ratio is k for deterministic algorithms and $H_k = \Theta(\log k)$ for randomized; those algorithms and matching lower bounds are folklore results for online algorithms [71, 62, 1].

The k-server problem in general metric spaces is much deeper and intriguing. In a landmark result, Koutsoupias and Papadimitriou [56] showed that the *Work Function Algorithm* (WFA) [56] is $(2k-1)$-competitive, which is almost optimal for deterministic algorithms since the competitive ratio is at least k [61]. For randomized algorithms, it is believed that an $O(\log k)$-competitive algorithm is possible; despite several breakthrough results over the last decade [7, 19, 20, 60], this conjecture still remains open.

Due to the massive interest of the scientific community on this problem and our lack of understanding many of its properties, many variations of the k-server problem have been proposed, such as (i) the *weighted k-server problem* [40, 10] where each server has a different weight that affects its moving cost, (ii) the *CNN problem* [58, 50] where there are two servers moving freely in two disjoint lines and (iii) the *(h, k)-server problem* [54, 71, 79, 53] where the algorithm has k servers but its performance is compared to that of an optimal offline algorithm that has $h \leq k$ servers.

In this thesis, we focus on the *generalized k-server problem*, a far-reaching extension of the k-server problem, introduced by Koutsoupias and Taylor [58]. Here, each server s_i lies in a different metric space M_i and a request is a tuple $(r_1, \ldots, r_k)$, where $r_i \in M_i$; to serve it, some server s_i should move to point r_i.

The generalized k-server problem has a much richer structure than the classic k-server problem and is much less understood. For general metric spaces, no $f(k)$-competitive algorithms are known, except from the special case of $k = 2$ [69, 70, 68]. For $k \geq 3$, competitive algorithms are known only for the following special cases:

1. Uniform Metrics: All metric spaces $M_1, \ldots, M_k$ are uniform (possibly with different number of points), with the same pairwise distance, say 1.

2. Weighted Uniform Metrics: All metrics are uniform, but they have different weights; the cost of moving in metric M_i is w_i.

Perhaps surprisingly, those two cases are qualitatively very different. For deterministic algorithms Bansal et. al. [11] obtained algorithms with (almost) optimal competitive ratio. For uniform metrics their algorithm is $(k \cdot 2^k)$-competitive, while the best possible ratio is at least $2^k - 1$ [58]. For weighted uniform metrics, they obtained a $2^{2^{k+3}}$-competitive algorithm (by extending an algorithm of Fiat and Ricklin [40] for weighted k-server on uniform metrics), while the lower bound for the problem is $2^{2^{k-4}}$ [10].

We note that for uniform metrics, if memory is allowed and we compare against oblivious adversaries, competitive randomized algorithms are known: Bansal et. al. [11] designed a $O(k^3 \log k)$-competitive randomized algorithm with memory; this was recently improved to $O(k^2 \log k)$ by Bienkowski et. al. [15].

The generalized k-server problem is named from the fact that it generalized many well-known online problems, including the k-server problem. Some well-studied special cases of the generalized k-server problem are:

- The *k-server problem*: the standard k-server problem is the very special case of generalized k-server where all metric spaces are identical, i.e., $M_i = M$ and all requests are of the form $(r, r, \ldots, r)$.

- The *weighted k-server problem* [40, 10]: this is the weighted variant of the classic k-server problem where each server has different weight and the cost of moving server

i by distance d is $w_i \cdot d$. This is the special case of generalized k-server where all metrics are scaled copies of a fixed metric M, i.e., $M_i = w_i M$ and all requests are of the form $(r, r, \dots, r)$.

- The *CNN problem* [58, 50]: in this problem we have two servers s_x and s_y, moving on the horizontal and the vertical lines respectively and a request is a point $r = (r_x, r_y)$ that has to be served by moving s_x to r_x or s_y to r_y. This is the special case of generalized k-server where $k = 2$ and M_1, M_2 are lines.

Memoryless Algorithms: One drawback of the online algorithms achieving the best-known competitive ratios for the k-server problem is that they are computationally inefficient. For example, the space used by the WFA is proportional to the number of different configurations of the servers, i.e., $\binom{n}{k}$, which makes the whole approach quite impractical.

This motivates the study of trade-offs between the competitive ratio and computational efficiency. A starting point in this line of research, is to determine the competitive ratio of *memoryless algorithms:* a memoryless algorithm, decides the next move based solely on the current configuration of the servers and the given request.

Memoryless algorithms for the k-server problem have been extensively studied (see e.g., [16, 55] for detailed surveys). The most natural memoryless algorithm is the *Harmonic Algorithm*, which moves each server with probability inversely proportional to its distance from the requested point. It is known that its competitive ratio is $O(2^k \cdot \log k)$ and $\Omega(k^2)$ [12]. It is conjectured that in fact the Harmonic Algorithm is $\frac{k(k+1)}{2} = O(k^2)$-competitive; this remains a long-standing open problem. For special cases such as uniform metrics (paging) and resistive metric spaces, an improved competitive ratio of k can be achieved and this is the best possible for memoryless algorithms [33].

We note that the study of memoryless algorithms for the k-server problem is of interest only for randomized algorithms; it is easy to see that any deterministic memoryless algorithm is not competitive. Throughout this thesis, we study two different adversarial settings: the *adaptive online setting*, which is the standard benchmark for evaluating the performance of randomized online algorithms, and the *oblivious setting* that usually allows for better competitive ratios. For a detailed discussion on the different adversary models and relations between them, see [16, 14]. We also discuss the main definitions and differences between the adversarial models in Chapter 2.

The primary focus of this thesis is the study of memoryless algorithms for the generalized k-server problem on uniform metrics. A memoryless algorithm for the generalized k-server problem receives a request $r = (r_1, \dots, r_k) \in [n]^k$ and decides which server to move based only on its current configuration $q = (q_1, \dots, q_k) \in [n]^k$ and r. For the case of uniform metrics, a memoryless algorithm is fully characterized by a probability distribution $p = (p_1, \dots, p_k)$; whenever it needs to move a server, it uses server s_i of metric M_i with probability p_i. Throughout the thesis we assume for convenience (possibly by relabeling the metrics) that given a memoryless algorithm we have that $p_1 \geq p_2 \geq \dots \geq p_k$.

We also assume that $p_i > 0$ for all i; otherwise it is trivial to show that the algorithm is not competitive.

In the context of generalized k-server on uniform metrics, the Harmonic Algorithm is a memoryless algorithm which moves at all metric spaces with equal probability, i.e., $p_i = 1/k$, for all $i \in [k]$. Due to the problem's symmetry, the only "reasonable" memoryless algorithm is in fact the Harmonic Algorithm; any other memoryless algorithm for the problem will be more likely to serve a request in some metric space M_i over some other metric space M_j, without any reason. This is formally shown in our work, by proving that against adaptive online adversaries, the Harmonic Algorithm has strictly the best competitive ratio out of all the memoryless algorithms.

Recently Chiplunkar and Vishnawathan [28] studied randomized memoryless algorithms in weighted uniform metrics. They showed tight doubly exponential ($\approx 1.6^{2^k}$) bounds on the competitive ratio. Interestingly, the memoryless algorithm achieving the optimal bound in this case is different from the Harmonic Algorithm.

Since the weighted uniform case seems to be much harder than the uniform case, it is natural to expect that a better bound can be achieved by memoryless algorithms in uniform metrics. Moreover, in weighted uniform metric spaces the competitive ratios of deterministic algorithms (with memory) and randomized memoryless algorithms are essentially the same. Recall that a similar phenomenon occurs for the paging problem (standard k-server on uniform metrics) where both deterministic and randomized memoryless algorithms have a competitive ratio of k.

Thus, it is natural to guess that for uniform metrics, a competitive ratio of order 2^k (i.e., same as the deterministic competitive ratio) can be achieved by memoryless algorithms. However, as we show in this work, this is not the case; the competitive ratio of any memoryless algorithm for the generalized k-server problem on uniform metrics is $\Omega(k!)$.

1.1 Contribution

In this thesis we study the power of memoryless algorithms for the generalized k-server problem in uniform metrics and we determine the exact competitive ratio by obtaining tight bounds.

We begin by turning our attention to the adaptive online adversarial setting since this is the standard benchmark to evaluate randomized memoryless algorithms.

First, we determine the competitive ratio of the Harmonic Algorithm on uniform metrics.

Theorem 1.1. *The Harmonic Algorithm for the generalized k-server problem on uniform metrics is $(k \cdot \alpha_k)$-competitive against adaptive online adversaries, where α_k is the solution of the recursion $\alpha_k = 1 + (k-1)\alpha_{k-1}$, with $\alpha_1 = 1$.*

It is not hard to see that $\alpha_k = \Theta((k-1)!)$, therefore the competitive ratio of the Harmonic Algorithm is $O(k!)$. This shows that indeed, uniform metric spaces allow for substantial improvement on the performance compared to weighted uniform metric spaces where there is a doubly-exponential lower bound.

To obtain this result, we analyse the Harmonic Algorithm using Markov Chains and random walks, based on the *Hamming distance* between the configuration of the algorithm and the adversary, i.e., the number of metric spaces where they have their servers in different points. Based on this, we then provide a proof using a potential function, which essentially captures the expected cost of the algorithm until it reaches the same configuration as the adversary.

Next we show that the upper bound of Theorem 1.1 is tight in the adaptive online setting, by providing a matching lower bound.

Theorem 1.2. *The competitive ratio of any randomized memoryless algorithm for the generalized k-server problem on uniform metrics with $n \geq 3$ points is at least $k \cdot \alpha_k$ against adaptive online adversaries.*

Here the analysis differs, since the Hamming distance is not the right metric to capture the "distance" between the algorithm and the adversary: assume that all their servers are at the same points, except one, say server s_i. Then, in the next request, the algorithm will reach the configuration of the adversary with probability p_i; clearly, if p_i is large, the algorithm is in a favourable position, compared to the case where p_i is small.

This suggests that the structure of the algorithm is not solely characterized by the number of different servers (i.e., the Hamming distance) between the algorithm and the adversary, but also the labels of the servers matter. For that reason, we need to focus on the subset of different servers, which gives a Markov Chain on 2^k states. Unfortunately, analyzing such chains in a direct way can be done only for easy cases like $k = 2$ or $k = 3$. For general values of k, we find an indirect way to characterize the solution of this Markov Chain. A similar approach was taken by Chiplunkar and Vishwanathan [28] for weighted uniform metrics; we use some of the properties they showed, but our analysis differs since we need to make use of the special structure of our problem to obtain our bounds.

In fact, as a direct corollary of Theorem 1.2 we are able to show that any memoryless algorithm other than the Harmonic has competitive ratio strictly larger than $k \cdot \alpha_k$ in metric spaces with $n \geq 3$ points against an adaptive online adversary.

Notice that in Theorem 1.2 we require that all metric spaces have at least $n \geq 3$ points. We observe that this is necessary, and that if all metric spaces have $n = 2$ points, the Harmonic Algorithm is $O(2^k)$-competitive, thus a lower bound of $k \cdot \alpha_k$ can not be achieved. We also show an adaptive online lower bound of $\Omega(2^k)$ for the Harmonic Algorithm, which leads to determining his competitive ratio.

Theorem 1.3. *The competitive ratio of the Harmonic Algorithm for the generalized k-server problem on uniform metrics with $n = 2$ points is $\Theta(2^k)$ against adaptive online adversaries.*

The underlying reason behind this difference in competitive ratio is the "limitations" enforced to the request sequence by the limited number of points; if any algorithm is at any configuration $q \in [2]^k$, then the only request that forces it to move is its anti-configuration $r = \bar{q}$. We study this case separately using Markov Chains once again, by proving tight bounds on the competitive ratio of the Harmonic Algorithm in metric spaces with $n = 2$ points.

Then, we turn our attention to the oblivious adversarial model. The upper bounds of Theorems 1.1 and 1.3 for the Harmonic Algorithm clearly extend to the oblivious adversarial setting since they hold against adaptive online adversaries. We have shown that in the adaptive online adversarial setting the Harmonic Algorithm is (strictly) the best memoryless algorithm; we believe that there is no reason for this to change in the oblivious setting. Thus, we focus on giving lower bounds for the Harmonic Algorithm in the oblivious setting.

Theorem 1.4. *The competitive ratio of the Harmonic Algorithm for the generalized k-server problem on uniform metrics with $n \geq 2k + 1$ is at least $k \cdot \alpha_k$ against oblivious adversaries.*

Theorem 1.5. *The competitive ratio of the Harmonic Algorithm for the generalized k-server problem on uniform metrics with $n = 2$ is $\Omega(2^k)$ against oblivious adversaries.*

We have shown oblivious lower bounds that match the upper bounds of the adaptive online setting; this indicates that for memoryless algorithms for the generalized k-server problem in uniform metrics, adaptive online and oblivious adversaries have the same power.

Summarizing, on the positive side, our results show that improved guarantees can be achieved compared to the weighted uniform case. On the other hand, the competitive ratio of memoryless algorithms ($\Theta(k!)$) is asymptotically worse than the deterministic competitive ratio of $2^{O(k)}$. This is somewhat surprising, since (as discussed above) in most uniform metric settings of k-server and generalizations, the competitive ratio of deterministic algorithms (with memory) and randomized memoryless is (almost) the same.

1.2 Organization

The first part of this thesis is dedicated to establishing the necessary background for our novel results.

In Chapter 2, we introduce the framework of *Online Algorithms*. We begin by formally defining an online problem, the notion of competitive ratio and the different kinds of adversarial settings. Then, we present the potential-based analysis framework that is commonly used in competitive analysis; in fact, we also rely on this technique to prove some of our results. As a concrete example for this framework, we define the *paging* problem and prove the main results on this area. The reason we present this problem is that (i) it has a simple structure that captures the properties of an online problem and (ii) it is the special case of the k-server problem on uniform metric spaces that are of interest to this thesis. Next, we discuss some of the most important online problems, namely the Metrical Task Systems (MTS) problem and the k-server problem, and briefly state their history and the known results. We conclude this chapter by formally defining the generalized k-server problem, the best known algorithms for it and how our work extends these results.

In Chapter 3, we provide the reader with the necessary background on *Random Walks* and random processes in general. After a short introduction on Markov Chains and their main properties, we study a special class of discrete-time, finite-space Markov Chains called Birth-Death Chains. This mathematical model is very important for our analysis, since (as we show) it captures the competitive ratio of the Harmonic Algorithm for the generalized k-server problem and constitutes the primary source of inspiration for our analysis. Specifically, we are interested in two special types of Birth-Death Chains, called the Harmonic Chain and the Binary Chain, that are defined and studied in this thesis. As we show, the properties of this chains (namely their Expected Extinction Time) immediately give bounds for the competitive ratio of the Harmonic Algorithm.

After we lay the groundwork for our analysis, in Chapter 4 we present our novel results on memoryless algorithms for the generalized k-server problem on uniform metrics that this thesis contributes. We begin by studying the adaptive online adversarial setting. We first give an upper bound on the competitive ratio of the Harmonic Algorithm in Theorem 1.1. Then we analyze any memoryless algorithm and show that the Harmonic Algorithm is the optimal and its competitive ratio is exactly $k \cdot \alpha_k$ as stated in Theorem 1.2. We also handle the special case of metric spaces with $n = 2$ points with Theorem 1.3. Next, we turn our attention to oblivious adversaries and the competitive ratio of the Harmonic Algorithm in this setting. In Theorems 1.4 and 1.5 we show that the bounds of the adaptive online setting are also tight in the oblivious setting, indicating that for memoryless algorithms for the generalized k-server problem in uniform metrics, adaptive online and oblivious adversaries have the same power.

Finally, in Chapter 5 we conclude this thesis by stating the main message of this work and how it extends the current knowledge of online algorithms and specifically the generalized k-server problem. We also discuss open directions in this area and suggest the next steps towards understanding the generalized k-server problem.

Chapter 2

Online Algorithms

While in traditional optimization, an algorithm is given an instance and needs to compute the best solution for it, in online computation an algorithm must produce a sequence of decisions based on past events and without information about the future. *Online algorithms* [16, 52] are a natural topic of interest in many disciplines such as computer science, economics an operations research, due to the fact that many computational problem are intrinsically online and that they require immediate decisions on real time.

The standard framework used to evaluate the performance of online algorithms is *competitive analysis*, which was introduced by Sleator and Tarjan [71]. Here, the performance of an online algorithm is compared to the optimal offline solution which knows the whole input in advance. The roots of competitive analysis can be found in classical combinatorial optimization problems (i.e., scheduling problem [25, 18]) and in the analysis of data structures (i.e., list accessing problem [71, 4, 42]).

Organization. In this chapter, we begin by formally defining the framework of competitive analysis in Section 2.1. There, we will define the competitive ratio of an online algorithm, the competitive ratio of a problem and the different adversarial settings that are used to evaluate randomized online algorithms. Then, in Section 2.2 we introduce the potential method technique as a way to prove upper bounds on the competitive ratio of an online algorithm; in fact, we are going to use this method to prove some of our novel results, so it is useful to formally define it in this chapter.

After establishing the basic background on competitive analysis, we study some of the most extensively studied problems in online computation; namely:

1. The *paging problem.* In this problem, we have a cache memory of size k and $n \geq k$ element, some of them occupying a cache cell. We are given requests in the form of one of the n points, and if this point is not in the cache then we need to move it in by evicting some other element. The objective is to minimize the cost of the algorithm which is given by the total number of evictions.

2. The *metrical task systems* (MTS) *problem.* This problem is a great generalization of

various well-studied online problems and creates a unifying framework that enables the study of online algorithms in a systematic way. In the MTS problem we are given a server which can be in one of N different states and a metric distance function d specifying the cost of switching between the states. At each time step, a task r arrives, represented by a vector $r = (r_1, r_2, \dots, r_N)$ where r_i is the cost of processing r at state i. Then, the algorithm needs to decide a state i to process the request r, paying the processing cost r_i and the movement cost (based on d) to move at i from his current state.

3. The *k-server problem.* In this problem, we are given a metric space of n points and k mobile servers located at points of the metric space. At each step, a request arrives at a point of th metric space and must be served by moving a server there. The goal is to minimize the total distance travelled by the servers.

In Section 2.3, we present the paging problem since it is an excellent introduction to the online framework. We state the main results for paging and also their proofs, in order to higlight some of the techniques used in competitive analysis. Then, in Sections 2.4 and 2.5 we define the MTS and k-server problems respectively and briefly mention their history and the main results.

Finally, in Section 2.6 we conclude this chapter by formally giving the definition of the generalized k-server problem, which is the focus of this thesis. We give a comprehensive statement of the main results for this problem, the corresponding techniques and how our work extends these results.

2.1 Competitive Analysis

Formally, in an online optimization problem the input σ is divided into requests $\sigma = \sigma_1, \dots, \sigma_m$. Whenever each request σ_t is received, some actions must be performed, without knowledge of the future requests $\sigma_{t+1}, \dots, \sigma_m$. An algorithm that solves an online optimization problem is called an *online algorithm.* An algorithm which reads the whole input σ and then produces a solution is called an *offline algorithm.*

The standard framework used to evaluate the performance of online algorithms is *competitive analysis*, which was introduced by Sleator and Tarjan [71]. Here, the performance of an online algorithm is compared to the optimal offline solution which knows the whole input in advance.

From now on, we focus on minimization problems, since this is the case for all problems considered in this thesis. Let P be a minimization problem and let I denote the set of all valid inputs for P. For an instance $\sigma \in I$, let $OPT(\sigma)$ denote the optimal cost on σ. For an online algorithm ALG, let $ALG(\sigma)$ denote the cost of ALG on σ.

Definition 2.1 (Competitive Ratio)**.** *An online algorithm ALG for an online minimization problem P is c-competitive if there exists a constant α such that for any input $\sigma \in I$,* *we have*

$$ALG(\sigma) \leq c \cdot OPT(\sigma) + \alpha.$$

If $\alpha = 0$ we say that algorithm ALG is strictly c-competitive [35]. The competitive ratio of an algorithm ALG is the infimum value c such that ALG is c-competitive. The competitive ratio of an online minimization problem P is the infimum value c for which a c-competitive algorithm for P exists.

Note that in the above definition, the constant α is allowed to depend on the parameters of the problem P (for example, the number of servers in k-server) but not on the instance I or on m.

An alternative view of competitive analysis is to think of each online problem as a game between an algorithm and an all-powerful adversary. The adversary knows the description of the algorithm and constructs an input in order to maximize the ratio between the cost of the algorithm and the optimal cost. For deterministic algorithms, the notion of competitive ratio is well-defined, and is given by the ratio of the algorithm's cost and the optimal offline algorithm's cost. However, a usual approach in the design of online algorithms is to use randomization. This raises the issue of how much information is available to the adversary when it constructs the instance. In competitive analysis, there are various adversary models [14] proposed to evaluate randomized algorithms.

1. **Oblivious Adversaries:** In this model, the adversary knows the description of the algorithm, but it does not know its random choices and it has to construct the whole input before the algorithm starts serving the requests. A randomized online algorithm ALG for a minimization problem P is c-competitive against oblivious adversaries if there exists a constant α, such that for any request sequence σ generated by an oblivious adversary, that is

$$\mathbb{E}[ALG(\sigma)] \leq c \cdot OPT(\sigma) + \alpha,$$

   where $OPT(\sigma)$ is used to denote the cost of the optimal offline algorithm on the instance.

2. **Adaptive Online Adversaries:** In this model, the adversary knows all the actions of the algorithm, including its random choices. At each step, the adversary generates a request in order to maximize the cost incurred by the algorithm. However, the adversary must also serve the request sequence online. This way, the costs of both the algorithm and the adversary depend on the random choices of the algorithm. A randomized online algorithm ALG for a minimization problem P is c-competitive against adaptive online adversaries, if there exists a constant α, such that for any request sequence σ generated by an adaptive online adversary ADV,

$$\mathbb{E}[ALG(\sigma)] \leq c \cdot ADV(\sigma) + \alpha,$$

   where $ADV(\sigma)$ is used to denote the cost of ADV to serve σ.

3. **Adaptive Offline Adversaries:** Similarly to the adaptive online adversary, an adaptive offline adversary knows all the actions of the algorithm, including its random choices, and at each step, it generates a request in order to maximize the cost incurred by the algorithm. However, in this setting, the adversary is allowed to solve the problem offline after the instance is constructed. A randomized online algorithm ALG for a minimization problem P is c-competitive against adaptive offline adversaries, if there exists a constant α, such that for any request sequence σ generated by an adaptive offline adversary ADV, that is

   $$\mathbb{E}[ALG(\sigma)] \leq c \cdot OPT(\sigma) + \alpha,$$

   where $OPT(\sigma)$ is used to denote the cost of the optimal offline algorithm on the instance.

Depending on the adversarial setting, the competitive ratio of the same online algorithm might vary drastically. Typically, the oblivious adversary setting is the "easiest" (often referred to as the "weak adversary") since it doesn't have access to the algorithm's random number generator. However, in practise we usually use the adaptive online adversarial setting since it is useful to compare the performance of our algorithm to that of other online algorithms. The hardest adversarial setting is the adaptive offline, since the adversary has access to the algorithm's random choices but it is also allowed to solve the instance offline. In fact, it has been shown [14] that randomization doesn't help against adaptive offline adversaries.

We quote some of the most useful results that show the connection between the different adversarial settings and highlight their differences.

Theorem 2.1. *If there is a randomized algorithm that is c-competitive against any adaptive offline adversary then there also exists a c-competitive deterministic algorithm. If G is a c-competitive randomized algorithm against any adaptive online adversary, and there is a randomized d-competitive algorithm against any oblivious adversary, then G is a randomized $(c \cdot d)$-competitive algorithm against any adaptive offline adversary.*

Note that until now, we didn't mention any restrictions on the computational resources used by an online algorithm. The reason for this is that in competitive analysis our primary goal is to understand the importance of knowing the future; the competitive ratio of an algorithm can be seen as a measurement of the loss due to missing information, assuming unlimited computational power. For example, the Work Function Algorithm [56] that achieves the best known deterministic competitive ratio for the k-server problem needs exponential time.

However, we would obviously prefer to have more "realistic" algorithms that can run in polynomial (for example) time to the instance. This motivates the study of trade-offs between the competitive ratio and computational efficiency. A starting point in this line of research, is to determine the competitive ratio of *memoryless algorithms:* a memoryless algorithm, decides the next move based solely on the current configuration of the servers and the given request. Memoryless algorithms are typically computationally efficient, but in some cases their competitive ratio can be much larger than the optimal or even unbounded; as an example, Chrobak and Sgall [32] and Koutsoupias and Taylor [58] independently showed that even for $k = 2$, there is no competitive memoryless algorithm for the weighted k-server problem in general metric spaces, a result that immediately transfers to the generalized k-server problem.

2.2 The Potential Method

The *potential method* [43, 34] is a technique commonly used in competitive analysis in order to prove upper bounds on the competitive ratio of memoryless algorithms. The potential method has also been used extensively in the field of data structures [34, 44] as a method used to analyze the *amortized* time and space complexity of a data structure; a measure of its performance over sequences of operations that smooths out the cost of infrequent but expensive operations.

On a high level, the potential method consists of the definition of an appropriate potential function that is used to measure the distance between the algorithm and the adversary; then the analysis shifts to proving some properties for this function that can immediately yield results for the algorithm's competitive ratio.

For example, if we find a potential such that (i) when the adversary moves the increase in the potential is bounded from above and (ii) when the algorithm moves the decrease in potential is bounded from below, then we get some knowledge on the way that the requests can affect the "distance" between the algorithm and the adversary, resulting to knowledge on the performance of the algorithm.

Configurations. In any online problem, we can define an algorithm's configuration at some moment as the state of the algorithm with respect to the outside world. For example, in the MTS problem the configuration of an algorithm is simply the point of the metric that it occupies. In the paging problem the configuration of the algorithm is the labels of the "pages" that are in the cache. In k-server, the configuration of an algorithm is the set of points that are occupied by its servers.

Potential. Now, let S_{ALG} and S_{ADV} be the sets of all possible configurations for an algorithm and an adversary respectively. Formally, a potential function Φ is a mapping $\Phi : S_{ALG} \times S_{ADB} \to \mathbb{R}$; that is a systematic way to append real numbers for any combination of the algorithm's and the adversary's states. Typically, we wish for the potential to satisfy some distance properties, in the sense that when the algorithm and the adversary have the same configuration the potential should be 0, otherwise some real positive number that reflects how "far" these two configurations are. This way, the potential can be seen as a measure of distance between the algorithm and the adversary.

We will now show that if the potential is selected in order to satisfy some specific properties, then an upper bound for the competitive ratio of the algorithm can be immediately determined. There are two primary potential based techniques [16] for proving upper bounds on the competitive ratio of ALG: the *Amortized Cost* method and the *Interleaving Moves* method.

For any online minimization problem P with request sequence $\sigma = \sigma_1, \dots, \sigma_m$, fix an online algorithm ALG, an adaptive online adversary ADV and a potential function Φ. Let ALG_i and ADV_i denote the cost incurred by the algorithm and the adversary respectively. Also, let Φ_i denote the potential between the configurations of ALG and ADV after request σ_i has been processed.

2.2.1 Amortized Cost

In this method, the potential between the algorithm's and the adversary's configurations can be seen as a "bank". Whenever an algorithm makes a move that increases the potential (i.e., its distance from the adversary) then it can ”borrow” some money from the bank in order to pay the cost. Respectively, if the algorithm makes a move than decreases the potential, then it can ”pay” some of its debt to the bank.

Using this intuition, instead of charging the algorithm with its actual cost, we charge it with its *amortized cost.* For each request r_i, define the amortized cost of the algorithm as:

$$\alpha_i = ALG_i + \Phi_i - \Phi_{i-1}$$

The amortized cost of an algorithm is a measure of its performance over sequences of operations that smooths out the cost of infrequent but expensive operations.

Theorem 2.2. *If there exists constant c such that for any request σ_i it holds*

$$\alpha_i \leq c \cdot ADV_i$$

and Φ_i is bounded by some constant independent of the request sequence for any i, then ALG is c-competitive.

Proof. Assume that for all i, $\alpha_i \leq c \cdot ADV_i$. Then, summing over $i = 1, 2, \dots, m$ we would get

$$\sum_{i=1}^{m} \alpha_i \leq c \cdot ADV(\sigma) \Rightarrow ALG(\sigma) + \Phi_m - \Phi_0 \leq c \cdot ADV(\sigma)$$

by the telescoping property. Since Φ_i are bounded by constants, the theorem follows. □

2.2.2 Interleaving Moves

Imagine that the algorithm and the adversary move separately with every request. As we have already mentioned, the potential can be viewed as a measurement of distance between the algorithm and the adversary. Since the adversary wished to maximize the ratio of the algorithm's cost to its own, then whenever it moves it wishes to increase the potential. A "good" algorithm on the other side, wishes to make moves that decrease the potential.

The method of interleaving moves formalized the above concept.

Theorem 2.3. *Let σ_i be any request and assume that the following properties hold:*

1. *There exists a constant c such that if only the adversary moves during σ_i and pays x, then the potential increases by at most $c \cdot x$, that is:*

$$\Delta\Phi = \Phi_i - \Phi_{i-1} \leq c \cdot ADV_i$$

2. *If only the algorithm moves during σ_i and pays x, then the potential decreases by at least x, that is:*

$$\Delta\Phi = \Phi_i - \Phi_{i-1} \leq -ALG_i$$

3. *Φ_i is bounded by some constant that doesn't depend on the request sequence.*

Then, ALG is c-competitive.

Proof. From properties 1 and 2, we get that if both the algorithm and the adversary move during σ_i, then $\Delta\Phi \leq c \cdot ADV_i - ALG_i$. Summing over all $i = 1, 2, \dots, m$ and using the telescoping property on the differences of the potentials, we get:

$$\Phi_m - \Phi_0 \leq c \cdot ADV(\sigma) - ALG(\sigma)$$

and since Φ_0, Φ_m are bounded by some constants independent of the request sequence, then this immediately gives that ALG is c-competitive, proving the Theorem. □

Both of these techniques have been used extensively in the literature as a simple and elegant way to prove competitive bounds. Since these techniques are pretty much standard, the whole weight of the analysis relies on the definitions of an appropriate potential function that will satisfy the properties of Theorem 2.2 and Theorem 2.3. However, determining what the appropriate potential is can be quite difficult and it relies on a deep understanding of the underlying problem's structure and properties.

As an example, in Chapter 4 we will prove that the Harmonic Algorithm is $O(k \cdot \alpha_k)$-competitive for the generalized k-server problem on uniform metrics using the potential method of interleaving moves. Mathematically speaking, the analysis is quite easy once the correct potential is defined. However, the potential function we use is inspired by the problem's structure and how it translates to random walks on Markov Chains and thus relies on a deep understanding of the problem's structure.

2.3 Paging

The paging (caching) problem is a classical problem in operating systems design [5]. We are given a two-level memory system, composed by the cache and the main memory. The cache is divided into k parts of equal size, called pages, while the main memory has larger capacity, but it is much slower. Whenever a page needs to be accessed by the CPU, it should be in the cache; if it is not, a page fault occurs. The page needs to be fetched in the cache, possibly by evicting some other page. Thus, any operating system needs a page eviction policy. Here, the input can be seen as a sequence of requests to pages, and the goal is to minimize the total number of page faults.

If we see this problem as a classical optimization problem where the entire request sequence is given in advance, it is easy to find the optimal solution: Whenever a page eviction is needed, we evict the page that will be requested the latest in the future.

However, in reality the future requests are not known, and the operating system has to decide which page to evict taking into account only the requests seen so far. An online algorithm for the paging problem needs to decide on what page it evicts at time t, without knowledge of the pages that will be requested at time $t+1, \dots, T$.

In the context of online algorithms, paging was first studied in the seminal work of Sleator and Tarjan [71], where the first results for this problem were presented. In this section, we will present some of these results, since the paging problem has a simple structure that makes it a good candidate for an introduction to competitive analysis. This results are considered folklore in the area of online algorithms, but they will allow us to review some of the main techniques used in competitive analysis.

2.3.1 Deterministic Algorithms for Paging

We begin with the study of deterministic algorithms for this problem. We will first prove a lower bound of k and then present the $MARKING$ algorithm that achieves the same bound.

Theorem 2.4. *The competitive ratio of any deterministic algorithm for the paging problem is at-least k.*

Proof. Let $p_1, p_2, \dots, p_k$ be the pages that are initially in the algorithm's cache memory and let p_{k+1} be a page that is initially outside the algorithm's cache. First, we request page p_{k+1} and since the algorithm is deterministic, we know that it will evict some page p_i. Then we request page p_i and the algorithm evicts some page p_j. We repeat this process for an arbitrary number of requests T, making the algorithm pay a total cost of T.

We will now argue that the optimal solution for this request sequence is at most $\frac{T}{k}$. Recall that the optimal offline algorithm for paging evicts the page that will be requested furthest in the future on page fault. Thus, since the cache has a total size of k, whenever the optimal strategy evicts a page to serve a request, we know that it will surely serve the next $(k-1)$ requests as well. Since the size of the instance is T, the optimal offline algorithm will pay at most $\frac{T}{k}$.

Thus, we get that

$$ALG = T = k \cdot \frac{T}{k} \geq k \cdot OPT$$

and that the algorithm is at least k-competitive. □

We will now present a deterministic algorithm whose competitive ratio matches this lower bound of k, known as the $MARKING$ algorithm. This is a phase based algorithm, where at the beginning of a phase all the pages are *unmarked*. Whenever a request is received, the algorithm marks the requested page (if it isn't already marked) and if it is not in the cache, then it evicts an unmarked page (based on some arbitrary deterministic policy) in order to put it in. If all the pages in the cache are already marked, then a new phase begins and all the pages become unmarked once again.

Theorem 2.5. *The competitive ratio of the $MARKING$ algorithm for paging is at most k.*

Proof. It is actually quite simple to analyze the $MARKING$ algorithm. Fix any phase of the algorithm. Initially, all the pages are unmarked and the phase ends when all the pages in the cache are marked. This immediately gives that the $MARKING$ algorithm evicts exactly k pages per phase and that a phase has to include exactly k distinct requests. Since the cache has a total size of k, we know that any solution (even the optimal) will be forced to evict at least one page per phase. Thus, in every phase the $MARKING$ algorithm evicts k pages and the optimal offline solution evicts at least 1 page, which gives that the competitive ratio of the $MARKING$ algorithm is indeed at most k. □

Theorems 2.4 and 2.5 completely solve the deterministic case, since we have a (computationally efficient) algorithm that achieves the best possible competitive ratio. We note that there are other algorithms that achieve the same competitive ratio of k with specific marking policies, such as LRU, $CLOCK$ and FWF [16].

2.3.2 Randomized Algorithms for Paging

We will now focus on randomized algorithms for the paging problem and show that randomization can indeed improve the competitive ratio of a problem. When studying randomized algorithms, one has to be careful to define the type of adversary. In this section, we will mainly focus on oblivious adversaries.

Theorem 2.6. *Any randomized algorithm for the paging problem is at least H_k competitive in the oblivious adversarial setting, where H_k is the k-th harmonic number.*

Note that since this lower bound holds for the oblivious setting, it clearly extends for adaptive online and adaptive offline adversaries as well.

We will now show that the randomized version of the $MARKING$ algorithm matches this lower bound (up to a constant) against oblivious adversaries. In the randomized version, whenever the $MARKING$ algorithm has to evict a page, instead of picking an unmarked page based on some deterministic eviction policy, it simply evicts an unmarked page at random.

Theorem 2.7. *The competitive ratio of the randomized $MARKING$ algorithm against an oblivious adversary is $2H_k - 1$, where H_k is the k-th harmonic number.*

We note that there are different randomized algorithms that achieve the (optimal) competitive ratio of H_k. For the proofs of Theorems 2.6 and 2.7 we refer the reader to Chapter 4 of [16]. These results show that randomization can reduce the competitive ratio of a problem by an exponential factor, and thus randomized algorithms should be studied for any online problem.

There are many other results for the paging problem, and it is generally considered to by one of the best-understood problems in the online framework. For many other results about different adversarial models and other variations of the paging problem we refer the reader to [41, 16].

2.4 Metrical Task Systems

The first online algorithms were analyzed in a rather ad-hoc way, using problem specific techniques that could not be applied in other problems and didn't contribute much to the field of competitive analysis. In order to create a unifying framework that would make the analysis of online algorithms easier and more systematic, Borodin et al. [17] defined the problem of metrical task systems (MTS) that generalized many of the well-studied online problems of this period.

Definition 2.2 (MTS). *In the MTS problem we are given a server which can be in one of N different states and a metric distance function d specifying the cost of switching between the states. At each time step, a task r arrives, represented by a vector $r = (r_1, \dots, r_N)$, where r_i is the cost of processing r at state i. The server has to decide in which state it will process the task. If it switches from state i to state j and processes the task there, it incurs a cost $d(i, j) + r_j$. Given an initial state and a sequence of tasks, the goal is to process all tasks at minimum cost.*

For MTS, the deterministic competitive ratio is $2N - 1$ [17] and for randomized algorithms the competitive ratio is $O(\log^2 N \log\log N)$ [37] and $\Omega(\log N)$ [17].

While the study of the MTS problem is of interest on its own, its main attribute is that it generalizes many well-studied online problems, more importantly the k-server problem. To see this, any instance of the k-server problem can be viewed as an instance of MTS by appending one state for any configuration of the algorithm's k-servers ($N = n^k$). Then, for any request r_i we set process cost 0 to all the server configurations that serve it and process cost ∞ to all other configurations. Finally, the distance between two states equals the minimum cost perfect matching between the corresponding configurations. Other notable special cases of MTS include fundamental data structure problems such as the list update problem [71, 65, 2] and the binary search tree problem [72, 3, 49].

Thus, the really interesting question is how to exploit the structure of special instances for MTS in order to go beyond the $\Omega(N)$ and $\Omega(\log N)$ bounds for the problem. On this note, in [30, 31, 61] a slight restriction of the MTS model was introduced, called *metrical service systems* (MSS). Here, each component of each task vector is either 0 or ∞. Therefore, each task can be processed only in a subset of the states, whose coefficient is 0 (we call them feasible states for this task).

For MSS, the deterministic competitive ratio is $N-1$ [17] and for randomized algorithms the competitive ratio is $O(\log^2 N \log\log N)$ [37] and $\Omega(\log N)$ [17]. This shows that there is not much improvement when restricting the values of the task vectors.

All of the problems considered in this thesis are special cases of MTS and MSS with special structure, where usually a competitive ratio independent of the number of states is possible. It is easy to see that the paging problem is a special case of both MTS and MSS: states correspond to all possible sets of pages in the cache and the cost of switching between states equals the number of different pages between the corresponding sets. Whenever a page p is requested, all the states that contain p in the cache are feasible and the rest of the states are infeasible. More importantly for this thesis, the generalized k-server problem in arbitrary metrics can also be seen as a special case of both MTS and MSS.

2.5 The k-Server Problem

The k-server problem is one of the most fundamental and extensively studied problems in the theory of online algorithms. As we mentioned in the previous section, this problem is one of the most important special cases of the MTS and MSS problems. The problem was first introduced by Manasse et al. [61] as a far-reaching generalization of various online problems, the most notable of which is the paging problem. In order to solve the k-server problem, many generic and powerful techniques have been devised that have led to many landmark results in the area of competitive analysis.

Definition 2.3 (k-server problem)**.** *Formally, the k-server problem is defined in a metric space $M = (U, d)$, where U is a set of n points and $d : U^2 \to \mathbb{R}$. The fact that M is a metric space means that d is a non-negative and symmetric distance function which satisfies the triangle inequality. There are k distinct servers, initially placed at some points of U. The input is a request sequence $r = r_1, r_2, \dots, r_m$ where $r_t \in U$ is the point requested at time t. To serve the request, some server must move to point r_t. The goal is to minimize the total distance traveled by the servers for serving r.*

As we have already mentioned, the k-server problem is a special case of Metrical Task

Systems with $N = n^k$ stated (one for every server configuration), distance between states that is equal to the minimum matching distance between the corresponding configurations and task vectors of value 0 for configurations that serve the request and 0 otherwise.

The paging problem that we studied in Section 2.3 is also a special case of the k-server problem on uniform metrics of distance 1. Here, the k-servers correspond to the k slots in the cache, and the pages correspond to the points. Evicting a page from the cache and bringing a new one maps to moving a server between the corresponding points at a cost of 1. Equivalently, we can think of a uniform metric as a star graph on n leaves corresponding to the n pages, where all edges have length $\frac{1}{2}$.

In their seminal paper, Manasse et al. [61] showed that the competitive ratio of deterministic algorithms is at least k, even if the metric space contains $n = k + 1$ points. Since for the special case of paging, deterministic algorithms of competitive ratio k are known (for example, the $MARKING$ algorithm), Manasse et al. [61] and many others have conjectured that the true competitive ratio for the k-server problem in the deterministic case is k. In the literature, this is known as the *k-server conjecture*.

Qualitatively, this means that general metrics are believed to be no harder than the simplest possible case of uniform metrics. The k-server conjecture attracted a lot of attention and it has influenced the research on online algorithms over the last three decades.

At the time the k-server conjecture was posed, it was not even known whether a competitive ratio $f(k)$ depending only on k is possible for general metric spaces. The initial research focused on special metrics like weighted stars, lines and trees, and for many cases tight k-competitive algorithms were obtained [29, 17, 39, 57]. For general metric spaces, Fiat et al. [38] obtained the first $f(k)$-competitive algorithm, with competitive ratio $O((k!)^3)$. Several improvements followed [6, 48, 14], but the ratio was still exponential in k.

In a breakthrough result, Koutsoupias and Papadimitriou [56] showed that the Work Function Algorithm (WFA) is $(2k-1)$-competitive for every metric space, almost resolving the conjecture. This remains up to date the best known upper bound on the competitive ratio of the k-server problem. At time t, the WFA algorithm solves (optimally) for any configuration $q \in [n]^k$ the problem of serving requests $r_1, \ldots, r_t$ and ending up in configuration q. Then, it moves to the configuration that has the lowest total cost. In the special case of $n = k + 1$, the competitive ratio of the WFA is known to be k, matching the lower bound for the problem. We refer the reader to [16, 13] for an extensive treatment of the large body of work on the k-server conjecture.

An enormous amount of research has also been done on randomized algorithms for the k-server problem. Typically, using randomization, an exponential improvement on the competitive ratio is possible. However, randomized online algorithms are far less understood compared to the deterministic ones. For the k-server problem, it is believed that, similarly to the deterministic case, the distance function does not affect the competitive ratio and an $O(\log k)$-competitive randomized algorithm against oblivious adversaries is possible in any metric space. In the literature, this is known as the *randomized k-server* *conjecture*

However, this is known to be true only for very special cases. In particular, for the paging problem several $O(\log k)$-competitive randomized algorithms are known [1, 36, 62]. Nevertheless, even the simple generalization of the weighted paging problem (which corresponds to the k-server problem on weighted star graphs) remained open for almost two decades, until Bansal et al. [9] gave an $O(\log k)$-competitive algorithm using the primal-dual method.

More recently, $polylog(k, n)$ competitive ratios for general metric spaces were obtained [7, 19]. Those bounds are better than the deterministic competitive ratio $2k - 1$ in case n is sub-exponential in k. The techniques developed in those works imply an $O(\log k)$-competitive randomized algorithm for hierarchically separated trees (HSTs) of constant depth. Recently, an $O(\log^6 k)$-competitive algorithm for any metric space was claimed [60].

2.6 The Generalized k-Server Problem

The study of the k-server problem has been essential in the development of powerful techniques for online algorithms. For example, the landmark result of Koutsoupias and Papadimitriou [56] on the k-server conjecture enabled the belief that the WFA (or the generalized WFA [23]) performs optimally for any metrical task system. Furthermore, the work on randomized k-server algorithms enabled the development of powerful techniques using the primal-dual method [22, 21, 8, 7] and more recently the mirror descent method [19].

Despite this progress, several natural variants and generalizations of the k-server problem are very poorly understood. In particular, they exhibit very different and intriguing behavior and the techniques for the standard k-server problem do not seem to apply to them. Getting a better understanding of such problems is a natural step towards building a deeper theory of online computation. Below we list some examples of server problems that are not captured by the standard k-server model:

- **The weighted k-server problem** [63]. Here, each server has a different weight $w_1, \ldots, w_k$ and the cost of moving the i-th server by distance d is $w_i \cdot d$. This problem is substantially different from the (unweighted) k-server problem. To get a feel for the problem, for uniform metrics the competitive ratio is $2^{2^{\Theta(k)}}$ [40, 27, 10], and no competitive algorithms are known for general metrics.

- **The CNN problem** [58]. In this problem we are given two servers in the euclidean plane, the one moving in the horizontal axis and the other in the vertical axis. At each time step a point (r_1, r_2) is requested, and in order to serve the request we should either move the horizontal server to point $x = r_1$ or the vertical server to $y = r_2$. This problem models the movement of the crew of a news network in

Manhattan: whenever an event occurs, a camera should be either in the same street or in the same avenue.

Motivated by all those variants of the k-server problem, Koutsoupias and Taylor [58] introduced a substantial generalization of the k-server problem, called the generalized k-server problem. Here, each server s_i lies in its own metric space M_i , with its own distance function d_i . A request is a k-tuple $r = (r_1, r_2, \dots, r_k)$ and must be served by moving some server s_i to the point $r_i \in M_i$.

Note that the standard k-server problem corresponds to the special case when all the metrics are identical, $M_1 = M_2 = \dots = M_k = M$, and the requests are of the form $(r, r, \dots, r)$, i.e., the k-tuple is identical in each coordinate. Similarly, the weighted k-server problem corresponds to the case when the metric spaces are scaled copies of each other, i.e. $M_i = w_1 \cdot M$ for some fixed M, and the requests have the form $(r, r, \dots, r)$. Finally, the CNN problem corresponds to the case where $k = 2$ and both M_1, M_2 are lines.

Sitters [68] highlights that the existence of an $f(k)$-competitive algorithm is among the most intriguing and important open problems in online computation. Despite the intense interest, this problem is poorly understood.

2.6.1 Results on the generalized k-server problem

Initially, competitive algorithms were known only for the special case of $k = 2$ [70, 68]. For an arbitrary number of servers k, all of the known results apply to the special cases of uniform metric spaces or uniform metric spaces of equal weights.

For deterministic algorithms, Koutsoupias and Taylor showed that even when the metric spaces have $n = 2$ points and are uniform, the competitive ratio of any deterministic algorithm is at least $2^k - 1$. For general metric spaces, the best known deterministic lower bound is $2^{2^{\Omega(k)}}$ [10], and comes from the weighted k-server problem. This also means that this lower bound also applies to the special case of weighted uniform metric spaces.

The first $f(k)$-competitive algorithms for the generalized k-server problem were given by Bansal et.al. [11]. In their work, they presented an $O(k2^k)$-competitive deterministic algorithm for uniform metric spaces. The basic idea of their algorithm is to partition the request sequence to phases, and keep the invariant that the configurations of their algorithm serve the entire (observed) request history of the current phase. A phase ends when there isn't a configuration that serves all the request in the previous phase, which means that even the optimal solution moves at least once per phase. By showing that the number of distinct requests in a phase can be at most 2^k, they proved their result.

In the same work, they also presented a $2^{2^{O(k)}}$-competitive deterministic algorithm for weighted uniform metric spaces as a natural modification of an algorithm due to Fiat and Ricklin [40] for the weighted k-server on uniform metrics.

Those are the only known results for deterministic algorithms and are (almost) optimal due to the already existing lower bounds.

Some work has also been done for randomized algorithms. In [11], Bansal et.al. also gave an $O(k^3 \log k)$-competitive randomized algorithm for uniform metric spaces against oblivious adversaries. This was then improved to $O(k^2 \log k)$ by Bienkowski et. al. [15], who also gave an $\Omega(k)$ lower bound for randomized algorithms on the uniform metric case. All of these results applied to the oblivious adversarial setting.

Until this point, memoryless algorithms for the generalized k-server problem had not been studied; in fact, all of the algorithms that achieve the aformentioned results heavily rely on memory.

2.6.2 Results on memoryless algorithms

Very recently, Chiplunkar and Vishwanathan [28] studied memoryless algorithms for the generalized k-server on weighted uniform metric spaces and showed tight $\approx 2^{2^k}$ bounds for any memoryless algorithm. In their work, they defined a potential function that needs to satisfy some properties in the form of a linear program. Then, by studying the constraints of this problem they were able to prove the existence of a potential that led to a doubly-exponential upper bound. Interestingly, the memoryless algorithm that achieves this bound is not the harmonic, i.e., the algorithm that serves each metric space with probability that is inversely proportional to its weight. With similar arguments, they also showed that their analysis was tight; the competitive ratio achieved by their algorithm is the optimal for memoryless algorithms. All of their results apply to adaptive adversaries.

In this section, we will present the main techniques used in their work in order to highlight how their results were achieved and how our own analysis differs from theirs.

Model. Recall that in the generalized k-server problem on weighted uniform metric spaces, each of the k metrics M_i has a weight w_i, which is the cost of moving a server on M_i. Any memoryless algorithm for this problem is fully characterized by a probability distribution $p_1, \ldots, p_k$ on the metric spaces; p_i is the probability that a request is served by moving a server on M_i. Throughout their work, they assume that $p_1 \geq p_2 \ldots \geq p_k > 0$.

At any time t, let $s_i \in [n]$ denote the position of the algorithm's server in metric space M_i and $a_i \in [n]$ denote the position of the adversary's server in metric space M_i. Since the metric spaces are uniform, the *state* of the algorithm is captured by the metric spaces where its server and the adversary's server occupy the same point, that is $S = \{i : a_i = s_i\}$. Thus, the state space for this problem is $2^{[k]}$ (since $S \in 2^{[k]}$).

Potential A *potential function* for this problem is a mapping $\Phi : 2^{[k]} \rightarrow \mathbb{R}$. Instead of directly defining an appropriate potential, Chiplunkar and Vishwanathan write down the conditions of the Interleaving Moves method (see Section 2.2) that have to be satisfied in order to get an upper bound of γ.

Recall that in the Interleaving Moves method, we wish to prove the following two properties:

1. If only the adversary moves and pays x, the potential increases by at most $\gamma \cdot x$.

2. If only the algorithm moves and pays x, the potential decreases by at least x.

As we have shown, these two conditions directly imply an upper bound of γ. We will now see how these conditions translate to the generalized k-server problem on weighted uniform metrics. Without loss of generality, we can assume that the adversary always makes requests that are not served by the algorithm's current configuration, forcing it to constantly move.

Assume that the adversary moves in some metric M_i to serve a request r by setting $a_i = r_i$. If $i \notin S$ (the algorithm's server in M_i isn't in point s_i) then the state doesn't change since $r_i \neq s_i$ by assumption. Thus, the potential doesn't increase. If $i \in S$, then the new state of the algorithm is $S \setminus \{i\}$ and the adversary pays w_i. Thus, condition 1 translates to the following:

$$\Phi(S \setminus \{i\}) - \Phi(S) \leq \gamma \cdot w_i \quad \text{for all } S \subseteq [k],\ i \in S \tag{2.1}$$

Now assume that the algorithm receives request r and has to move. Since the adversary has already served the request and the algorithm doesn't serve it, $r_i = a_i$ for some $i \notin S$. We distinguish between the following three cases:

- If the algorithm moves on metric M_i, then the new state becomes $S \cup \{i\}$.
- If the algorithm moves on metric M_j for some $j \notin S \cup \{i\}$, then the state doesn't change.
- If the algorithm moves on metric M_j for some $j \in S$, then the new state becomes $S \setminus \{j\}$.

Thus, on expectation the decrease in potential is:

$$p_i(\Phi(S) - \Phi(S \cup \{i\})) + \sum_{j \notin S \cup \{i\}} p_j(\Phi(S) - \Phi(S)) + \sum_{j \in S} p_j(\Phi(S) - \Phi(S \setminus \{j\}))$$

while the expected cost of the algorithm is $\sum_{j=1}^{k} p_j w_j$. Thus, condition 2 translates to the following:

$$p_i(\Phi(S) - \Phi(S \cup \{i\})) - \sum_{j \in S} p_j(\Phi(S \setminus \{j\}) - \Phi(S)) \geq \sum_{j=1}^{k} p_j w_j \quad \text{for all } S \subseteq [k],\ i \notin S \tag{2.2}$$

Any potential Φ and any γ that satisfy conditions (2.1) and (2.2) directly imply an upper bound of γ for the problem. By changing variables

$$\phi(S) = -\frac{\Phi(S)}{\sum_{j=1}^{k} p_j w_j}$$

we can write the optimal upper bound achieved by this method as the solution of the following linear program:

Minimize γ subject to:

For all $S \in 2^{[k]}$ and $i \in S$:

$$\gamma \geq (\sum_{j=1}^{k} p_j w_j) \frac{\phi(S) - \phi(S \setminus \{i\})}{w_i} \tag{2.3}$$

For all $S \in 2^{[k]}$ and $i \notin S$:

$$p_i(\phi(S \cup \{i\}) - \phi(S)) - \sum_{j \in S} p_j(\phi(S) - \phi(S \setminus \{j\})) \geq 1 \tag{2.4}$$

$$\phi(\emptyset) = 0 \tag{2.5}$$

where (w.l.o.g.) we have set $\phi(\emptyset) = 0$.

Then, they proceed by determining a feasible solution for this linear program. For that purpose, they focus on constraints (2.4) and (2.5). This is a set of approximately $k2^k$ linear constraints on 2^k variables. One of their key contributions is to show that by focusing on 2^k of this constraints *as equations*, the solution of this system satisfies all of the constraints. On a high level, they focus on the "hard" constraints that directly imply all the others. This constraints are the following:

$$p_i(\phi(S \cup \{i\}) - \phi(S)) - \sum_{j \in S} p_j(\phi(S) - \phi(S \setminus \{j\})) = 1 \quad \forall S \neq [k],\ i = \min([k] \setminus S) \tag{2.6}$$

with $\phi(\emptyset) = 0$.

Theorem 2.8. *The solution of* (2.6) *exists for any distribution* p *and weights* w *and satisfies constraints* (2.4) *and* (2.5)*.*

Proof. This result relies on a rather technical method called the Gauss-Seidel Trick. Using the matrix form of linear system (2.6) and the fact that is is strictly diagonally dominant, one can prove convergence of an iterative algorithm that computes its solution (and thus the existence of this solution). Then, using a monotonicity property on this solutions, we are able to prove that the chosen constraints are in fact the “hardest”, in the sense all the other constraints are also satisfied by the solution. □

Having achieved feasibility, they then focus on proving specific properties and bounds for the solutions of (2.6). We quote the most important ones:

Theorem 2.9 (Supermodularity)**.** *For any* $S \subseteq 2^{[k]}$ *and* $i, j \notin S$*, the solutions of* (2.6) *satisfy*

$$\phi(S \cup \{i\}) + \phi(S \cup \{j\}) \leq \phi(S \cup \{i, j\}) + \phi(S)$$

Theorem 2.10 (Bounds). *For any $S \subseteq 2^{[k]}$ and $i = \min([k] \setminus S)$, the solutions of* (2.6) *satisfy*

$$p_i(\phi(S \cup \{i\}) - \phi(S)) \leq C_S$$

where the constants C_S are defined recursively by the formula $C_\emptyset = 1$ and $C_S = 1 + \sum_{j \in S} C_{S \setminus \{j\} \cup [j-1]}$.

We are now ready to determine the upper bound. From constraints (2.3), the best upper bound (when the potential is defined as the solution of (2.6)) is given by

$$\gamma = (\sum_{j=1}^{k} p_j w_j) \max_{S,\, i \in S} \frac{\phi(S) - \phi(S \setminus \{i\})}{w_i}$$

Due to Theorem 2.9, this expression becomes:

$$\gamma = (\sum_{j=1}^{k} p_j w_j) \max_{i \in [k]} \frac{\phi([k]) - \phi([k] \setminus \{i\})}{w_i} = (\sum_{j=1}^{k} p_j w_j) \max_{i \in [k]} \frac{p_i(\phi([k]) - \phi([k] \setminus \{i\}))}{p_i w_i}$$

and by Theorem 2.10, we get

$$\gamma \leq (\sum_{j=1}^{k} p_j w_j) \max_{i \in [k]} \frac{C_{[k] \setminus \{i\}}}{p_i w_i}$$

Thus, by selecting $p_i = \frac{C_{[k] \setminus \{i\}}}{w_i}$, the achieved competitive ratio is upper bounded by:

$$\sum_{j=1}^{k} C_{[k] \setminus \{i\}} = C_{[k]} - 1 = \gamma_k$$

where γ_k is the solution to the recursion $\gamma_k = \gamma_{k-1}^2 + 3\gamma_{k-1} + 1$ with initial condition $\gamma_1 = 1$.

By the definition of γ_k, it is not hard to prove that this upper bound is doubly-exponential to k. Of course, this bound is achieved by a specific potential and specific probabilities $p_i = \frac{C_{[k] \setminus \{i\}}}{w_i}$. Changing the potential or the algorithm could give a better upper bound for the problem. However this is not the case, as they prove next.

Specifically, they complement their upper bound result by proving that γ_k is a lower bound for the competitive ratio of any memoryless algorithm for the generalized k-server problem on weighted uniform metric spaces. In fact, they study the weighted k-server problem, which is a special case of the generalized k-server problem on weighted uniform metric spaces where the requests are in the form of $(r, \ldots, r)$. Clearly, any lower bound for weighted k-server immediately transfers to generalized k-server in weighted uniform metrics. In order to get the lower bound, they construct an adaptive online adversary and by using many of their results shown for the upper bound, they are able to show that their analysis was tight and that γ_k was in fact the optimal competitive ratio for their setting.

This work aims to study the case of memoryless algorithms for uniform metrics and improve the doubly-exponential bound of Chiplunkar and Vishwanathan to a factorial bound. This is mostly done by using a different analysis that heavily relies on specific classes of Markov Chains that we can study. Furthermore, we extend some of our results to the oblivious case, which had not been done before for memoryless algorithms for the generalized k-server problem.

Chapter 3

Random Walks

In this chapter, we provide the reader with the background on *random processes* that is necessary for the analysis of memoryless algorithms for the generalized k-server problem. We focus on the stochastic model of *Markov Chains* and study the properties of a *random walk* on this model and several of its sub-classes.

In Section 3.1, we give the formal definition of a Markov Chain and state the basic properties of this model. Then, in Section 3.2 we focus on a special type of Markov Chains, called *Birth-Death Chains*, and prove Theorem 3.3 that states the *Expected Extinction Time* of a random walk on this model. Finally, in Sections 3.3 and 3.4 we define two novel sub-classes of Birth-Death Chains that we refer to as *Harmonic* and *Binary Chains* and compute their Expected Extinction Times in Theorems 3.4 and 3.5 respectively. As we show in Chapter 4, these random processes capture the competitive ratio of the Harmonic Algorithm for the generalized k-server problem and thus, they will be used extensively in our analysis.

3.1 Markov Chains

A *random* (or stochastic) *process* [66] can be viewed as a sequence of random variables that evolve in some random but prescribed way. A typical random process X is a family $\{X_t : t \in T\}$ of random variables that are indexed by some set T. If T is a discrete set (e.g. the set of integers $\mathbb{Z}$) then the process is called a "discrete-time" process. The set $\mathcal{X} = \{i : X_t = i\}$ of all the possible values of X_t is called the *state space* of the random process. We focus on random processes with discrete state spaces.

In the context of this thesis, we are interested in a special class of discrete-time and discrete-space random processes called **Markov Chains** [46, 47]. A Markov Chain is a random process where the probability of an event depends only on the state attained in the previous event. Formally:

Definition 3.1 (Markov Chain)**.** *A Markov Chain is a random process X of discrete state space S and discrete index set $T = \{0, 1, 2, \dots\}$ that satisfies the following Markov* *condition:*

$$\mathbb{P}(X_t = s | X_0 = x_0, X_1 = x_1, \ldots, X_{t-1} = x_{t-1}) = \mathbb{P}(X_t = s | X_{t-1} = x_{t-1})$$

for all $t \geq 1$ *and* $s, x_0, \ldots, x_{t-1} \in \mathcal{X}$.

Markov Chains, named after the Russian mathematician Andrey Markov, have been extensively studied in the literature due to their nice properties and their applications [75, 24, 78] in numerous fields such as economics, statistics, physics, biology, information theory and machine learning.

We are interested in *homogeneous* Markov Chains, where the probabilities of transitioning from some state $s \in \mathcal{X}$ to some state $s' \in \mathcal{X}$ don't depend on time.

Definition 3.2 (Homogeneous Markov Chain)**.** *A Markov Chain X is called* ***homogeneous*** *if*

$$\mathbb{P}(X_t = s' | X_{t-1} = s) = \mathbb{P}(X_1 = s' | X_0 = s)$$

for all $t \geq 1$ *and* $s, s' \in \mathcal{X}$.

A homogeneous Markov Chain can be fully characterized by an **initial state** $X_0 \in \mathcal{X}$ and a **transition matrix** $P = [p_{ij}]$ such that

$$\mathbb{P}(X_t = j | X_{t-1} = i) = p_{ij}$$

for all $t \geq 1$. From now on, unless stated otherwise, when we refer to a Markov Chain we will assume that it is homogeneous.

We will now proceed to classify the states of a Markov Chain based on the transition matrix P of the chain.

Definition 3.3 (Recurrent and Transient States)**.** *A state $i \in \mathcal{X}$ is called* ***recurrent*** *(or persistent) if*

$$\mathbb{P}(\exists t > 0 : X_t = i | X_0 = i) = 1$$

which is to say that the probability of eventual return to state i, having started from it, is 1. *If this probability is strictly less than* 1, *then the state i is called* ***transient***.

In order to give another useful classification of the states, we need to define the following quantity:

Definition 3.4 (Mean Recurrence Time)**.** *For any state $j \in \mathcal{X}$, let T_j be the random variable defined as $T_j = \min\{t \geq 1 : X_t = j\}$, that is the time of the first visit to state j. Then, the* ***mean recurrence time*** *of state j is defined as*

$$\mu_j = \mathbb{E}(T_j | X_0 = j),$$

that is the expected time to visit j for the first time, starting from it.

Based on the mean recurrence time of a recurrent state, we can classify it as either *null* or *positive*.

Definition 3.5 (Null and Positive States)**.** *A recurrent state $i \in \mathcal{X}$ is called* ***null*** *if $\mu_i = \infty$; if $\mu_i < \infty$ then it is called* ***positive****.*

Finally, we need to define an *irreducible* set of states.

Definition 3.6 (Irreducible Set of States)**.** *A set of states $S \subseteq \mathcal{X}$ is called* ***irreducible*** *if for any states $i, j \in S$, the probability that we visit state j starting from state i is non-zero. If the state space $\mathcal{X}$ is irreducible, then we say that the Markov Chain is irreducible.*

We are now ready to answer the question of how does a Markov Chain behave after a long time has elapsed. One of the most important properties of Markov Chains is that under mild assumptions on their structure, the probability of being in a state $i \in S$ as $t \to \infty$ converges to a fixed value that depends only on the transition matrix P and not on the initial state.

Definition 3.7 (Stationary Distribution)**.** *The vector π is called a* ***stationary distribution*** *of a Markov Chain with state space X and transition matrix P if it has entries $(\pi_j : j \in \mathcal{X})$ such that:*

1. $\pi_j \geq 0 \, \forall j \in \mathcal{X}$,
2. $\sum_{j \in \mathcal{X}} \pi_j = 1$,
3. $\pi = \pi P$.

Finally, we conclude with the following theorem that specifies the conditions under which a Markov Chain has a stationary distribution and associates its mean recurrence times with its stationary distribution.

Theorem 3.1. *An irreducible chain has a stationary distribution π if and only if all the states are positive recurrent; in this case, π is the unique stationary distribution given by $\pi = \pi P$ and it holds that $\pi_j = \frac{1}{\mu_j}$ for any state $j \in \mathcal{X}$.*

3.2 Random Walks on Birth-Death Chains

For the purposes of this thesis, we only need to study a special class of Markov Chains called *Birth-Death Chains* [76]. This is a type of chain that limits transitions from a state only to its "adjacent" states and is characterized by two terminal states $X = 0$ and $X = k$ from which there is only one possible next state.

Definition 3.8 (Birth-Death Chain)**.** *A Markov process with state-space $\mathcal{X} = \{0, 1, \ldots, k\}$ for some $k \in \mathbb{N}$ is characterized as a* ***Birth-Death Chain*** *if its transition matrix $P = [P_{ij}]$ has the following form:*

$$P_{ij} = \begin{cases} p_i & ,j = i+1 \\ q_i & ,j = i-1 \\ 1 - p_i - q_i & ,j = i \\ 0 & ,otherwise \end{cases} \qquad \forall i, j \in \mathcal{X}$$

where $q_0 = 0$ and $p_k = 0$ for the end-points of the chain. Equivalently, a Birth-Death Chain can be defined as a finite Markov Chain with three-diagonal transition matrix. A graphical representation of a Birth-Death Chain is given in Figure 3.1.

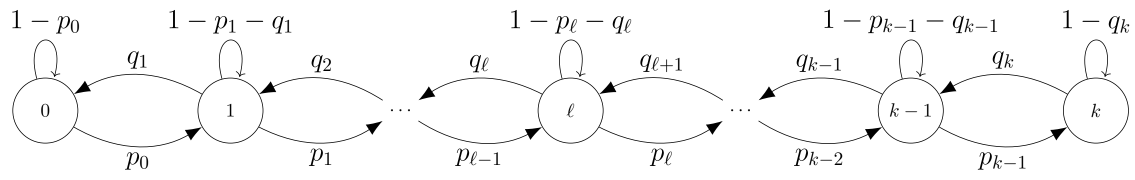


Figure 3.1: A Birth-Death Chain

We refer to q_i as the *forward* probability of state i and to p_i as the *backward* probability of state i. Furthermore, a Birth-Death Chain will be called **absorbing** on the state $X = 0$ if $p_0 = 0$, which means that the random process will remain on the state $X = 0$ if it ever reaches it. We are interested in Birth-Death Chains where $q_i, p_i > 0$ for all $i \in \{1, \dots, k-1\}$ and $q_k > 0$; in that case, we know that with probability 1 the state 0 will be visited at some time [59].

This chain's name is due to the fact that its state space can be used to model the size of a population of samples; in that case, the forward probabilities correspond to the death of a sample and the backward probabilities correspond to the birth of a sample. If state $X = 0$ is reached, then everyone in the population is dead and obviously no one else can die. Furthermore, we assume that there are some limitations that don't allow the number of samples in the population to exceed some fixed size $k \in \mathbb{N}$. Finally, the state $X = 0$ is usually considered to be absorbing, since a population without members can't reproduce to increase its size. Due to this natural interpretation, the model of Birth-Death Chains has applications in many different areas such as biology [64, 67], queuing systems [26] and economics [73].

We will now proceed to compute the stationary distributions and the Expected Extinction Time of a Birth-Death Chain. While these results are by no means novel (e.g. [51]), the common definition of a Birth-Death Chain considers a (countably) infinite state space $\mathcal{X} = \mathbb{N}$ while we are interested in Birth-Death Chains with finite state spaces. It is thus useful to prove the following results instead of just stating them.

3.2.1 Stationary Distribution of Birth-Death Chains

We begin by computing the stationary distribution of any Birth-Death chain. For this section we will assume that none of the states are absorbing; that is $q_i, p_i > 0$ for all $i \in \{0, 1, \dots k\}$. Under this assumption, it is clear that all the states are positive recurrent and the chain is irreducible. Thus, from Theorem 3.1 we know that the stationary distribution π of the chain exists and is unique.

Theorem 3.2. *For any Birth-Death Chain with state space $\mathcal{X} = \{0, 1, \dots, k\}$ such that the forward and backward probabilities satisfy $p_i, q_i > 0$, the stationary distribution is given* *by*

$$\pi_0 = \frac{1}{1 + \sum_{\ell=1}^{k} \frac{p_0 p_1 \cdots p_{\ell-1}}{q_1 q_2 \cdots q_\ell}}$$

on state 0 *and*

$$\pi_\ell = \frac{p_{\ell-1} p_{\ell-2} \cdots p_0}{q_\ell q_{\ell-1} \cdots q_1} \pi_0$$

on any other state $\ell \in \mathcal{X} \setminus \{0\}$.

Proof. We could compute the distribution π by solving the linear system of equations $\pi = \pi P$, where P is the transition matrix of the Birth-Death Chain. However, the analysis can be simplified by a simple argument that exploits the special structure of those chains.

Fix any two adjacent states $i, (i+1) \in \mathcal{X}$. As $t \to \infty$, we know that the probability of being in a state converges to the corresponding probability of the stationary distribution, and thus $X_t = i$ with probability π_i and $X_t = i+1$ with probability π_{i+1}. Now, imagine cutting the chain between those two states. As the random process evolves, the number of times that the cut is crossed from the left to the right (with probability $\pi_i p_i$) must become equal to the number of times that the cut is crossed from the right to the left (with probability $\pi_{i+1} q_{i+1}$). This simple observation gives that for all $i \in \{0, 1, \ldots, k-1\}$:

$$\pi_i p_i = \pi_{i+1} q_{i+1}$$

By recursively solving the above equation, we immediately get that

$$\pi_\ell = \frac{p_{\ell-1} p_{\ell-2} \cdots p_0}{q_\ell q_{\ell-1} \cdots q_1} \pi_0 \qquad \forall \ell \in \{1, 2, \ldots, k\} \tag{3.1}$$

Combining (3.1) with the fact that $\sum_{\ell=0}^{k} \pi_\ell = 1$, we get that

$$\pi_0 = \frac{1}{1 + \sum_{\ell=1}^{k} \frac{p_0 p_1 \cdots p_{\ell-1}}{q_1 q_2 \cdots q_\ell}} \tag{3.2}$$

With equations (3.1) and (3.2) we can compute π_ℓ for any $\ell \in \mathcal{X}$ and the theorem follows. □

3.2.2 Expected Extinction Time of Birth-Death Chains

We are now ready to formally define and compute the Expected Extinction Time (EET) of a Birth-Death Chain with an absorbing state $X = 0$; this is in fact the purpose of this chapter - to derive a general expression of this quantity and then apply this result to specific types of Birth-Death Chains that are deeply tied with the competitive ratio of the Harmonic Algorithm for the generalized k-server problem on uniform metrics.

If a Birth-Death Chain has an absorbing state $X = 0$, then it is natural to want to compute the expected time that this state is reached. Clearly, this quantity depends on the initial state of the chain. If the states of the chain represent the size of a population, then this can be seen as the expected time that the population becomes extinct; thus the name Expected Extinction Time.

Definition 3.9 (Expected Extinction Time). *Consider a Birth-Death Chain with state space $\mathcal{X} = \{0, 1, \ldots k\}$ and a single absorbing point $X = 0$ ($p_0 = 0$). Then, for any state $\ell \in \mathcal{X}$, the Expected Extinction Time (EET) of the chain is defined as:*

$$h(\ell) = \mathbb{E}(N|X_0 = \ell)$$

where $N = \min_{t\geq 0}\{X_t = 0\}$.

For the rest of this section we show how to compute the EET of a Birth-Death Chain and prove the following theorem:

Theorem 3.3. *For any Birth-Death Chain with states $\mathcal{X} = \{0, 1, \ldots k\}$ and absorbing state $X = 0$, the Expected Extinction Time of any initial state $\ell \in \mathcal{X} \setminus \{0\}$ is given by:*

$$h(\ell) = \frac{1}{q_1} + \sum_{i=2}^{k} \frac{p_1 \cdots p_{i-1}}{q_1 \cdots q_i} + \sum_{i=1}^{\ell-1} \left(\frac{q_1 \cdots q_i}{p_1 \cdots p_i} \sum_{j=i+1}^{k} \frac{p_1 \cdots p_{j-1}}{q_1 \cdots q_j} \right)$$

with $h(0) = 0$.

Proof. Using Bayes' Rule [77] on the conditional expectation [74] $h(\ell)$, we get that for any $\ell \in \mathcal{X}$:

$$h(\ell) = 1 + \sum_{j\in\mathcal{X}} p_{\ell j} h(j) = 1 + q_\ell h(\ell-1) + p_\ell h(\ell+1) + (1 - q_\ell - p_\ell) h(\ell)$$

and by re-arranging this equation, we get:

$$\left\{ \begin{array}{c} (p_\ell + q_\ell) h(\ell) = 1 + q_\ell h(\ell-1) + p_\ell h(\ell+1) \quad \forall \ell \in \{1, 2, \ldots, k\} \\ h(0) = 0 \end{array} \right\} \tag{3.3}$$

Equation (3.3) is a second-order non-homogeneous linear recurrence relation with variable coefficients [45]. First of all, in order to solve it we need a second initial condition $h(1)$; we will use the properties of the Markov Chain in order to compute it. Then, we can solve this recurrence relation on the differences $\Delta(\ell) = h(\ell+1) - h(\ell)$ instead of $h(\ell)$; this reduces it to a first-order non-homogeneous linear recurrence relation with variable coefficients that is much easier to solve.

Computation of initial conditions. A second-order recurrence relation typically requires two initial conditions to be solved; however (3.3) only has $h(0) = 0$. Thus, we need a way to compute $h(1)$ before trying to solve the recursion.

This can be done by using the stationary distribution of a Birth-Death Chain similar to the one we study. Since state $X = 0$ is absorbing ($p_0 = 0$), the stationary distribution is not defined for our chain. However, imagine a different Birth-Death Chain that is exactly the same but with $p_0 = 1$. Then, this chain is clearly irreducible and by Theorem 3.2 the probability of the stationary distribution on state $X = 0$ is

$$\pi_0 = \frac{1}{1 + \sum_{\ell=1}^{k} \frac{p_0 p_1 \cdots p_{\ell-1}}{q_1 q_2 \cdots q_\ell}}$$

By Theorem 3.1, this immediately gives that the first recurrence time of state 0 is

$$\mu_0 = \pi_0^{-1} = 1 + \sum_{\ell=1}^{k} \frac{p_1 \cdots p_{\ell-1}}{q_1 q_2 \cdots q_\ell}$$

since $p_0 = 1$.

Observe that $\mu_0 = 1 + h(1)$. To see this, since $p_0 = 1$ we know that the first transition from state 0 will be to state 1. Then, we can set $p_0 = 0$ like before and the (expected) time until state 0 is reached again is given by $h(1)$. This observation immediately yields

$$h(1) = \sum_{\ell=1}^{k} \frac{p_1 \cdots p_{\ell-1}}{q_1 q_2 \cdots q_\ell} \tag{3.4}$$

and now that we have a second initial condition, we can proceed to solving recurrence relation (3.3).

Solving the recurrence relation. As we mentioned, instead of solving the second-order recurrence relation (3.3), we can solve a first-order recurrence relation. Define

$$\Delta(\ell) = h(\ell+1) - h(\ell)$$

for all $\ell \in \{0, 1, \ldots, k-1\}$. Then, (3.3) can be written as:

$$\left\{ \begin{array}{c} p_\ell \Delta(\ell) = 1 + q_\ell \Delta(\ell-1) \quad \forall \ell \in \{1, 2, \ldots, k-1\} \\ \Delta(0) = h(1) \end{array} \right\} \tag{3.5}$$

This is a first-order non-homogeneous linear recurrence relation with variable coefficients that is much easier to solve. Indeed, observe that:

$$\begin{aligned} \Delta(\ell) &= \frac{1}{p_\ell} + \frac{q_\ell}{p_\ell}\Delta(\ell-1) \\ &= \frac{1}{p_\ell} + \frac{q_\ell}{p_\ell p_{\ell-1}} + \frac{q_\ell q_{\ell-1}}{p_\ell p_{\ell-1}}\Delta(\ell-2) \\ &= \frac{1}{p_\ell} + \frac{q_\ell}{p_\ell p_{\ell-1}} + \frac{q_\ell q_{\ell-1}}{p_\ell p_{\ell-1} p_{\ell-2}} + \frac{q_\ell q_{\ell-1} q_{\ell-2}}{p_\ell p_{\ell-1} p_{\ell-2}}\Delta(\ell-3) \\ &\vdots \\ &= \sum_{i=2}^{\ell+1} \frac{q_\ell q_{\ell-1} \cdots q_i}{p_\ell p_{\ell-1} \cdots p_{i-1}} + \frac{q_1 \cdots q_\ell}{p_1 \cdots p_\ell}\Delta(0). \end{aligned} \tag{3.6}$$

Finally, since $\Delta(\ell) = h(\ell+1) - h(\ell)$, using telescoping sums and the fact that $h(0) = 0$ we have $h(\ell) = \sum_{i=0}^{\ell-1} \Delta(i)$. Combining this with equations (3.4) and (3.6), the theorem follows. □

3.3 Random Walks on the Harmonic Chain

In this section, we define and compute the EET of a special type of Birth-Death Chain we refer to as the *Harmonic Chain* and we will use in the analysis of the Harmonic Algorithm for the generalized k-server problem.

Definition 3.10. *The* ***Harmonic Chain*** *is a class of Birth-Death Chains with state space $\mathcal{X} = \{0, 1, \dots, k\}$ for some $k \in \mathbb{N}$, absorbing state $X = 0$, forward probabilities $q_\ell = \frac{1}{k}$ and backward probabilities $p_\ell = \frac{k-\ell}{k}$. A graphical representation of a Harmonic Chain is given in Figure 3.2.*

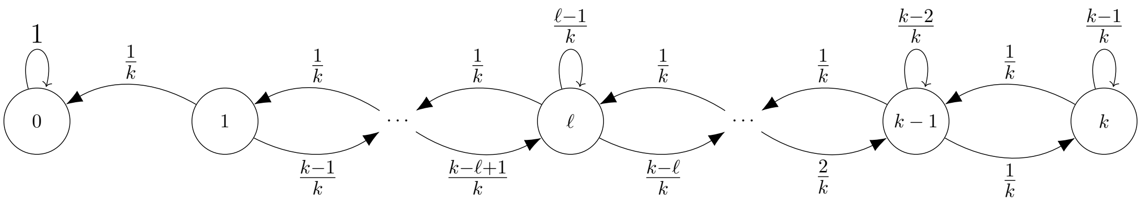


Figure 3.2: The Harmonic Chain

Theorem 3.4. *The Expected Extinction Time of a Harmonic Chain with state space $\mathcal{X} = \{0, 1, \dots, k\}$ for some $k \in \mathbb{N}$ on any initial state $\ell \in \mathcal{X}$ is given by*

$$h(\ell) = k \sum_{i=k-\ell+1}^{k} \alpha_i$$

where α_n is the ***harmonic recursion*** *defined as the solution of $\alpha_n = 1 + (n-1)\alpha_{n-1}$ with $\alpha_1 = 1$.*

Proof. For $q_i = \frac{1}{k}$ and $p_i = \frac{k-i}{k}$, from Theorem 3.3 we get that for any $\ell \in \{1, \dots, k\}$:

$$h(\ell) = k + k! \sum_{j=0}^{k-2} \frac{1}{j!} + k \sum_{i=k-\ell}^{k-2} i! \sum_{j=0}^{i} \frac{1}{j!} \tag{3.7}$$

We need a convenient way to write the sums of inverse factorials. By definition of the harmonic recursion, we have:

$$\alpha_\ell = 1 + (\ell-1)\alpha_{\ell-1} = 1 + (\ell-1) + (\ell-1)(\ell-2)\alpha_{\ell-2} = \ldots = (\ell-1)! \sum_{i=0}^{\ell-1} \frac{1}{i!}$$

which gives

$$\sum_{i=0}^{\ell} \frac{1}{i!} = \frac{\alpha_{\ell+1}}{\ell!} \tag{3.8}$$

Substituting (3.8) on (3.7), we get

$$\begin{aligned} h(\ell) &= k + k!\frac{\alpha_{k-1}}{(k-2)!} + k\sum_{i=k-\ell}^{k-2}\alpha_{i+1} \\ &= k(1+(k-1)\alpha_{k-1}) + k\sum_{i=k-\ell}^{k-2}\alpha_{i+1} \\ &= k\alpha_k + k\sum_{i=k-\ell}^{k-2}\alpha_{i+1} \\ &= k\sum_{i=k-\ell+1}^{k}\alpha_i \end{aligned}$$

as stated by the theorem. □

We conclude this section by making some observations on the EET of the Harmonic Chain that will be useful in the analysis of Chapter 4.

Observation 3.1. *The EET $h(\ell)$ of the Harmonic Chain is strictly increasing on ℓ.*

Proof. For any $\ell \in \{1, 2, \ldots, k\}$, we have

$$h(\ell) - h(\ell-1) = k\sum_{i=k-\ell+1}^{k}\alpha_i - k\sum_{i=k-\ell+2}^{k}\alpha_i = k\alpha_{k-\ell+1} > 0$$

since $\alpha_i > 0$ for all $i \in \mathbb{N}$. □

Observation 3.2. *For any $\ell \in \{1, 2, \ldots, k\}$ the EET of the Harmonic Chain is order of $k!$, that is $h(\ell) = \Theta(k!)$.*

Proof. Clearly, the harmonic recursion $\alpha_i = 1 + (i-1)\alpha_{i-1}$ is (strictly) increasing on i. Thus, we get:

$$\begin{aligned} h(\ell) &= k\sum_{i=k-\ell+1}^{k}\alpha_i \\ &= k\alpha_k + k\sum_{i=k-\ell+1}^{k-1}\alpha_i \\ &\le k\alpha_k + k(\ell-1)\alpha_{k-1} \\ &\le k\alpha_k + k(k-1)\alpha_{k-1} \\ &= k\alpha_k + k(\alpha_k - 1) \\ &\le 2k\alpha_k \end{aligned}$$

By the closed form expression of the harmonic recursion (3.8) we have

$$\alpha_k = (k-1)!\sum_{i=0}^{k-1}\frac{1}{i!} \le e(k-1)!$$

and thus, we get that for any $\ell \in \{1,\dots,k\}$, $h(\ell) \leq 2ek!$. Finally, from observation 3.1 we have that $h(\ell) \geq h(1) = k\alpha_k = k!\sum_{i=0}^{k-1}\frac{1}{i!} \geq k!$. On conclusion, we get that $h(\ell) = \Theta(k!)$ for all $\ell \in \{1,\dots,k\}$. □

Observation 3.3. *For any $\ell \in \{1,2,\dots,k\}$, $\frac{h(\ell)}{\ell} \leq h(1)$ in the Harmonic Chain, with the equality holding only for $\ell = 1$.*

Proof. Fix any $\ell \in \{2,3,\dots,k\}$. Then:

$$\frac{h(\ell)}{\ell} = \frac{k\alpha_k + k\sum_{i=k-\ell}^{k-2}\alpha_{i+1}}{\ell} < \frac{k\alpha_k + k(\ell-1)\alpha_{k-1}}{\ell} < \frac{k\alpha_k + (\ell-1)k\alpha_k}{\ell} = k\alpha_k = h(1).$$

where both inequalities hold from the fact that α_i is *strictly* increasing. □

3.4 Random Walks on the Binary Chain

In this section, we define and compute the EET of another special type of Birth-Death Chain we refer to as the *Binary Chain* and we will use in the analysis of the Harmonic Algorithm for the generalized k-server problem for metric spaces with $n = 2$ points.

Definition 3.11. *The **Binary Chain** is a class of Birth-Death Chains with state space $\mathcal{X} = \{0,1,\dots,k\}$ for some $k \in \mathbb{N}$, absorbing state $X = 0$, forward probabilities $q_\ell = \frac{\ell}{k}$ and backward probabilities $p_\ell = \frac{k-\ell}{k}$. A graphical representation of a Binary Chain is given in Figure 3.3.*

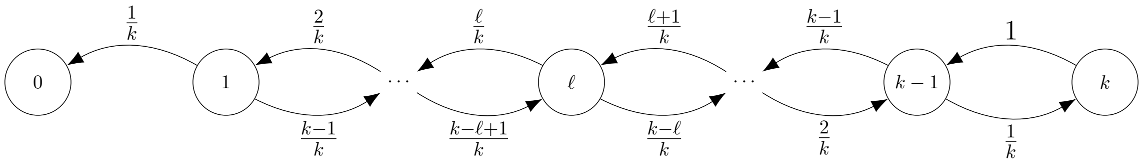


Figure 3.3: The Binary Chain

Theorem 3.5. *The Expected Extinction Time of a Binary Chain with state space $\mathcal{X} = \{0,1,\dots,k\}$ for some $k \in \mathbb{N}$ on any initial state $\ell \in \mathcal{X} \setminus \{0\}$ is given by*

$$h(\ell) = 2^k - 1 + \sum_{i=1}^{\ell-1}\frac{1}{\binom{k-1}{i}}(2^k - \sum_{j=0}^{i}\binom{k}{j})$$

with $h(0) = 0$.

Proof. Using $q_i = \frac{i}{k}$ and $p_i = \frac{k-i}{k}$, we compute:

$$\frac{p_1p_2\cdots p_i}{q_1q_2\dots q_i} = \frac{(k-1)\cdot(k-2)\dots(k-i)\cdot k^{-i}}{1\cdot 2\cdots i\cdots k^{-i}} = \frac{\frac{(k-1)!}{(k-i-1)!}}{i!} = \binom{k-1}{i} \tag{3.9}$$

Using equation (3.9) on Theorem 3.3, we get that for any $\ell \in \{1, 2, \dots, k\}$:

$$\begin{aligned}h(\ell) &= k + \sum_{i=2}^{k} \frac{\binom{k-1}{i}}{\frac{k-i}{k}} + \sum_{i=1}^{\ell-1} \frac{1}{\binom{k-1}{i}} \sum_{j=i+1}^{k} \frac{\binom{k-1}{j}}{\frac{k-j}{k}} \\ &= \sum_{i=1}^{k} \binom{k}{i} + \sum_{i=1}^{\ell-1} \frac{1}{\binom{k-1}{i}} \sum_{j=i+1}^{k} \binom{k}{j}\end{aligned}$$

Using the identity $\sum_{i=0}^{n} \binom{n}{i} = 2^n$ of the binomial coefficients, we finally get

$$h(\ell) = 2^k - 1 + \sum_{i=1}^{\ell-1} \frac{1}{\binom{k-1}{i}} (2^k - \sum_{j=0} i \binom{k}{j})$$

which concludes the proof of the theorem. $\square$

As we did for the Harmonic Chain, we conclude by making some observations on the EET of the Binary Chain that will be useful in the analysis of Chapter 4.

Observation 3.4. *The EET $h(\ell)$ of the Binary Chain is strictly increasing on ℓ.*

Proof. Clearly, $h(1) = 2^k - 1 > 0 = h(0)$. For any $\ell \in \{2, \dots, k\}$, we have

$$h(\ell) - h(\ell - 1) = \frac{1}{\binom{k-1}{\ell-1}} (2^k - \sum_{j=0}^{\ell-1} \binom{k}{j}) > \frac{1}{\binom{k-1}{\ell-1}} (2^k - 2^{k-1}) > 0$$

$\square$

Observation 3.5. *For any $\ell \in \{1, 2, \dots, k\}$ the EET of the Binary Chain is order of 2^k, that is $h(\ell) = \Theta(2^k)$.*

Proof. From Observation 3.4, it suffices to show that $h(1) = \Omega(2^k)$ and $h(k) = O(2^k)$. Since $h(1) = 2^k - 1$, the first part clearly holds. Now, we focus on bounding $h(k)$. We have:

$$h(k) = 2^k - 1 + \sum_{i=1}^{k-1} \frac{1}{\binom{k-1}{i}} (2^k - \sum_{j=0}^{i} \binom{k}{j}) < 2^k + 2^k \sum_{i=1}^{k-1} \frac{1}{\binom{k-1}{i}} \tag{3.10}$$

For the binomial coefficients, we know that $\binom{\ell}{i} \geq \binom{\ell}{2}$ for any $i \in [2, \ell - 2]$. Thus

$$\sum_{i=1}^{k-1} \frac{1}{\binom{k-1}{i}} = 1 + \frac{2}{k-1} + \sum_{i=2}^{k-3} \frac{1}{\binom{k-1}{i}} \leq 1 + \frac{2}{k-1} + \frac{k-4}{\binom{k-1}{2}} = 1 + \frac{2}{k-1} + \frac{2k-8}{(k-1)(k-2)} \tag{3.11}$$

Combining (3.10) and (3.11), we get

$$h(k) < 2 \cdot 2^k + \frac{2 \cdot 2^k}{k-1} + \frac{2 \cdot 2^k (k-4)}{(k-1)(k-2)} \leq 6 \cdot 2^k$$

for sufficiently large k, and thus $h(k) = O(2^k)$, concluding the proof. $\square$

Chapter 4

Memoryless Algorithms for the Generalized k-Server Problem

In this section we present the novel results of this thesis about memoryless algorithms for the generalized k-server problem on uniform metrics. As we have already mentioned, a memoryless online algorithm responds to a request based only on the request and its current configuration. In the context of the generalized k-server problem, an algorithm in configuration $q \in [n]^k$ receives a request r and moves to a new configuration q' in order to serve it. It makes sense for memoryless algorithms to be *lazy*, that is to move only when the new request is not served and to do so by moving only one out of the k servers.

Thus, upon receiving a request that is not served by its current configuration, a memoryless algorithm for the generalized k-server problem picks a metric space and move its server there to match the request. Furthermore, due to the symmetry of the uniform case (all metric spaces are identical) the labels of the requested points shouldn't matter.

These observations can be summarized in the following definition:

Definition 4.1. *A memoryless algorithm for the generalized k-server problem on uniform metrics is fully characterized by a distribution $p = [p_1, \ldots, p_k]$ such that $\sum_{i=1}^{k} p_i = 1$ and $p_i \geq 0$ for all $i \in [k]$. Whenever the algorithm is at a configuration $q \in [n]^k$ and receives a request r, one of the following happens:*

1. *If q serves r (i.e., $\exists i : r_i = q_i$) then the algorithm doesn't move.*

2. *If q doesn't serve r, then the algorithm randomly picks a metric M_i based on its distribution p and moves its server in M_i from point q_i to point r_i in order to serve the request.*

Since all the metric spaces are equivalent, the only "reasonable" memoryless algorithm is the Harmonic Algorithm, i.e., the algorithm that has uniform distribution $p_i = \frac{1}{k}$ for all $i \in [k]$. Furthermore, it is trivial to show that if $p_i = 0$ for any i then the algorithm is not competitive, thus we assume that $p_i \in (0, 1)$ for all $i \in [k]$.

Our contribution is determining the competitive ratio of memoryless algorithms for the generalized k-server problem on uniform metric spaces based on their distribution p, in different adversarial settings.

We can summarize our results with the following theorems that were stated in the introduction:

Theorem 4.1. *The Harmonic Algorithm for the generalized k-server problem on uniform metrics is $(k \cdot \alpha_k)$-competitive against adaptive online adversaries, where α_k is the solution of the recursion $\alpha_k = 1 + (k-1)\alpha_{k-1}$, with $\alpha_1 = 1$.*

Theorem 4.2. *The competitive ratio of any randomized memoryless algorithm for the generalized k-server problem on uniform metrics with $n \geq 3$ points is at least $k \cdot \alpha_k$ against adaptive online adversaries.*

Theorem 4.3. *The competitive ratio of the Harmonic algorithm for the generalized k-server problem on uniform metrics with $n = 2$ points is $\Theta(2^k)$ against adaptive online adversaries.*

Theorem 4.4. *The competitive ratio of the Harmonic algorithm for the generalized k-server problem on uniform metrics with $n \geq 2k+1$ is at least $k \cdot \alpha_k$ against oblivious adversaries.*

Theorem 4.5. *The competitive ratio of the Harmonic algorithm for the generalized k-server problem on uniform metrics with $n = 2$ is $\Omega(2^k)$ against oblivious adversaries.*

For the rest of this chapter, we will be proving all of these theorems.

4.1 Adaptive Upper Bound for the Harmonic Algorithm

In this section we prove Theorem 4.1. More precisely, we use a potential function argument to show that for any request sequence, the expected cost of the Harmonic Algorithm is at most $k \cdot \alpha_k$ times the cost of the adversary.

Organization. In Section 4.1.1, we define a potential between the Harmonic Algorithm's and the adversary's configurations that is inspired by random walks on a special type of Markov Chains [47] we refer to as the "*Harmonic Chain*". The required background of Markov Chains has already been presented in Chapter 3. Then, in Section 4.1.2 we will use this potential to prove the upper bound of Theorem 4.1 with a standard potential-based analysis.

4.1.1 Definition of the Potential Function

We begin by presenting the intuition behind the definition of our potential function. Our first observation is that since (i) the metrics are uniform with equal weights and (ii) the Harmonic Algorithm does not distinguish between metrics since it has equal probabilities

$\frac{1}{k}$, it makes sense for the potential between two configurations $p, q \in [n]^k$ to depend only on their Hamming distance and not on the labels of their points. In order to come up with an appropriate potential, we need to understand how the Hamming distance between the Harmonic Algorithm's and the adversary's configurations evolves over time.

Imagine that the adversary moves to an "optimal" configuration of his choice and then it serves requests until the Harmonic Algorithm reaches this configuration as well. Since the adversary must serve all the requests using a server from its configuration, we know that for each request $r = (r_1, \dots, r_k)$, at least one of the requested points r_i should coincide with the i-th server of the adversary. In that case, with probability $\frac{1}{k}$ the Harmonic Algorithm moves in metric M_i, thus it decreases his Hamming distance from the adversary by 1. On the other hand, assume that ℓ servers of the algorithm coincide with the ones of the adversary. Then, with probability $\frac{\ell}{k}$ it would increase its Hamming distance from the optimal configuration by 1. This shows that the evolution of the Hamming distance between the Harmonic Algorithm's and the adversary's configurations is captured by a random walk on the following Markov Chain that we refer to as the *Harmonic Chain*.

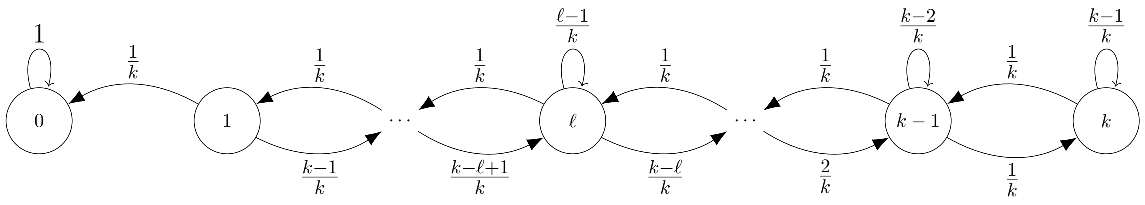


Figure 4.1: The Harmonic Chain - Here, the states of the chain denote the Hamming distance between the configurations of the Harmonic Algorithm and the adversary.

Recall that we have already formally defined and studied this class of Markov Chains in Section 3.3 of Chapter 3. In the scenario we described above, the expected movement cost of the Harmonic Algorithm until it reaches the adversary's configuration with an initial Hamming distance of ℓ would be $\mathbb{E}[N|X_0 = \ell]$ where N denotes a random variable defined as $N = min_{\tau \geq 0}\{X_\tau = 0\}$ and X_t denotes the state of the Harmonic Chain at time t. As we discussed in Section 3.2 of Chapter 3, this quantity is known as the *Expected Extinction Time* (ETT) [47] of a Markov Birth-Death Chain and we use $h(\ell)$ to denote it. Intuitively, $h(k)$ should immediately give an upper bound on the competitive ratio of the Harmonic Algorithm.

The Harmonic Chain has already been studied in Chapter 3, where we proved Theorem 3.4. For the convenience of the reader, we re-state the theorem without the proof here:

Theorem 4.6. *For any initial state $\ell \in \{0, 1, \dots, k\}$, the EET of the Harmonic Chain is given by*

$$h(\ell) = k \sum_{i=k-\ell+1}^{k} \alpha_i$$

where α_i is the solution to the recursion $\alpha_i = 1 + (i-1)\alpha_{i-1}$.

We have already shown that $h(\ell)$ is strictly increasing (Observation 3.1) and that $h(\ell) = \Theta(\ell!)\ \forall \ell \in \{1, \dots, k\}$ (Observation 3.2).

Suppose that the adversary moves ℓ servers whenever the algorithm reaches its configuration and then it doesn't move until the algorithm reaches its new configuration. Intuitively, the competitive ratio would be $\frac{h(\ell)}{\ell}$ which is maximized for $\ell = 1$ by Observation 3.3. This means that $h(1) = k \cdot \alpha_k$ is an upper bound for the competitive ratio of the Harmonic Algorithm. While this intuition is very important, it is not enough to formally prove Theorem 4.1. However, motivated by it, we will define the potential between two configurations of Hamming distance ℓ as $h(\ell)$. Formally,

Definition 4.2 (Potential Function)**.** *The potential between two configurations $p, q \in [n]^k$ is defined as*

$$\phi(p, q) = h(d_H(p, q)).$$

4.1.2 Bounding the Competitive Ratio

In this section, we will prove the upper bound of Theorem 4.1 by using the potential we defined in Section 4.1.1. Fix any request sequence $\bar{r} = [r^1, \dots, r^T]$ for any $T \in \mathbb{N}$ such that $r^t \in [n]^k\ \forall t \in [T]$. Let $q^t \in [n]^k$ be the configuration of the Harmonic Algorithm and $\mathcal{A}^t \in [n]^k$ the configuration of the adversary after serving request r^t. Also, let $q^0 = \mathcal{A}^0$ be the initial configuration of the instance. We will prove that when the adversary moves x servers the increase in potential is at most $k \cdot \alpha_k \cdot x$ and when the Harmonic Algorithm moves one server, the expected decrease in potential is at least 1. Then, using these properties, we will prove Theorem 4.1.

To simplify the analysis, we make the following observation for the potential function.

Observation 4.1. *For any $\ell, \ell' \in \{0, 1, \dots, k\}$ such that $\ell < \ell'$ it holds that*

$$h(\ell') - h(\ell) = k \sum_{i=\ell}^{\ell'-1} \alpha_{k-i}$$

Proof. By telescoping we have

$$h(\ell') - h(\ell) = \sum_{i=\ell}^{\ell'-1} (h(i+1) - h(i)) = \sum_{i=\ell}^{\ell'-1} (k \sum_{j=k-i}^{k} \alpha_j - k \sum_{j=k-i+1}^{k} \alpha_j) = k \sum_{i=\ell}^{\ell'-1} \alpha_{k-i}$$

where the second equality holds by the definition of the potential. □

Using this observation, we are now ready to prove the following lemmata:

Lemma 4.1 (Adversary Moves)**.** *For any $t \in \{1, \dots, T\}$ it holds that*

$$\phi(q^{t-1}, \mathcal{A}^t) - \phi(q^{t-1}, \mathcal{A}^{t-1}) \leq k \cdot \alpha_k \cdot d_H(\mathcal{A}^t, \mathcal{A}^{t-1}).$$

Proof. Let $\ell^{t-1} = d_H(q^{t-1}, \mathcal{A}^{t-1})$ and $\ell^t = d_H(q^{t-1}, \mathcal{A}^t)$. Clearly, $\ell^{t-1}, \ell^t \in \{0, 1, \dots, k\}$. Since the potential $h(\ell)$ is strictly increasing on ℓ, if $\ell^t \leq \ell^{t-1}$ then this means that the adversary's move didn't increase the potential and then the Lemma follows trivially. Thus, we only need to prove the Lemma for $0 \leq \ell^{t-1} < \ell^t \leq k$. We have:

$$h(\ell^t) - h(\ell^{t-1}) = k \sum_{i=\ell^{t-1}}^{\ell^t - 1} \alpha_i \leq (\ell^t - \ell^{t-1}) k \alpha_k \tag{4.1}$$

where the equality is given from Observation 4.1 and the inequality from the fact that the recursion α_ℓ is increasing. Thus, we have proven that $\phi(q^{t-1}, \mathcal{A}^t) - \phi(q^{t-1}, \mathcal{A}^{t-1}) \leq (\ell^t - \ell^{t-1}) \cdot k \cdot \alpha_k$. To conclude the proof of the Lemma, by the triangle inequality of the Hamming distance we have

$$d_H(q^{t-1}, \mathcal{A}^{t-1}) + d_H(\mathcal{A}^{t-1}, \mathcal{A}^t) \geq d_H(q^{t-1}, \mathcal{A}^t)$$

which gives $\ell^t - \ell^{t-1} \leq d_H(\mathcal{A}^{t-1}, \mathcal{A}^t)$. Combined with (4.1), we get the Lemma. □

Lemma 4.2 (Harmonic Moves). *For any $t \in \{1, \dots, T\}$ it holds that*

$$\mathbb{E}[\phi(q^{t-1}, \mathcal{A}^t) - \phi(q^t, \mathcal{A}^t)] \geq d_H(q^{t-1}, q^t).$$

Proof. If the Harmonic Algorithm serves the request, then $q^t = q^{t-1}$ and the Lemma follows trivially. Otherwise, by definition, it moves to a configuration q^t such that $d_H(q^{t-1}, q^t) = 1$. Let $\ell^{t-1} = d_H(q^{t-1}, \mathcal{A}^t)$ and $\ell^t = d_H(q^t, \mathcal{A}^t)$. Also, let $C = |\{i : \mathcal{A}_i^t = r_i^t\}|$, i.e., the number of the adversary's servers that could serve the current request. By definition, $\mathcal{A}^t$ must serve r^t which gives $C \geq 1$. Furthermore, q^{t-1} doesn't serve the request but $\mathcal{A}^t$ does, and thus $\ell^{t-1} \geq 1$.

Recall that the Harmonic Algorithm randomly moves at a metric with equal probabilities in order to serve a request. If it moves in any of the C metrics where the adversary serves the request, we get $\ell^t = \ell^{t-1} - 1$ and the potential decreases with probability $\frac{C}{k}$. If it moves on any of the $k - \ell^{t-1}$ metrics where $a_i^t = q_i^{t-1}$, we get $\ell^t = \ell^{t-1} + 1$ and the potential increases with probability $\frac{k-\ell^{t-1}}{k}$. In any other case, we have $\ell^t = \ell^{t-1}$ and the potential doesn't change. To simplify the notation, we define $j \in \{1, \dots, k\}$ as $j = k - \ell^{t-1} + 1$. We have:

$$\begin{aligned}
\mathbb{E}[\phi(q^{t-1}, \mathcal{A}^t) - \phi(q^t, \mathcal{A}^t)] &= \mathbb{E}[h(\ell^{t-1}) - h(\ell^t)] \\
&= \frac{C}{k}(h(\ell^{t-1}) - h(\ell^{t-1} - 1)) + \frac{k - \ell^{t-1}}{k}(h(\ell^{t-1}) - h(\ell^{t-1} + 1)) \\
&= C\alpha_j - (j-1)\alpha_{j-1} \\
&= C\alpha_j - (\alpha_j - 1) = (C-1)\alpha_j + 1 \\
&\geq 1 = d_h(q^{t-1}, q^t)
\end{aligned}$$

where the first equality follows from the definition of the potential, the second equality from the possible changes in the Hamming distance between the algorithm and the adversary, the third equality follows from Observation 4.1 and the definition of j, the fourth equality follows from the definition of the recursion α_ℓ and the inequality follows from $C \geq 1$. □

Proof of Theorem 4.1 We are now ready to prove Theorem 4.1. By combining lemmata 4.1 and 4.2, we get that for any $t \in \{1, \dots, T\}$, the expected difference in potential is

$$\mathbb{E}[\Delta\phi^t] = \mathbb{E}[\phi(q^t, \mathcal{A}^t) - \phi(q^{t-1}, \mathcal{A}^{t-1})] \leq k\alpha_k d_H(\mathcal{A}^t, \mathcal{A}^{t-1}) - d_H(q^{t-1}, q^t)$$

Now, let $ADV = \sum_{t=1}^{T} d_H(\mathcal{A}^t, \mathcal{A}^{t-1})$ be used to denote the total cost of the adversary and $ALG = \sum_{t=1}^{T} d_H(q^t, q^{t-1})$ be used to denote the expected cost of the Harmonic Algorithm. Summing over all $t \in \{1, 2, \dots, T\}$ we finally get

$$\sum_{t=1}^{T} \Delta\phi^t = \phi(q^T, \mathcal{A}^T) - \phi(q^0, \mathcal{A}^0) \leq k\alpha_k \cdot ADV - ALG$$

and since $\mathcal{A}^0 = q^0$ (i.e., $\phi(q^0, \mathcal{A}^0) = 0$) and $\phi(q^T, \mathcal{A}^T) \geq 0$, we get that $ALG \leq k \cdot \alpha_k \cdot ADV$, which concludes the proof of Theorem 4.1.

4.2 General Adaptive Lower Bound

In this section we prove Theorem 4.2. More precisely, we construct an adversarial request sequence against any memoryless algorithm and prove that its competitive ratio is lower bounded by the solution of a linear system of 2^k equations. Since solving this system directly is possible only for easy cases like $k = 2$ or $k = 3$, we show how to get a lower bound for the solution (similarly to the approach taken by Chiplunkar and Vishwanathan [28] for weighted uniform metric spaces) and thus the competitive ratio of any memoryless algorithm.

Organization. In Section 4.2.1 we formally define the adversarial request sequence and the intuition behind it. In Section 4.2.2 we state the linear system of equations that our request sequence results to and prove a lower bound on its solution. This leads to the proof of Theorem 4.2.

4.2.1 Constructing the adversarial instance

Before we state the adversarial instance, it is useful to give the intuition behind it. It is natural to construct an adversary that moves only when it has the same configuration with the algorithm.

In fact, we construct an adversary that moves in only one metric space: the one that the algorithm uses with the smallest probability (ties are broken arbitrarily). Recall that in the analysis of the harmonic algorithm from Section 4.1, the competitive ratio is also maximized when in each "phase" the adversary starts with only one different server than the algorithm and does not move until the configurations (of algorithm and adversary) match (Observation 3.3).

Let ALG be any online algorithm and ADV be the adversary. Consider a “phase” to be a part of the request sequence where in the beginning the configurations of ALG and ADV coincide and it ends when ALG matches the configuration of ADV. Since ADV must serve all requests, in each request r one point r_i is such that $a_i = r_i$; we say that the i-th position of ADV is revealed in such a request. Thus every request will reveal to the algorithm exactly one of the positions of the adversary's servers in some metric space M_i. The main idea behind our lower bound instance is that, in each request, out of the metric spaces that servers of ALG and ADV differ, we reveal to the algorithm the position of ADV in the metric that ALG serves with the highest probability; this implies that whenever ALG and ADV differ by only one server, this will be in metric M_k. Intuitively, this way we exploit best the “assymetries” in the distribution of ALG (this is formalized in Lemma 4.3).

The instance. Recall that any memoryless algorithm for the generalized k-server problem on uniform metric spaces is fully characterized by a probability distribution $p = [p_1, p_2, \dots, p_k]$ over the k-metric spaces $M_1, M_2, \dots, M_k$. W.l.o.g., we can assume that $p_1 \geq p_2 \geq \dots \geq p_k$. Let $q^t, \mathcal{A}^t$ be used to denote the configurations of the algorithm and the adversary after serving request r^t respectively. Also, let $q^0 = \mathcal{A}^0$ be used to denote the initial configuration of both the algorithm and the adversary. We will now construct the request sequence. For $t = 1, 2, \dots, T$:

1. Observe q^{t-1}, i.e., the algorithm's current configuration.
2. If $q^{t-1} = \mathcal{A}^{t-1}$, then:

   $\mathcal{A}^t = [q_1^0, q_2^0, \dots, q_{k-1}^0, Z]$ for any $Z \in [n]$ such that $Z \neq \mathcal{A}_k^{t-1}$ and $Z \neq q_k^{t-1}$.

   otherwise:

   $\mathcal{A}^t = \mathcal{A}^{t-1}$.
3. Determine $m = \min(\{j : q_j^{t-1} \neq \mathcal{A}_j^t\})$.
4. Pick any $r^t \in [n]^k$ such that $r_m^t = \mathcal{A}_m^t$ and $r_j^t \neq \mathcal{A}_j^t$, $r_j^t \neq q_j^{t-1}$ $\forall j \in [k] \setminus \{m\}$.

Note that for steps 2 and 4, we need to have at least $n \geq 3$ points in order to pick a point that isn't occupied by neither the algorithm's nor the adversary's servers. As we explain in Section 4.3, this is a necessary requirement; if all metrics have $n = 2$ points, then the competitive ratio of the Harmonic Algorithm is $O(2^k)$ and therefore a lower bound of order $k!$ is not possible.

As an example of our instance, for $k = 4$, let $\mathcal{A}^{t-1} = [0, 0, 0, 0]$ and $q^{t-1} = [1, 0, 0, 1]$ for some t. Clearly, the algorithm and the adversary have different servers in metric M_1 and M_4. From step 3, $m = \min(1, 4) = 1$, i.e., M_1 is the metric space that the algorithm serves with highest probability out of the metric spaces that it and the adversary have their servers in different points. Then, from step 4, $r^t = [0, 2, 2, 2]$ (actually, the selection of the last three coordinates is arbitrary as long as neither the algorithm nor the adversary have their server on this point).

Notice that ADV moves one server in metric space M_k whenever it has the same configuration with ALG. On the other hand, ALG never serves request r^t with configuration q^{t-1} and thus moves at every time step. This means that the competitive ratio of ALG is lower bounded by the (expected) number of requests it takes for it to reach configuration of ADV.

4.2.2 Proving the Lower Bound

Our approach. We define the *state* of the algorithm at time t as $S^t = \{i : q_i^t \neq \mathcal{A}_i^t\}$, i.e., the subset of metric spaces with different servers between the algorithm and the adversary. In this context, $h(S)$ is used to denote the expected number of requests it takes for the algorithm to reach the adversary's configuration, i.e. state $\emptyset$, starting from some state $S \subseteq [k]$. From the request sequence we defined, $h(\{k\})$ is a lower bound for the competitive ratio of any memoryless algorithm.

By observing how the state S of the algorithm (and by extension $h(S)$) evolves under the request sequence, we can write down a linear system of 2^k equations on the 2^k variables $h(S)\ \forall S \subseteq [k]$. In fact, these equations give the EET of a random walk in a Markov Chain of 2^k states. We then prove a lower bound on $h(\{k\})$ and thus the competitive ratio of any memoryless algorithm. Notice that for the given instance, if we were analyzing the Harmonic Algorithm, then the Hamming distance between it and the adversary would be captured by the Harmonic Chain and we would immediately get that $h(\{k\}) = k \cdot \alpha_k$.

Analysis. Fix any two different configurations $q, \mathcal{A}$ for the algorithm and the adversary that are represented by state $S = \{i : q_i \neq \mathcal{A}_i\} \neq \emptyset$ with $\min(S) = m$. Then, we know that for the next request r we have constructed it holds that $r_m = \mathcal{A}_m \neq q_m$ and $r_j \neq \mathcal{A}_j \neq q_j \neq r_j$ for any $j \in [k] \setminus \{m\}$. Recall that the memoryless algorithm will randomly move to some state M_j and move to a different configuration $q' = [q_1, \dots, q_{j-1}, r_j, q_{j+1}, \dots, q_k]$ that is captured by state S'. We distinguish between the following three cases:

1. If $j \notin S$, then this means that $q_j = \mathcal{A}_j$ and $q'_j = r_j \neq \mathcal{A}_j$ and thus $S' = S \cup \{j\}$.
2. If $j = m$, then $q_j \neq \mathcal{A}_j$ and $q'_j = \mathcal{A}_j = r_m$ and thus $S' = S \setminus \{m\}$.
3. If $j \in S \setminus \{m\}$ then $q_j \neq \mathcal{A}_j$ and $q'_j \neq \mathcal{A}_j$ and thus $S' = S$.

Since $h(S)$ denotes the expected number of steps until the state of the algorithm becomes $\emptyset$ starting from S, from the above cases we have that for any state $S \neq \emptyset$:

$$h(S) = 1 + p_m \cdot h(S \setminus \{m\}) + \sum_{j \notin S} p_j \cdot h(S \cup \{j\}) + \sum_{j \in S \setminus \{m\}} p_j \cdot h(S), \quad m = \min(S).$$

Combined we the fact that obviously $h(\emptyset) = 0$ and $\sum_{j=1}^{k} p_j = 1$, we get the following set of 2^k linear equations with 2^k variables:

$$\left\{ \begin{array}{c} h(\emptyset) = 0 \\ p_m(h(S) - h(S \setminus \{m\})) = 1 + \sum_{j \notin S} p_j(h(S \cup \{j\}) - h(S)), \quad \forall S \neq \emptyset, m = \min(S) \end{array} \right\} \tag{4.2}$$

Normally, we would like to solve this linear system to compute $h(\{k\})$ and this would be the proven lower bound for the memoryless algorithm. However, even for $k = 4$ it is hopeless to find closed form expressions for the solutions of this system. Interestingly, similar equations were studied by Chiplunkar and Vishnawathan [28] for the weighted uniform metric case. In their study, they showed a monotonicity property on the solutions of their linear system that directly transfers to our setting and is stated in Lemma 4.3 below. Using this, combined with the special structure of our problem, we show how to derive a lower bound of $k \cdot \alpha_k$ for $h(\{k\})$ instead of solving (4.2) to directly compute it.

Lemma 4.3. *For any $S \subseteq [k]$ with $i, j \in S$ such that $i < j$ (and thus $p_i \geq p_j$), the solutions of linear system* (4.2) *satisfy*

$$h(S) - h(S \setminus \{j\}) \geq \frac{p_i}{p_j}(h(S) - h(S \setminus \{i\}))$$

Proof. We begin by showing that (4.2) is equivalent to the equations studied in [28]. By introducing a new set of variables ϕ, defined as

$$\phi(S) = h([k]) - h([k] \setminus S), \ \ \forall S \subseteq [k] \tag{4.3}$$

we get that $\phi(\emptyset) = 0$ and $\forall S \neq \emptyset$ with $m = \min(S)$ we get

$$p_m(\phi(([k] \setminus S) \cup \{m\}) - \phi([k] \setminus S)) = 1 + \sum_{j \notin S} p_j(\phi([k] \setminus S) - \phi([k] \setminus S \setminus \{j\}))$$

Lastly, by re-writing the equations using $\bar{S} = [k] \setminus S$, we end up with the following (equivalent) linear system:

$$\left\{ \begin{array}{c} \phi(\emptyset) = 0 \\ p_m(\phi(\bar{S} \cup \{m\}) - \phi(\bar{S}) = 1 + \sum_{j \in \bar{S}} p_j(\phi(\bar{S}) - \phi(\bar{S} \setminus \{j\})), \quad \forall \bar{S} \neq [k], \ m = \min([k] \setminus \bar{S}) \end{array} \right\} \tag{4.4}$$

We remark that this is the exact set of equations studied by Chiplunkar and Vishnawathan (see equations (6),(7) in [28]). Using the Gauss-Seidel Trick technique, they prove that the solutions of (4.4) (and thus the solutions of (4.2)) always exist. Then, in Lemma 3.3 of their paper they proved the following property for the solutions of linear system (4.4):

Lemma 4.4 (Lemma 3.3 of Chiplunkar et. al.). *For any $\bar{S} \subseteq [k]$ with $i, j \notin \bar{S}$ and $i < j$ (thus $p_i \geq p_j$), the solution of linear system* (4.4) *satisfies*

$$p_i(\phi(\bar{S} \cup \{i\}) - \phi(\bar{S})) \leq p_j(\phi(\bar{S} \cup \{j\}) - \phi(\bar{S}))$$

By re-writing this lemma for $S = [k] \setminus \bar{S}$, we get that for any $S \subseteq [k]$ with $i, j \in S$ and $i < j$, the solution of linear system (4.4) satisfies

$$p_i(\phi([k] \setminus (S \setminus \{i\})) - \phi([k] \setminus S)) \leq p_j(\phi([k] \setminus (S \setminus \{j\})) - \phi([k] \setminus S))$$

Lastly, by equation (4.3) we get that for any $S \subseteq [k]$ with $i, j \in S$ and $i < j$, the solution of linear system (4.2) satisfies

$$p_i(h(S) - h(S \setminus \{i\})) \leq p_j(h(S) - h(S \setminus \{i\}))$$

that is, we have that Lemma 4.3 holds. $\square$

Let us first see the intuition behind the inequality of Lemma 4.3. Let S be the subset of metric spaces where the servers of *ALG* and *ADV* occupy different points: then, in the next move, the expected time to match *ADV* decreases the most, if *ALG* matches first the j-th server of the adversary (i.e., the "state" changes from S to $S \setminus \{j\}$) where j is the metric with the smallest the probability p_j. This explains why in our adversarial instance we choose to reveal to *ALG* the location of *ADV* in the metric it serves with the highest probability: this makes sure that the decrease in the expected time to reach *ADV* is minimized.

Using Lemma 4.3, we can now prove the following:

Lemma 4.5. *For any $S \subseteq [k]$ with $S \neq \emptyset$ and $i \in S$, the solutions of linear system* (4.2) *satisfy*

$$p_i(h(S) - h(S \setminus \{i\})) \geq 1 + \sum_{j \notin S} p_j(h(S \cup \{j\}) - h(S))$$

Proof. Fix any non-empty set $S \subseteq [k]$ and any $i \in S$. Let $m = \min(S) \leq i$. Then, by Lemma 4.3 we have

$$p_i(h(S) - h(S \setminus \{i\})) \geq p_m(h(S) - h(S \setminus \{m\}))$$

Since $m = \min(S)$, and we study the solution of linear system (4.2), we have

$$p_m(h(S) - h(S \setminus \{m\})) = 1 + \sum_{j \notin S} p_j(h(S \cup \{j\}) - h(S))$$

and the lemma follows. $\square$

We are now ready to prove the main theorem of this section.

Theorem 4.7. *The solution of linear system* (4.2) *satisfies*

$$h(\{k\}) \geq \frac{\alpha_k}{p_k}$$

Proof. In order to prove the theorem, it suffices to show that for any $S \subseteq [k]$ such that $S \neq \emptyset$ and $i \in S$, it holds that

$$p_i(h(S) - h(S \setminus \{i\})) \geq \alpha_{k-|S|+1}$$

Then, by setting $S = \{k\}$ ($|S| = 1$) and $i = k \in S$, we get $p_k(h(\{k\}) - h(\emptyset)) \geq \alpha_k$, and since $h(\emptyset) = 0$ by definition, the Theorem follows. It remains to prove the desired property. This can be shown by induction on the size of S.

Base case: If $|S| = k$ (this means that $S = [k]$) then for any $i \in S$, by (4.2) we have

$$p_i(h(S) - h(S \setminus \{i\})) = 1 = \alpha_1 = \alpha_{k-|S|+1}.$$

Inductive hypothesis: Suppose that for any $S \subseteq [k]$ with $|S| = \ell > 1$ and any $i \in S$, we have

$$p_i(h(S) - h(S \setminus \{i\})) \geq \alpha_{k-\ell+1}.$$

Inductive step: Let $S \subseteq [k]$ be any set with $|S| = \ell - 1 > 0$ and $i \in S$ be any element of this set. By Lemma 4.5, we have that

$$p_i(h(S) - h(S \setminus \{i\})) \geq 1 + \sum_{j \notin S} p_m(h(S \cup \{j\}) - h(S))$$

Now, for any $j \notin S$ we can use the hypothesis on the set $S \cup \{j\}$ with size ℓ. Thus, we have

$$p_j(h(S \cup \{j\}) - h(S)) \geq \alpha_{k-\ell+1} = \alpha_{k-|S|}$$

for any $j \notin S$. Combining, we get

$$p_i(h(S) - h(S \setminus \{i\})) \geq 1 + (k - |S|)\alpha_{k-|S|} = \alpha_{k-|S|+1}.$$

□

Proof of Theorem 4.2. Since $p_1 \geq p_2 \geq \cdots \geq p_k$, we have that $p_k \leq \frac{1}{k}$. Thus, by Theorem 4.7 we have that $h(\{k\}) \geq k \cdot \alpha_k$ for any distribution. Since $h(\{k\}$ is a lower bound for any memoryless algorithm, the Theorem follows.

Corollary 4.1. *The Harmonic Algorithm is the only memoryless algorithm with a competitive ratio of $k \cdot \alpha_k$.*

Proof. By Theorem 4.7, the competitive ratio of the Harmonic Algorithm is at least $k \cdot \alpha_k$ and combined with the upper bound of Theorem 4.1 we get that the Harmonic Algorithm is $(k \cdot \alpha_k)$-competitive. Assuming $p_1 \geq \cdots \geq p_k$, any other memoryless algorithm will have $p_k < \frac{1}{k}$. Thus, by Theorem 4.7 its competitive ratio will be lower bounded by $h(\{k\}) > k \cdot \alpha_k$ which is strictly worse that the competitive ratio of the Harmonic Algorithm. □

4.3 Special Case: Two Point Metric Spaces

In this Section, we prove Theorem 4.3 and improve the $\Theta(k!)$ bound of the Harmonic Algorithm in uniform metric spaces to $\Theta(2^k)$ in the special case of metric spaces with $n = 2$ points. On a high level, we tie the performance of the Harmonic Algorithm with the EET on a special type of Markov Chain and use the same analysis to prove both the upper bound and the lower bound on its competitive ratio.

Recall that in the proofs of Theorems 4.1 and 4.2, we made the assumption that at any time t that the adversary was in configuration $\mathcal{A}^t$ and the algorithm was in configuration $q^t \neq \mathcal{A}^t$, the next request r^{t+1} was not served by q^t and was served by $\mathcal{A}^t$ in only one metric space M_i. This can be true in metric spaces with $n \geq 3$ points, as we showed in the construction of the adversarial instance for Theorem 4.2. However, in metric spaces with $n = 2$ points, the only request that is not served by q^t is its anti-configuration $r^{t+1} = \bar{q}^t$. This restricts the analysis, since for $r^{t+1} = \bar{q}^t$ the adversary will serve the request in $d_H(q^t, \mathcal{A}^t)$ metric spaces instead of one; this shows that at every time-step there are more than one "good" choices for the algorithm.

In the worst case for the algorithm, it starts from the adversary's anti-configuration and is continuously requested its anti-configuration until it reaches the adversary. As we did for the analysis of the general upper bound for the Harmonic Algorithm, we capture the algorithm's distance from the adversary using the Hamming distance of their configurations. For any $\ell \in \{0, 1, \ldots, k\}$, let $h(\ell)$ denote the expected number of requests needed for the algorithm to reach the adversary, starting from a configuration that has Hamming distance ℓ from it. Clearly $h(0) = 0$. Then, $h(k)$ *is an upper bound for the competitive ratio of the Harmonic Algorithm.*

It remains to formally define and study the Markov Chain that captures the evolution of the Hamming distance between the algorithm and the adversary. Fix any configuration $\mathcal{A}$ for the adversary and q for the algorithm such that $q \neq \mathcal{A}$. Also, let $\ell = d_H(q, \mathcal{A}) \in \{1, 2, \ldots, k\}$. The algorithm is requested $r = \bar{q}$. Clearly, $\mathcal{A}$ serves r in ℓ metric spaces.

We distinguish between the following two cases:

1. If the Harmonic Algorithm moves in any of the ℓ metric spaces where $q_i \neq \mathcal{A}_i$, then its Hamming distance from the adversary decreases by 1. This happens with probability $\frac{\ell}{k}$.

2. If the Harmonic Algorithm moves in any of the $(k - \ell)$ metric spaces where $q_i = \mathcal{A}_i$, then its Hamming distance from the adversary increases by 1. This happens with probability $\frac{k-\ell}{k}$.

From this, we get that the evolution of the Hamming distance between the Harmonic Algorithm and the adversary is captured by the Markov Chain of Figure 4.2 that we refer to as the Binary Chain.

We have already formally defined and studied the Binary Chain in Section 3.4 of Chapter 3. In this part of the thesis, we computed the general expression for the EET

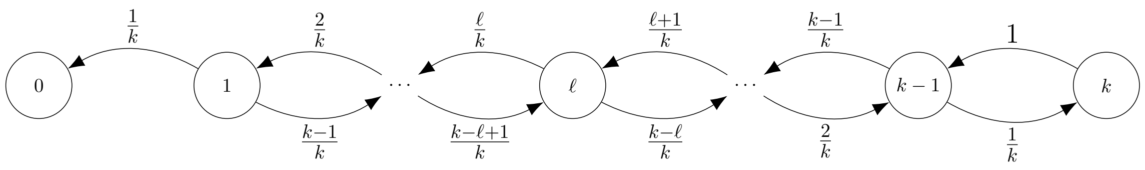


Figure 4.2: The evolution of the Harmonic Algorithm's Hamming distance from the adversary when $n = 2$.

on the Binary Chain (Theorem 3.5) and proved that for any $\ell \in \{1, 2, \dots, k\}$, the EET is order of 2^k.

We have already argued that $h(k)$ is an upper bound for the competitive ratio of the Harmonic Algorithm. Since $h(k) = \Theta(2^k)$, this gives that the competitive ratio of the Harmonic Algorithm in metric spaces with $n = 2$ points is $O(2^k)$.

Furthermore, it is easy to see that $h(1)$ is a lower bound for the competitive ratio of the Harmonic Algorithm. Consider an instance where the adversary (i) moves a server in metric space M_1 when it has the same configuration with the algorithm and (ii) always requests the algorithm's anti-configuration. Then, for every server the adversary moves, the algorithm pays (on expectation) a total movement cost of $h(1)$. Since $h(1) = 2^k - 1$, this gives that the competitive ratio of the Harmonic Algorithm in metric spaces with $n = 2$ points is $\Omega(2^k)$.

In conclusion, we get that the competitive ratio of the Harmonic Algorithm in metric spaces with $n = 2$ points is $\Theta(2^k)$ and conclude the proof of Theorem 4.3.

4.4 Oblivious Bounds

Up until now, we focused on analyzing memoryless algorithms against adaptive online adversaries. In this section, we turn our attention to oblivious adversaries. While our proven upper bounds for the Harmonic Algorithm clearly hold in the oblivious setting, there is no guarantee that the competitive ratio of the Harmonic Algorithm is $\Omega(k \cdot \alpha_k)$ in general metric spaces or $\Omega(2^k)$ in metric spaces with $n = 2$ points when the adversary is oblivious.

In fact, the true competitive ratio of the Harmonic Algorithm might be better in the oblivious setting. In this part of the thesis we will show that this isn't true by proving Theorems 4.4 and 4.5 that state that against the Harmonic Algorithm, the power of oblivious and adaptive online adversaries is the same.

4.4.1 Oblivious Lower Bound for the Harmonic Algorithm

We will prove Theorem 4.4, that is that in metric spaces with $n \geq 2k + 1$ points, the competitive ratio of the Harmonic Algorithm is at least $(k \cdot \alpha_k)$. We will show the lower bound $k \cdot \alpha_k$ by constructing a request sequence where the (expected) cost of the Harmonic Algorithm is at least $k\alpha_k$ times that of the optimal solution.

The only (mild) assumption we will need to make is that the metrics $M_1, M_2, \dots, M_k$ have at least $(2k+1)$-points each. Let $\{0, 1, \dots, 2k\}$ be the labels of each metrics points. If $n > 2k+1$, we can ignore any extra points by never requesting them. Also, w.l.o.g. by re-labeling the metrics we can assume that the initial configuration for the instance is $q^0 = [0, 0, \dots, 0]$.

We will now construct an oblivious request sequence (that is, without any assumptions on the algorithm's random choices) of $T = 4kLP$ requests for $L, P \in \mathbb{N}$ to be defined later. It will be helpful to define the following sub-sequences of requests, each with $2k$-requests.

Request Sub-Sequence A:

- $r^1 = [\mathbf{1}, 1, 1, \dots, 1, 1]$
- $r^2 = [\mathbf{1}, 2, 2, \dots, 2, 2]$
- $r^3 = [3, \mathbf{0}, 3, \dots, 3, 3]$
- $r^4 = [4, \mathbf{0}, 4, \dots, 4, 4]$
- $r^5 = [5, 5, \mathbf{0}, \dots, 5, 5]$
- $r^6 = [6, 6, \mathbf{0}, \dots, 6, 6]$

  $\vdots$

- $r^{2k-1} = [2k-1, 2k-1, 2k-1, \dots, 2k-1, \mathbf{0}]$
- $r^{2k} = [2k, 2k, 2k, \dots, 2k, \mathbf{0}]$

Request Sub-Sequence B

- $r^1 = [\mathbf{0}, 1, 1, \dots, 1, 1]$
- $r^2 = [\mathbf{0}, 2, 2, \dots, 2, 2]$
- $r^3 = [3, \mathbf{0}, 3, \dots, 3, 3]$
- $r^4 = [4, \mathbf{0}, 4, \dots, 4, 4]$
- $r^5 = [5, 5, \mathbf{0}, \dots, 5, 5]$
- $r^6 = [6, 6, \mathbf{0}, \dots, 6, 6]$

  $\vdots$

- $r^{2k-1} = [2k-1, 2k-1, 2k-1, \dots, 2k-1, \mathbf{0}]$
- $r^{2k} = [2k, 2k, 2k, \dots, 2k, \mathbf{0}]$

Then, the request sequence we construct is given by $\bar{r} = (A^L|B^L)^P$. Here, $A^L|B^L$ denotes repeating sub-sequence A exactly L-times and then repeating sub-sequence B exactly L-times, and $(A^L|B^L)^P$ means that we repeat $(A^L|B^L)$ exactly P-times. We will refer to the sub-sequence $(A^L|B^L)$ as a *phase*, and the instance consists by P-phases. Also, it should be clear that the size of the request sequence is $4kLP$.

Notice that request sub-sequence A was constructed in such a way such that the only configuration that serves the entire sub-sequence A is $[1, 0, \dots, 0]$. To show this, a simple pigeon principle argument suffices. Respectively, the only configuration that serves the entire sub-sequence B is $[0, 0, \dots, 0]$.

First, we will argue that the optimal cost for this instance is at most $2P$. To see this, we only need to construct a solution with total movement cost $2P$ and then the claim follows. Now, consider a solution that serves A^L by moving to $[1, 0, 0, \dots, 0]$ and B^L by moving to $[0, 0, \dots, 0]$. Since the Hamming distance between these configurations is 1 and the starting configuration is $[0, 0, \dots, 0]$, the total movement cost of this solution is indeed exactly $2P$. Thus, if we use OPT to denote the cost of the optimal solution we get $OPT \leq 2P$.

We will now show that the expected cost of the harmonic algorithm on this instance is at least $2Pk\alpha_k$, proving the lower bound. To do so, we begin with the following lemma.

Lemma 4.6. *If the harmonic algorithm starts from any configuration other that* $[1, 0 \dots, 0]$ *(resp.* $[0, 0, \dots, 0]$*) then the expected movement cost until it reaches configuration* $[1, 0, \dots, 0]$ *(resp.* $[0, 0, \dots, 0]$*) when presented with request sub-sequence* A^L *(resp.* B^L*) is at least* $\min(k\alpha_k, L)$.

Proof. We will prove the lemma for the first case. Suppose any initial configuration $q \neq [1, 0, \dots, 0]$ and let $\ell = d_H(q, [1, 0, \dots, 0]) > 0$ be the initial number of "wrong" servers in q. We distinguish between two scenarios:

1. The algorithm never reaches $[1, 0, \dots, 0]$ while serving A^L. Then, since the only configuration that serves the entire sub-sequence A is $[1, 0, \dots, 0]$, we immediately get that the algorithm moves at-least once to serve A and thus its total movement cost will be at least L.

2. The algorithm reaches $[1, 0, \dots, 0]$ at some point. Notice that every request in A is served by exactly one server in $[1, 0, \dots, 0]$. We can ignore any request that is served by the algorithm without forcing it to move, since they don't affect the movement cost. The algorithm initially has ℓ non-optimal points, and in every request it is presented with only one optimal point. Furthermore, every point that is already optimal can be moves to a non-optimal position. This forms a Harmonic Chain, exactly like the one we used for the upper bound of the Harmonic Algorithm, and we already know that the expected cost (starting from $\ell > 0$ non-optimal points) is at-least $k\alpha_k$ (tight only for $\ell = 1$).

Combining both cases, we get that the expected movement cost of the algorithm is at least $min(k\alpha_k, L)$. □

We are now ready to bound the total movement cost of the algorithm. From the previous lemma, in any sub-sequence A^L or B^L, the algorithm is expected to pay:

- 0 if the initial configuration was the optimal for the sub-sequence.
- At least L if the optimal of the sub-sequence was never reached.
- $k\alpha_k$ on expectation if the optimal of the sub-sequence was eventually reached.

Let x be the number of sub-sequences such that case 1 holds (the algorithm pays nothing). By construction, this does not apply to the first sub-sequence A^L. Now, we claim that at least in x sub-sequences case 2 happened. This is true, since if at the beginning of the sub-sequence the algorithm is already in the optimal configuration (case 1), then this means that the optimal configuration of the previous sub-sequence was never reached (case 2).

Thus, the total movement cost of the Harmonic Algorithm is on expectation at least

$$x \cdot 0 + x \cdot L + (2P - 2x)k\alpha_k = 2Pk\alpha_k + x(L - 2k\alpha_k)$$

Here, we have assumed that $L \geq 2k\alpha_k$ so that in the best case there are only x sub-sequences where the algorithm paid L. By setting $L = 2k\alpha_k$, we get that the expected movement cost of the algorithm is lower bounded by $2Pk\alpha_k$ and since $OPT \leq 2P$, we get that the competitive ratio of the Harmonic Algorithm against oblivious adversaries is at least $k \cdot \alpha_k$.

4.4.2 Special Case: Two Point Metrics

We will prove Theorem 4.5, that is that in metric spaces with $n = 2$ points, the competitive ratio of the Harmonic Algorithm is $\Omega(2^k)$. We will show this lower bound by constructing a request sequence where the (expected) cost of the Harmonic Algorithm is at least $\Theta(2^k)$ times that of the optimal solution.

By re-labeling, we can assume that each metric M_i has $n = 2$ points labeled $\{0, 1\}$ and that the initial configuration is $q^0 = [0, 0, \dots, 0] \in \{0, 1\}^k$.

For any request $r \in \{0, 1\}^k$, we will use $R(r) = \{0, 1\}^k \setminus \{\bar{r}\}$ to denote a request sequence with every possible request other that $\bar{r}$ which is the anti-configuration of r. In the sequence, the requests can be ordered in an arbitrary way (for example in binary order). Clearly we have $|R(r)| = 2^k - 1$. Finally, we fix 2 "central" configurations $\mathcal{O}_1 = [1, 0, \dots, 0]$ and $\mathcal{O}_2 = [0, 0, \dots, 0]$.

Then, for some $L, P \in \mathbb{N}$ to be defined later, the oblivious request sequence we construct is given by

$$\bar{r} = (R^L(\mathcal{O}_1) | R^L(\mathcal{O}_2))^P$$

where $R^L(\mathcal{O}_1)$ denotes that we repeat $R(\mathcal{O}_1)$ L-times, $R^L(\mathcal{O}_2)$ denotes that we repeat $R(\mathcal{O}_2)$ L-times and $(R^L(\mathcal{O}_1)|R^L(\mathcal{O}_2))^P$ denotes that we repeat $R^L(\mathcal{O}_1)$ followed by $R^L(\mathcal{O}_2)$ exactly P-times.

We first argue that the optimal movement cost for this instance is at most $2P$. By definition of $R(r)$, we know that configuration r is the only configuration that serves the entire sequence $R(r)$. Thus, we can construct a solution of total movement cost $2P$ that interpolates between configurations $\mathcal{O}_1$ and $\mathcal{O}_2$ to serve $R(\mathcal{O}_1)$ and $R(\mathcal{O}_2)$ respectively. Since $q^0 = \mathcal{O}_2$ and $d_H(\mathcal{O}_1, \mathcal{O}_2) = 1$, we get that the total movement cost of this solution (which is an upper bound for the optimal movement cost) is $2P$.

We will analyze the (expected) movement cost of the harmonic algorithm in this instance. Assume that the algorithm is at some configuration q_i and needs to serve the request sub-sequence $R^L(\mathcal{O}_1)$. We distinguish between the following cases:

1. Assume $q_i \neq \mathcal{O}_1$. Then, the algorithm will either reach $\mathcal{O}_1$ eventually and then pay nothing for the rest of the sub-sequence, or it will never reach $\mathcal{O}_1$ and pay at-least L for the sub-sequence (since only $\mathcal{O}_1$ can serve the entire $R(\mathcal{O}_1)$. In order to reach $\mathcal{O}_1$ the algorithm will have to pay $\Theta(2^k)$ on expectation, by the standard binary chain arguments we have used so far.

2. Assume $q_i = \mathcal{O}_1$. Then, the algorithm will not move for the entire sub-sequence $R^L(\mathcal{O}_1)$. However, this means that the algorithm didn't reach $\mathcal{O}_2$ during the previous sub-sequence $R^L(\mathcal{O}_2)$ and thus paid L.

The exact same case-analysis can be done for the sub-sequence $R^(\mathcal{O}_2)$. Now, let x be the number of sub-sequences where the harmonic algorithm paid 0. By case 2, this means that there were at least x sub-sequences where the algorithm paid L. By setting L to be asymptotically larger that 2^k (for example 2^{2^k}), we get that in the best case the total movement cost of the harmonic algorithm is on expectation

$$x \cdot 0 + x \cdot 2^{2^k} + (2P - 2x)\Theta(2^k)$$

which is at least $2P\Theta(2^k)$, concluding the lower bound.

Chapter 5

Concluding Remarks

In this thesis, we provided tight bounds on the competitive ratio of randomized memoryless algorithms for generalized k-server in uniform metrics. Combining our results with the work of Chiplunkar and Vishwanathan [28], the power of memoryless algorithms in uniform and weighted uniform metrics is completely characterized for the online adaptive adversarial setting.

We also made significant progress in the study of the oblivious setting, by proving that the Harmonic Algorithm, which is the optimal memoryless algorithm in the online adaptive setting, has the same competitive ratio against oblivious and adaptive online adversaries.

As we have already mentioned, for uniform metrics, if memory is allowed and we compare against oblivious adversaries, competitive randomized algorithms are known: Bansal et. al. [11] designed a $O(k^3 \log k)$-competitive randomized algorithm with memory; this was recently improved to $O(k^2 \log k)$ by Bienkowski et. al. [15]. This is perhaps surprising, since usually in the uniform case the competitive ratio of deterministic algorithms (with memory) and memoryless randomized algorithms is essentially the same; this is true for paging (k-server on uniform metric spaces) and generalized k-server on weighted uniform metric spaces. However, our results imply that this doesn't hold for the generalized k-server problem on uniform metrics.

It might be interesting to determine the power of memoryless algorithms for other metric spaces such as e.g., weighted stars. However we note that memoryless algorithms are not competitive on arbitrary metric spaces, even for $k = 2$; this was shown by Chrobak and Sgall [32] and Koutsoupias and Taylor [58] independently.

All of the current results about the generalized k-server problem (for arbitrary k) apply to uniform metric spaces or weighted uniform metric spaces. The next big step would be to get competitive algorithms for more general metric spaces such as stars and trees, as it has already been done for the k-server problem.

Bibliography

[1] Dimitris Achlioptas, Marek Chrobak, and John Noga. Competitive analysis of randomized paging algorithms. *Theor. Comput. Sci.*, 234(1-2):203–218, 2000.

[2] Susanne Albers. Improved randomized on-line algorithms for the list update problem. *SIAM J. Comput.*, 27(3):682–693, 1998.

[3] Susanne Albers and Jeffery Westbrook. Self-organizing data structures. In *Online Algorithms, The State of the Art*, pages 13–51, 1996.

[4] Christoph Ambühl. Offline list update is np-hard. In *Proceedings of the 8th Annual European Symposium on Algorithms*, ESA '00, page 42–51, Berlin, Heidelberg, 2000. Springer-Verlag.

[5] Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau. *Operating Systems: Three Easy Pieces.* CreateSpace Independent Publishing Platform, North Charleston, SC, USA, 2018.

[6] Yossi Azar, Andrei Z. Broder, and Mark S. Manasse. On-line choice of on-line algorithms. In *Proceedings of the Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms (SODA)*, pages 432–440, 1993.

[7] Nikhil Bansal, Niv Buchbinder, Aleksander Madry, and Joseph Naor. A polylogarithmic-competitive algorithm for the k-server problem. *J. ACM*, 62(5):40, 2015.

[8] Nikhil Bansal, Niv Buchbinder, and Joseph Naor. Towards the randomized k-server conjecture: A primal-dual approach. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 40–55, 2010.

[9] Nikhil Bansal, Niv Buchbinder, and Joseph Naor. A primal-dual randomized algorithm for weighted paging. *J. ACM*, 59(4):19:1–19:24, 2012.

[10] Nikhil Bansal, Marek Eliáš, and Grigorios Koumoutsos. Weighted k-server bounds via combinatorial dichotomies. In *58th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 493–504, 2017.

[11] Nikhil Bansal, Marek Eliáš, Grigorios Koumoutsos, and Jesper Nederlof. Competitive algorithms for generalized k-server in uniform metrics. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 992–1001, 2018.

[12] Yair Bartal and Eddie Grove. The harmonic k-server algorithm is competitive. *J. ACM*, 47(1):1–15, 2000.

[13] Yair Bartal and Elias Koutsoupias. On the competitive ratio of the work function algorithm for the k-server problem. *Theor. Comput. Sci.*, 324(2–3):337–345, 2004.

[14] Shai Ben-David, Allan Borodin, Richard M. Karp, Gábor Tardos, and Avi Wigderson. On the power of randomization in on-line algorithms. *Algorithmica*, 11(1):2–14, 1994.

[15] Marcin Bienkowski, Lukasz Jez, and Pawel Schmidt. Slaying hydrae: Improved bounds for generalized k-server in uniform metrics. In *30th International Symposium on Algorithms and Computation, ISAAC 2019*, pages 14:1–14:14, 2019.

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

[17] Allan Borodin, Nathan Linial, and Michael E. Saks. An optimal on-line algorithm for metrical task system. *J. ACM*, 39(4):745–763, 1992.

[18] Peter Brucker. *Scheduling Algorithms*. Springer-Verlag, Berlin, Heidelberg, 3rd edition, 2001.

[19] Sébastien Bubeck, Michael B. Cohen, Yin Tat Lee, James R. Lee, and Aleksander Madry. k-server via multiscale entropic regularization. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 3–16, 2018.

[20] Niv Buchbinder, Anupam Gupta, Marco Molinaro, and Joseph (Seffi) Naor. k-servers with a smile: Online algorithms via projections. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*, pages 98–116, 2019.

[21] Niv Buchbinder and Joseph Naor. The design of competitive online algorithms via a primal-dual approach. *Foundations and Trends in Theoretical Computer Science*, 3(2-3):93–263, 2009.

[22] Niv Buchbinder and Joseph Naor. Online primal-dual algorithms for covering and packing. *Math. Oper. Res.*, 34(2):270–286, 2009.

[23] William R. Burley. Traversing layered graphs using the work function algorithm. *J. Algorithms*, 20(3):479–511, 1996.

[24] Ka Chan, C. Lenard, and Terence Mills. An introduction to markov chains, 2012.

[25] Bo Chen, Chris Potts, and Gerhard Woeginger. *A Review of Machine Scheduling: Complexity, Algorithms and Approximability.* Springer-Verlag, 01 1998.

[26] Hong Chen and David D. Yao. *Birth-Death Queues.* Springer New York, 2001.

[27] Ashish Chiplunkar and Sundar Vishwanathan. On randomized memoryless algorithms for the weighted k-server problem. In *FOCS*, pages 11–19, 2013.

[28] Ashish Chiplunkar and Sundar Vishwanathan. Randomized memoryless algorithms for the weighted and the generalized k-server problems. *ACM Trans. Algorithms*, 16(1):14:1–14:28, 2020.

[29] Marek Chrobak, Howard J. Karloff, Thomas H. Payne, and Sundar Vishwanathan. New results on server problems. *SIAM J. Discrete Math.*, 4(2):172–181, 1991.

[30] Marek Chrobak and Lawrence L. Larmore. The server problem and on-line games. In *On-line Algorithms, volume 7 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 11–64. AMS/ACM, 1992.

[31] Marek Chrobak and Lawrence L Larmore. *Metrical Service System: Deterministic Strategies.* Citeseer, 1993.

[32] Marek Chrobak and Jiří Sgall. The weighted 2-server problem. *Theor. Comput. Sci.*, 324(2-3):289–312, 2004.

[33] Don Coppersmith, Peter Doyle, Prabhakar Raghavan, and Marc Snir. Random walks on weighted graphs and applications to on-line algorithms. *J. ACM*, 40(3):421–453, 1993.

[34] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition.* The MIT Press, 3rd edition, 2009.

[35] Yuval Emek, Pierre Fraigniaud, Amos Korman, and Adi Rosén. On the additive constant of the k-server work function algorithm. In *Approximation and Online Algorithms, 7th International Workshop (WAOA)*, pages 128–134, 2009.

[36] Amos Fiat, Richard M. Karp, Michael Luby, Lyle A. McGeoch, Daniel Dominic Sleator, and Neal E. Young. Competitive paging algorithms. *J. Algorithms*, 12(4):685–699, 1991.

[37] Amos Fiat and Manor Mendel. Better algorithms for unfair metrical task systems and applications. *SIAM J. Comput.*, 32(6):1403–1422, 2003.

[38] Amos Fiat, Yuval Rabani, and Yiftach Ravid. Competitive k-server algorithms. *J. Comput. Syst. Sci.*, 48(3):410–428, 1994.

[39] Amos Fiat, Yuval Rabani, Yiftach Ravid, and Baruch Schieber. A deterministic o(k^3)-competitive k-server algorithm for the circle. *Algorithmica*, 11(6):572–578, 1994.

[40] Amos Fiat and Moty Ricklin. Competitive algorithms for the weighted server problem. *Theor. Comput. Sci.*, 130(1):85–99, 1994.

[41] Amos Fiat and Gerhard J. Woeginger, editors. *Online Algorithms, The State of the Art (the book grow out of a Dagstuhl Seminar, June 1996)*, volume 1442 of *Lecture Notes in Computer Science*. Springer, 1998.

[42] Dimitris Fotakis, Loukas Kavouras, Grigorios Koumoutsos, Stratis Skoulakis, and Manolis Vardas. The online min-sum set cover problem. *ArXiv*, abs/2003.02161, 2020.

[43] Michael Goodrich and Roberto Tamassia. *Algorithm design: Foundations, analysis, and Internet examples.* Wiley Publishing, 01 2002.

[44] Michael T. Goodrich, Roberto Tamassia, and Michael H. Goldwasser. *Data Structures and Algorithms in Java.* Wiley Publishing, 6th edition, 2014.

[45] D. Green and D. Knuth. *Recurrence Relations*, pages 11–30. Birkhäuser Boston, Boston, MA, 1990.

[46] G.R. Grimmett and D.R. Stirzaker. *Probability and random processes*, volume 80. Oxford university press, 2001.

[47] Charles M. Grinstead and J. Laurie Snell. *Introduction to Probability.* AMS, 2003.

[48] Edward F. Grove. The harmonic online k-server algorithm is competitive. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 260–266, 1991.

[49] John Iacono. In pursuit of the dynamic optimality conjecture. In *Space-Efficient Data Structures, Streams, and Algorithms - Papers in Honor of J. Ian Munro on the Occasion of His 66th Birthday*, pages 236–250, 2013.

[50] Kazuo Iwama and Kouki Yonezawa. Axis-bound CNN problem. *IEICE TRANS*, pages 1–8, 2001.

[51] Yuri M. Kaniovski, Georg Ch. Pflug, and Georg Ch. Pflug. Limit theorems for stationary distributions of birth-and-death processes. *Communications in Statistics. Stochastic Models*, 15(1):103–123, 1999.

[52] Richard M. Karp. On-line algorithms versus off-line algorithms: How much is it worth to know the future? In *Proceedings of the IFIP 12th World Computer Congress on Algorithms, Software, Architecture - Information Processing '92, Volume 1 - Volume I*, page 416–429, NLD, 1992. North-Holland Publishing Co.

[53] G. Koumoutsos. *Algorithms for k-server problems.* PhD thesis, Department of Mathematics and Computer Science, 9 2018. Proefschrift.

[54] Elias Koutsoupias. Weak adversaries for the k-server problem. In *Proc. of the 40th Symp. on Foundations of Computer Science (FOCS)*, pages 444–449, 1999.

[55] Elias Koutsoupias. The k-server problem. *Computer Science Review*, 3(2):105–118, 2009.

[56] Elias Koutsoupias and Christos H. Papadimitriou. On the k-server conjecture. *J. ACM*, 42(5):971–983, 1995.

[57] Elias Koutsoupias and Christos H. Papadimitriou. The 2-evader problem. *Inf. Process. Lett.*, 57(5):249–252, 1996.

[58] Elias Koutsoupias and David Scot Taylor. The CNN problem and other k-server variants. *Theor. Comput. Sci.*, 324(2-3):347–359, 2004.

[59] Gregory F. Lawler and Vlada Limic. *Random Walk: A Modern Introduction.* Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2010.

[60] James R. Lee. Fusible hsts and the randomized k-server conjecture. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018*, pages 438–449, 2018.

[61] Mark S. Manasse, Lyle A. McGeoch, and Daniel D. Sleator. Competitive algorithms for server problems. *J. ACM*, 11(2):208–230, 1990.

[62] Lyle A. McGeoch and Daniel Dominic Sleator. A strongly competitive randomized paging algorithm. *Algorithmica*, 6(6):816–825, 1991.

[63] Lee Newberg. The k-server problem with distinguishable servers. *Master's Thesis, Univ. of California at Berkeley*, 1991.

[64] Artem S. Novozhilov, Georgy P. Karev, and Eugene V. Koonin. Biological applications of the theory of birth-and-death processes. *Briefings in Bioinformatics*, 7(1):70–85, 03 2006.

[65] Nick Reingold, Jeffery Westbrook, and Daniel Dominic Sleator. Randomized competitive algorithms for the list update problem. *Algorithmica*, 11(1):15–32, 1994.

[66] Sidney I. Resnick. *Adventures in Stochastic Processes.* Birkhauser Verlag, CHE, 1992.

[67] Russell Schwartz. Stochastic Modelling for Systems Biology.Darren J. Wilkinson. *Briefings in Bioinformatics*, 8(3):204–205, 02 2007.

[68] René Sitters. The generalized work function algorithm is competitive for the generalized 2-server problem. *SIAM J. Comput.*, 43(1):96–125, 2014.

[69] René Sitters, Leen Stougie, and Willem de Paepe. A competitive algorithm for the general 2-server problem. In *ICALP*, pages 624–636, 2003.

[70] René A. Sitters and Leen Stougie. The generalized two-server problem. *J. ACM*, 53(3):437–458, 2006.

[71] Daniel Dominic Sleator and Robert Endre Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28(2):202–208, 1985.

[72] Daniel Dominic Sleator and Robert Endre Tarjan. Self-adjusting binary search trees. *J. ACM*, 32(3):652–686, 1985.

[73] J. Steele. *Stochastic Calculus and Financial Applications*, volume 5. Springer, 01 2001.

[74] R. Steyer and W. Nagel. *Probability and Conditional Expectation: Fundamentals for the Empirical Sciences*. Wiley Series in Probability and Statistics. Wiley, 2017.

[75] Philipp Von, Hilgers, and Amy N. Langville. The five greatest applications of markov chains.

[76] Zikun Wang and Hsiang-chun Yang. *Birth and death processes and Markov chains*. Berlin; New York : Springer, rev. edition, 1992. "Revised edition of the original Chinese edition".

[77] Geoffrey I. Webb. *Bayes' Rule*, pages 99–99. Springer US, Boston, MA, 2017.

[78] Xinye Yang. Markov chain and its applications, 2020.

[79] Neal E. Young. The k-server dual and loose competitiveness for paging. *Algorithmica*, 11(6):525–541, 1994.