ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΛΟΓΙΚΗΣ ΚΑΙ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Μοντέλα Τυχαίας Διάταξης και Προφήτη για Άμεσα Επαυξήσιμα Ακέραια Προγράμματα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΛΕΞΑΝΔΡΟΣ ΚΟΥΡΙΔΑΚΗΣ

Επιβλέπων : Δημήτριος Φωτάκης
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2024

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Λογικής και Επιστήμης Υπολογιστών

Random Order and Prophet Models for Online Augmentable Integer Programmes

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΛΕΞΑΝΔΡΟΣ ΚΟΥΡΙΔΑΚΗΣ

Επιβλέπων : Δημήτριος Φωτάκης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 11η Ιουλίου 2024.

....	....	....
Δημήτριος Φωτάκης	Αριστείδης Παγουρτζής	Χρήστος Τζάμος
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Αν. Καθηγητής Ε.Κ.Π.Α.

Αθήνα, Ιούλιος 2024

....
Αλέξανδρος Κουριδάκης
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Στην παρούσα διπλωματική μελετάμε προβλήματα της οικογένειας των Άμεσων Επαυξήσιμων Ακέραιων Προγραμμάτων (Online Augmentable Integer Programmes, AIPs) στα μοντέλα Τυχαίας Διάταξης και Προφήτη. Τα δύο αυτά μοντέλα είναι λιγότερο απαισιόδοξα από την κλασσική ανάλυση χειρότερης περίπτωσης, παραμένοντας πάρα τάυτα ιδιαίτερα ελκυστικά από θεωρητικής σκοπιάς, με αποτέλεσμα πλήθος προβλημάτων να έχει αναλυθεί υπό το πρίσμα τους. Ξεκινάμε εξετάζοντας διάφορα σημαντικά προβλήματα που έχουν μελετηθεί σε καθένα από τα δύο μοντέλα, δίνοντας ιδιαίτερη έμφαση στα θεμελιώδη Secretary Problem (στο μοντέλο Τυχαίας Διάταξης) και Prophet Inequality Problem (στο οποίο οφείλει το όνομά του το μοντέλο Προφήτη), καθώς και στις επεκτάσεις τους. Για κάθε πρόβλημα που εξετάζουμε, παρουσιάζουμε και αναλύουμε γνωστούς, απλούς αλγορίθμους που παρ' όλα αυτά είναι βέλτιστοι σε κάποιες περιπτώσεις και σχεδόν βέλτιστοι στις υπόλοιπες. Επίσης αναφέρουμε τα τρέχοντα καλύτερα άνω και κάτω φράγματα για κάθε πρόβλημα.

Έπειτα, εστιάζουμε στην κλάση των AIPs, η οποία περιλαμβάνει διάφορα ενδιαφέροντα προβλήματα, μεταξύ των οποίων τα Άμεσο Κάλυμμα Συνόλων (Online Set Cover), Άμεσα Ακέραια Προγράμματα Κάλυψης (Online Covering Integer Programmes), Άμεση Μετρική και Μη-Μετρική Χωροθέτηση Υπηρεσιών (Online Metric and Non-Metric Facility Location) και Άμεσο Δέντρο Steiner (Online Steiner Tree). Σε όλα αυτά τα προβλήματα πλην του τελευταίου, πρόσφατα αποτελέσματα έχουν οδηγήσει σε αλγορίθμους για το μοντέλο Τυχαίας Διάταξης, των οποίων οι λόγοι ανταγωνιστικότητας είναι σημαντικά βελτιωμένοι συγκριτικά με τους καλύτερους εφικτούς στο πλήρως ανταγωνιστικό μοντέλο. Επιπλέον, μία πρόσφατα αποδεδειγμένη αναγωγή δείχνει πως η ύπαρξη ενός ανταγωνιστικού αλγορίθμου για ένα AIP στο μοντέλο Τυχαίας Διάταξης συνεπάγεται την ύπαρξη ενός ανταγωνιστικού αλγορίθμου στο μοντέλο Προφήτη. Παρουσιάζουμε τους αλγορίθμους και τα αποτελέσματα αυτά λεπτομερώς, εξετάζοντας σε βάθος τη θεωρητική ανάλυσή τους και τη διαίσθηση στην οποία βασίζονται. Τέλος, κάνουμε ιδιαίτερη αναφορά στο Άμεσο Δέντρο Steiner, για το οποίο είναι γνωστό ότι, παρότι είναι εύκολο να αντιμετωπιστεί στο μοντέλο Προφήτη, εντούτοις στο μοντέλο Τυχαίας Διάταξης δεν είναι ευκολότερο από ότι στο πλήρως ανταγωνιστικό μοντέλο.

Λέξεις Κλειδιά

Άμεσο Κάλυμμα Συνόλων, Άμεση Χωροθέτηση Υπηρεσιών, Άμεσο Δέντρο Steiner, Μοντέλο Τυχαίας Διάταξης, Μοντέλο Προφήτη, Άμεσοι Αλγόριθμοι

Abstract

In this diploma thesis we study problems in the class of Online Augmentable Integer Programmes (AIPS, introduced in [1]) under the Random-Order and Prophet models. Both of those models allow us to tackle problems in rich settings that are less pessimistic than the classical fully-adversarial context, thus they have been applied to a number of problems and led to intriguing theoretical results. We begin by examining various prominent problems that have been studied in each of the two models, giving particular emphasis to the fundamental Secretary Problem (in the Random-Order model) and Prophet Inequality Problem (from which the Prophet model takes its name), as well as their extensions. For each problem we study, we present and analyse well-known, simple algorithms that are nonetheless optimal in a few cases and nearly-optimal in the rest. We also mention the current best known upper and lower bounds for each problem.

Afterwards, we focus on the class of AIPs, which contains a number of interesting problems, particularly Online Set Cover, Online Covering Integer Programmes, Online Metric and Non-Metric Facility Location, as well as Online Steiner Tree. For all but the last problem, recent developments have led to algorithms in the Random-Order model, whose competitive ratios significantly improve upon the best possible for the fully-adversarial setting. Additionally, a recently proven reduction shows how the existence of a competitive algorithm for an AIP in the Random-Order model implies the existence of a competitive algorithm for the Prophet model. We present those algorithms and results in detail, closely examining their analyses and the intuition behind them. Finally, we make special note of the Online Steiner Tree problem, for which it is known that, while it is easy to tackle in the Prophet case, the Random-Order model is in fact not easier than the fully-adversarial setting.

Keywords

Online Set Cover, Online Facility Location, Online Steiner Tree, Beyond-Worst-Case Analysis, Random Order Models, Prophet Models, Online Algorithms

Ευχαριστίες

Πρώτα από όλα, θα ήθελα να ευχαριστήσω τον επιβλέποντα της διπλωματικής μου εργασίας, τον κύριο Δημήτρη Φωτάκη, αρχικά επειδή μέσα από την εξαιρετική διδασκαλία του αγάπησα το αντικείμενο τον αλγορίθμων και επέλεξα να ασχοληθώ με αυτό. Κυρίως όμως επειδή, τον τελευταίο ενάμιση χρόνο, μου έχει προσφέρει ανεκτίμητη υποστήριξη, βοηθώντας με να γνωρίσω τον κόσμο της Θεωρητικής Πληροφορικής, να ανακαλύψω τις πραγματικές μου κλίσεις και, σημαντικότερο από όλα, να αποφασίσω ποιο θέλω να είναι το επόμενο μεγάλο βήμα στη ζωή μου και να το πραγματοποιήσω. Χωρίς την καθοριστική επίδρασή του και την εμπιστοσύνη που μου έδειξε, σίγουρα δεν θα είχα φτάσει στο σημείο όπου βρίσκομαι τώρα, και για αυτό τον ευχαριστώ βαθύτατα. Έπειτα, θέλω να ευχαριστήσω τα εξαιρετικά μέλη του CoReLab που είχα τη χαρά να γνωρίσω αυτό και το προηγούμενο ακαδημαϊκό έτος. Οι εβδομαδιαίες μας συναντήσεις στο πλαίσιο του study group με έκαναν να αισθάνομαι ότι ανήκω σε μία ζωντανή ακαδημαϊκή κοινότητα, γεμάτη με χαρισματικά άτομα, με τα οποία μπορούσα πάντα να κάνω ενδιαφέρουσες συζητήσεις, επιστημονικού ή μη περιεχομένου. Ιδιαίτερα θέλω να ευχαριστήσω τα άτομα εκείνα με τα οποία η επαφή μου επεκτάθηκε και πέρα από τα πλαίσια του study group, τα οποία θεωρώ φίλες και φίλους μου.

Περισσότερο από οποιονδήποτε, όμως, θέλω να ευχαριστήσω τη Νικόλ, η οποία είναι δίπλα μου από την αρχή της φοιτητικής μου σταδιοδρομίας μέχρι και σήμερα, είναι το σταθερό σημείο στη ζωή μου όπου μπορώ πάντα να βασιστώ, μοιράζεται τις χαρές μου και με υποστηρίζει ανελλιπώς στις δύσκολες στιγμές μου, με βοηθάει να είμαι πάντα η καλύτερη εκδοχή του εαυτού μου και γενικώς μου δείχνει με κάθε τρόπο την έμπρακτη αγάπη της. Σε αυτήν αφιερώνω την παρούσα διπλωματική.

Αλέξανδρος Κουριδάκης,

Αθήνα, 11η Ιουλίου 2024

Περιεχόμενα

Εκτεταμένη Ελληνική Περίληψη

0.1 Εισαγωγή

Μία βασική πρόκληση που ανακύπτει σε διάφορες καταστάσεις λήψης αποφάσεων είναι η διαχείριση της αβεβαιότητας για το μέλλον. Σε καθημερινή βάση και σε ποικίλα περιβάλλοντα (π.χ. στην επαγγελματική, οικονομική ή προσωπική μας δραστηριότητα), καλούμαστε να επιλέξουμε μία ενέργεια για την αντιμετώπιση ενός προβλήματος, δίχως να έχουμε πλήρη επίγνωση των αποτελεσμάτων αυτής της ενέργειας σε βάθος χρόνου. Το επιστημονικό πεδίο των *άμεσων* αλγορίθμων ασχολείται με προβλήματα της Θεωρητικής Επιστήμης Υπολογιστών στα οποία καλούμαστε να αντιμετωπίσουμε αυτήν ακριβώς την αβεβαιότητα. Σε ένα τέτοιο πρόβλημα, η είσοδος αποτελείται από n αντικείμενα τα οποία φανερώνονται σειριακά, σε γύρους. Κατά την άφιξη ενός αντικειμένου καλούμαστε να λάβουμε αμέσως μία αμετάκλητη απόφαση. Σε προβλήματα μεγιστοποίησης, αυτή η απόφαση αφορά στο εάν θα θυσιάσουμε πόρους προκειμένου να καρπωθούμε την αξία του αντικειμένου, ενώ σε προβλήματα ελαχιστοποίησης η απόφαση αφορά στο πώς θα ικανοποιήσουμε τους περιορισμούς του προβλήματος που επιβάλλονται από το αντικείμενο – συνήθως επωμιζόμενοι κάποιο κόστος. Η τελική μας επίδοση συγκρίνεται με την επίδοση ενός βέλτιστου *offline* αλγορίθμου ο οποίος γνωρίζει εκ των προτέρων ολόκληρη την είσοδο. Εάν το κέρδος (αντ. κόστος) ενός άμεσου αλγορίθμου είναι το πολύ α φορές μεγαλύτερο (αντ. μικρότερο) από το κόστος του offline βέλτιστου *για οποιαδήποτε ακολουθία εισόδου*, τότε καλούμε τον αλγόριθμο αυτόν α-ανταγωνιστικό (α-competitive), και το α καλείται ο *λόγος ανταγνωιστικότητας* (competitive ratio).

Το πεδίο των άμεσων αλγορίθμων έχει γνωρίσει σημαντική άνθιση. Ενδεικτικά, ορισμένα σημαντικά προβλήματα που έχουν μελετηθεί είναι το Κάλυμμα Συνόλων (Set Cover, [2]–[4]), η Χωροθέτηση Υπηρεσιών (Facility Location, [5]–[8]), το Δέντρο Steiner (Steiner Tree, [9]), το πρόβλημα των k Εξυπηρετητών (k-Server Problem [10]–[13]) και τα Μετρικά Συστήματα Εργασιών (Metrical Task Systems, [14]–[16]). Συνήθως, ένας άμεσος αλγόριθμος αναλύεται στο *πλήρως ανταγωνιστικό* μοντέλο, όπου ένας αντίπαλος κατασκευάζει ολόκληρη την ακολουθία εισόδου, και αξιολογούμε τον αλγόριθμο βάσει της επίδοσής του στη χειρότερη δυνατή είσοδο που μπορεί να κατασκευάσει ο αντίπαλος. Παρότι αυτό το μοντέλο δίνει ισχυρές θεωρητικές εγγυήσεις, εντούτοις είναι αρκετά απαισιόδοξο, υπό την έννοια ότι η επίδοση ενός αλγορίθμου στην πράξη μπορεί να είναι σημαντικά καλύτερη από ότι προβλέπει η ανάλυσή του στο πλήρως ανταγωνιστικό μοντέλο. Για τον λόγο αυτόν, έχουν προταθεί διάφορα πιο ελαστικά μοντέλα αξιολόγησης άμεσων αλγορίθμων, στόχος των οποίων είναι να αντικατοπτρίζουν καλύτερα την επίδοση ενός αλγορίθμου σε ρεαλιστικά σενάρια, ή να ενσωματώνουν υπάρχουσα γνώση για τις ακολουθίες εισόδου ενός προβλήματος που αναμένουμε να αντιμετωπίσουμε στην πράξη.

Οι δύο κατηγορίες μοντέλων που θα μας απασχολήσουν είναι τα *Μοντέλα Τυχαίας Διάταξης* (Random-Order Models) και τα *Μοντέλα Προφήτη* (Prophet Models). Στην πρώτη περίπτωση, ο αντίπαλος επιλέγει τα αντικείμενα που θα αποτελέσουν την ακολουθία εισόδου, αλλά η σειρά εμφάνισής τους είναι ομοιόμορφα τυχαία ανάμεσα σε όλες τις πιθανές διατάξεις αυτών των αντικειμένων. Για ένα δεδομένο σύνολο αντικειμένων εισόδων, ενδιαφερόμαστε για την *αναμενόμενη* επίδοση ενός αλγορίθμου πάνω σε όλες τις πιθανές διατάξεις εισόδου, ενώ και πάλι συγκρινόμαστε με τη βέλτιστη offline λύση για αυτό το σύνολο αντικειμένων. Όσον αφορά τα Μοντέλα Προφήτη, εδώ ο αντίπαλος επιλέγει εκ των προτέρων μία ακολουθία κατανομών πάνω στα πιθανά αντικείμενα εισόδου, και κατά την εκτέλεση του αλγορίθμου κάθε αντικείμενο που εμφανίζεται είναι τραβηγμένο από την κατανομή που του αντιστοιχεί. Πάλι ενδιαφερόμαστε για την αναμενόμενη επίδοση του αλγορίθμου μας πάνω σε όλες τις δυνατές υλοποιήσεις της ακολουθίας εισόδου, αλλά πλέον συγκρινόμαστε με την *αναμενόμενη* βέλτιστη offline λύση (η οποία είναι ανεξάρτητη από την υλοποιημένη ακολουθία εισόδου).

0.2 Μοντέλα Τυχαίας Διάταξης

Ίσως το πιο κλασσικό πρόβλημα που έχει μελετηθεί στο μοντέλο Τυχαίας Διάταξης είναι το Secretary Problem, μία ιστορική αναδρομή του οποίου παρατίθεται στο [17]. Σε αυτό το πρόβλημα δίνονται n αντικείμενα και ένας αντίπαλος επιλέγει μία μη-αρνητική πραγματική αξία για καθένα εξ αυτών. Τα αντικείμενα παρουσιάζονται σειριακά σε τυχαία διάταξη. Όταν εμφανίζεται ένα αντικείμενο, πρέπει αμέσως είτε να το απορρίψουμε αμετάκλητα είτε να το επιλέξουμε, και μπορούμε να επιλέξουμε μόνο ένα αντικείμενο συνολικά. Επιθυμούμε να μεγιστοποιήσουμε την αξία του αντικειμένου που επιλέγουμε, και συγκρινόμαστε με τη μέγιστη αξία ανάμεσα σε όλα τα αντικείμενα.

Για αυτό το πρόβλημα, γνωρίζουμε έναν απλό αλγόριθμο που, για μεγάλες τιμές του n, επιλέγει το αντικείμενο μέγιστης αξίας με πιθανότητα τουλάχιστον $\frac{1}{e}$, άρα είναι τουλάχιστον $\frac{1}{e}$-ανταγωνιστικός. Ο αλγόριθμος αυτός αρχικά απορρίπτει τα πρώτα $\frac{n}{e}$ αντικείμενα και θέτει ένα φράγμα τ ίσο με τη μέγιστη αξία αντικειμένου που απέρριψε. Έπειτα, εξετάζει ένα-ένα τα υπόλοιπα αντικείμενα και επιλέγει το πρώτο που θα εμφανιστεί με αξία μεγαλύτερη από τ. Ουσιαστικά, ο αλγόριθμος "θυσιάζει" ένα μέρος της ακολουθίας εισόδου προκειμένου να "μάθει" μία καλή τιμή (όχι πολύ υψηλή ούτε πολύ χαμηλή) για το φράγμα τ. Είναι εύκολο να δείξουμε ότι αυτός ο απλός αλγόριθμος είναι και ο βέλτιστος δυνατός για το Secretary Problem.

Το πρόβλημα αυτό επιδέχεται διάφορες ενδιαφέρουσες γενικεύσεις. Η πιο απλή είναι το k-Secretary Problem, στο οποίο η μόνη διαφορά με προηγουμένως είναι ότι επιτρέπεται να διαλέξουμε μέχρι k διαφορετικά αντικείμενα για κάποιο γνωστό $k \geq 1$, και συγκρινόμαστε με το σύνολο των k αντικειμένων μέγιστης αξίας. Για το πρόβλημα αυτό, μία φυσιολογική επέκταση του παραπάνω αλγορίθμου αγνοεί τα πρώτα δn αντικείμενα με $\delta = O\left(\frac{\log k}{k^{1/3}}\right)$ και θέτει το φράγμα τ περίπου ίσο με το k-οστό υψηλότερο αντικείμενο. Αυτή η προσέγγιση περιγράφεται στον Αλγόριθμο 2.2, ο οποίος επιτυγχάνει λόγο ανταγωνιστικότητας $1 - O\left(\frac{\log k}{k^{1/3}}\right)$. Το αποτέλεσμα αυτό μπορεί να βελτιωθεί σημαντικά, εάν σκεφτούμε επιπλέον ότι δεν χρειάζεται να διατηρούμε το ίδιο φράγμα σε όλη την εκτέλεση του αλγορίθμου, αλλά μπορούμε να το προσαρμόζουμε δυναμικά καθώς βλέπουμε περισσότερα στοιχεία. Αυτή η ιδέα οδηγεί στον Αλγόριθμο 2.3 που εισήχθη στο [18]. Ο αλγόριθμος αυτός λειτουργεί σε φάσεις εκθετικά αυξανόμενου μήκους, σε καθεμία εκ των οποίων χρησιμοποιεί όλα τα αντικείμενα που έχει δει στις προηγούμενες φάσεις προκειμένου να ορίσει κατάλληλα το φράγμα τ βάσει του οποίου επιλέγει αντικείμενα. Ο λόγος ανταγωνιστικότητας του είναι $1 - O\left(\sqrt{\frac{\log k}{k}}\right)$. Τέλος, είναι εφικτό να πετύχουμε μία περαιτέρω βελτίωση μέσω αρκετά πιο σύνθετων ιδεών, χρησιμοποιώντας τον αναδρομικό Αλγόριθμο 2.4 του [19]. Αυτός επιτυγχάνει λόγο ανταγωνιστικότητας ίσο με $1 - O\left(\frac{1}{\sqrt{k}}\right)$, ο οποίος είναι και βέλτιστος.

Μία ιδιαίτερα μελετημένη επέκταση του Secretary Problem, η οποία γενικεύει σημαντικά τα προηγούμενα, είναι το Matroid Secretary Problem. Τα matroids είναι μαθηματικά αντικείμενα υψηλού θεωρητικού ενδιαφέροντος, καθώς γενικεύουν βασικές έννοιες σε διάφορα πεδία, όπως η Θεωρία Γραφημάτων, η Συνδυαστική Βελτιστοποίηση και η Γραμμική Άλγεβρα. Βασικοί ορισμοί για matroids παρατίθενται στο Παράρτημα (Definition A.0.1).

Στο Matroid Secretary Problem, τα αντικείμενα της εισόδου αποτελούν πλέον στοιχεία ενός matroid, και το σύνολο των επιλεγμένων αντικειμένων μας πρέπει ανά πάσα στιγμή να αποτελεί ανεξάρτητο σύνολο του matroid. Συγκρινόμαστε με την υψηλότερης αξίας βάση του matroid. Το πρόβλημα αυτό εξετάστηκε αρχικά στο [20], όπου δόθηκε ο Αλγόριθμος 2.5. Όπως και προηγουμένως, ο αλγόριθμος αυτός “θυσιάζει” ένα μέρος των στοιχείων ώστε να ορίσει κατάλληλα ένα φράγμα τ. Έπειτα, επιλέγει κάθε επόμενο στοιχείο με αξία τουλάχιστον τ, υπό την προϋπόθεση πως το σύνολο των επιλεγμένων στοιχείων παραμένει ανεξάρτητο. Ο λόγος ανταγωνιστικότητάς του είναι $O\left(\frac{1}{\log r}\right)$, όπου r είναι ο βαθμός του matroid. Σημαντικό είναι ότι ο αλγόριθμος λειτουργεί για οποιοδήποτε matroid. Μετά από την εισαγωγή του προβλήματος έχουν προταθεί διάφοροι άλλοι αλγόριθμοι με βελτιωμένη επίδοση, με τον καλύτερο γνωστό λόγο ανταγωνιστικότητας να είναι $O\left(\frac{1}{\log\log r}\right)$, όπως αποδεικνύεται στα [21], [22]. Αξιοσημείωτο είναι το γεγονός ότι το καλύτερο κάτω φράγμα που γνωρίζουμε για το πρόβλημα είναι απλώς το $\frac{1}{e}$ που προκύπτει από το Single-Secretary Problem.

Μία διαφορετική προσέγγιση για το Matroid Secretary Problem είναι να μην προσπαθήσουμε να αναπτύξουμε αλγορίθμους που λειτουργούν για οποιοδήποτε matroid, αλλά να εστιάσουμε σε συγκεκριμένες κατηγορίες matroids και να σχεδιάσουμε αλγορίθμους που εκμεταλλεύονται τις ιδιότητές τους. Για παράδειγμα, για matroids διαμέρισης, μπορούμε απλώς να τρέξουμε τον αλγόριθμο για το Single-Secretary Problem σε καθένα από τα τμήματα της διαμέρισης και να είμαστε $\frac{1}{e}$-ανταγωνιστικοί. Επιπλέον, στο [23] αποδεικνύεται πως η περίπτωση των γραφικών matroids ανάγεται στην περίπτωση των matroids διαμέρισης, οπότε προκύπτει ένας απλός $\frac{1}{2e}$-ανταγωνιστικός αλγόριθμος. Το τελευταίο αποτέλεσμα βελτιώνεται στο [24], όπου δίνεται ένας πιο σύνθετος $\frac{1}{4}$-ανταγωνιστικός αλγόριθμος για γραφικά matroids.

Πέραν των Secretary Problems, υπάρχουν και άλλα ενδιαφέροντα προβλήματα που έχουν μελετηθεί στο μοντέλο Τυχαίας Διάταξης. Εστιάζουμε σε δύο προβλήματα Άμεσου Ταιριάσματος. Το πρώτο πρόβλημα που μελετάμε είναι το Άμεσο Ταίραισμα Μεγίστου Βάρους. Σε αυτό το πρόβλημα δίνεται ένα διμερές γράφημα $G = (L, R; E)$ με n κορυφές στο αριστερό μέρος L και m στο δεξιό μέρος R. Οι ακμές έχουν μη-αρνητικά βάρη. Οι κορυφές του R δίνονται εξ αρχής, και οι κορυφές του L φανερώνονται σειριακά σε τυχαία διάταξη. Όταν φανερώνεται μία κορυφή, φανερώνονται και οι ακμές που προσπίπτουν σε αυτήν, και μπορούμε επιτόπου να επιλέξουμε μία εξ αυτών, υπό τον περιορισμό ότι οι ακμές που έχουμε επιλέξει πρέπει ανά πάσα στιγμή να αποτελούν ταίριασμα στο G (δηλαδή δεν μπορεί να υπάρχει κορυφή για την οποία έχουμε επιλέξει περισσότερες από μία προσπίπτουσες ακμές). Συγκρινόμαστε με το ταίριασμα μεγίστου βάρους του τελικού γραφήματος. Παρατηρούμε πως το πρόβλημα αυτό γενικεύει το Matroid Secretary για την περίπτωση των transversal matroids.

Για το Άμεσο Ταίριασμα Μεγίστου Βάρους, στο [25] μελετήθηκε ο Αλγόριθμος 2.6, ο οποίος επιτυγχάνει τον βέλτιστο λόγο ανταγωνιστικότητας $\frac{1}{e}$. Ο αλγόριθμος αυτός έχει εξαιρετικά απλή περιγραφή: αγνοεί τις πρώτες $\frac{n}{e}$ κορυφές και έπειτα, όταν φτάνει μία νέα κορυφή, υπολογίζει ένα ταίριασμα μεγίστου βάρους στο γράφημα που έχει φανερωθεί μέχρι στιγμής. Εάν η ακμή αυτού του ταιριάσματος που αντιστοιχεί στην νεοαφιχθείσα κορυφή μπορεί να προστεθεί στο σύνολο των ήδη επιλεγμένων ακμών, ώστε αυτό να εξακολουθήσει να είναι ταίριασμα, τότε επιλέγεται από τον αλγόριθμο.

Το δεύτερο πρόβλημα που μελετάμε σε αυτήν την κατηγορία είναι το Άμεσο Ταίριασμα Ελαχίστων Επαυξήσεων. Ομοίως με πριν, δίνεται ένα διμερές γράφημα $G = (L, R; E)$ με $|L| = |R| = n$. Οι κορυφές του R δίνονται εξ αρχής, και οι κορυφές του L εμφανίζονται σειριακά σε τυχαία διάταξη. Σε κάθε γύρο πρέπει να επιλέγουμε ένα μέγιστο ταίριασμα με τις κορυφές του L που έχουν ήδη εμφανιστεί, και μπορούμε να αναιρούμε επιλογές που κάναμε στο παρελθόν. Κάθε φορά που επιλέγουμε μία ακμή που δεν είχαμε επιλέξει στον αμέσως προηγούμενο γύρο, χρεωνόμαστε μία μονάδα κόστους, και επιθυμούμε να ελαχιστοποιήσουμε το συνολικό κόστος μας. Για το πρόβλημα αυτό, στο [26] δίνεται ο Αλγόριθμος 2.7 με αναμενόμενο πλήθος επαυξήσεων $O(n \log n)$, το οποίο αποδεικνύεται ότι είναι βέλτιστο.

0.3 Μοντέλα Προφήτη

Το απλούστερο και πλέον κλασσικό πρόβλημα που έχει αναλυθεί στο μοντέλο Προφήτη είναι ανάλογο του Secretary Problem, και ονομάζεται το πρόβλημα της Ανισότητας Προφήτη (Prophet Inequality), από το οποίο λαμβάνει το όνομά του και το ίδιο το μοντέλο. Ένας αντίπαλος επιλέγει n ανεξάρτητες κατανομές $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(n)}$ πάνω στους μη-αρνητικούς πραγματικούς αριθμούς. Σε κάθε γύρο i, εμφανίζεται ένα αντικείμενο του οποίου η αξία ακολουθεί την κατανομή $\mathcal{D}^{(i)}$, και εμείς μπορούμε είτε να το απορρίψουμε αμετάκλητα είτε να το επιλέξουμε. Μπορούμε να επιλέξουμε μόνο ένα αντικείμενο, και θέλουμε να μεγιστοποιήσουμε την αναμενόμενη αξία μας. Συγκρινόμαστε με την αναμενόμενη τιμή της μέγιστης αξίας $X_{\max}$ ανάμεσα σε όλα τα αντικείμενα. Θεωρούμε πως έχουμε πλήρη γνώση όλων των κατανομών για τον σχεδιασμό του αλγορίθμου. Αξιοποιώντας αυτή τη γνώση, μπορούμε και πάλι να ορίσουμε ένα κατάλληλο φράγμα και να επιλέξουμε το πρώτο αντικείμενο με αξία μεγαλύτερη του φράγματος. Ειδικότερα, μπορούμε να ορίσουμε το φράγμα ίσο είτε με τη διάμεσο τιμή του $X_{\max}$, είτε με το μισό της αναμενόμενης τιμής του $X_{\max}$. Και οι δύο επιλογές μας δίνουν λόγο ανταγωνιστικότητας ίσο με $\frac{1}{2}$, και είναι εύκολο να δούμε πως αυτός είναι ο βέλτιστος δυνατός.

Εάν δεν διαθέτουμε πλήρη γνώση των κατανομών, αλλά μας δίνεται πρόσβαση σε δείγματά τους, είναι πάλι εφικτό να επιτύχουμε τον βέλτιστο λόγο ανταγωνιστικότητας με ένα μόλις δείγμα από κάθε κατανομή, χρησιμοποιώντας τον απλό Αλγόριθμο 3.1 του [27]. Ο αλγόριθμος αυτός τραβάει αρχικά ένα δείγμα από κάθε κατανομή, θέτει ένα φράγμα τ ίσο με το μέγιστο δείγμα, και έπειτα επιλέγει το πρώτο αντικείμενο με αξία τ.

Όπως και στα Secretary Problems, η απλούστερη επέκταση του Single-Choice Prophet Inequality είναι το k-Choice Prophet Inequality, όπου η μόνη διαφορά με προηγουμένως είναι ότι επιτρέπεται να διαλέξουμε έως και k αντικείμενα και συγκρινόμαστε με την αναμενόμενη αξία των k υψηλότερων στοιχείων. Και σε αυτήν την περίπτωση, εάν έχουμε πλήρη γνώση των κατανομών, μπορούμε να ορίσουμε ένα κατάλληλο φράγμα τ με απλό τρόπο, προκειμένου να επιτύχουμε λόγο ανταγωνιστικότητας $1 - O\left(\sqrt{\frac{\log k}{k}}\right)$. Δύο τρόποι με τους οποίους μπορούμε να ορίσουμε το φράγμα αυτό φαίνονται στους Αλγορίθμους 3.2 και 3.3 των [28], [29] αντίστοιχα. Το αποτέλεσμα αυτό μπορεί να βελτιωθεί, καθώς στο [30] δίνεται ένας αρκετά πιο σύνθετος αλγόριθμος με λόγο ανταγωνιστικότητας $1-O\left(\frac{1}{\sqrt{k}}\right)$, που είναι το βέλτιστο δυνατό όπως αποδεικνύεται στο [28]. Μάλιστα, στο [31] δείχνεται, μεταξύ άλλων, πως αυτός ο βέλτιστος λόγος ανταγωνιστικότητας μπορεί και πάλι να επιτευχθεί εάν διαθέτουμε ένα μόλις δείγμα από κάθε κατανομή.

Συνεχίζοντας, μπορούμε κατ' αναλογία με προηγουμένως να μελετήσουμε το πρόβλημα Matroid Prophet Inequality, όπου και πάλι τα αντικείμενα είναι πλέον στοιχεία ενός matroid και οφείλουμε ανά πάσα στιγμή να έχουμε επιλέξει ένα ανεξάρτητο σύνολο του matroid. Το πρόβλημα αυτό μελετήθηκε αρχικά στο [32], όπου δόθηκε ο Αλγόριθμος 3.4, ο οποίος είναι $\frac{1}{2}$-ανταγωνιστικός αλγόριθμος για *οποιοδήποτε* matroid. Και αυτός ο αλγόριθμος βασίζεται σε ένα κατάλληλα ορισμένο φράγμα, το οποίο αυτή τη φορά προσαρμόζεται δυναμικά σε κάθε γύρο. Εφόσον το Matroid Prophet Inequality είναι γενίκευση του απλού Single-Choice Prophet Inequality, το κάτω φράγμα $\frac{1}{2}$ εξακολουθεί να ισχύει, άρα ο αλγόριθμος αυτός είναι βέλτιστος.

Πέραν των Prophet Inequality προβλημάτων, μπορούμε και εδώ να συζητήσουμε για προβλήματα ταιριάσματος στο μοντέλο Προφήτη, στα οποία υπάρχει ένα υποκείμενο γράφημα $G = (V, E)$ με μη-αρνητικά βάρη στις ακμές. Εδώ, μπορούμε να θεωρήσουμε πως, σε κάθε γύρο, είτε φανερώνεται μία από τις ακμές του γραφήματος (αφίξεις ακμών), είτε μία κορυφή με όλες τις προσκείμενες ακμές της (αφίξεις κορυφών). Και στις δύο περιπτώσεις, όταν φανερώνεται μία ακμή μπορούμε είτε να την απορρίψουμε αμετάκλητα είτε να την επιλέξουμε, και πρέπει ανά πάσα στιγμή οι επιλεγμένες ακμές μας να αποτελούν ταίριασμα στο G. Συγκρινόμαστε με την αναμενόμενη αξία του μέγιστου ταιριάσματος στο τελικό γράφημα. Στην περίπτωση που το γράφημά μας είναι διμερές και έχουμε αφίξεις κορυφών, ο Αλγόριθμος 3.5 του [30] είναι $\frac{1}{2}$-ανταγωνιστικός (και άρα βέλτιστος). Ομοίως με προηγουμένως, αυτός ο αλγόριθμος βασίζεται στον υπολογισμό κατάλληλου φράγματος τ_u, ενός για κάθε κορυφή $u \in R$, βάσει του οποίου αποφασίζεται αν η κορυφή u θα συνδεθεί με κάποιον γείτονά της όταν φανερωθεί (και αν ναι, με ποιον). Για γενικά γραφήματα, στο [33] δίνεται ένας σαφώς πιο σύνθετος $\frac{1}{2}$-ανταγωνιστικός αλγόριθμος για αφίξεις κορυφών, καθώς και ένας 0.337-ανταγωνιστικός αλγόριθμος για αφίξεις ακμών.

0.4 Επαυξήσιμα Ακέραια Προγράμματα

Ο ορισμός των *Επαυξήσιμων Ακέραιων Προγραμμάτων* (Augmentable Integer Programmes, AIPs) δόθηκε στο [1]. Έστω ένα ακέραιο γραμμικό πρόγραμμα με διάνυσμα μεταβλητών z, διάνυσμα κόστους c και σύνολο περιορισμών V. Για οποιαδήποτε υποσύνολα περιορισμών $W, V' \subseteq V$, εάν το z ικανοποιεί τους περιορισμούς του V', συμβολίζουμε με $\textsc{Aug}(W \mid z, V')$ το *κόστος επαύξησης* του z ώστε να ικανοποιεί το W, δηλαδή το ελάχιστο κόστος που μπορούμε να πληρώσουμε ώστε να αυξήσουμε τις μεταβλητές του z για να ικανοποιούν, πέραν του V', και τους περιορισμούς του W. Με αυτόν τον συμβολισμό, ένα *Επαυξήσιμο* Ακέραιο Πρόγραμμα χαρακτηρίζεται από την ιδιότητα των *μονότονων* κοστών επαύξησης, υπό την έννοια ότι για οποιαδήποτε $V' \subseteq V'' \subseteq V$ και $z' \leq z''$, εάν το z' ικανοποιεί το V' και το z'' το V'', τότε $\textsc{Aug}(W \mid z'', V'') \leq \textsc{Aug}(W \mid z', V')$ για κάθε $W \subseteq V$.

Ο λόγος που η κλάση των AIPs έχει ενδιαφέρον είναι, αρχικά, ότι εμπεριέχει την κλάση των Ακεραίων Προγραμμάτων Κάλυψης (Covering Integer Programmes, CIPs), εντός της οποίας βρίσκονται το θεμελιώδες προβλήμα του Καλύμματος Συνόλων (και οι υποπεριπτώσεις του, το Κάλυμμα Κορυφών και το Κάλυμμα Ακμών). Επιπλέον, όμως, η κλάση των AIPs εμπεριέχει και ενδιαφέροντα προβλήματα που δεν είναι CIPs, συγκεκριμένα την Χωροθέτηση Υπηρεσιών (Facility Location) και το Δέντρο Steiner (Steiner Tree). Ειδικότερα, μας ενδιαφέρουν αλγόριθμοι για τις άμεσες εκδοχές των προβλημάτων αυτών, στα μοντέλα Τυχαίας Διάταξης και Προφήτη που παρουσιάσαμε προηγουμένως.

Στο πρόβλημα του Άμεσου Καλύμματος Συνόλων, μας δίνεται ένα σύμπαν αντικειμένων, μία συλλογή m συνόλων αντικειμένων του σύμπαντος και ένα κόστος c_S για κάθε σύνολο S. Ένας αντίπαλος επιλέγει n αντικείμενα από το σύμπαν. Στο μοντέλο Τυχαίας Διάταξης, τα n αυτά στοιχεία παρουσιάζονται σε τυχαία σειρά. Μόλις φανερώνεται ένα στοιχείο, εάν δεν έχουμε ήδη επιλέξει ένα σύνολο που το καλύπτει, είμαστε υποχρεωμένοι να το κάνουμε επιτόπου, και χρεωνόμαστε το κόστος του συνόλου που επιλέξαμε. Συγκρινόμαστε με τη συλλογή συνόλων ελαχίστου κόστους που καλύπτει όλα τα n στοιχεία. Για αυτό το πρόβλημα, πρόσφατα δόθηκε στο [34] ένας $O(\log mn)$-ανταγωνιστικός αλγόριθμος, ο οποίος διατηρεί ένα είδος κατανομής πιθανότητας x πάνω στα σύνολα, η οποία μπορεί επίσης να ειδωθεί ως (μη εφικτή) κλασματική λύση για το πρόβλημα. Καθώς φανερώνεται ένα στοιχείο, ο αλγόριθμος δειγματοληπτεί σύνολα βάσει του x, ανανεώνει πολλαπλασιαστικά το βάρος x_S κάθε συνόλου S που περιέχει το αφιχθέν στοιχείο, και έπειτα καλύπτει αυτό το στοιχείο με το σύνολο ελαχίστου κόστους. Συμβολίζουμε με κ_{v^t} το σύνολο ελαχίστου κόστους με το οποίο μπορούμε να καλύψουμε το στοιχείο v^t. Επίσης, θεωρούμε πως γνωρίζουμε ένα φράγμα β τουλάχιστον ίσο με το κόστος της βέλτιστης λύσης και το πολύ ίσο με το διπλάσιο αυτού (αυτή η υπόθεση εύκολα άρεται με κόστος έναν επιπλέον παράγοντα 2 στον λόγο ανταγωνιστικότητας). Με αυτόν τον συμβολισμό, ο αλγόριθμος του [34] είναι ο εξής

Algorithm 0.1: LearnOrCover

```
1  Let $m' \leftarrow |\{S : c_S \leq \beta\}|$
2  Initialise $x_S^0 \leftarrow \frac{\beta}{c_S \cdot m'}$ for all sets $S$
3  for $t \in [n]$ do
4      $v^t \leftarrow t^{th}$ element in random order
5      if $v^t$ uncovered then
6          For each set $S$, pick $S$ with probability $\min(\kappa_{v^t} \cdot x_S^{t-1}/\beta, 1)$
7          if $\sum_{S \ni v^t} x_S^{t-1} < 1$ then
8              For every set $S$, update $x_S^t \leftarrow x_S^{t-1} \cdot \exp\{\mathbf{1}_{\{S \ni v^t\}} \cdot \kappa_{v^t}/c_S\}$
9              Let $Z^t = \langle c, x^t \rangle/\beta$ and normalise $x^t \leftarrow x^t/Z^t$
10         else
11             $x^t \leftarrow x^{t-1}$
12         Pick the cheapest set containing $v^t$
```

Διαισθητικά, ο αλγόριθμος αυτός λειτουργεί γιατί εξασφαλίζει πως, σε κάθε γύρο, είτε θα καλύψει μεγάλο ποσοστό στοιχείων που δεν έχουν φανερωθεί ακόμη, είτε το διάνυσμα x που διατηρεί θα "πλησιάσει" πολύ προς την βέλτιστη κλασματική λύση κόστους β για το πρόβλημα (δηλαδή ο αλγόριθμος κάνει πρόοδο είτε στη "μάθηση" είτε στην "κάλυψη"). Αξίζει να σημειωθεί πως, για το ίδιο πρόβλημα στο πλήρως ανταγωνιστικό μοντέλο, στο [2] δόθηκε ένας $O(\log m \log n)$-ανταγωνιστικός αλγόριθμος, ο οποίος δείχθηκε στο [35] ότι είναι βέλτιστος. Επομένως, αναλύοντας το πρόβλημα στο μοντέλο Τυχαίας Διάταξης, αποδυναμώνουμε επαρκώς τον αντίπαλο ώστε να επιτύχουμε σημαντικά βελτιωμένο λόγο ανταγωνιστικότητας, καλύτερο από τον βέλτιστο εφικτό στο πλήρως ανταγωνιστικό μοντέλο.

Το επόμενο AIP που μελετάμε είναι το πρόβλημα της Άμεσης Χωροθέτησης Υπηρεσιών. Σε αυτό το πρόβλημα, μας δίνεται ένα σύνολο πιθανών υπηρεσιών, καθεμία από τις οποίες έχει ένα κόστος ανοίγματος c_f. Ένας αντίπαλος επιλέγει n πελάτες, καθένας από τους οποίους μπορεί να συνδεθεί με την υπηρεσία f με κόστος c_{fv}. Στο μοντέλο Τυχαίας Διάταξης, οι πελάτες εμφανίζονται με τυχαία σειρά. Μόλις εμφανιστεί ένας πελάτης, μπορούμε να ανοίξουμε οποιεσδήποτε υπηρεσίες (πληρώνοντας το κόστος ανοίγματος) και οφείλουμε να συνδέσουμε τον πελάτη σε μία ανοικτή υπηρεσία (πληρώνοντας το κόστος σύνδεσης). Ίσως η περισσότερο μελετημένη εκδοχή του προβλήματος είναι η Άμεση *Μετρική* Χωροθέτηση Υπηρεσιών, στην οποία οι υπηρεσίες και οι πελάτες αντιστοιχούν σε σημεία σε έναν μετρικό χώρο (όπου οι αποστάσεις ικανοποιούν την τριγωνική ανισότητα), και το κόστος c_{fv} ισούται με την απόσταση της υπηρεσίας f και του πελάτη v.

Το πρόβλημα της Άμεσης Μετρικής Χωροθέτησης Υπηρεσιών στο μοντέλο Τυχαίας Διάταξης μελετήθηκε αρχικά στο [5], όπου δόθηκε ο απλός Αλγόριθμος 4.3, ο οποίος είναι 8-ανταγωνιστικός για την περίπτωση όπου όλες οι υπηρεσίες έχουν το ίδιο κόστος ανοίγματος. Οι ιδέες του αλγορίθμου επεκτείνονται και στην περίπτωση όπου έχουμε διαφορετικά κόστη ανοίγματος, οπότε λαμβάνουμε έναν 33-ανταγωνιστικό αλγόριθμο. Αξίζει να σημειωθεί πως ο αλγόριθμος αυτός επιτυγχάνει λόγο ανταγωνιστικότητας $O\left(\frac{\log n}{\log\log n}\right)$ στο πλήρως ανταγωνιστικό μονέλο, ο οποίος αποδεικνύεται ότι είναι βέλτιστος στο [6]. Επίσης αναφέρουμε πως, για το μοντέλο Τυχαίας Άφιξης, ο παραπάνω αλγόριθμος του [5] δεν είναι βέλτιστος, καθώς στο [36] δίνεται ένας 3-ανταγωνιστικός αλγόριθμος και δίνεται ένα κάτω φράγμα ίσο με 2.

Για το πρόβλημα της Άμεσης Μη-Μετρικής Χωροθέτησης Υπηρεσιών, πλέον οι υπηρεσίες και οι πελάτες δεν αποτελούν σημεία σε έναν μετρικό χώρο, επομένως τα κόστη σύνδεσης c_{fv} μπορούν να είναι αυθαίρετα. Εύκολα βλέπουμε πως, στην περίπτωση που όλα τα κόστη σύνδεσης είναι 0 ή ∞, ανακτούμε το Άμεσο Κάλυμμα Συνόλων ως ειδική περίπτωση, επομένως δεν μπορούμε να έχουμε λόγο ανταγωνιστικότητας καλύτερο από το κάτω φράγμα $O(\log m \log n)$ που αναφέραμε παραπάνω για αυτό το πρόβλημα στο πλήρως ανταγωνιστικό μοντέλο. Πράγματι, στο [37] δίνεται ένας αλγόριθμος που επιτυγχάνει αυτόν ακριβώς τον λόγο ανταγωνιστικότητας. Πρόσφατα, στο [1] μελετήθηκε το ίδιο πρόβλημα στο μοντέλο Τυχαίας Διάταξης. Εκεί δόθηκε ο Αλγόριθμος 4.4, ο οποίος επεκτείνει τις ιδέες του Αλγορίθμου 0.1, αντιμετωπίζοντας τις υπηρεσίες ως “σύνολα” και τους πελάτες ως “στοιχεία” που ανήκουν σε αυτά. Η βασική διαφορά είναι πως το σύνολο πελατών που “καλύπτει” κάθε υπηρεσεία δεν είναι σταθερό με τον χρόνο (όπως στο Κάλυμμα Συνόλων), αλλά μεταβάλλεται σε κάθε γύρο, επομένως προστίθεται επιπλέον πολυπλοκότητα στον αλγόριθμο και στην ανάλυσή του. Ο λόγος ανταγωνιστικότητας του αλγορίθμου είναι $O(\log mn)$, οπότε ομοίως με προηγουμένως έχουμε σημαντική βελτίωση συγκριτικά με το πλήρως ανταγωνιστικό μοντέλο.

Μία σημαντική ιδιότητα της οικογένειας των Άμεσων Επαυξήσιμων Ακεραίων Προγραμμάτων είναι ότι, εάν για κάποιο πρόβλημα της οικογένειας διαθέτουμε έναν αλγόριθμο $\mathcal{A}$ για το μοντέλο Τυχαίας Διάταξης, μπορούμε να επεκτείνουμε με πολύ απλό τρόπο τον αλγόριθμο αυτόν για να αντιμετωπίσουμε το ίδιο πρόβλημα στο μοντέλο Προφήτη. Συγκεκριμένα, εάν τα n αντικείμενα της εισόδου (π.χ. στοιχεία στο Κάλυμμα Συνόλου, πελάτες στη Χωροθέτηση Υπηρεσιών) προκύπτουν από τις ανεξάρτητες κατανομές $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(n)}$, τότε μπορούμε να αντλήσουμε ένα μόλις δείγμα από κάθε κατανομή και να εκτελέσουμε τον $\mathcal{A}$ με είσοδο μία ομοιόμορφα τυχαία διάταξη αυτών των δειγμάτων. Έπειτα, μπορούμε να “αγοράσουμε” όσα σύνολα/υπηρεσίες αγόρασε και ο $\mathcal{A}$ *πριν* αρχίσουμε να βλέπουμε την πραγματική ακολουθία εισόδου. Τέλος, όταν εμφανίζεται ένα αντικείμενο της πραγματικής ακολουθίας, εάν δεν το έχουμε ήδη καλύψει, τότε το καλύπτουμε επιτόπου με το μικρότερο δυνατό κόστος (επιλέγοντας το κατάλληλο σύνολο/υπηρεσία). Στο [1] αποδεικνύεται πως, με αυτήν την απλή προσέγγιση, εάν για κάποιο Άμεσο Επαυξήσιμο Ακέραιο Πρόγραμμα έχουμε έναν Δ-ανταγωνιστικό αλγόριθμο $\mathcal{A}$ στο μοντέλο Τυχαίας Διάταξης, τότε δημιουργύμε έναν 2Δ-ανταγωνιστικό αλγόριθμο για το μοντέλο Προφήτη. Επομένως, βάσει όσων αναφέραμε προηγουμένως, προκύπτουν απευθείας $O(\log mn)$-ανταγωνιστικοί αλγόριθμοι στο μοντέλο Προφήτη για το Άμεσο Κάλυμμα Συνόλων και την Άμεση Μη-Μετρική Χωροθέτηση Υπηρεσιών, καθώς και ένας 6-ανταγωνιστικός αλγόριθμος για την Άμεση Μετρική Χωροθέτηση Υπηρεσιών.

Το τελευταίο πρόβλημα της οικογένειας των Άμεσων Επαυξήσιμων Ακεραίων Προγραμμάτων που μελετάμε είναι το Άμεσο Δέντρο Steiner. Στο πρόβλημα αυτό μας δίνεται ένας μετρικός χώρος k σημείων, ο οποίος μπορεί να ειδωθεί ως ένα πλήρες γράφημα k κορυφών. Στο μοντέλο Τυχαίας Διάταξης, ένας αντίπαλος επιλέγει ένα υποσύνολο n κορυφών, οι οποίες εμφανίζονται σε τυχαία σειρά. Στο μοντέλο Προφήτη, ένας αντίπαλος επιλέγει n κατανομές πάνω στις κορυφές του γραφήματος, και σε κάθε γύρο η κορυφή που εμφανίζεται προκύπτει από την αντίστοιχη κατανομή. Και στις δύο περιπτώσεις, μόλις εμφανιστεί μία κορυφή (από τη δεύτερη και έπειτα) είμαστε υποχρεωμένοι να αγοράσουμε ακμές προκειμένου να τη συνδέσουμε με μία από τις κορυφές που έχουν εμφανιστεί προηγουμένως, και ο στόχος μας είναι να ελαχιστοποιήσουμε το συνολικό κόστος μας. Σημειώνουμε πως, εάν τα βάρη του γραφήματός μας δεν ικανοποιούν την τριγωνική ανισότητα, μπορούμε απλώς να εργαστούμε με την μετρική κλειστότητα του γραφήματος, άρα δεν υπάρχει λόγος να μελετήσουμε τη μη-μετρική περίπτωση του Δέντρου Steiner.

Στο πλήρως ανταγωνιστικό μοντέλο για το Άμεσο Δέντρο Steiner, είναι γνωστό ότι ο προφανής άπληστος αλγόριθμος που συνδέει κάθε κορυφή με τον φτηνότερο δυνατό τρόπο είναι $O(\log n)$ ανταγωνιστικός, και αυτό είναι βέλτιστο. Η ενδιαφέρουσα διαφορά του Άμεσου Δέντρου Steiner σε σχέση με τα προηγούμενα προβλήματα που μελετήσαμε είναι ότι, επιβάλλοντας Τυχαία Διάταξη, *δεν* αποδυναμώνουμε καθόλου τον αντίπαλο και παραμένει το ίδιο κάτω φράγμα $O(\log n)$ στον λόγο ανταγωνιστικότητας. Το θεώρημα αυτό αποδείχθηκε στο [38], μέσω ενός αντιπαραδείγματος που βασίζεται στην κατασκευή των λεγόμενων diamond graphs. Η κατασκευή αυτή ξεκινάει από το αντίστοιχο αντιπαράδειγμα για το πλήρως ανταγωνιστικό μοντέλο του προβλήματος και δημιουργεί πολλά αντίτυπα κάθε κορυφής στην ακολουθία εισόδου. Εφόσον, για κάθε κορυφή, μόνο η πρώτη εμφάνισή της στην ακολουθία εισόδου έχει σημασία, διαισθητικά, αν ανακατέψουμε τυχαία την προκύπτουσα ακολουθία και διατηρήσουμε μόνο το πρώτο αντίτυπο κάθε κορυφής, δεν αναμένουμε το αποτέλεσμα να αποκλίνει σημαντικά από το αρχικό μας αντιπαράδειγμα. Αυτή η παρατήρηση σε συνδυασμό με μία κρίσιμη ιδιότητα αυτού του αντιπαραδείγματος καθιστά την κατασκευή εφική. Από την άλλη πλευρά, στο μοντέλο Προφήτη μπορούμε με απλό τρόπο να σχεδιάσουμε έναν 4-ανταγωνιστικό αλγόριθμο, όπως αποδεικνύεται ξανά στο [38]. Αρχικά, τραβάμε ένα δείγμα από κάθε κατανομή, και αγοράζουμε το Ελάχιστο Συνδετικό Δέντρο πάνω στα δείγματα που τραβήξαμε, *πριν* δούμε την πραγματική ακολουθία εισόδου. Έπειτα, όταν εμφανίζεται μία κορυφή της πραγματικής ακολουθίας, τη συνδέουμε με τις ακμές που έχουμε ήδη αγοράσει με τον φτηνότερο δυνατό τρόπο. Επομένως, το Άμεσο Δέντρο Steiner είναι ένα ιδιαίτερο παράδειγμα ενός AIP όπου η διαφορά στη δυσκολία ανάμεσα στο μοντέλο Τυχαίας Διάταξης και στο μοντέλο Προφήτη είναι δραματική.

Κείμενο στα Αγγλικά

Chapter 1

Introduction

The challenge of making decisions with imperfect information is one that naturally arises in a number of contexts. For instance, consider the scenario where an individual wishes to sell an item at the highest possible price, through an online platform. That person might receive offers from various potential sellers and, upon receiving one such offer, must either accept or reject it. Both of those options entail risk, due to the fundamental issue of uncertainty about the future; the seller might reject an offer without knowing it is the best one they will ever receive, or they might accept an offer only to find out they would have received a much better one the next day.

The field of online algorithms attempts to capture precisely this challenge of decision-making under uncertainty. In this setting, the input to an online algorithm is not fully revealed at once, but is given incrementally. Concretely, in a given problem, the input might consist of a sequence of items $I = (x_1, x_2, \ldots, x_n)$ revealed over n rounds to the algorithm, where item x_i is revealed on the i-th round. Upon receiving x_i, the algorithm must immediately make some irrevocable decision to satisfy the constraints of the problem, without having knowledge of the future, i.e. of the items $x_{i+1}, \ldots, x_n$.

In order to measure the performance of an online algorithm in this setting, the concept of *competitive analysis* was introduced in [39]. Given an online algorithm $\mathcal{A}$, we compare it with the optimal *offline* algorithm which knows the entire input sequence in advance. Thus, for an optimisation problem and an input sequence I to that problem, we define $\mathcal{A}(I)$ to be the value of the solution of $\mathcal{A}$ when given I as input and $\textsc{Opt}(I)$ to be the value of the optimal offline solution. In practically all problems of interest, we cannot hope to develop an online algorithm that matches the performance of the offline optimal. An online algorithm $\mathcal{A}$ is called α-*competitive* for a problem if, for any input sequence I to that problem we have

$$\begin{cases} \mathcal{A}(I) \geq \alpha \cdot \textsc{Opt}(I) - \beta & \text{for a maximization problem} \\ \mathcal{A}(I) \leq \alpha \cdot \textsc{Opt}(I) + \beta & \text{for a minimization problem} \end{cases}$$

where α is called the *competitive ratio* and $\beta \geq 0$ is a constant. In the case where $\beta = 0$, the algorithm $\mathcal{A}$ is called *strictly* α-competitive.

This field of research has grown significantly over the past few decades and a substantial volume of interesting results has been produced in it. Many well-known (offline) combinatorial optimisation problems have been analysed in the online setting with great success, including Set Cover ([2]–[4]), Facility Location ([5]–[8]), Steiner Tree ([9]) and Bipartite Matching ([40]–[42]) among others. On the other hand, a large number of inherently online problems have been proposed and have attracted considerable interest, for example

Online Paging ([43]), the k-Server Problem ([10]–[13]), Metrical Task Systems ([14]–[16]), Bipartite Matching Maintenance ([44]–[46]) and Multistage Matroid Maintenance ([47]). This rapid growth of the field of online algorithms can likely be attributed, in part, to the fact that many of the problems studied in the area are not only highly intriguing from a theoretical point of view, but also find applications in various real-life contexts.

One of the major limitations, however, of the study of online algorithms is that, for the most part, it is concerned with the *worst-case* analysis for each algorithm. That is, by the definition of competitive ratio given above, an algorithm is α-competitive only if it is at most α times worse than the offline optimal *for any possible input sequence*. Of course, worst-case analysis is undoubtedly useful; not only has it led to a plethora of interesting results and a deep understanding of many problems, but moreover, establishing good worst-case guarantees for an algorithm certifies, in a strong sense, its utility and its robustness against any type of input. Nonetheless, it is often the case that worst-case analysis can be too pessimistic about the performance of an algorithm, and might not accurately express the algorithm's usefulness in the real world. In other words, it may be the case that, although an algorithm performs adequately well on all "typical" inputs of interest encountered in practice, it fails in some very specific or artificially created inputs, thus its theoretical worst-case guarantees do not reflect its actual applicability.

Due to this limitation of worst-case analysis, over the years researchers have considered a number of different models of algorithm analysis. In broad terms, those models do not consider the performance of an algorithm under all possible inputs generated by an adversary, but rather restrict the adversary's power in some way. For example, they might constrain the space of possible inputs from which the adversary can choose, or introduce randomness to the input which cannot be controlled by the adversary. The benefit of using such a model is that, ideally, it can encompass the prior knowledge we may have about the problem instances we are interested in, thus we can develop practically useful algorithms and provide theoretical guarantees for them, outside of the limiting scope of worst-case analysis. This study of going "beyond the worst-case analysis" of algorithms has been gaining in popularity in the past few years, and a recent survey of various approaches that have been used can be found in [48].

In this thesis, we will concern ourselves with two of the more popular models for going "beyond the worst-case" in online algorithms; namely, *random-order* models and *prophet* models. Our ultimate goal is to examine recent developments in applying those models to a class of problems named *Augmentable Integer Programmes* (AIPs), formally defined in [1]. First, we will give basic definitions for the above models and for the class of AIPs.

1.1 Preliminaries and Definitions

1.1.1 Notation

In the remainder of the thesis, we denote by by $\mathbb{Z}_{\geq 0}, \mathbb{R}_{\geq 0}$ the sets of nonnegative integer and real numbers respectively. We denote by $[n] = \{1, \dots, n\}$ the set of positive integer less than or equal to n. The probability of an event A is denoted $\mathbb{P}[A]$, and the expectation of a random variable X is denoted $\mathbb{E}[X]$. Given two vectors a, b of n elements, the regular dot product is denoted $\langle a, b \rangle := \sum_{i=1}^{n} a_i b_i$. All logarithms are base e.

1.1.2 Random-Order Models

Consider, again, an arbitrary online problem in which the input consists of a sequence of items $I = (x_1, x_2, \ldots, x_n)$ revealed over n rounds to the algorithm, where the items x_i belong in some domain $\mathcal{X}$. In classical worst-case analysis, the adversary chooses both which n items from $\mathcal{X}$ will be presented to the algorithm and the order in which they will appear. In contrast, in a *random-order* model, the adversary again chooses the set of n items $\{x_1, \ldots, x_n\}$, $x_i \in \mathcal{X}$, that will be presented to the algorithm, but the actual order in which they appear is uniformly random over all possible permutations of the set $\{x_1, \ldots, x_n\}$ (that is, each possible ordering of the n items is equally likely). In this setting, an online algorithm $\mathcal{A}$ is called *α-competitive in the random-order model* if, for any input set $I = \{x_1, \ldots, x_n\}$ we have

$$\begin{cases} \mathbb{E}[\mathcal{A}(I^\sigma)] \geq \alpha \cdot \textsc{Opt}(I) - \beta & \text{for a maximization problem} \\ \mathbb{E}[\mathcal{A}(I^\sigma)] \leq \alpha \cdot \textsc{Opt}(I) + \beta & \text{for a minimization problem} \end{cases}$$

Above, by I^σ we denote the sequence produced by ordering I according to permutation $\sigma : [n] \to [n]$. The expectation is taken over all (equiprobable) permutations σ and, if $\mathcal{A}$ is randomised, over the internal randomness of $\mathcal{A}$. As before, α is called the *competitive ratio* and $\beta \geq 0$ is a constant.

The random-order model was introduced in [49], where it was applied to the Online Bin Packing problem. In that paper, it was shown that the Best-Fit algorithm of [50] achieves a better competitive ratio in the random-order model than in the worst-case model. Indeed, the competitive ratio that Best-Fit achieves in the random-order model is better than any competitive ratio achievable for this problem in the worst-case model. This observation holds in many problems which have been successfully analysed in the random-order model (as we shall examine in Chapter 2); indeed, imposing a random order of arrival is often enough to significantly reduce the power of the adversary and can lead to theoretical guarantees that not only are better than existing worst-case ones, but may also break worst-case lower bounds.

Perhaps the most well-known problem (and indeed one of the oldest ones) studied in the random-order model is actually a classical problem from optimal stopping theory; the so-called Secretary Problem. In this problem, the input consists of a set of n items, each with a value $x_i \in \mathbb{R}_{\geq 0}$. The items are presented one-by-one to the algorithm and, upon seeing an item, the algorithm must immediately decide to either keep the item or forever discard it. The algorithm can only choose to keep one item, and the goal is to maximise the value of the item kept. It is easy to see that, in the worst-case model where both the item values and their order is chosen by an adversary, no algorithm can perform particularly well. However, in the random-order model, no matter what values are chosen from the adversary, a simple algorithm guarantees that, with probability at least $\frac{1}{e}$, we will pick the maximum value among all items. We will use the Secretary Problem as our first stepping stone in order to survey various important results in random-order models in Chapter 2.

1.1.3 Prophet Models

As before, consider an arbitrary online problem in which the input consists of a sequence of items $I = (x_1, x_2, \ldots, x_n)$ revealed over n rounds to the algorithm, where the items x_i belong in some domain $\mathcal{X}$. In the *prophet* setting, the adversary chooses a collection of n distributions over $\mathcal{X}$ and an ordering of those distributions. Let this ordering be $(\mathcal{D}^1, \ldots, \mathcal{D}^n)$. Afterwards, on round i, an item is drawn from distribution $\mathcal{D}^i$ – independently of all others

– and is presented to the algorithm. That is, the input sequence is generated according to $(x_1, \dots, x_n) \sim \mathcal{D}^1 \times \dots \times \mathcal{D}^n$.

In this setting, we consider that our algorithm is given some information about the underlying distributions $\mathcal{D}^1, \dots, \mathcal{D}^n$ *before* the input items start arriving; for instance, we might be given each distribution in closed form, or – more realistically – we may be granted sample access to the distributions (in which case it is desirable to draw as few samples from each distribution as possible). An online algorithm $\mathcal{A}$ is called *α-competitive in the prophet model* if, for any set of n distributions over $\mathcal{X}$ and any ordering $(\mathcal{D}^1, \dots, \mathcal{D}^n)$ of those distributions we have

$$\begin{cases} \mathbb{E}[\mathcal{A}(I)] \geq \alpha \cdot \mathbb{E}[\textsc{Opt}(I)] - \beta & \text{for a maximization problem} \\ \mathbb{E}[\mathcal{A}(I)] \leq \alpha \cdot \mathbb{E}[\textsc{Opt}(I)] + \beta & \text{for a minimization problem} \end{cases}$$

Here, both expectations are over the random input sequence $I \sim \mathcal{D}^1 \times \dots \times \mathcal{D}^n$. If $\mathcal{A}$ is randomised, the first expectation is also over the internal randomness of $\mathcal{A}$. Notice that the quality against which we are compared, i.e. the expected optimal value $\mathbb{E}[\textsc{Opt}(I)]$, is *independent* of the actually realised instance I; what we want to achieve in this setting is to be on expectation at most α times worse than the expected offline optimum, even if there exist specific realisations I on which we may be significantly worse than the realised offline optimum.

Arguably the most well-known and fundamental problem in the prophet setting is similar in concept to the Secretary Problem mentioned above. An adversary chooses an ordering of n distributions $(\mathcal{D}^1, \dots, \mathcal{D}^n)$, each over the set $\mathbb{R}_{\geq 0}$.We are given full knowledge of each distribution. An input sequence $(x_1, \dots, x_n) \sim \mathcal{D}^1 \times \dots \times \mathcal{D}^n$ is drawn, and each value is presented to the algorithm in order. Again, upon seeing an value, the algorithm must either keep it or forever discard it, and we can only pick a single value. Here, a simple thresholding rule guarantees that we will, on expectation, obtain a value equal to at least $\frac{1}{2}$ of the expected offline optimal. This has come to be known as the "Prophet Inequality" problem, and a historical survey of related problems can be found in [51]. In Chapter 3, we will start from this simple problem to examine various other problems in the prophet setting.

It is worth noting that prophet models have attracted significant attention in recent years, in part due to the application of prophet inequalities in mechanism design. Indeed, the simple thresholding rule suggested by the solution to the classical Prophet Inequality problem indicates a pricing scheme for the following problem: we wish to sell an item at the highest possible price, and there are n potential buyers, each with a private valuation x_i drawn from some distribution $\mathcal{D}^i$. We wish to set a single price for this item, so that the buyers can arrive one at a time in order, and upon arrival choose whether to buy the item or not (based on if their valuation for the item is higher than our price). This simple posted-price mechanism offers a number of advantages, as explained in the seminal paper [28] that established the connection between prophet inequalities and mechanism design.

1.1.4 Covering Integer Programmes

Consider a linear programme in normal form:

$$\begin{aligned}
\min \quad & \sum_{i=1}^{m} c_i x_i & \\
\text{s.t.} \quad & \sum_{i=1}^{m} a_{ji} x_i \geq b_j & \forall j \in [n] \\
& x_i \geq 0 & \forall i \in [m]
\end{aligned}$$

If all coefficients of this LP are nonnegative, i.e. $a_{ji}, b_j, c_i \geq 0$ for all $i \in [n]$ and $j \in [m]$, then the above is called a *covering linear programme*. If additionally the variables x_i are only allowed to take values in some subset of $\mathbb{Z}_{\geq 0}$, then we have a *covering integer programme* (CIP).

Arguably the most prominent combinatorial opmtimisation problem that can be viewed as a CIP is the (weighted) Set Cover problem (and its special cases, i.e. Vertex Cover and Edge Cover). Indeed, given an instance of Set Cover with m sets $\{S_1, \ldots, S_m\}$, n items $\{1, \ldots, n\}$ and a vector $c = (c_1, \ldots, c_m)$ of set costs, the natural integer programme formulation is the following:

$$\begin{aligned}
\min \quad & \sum_{i=1}^{m} c_i x_i & \\
\text{s.t.} \quad & \sum_{S_i \ni j} x_i \geq 1 & \forall j \in [n] \\
& x_i \in \{0, 1\} & \forall i \in [m]
\end{aligned}$$

Here, each coefficient a_{ji} of the general CIP formulation is equal to $a_{ji} = \mathbf{1}_{\{j \in S_i\}}$.

In the online setting, the algorithm is not given the n items of the Set Cover instance up front, but they are revealed one at a time over n rounds. Whenever an item arrives, if it is not already covered by the sets already picked by the algorithm, we must pick at least one new set to cover it. In the more general case of a CIP, the n covering constraints are presented over n rounds and, whenever a constraint arrives, if it is not satisfied by the current assignment of variables maintained by the algorithm, we must increment some variables in order to satisfy it (note that variables cannot be decreased at any point). In both cases, the goal is to minimise the total cost incurred, and we are compared against an offline optimum that sees all items/constrains in advance.

1.1.5 Augmentable Integer Programmes

Consider an instance of Set Cover, and let $\mathcal{S}', \mathcal{S}'' \subseteq \{S_1, \ldots, S_m\}$ be two collections of sets, such that $\mathcal{S}' \subseteq \mathcal{S}''$. Suppose that we are given either $\mathcal{S}'$ or $\mathcal{S}''$, alongside a collection of elements $W \subseteq [n]$, each of which may or may not be covered by the collection we were given. We wish to pick additional sets (that is, to *augment* the collection of sets we were given) in the least expensive way, so as to cover all elements in W. A crucial property of Set Cover is that, since $\mathcal{S}' \subseteq \mathcal{S}''$, the minimum cost of augmenting $\mathcal{S}''$ to cover the elements of W will never be larger than the corresponding minimum cost of augmenting $\mathcal{S}'$. That is to say, the marginal cost of augmenting a solution of Set Cover so that it covers additional elements is *monotonically non-increasing*.

This property of Set Cover and CIPs in general does not fully define them, as there are problems which satisfy this monotonicity property but are not covering problems. In order to study such problems, the class of *Augmentable Integer Programmes* (AIPs) was defined in [1]. Formally, consider an integer programme defined by a vector of variables $z \in \mathbb{Z}^m$, a cost vector $c \in \mathbb{R}^m$ and a set of constrains V (each of the form $\langle a_j, z \rangle \geq b_j$). For any subset $V' \subseteq V$ of constraints, let $\textsc{Sols}(V') \subseteq \mathbb{Z}^m$ be the subset of solutions that are feasible to V' (that is, the set of values of z that satisfy all constraints in V'). Also, for any subset of constraints $V' \subseteq V$, any solution $z \in \textsc{Sols}(V')$ and any other subset of constrains $W \subseteq V$, define the *augmentation cost*

$$\textsc{Aug}(W \mid z, V') \coloneqq \min_{w} \{ \langle c, w \rangle : w + z \in \textsc{Sols}(V' \cup W) \}$$

or ∞ if no such w exists. In words, this is the minimum cost of augmenting z (which already satisfies the constraints of V') so that it also satisfies the constraints of W. Using this definition, the class of AIPs is defined as follows:

Definition 1.1.1 (AIPs). An *augmentable* integer programme (AIP) is one in which the augmentation costs are *monotone*, i.e. for any $V' \subseteq V'' \subseteq V$ and any $z' \leq z''$ such that $z' \in \textsc{Sols}(V')$ and $z'' \in \textsc{Sols}(V'')$, we have $\textsc{Aug}(W \mid z'', V'') \leq \textsc{Aug}(W \mid z', V')$ for any constraint set $W \subseteq V$.

As we mentioned, all CIPs satisfy this monotonicity property, thus the class of AIPs is a generalisation of CIPs. The fundamental reason we are concerned with this class is because it contains three major combinatorial optimisation problems of interest, namely the Set Cover, Facility Location and Steiner Tree problems. Particularly in the former two problems, recent advances in the random-order and prophet models ([1], [34]) have introduced novel analysis techniques and have paved the way for intriguing new research directions. In Chapter 4 we will delve deeper into the field of random-order and prophet models for AIPs.

Chapter 2

Random-Order Models

2.1 Secretary Problems

2.1.1 Single-Secretary Problem

As we stated in Chapter 1, the (Single-)Secretary Problem can be described as follows. An adversary selects n nonnegative values $v_1, \dots, v_n \in \mathbb{R}_{\geq 0}$, and they are presented to our algorithm one by one in a random order. Whenever an element arrives, our algorithm can either pick it or forever discard it, and we can only pick one item in total. If $v_{\max}$ is the maximum among all presented elements, we wish to minimise our competitive ratio, that is, the ratio of $v_{\max}$ over our expected value. Tracing where the problem was initially stated and who initially proposed this algorithm is not as simple as it sounds, see [17] for a historical survey.

Before we present a solution to this problem, we will first examine *why* we need random order – in other words, why it is impossible to be competitive in the fully adversarial setting. If we use a deterministic algorithm, it is easy to see how our value can be arbitrarily smaller than $x_{\max}$, even for $n = 2$. Consider two input sequences, $(1, M)$ and $(1, \frac{1}{M})$, where $M \gg 1$. Since our algorithm is deterministic, upon seeing the value $x_1 = 1$, it must always decide to either pick it or discard it. If the algorithm picks x_1, it attains a competitive ratio of $\frac{1}{M}$ in the first sequence, while if it discards x_1 it attains the same competitive ratio in the second sequence. Thus, for any deterministic algorithm, there exists an input sequence on which it suffers an arbitrarily bad competitive ratio $\frac{1}{M}$ for any M.

Using a randomised algorithm helps only a little; we cannot do better than selecting a uniformly random position $i \in [n]$ a priori and picking the i-th value when it arrives. In this strategy, since we pick each value with probability $\frac{1}{n}$, our expected value is $\sum_{i \in [n]} \frac{x_i}{n} \geq \frac{x_{\max}}{n}$, so our competitive ratio is $\frac{1}{n}$. We can show that this is the best any randomised algorithm can do. Indeed, one strategy the adversary can use is to pick a large $M \gg 1$, uniformly at random pick an index $j \in [n]$, and present the sequence $(1, M, M^2, \dots, M^{j-1}, 0, \dots, 0)$. In order for any algorithm to be competitive, it must pick the last element in the ascending chain upon picking it with good probability, but this is tantamount to guessing j. Indeed, the following lemma can be proven by using Yao's Principle.

Lemma 2.1.1. *No randomised algorithm can achieve a competitive ratio better than $\frac{1}{n}$ for the Single-Secretary Problem in the fully adversarial setting.*

The above discussion sheds some light into the reason why the Secretary Problem is hard in the fully adversarial setting. Since the presented values can be arbitrary nonnegative real numbers, and because we have given the adversary full power in choosing their order, the values we have encountered in previous rounds contain no information that can help us make better decisions, that is, we cannot infer what a “large” or “small” value is for a given input until we have seen the entire input. Thus, in order to weaken the adversary, we consider the problem in the random-order.

It turns out that this reduction in power is enough to make the problem significantly less challenging. Indeed, taking advantage of the random order, we can use part of the input sequence to obtain an sort of “estimate” of the range of values in that sequence. That is, we consider the following algorithm:

Algorithm 2.1: Single-Secretary Problem

```
1 Reject the first $\frac{n}{e}$ elements
2 Set threshold $\tau \leftarrow$ maximum among rejected elements
3 Pick the first value after that which is larger than $\tau$
```

We will call an algorithm *wait-and-pick* if it rejects the first m elements and then picks the first value that is larger than all before it. Clearly, Algorithm 2.1 is a wait-and-pick algorithm with $m = \frac{n}{e}$. Because $\frac{1}{e} \approx 37\%$, this algorithm is known as the 37%-rule. Why $m = \frac{n}{e}$ is the correct choice for a wait-and-pick algorithm will become clear in the analysis of the algorithm.

Theorem 2.1.2. *As $n \to \infty$, Algorithm 2.1 picks the maximum value with probability at least $\frac{1}{e}$, thus it is $\frac{1}{e}$-competitive.*

Proof. Consider any wait-and-pick algorithm A that rejects the first m elements of the sequence. The probability that A picks the maximum value is:

$$\mathbb{P}[A \text{ picks max}] = \sum_{t=m+1}^{n} \mathbb{P}[A \text{ picks max} \mid v_t \text{ is max}] \cdot \mathbb{P}[v_t \text{ is max}]$$

$$= \sum_{t=m+1}^{n} \mathbb{P}[\text{max among } \{v_1, \ldots, v_{t-1}\} \text{ falls in } m \text{ first positions}] \cdot \mathbb{P}[v_t \text{ is max}]$$

$$= \sum_{t=m+1}^{n} \frac{m}{t-1} \cdot \frac{1}{n} = \frac{m}{n}(H_{n-1} - H_{m-1})$$

where in the second-to-last equality we used the random order of arrivals and $H_k = \sum_{i=1}^{k} \frac{1}{k}$ is the k-th harmonic number. Using the approximation $H_k \approx \log k + 0.57$ for $k \to \infty$, the above probability is approximately equal to $\frac{m}{n} \log \frac{n-1}{m-1}$ for large m, n. This quantity attains maximum value $\frac{1}{e}$ by setting $m = \frac{n}{e}$. □

Thus, by exploiting the random order of arrivals, we get a very simple wait-and-pick strategy that is *constant-competitive* in expectation. This is in stark contrast with the $\frac{1}{n}$ competitive ratio we managed to achieve in the fully adversarial setting. Wait-and-pick strategies seem like a natural way to approach this problem, as it is difficult to imagine a reasonable algorithm taking any other form. Indeed, the following theorem from [52] formalises this intuition.

Theorem 2.1.3. *The strategy that maximises the probability of picking the highest number can be assumed to be a wait-and-pick strategy.*

We omit the proof for brevity. We note only that the two previous theorems imply that, for $n \to \infty$, no algorithm can guarantee that it picks the maximum value with probability more than $\frac{1}{e}$. Crucially, this implies that no algorithm can guarantee a competitive ratio larger than $\frac{1}{e}$ for the Secretary Problem.

2.1.2 Multiple-Secretary Problem

A natural question to consider after studying the Single-Secretary problem is whether we can extend the insights we obtained in the case where we are not forced to pick only one value, but instead can continue picking elements as long as we do not violate a given constraint. The simplest such constraint is if we are allowed to pick k items in total, for a known $k \geq 1$. As before, we wish to maximise the expected total value we obtain on a given set of elements, and we are compared against the offline optimal that picks the k highest values in the set. This problem was first examined in [19]. In all that follows, we will denote by $S^\star$ the set chosen by the offline optimal algorithm and by $V^\star$ the corresponding optimal value.

The first algorithm we will examine draws direct inspiration from the Single-Secretary case; we will ignore (on expectation) the first δn values for some appropriately chosen $\delta \in (0,1)$, then set a threshold τ and pick the first k values larger than τ.

Algorithm 2.2: k-Secretary

```
1 Set $\varepsilon = \delta = O\left(\frac{\log k}{k^{1/3}}\right)$
2 Draw $m \sim \text{Binomial}(n, \delta)$
3 Ignore the first $m$ elements
4 Set threshold $\tau \leftarrow (1-\varepsilon)\delta k^{th}$ highest ignored value
5 Pick the next first $k$ elements greater than $\tau$
```

The reason for setting δ as above will become clear in the analysis. Since $\mathbb{E}[m] = \delta n$, we are indeed ignoring the first δn elements on expectation, thus we will ignore δk elements from $S^\star$ on expectation. Hence it would make sense to set the threshold equal to the δk^{th} highest ignored value, to try and match the lowest ignored value from $S^\star$, but we actually set it a bit higher to account for the variance in how many elements of $S^\star$ we ignore. The final point to discuss before analysing the algorithm's performance is the reason why we do not simply ignore the first δn elements outright, but instead ignore the first $m \sim \text{Binomial}(n, \delta)$ elements. This is done to ensure that each element is ignored with the same probability δ, independently of others (simply ignoring the first δn elements introduces correlations. i.e. if an element is ignored then each other is slightly less likely to do so). The reason we want this independence to hold is so that we may use Chernoff bounds in our analysis (Theorem A.0.2).

We have now established all we need to prove our competitive ratio:

Theorem 2.1.4. *Algorithm 2.2 achieves a competitive ratio of* $1 - O(\delta)$, *where* $\delta = O\left(\frac{\log k}{k^{1/3}}\right)$

Proof. Let $v' := \min_{v \in S^\star} v$ be the lowest value in $S^\star$ and v'' be the $(1-2\varepsilon)k^{th}$ highest value in $S^\star$. Intuitively, we want to set our threshold τ neither too low (because we might pick up too many low values) nor too high (because we might reject many high values). We consider the "bad" events $L = \{\tau < v'\}$ (corresponding to τ being too low) and $H = \{\tau > v''\}$ (corresponding to τ being too high). We will show that both of those events happen rarely.

Let T be the set of ignored elements, i.e. the set of elements that fall in the first m positions of the input. Recall that we set τ equal to the $(1-\varepsilon)\delta k^{th}$ highest value in T.

- For event L to hold, less than $(1-\varepsilon)\delta k$ elements from $S^\star$ appear in T. Since the expected number of values from $S^\star$ that appear in T is δk, by a Chernoff bound we have that L holds with probability at most $\exp(-\varepsilon^2\delta k/2)$.

- Similarly, for H to hold, more than $(1-\varepsilon)\delta k$ elements above v'' must appear in T. By definition of v'', the expected number of elements above v'' that appear in T is $(1-2\varepsilon)\delta k$, so by a Chernoff bound we have that H holds with probability at most $\exp(-\varepsilon^2\delta k/3)$.

Thus, by setting $\varepsilon = \delta = O\left(\frac{\log k}{k^{1/3}}\right)$, we have that both L and H happen with probability at most $O(\delta) = \frac{1}{\text{poly}(k)}$, and by a union bound we have that, with probability at least $1 - O(\delta)$, the event $\bar{L} \cap \bar{H}$ holds.

Notice that if $\tau \geq v'$, we will never run out of budget k. Also note, by the analysis of event H, that if $\bar{H}$ holds, then no more than $(1-\varepsilon)\delta k$ elements above v'' appear in T, so at least $(1-2\varepsilon-(1-\varepsilon)\delta)k = (1-O(\delta))k$ elements of $S^\star$ appear in the last $n-m$ positions. Thus, if $\bar{L} \cap \bar{H}$ holds, we will only pick values in $S^\star$ (because $\tau \geq v'$) and in fact we will pick all but $O(\delta k)$ such values. Due to symmetry, each of those values is equally likely not to be picked, thus we will lose value $O(\delta V^\star)$ on expectation. Overall, with probability at least $1 - O(\delta)$ we obtain an expected value of $V^\star(1-O(\delta))$ (and with probability at most $O(\delta)$ we might as well earn zero value), so our overall expected value is $V^\star(1-O(\delta))$ and the claim follows. □

Algorithm 2.2 provides a natural way to extend our strategy from the Single-Secretary setting to the k-Secretary one; indeed, we again ignore a fraction of the elements, use them to estimate a threshold between the top k values and the rest, and then simply accept any values we find that are above this threshold. The clever bit is that, instead of ignoring a constant fraction of values, the fraction of ignored elements is a function of k, and indeed $\delta \to 0$ as $k \to \infty$. This is quite natural, as the more values we can pick the less selective we need to be in order to remain competitive, thus we can be less careful when setting τ and we can use fewer "samples" to "learn" a good threshold.

However, the above idea can be significantly improved, by realising that there is no reason to use the same threshold for the entirety of the input sequence. Indeed, it is reasonable to assume that, the more elements of the input we see, the better able we are to estimate a good threshold, so we should try and *adapt* τ as the algorithm goes on. This is the basic idea behind Algorithm 2.3, due to [18].

Algorithm 2.3 runs in phases of exponentially increasing length. At the beginning of each phase, a threshold is calculated based on *all* values seen in the previous phases. At the beginning of phase j, on expectation $k_j = 2^j\delta k$ elements from $S^\star$ will have appeared in the previous phases, so as in the previous algorithm we aim for the k_j-th largest of those values and we set τ_j as the $(1-\varepsilon_j)k_j$-th largest such value, so as not to fall below the desired k_j-th largest value. In contrast with our previous algorithm, however, we take advantage of the fact that, the more values we have seen in previous phases, the less variance there is in the number of elements of $S^\star$ that have appeared in those phases, thus we can be more confident and decrease ε_j as phases go on.

The following theorem shows the competitive ratio of our improved algorithm, and its proof uses the same tools as the previous theorem's. Note that since n_j is the sum of independent binomial variables, it is itself a random variable Binomial$(n, 2^j\delta)$.

Algorithm 2.3: k-Secretary, Adaptive Threshold

```
1 Set $\delta := O\left(\sqrt{\frac{\log k}{k}}\right)$
2 Draw $m_0 \sim$ Binomial$(n, \delta)$ and $m_j \sim$ Binomial$(n, 2^{j-1}\delta)$ for $j \in [\log \frac{1}{\delta}]$
3 Denote $n_j := \sum_{i\in[j]} m_i$ for $j \in [\log \frac{1}{\delta}]$
4 Ignore the first $n_0 = \delta n$ items
5 Let $W_j := (n_j, n_{j+1}]$ be the *window* in which phase $j$ runs
6 for $j \in [0, \log \frac{1}{\delta}]$ do
7     Set $k_j := \frac{k}{n} n_j$ and $\varepsilon_j := \sqrt{\frac{\delta}{2^j}}$
8     Set threshold $\tau_j$ to be the $(1-\varepsilon_j)k_j$-th largest value among the first $n_j$ items
9     Choose any value in window $W_j$ above $\tau_j$ (until budget $k$ is exhausted)
```

Theorem 2.1.5. *Algorithm 2.3 has competitive ratio* $\left(1 - O\left(\sqrt{\frac{\log k}{k}}\right)\right)$.

Proof. Fix a phase $j \in [0, \log \frac{1}{\delta}]$, and let T_j be the set of the first n_j values (i.e. the values seen in the previous phases). As in the proof of Theorem 2.1.4, let $v' := \min_{v \in S^\star} v$ be the lowest value in $S^\star$ and v_j'' be the $(1-2\varepsilon_j)k_j^{th}$ highest value in $S^\star$. Since the expected number of elements above v' in T_j is $k_j = 2^j \delta k$ and the expected number of elements above v'' in T_j is $(1-2\varepsilon_j)k_j$, by a Chrenoff bound we have that $\mathbb{P}[\tau_j < v'] \leq \exp(-\varepsilon_j^2 k_j/2) = \exp(-\delta k/2) = \frac{1}{\text{poly}(k)}$ and $\mathbb{P}[\tau_j > v_j''] \leq \exp(-\varepsilon_j^2 k_j/3) = \frac{1}{\text{poly}(k)}$.

Taking a union bound over all $j \in [0, \log \frac{1}{\delta}]$ we have that the event $G = \{\forall j \in [\log \frac{1}{\delta}] : v' \leq \tau_j \leq v_j''\}$ holds with probability at least $1 - \frac{\log(1/\delta)}{\text{poly}(k)}$. Condition on this "good" event G. In this case, since $\tau_j \geq v'$ for all j, we only pick items in $S^\star$ and never run out of budget. Additionally, by the definition of v_j'' and ε_j, we have that $v_0'' \geq v_1'' \geq \ldots \geq v''_{\log \frac{1}{\delta}} = v'$. Hence, any element higher than v_0'' will be picked iff it is not in T_0, and for all $j \geq 1$, any element in $[v_j'', v_{j-1}'')$ will be picked iff it is not in T_j. Each element falls in T_j with probability $2^j\delta$ (independently of each other), thus we get an expected value of:

$$\begin{aligned}
&\sum_{v \geq v_0''} v(1-\delta) + \sum_{j=1}^{\log \frac{1}{\delta}} \sum_{v_j'' \leq v < v_{j-1}''} v(1 - 2^j\delta) \\
&= V^\star - \sum_{v \geq v_0''} v\delta - \sum_{j=1}^{\log \frac{1}{\delta}} \sum_{v_j'' \leq v < v_{j-1}''} v 2^j \delta \\
&\geq V^\star - (1-2\varepsilon_0)v_0''\delta - \sum_{j=1}^{\log \frac{1}{\delta}} 2(\varepsilon_{j-1} - \varepsilon_j)k v_j'' 2^j \delta \\
&\geq V^\star(1-\delta) - 2(\sqrt{2}-1)\delta^{3/2} k \sum_{j=1}^{\log \frac{1}{\delta}} v_j'' 2^{j/2}
\end{aligned}$$

where the first inequality follows from the fact that, if $v_1 > v_2 > \ldots > v_k$ are the top k items, the elements v such that $v_j'' \leq v < v_{j-1}''$ are precisely the elements in $\{v_i : (1 - 2\varepsilon_{j-1})k \leq i \leq (1-2\varepsilon_j)k - 1\}$, thus their cardinality is $2(\varepsilon_{j-1} - \varepsilon_j)k$. Under the constraint $v_0'' > v_1'' > \ldots > v''_{\log \frac{1}{\delta}}$, the sum in the last term is maximised when all elements have value

$V^\star/k$ (so as to maximise the value $v''_{\log \frac{1}{\delta}}$ which has the largest coefficient), thus the negative term becomes $O(V^\star\delta^2)$. Hence, conditioned on the good event G we obtain value $V^\star(1 - O(\delta))$, and because G happens with probability at least $1 - O(\delta)$ the claim follows. □

By utilising the idea of adaptive thresholds, Algorithm 2.3 achieves significantly improved performance compared to Algorithm 2.2, and in fact its competitive ratio is nearly optimal, as shown by the following theorem due to [19].

Theorem 2.1.6. *No algorithm can achieve a competitive ratio better than $1 - O\left(\frac{1}{\sqrt{k}}\right)$ for the k-Secretary Problem.*

It is possible to actually reach this optimal $1 - O\left(\frac{1}{\sqrt{k}}\right)$ competitive ratio by a more involved recursive algorithm, again due to [19].

Algorithm 2.4: k-Secretary Problem, Recursive

```
Function REC($k$, $(v_1, \ldots, v_n)$):
    if $k = 1$ then
        Run Algorithm 2.1
    else
        Draw $m \sim$ Binomial$(m, 1/2)$
        $\ell \leftarrow \lfloor k/2 \rfloor$
        Run REC($\ell$, $(v_1, \ldots, v_m)$)
        Let $y_1 > y_2 > \ldots > y_m$ be the first $\ell$ samples ordered
        After the $m$-th sample, pick every element that exceeds $y_\ell$ (until budget $k$ is
          exhausted)
```

Theorem 2.1.7. *Algorithm 2.4 is $\left(1 - \frac{5}{\sqrt{k}}\right)$-competitive.*

We omit the proofs of the last two theorems, as they are rather involved and not central to the thesis. Note that the bound guaranteed by the last algorithm, though theoretically optimal and tending to zero as $k \to \infty$, is vacuous for small values of k. In [53] two simple algorithms are proposed and it is shown that they are $\frac{1}{e}$-competitive for all k. This result is improved upon in [54], where a natural algorithm is proposed with competitive ratio strictly grater than $\frac{1}{e}$ for small $k \geq 2$, and it is specifically shown that its competitive ratio increases from 0.41 for $k = 2$ to 0.75 for $k = 100$.

2.1.3 Matroid Secretary Problem

In the previous section, we considered how the Single-Secretary problem would change if, instead of being allowed to pick only one item, we were allowed to pick items so long as we did not violate some sort of constraint. We saw that the simplest kind of such constraint is that we may be allowed to pick no more than k elements. However, we can generalise and consider a significantly more rich class of constraints, called *matroid constraints*. Matroids are mathematical objects of major theoretical interest, as they elegantly generalise and abstract concepts in many fields, including graph theory, combinatorial optimisation, linear algebra, geometry and topology. Basic definitions on matroids can be found in the appendix (Definition A.0.1).

In the *Matroid Secretary problem*, an adversary constructs a matroid $\mathcal{M} = (\mathcal{U}, \mathcal{I})$ and assigns a value $v : \mathcal{U} \to \mathbb{R}_{\geq 0}$ to each element in the ground set. We are presented each element of $\mathcal{U}$ in a random order. Upon receiving one element, we can either pick it or forever discard it, and we cannot drop previously selected elements. Our aim is to maximise the total value of the elements we pick. However, the set of items we have picked must always be an independent set of the matroid; that is, we cannot pick an element if adding it to the ones we have already picked would create a dependent set. This problem was introduced in [20]. Observe that the k-Secretary problem we considered in the previous section is a special case of the Matroid Secretary problem, where the underlying matroid is k-uniform.

The first algorithm introduced for the Matroid Secretary problem was Algorithm 2.5 due to [20]. In what follows, $M = (\mathcal{U}, \mathcal{I})$ is the underlying matroid, r is its rank and $n = |\mathcal{U}|$. We denote by $e_1, \dots, e_n$ the elements of the matroid and by $v_1, \dots, v_n$ their respective values.

Algorithm 2.5: Matroid-Secretary, Threshold Pricing

```
1 Ignore the first $s = \lceil n/2 \rceil$ elements and let $\hat{v}$ be their highest value
2 Select a uniformly random $j \in \{0, \dots, \log n\}$
3 Set threshold $\tau := \hat{v}/2^j$
4 Initialise the set of selected elements $B \leftarrow \emptyset$
5 for $t \in \{\lceil n/2 \rceil + 1, \dots, n\}$ do
6     if $v_t \geq \tau$ and $B \cup \{e_t\} \in \mathcal{I}$ then
7         $B \leftarrow B \cup \{e_t\}$
```

Theorem 2.1.8. *Algorithm 2.5 is $\frac{1}{32\lceil \log r \rceil}$-competitive for any matroid of rank r.*

Proof. Let $B^\star$ denote the max-weight basis of the matroid, consisting of elements $e_1, \dots, e_r$ with values $v_1 > v_2 > \dots > v_r$ respectively, and let $V^\star = \sum_{i \in [r]} v_i$ be the value of $B^\star$. Let $q \in [r]$ be the maximum index such that $v_q \geq v_1/r$, so either $q = r$ or $v_{q+1} < v_1/r$. By this definition, $\sum_{i=q+1}^{r} v_i \leq v_1$, so $\sum_{i=q+1}^{r} v_i \leq V^\star/2$ and hence $\sum_{i \in [q]} v_i \geq V^\star/2$.

For any set $A \subseteq \mathcal{U}$, let $n_i(A) = |\{e \in A : v_e \geq v_i\}|$ be the number of elements of A with value at least v_i and let $m_i(A) = |\{e \in A : v_e \geq v_i/2\}|$ be the number of elements of A with value at least $v_i/2$. The sum of the q largest values of elements in $B^\star$ is

$$v_q n_q(B^\star) + \sum_{i=1}^{q-1} (v_i - v_{i+1}) n_i(B^\star)$$

where by definition $n_i(B^\star) = i$. By our above argument, this sum is at least equal to $V^\star/2$. Let B be the independent set output by our algorithm. By the definition of m_i, the value of B is at least

$$\frac{1}{2} v_q m_q(B) + \frac{1}{2} \sum_{i=1}^{q-1} (v_i - v_{i+1}) m_i(B)$$

Combining the above, in order to show that B obtains value at least $\frac{1}{32 \log r} V^\star$, it suffices to show $\mathbb{E}[m_i(B)] \geq \frac{1}{8 \log r} n_i(B^\star)$ for all $i \in [q]$.

The case $i = 1$ is a special case. Let T be the set of ignored elements. With probability $1/4$, $e_1 \notin T$ and $e_2 \in T$. Conditional on this event, with probability $1/\log r$ we pick $j = 0$ in our algorithm and thus set $\tau = v_2$, so we will pick e_1 as it is the only element with value more than v_2. Thus we pick e_1 with probability at least $\frac{1}{4\log r}$, so $\mathbb{E}[m_1(B)] \geq \frac{1}{4\log r}$, while $n_1(B^\star) = 1$.

Now fix some $i \in \{2, \dots, q\}$. Consider the event E that $e_1 \in T$ and additionally we pick j in our algorithm such that $\frac{v_1}{2^{j-1}} > v_i \geq \frac{v_1}{2^j}$. Since $i \leq q$, by the definition of q we have $v_i \geq v_1/r$, thus such a $j \in \{0, \dots, \log r\}$ always exists. Because e_1 is the maximum valued element in the matroid, under event E we will set the threshold equal to $v_1/2^j$ for the above j. Since $e_1 \in T$ with probability $1/2$, we have $\mathbb{P}[E] = \frac{1}{2\log r}$.

Condition on event E. Then there is an independent set A of size at least i each of whose values exceeds τ (namely, $\{e_1, \dots, e_i\}$). Let $A' \subseteq A$ be the set of elements of A that appear in the second half of the input, thus $\mathbb{E}[|A'| \mid E] \geq \frac{i-1}{2} \geq i/4$. By the hereditary property, A' is also an independent set. Our set B will contain at least $|A'|$ elements; otherwise, if $|B| < |A'|$, then by the exchange property there exists an $e \in A' \setminus B$ such that $B \cup \{e\}$ is independent, but because $e \in A$ we have $v_e \geq \tau$ and we should have picked e in B. Therefore, we have that $\mathbb{E}[|B| \mid E] \geq i/4$, and because each element of B has value at least $\tau \geq v_i/2$, we have $m_i(B) = |B|$, so $\mathbb{E}[m_i(B) \mid E] \geq i/4 = n_i(B^\star)/4$. Removing the conditioning on E and recalling that $\mathbb{P}[E] = \frac{1}{2\log k}$, we obtain that $\mathbb{E}[m_i(B)] \geq \frac{1}{8\log r} n_i(B^\star)$ and the claim follows. □

The above algorithm is rather elegant and, much like the previous ones we considered, relies on the idea of ignoring a fraction of the elements and using them to learn a good threshold. Its simplicity and the fact that it works for general matroids make it appealing; however, its competitive ratio can be exponentially improved from $O\left(\frac{1}{\log r}\right)$ to $O\left(\frac{1}{\log\log r}\right)$ using more sophisticated techniques. The first algorithm achieving this competitive ratio was presented in [21]. Afterwards, in [22], a different algorithm with the same asymptotic guarantees was proposed, which is considerably simpler and greatly improves on the constants hidden by the asymptotic notation. We will not present those algorithms or their analyses, as they are rather involved. A survey of the work leading up to these results can be found in [55]. On the negative side, the only known lower bound for general matroids is the factor of $\frac{1}{e}$ from the Single-Secretary case. Closing this gap is a highly interesting open problem.

Instead of trying to design an algorithm that performs well on any matroid, a natural approach is to limit ourselves to specific matroid structures of interest and see whether we can take advantage of their additional properties in order to go below the $O\left(\frac{1}{\log\log r}\right)$ barrier. In fact, we have already done this in the k-Secretary Problem (which corresponds to k-uniform matroids) to obtain a significantly better $1 - O\left(\frac{1}{\sqrt{k}}\right)$ competitive ratio. We will examine the cases of partition, graphic and degree-d transversal matroids, and we will show the following theorem which combines the results of [56] and [23].

Theorem 2.1.9. *The Matroid Secretary problem admits a $\frac{1}{e}$-competitive algorithm for partition matroids, a $\frac{1}{2e}$-competitive algorithm for graphic matroids and a $\frac{1}{ed}$-competitive algorithm for degree-d transversal matroids.*

First, note that the case of partition matroids is trivial, since we can simply run the $\frac{1}{e}$-competitive Algorithm 2.1 on each part of the partition to pick a signle element from each part. Since the maximum value base of a partition matroid consists simply of the maximum value element from each part of the partition, and because the element we select in each part of the partition has expected value at least equal to $\frac{1}{e}$ of the maximum element of that part, the claim directly follows. For the other two cases, we will reduce the problem to the case of partition matroids.

For a matroid $\mathcal{M}$ and a value function $w : \mathcal{U} \to \mathbb{R}_{\geq 0}$, we denote by $\textsc{Opt}(\mathcal{M}, w)$ the maximim value basis of $\mathcal{M}$. We will make use of the following property of graphic and transversal matroids.

Definition 2.1.1. A matroid $\mathcal{M} = (\mathcal{U}, \mathcal{I})$ has the *α-partition property* if we can randomly construct a partition matroid $\mathcal{M}' = (\mathcal{U}', \mathcal{I}')$ with $\mathcal{U}' \subseteq \mathcal{U}$, such that for every value function w.

- $\mathbb{E}[\textsc{Opt}(\mathcal{M}', w)] \geq \frac{1}{\alpha} \cdot \textsc{Opt}(\mathcal{M}, w)$
- $\mathcal{I}' \subseteq \mathcal{I}$

If a matroid $\mathcal{M}$ has the α-partition property, then we can randomly construct the partition matroid $\mathcal{M}'$ guaranteed by this property and then run the $\frac{1}{e}$-competitive algorithm described above to obtain an independent set of value at least $\frac{1}{e}\,\mathbb{E}[\textsc{Opt}(\mathcal{M}', v)] \geq \frac{1}{\alpha e} \cdot \textsc{Opt}(\mathcal{M}, v)$. Since $\mathcal{I}' \subseteq \mathcal{I}$, the independent set returned by this algorithm remains independent in the original matroid $\mathcal{M}$. This implies the following lemma.

Lemma 2.1.10. *If a matroid $\mathcal{M}$ has the α-partition property, there exists a $\frac{1}{\alpha e}$-competitive algorithm for it*

All that remains now is to demonstrate that graphic matroids and degree-d transversal matroids have the α-partition property for some α.

Lemma 2.1.11. *Transversal matroids with maximum degree d have the d-partition property.*

Proof. Given a bipartite graph $G = (U, V; E)$ where the vertices in U are the elements of a transversal matroid $\mathcal{M}$, we form a partition matroid $\mathcal{M}'$ by creating one bin P_v of the partition for each vertex in $v \in V$, and then assigning each element of U to the bin of one of its neighbours in V uniformly at random. Clearly, any independent set U' in $\mathcal{M}'$ is also independent in $\mathcal{M}$, since if $u \in U'$ belongs to bin P_v in $\mathcal{M}'$ then we can match u to v in $\mathcal{M}$. Additionally, for each element u in the optimal base $B^\star$ for $\mathcal{M}$, if u gets matched to v in $B^\star$, let X_u be equal to 1 if u belongs to bin P_v in $\mathcal{M}'$ and 0 otherwise. Clearly $\mathbb{P}[X_u = 1] = 1/d$, and if we only pick the $u \in U$ such that $X_u = 1$ in $\mathcal{M}'$ we create an independent set of $\mathcal{M}'$ whose expected value is at least $\sum_{u \in B^\star} w(u)\,\mathbb{P}[X_u = 1] = \frac{1}{d} \cdot \textsc{Opt}(\mathcal{M}, w)$. The optimal independent set has expected value at least that large, thus the claim follows. □

Lemma 2.1.12. *Graphic matroids have the 2-partition property.*

Proof. Let $G = (V, E)$ be the graph defining the graphic matroid $\mathcal{M}$. Arbitrarily order the vertices $v_1, v_2, \ldots, v_n$ of V. With probability $1/2$ orient each edge of G from the vertex with the smaller index to that with the larger index, otherwise orient each edge in the opposite direction. Clearly, this leaves us with no directed cycles, as the vertex indices along any directed path must be either strictly increasing or strictly decreasing. Consider the partition matroid $\mathcal{M}'$ formed by creating one bin for each vertex $v \in V$ and adding each edge e to the bin corresponding to its tail. Since we do not have any directed cycles, any *undirected* cycle needs to contain at least two edges with the same tail, thus by picking at most one outgoing edge for each vertex we create no undirected cycles. In other words, any independent set in $\mathcal{M}'$ is also independent in $\mathcal{M}$.

To analyse the value of $\textsc{Opt}(\mathcal{M}', w)$, consider the optimal basis of the original matroid $\mathcal{M}$ which corresponds to a forest F in G. Root each tree of F at an arbitrary vertex. For each $e \in F$, let X_e be equal to 1 if e is oriented towards the root of the tree in which it belongs in F and 0 otherwise. Clearly $\mathbb{P}[X_e = 1] = 1/2$, and an independent set of $\mathcal{M}'$ can be formed by picking only the edges with $X_e = 1$, in which case the expected value is at least $\sum_{e \in F} w(e)\,\mathbb{P}[X_e = 1] = \frac{1}{2}\textsc{Opt}(\mathcal{M}, w)$. The optimal independent set has expected value at least that large, thus the claim follows. □

From the last three lemmas and the above discussion, Theorem 2.1.9 follows. Thus we have designed $O(1)$-competitive algorithms for the interesting special cases of partition and graphic matroids, as well as a $O(\frac{1}{d})$-competitive algorithm for degree-d transversal matroids. While those algorithms are notable for their simplicity, they are not optimal. Indeed, in [24] a $\frac{1}{4}$-competitive algorithm for graphic matroids is presented, while in [25] a $\frac{1}{e}$-competitive algorithm is given for the random-order weighted bipartite matching problem, which generalises the transversal matroid secretary problem. We will not present the former algorithm for brevity, while we will examine the latter one in Section 2.2.

2.2 Online Matching Problems

2.2.1 Maximum Weight Matching

Consider a bipartite graph $G = (L, R; E)$ with n vertices in L and m vertices in R. We can think of the vertices in L as representing agents and of the vertices in R as representing items the agents wish to obtain. Each agent $i \in [n]$ has a value $v_{ij} \in \mathbb{R}_{\geq 0}$ for item $j \in [m]$. In the online setting, we are given all m items up front and the n agents arrive in sequence. Upon arrival, each agent i reveals their valuations v_{ij} for each item, and we may irrevocably assign up to one of the remaining items to i to obtain value v_{ij}. Our objective is to maximise the total value obtained, and we again examine the random-order case. Note that if each agent i has the exact same valuation v_i for each item, then we recover the Matroid Secretary problem for transversal matroids discussed in Section 2.1.3.

The algorithm we will examine, due to [25], is remarkably simple, yet achieves the optimal competitive ratio.

Algorithm 2.6: Random-Order Max-Weight Matching

```
1 Let $L'$ be the first $\lfloor n/e \rfloor$ vertices; ignore those vertices
2 $M \leftarrow \emptyset$
3 for $\ell \in \{\lceil n/e \rceil, \dots, n\}$ do
4     $L' \leftarrow L' \cup \{\ell\}$
5     Compute a max-value matching $M^{(\ell)}$ for $(L', R)$
6     If $\ell$ is matched to $r$ in $M^{(\ell)}$, let $e^{(\ell)} = (\ell, r)$
7     if $M \cup \{e^{(\ell)}\}$ is a matching then
8         $M \leftarrow M \cup \{e^{(\ell)}\}$
```

Theorem 2.2.1. *Algorithm 2.6 is $\frac{1}{e}$-competitive.*

Proof. Let M be the matching returned by the algorithm. Define $\textsc{Opt}$ to be the weight of the optimal offline matching and $w\left(M^{(\ell)}\right)$ to be the weight of matching $M^{(\ell)}$. For each $\ell \in L$, define a random variable A_ℓ equal to the weight of the edge assigned to ℓ in M (so $A_\ell = 0$ if ℓ is not matched in M). We will show the following, from which the theorem will directly follow:

$$\mathbb{E}[A_\ell] \geq \frac{\lfloor n/e \rfloor}{\ell - 1} \cdot \frac{\textsc{Opt}}{n} \tag{2.1}$$

Whenever vertex k arrives, we can model the choice of the random permutation of the vertices $(1, \dots, k)$ that have arrived so far as a sequence of the following *independent* random experiments: first choose a uniformly random set of size k from L, then determine the order of those k vertices by iteratively selecting a vertex at random and removing it. We will utilise this interpretation to first argue about the expected weight of an edge $e^{(\ell)}$ considered by the algorithm, and then argue about the probability that this edge will be included in M.

Fix a step $\ell \in [n]$, so that $|L'| = \ell$. To begin with, since the current vertex ℓ can be viewed as being selected uniformly at random from the set L', and because $e^{(\ell)}$ is the edge assigned to ℓ in the matching $M^{(\ell)}$, we have $\mathbb{E}\left[w\left(e^{(\ell)}\right)\right] \geq \frac{w\left(M^{(\ell)}\right)}{\ell}$. Also, since L' can be viewed as being selected uniformly at random from set L, we have $\mathbb{E}\left[w\left(M^{(\ell)}\right)\right] \geq \frac{\ell}{n}\textsc{Opt}$. Combining the above we have:

$$\mathbb{E}\left[w\left(e^{(\ell)}\right)\right] \geq \frac{\textsc{Opt}}{n} \tag{2.2}$$

Notice that, up to this point, we have used the random choice of the set L' from L and the random choice of vertex ℓ from L', but we have not utilised the random choice of vertices $\{1, \dots, \ell - 1\}$. This is precisely what we will need to argue about the probability that $e^{(\ell)}$ will be included in M.

The edge $e^{(\ell)} = (\ell, r) \in M^{(\ell)}$ will be added to M iff r has not already been picked in an earlier step, that is, iff $r \notin e^{(k)}$ for all previous steps $k \in \{\lceil n/e \rceil, \dots, \ell - 1\}$. For any such step, since the vertex k is viewed as being picked uniformly at random from the set of vertices revealed up to that point, we have that $\mathbb{P}\left[r \notin e^{(k)}\right] \geq 1 - \frac{1}{k}$. As before, the order of the vertices that arrived before k is irrelevant for the event $\{r \notin e^{(k)}\}$, so the respective events $\{r \notin e^{(k')}\}$ for $k' < k$ can be regarded as independent. Thus, by applying the same reasoning inductively from $k = \ell - 1$ down to $\lceil n/e \rceil$ we obtain:

$$\mathbb{P}\left[\bigcap_{k=\lceil n/e \rceil}^{\ell-1} \{r \notin e^{(k)}\}\right] = \prod_{k=\lceil n/e \rceil}^{\ell-1} \mathbb{P}[r \notin e^{(k)}] \geq \prod_{k=\lceil n/e \rceil}^{\ell-1} \left(1 - \frac{1}{k}\right) = \frac{\lceil n/e \rceil - 1}{\ell - 1}$$

Thus, the probability that r has not been picked up to step ℓ is at least $\frac{\lceil n/e \rceil}{\ell - 1}$. Combining this with (2.2) and recalling the definition of A_ℓ, we obtain (2.1).

Having shown (2.1), we can now lower bound the expected weight of the obtained matching M by summing over all variables A_ℓ:

$$\begin{aligned}\mathbb{E}[w(M)] = \mathbb{E}\left[\sum_{\ell=1}^{n} A_\ell\right] &\geq \sum_{\ell=\lceil n/e \rceil}^{n} \frac{\lfloor n/e \rfloor}{\ell - 1} \cdot \frac{\textsc{Opt}}{n} \\ &= \frac{\lfloor n/e \rfloor}{n} \cdot \textsc{Opt} \cdot \sum_{\ell=\lceil n/e \rceil}^{n-1} \frac{1}{\ell} \geq \left(\frac{1}{e} - \frac{1}{n}\right) \cdot \textsc{Opt}\end{aligned}$$

where in the last inequality we used the fact that $\frac{\lfloor n/e \rfloor}{n} \geq \frac{1}{e} - \frac{1}{n}$ and also $\sum_{\ell=\lceil n/e \rceil}^{n-1} \frac{1}{\ell} \geq \log\left(\frac{n}{\lceil n/e \rceil}\right) \geq 1$. $\square$

Since the Single-Secretary problem is the special case of the Random-Order Max-Weight Matching problem (in the case where there is only one item), the lower bound of $\frac{1}{e}$ from the Single-Secretary case holds in this setting, thus Algorithm 2.6 indeed achieves the optimal competitive ratio.

2.2.2 Minimum Augmentation Matching

We will now consider a matching problem of significantly different flavour than Max-Weight Matching and the other secretary-like problems we have examined. First of all, it will be a *minimisation* problem, where we will be able to "undo" past choices as the input sequence is revealed. However, instead of being concerned with the quality of the solution we obtain, we will instead focus on maintaining a feasible solution throughout the execution of the algorithm, while performing as few "undos" as possible (in other words, we are concerned not with the quality but with the *stability* of our solutions).

Concretely, consider a bipartite graph $G = (U, V; E)$ with $|U| = |V| = n$. For simplicity, assume that G has a perfect matching. We can think of the vertices in U as representing clients and the vertices in V as representing (unit-capacity) servers, in which case the perfect matching in G represents a way to assign each client to a server. We are given the servers in V up front, while the clients in U arrive sequentially, in random order. When a client $u_t \in U$ arrives, all edges adjacent to it (i.e. all servers to which it can be connected) are revealed to us. We must, at every timestep, output a maximum matching between the revealed clients and the servers (i.e. we must serve each client), and we are allowed to drop edges chosen at a previous timesteps or to pick edges previously dropped. At each time, we pay the *switching cost* between our current matching and the previous one, that is, if $M^{(t-1)}$ and $M^{(t)}$ are the matchings outputted by our algorithm at timesteps $t-1$ and t, the cost we incur is $M^{(t-1)} \triangle M^{(t)}$.

Suppose we have a matching $M^{(t-1)}$ at the end of timestep $t-1$, and that client u_t arrives at timestep t (and is therefore unmatched). We define an *augmenting path* for u_t in $M^{(t-1)}$ to be a path starting from u_t that alternates between edges not in $M^{(t-1)}$ and edges in $M^{(t-1)}$, and terminates in an (unmatched) server in V. Clearly, such a path always exists. We will say that we augment a matching M *along an augmenting path* P if we update M to be equal to $M \triangle P$. We will examine the following simple algorithm due to [26].

Algorithm 2.7: Random Order Minimum Augmentation Matching

```
1 $M \leftarrow \emptyset$
2 for $t \in [n]$ do
3     Compute a *shortest* augmenting path $P$ for $u_t$ and $M$
4     Augment $M$ along $P$
```

Theorem 2.2.2. *Algorithm 2.7 has expected switching cost $O(n \log n)$.*

Proof. Fix a perfect offline matching M^* and a round $n-k$. If the remaining k clients were to arrive all at once, then it is easy to see how $M^* \triangle M$ forms k disjoint augmenting paths. Indeed, if u is one of those clients, starting from u we can follow the unique edge $(u, v_1) \in M^*$, then if v_1 is unmatched in M we are done, otherwise follow the edges $(v_1, u_1) \in M$ and $(u_1, v_2) \in M^*$ and repeat. The paths are disjoint because M and M^* are matchings, so each vertex in $M * \triangle M$ has degree at most 1. Therefore, we can augment M along those

k paths to obtain a perfect matching, and in doing so each client switches servers at most once, so our total augmenting cost is at most $2n$.

Now, since the next client is chosen uniformly at random among the k remaining, its corresponding augmenting path in the above collection has expected length at most $2n/k$. Since we pick the shortest augmenting path for the arrived client, we pay an expected switching cost of at most that much at round $n-k$. Thus our total expected switching cost is at most $\sum_{k=1}^{n} \frac{2n}{k} = 2nH_n = O(n \log n)$. $\square$

It is worth noting that, in the same paper ([26]), two other interesting results were shown. First, a more sophisticated analysis of Algorithm 2.7 shows that it not only achieves an expected switching cost of $O(n \log n)$ on expectation, but it actually suffers cost at most that much *with high probability* $1 - n^{-2}$. Secondly, it is shown that any online algorithm must suffer expected switching cost at least $\Omega(n \log n)$. Thus, Algorithm 2.7 is in fact an optimal algorithm for the problem of Random-Order Min-Augmentation Matching.

It is also worthwhile to discuss the known upper and lower bounds for this problem in the fully adversarial setting and compare them with those we presented above. The best known algorithm for this case is, in fact, the same Algorithm 2.7 that works in the random-order setting. Through a considerably more complicated analysis, it can be shown that this algorithm suffers a switching cost of at most $O(n \log^2 n)$ in the worst case – see [45], [46]. On the negative side, the best lower bound is the same $\Omega(n \log n)$ that we mentioned above. Thus, we obtain another example where enforcing random-order weakens the adversary enough to significantly simplify the analysis and lead to better performance guarantees compared to the worst case.

Chapter 3

Prophet Models

3.1 Prophet Inequalities

We will now turn our attention to the prophet setting as described in Chapter 1, and we will begin our examination from the simplest problem we can consider in this model, which is analogous to the Secretary one. There are n random variables $X_1, \ldots, X_n$ taking values in $\mathbb{R}_{\geq 0}$. The distribution of each X_i is chosen by an adversary to be $\mathcal{D}^{(i)}$, and then the realised values are revealed in order from 1 to n. Whenever a value is revealed, we may either pick the value or forever discard it, and our goal is to maximise the total reward obtained. Similarly to the way we proceeded in Section 2.1, we will analyse this problem in various settings, which differ as to the constrains imposed on the elements we can pick. Before we continue, it is useful to recall that, in the prophet setting, we compare our algorithm's expected reward $\mathbb{E}[\textsc{Alg}]$ with the *expected* offline optimal reward $\mathbb{E}[\textsc{Opt}]$, where both expectations are under the product distribution $\prod_{i\in[n]} \mathcal{D}^{(i)}$.

3.1.1 Single-Choice Setting

In the simplest setting of the Prophet Inequality problem, we are only allowed to pick a single element from the input. Defining $X_{\max} := \max\{X_1, \ldots, X_n\}$, our expected reward is compared against $\mathbb{E}[X_{\max}]$. In this setting, it is easy to see that we cannot hope to obtain a competitive ratio better than 2. Indeed, consider the case with two random variables X_1 and X_2, where $\mathbb{P}[X_1 = 1] = 1$, $\mathbb{P}\left[X_2 = \frac{1}{\varepsilon}\right] = \varepsilon$ and $\mathbb{P}[X_2 = 0] = 1-\varepsilon$ for some $\varepsilon \ll 1$. Any algorithm picks value 1 with some probability p, so its expected reward is $p + (1-p) = 1$, whereas the expected offline optimal value is 1 with probability $1-\varepsilon$ and $\frac{1}{\varepsilon}$ with probability ε, so $\mathbb{E}[X_{\max}] = 2 - \varepsilon$.

What may be surprising is that this example encompasses the worst case for the problem. Indeed, in [57] an algorithm was proposed with expected reward at least $\frac{1}{2}\,\mathbb{E}[X_{\max}]$. Later, it was shown in [58] that this guarantee can be obtained by a simple thresholding rule, which we will now examine.

We consider that we are given full knowledge of the distributions $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(n)}$ beforehand. The thresholding rule we will analyse is simple: compute the median τ of the distribution of $X_{\max}$, i.e. $\mathbb{P}[X_{\max} \geq \tau] = 1/2$, and pick the first X_i that exceeds τ.

Theorem 3.1.1. *The median rule is $\frac{1}{2}$-competitive for the Single-Choice Prophet Inequality problem.*

Proof. We denote $(x)^+ = \max\{x, 0\}$ and let $p_\tau = \mathbb{P}[\exists X_i \geq \tau]$ for some $\tau \in \mathbb{R}_{\geq 0}$. For simplicity, assume that there is no point mass at τ (the proof can be extended easily to discrete distributions). Consider an algorithm $\textsc{Alg}_\tau$ that picks the first value higher than τ. Then the probability that $\textsc{Alg}_\tau$ picks some item is p_τ, so its expected reward is:

$$\begin{aligned}
\mathbb{E}[\textsc{Alg}_\tau] &= \sum_{i=1}^{n} \mathbb{P}[\forall j < i : X_j < \tau]\, \mathbb{P}[X_i \geq \tau]\, \mathbb{E}[X_i \mid X_i \geq \tau] \\
&= \tau \cdot \sum_{i=1}^{n} \mathbb{P}[\{\forall j < i : X_j < \tau\} \cap \{X_i \geq \tau\}] \\
&\qquad + \sum_{i=1}^{n} \mathbb{P}[\forall j < i : X_j < \tau]\, \mathbb{E}[\mathbf{1}_{\{X_i \geq \tau\}}(X_i - \tau)] \\
&\geq \tau \cdot p_\tau + (1 - p_\tau) \sum_{i=1}^{n} \mathbb{E}[(X_i - \tau)^+]
\end{aligned}$$

where in the last line we used the fact that $\{\forall j < i : X_j < \tau\} \supseteq \{\forall j \in [n] : X_j < \tau\}$. Now, observing that $\mathbb{E}[X_{\max}] \leq \tau + \mathbb{E}[(X_{\max} - \tau)^+] \leq \tau + \mathbb{E}\left[\sum_{i\in[n]}(X_i - \tau)^+\right]$, we obtain:

$$\mathbb{E}[\textsc{Alg}_\tau] \geq \tau \cdot p_\tau + (1 - p_\tau)\left(\mathbb{E}[X_{\max}] - \tau\right)$$

Hence, setting τ such that $p_\tau = \mathbb{P}[X_{\max} \geq \tau] = 1/2$ proves the claim. $\square$

From the last inequality of the proof, it can be seen directly that if τ is instead chosen to be equal to half the expected value of the maximum value, i.e. $\tau = \frac{1}{2}\mathbb{E}[X_{\max}]$, then the exact same guarantee holds. Thus, we have shown that two very natural and simple thresholding rules obtain expected reward equal to at least half the expected offline optimal. However, the drawback of those strategies is that they require full knowledge of the underlying distributions $\mathcal{D}^{(1)}, \dots, \mathcal{D}^{(n)}$ in order to calculate τ (or, at least, they demand a large number of samples from each distribution so that we can estimate the true value of τ with high confidence). To address this issue, in [27] a remarkably simple single-sample algorithm was analysed and shown to be optimal.

Algorithm 3.1: Single-Choice Prophet Inequality, Single-Sample

```
1 Draw one sample $\tilde{X}_i \sim \mathcal{D}^{(i)}$ from each distribution
2 Set threshold $\tau := \max\{\tilde{X}_1, \dots, \tilde{X}_n\}$
3 Pick the first value value above $\tau$
```

Theorem 3.1.2. *Algorithm 3.1 is $\frac{1}{2}$-competitive.*

Proof. For simplicity, we assume that each distribution is continuous, but the proof is easily extended in the case of discrete random variables. The analysis will rely on the principle of deferred decisions. We can consider that the n samples $\tilde{X}_1, \dots, \tilde{X}_n$ and the n realised values $X_1, \dots, X_n$ are not drawn one after the other, but instead are generated via the following process:

1. Draw 2 samples (Y_i, Z_i) from each distribution $\mathcal{D}^{(i)}$ to obtain $2n$ independently drawn samples total.

2. Without loss of generality, relabel the samples so that $Y_i > Z_i$

3. Independently flip n fair coins. If coin i is heads, set $X_i := Y_i$ and $\tilde{X}_i := Z_i$, otherwise set $X_i := Z_i$ and $\tilde{X}_i := Y_i$.

Fix the drawn samples $Y_1, \ldots, Y_n, Z_1, \ldots, Z_n$, sort them in descending order and relabel them as $W_1 > W_2 > \ldots > W_{2n}$. If W_j corresponds to Y_i or Z_i, we denote $\text{index}(W_j) = i$. Define the ***pivotal index*** j^* as the index of the first Z random variable in the above order, that is, there are exactly $j^* - 1$ random variables of type Y exceeding the largest Z random variable.

For each $W_1, \ldots, W_{j^*-1}$, let C_j denote the outcome of the coin flip for $\text{index}(W_j)$. For any $i, j < j^*$ with $i \neq j$ we have that $\text{index}(W_i) \neq \text{index}(W_j)$, hence the random variables $C_1, \ldots, C_{j^*-1}$ are independent. Also, C_{j^*} is deterministic conditioned on $C_1, \ldots, C_{j^*-1}$, since it is the exact same coin flip as one of the earlier indices (by definition of j^*). These two observations will be critical in our analysis.

Having established the above, we will first analyse the expected offline optimal reward, which is equal to $\max_i\{X_i\}$. For each $j < j^*$, we have that W_j is equal to $\max_i\{X_i\}$ iff C_j is heads and C_ℓ is tails for each $\ell < j$, which happens with probability precisely $1/2^j$. For j^*, we have that W_{j^*} is equal to $\max_i\{X_i\}$ iff C_ℓ is tails for each $\ell < j^*$, which happens with probability $1/2^{j^*-1}$. Since either there will be at least one heads among the first $j^* - 1$ coins or all of them will be tails, the expected offline optimal is equal to:

$$\sum_{j=1}^{j^*-1} \frac{W_j}{2^j} + \frac{W_{j^*}}{2^{j^*-1}}$$

Continuing, we will analyse the algorithm's expected reward. If C_1 is tails, then we obtain no reward because the threshold is higher than all revealed elements. For $j < j^* - 1$, if all $C_1, \ldots, C_j$ are heads and C_{j+1} is tails, we set the threshold equal to W_{j+1} and obtain value at least W_j. This occurs with probability $1/2^{j+1}$. Otherwise, if C_j is heads for all $j < j^*$, then C_{j^*} will be tails and we will set the threshold equal to W_{j^*}, obtaining value at least W_{j^*-1}. This happens with probability $1/2^{j^*-1}$. Thus the expected value we obtain is at least equal to:

$$\begin{aligned}\sum_{j=1}^{j^*-2} \frac{W_j}{2^{j+1}} + \frac{W_{j^*-1}}{2^{j^*-1}} &\geq \sum_{j=1}^{j^*-1} \frac{W_j}{2^{j+1}} + \frac{W_{j^*}}{2^{j^*}} \\ &= \frac{1}{2}\left(\sum_{j=1}^{j^*-1} \frac{W_j}{2^j} + \frac{W_{j^*}}{2^{j^*-1}}\right)\end{aligned}$$

Combining the above, the claim follows. □

3.1.2 Multiple-Choice Setting

Clearly, the most natural generalisation of the Single-Choice setting for prophet inequalities arises when we are not constrained to picking only one element, but can instead pick up to k values for some $k \geq 1$ known in advance. Here, we wish to be competitive against the expected total value of the k largest elements. We again consider that we are given full knowledge of the distributions $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(n)}$ beforehand.

This problem was first examined in [28]. The algorithm presented is again threshold-based like the previous ones we discussed. In what follows, let S_τ be the set of values that are above some threshold τ in a given realisation of the random variables.

Algorithm 3.2: Multiple-Choice Prophet, Single Threshold

```
1 Set $\delta = O(\sqrt{k \log k})$
2 Set $\tau$ such that $\mathbb{E}[|S_\tau|] = k - \delta$
3 for $i \in [n]$ do
4     Pick $X_i$ if $X_i > \tau$ (until budget $k$ is exhausted)
```

Theorem 3.1.3. *Algorithm 3.2 is* $\left(1 - O\left(\sqrt{\frac{\log k}{k}}\right)\right)$*-competitive.*

Proof. Fix some realisations of the random variables $X_1, \dots, X_n$. Since the random variables are independent and because $\mathbb{E}[|S_\tau|] = k - \delta$, by a Chernoff bound we have:

$$\mathbb{P}[k - 2\delta \leq |S_\tau| \leq k] \leq \frac{3}{k}$$

Thus we have that $k - 2\delta \leq |S_\tau| \leq k$ with high probability. Condition on this event. Then the reward we obtain is:

$$\sum_{i \in S_\tau} X_i = \tau \cdot |S_\tau| + \sum_{i \in S_\tau} (X_i - \tau) \geq (k - 2\delta) \cdot \tau + \sum_{i \in S_\tau} (X_i - \tau)$$

If S^* is the set of the k largest values, then the offline optimal reward is:

$$\textsc{Opt} = \sum_{i \in S^*} X_i = k\tau + \sum_{i \in S^*} (X_i - \tau) \leq k\tau + \sum_{i \in S^*} (X_i - \tau)^+$$

Since $|S_\tau| \leq k$, by the definition of S_τ we accepted each value that exceeded τ, thus:

$$\begin{aligned} \sum_{i \in S_\tau} (X_i - \tau) &= \sum_{i \in S^*} (X_i - \tau)^+ \geq \textsc{Opt} - k\tau \geq \frac{k - 2\delta}{k} (\textsc{Opt} - k\tau) \\ &= \left(1 - \frac{2\delta}{k}\right) \textsc{Opt} - (k - 2\delta)\tau \end{aligned}$$

Combining the above, we obtain:

$$\sum_{i \in S_\tau} X_i \geq \left(1 - \frac{2\delta}{k}\right) \textsc{Opt}$$

and by substituting $\delta = O(\sqrt{k \log k})$ we obtain the result. □

We will now examine another threshold-based algorithm which, however, uses a different threshold for each random variable X_i instead of a single threshold for all of them. The algorithm is essentially a simple modification of the $\frac{1}{2}$-competitive one proposed in [29], which allows us to obtain the same competitive ratio as before.

Algorithm 3.3 requires knowledge of the probability p_i that X_i is among the maximum k elements and of the p_i-th percentile for X_i, in order to set the thresholds τ_i. In order not to pick items too eagerly, the algorithm maintains some probability of outright rejecting an item without even examining it.

Theorem 3.1.4. *Algorithm 3.3 is* $\left(1 - O\left(\sqrt{\frac{\log k}{k}}\right)\right)$*-competitive*

Algorithm 3.3: Multiple-Choice Prophet, Multiple-Threshold

```
1 Let $p_i := \mathbb{P}[X_i \text{ is among } k \text{ max}]$ for each $i \in [n]$
2 Let $\tau_i$ be such that $\mathbb{P}[X_i \geq \tau_i] = p_i$ for each $i \in [n]$
3 Set $\delta = O\left(\sqrt{\frac{\log k}{k}}\right)$
4 for $i \in [n]$ do
5     With probability $\delta$, reject $X_i$
6     Otherwise, pick $X_i$ if $X_i \geq \tau_i$ (until budget $k$ is exhausted)
```

Proof. Let S^* be the k largest elements. We will make use of the following bound on the expected optimal value:

$$\begin{aligned}
\mathbb{E}\left[\sum_{i\in S^*} X_i\right] &= \mathbb{E}\left[\sum_{i\in S^*}(X_i - \tau_i)\right] + \mathbb{E}\left[\sum_{i\in S^*}\tau_i\right] \\
&\leq \mathbb{E}\left[\sum_{i\in S^*}(X_i - \tau_i)^+\right] + \mathbb{E}\left[\sum_{i=1}^{n}\tau_i \mathbf{1}_{\{i\in S^*\}}\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{n}(X_i - \tau_i)^+\right] + \sum_{i=1}^{n}\tau_i p_i \\
&= \sum_{i=1}^{n}\mathbb{E}[X_i - \tau_i \mid X_i \geq \tau_i]\,\mathbb{P}[X_i \geq \tau_i] + \sum_{i=1}^{n}\tau_i p_i \\
&= \sum_{i=1}^{n}\mathbb{E}[X_i \mid X_i \geq \tau_i] p_i
\end{aligned}$$

Continuing, we will bound the expected reward obtained by the algorithm. Since we discard each item with probability at least δ, the expected number of elements we select is at most $(1-\delta)k$. Thus, by a Hoeffding bound (Theorem A.0.1) we have that with probability at least $1 - \exp(\delta^2 k) = 1 - \frac{1}{k}$ we pick less than k elements in total. Conditioned on this event, we will never run out of budget, so we will see all elements and will pick an element X_i iff we do not outright reject it (which happens with probability $1-\delta$) and also $X_i \geq \tau_i$ (which happens with probability p_i). Thus, conditioned on this event, our expected reward is at least $(1-\delta)\sum_{i\in[n]} p_i \,\mathbb{E}[X_i \mid X_i \geq \tau_i]$, and combining the above we obtain the claim. $\square$

It is worth noting that the above algorithm, as it was originally stated, maintains two variables r_i, q_i during its execution, initialised at $r_1 = 1$ and $q_1 = 1/2$ and updated as $r_{i+1} = r_i(1 - p_i q_i)$ and $q_{i+1} = \frac{1}{2kr_{i+1}}$. Instead of outright rejecting each element with the same probability, it instead rejects element X_i with probability q_i. It is easy to see, with mostly the same analysis as before, that this gives us a $\frac{1}{2}$-competitive algorithm.

Thus, we have shown two different algorithms with the same asymptotic guarantees, each of which requires different statistical information about the underlying distributions $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(n)}$. Importantly, their competitive ratios approach 1 as $k \to \infty$. In [28], a lower bound of $1 - O\left(\frac{1}{\sqrt{k}}\right)$ was also shown for the Multiple-Choice Prophet problem. This lower bound was subsequently achieved in [30], where a much more sophisticated $\left(1 - \frac{1}{\sqrt{k+3}}\right)$-competitive algorithm was presented. For $k = 1$, this competitive ratio nicely matches the simple $\frac{1}{2}$ for the Single-Choice setting.

A natural question to ask is whether this optimal competitive ratio can be achieved using only a limited number of samples from each distribution (i.e. without full knowledge of each). This is indeed possible, and we will discuss it briefly in Section 3.3.

3.1.3 Matroid Setting

As in the Secretary setting, the natural extension of the Single-Choice and Multiple-Choice Prophet Inequality problems is the case where each random variable X_i is associated with an element i of a matroid $M = (\mathcal{U}, \mathcal{I})$, and we are allowed to pick elements so long as they do not form a dependent set of the matroid. This problem was initially studied in [32], where a $\frac{1}{2}$-competitive algorithm was presented. The algorithm maintains an adaptive threshold τ_i at each timestep $i \in [n]$, and picks element i iff picking it does not form a dependent set with the previously selected elements and additionally $X_i \geq \tau_i$. As the analysis of the algorithm is rather involved, we will only present part of it.

First, we require some definitions. Consider a sequence of values $w'_1, \dots, w'_n$ drawn from the distributions $\mathcal{D}^{(1)}, \dots, \mathcal{D}^{(n)}$, and let the maximum-weight basis of the matroid for those realised values be B. For an independent set A of the matroid, we define the *remainder* $R(A)$ of A to be the maximum-weight subset of B such that $A \cup R(A) \in \mathcal{I}$, and we define the *cost* $C(A) := B \setminus R(A)$ of A to be the remaining elements of B. We also denote $w'(R(A))$ and $w'(C(A))$ the corresponding values of those sets.

By the exchange property of matroids, it is easy to see that $A \cup R(A)$ is a basis of the matroid, and indeed it is the maximum-weight basis that contains A. In fact, $A \cup R(A)$ can be constructed by greedily adding to A the maximum-weight element of B that does not cause A to become dependent.

Definition 3.1.1. For a parameter $\alpha > 0$, a threshold-based algorithm for the Matroid Prophet Inequality problem has *α-balanced thresholds* if it has the following property. For any input sequence, if A is the set of elements the algorithm outputs, B is a set disjoint from A such that $A \cup B \in \mathcal{I}$ and $\tau_1, \dots, \tau_n$ are the thresholds set by the algorithm at each timestep $i \in [n]$, the following holds:

$$\sum_{x_i \in A} \tau_i \geq \frac{1}{\alpha} \cdot \mathbb{E}[w'(C(A))] \tag{3.1}$$

$$\sum_{x_i \in B} \tau_i \leq \left(1 - \frac{1}{\alpha}\right) \cdot \mathbb{E}[w'(R(A))] \tag{3.2}$$

where the expectation is over $w' \sim \prod_{i \in [n]} \mathcal{D}^{(i)}$.

The algorithm relies crucially on the following theorem:

Theorem 3.1.5. *If an algorithm has α-balanced thresholds, then it is α-competitive.*

Proof. In what follows, we denote by w the actual realisation of weights seen by the algorithm, w' any realisation of weights drawn from $\prod_{i \in [n]} \mathcal{D}^{(i)}$, and A the set of elements chosen by the algorithm. We have that:

$$\textsc{Opt} = \mathbb{E}[w'(C(A)) + w'(R(A))]$$

since, by definition, $C(A) \cup R(A)$ is a maximum-weight basis with respect to w', and w' has the same distribution as w. We will derive the following three inequalities:

$$\mathbb{E}\left[\sum_{x_i \in A} \tau_i\right] \geq \frac{1}{\alpha}\,\mathbb{E}[w'(C(A))] \tag{3.3}$$

$$\mathbb{E}\left[\sum_{x_i \in A} (w_i - \tau_i)^+\right] \geq \mathbb{E}\left[\sum_{x_i \in R(A)} (w_i' - \tau_i)^+\right] \tag{3.4}$$

$$\mathbb{E}\left[\sum_{x_i \in R(A)} (w_i' - \tau_i)^+\right] \geq \frac{1}{\alpha}\,\mathbb{E}[w'(R(A))] \tag{3.5}$$

Summing (3.3), (3.4), (3.5) and using the fact that if x_i was picked by the algorithm then $w_i \geq \tau$, thus $\tau_i + (w_i - \tau_i)^+ = w_i$ for all $x_i \in A$, we obtain:

$$\mathbb{E}[w(A)] \geq \frac{1}{\alpha}\,\mathbb{E}[w'(C(A))] + \frac{1}{\alpha}\,\mathbb{E}[w'(R(A))]$$

and the claim follows.

(3.3) follows directly from (3.1) in the definition of α-balanced thresholds. (3.4) is derived as follows. First, since the algorithm selects every i such that $w_i \geq \tau_i$, we have that $\sum_{x_i \in A}(w_i - \tau_i)^+ = \sum_{i \in [n]}(w_i - \tau_i)^+$. Second, since the threshold τ_i depends only on $(x_1, w_1), \ldots, (x_{i-1}, w_{i-1})$, the random variables w_i, w_i', τ_i are independent, and because w_i, w_i' are i.i.d. it follows that:

$$\mathbb{E}\left[\sum_{i=1}^{n}(w_i - \tau_i)^+\right] = \mathbb{E}\left[\sum_{i=1}^{n}(w_i' - \tau_i)^+\right] \geq \mathbb{E}\left[\sum_{x_i \in R(A)} (w_i' - \tau_i)^+\right]$$

and (3.4) is established. Finally, by using (3.2) in the definition of α-balanced thresholds and setting $B = R(A)$, we obtain:

$$\begin{aligned}\mathbb{E}\left[\sum_{x_i \in R(A)} w_i'\right] &\leq \mathbb{E}\left[\sum_{x_i \in R(A)} \tau_i\right] + \mathbb{E}\left[\sum_{x_i \in R(A)} (w_i' - \tau_i)^+\right] \\ &\leq \left(1 - \frac{1}{\alpha}\right)\mathbb{E}\left[\sum_{x_i \in R(A)} w_i'\right] + \mathbb{E}\left[\sum_{x_i \in R(A)} (w_i' - \tau_i)^+\right]\end{aligned}$$

where in the first inequality we used the fact that $w_i' \leq \tau_i + (w_i' + \tau_i)^+$. Rearranging, (3.5) follows and the proof is complete. □

Having established that α-balanced thresholds are sufficient for $\frac{1}{\alpha}$-competitiveness, it suffices to devise an algorithm that achieves 2-balanced thresholds. The algorithm presented in [32] is Algorithm 3.4.

For Algorithm 2.5, Eq. (3.1) in the definition of α-balanced thresholds follows from a

Algorithm 3.4: Matroid Prophet, Balanced Thresholds

```
1  Initialise $A_0 := \emptyset$
2  for $i \in [n]$ do
3      Receive element $x_i$
4      Set threshold $\tau_i = \frac{1}{2}\mathbb{E}[w'(C(A_{i-1} \cup \{x_i\})) - w'(C(A_{i-1}))]$
5      if $A_{i-1} \cup \{x_i\} \in \mathcal{I}$ and $w(x_i) \geq \tau_i$ then
6          $A_i = A_{i-1} \cup \{x_i\}$
7      else
8          $A_i = A_{i-1}$
```

telescoping sum.

$$\begin{aligned}\sum_{x_i \in A} \tau_i &= \frac{1}{2} \sum_{x_i \in A} \frac{1}{2} \mathbb{E}[w'(C(A_{i-1} \cup \{x_i\})) - w'(C(A_{i-1}))] \\ &= \frac{1}{2} \sum_{x_i \in A} \frac{1}{2} \mathbb{E}[w'(C(A_{i-1} \cup \{x_i\})) - w'(C(A_{i-1}))] \\ &= \frac{1}{2} \mathbb{E}[w'(C(A_n)) - w'(C(A_0))] = \frac{1}{2} \mathbb{E}[w'(C(A))]\end{aligned}$$

The proof of Eq. (3.2) requires more work, and we will not present it here. The ultimate result is the following:

Theorem 3.1.6. *Algorithm 3.4 has 2-balanced thresholds and is therefore $\frac{1}{2}$-competitive.*

As we have established, we cannot hope to obtain a competitive ratio better that $\frac{1}{2}$ even for 1-uniform matroids, so the above result is in fact optimal for *general* matroids. Notice that this is the first point in this thesis where known results in random-order and prophet settings deviate significantly. Indeed, we have constant-competitive algorithms for both Single-Secretary and Single-Choice Prophet, and we also have $\left(1 - O\left(\frac{1}{\sqrt{k}}\right)\right)$-competitive algorithms for both k-Secretary and k-Choice Prophet. However, while we have an *optimal* constant-competitive algorithm for Matroid Prophet, the best known algorithm for Matroid Secretary is $\frac{1}{\log\log r}$-competitive, where r is the rank of the matroid. This discrepancy is indicative of the fact that, in general terms, the prophet version of a problem is "easier" than the random-order one. We will examine this connection more in Section 3.3.

3.2 Matching Problems

As we did in our discussion of Random-Order models, we now turn our attention to matching problems in graphs in the Prophet setting. In this setting, an adversary forms a graph $G = (V, E)$ and, for each edge $e \in E$, chooses a distribution $\mathcal{D}_e$. Now, there are two different versions of the problem we can consider, based on whether the elements of the graph that arrive on each round are the graph's vertices or its edges. In the first case, the adversary chooses the order in which the vertices will arrive and, whenever a vertex arrives, all of its incident edges are revealed and their weights are drawn from their corresponding distributions. In the second case, the adversary chooses the order in which the edges will arrive and, whenever an edge e arrives, its weight w_e is drawn from $\mathcal{D}_e$. In both cases, we can pick an edge only on the timestep it is revealed, the set of edges we pick must be a matching in the graph, and our goal is to maximise the expected weight of our matching.

We will begin by examining the simplest problem in the above setting, in which the underlying graph is bipartite and we have vertex arrivals. Thus, there is a graph $G = (L, R; E)$ where the vertices in L corresponds to agents and those in R correspond to items. The agents in L arrive sequentially, revealing their incident edges as they do. We assume that we are given full knowledge of the graph structure and of the distributions $\mathcal{D}_e$. The algorithm we will examine for this setting is due to [59]. In what follows, we denote by $w(v, u)$ the weight of edge (v, u) under the weight function w, and we denote by $N(v)$ the set of neighbours of $v \in L$ that are available when v arrives. Also, for any item $u \in R$, we denote by $\mu^*_w(u) \in L$ the agent to which u is matched in the optimal offline matching under the weight function w.

Algorithm 3.5: Prophet Bipartite Matching, Vertex Arrival

```
1 Set thresholds $\tau_u = \frac{1}{2}\mathbb{E}[w(\mu^*_w(u), u)]$ for all $u \in R$
2 for $v \in L$ do
3     if $\exists u \in N(v)$ such that $w(v, u) - \tau_u > 0$ then
4         Match $v$ to $\arg\max_{u \in N(v)}\{w(v, u) - \tau_u\}$
```

Theorem 3.2.1. *Algorithm 3.5 is $\frac{1}{2}$-competitive.*

Proof. Fix an agent $v \in L$. For any $u \in R$, denote by A_{vu} the event that u is available when v arrives and by M_{vu} the event that $u = \arg\max_{u \in N(v)}\{w(v, u) - \tau_u\}$ when v arrives. Our expected reward due to v is:

$$\begin{aligned}&\sum_{u \in R} \mathbb{E}\left[w(v, u)\mathbf{1}_{\{A_{vu}\}}\mathbf{1}_{\{w(v,u)>\tau_u\}}\mathbf{1}_{\{M_{vu}\}}\right] \\ &= \sum_{u \in R} \tau_u \cdot \mathbb{P}[A_{vu}] \cdot \mathbb{E}\left[\mathbf{1}_{\{w(v,u)>\tau_u\}}\mathbf{1}_{\{M_{vu}\}}\right] + \sum_{u \in R} \mathbb{E}\left[(w(v, u) - \tau_u)^+ \cdot \mathbf{1}_{\{A_{vu}\}} \cdot \mathbf{1}_{\{M_{vu}\}}\right]\end{aligned}$$

By the definition of M_{vu} and linearity of expectation, the only surviving term in the second sum is the one corresponding to the $u \in N(v)$ that maximises $w(v, u) - \tau_u$. Therefore, if we denote by T_{vu} the event that v is matched to u in the optimal offline matching, we have that:

$$\begin{aligned}\sum_{u \in R} \mathbb{E}\left[(w(v, u) - \tau_u)^+ \cdot \mathbf{1}_{\{A_{vu}\}} \cdot \mathbf{1}_{\{M_{vu}\}}\right] &\geq \sum_{u \in R} \mathbb{E}\left[(w(v, u) - \tau_u)^+ \cdot \mathbf{1}_{\{A_{vu}\}} \cdot \mathbf{1}_{\{T_{vu}\}}\right] \\ &= \sum_{u \in R} \mathbb{P}[A_{vu}] \cdot \mathbb{E}\left[(w(v, u) - \tau_u)^+ \cdot \mathbf{1}_{\{T_{vu}\}}\right]\end{aligned}$$

Combining the above and summing over all $v \in L$, we have that our total expected reward is at least:

$$\sum_{u \in R} \tau_u \cdot \sum_{v \in L} \mathbb{P}[A_{vu}] \cdot \mathbb{E}\left[\mathbf{1}_{\{w(v,u)>\tau_u\}}\mathbf{1}_{\{M_{vu}\}}\right] + \sum_{u \in R}\sum_{v \in L} \mathbb{P}[A_{vu}] \cdot \mathbb{E}\left[(w(v, u) - \tau_u)^+ \cdot \mathbf{1}_{\{T_{vu}\}}\right]$$

We will examine each of the two terms separately. For the first term, notice that the sum $\sum_{v \in L} \mathbb{P}[A_{vu}] \cdot \mathbb{E}[\mathbf{1}_{\{w(v,u)>\tau_u\}}\mathbf{1}_{\{M_{vu}\}}]$ is precisely the probability that item u will be matched during the execution of the algorithm. Denoting this probability by q_u, we have that the first term equals $\sum_{u\in R} \tau_u \cdot q_u$. For the second term, notice that if u is not matched at all then A_{vu} holds, thus $\mathbb{P}[A_{vu}] \geq 1 - q_u$, and we can therefore lower bound the second term by:

$$\begin{aligned}
&\sum_{u\in R}(1-q_u)\sum_{v\in L}\mathbb{E}\big[(w(v,u)-\tau_u)^+ \cdot \mathbf{1}_{\{T_{vu}\}}\big] \\
&\geq \sum_{u\in R}(1-q_u)\sum_{v\in L}\mathbb{E}\big[w(v,u)\cdot \mathbf{1}_{\{T_{vu}\}}\big] - \sum_{u\in R}\tau_u\cdot(1-q_u) \\
&= \sum_{u\in R}(1-q_u)\,\mathbb{E}[w(\mu^*_w(u),u)] - \sum_{u\in R}\tau_u\cdot(1-q_u) \\
&= \sum_{u\in R}(1-q_u)\cdot 2\tau_u - \sum_{u\in R}\tau_u\cdot(1-q_u) = \sum_{u\in R}\tau_u\cdot(1-q_u)
\end{aligned}$$

where in the last line we used the definition of τ_u. Combining the above, we get that our expected reward is at least $\sum_u \tau_u$, which by the definition of τ_u is at least equal to $\frac{1}{2}\,\mathbb{E}[\textsc{Opt}]$. □

Thus, we have arrived at an optimal algorithm for the Prophet Bipartite Matching problem in the case of vertex arrivals. In the case where there is only one item, i.e. in the Single-Choice Prophet setting, the algorithm nicely reduces to the median rule. What is interesting in the prophet setting is that we have strong results not only for the bipartite case, but for matchings in general graphs as well, both in the vertex arrival and in the edge arrival setting. Indeed, we have the following theorem due to [33]

Theorem 3.2.2. *For the Prophet Matching problem in general (non-bipartite) graphs, there exist a $\frac{1}{2}$-competitive algorithm for the vertex arrival setting and a 0.337-competitive algorithm for the edge arrival setting.*

The ideas behind those algorithms are rather complex (they rely on the concept of *Online Contention Resolution Schemes*), so we will not present them. However, for the simple special case where each edge e has weight $w_e = x_e$ with probability p_e and $w_e = 0$ otherwise, in the edge arrival setting, the above algorithm takes on a rather simple form. Consider the following linear programme

$$\begin{aligned}
\max \quad & \sum_{e\in E} y_e x_e \\
\text{s.t.} \quad & \sum_{e\ni u} y_e \leq 1 && \forall u \in V \\
& 0 \leq y_e \leq p_e && \forall e \in E
\end{aligned}$$

We say that an edge e is available upon arrival if it can be added to our current matching, i.e. if its endpoints are both unmatched. Then the algorithm for this special case can be stated as follows.

For this special case and the above algorithm, it is easy to show a competitive ratio slightly worse than the best known 0.337.

Theorem 3.2.3. *Algorithm 3.6 is $\frac{1}{3}$-competitive*

Algorithm 3.6: Prophet Matching, Edge Arrival, Special Case

```
1 Solve the above LP to obtain solution $y$
2 for $e \in E$ do
3     Set $a_e := \mathbb{P}[e \text{ is available upon arrival}]$
4     if $w_e = x_e$ and $e$ is available then
5         Pick $e$ with probability $\frac{y_e}{3\alpha_e p_e}$
```

Proof. First, note that if y is the optimal solution to the LP, then the offline optimal matching M^* has value Opt at most $\sum_e y_e x_e$. Indeed, setting y_e equal to the probability that e belongs to the M^* is a feasible solution to the LP (since e can only belong to that matching if $w_e > 0$, which happens with probability p_e), with objective value equal to Opt.

Now, the probability that e is picked upon arrival is:

$$\mathbb{P}[e \text{ is available upon arrival}] \cdot \mathbb{P}[w_e = x_e] \cdot \frac{y_e}{3\alpha_e p_e} = \alpha_e \cdot p_e \cdot \frac{y_e}{3\alpha_e p_e} = \frac{y_e}{3}$$

For the algorithm to be well defined, we need to show that $\frac{y_e}{3\alpha_e p_e} \leq 1$. Since $y_e \leq p_e$ by the LP constraints, it suffices to show $\alpha_e \geq \frac{1}{3}$. Recalling the definition of α_e, by a union bound we have that, if $e = (u, v)$:

$$\alpha_e \geq 1 - \mathbb{P}[u \text{ is already matched}] - \mathbb{P}[v \text{ is already matched}]$$

Note that:

$$\mathbb{P}[u \text{ is already matched}] = \sum_{e' < e : u \in e'} \mathbb{P}[e' \text{ is picked}] = \sum_{e' < e : u \in e'} \frac{y'_e}{3} \leq \frac{1}{3}$$

where we used the LP constraints. Thus we indeed have $\alpha_e \geq \frac{1}{3}$.

Therefore, since each edge is picked with probability $y_e/3$, our expected reward is $\sum_e \frac{y_e}{3} \cdot x_e \geq \frac{1}{3}$Opt. □

3.3 Connections with Random-Order Models

We mentioned in passing previously that, in broad terms, the Random-Order version of a problem is more "difficult" than its Prophet version. This intuition is expanded upon in [31], where it is shown that, for a wide class of problems, if we are given an algorithm for a Random-Order problem that satisfies a simple property, we can use the same algorithm to solve the Prophet version of the problem by drawing only a single sample from the underlying distributions. Formally, we say that an algorithm $\mathcal{A}$ for a Random-Order problem is *order-oblivious* if, for any input sequence $I = (x_1, \ldots, x_n)$, it has the following two-phase structure:

1. In the first phase, $\mathcal{A}$ sets a number k and is given a uniformly random subset $J \subset I$ with $|J| = k$. The algorithm cannot pick any items during this phase (it can only use this information in the second phase).
2. In the second phase, the remaining $n - k$ items from $I \setminus J$ are presented to the $\mathcal{A}$. The algorithm must maintain its competitive ratio even if those items are presented in an *adversarial* order.

Note that the first property pertains to the algorithm's operation, while the second one concerns its analysis. The following simple theorem shows how an order-oblivious algorithm for a Secretary problem can be used to tackle the Prophet problem as well:

Theorem 3.3.1. *If $\mathcal{A}$ is an α-competitive order-oblivious algorithm for the Matroid Secretary problem, then there exists a single-sample α-competitive algorithm $\mathcal{P}_{\mathcal{A}}$ for the Matroid Prophet problem.*

Proof. Algorithm $\mathcal{P}_{\mathcal{A}}$ draws a sample $s \sim \prod_{i\in[n]} \mathcal{D}^{(i)}$, shuffles it into a random permutation $(s_{j_1}, \dots, s_{j_n})$, takes the first k elements of the permutation $s_{j_1}, \dots, s_{j_k}$ and passes them to the order-oblivious algorithm $\mathcal{A}$ for its first phase. After that, as the actual input sequence $(x_1, \dots, x_n)$ is revealed, any x_i with $i \in \{j_1, \dots, j_k\}$ is discarded, while all other x_i are presented to $\mathcal{A}$ for its second phase (in an arbitrary order). Since $\mathcal{A}$ is order oblivious and $\{j_1, \dots, j_k\}$ were chosen uniformly at random among the n indices, $\mathcal{A}$ outputs a set of expected weight at least $\alpha \cdot \textsc{Opt}(x)$, where $\textsc{Opt}(x)$ is the optimal set under realised values $x_1, \dots, x_n$. Since this holds for any realisation of the input values x, it also holds in expectation over $\prod_{i\in[n]} \mathcal{D}^{(i)}$, thus the expected reward of $\mathcal{P}_{\mathcal{A}}$ is at least $\alpha \cdot \mathbb{E}[\textsc{Opt}]$. □

Thus, the single-sample Matroid Prophet problem reduces to the order-oblivious Matroid Secretary problem. As is explained in [31], this implies the existence of single-sample constant-competitive algorithms for the Matroid Prophet problem for graphic, transversal and laminar matroids, as well as for general matroids with i.i.d. weights. Additionally, in the same paper, a $\left(1 - O\left(\frac{1}{\sqrt{k}}\right)\right)$-competitive order-oblivious algorithm is presented for the k-Secretary problem and a $\frac{1}{6.75}$-competitive order-oblivious algorithm is given for the constant-degree Random-Order Bipartite Matching problem. Those algorithms imply the existence of single-sample algorithms for those problems in the Prophet setting, with the same competitive ratio guarantees.

In Section 4.4, we will see how essentially the same idea can be applied to use an algorithm for the Random-Order version of an Online Augmentable Integer Programme in order to be competitive in the Prophet version of that problem.

Chapter 4

Random Order and Prophet AIPs

At this point, we have surveyed a number of fundamental results and interesting ideas for online problems in the Random-Order and Secretary models. We have mostly considered maximisation problems, where we were presented a sequence of elements and wanted to pick a maximum-value subset of them, while being in some way constrained in how many elements we can pick. We will now examine some recent results in the Random-Order and Secretary settings concerning a wide class of *minimisation* problems, namely, the class of *Augmentable Integer Programmes* (AIPs) we described in Section 1.1.5. Specifically, we will focus on the Set Cover, Facility Location and Steiner Tree problems, which fall under the category of AIPs, and we will examine how lower bounds for the fully adversarial model can be circumvented under the Random-Order and Prophet settings.

4.1 Random Order Set Cover

The Set Cover problem is one of the most prominent and widely studied combinatorial optimisation problems. In the classical offline version of the problem, we are given a universe $\mathcal{U}$ of N items, a collection $\mathcal{S}$ of m subsets of $\mathcal{U}$ and a cost function $c : \mathcal{S} \to \mathbb{R}_{\geq 0}$. We are also given a subset of n items $U \subseteq \mathcal{U}$. We are tasked with finding a minimum-cost collection of sets in $\mathcal{S}$, such that the union of the chosen sets contains all elements in U. It is well known that this problem is NP-hard, that a simple greedy algorithm achieves an approximation ratio of $O(\log n)$, and that this is the best approximation possible unless P = NP. A natural LP relaxation for this problem is the following:

$$\begin{aligned} \min \quad & \sum_{S \in \mathcal{S}} w_S x_S & & \\ \text{s.t.} \quad & \sum_{S \ni e} x_S \geq 1 & & \forall e \in U \\ & x_S \geq 0 & & \forall S \in \mathcal{S} \end{aligned} \tag{LP-SC}$$

In the *Online Set Cover* problem, we are again given a set system $(\mathcal{U}, \mathcal{S})$, however the (adversarially chosen) elements of U are revealed one-by-one in an online fashion. Whenever an element e arrives, if we have not picked a set containing e in a previous round, we must immediately pick such a set and incur its cost. In terms of the above LP, we can analogously think that, whenever an element e arrives, its corresponding LP constraint $\sum_{S \ni e} x_S \geq 1$ becomes known to us, and we must satisfy it. We are not allowed to drop sets we have previously picked (analogously, we can only increase the values of the LP variables x_S), and our goal is again to incur as low a cost as possible. This problem was initially studied in [2], where an $O(\log m \log n)$-competitive algorithm was given. This result was later shown to be tight in [35]. The algorithm maintains a feasible fractional solution to the LP at all timesteps, and uses randomised rounding in an online fashion to form integral solutions. Essentially the same algorithm was examined in [3], this time from a primal-dual perspective.

Recently, in [34], the problem was examined in the Random-Order setting, Here, after the adversary chooses U, the elements of U are presented in a uniformly random order to our algorithm. It turns out that this weakening of the adversary's power is enough to allow us to obtain a significantly improved competitive ratio of $O(\log mn)$. The main idea behind the algorithm is that we can exploit the randomness in the arrival order in order to *learn* about the underlying set system (recall how we also used the random arrival order in the Secretary problems to "learn" good thresholds).

In all that follows, we will assume that we know a bound β to the cost of the optimal solution, such that $c(\textsc{Opt}) \leq \beta \leq 2 \cdot c(\textsc{Opt})$. We can argue that we know such a bound by using a "guess-and-double" procedure. Concretely, we can initially guess $\beta = \min_{S \in \mathcal{S}} x_S$, and then run our algorithm using this bound for the cost of the optimal solution. If our guess is correct, the cost of our algorithm will always be $O(\beta \log mn)$. Thus, if during the run of our algorithm the total cost we have incurred exceeds $\Theta(\beta \log mn)$, we can "forget" about all sets chosen so far, update the value of β by doubling it and continue on. By "forgetting" we mean that we may "buy" again a set we have previously bought. Therefore, the last time we double our bound and set it equal to β, the total cost we will have paid up to that point is $\left(\frac{\beta}{2} + \frac{\beta}{4} + \ldots + 1\right) \cdot O(\log mn) = O(\beta \log mn)$, and our algorithm guarantees that we will pay an additional $O(\beta \log mn)$ from that point on. Hence, we can use this approach to obtain a β such that $c(\textsc{Opt}) \leq \beta \leq 2 \cdot c(\textsc{Opt})$, at the cost of an additional factor of 2, which is hidden in the asymptotic notation.

4.1.1 Exponential-Time Algorithm

In order to build intuition, first an exponential-time algorithm is presented for the case where all set costs are equal to one. In what follows, we assume that we know a number $k \in [c(\textsc{Opt}), 2 \cdot c(\textsc{Opt})]$ by "guess-and-double". We denote by $\binom{\mathcal{S}}{k} = \{\mathcal{T} \subseteq \mathcal{S} : |\mathcal{T}| = k\}$ the collection of all k-tuples of sets in $\mathcal{S}$.

Algorithm 4.1: LearnOrCoverSimple

```
1 Initialise $\mathfrak{T}^0 \leftarrow \binom{\mathcal{S}}{k}$ and $\mathcal{C}^0 \leftarrow \emptyset$
2 for $t \in [n]$ do
3     $v^t \leftarrow t^{th}$ element in random order
4     if $v^t$ uncovered then
5         Choose $\mathcal{T} \sim \mathfrak{T}^{t-1}$ uniformly at random
6         Choose $T \sim \mathcal{T}$ uniformly at random
7         Add $\mathcal{C}^t \leftarrow \mathcal{C}^{t-1} \cup \{S, T\}$ for any set $S$ containing $v^t$
8     Update $\mathfrak{T}^t \leftarrow \{\mathcal{T} \in \mathfrak{T}^{t-1} : v^t \in \cup_{T \in \mathcal{T}} T\}$
```

In essence, Algorithm 4.1 maintains a list $\mathfrak{T}$ of candidate solutions of k sets (initially containing all possible k-tuples). On each round, it randomly samples a set from those candidates and afterwards eliminates all candidates that cannot be a solution to the problem instance. As we will see in the proof, the reason this algorithm works is that, intuitively, on each round it either makes progress *learning* (i.e. it eliminates many candidates) or *covering* (i.e. it covers many unseen elements).

Theorem 4.1.1. *For Random-Order Set Cover with unit set costs, Algorithm 4.1 is $O(\log mn)$-competitive.*

Proof. Denote by U^t the set of elements that are uncovered at the end of timestep $t \in [n]$. Fix any timestep t where the element v^t is uncovered on arrival. Before the algorithm takes action, there are two cases:

Case 1 (Cover Step): At least half the tuples in $\mathfrak{T}$ cover at least half of the as-of-yet-uncovered elements U^{t-1}. In this case, we pick one of those tuples with probability at least $\frac{1}{2}$, so if $\mathcal{T}$ is the picked tuple we have $\mathbb{E}[|(\cup_{T\in\mathcal{T}} T) \cap U^{t-1}|] \geq \frac{|U^{t-1}|}{4}$, and because we pick a set $T \in \mathcal{T}$ uniformly at random we have $\mathbb{E}[|T \cap U^{t-1}|] \geq \frac{|U^{t-1}|}{4k}$. That is, we cover at least $\frac{1}{4k}$ of the uncovered elements on expectation.

Case 2 (Learn Step): At least half of the tuples in $\mathfrak{T}$ cover less than half of the uncovered elements U^{t-1}. Due to the random order of arrivals, for each of those tuples, the probability that an element not covered by the tuple arrives on round t (thus the tuple is removed from $\mathfrak{T}$) is at least $\frac{1}{2}$. Thus we eliminate at least $\frac{1}{4}$ of the remaining candidates on expectation.

Since the total number of elements is n, and we cover at least $\frac{1}{4k}$ of uncovered elements on expectation on each Cover Step, it follows that after $10k \log n$ Cover Steps there will be at most $n\left(1 - \frac{1}{4k}\right)^{10\log n} < 1$ uncovered elements left on expectation, so we will not buy another set. Also, since initially $|\mathfrak{T}^0| = \binom{m}{k} \leq m^k$ and we eliminate at least $\frac{1}{4}$ of candidates on expectation on each Learn Step, it follows that after $10k \log m$ Learn Steps there will be at most $m^k \left(\frac{3}{4}\right)^{10k\log m} \leq 1$ candidates left. That is, we will have only the optimal solution remaining and we will buy at most k more sets. Overall, after $10k \log m + 10k \log n$ rounds where uncovered elements arrived, we will have bought at most $20k \log mn$ sets and we will have performed either $10k \log m$ Cover Steps or $10k \log n$ Learn Steps, so we will need to buy at most k additional sets on expectation. Thus, our total expected cost is $O(k \log mn)$ and the claim follows by the definition of k. $\square$

Of course, we still need to present an algorithm that runs in polynomial time and that works for general set costs. However, the intuition behind Algorithm 4.1 will be very useful in understanding the analysis of the next algorithm.

4.1.2 Polynomial-Time Algorithm

In order to obtain an efficient algorithm for our problem, we will no longer maintain a set of all possible candidate solutions. Instead, we will maintain a form of probability distribution x over the m sets, from which we will sample at each round. This can be viewed as a form of *dimensionality reduction* when compared to Algorithm 4.1. The "learning" part of our algorithm will thus not be performed by eliminating candidate solutions, but instead by updating x on each round based on the arrived element. These updates will be implemented using a multiplicative weights scheme.

In what follows, we assume that, by "guess-and-double", we know a bound β such that $\text{LP}_{\text{OPT}} \leq \beta \leq 2 \cdot \text{LP}_{\text{OPT}}$, where LP_{OPT} denotes the cost of the optimal LP solution to the final unknown instance. We also define $\kappa_v := \min\{c_S : S \ni v\}$ to be the cost of the cheapest set covering element v.

The above algorithm maintains a fractional vector x which is a guess for the LP solution of cost β to the set cover instance. Two crucial differences from the algorithm of [2] for

Algorithm 4.2: LEARNORCOVER

```
1  Let $m' \leftarrow |\{S : c_S \leq \beta\}|$
2  Initialise $x_S^0 \leftarrow \frac{\beta}{c_S \cdot m'}$ for all sets $S$
3  for $t \in [n]$ do
4      $v^t \leftarrow t^{th}$ element in random order
5      if $v^t$ uncovered then
6          For each set $S$, pick $S$ with probability $\min(\kappa_{v^t} \cdot x_S^{t-1}/\beta, 1)$
7          if $\sum_{S \ni v^t} x_S^{t-1} < 1$ then
8              For every set $S$, update $x_S^t \leftarrow x_S^{t-1} \cdot \exp\{\mathbf{1}_{\{S \ni v^t\}} \cdot \kappa_{v^t}/c_S\}$
9              Let $Z^t = \langle c, x^t \rangle/\beta$ and normalise $x^t \leftarrow x^t/Z^t$
10         else
11             $x^t \leftarrow x^{t-1}$
12         Pick the cheapest set containing $v^t$
```

non-random-order Online Set Cover is that, here, the fractional vector maintained always has the same cost β, and additionally it is not required that this fractional vector be a *feasible* solution to the LP. This vector x is used to sample sets on each round, and whenever an uncovered element v^t arrives we increase the value x_S of all $S \ni v^t$ and renormalise, so that $\langle c, x \rangle = \beta$. That is, the algorithm maintains the following invariant:

Invariant 1. For all rounds t, it holds that $\langle c, x^t \rangle = \beta$.

The last line of the algorithm greedily buys a backup set to ensure that the revealed element is covered. Note that Algorithm 4.2 scales its sampling and learning rates with κ_{v^t} each round. This is done to ensure that the expected cost of the sets bought by sampling is the same as the cost of the backup set, and that those costs are offset by the change in a *potential function* which we will shortly define and which will be crucial for the analysis.

We will start defining some notation. We will need the following weighted generalisation of the KL divergence.

$$\mathrm{KL}_c(x \parallel y) := \sum_{i=1}^{n} c_i \left[x_i \log \frac{x_i}{y_i} - x_i + y_i \right]$$

For any x, y we have that $\mathrm{KL}_c(x \parallel y) \geq 0$. This follows directly from the above definition by using the inequality $\log x \geq 1 - \frac{1}{x}$.

We denote by x^* the optimal LP solution to the final, unknown set cover instance. We also denote by $X_v^t := \sum_{S \ni v} x_S^t$ the "fractional coverage" provided to element v by our current solution x^t, and by U^t the set of elements remaining uncovered at the end of round t (so U^0 is the set of all elements). Lastly, we denote by $\rho^t := \sum_{u \in U^t} \kappa_u$ the sum of the minimum coverage costs for all uncovered elements at round t.

We are now ready to define potential function which will measure the progress we make either learning or covering at each round, and which will be the main tool for our analysis. First, we define the *learning* and *covering* potential functions as follows:

$$\Phi_L(t) := \mathrm{KL}_c(x^* \parallel x^t)$$

$$\Phi_C(t) := \beta \cdot \log \frac{\rho^t}{\beta}$$

Then our overall potential function is defined as:

$$\Phi(t) := C_1 \cdot \Phi_L(t) + C_2 \cdot \Phi_C(t)$$

where C_1, C_2 are constants to be defined later. Intuitively, we can think that the learning potential measures the distance of our current solution x^t from the optimal solution x^*, while the covering potential measures (the logarithm of) how many uncovered elements are remaining. Thus, we can think of the first term decreasing as the algorithm making progress in *learning* and of the second term decreasing as the algorithm making progress in *covering*. What we will show is that, on each round when an uncovered element arrives, either the first or the second part of the potential decreases noticeably; that is, the algorithm makes progress either learning or covering. This is precisely the intuition we have obtained from the exponential-time Algorithm 4.1.

More formally, define the event $\Upsilon^t := \{v^t \in U^{t-1}\}$ that the element v^t is uncovered on arrival. Note that when Υ^t does not hold the algorithm does not take any action and the potential does not change. For rounds when Υ^t holds, denote by $\mathcal{R}^t$ the sets sampled by the algorithm (in line 6). We will show the following two lemmas.

Lemma 4.1.2 (Learning Potential). *For rounds t when Υ^t holds, the expected change in the learning potential is bounded by:*

$$\mathop{\mathbb{E}}_{v^t,\mathcal{R}^t}[\Phi_L(t) - \Phi_L(t-1) \mid x^{t-1}, U^{t-1}, \Upsilon^t] \leq \mathop{\mathbb{E}}_{v\sim U^{t-1}}[(e-1)\kappa_v \min(X_v^{t-1}, 1) - \kappa_v]$$

Lemma 4.1.3 (Covering Potential). *For rounds t when Υ^t holds, the expected change in the covering potential is bounded by:*

$$\mathop{\mathbb{E}}_{v^t,\mathcal{R}^t}[\Phi_C(t) - \Phi_C(t-1) \mid x^{t-1}, U^{t-1}, \Upsilon^t] \leq -(1-e^{-1}) \cdot \mathop{\mathbb{E}}_{u\sim U^{t-1}}[\kappa_u \cdot \min(X_u^{t-1}, 1)]$$

By $v \sim U^{t-1}$ we mean that v is drawn uniformly at random among the uncovered elements U^{t-1}. On a high level, the above lemmas tell us that if $\mathbb{E}_{u\sim U^{t-1}}[\kappa_u \cdot \min(X_u^{t-1}, 1)]$ is "large" (intuitively, if most uncovered elements have high fractional coverage from x^t) then the covering potential will decrease "noticably", otherwise the learning potential will do so. Again we see the usefulness of the intuition from Algorithm 4.1; in that algorithm, the case of the "Learn Step" (where most candidate solutions cover most uncovered elements) can be thought of as corresponding to the case where the learning potential decreases in our algorithm (where our current solution gives high fractional coverage to most uncovered elements). Similarly the "Cover Step" can be thought of as corresponding to the case where the covering potential decreases.

We will defer the proof of the above lemmas for a bit later. We will first show how those lemmas can be combined to show the competitiveness of Algorithm 4.2. To do so, we will need an additional lemma showing the value of the initial potential:

Lemma 4.1.4 (Initial Potential). *The initial potential is bounded as $\Phi(0) = O(\beta \log mn)$.*

Using the above three lemmas, we can prove the following main theorem:

Theorem 4.1.5. *Algorithm 4.2 is $O(\log mn)$-competitive for Random-Order Set Cover.*

Proof. In every round t in which Υ^t holds, the expected cost of the sampled sets $\mathcal{R}^t$ is $\kappa_{v^t} \cdot \langle c, x^{t-1}\rangle/\beta = \kappa_{v^t}$ by Invariant 1. The algorithm pays an additional κ_{v^t} to greedily cover the v^t at the last step of round t, so its total expected cost per round is $2 \cdot \kappa_{v^t}$.

Combining Lemmas 4.1.2 and 4.1.3 and setting the constants $C_1 = 2$ and $C_2 = 2e$ in Φ, we have that the expected decrease in potential each round is at least:

$$\begin{aligned}\mathop{\mathbb{E}}_{v^t,\mathcal{R}^t}[\Phi(t) - \Phi(t-1) \mid v^1, \ldots, v^{t-1}, \mathcal{R}^1, \ldots, \mathcal{R}^{t-1}, \Upsilon^t] \\ \leq - \mathop{\mathbb{E}}_{v^t,\mathcal{R}^t}[2 \cdot \kappa_{v^t} \mid v^1, \ldots, v^{t-1}, \mathcal{R}^1, \ldots, \mathcal{R}^{t-1}, \Upsilon^t]\end{aligned}$$

which offsets the expected change in the algorithm's cost. Since neither the potential Φ nor the total cost of the algorithm change during rounds in which Υ^t does not hold, we have the inequality:

$$\mathop{\mathbb{E}}_{v^t,\mathcal{R}^t}[\Phi(t) - \Phi(t-1) + c(\textsc{Alg}(t)) - c(\textsc{Alg}(t-1)) \mid v^1, \ldots, v^{t-1}, \mathcal{R}^1, \ldots, \mathcal{R}^{t-1}] \leq 0$$

Let t^* be the last round in which $\Phi(t^*) \geq 0$. By repeatedly applying the above inequality for all $t \in [1, t^*]$ we obtain:

$$\mathop{\mathbb{E}}_{v^t,\mathcal{R}^t}[\Phi(t^*) - \Phi(0) + c(\textsc{Alg}(t^*)) - c(\textsc{Alg}(0))] \leq 0$$

Since $c(\textsc{Alg}(0)) = 0$ and $\Phi(t^*) \geq 0$, this gives us:

$$\mathop{\mathbb{E}}_{v^t,\mathcal{R}^t}[c(\textsc{Alg}(t^*))] \leq \Phi(0) \leq O(\beta \log mn)$$

where we used Lemma 4.1.4.

It remains to bound the expected cost paid by the algorithm after round t^*. Since the KL divergence is nonnegative, Φ is negative only if $\rho^t \leq \beta$. Since the algorithm pays $O(\kappa_{v^t})$ in expectation only during rounds t where $v^t \in U^{t-1}$ and pays 0 otherwise, its expected cost after t^* is at most $\sum_{v \in U^{t^*}} \kappa_v = \rho^{t^*} = O(\beta)$. □

We will now prove the three lemmas that lead to the above theorem. We start by showing the bound on the initial potential.

Proof of Lemma 4.1.4. First, we will show that there exists an optimal fractional solution to the LP corresponding to the final Set Cover instance which is supported only on sets with cost at most β. Suppose otherwise and let x^* be an optimal LP solution. Let T be a set with cost $c_T > \beta$ and $x^*_T > 0$. Clearly $x^*_T < 1$, otherwise the cost of x^* would exceed β, which contradicts the definition of β. Define a new vector x by:

$$x_S = \begin{cases} 0 & \text{if } S = T \\ \dfrac{x^*_S}{1 - x^*_T} & \text{otherwise} \end{cases}$$

For any element e:

$$\sum_{S \ni e} x_S = \frac{1}{1 - x^*_T}\left(\sum_{S \ni e} x^*_S - x^*_T \cdot \mathbf{1}_{\{T \ni e\}}\right) \geq \frac{1}{1 - x^*_T}\left(1 - x^*_T \cdot \mathbf{1}_{\{T \ni e\}}\right) \geq 1$$

where the first inequality uses the feasibility of x^* and the second uses $x^*_T < 1$. Thus x is a feasible solution to the LP. The cost of x is:

$$\langle c, x \rangle = \frac{\langle c, x^* \rangle - c_T x^*_T}{1 - x^*_T} < \frac{\langle c, x^* \rangle - c_T \langle c, x^* \rangle}{1 - x^*_T} = \langle c, x^* \rangle$$

where we used $c_T > \beta \geq \langle c, x^* \rangle$. Thus x is a feasible solution with cost lower than x^*, which contradicts the optimality of x^*.

Having shown the above, we may assume w.l.o.g. that the optimal solution x^* in the potential is only supported on sets with cost at most β, thus $\text{support}(x^*) \subseteq \text{support}(x^0)$. Recalling the initialisation of x^0, we can bound the initial learning potential by:

$$\Phi_L(0) = \mathrm{KL}_c(x^* \parallel x^0) = \sum_{S: c_S \leq \beta} c_S x_S^* \log\left(x_S^* \frac{c_S \cdot m'}{\beta}\right) + \frac{\beta}{m'} - x_S^* \leq \beta(\log m + 1)$$

where we used $\langle c, x^* \rangle \leq \beta$ and that $m' \leq m$ is the number of sets with cost at most β.

As for the initial covering potential, we have that $\beta \log(\rho^0/\beta) = \beta \log(\sum_{v \in U^0} \kappa_v/\beta) \leq \beta \log |U^0| = \beta \log n$, since the cheapest cost to cover each element e is at most β. Since C_1 and C_2 are constants, the claim follows. □

We continue by proving the bound on the change of the learning potential.

Proof of Lemma 4.1.2. If $X_v^{t-1} \geq 1$, Algorithm 4.2 sets $x^t = x^{t-1}$, so the change in the learning potential is 0, thus:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{v^t, \mathcal{R}^t}[\Phi_L(t) - \Phi_L(t-1) \mid x^{t-1}, U^{t-1}, \Upsilon^t, X_{v^t}^{t-1} < 1] \\
&\leq \mathop{\mathbb{E}}_{v \sim U^{t-1}}[(e-1) \cdot \kappa_v \cdot \min(X_v^{t-1}, 1) - \kappa_v \mid X_{v^t}^{t-1} \geq 1] \qquad (4.1)
\end{aligned}$$

We therefore focus on the case $X_v^{t-1} < 1$.

Notice that the expected change in the learning potential depends only on the arriving uncovered element v^t. Also note that, due to the random arrival order, that element is chosen uniformly at random from U^t. Thus, expanding definitions we have that

$$\begin{aligned}
&\mathop{\mathbb{E}}_{v^t, \mathcal{R}^t}[\Phi_L(t) - \Phi_L(t-1) \mid x^{t-1}, U^{t-1}, \Upsilon^t, X_{v^t}^{t-1} < 1] \\
&= \mathop{\mathbb{E}}_{v \sim U^{t-1}}\left[\sum_S c_S \cdot x_S^* \cdot \log \frac{x_S^{t-1}}{x_S^t} \,\middle|\, X_{v^t}^{t-1} < 1\right] \\
&= \mathop{\mathbb{E}}_{v \sim U^{t-1}}\left[\langle c, x^* \rangle \cdot \log Z^t - \sum_{S \ni v} c_S \cdot x_S^* \cdot \log e^{\kappa_v / c_S} \,\middle|\, X_{v^t}^{t-1} < 1\right] \\
&\leq \mathop{\mathbb{E}}_{v \sim U^{t-1}}\left[\beta \cdot \log\left(\sum_{S \ni v} \frac{c_S}{\beta} \cdot x_S^{t-1} \cdot e^{\kappa_v / c_S} + \sum_{S \not\ni v} \frac{c_S}{\beta} \cdot x_S^{t-1}\right) - \kappa_v \cdot \sum_{S \ni v} x_S^* \,\middle|\, X_{v^t}^{t-1} < 1\right]
\end{aligned}$$

where in the last step we expanded the definition of Z^t and used $\langle c, x^* \rangle \leq \beta$. Since x^* is a feasible solution to the LP, we have that $\sum_{S \ni v} x_S^* \geq 1$. By using this, along with the inequality $e^y \leq 1 + (e-1) \cdot y$ for all $y \in [0, 1]$, we can bound the above by:

$$\leq \mathop{\mathbb{E}}_{v \sim U^{t-1}}\left[\beta \cdot \log\left(\sum_S \frac{c_S}{\beta} \cdot x_S^{t-1} + (e-1) \sum_{S \ni v} \frac{\kappa_v}{\beta} \cdot x_S^{t-1}\right) - \kappa_v \,\middle|\, X_{v^t}^{t-1} < 1\right]$$

Finally, using Invariant 1 along with the inequality $\log(1+y) \leq y$, we can bound the above by:

$$\begin{aligned}
&\leq \mathop{\mathbb{E}}_{v \sim U^{t-1}}[(e-1) \cdot \kappa_v \cdot X_v^{t-1} - \kappa_v \mid X_{v^t}^{t-1} < 1] \\
&\leq \mathop{\mathbb{E}}_{v \sim U^{t-1}}[(e-1) \cdot \kappa_v \cdot \min(X_v^{t-1}, 1) - \kappa_v \mid X_{v^t}^{t-1} < 1] \qquad (4.2)
\end{aligned}$$

Combining (4.1), (4.2) and using the law of total expectation, we obtain the claim. □

Lastly, we will show the bound on the change of the covering potential.

Proof of Lemma 4.1.3. Conditioned on $v^t = v$ for any fixed element v, the expected change in the covering potential depends only on the set $\mathcal{R}^t$ of sampled elements at round t. Hence

$$\begin{aligned}
&\mathop{\mathbb{E}}_{v^t,\mathcal{R}^t}[\Phi_C(t) - \Phi_C(t-1) \mid x^{t-1}, U^{t-1}, \Upsilon^t] \\
&= \mathop{\mathbb{E}}_{\mathcal{R}^t}[\beta(\log\rho^t - \log\rho^{t-1}) \mid x^{t-1}, U^{t-1}, \Upsilon^t, v^t = v] \\
&= \mathop{\mathbb{E}}_{\mathcal{R}^t}\left[\beta\log\left(1 - \frac{\rho^{t-1}-\rho^t}{\rho^{t-1}}\right) \middle| U^{t-1}, v^t = v\right] \\
&\le -\frac{\beta}{\rho^{t-1}}\mathop{\mathbb{E}}_{\mathcal{R}^t}\left[\rho^{t-1} - \rho^t \mid U^{t-1}, v^t = v\right]
\end{aligned}$$

where in the last line we used the inequality $\log(1-y) \le -y$. Expanding the definition of ρ^t we can bound the above by:

$$\begin{aligned}
&\le -\frac{\beta}{\rho^{t-1}}\mathop{\mathbb{E}}_{\mathcal{R}^t}\left[\sum_{u\in U^{t-1}} \kappa_u \mathbf{1}_{\{u\in U^{t-1}\setminus U^t\}} \middle| U^{t-1}, v^t = v\right] \\
&= -\frac{\beta}{\rho^{t-1}}\sum_{u\in U^{t-1}} \kappa_u \mathop{\mathbb{P}}_{\mathcal{R}^t}\left[u \notin U^t \mid u \in U^{t-1}, v^t = v\right] \\
&\le -(1-e^{-1}) \cdot \frac{1}{\rho^{t-1}} \cdot \kappa_v \cdot \sum_{u\in U^{t-1}} \min(X_u^{t-1}, 1) \\
&= -(1-e^{-1}) \cdot \frac{|U^{t-1}|}{\rho^{t-1}} \cdot \kappa_v \cdot \mathop{\mathbb{E}}_{u\sim U^{t-1}}[\kappa_u \min(X_u^{t-1}, 1)] \qquad (4.3)
\end{aligned}$$

The last inequality holds due to the way that $\mathcal{R}$ is formed. That is, each set is sampled independently with probability $\min(\kappa_v x_S^{t-1}/\beta, 1)$, so the probability that any given element $u \in U^{t-1}$ is covered is:

$$\begin{aligned}
1 - \prod_{S\ni u}\left(1 - \min\left(\frac{\kappa_v x_S^{t-1}}{\beta}, 1\right)\right) &\ge 1 - \exp\left\{-\min\left(\frac{\kappa_v}{\beta}X_u^{t-1}, 1\right)\right\} \\
&\ge (1-e^{-1}) \cdot \min\left(\frac{\kappa_v}{\beta}X_u^{t-1}, 1\right) \\
&\ge (1-e^{-1}) \cdot \frac{\kappa_v}{\beta}\min\left(X_u^{t-1}, 1\right)
\end{aligned}$$

where we used the inequalities $1-e^{-y} \ge (1-e^{-1})y$ and $\min(ab,1) \ge a\cdot\min(b,1)$ for $a \le 1$ (recall that $\kappa_v/\beta \le 1$ by the definitions of κ_v and β). Therefore, by taking the expectation of (4.3) over $v^t \sim U^{t-1}$ and using the fact that $\mathbb{E}_{v\sim U^{t-1}}[\kappa_v] = \rho^{t-1}/|U^{t-1}|$, we obtain that the expected change in the covering potential is bounded by

$$\mathop{\mathbb{E}}_{v^t,\mathcal{R}^t}[\Phi_C(t) - \Phi_C(t-1) \mid x^{t-1}, U^{t-1}, \Upsilon^t] \le -(1-e^{-1}) \cdot \mathbb{E}[u \sim U^{t-1}]\kappa_u \min(X_u^{t-1}, 1)$$

as desired. □

Thus, we have completed the proof of the competitive ratio of Algorithm 4.2. It is worth mentioning that this $O(\log mn)$ competitive ratio is nearly tight; indeed, in the same paper ([34]) two lower bounds of $\Omega(\log n)$ and $\Omega\left(\frac{\log m}{\log\log m}\right)$ are shown for the Random-Order Online Set Cover problem. That is, if $m = \text{poly}(n)$ then indeed the algorithm's competitive ratio is optimal, while if m is exponential in n then there is a gap of $\frac{1}{\log\log m}$ between the algorithm's competitive ratio and the above lower bound.

4.2 Random Order Facility Location

Another classical combinatorial optimisation problem is the Facility Location problem. In the offline version of the problem, we are given a set of n clients and m facilities. Each facility f has an *opening cost* c_f, while each facility-client pair (f, v) has a *connection cost* c_{fv}. We can choose to open any number of facilities (paying the opening cost for each), and we must connect each client to exactly one open facility (paying the corresponding connection cost). We wish to minimise our total cost. A natural LP relaxation for this problem is the following

$$\begin{aligned} \min \quad & \sum_f c_f x_f + \sum_{f,v} c_{fv} y_{fv} & \\ \text{s.t.} \quad & \sum_f y_{fv} \geq 1 & \forall v \\ & y_{fv} \leq x_f & \forall f, v \\ & x_f, y_{fv} \geq 0 & \forall f, v \end{aligned} \tag{LP-FL}$$

This problem has received significant attention in the important special case of *Metric* Facility Location. In this setting, we are given a metric space (M, d) where M is a set of points and $d : M \times M \to \mathbb{R}_{\geq 0}$ is a distance function that satisfies the triangle inequality. Each of the clients is a point in M, and we can open a facility at any point in M. The connection cost for a facility-client (f, v) pair is $c_{fv} = d(f, v)$. For the offline version of the problem, various constant-competitive algorithms are known, see for example [60], and there exists a small gap between the optimal known algorithm and the best known lower bound.

In the *online* version of Metric Facility Location, the clients arrive in sequence and must be immediately and irrevocably assigned to a facility upon arrival. A client can only be assigned to an open facility, and a facility cannot be closed once it has been opened. For the adversarial version of this problem, an $O\left(\frac{\log n}{\log\log n}\right)$-competitive algorithm was given in [6], and it was shown that this competitive ratio is tight. The situation is dramatically different in the *random-order* setting, as we will now examine.

4.2.1 Metric Facility Location

The Random-Order Metric Facility Location problem was first examined in [5]. There, a simple algorithm was given that is *constant*-competitive in the random order. This is a drastic improvement over the $\Theta\left(\frac{\log n}{\log\log n}\right)$ competitive ratio for the adversarial case, and once again demonstrates how imposing random-order can significantly reduce the difficulty of a problem. For simplicity, we will only examine the algorithm given in that paper for the case where the opening cost for each facility is the same, equal to f.

Theorem 4.2.1. *Algorithm 4.3 is 8-competitive.*

Proof. Suppose the optimal solution opens k facilities $c_1^*, \ldots, c_k^*$. Let $d_v^* := \min_{i \in [k]} d(v, c_i^*)$ for each client v. Define by C_i^* the set of clients connected to facility c_i^* in the optimum. Also define $A_i^* := \sum_{v \in C_i^*} d_v^*$ and $a_i^* := A_i^* / |C_i^*|$.

Let γ_v be the cost incurred by our algorithm when client v arrives, and denote by δ_v the distance from v to the closest facility we have opened when v arrives. It follows that

Algorithm 4.3: Random-Order Metric Facility Location

```
1  for $t \in [n]$ do
2      $v^t \leftarrow t^{th}$ client in the random order
3      $\delta \leftarrow$ distance from $v^t$ to the closest open facility
4      With probability $\min(\frac{\delta}{f}, 1)$ open a facility at $v^t$
5      Connect $v^t$ to the closest open facility
```

$\mathbb{E}[\gamma_v \mid \delta_v] = \frac{\delta_v}{f} \cdot f + \left(1 - \frac{\delta_v}{f}\right) \delta_v \leq 2\delta_v$, thus:

$$\mathbb{E}[\gamma_v] \leq 2\,\mathbb{E}[\delta_v] \tag{4.4}$$

Thus we only need to bound δ_v for each client. Fix a cluster C_i^*. We will call a client in C_i^* "good" if it is among the $|C_i^*|/2$ closest ones to c_i^* in C_i^*, and we will call it "bad" otherwise. We will analyse the cost for good and bad clients separately.

Lemma 4.2.2. *The total expected cost of good clients $g \in C_i^*$ is bounded by $\mathbb{E}[\sum_g \gamma_g] \leq 2f + 2A_i^* + 2\sum_g d_g^*$, regardless of the order in which the clients arrive.*

Proof. We discern two cases. First, suppose there exists an open facility within distance $2a_i^*$ from c_i^*. Then, by the triangle inequality, for any good client g we have $\delta_g \leq 2a_i^* + d_g^*$, thus $\mathbb{E}[\gamma_g] \leq 2(2a_i^* + d_g^*)$ by (4.4). Summing over the good clients, we have $\mathbb{E}[\sum_g \gamma_g] \leq 2A_i^* + 2\sum_g d_g^*$, since there are $|C_i^*|/2$ good clients.

Now, suppose that such a facility does not exist. Whenever a good client g arrives, it opens a facility with probability δ_g/f, and the connection cost paid if it does not open a facility is δ_g. This implies that the expected cost before opening a facility is f. By Markov's inequality, all good clients are within distance $2a_i^*$ of c_i^*. Hence, we will pay expected cost f before opening a facility, then we will pay an additional f to open a facility within distance $2a_i^*$ of c_i^*, and then we will pay an expected $2A_i^* + 2\sum_g d_g^*$ by the analysis of the first case. □

We now turn our attention to the "bad" clients. Since the above lemma holds regardless of the order of arrival of clients, we can consider that we first create an arbitrary ordering of the good clients and then randomly inject bad clients in that ordering. We will use this viewpoint to bound the cost for the bad clients.

Lemma 4.2.3. *The expected cost of any bad client $b \in C_i^*$ is bounded by $\mathbb{E}[\gamma_b] \leq 2d_b^* + \frac{2}{|C_i^*|}\left[f + \sum_g(\mathbb{E}[\gamma_g] + 2d_g^*)\right]$.*

Proof. By the above viewpoint, with probability at most $\frac{2}{|C_i^*|}$, the bad client b arrives before all good clients, in which case we pay a cost of at most f for this client. Otherwise, condition on the event that when a bad client b arrives, the most recent good client was g. By the above viewpoint, each good client has equal probability $\frac{2}{|C_i^*|}$ to be this client g. Suppose that when b arrives, the nearest facility to c_i^* that we have opened is at distance x from c_i^*. By the triangle inequality we have $\delta_b \leq x + d_b^*$, thus by (4.4) we have $\mathbb{E}[\gamma_b] \leq 2(x + d_b^*)$. Furthermore, when g arrived, the nearest facility to c_i^* was at distance at least x from c_i^*, thus $\mathbb{E}[\gamma_g] \geq 2(x - d_g^*)$. Hence, $\mathbb{E}[\gamma_b] \leq \mathbb{E}[\gamma_g] + 2d_b^* + 2d_g^*$. The lemma follows by simple calculations. □

Now we can use the above two lemmas and simply sum over the expected costs of all good and bad clients in each cluster. This gives us the following bound

$$5f + 4A_i^* + 4\sum_g d_g^* + 2\sum_b d_b^* = 5f + 6A_i^* + 2\sum_g d_g^*$$

where we used the definition of $A_i^* = \sum_g d_g^* + \sum_b d_b^*$. Since the good clients are exactly the $|C_i^*|/2$ closest ones to c_i^*, we have that $\sum_g d_g^* \leq \frac{1}{2}A_i^*$, thus our expected cost for cluster C_i^* is a most $5f + 8A_i^*$. The offline optimal pays $f + A_i$ for each cluster, thus we indeed obtain a competitive ratio of 8. □

While Algorithm 4.3 only works for uniform opening costs, it was shown in the same paper ([5]) that the ideas from this algorithm and its analysis can be used to design a similar, simple algorithm that is 33-competitive in the random-order case for *arbitrary* opening costs. A remarkable fact about Algorithm 4.3 is that it also attains the optimal competitive ratio of $O\left(\frac{\log n}{\log\log n}\right)$ for the fully adversarial setting. Lastly, it should be pointed out that, while Algorithm 4.3 was the first one to tackle the uniform-cost random-order online Facility Location problem, it has since been improved upon, with the algorithm of [36] achieving 3-competitiveness. In this last paper, it is also shown that no algorithm can achieve a competitive ratio better than 2 for this problem.

4.2.2 Non-Metric Facility Location

In the *non-metric* version of Facility Location, the facilities and clients are no longer points in a metric space, and the costs c_{fv} to connect a client v to a facility f can be arbitrary. This problem is significantly harder than the metric case. To begin with, in the offline setting, non-metric Facility Location is equivalent to Set Cover with respect to approximation, in the following sense. An instance of Facility Location can be formulated as an instance of Set Cover and the standard greedy algorithm for Set Cover can be used to obtain a $O(\log n)$ approximation. Also, Set Cover is a special case of Facility Location (when $c_{fv} \in \{0, \infty\}$), thus the lower bound of $\Omega(\log n)$ for Set Cover also holds for Facility Location. Thus, offline non-metric Facility Location is $\Theta(\log n)$-approximable, while the metric case is constant-approximable.

Moving to the online setting, the fully-adversarial case was analysed in [37], where an $O(\log m \log n)$-competitive algorithm was given (m is the number of potential facilities we can open and n is the number of clients). Since Online Set Cover is a special case of Online Facility Location, and because of the $\Omega(\log m \log n)$ lower bound on the competitive ratio of the former, it follows that the above result is tight. Similarly to the Online Set Cover algorithm of [2], the algorithm of [37] maintains a feasible fractional solution to (LP-FL) at each timestep and performs randomised rounding online.

The *random-order* setting for the Online Non-Metric Facility Location problem was only recently examined in [1]. This paper builds upon Algorithm 4.2 of [34] to present an $O(\log mn)$-competitive algorithm for this problem, again improving significantly over the adversarial case.

Before presenting the algorithm, we will need some definitions. Denote by $\mathcal{C}^t$ the facilities we have opened by the end of round t. Similarly to how we did in Section 4.1.2, for any client v we denote by κ_v^t the cheapest cost of connecting that client to a facility at round t, which may include opening a facility. We also define $f^t(v)$ to be exactly this facility to which we can connect client v in the cheapest way possible at round t. Formally:

$$\kappa_v^t := \min_f(\mathbf{1}_{\{f \notin \mathcal{C}^t\}} \cdot c_f + c_{fv})$$
$$f^t(v) := \arg\min_f(\mathbf{1}_{\{f \notin \mathcal{C}^t\}} \cdot c_f + c_{fv})$$

We now need a notion of *coverage*, that is, we need to view facilities as corresponding to sets in an instance of Set Cover, and we need to decide which clients each facility covers. To do so, we first define the following for each client v:

$$\Gamma^t(v) := \{f : c_{fv} \leq \kappa_v^t/2\}$$

In words, $\Gamma^t(v)$ is the set of facilities that, if opened, would reduce the marginal cost of connecting v by a factor of at least 2. We will now say that a facility f *covers* a client v at time t if $f \in \Gamma^t(v)$. Intuitively, an unopened facility f covers a client v if f is within "distance" at most $\kappa_v^t/2$ of v. Thus, we can view our approach as modifying Algorithm 4.2 for a *dynamically changing* set system, where on each timestep the facilities are the sets, the clients are the elements and coverage is determined as we just described.

As we did before, by "guess-and-double" we assume that we know a bound β such that $\text{LP}_{\text{OPT}} \leq \beta \leq 2 \cdot \text{LP}_{\text{OPT}}$, where LP_{OPT} denotes the cost of the optimal LP solution to the final unknown instance (LP-FL). We are now ready to present Algorithm 4.4.

Algorithm 4.4: LearnOrCoverNMFL

```
1  Let $\mathcal{F}' \leftarrow \{f : \beta/m \leq c_f \leq \beta\}$ and $m' \leftarrow |\mathcal{F}'|$
2  Initialise $x_f^0 \leftarrow \frac{\beta}{c_f \cdot m'} \mathbf{1}_{\{f \in \mathcal{F}'\}}$
3  for $t \in [n]$ do
4      $v^t \leftarrow t$-th element in random order
5      if $\kappa_{v^t}^{t-1} \geq \beta/t$ then
6          For each facility $f$, open $f$ with probability $\min(\kappa_{v^t}^{t-1} \cdot x_f^{t-1}/\beta, 1)$
7          if $\sum_{f \in \Gamma^{t-1}(v^t)} x_f^{t-1} < 1$ then
8              For every facility $f$, update $x_f^t \leftarrow x_f^{t-1} \cdot \exp\{\mathbf{1}_{\{f \in \Gamma^{t-1}(v^t)\}} \cdot \kappa_{v^t}^{t-1}/c_f\}$
9              Let $Z^t = \langle c, x^t \rangle/\beta$ and normalise $x^t \leftarrow x^t/Z^t$
10         else
11             $x^t \leftarrow x^{t-1}$
12     else
13         $x^t \leftarrow x^{t-1}$
14     Open $f^{t-1}(v^t)$ and connect $v^t$ to $f^{t-1}(v^t)$
```

Notice how, if we adopt the dynamically-changing set system viewpoint described above, Algorithm 4.4 is very similar to Algorithm 4.2. Indeed, all the intuition behind Algorithm 4.2 also holds for Algorithm 4.4, while also the analysis for the latter is similar to that of the former. For this reason, we will not again go into as much detail in the proofs, and we will only sketch the main points.

First, we need some more notation, similar to Section 4.1.2. We denote by (x^*, y^*) the optimal fractional solution to (LP-FL). Let $U^t = \{v^{t+1}, \ldots, v^n\}$ be the clients that have not yet been connected at the end of round t. Define $X^t(v) := \sum_{f \in \Gamma^{t-1}(v)} x_f^{t-1}$ to be the "fractional coverage" offered by x to client v at round $t-1$ (recall that a facility f "covers" a client v if $f \in \Gamma^{t-1}(v)$). Lastly, define $\rho^t := \sum_v \kappa_v^t$.

As in Section 4.1.2, the central componenent of our analysis will be a potential function $\Phi(t) = C_1 \cdot \Phi_L(t) + C_2 \cdot \Phi_C(t)$, consisting of the following *learning* and *covering* components

$$\Phi_L(t) := \mathrm{KL}_c(x^* \parallel x^t) + 2 \cdot \sum_{v \in U^t} \sum_f c_{fv} \cdot y^*_{fv}$$
$$\Phi_C(t) := \beta \cdot \log\left(\frac{\rho^t}{\beta} + \frac{1}{n}\right)$$

The only major difference from the potential function used in the analysis of Algorithm 4.2 is the inclusion of an additional term in the learning component. This term is necessary in the analysis precisely because our set system is no longer static, but changes each round. We will examine this a bit more carefully later. For now, we will present the three lemmas that, as in the case of Algorithm 4.2, bound the initial potential and the change in each component of the potential, guaranteeing that either the learning or the covering component will decrease noticeably. Denote by Ξ^t the event that $\kappa^{t-1}_{v^t} \geq \beta/t$, thus the bulk of Algorithm 4.4 is executed.

Lemma 4.2.4 (Bounds on Φ). *The initial potential is bounded as $\Phi(0) = O(\beta \log mn)$, and $\Phi(t) \geq -\beta \log n$ for all t.*

Lemma 4.2.5 (Learning Potential). *For rounds t when Ξ^t holds, the expected change in the learning potential is bounded by*

$$\mathop{\mathbb{E}}_{v^t, \mathcal{R}^t}[\Phi_L(t) - \Phi_L(t-1) \mid x^{t-1}, U^{t-1}, \Xi^t] \leq \mathop{\mathbb{E}}_{v \sim U^{t-1}}\left[\frac{e^2-1}{2}\kappa_v^{t-1} \min(X^t(v), 1) - \kappa_v^{t-1}\right]$$

When Ξ^t does not hold:

$$\Phi_L(t) - \Phi_L(t-1) \leq 0$$

Lemma 4.2.6 (Covering Potential). *For rounds t when Ξ^t holds, the expected change in the covering potential is bounded by*

$$\mathop{\mathbb{E}}_{v^t, \mathcal{R}^t}[\Phi_C(t) - \Phi_C(t-1) \mid x^{t-1}, U^{t-1}, \Xi^t] \leq -\frac{1-e^{-1}}{4} \cdot \mathop{\mathbb{E}}_{u \sim U^{t-1}}[\kappa_u^{t-1} \cdot \min(X^t(u), 1)]$$

When Ξ^t does not hold:

$$\Phi_C(t) - \Phi_C(t-1) \leq 0$$

The proofs of all three lemmas are rather similar to those of the corresponding lemmas for Algorithm 4.2, so we will omit them. We will only examine the impact of the new term in the learning potential. Using analogous techniques as in Algorithm 4.2, we can show that the expected change in the KL divergence part of the learning potential when Ξ^t holds and $\Lambda^t = \{X^t(v^t) < 1\}$ holds is bounded by:

$$\mathop{\mathbb{E}}_{v \sim U^{t-1}}\left[\frac{e^2-1}{2}\kappa_v^{t-1} \min(X^t(v), 1) - \kappa_v^{t-1} \cdot \sum_{f \in \Gamma^{t-1}(v)} x^*_f \;\middle|\; \Xi^t, \Lambda^t\right]$$

In the proof of Algorithm 4.2 when our set system was static, we used the fact that $\sum_{S \ni v} x^*_S \geq 1$ due to the constraints of (LP-SC), however no such claim can be made here for $\sum_{f \in \Gamma^{t-1}(v)} x^*_f$. This is where the change of the second term of the learning potential comes into play. On round t, this change is equal to $-2 \cdot \sum_f c_{fv^t} \cdot y^*_{fv^t}$, thus the change in the total learning potential is bounded by

$$\mathop{\mathbb{E}}_{v\sim U^{t-1}}\left[\frac{e^2-1}{2}\kappa_v^{t-1}\min(X^t(v),1) - \left(\kappa_v^{t-1}\cdot\sum_{f\in\Gamma^{t-1}(v)} x^*_f + 2\cdot\sum_f c_{fv}\cdot y^*_{fv}\right)\middle|\, \Xi^t, \Lambda^t\right]$$

Here is where the definition of Γ^t is useful. We have that $2\cdot\sum_f c_{fv}\cdot y^*_{fv} \geq 2\cdot\sum_{f\notin\Gamma^{t-1}(v)} c_{fv}\cdot y^*_{fv} \geq 2\cdot\sum_{f\notin\Gamma^{t-1}(v)} \frac{\kappa_v^{t-1}}{2}\cdot y^*_{fv}$. Hence, we can finally bound the change in learning potential by

$$\begin{aligned}&\mathop{\mathbb{E}}_{v\sim U^{t-1}}\left[\frac{e^2-1}{2}\kappa_v^{t-1}\min(X^t(v),1) - \kappa_v^{t-1}\cdot\left(\sum_{f\in\Gamma^{t-1}(v)} x^*_f + \sum_{f\notin\Gamma^{t-1}(v)} y^*_{fv}\right)\middle|\, \Xi^t, \Lambda^t\right]\\ &\leq \mathop{\mathbb{E}}_{v\sim U^{t-1}}\left[\frac{e^2-1}{2}\kappa_v^{t-1}\min(X^t(v),1) - \kappa_v^{t-1}\,\middle|\, \Xi^t, \Lambda^t\right]\end{aligned}$$

where the last line follows from the constraints of (LP-FL), specifically from $1 \leq \sum_f y^*_{fv} = \sum_{f\in\Gamma^{t-1}(v)} y^*_{fv} + \sum_{f\notin\Gamma^{t-1}(v)} y^*_{fv} \leq \sum_{f\in\Gamma^{t-1}(v)} x^*_f + \sum_{f\notin\Gamma^{t-1}(v)} y^*_{fv}$.

Equipped with the above lemmas, we may now show the competitiveness of our algorithm, similarly to how we did for Algorithm 4.2.

Theorem 4.2.7. *Algorithm 4.4 is $O(\log mn)$-competitive.*

Proof. Let $c(\textsc{Alg}(t))$ be the cost paid by our algorithm up to and including round t. We break this in two components, letting $c'(\textsc{Alg}(t))$ be the cost for rounds when Ξ^t holds and $c''(\textsc{Alg}(t))$ the cost for the other rounds. When Ξ^t does not hold, we simply connect the client in the cheapest way possible and incur cost $\kappa_{v^t}^{t-1} < \beta/t$, so we have the following simple bound:

$$c''(\textsc{Alg}(n)) = \sum_{t=1}^n c''(\textsc{Alg}(t)) - c''(\textsc{Alg}(t-1)) \leq \sum_{t=1}^n \frac{\beta}{t} = O(\beta\cdot\log n)$$

We now consider $c'(\textsc{Alg}(n))$. When Ξ^t holds, the expected cost of the randomly opened facilities is $\kappa_{v^t}^{t-1}\cdot\langle c, x^{t-1}\rangle/\beta = \kappa_{v^t}^{t-1}$, and we pay an additional $\kappa_{v^t}^{t-1}$ to greedily connect the client in the last line of the algorithm, thus our expected total cost is at most $2\cdot\kappa_{v^t}^{t-1}$.

By combining lemmas Lemmas 4.2.5 and 4.2.6 and setting the constants $C_1 = 2$ and $C_2 = 4e(e+1)$ we have that the expected change in potential is:

$$\begin{aligned}&\mathop{\mathbb{E}}_{v^t,\mathcal{R}^t}[\Phi(t)-\Phi(t-1) \mid v^1,\dots,v^{t-1},\mathcal{R}^1,\dots,\mathcal{R}^{t-1},\Xi^t]\\ &\qquad\qquad\leq -\mathop{\mathbb{E}}_{v^t,\mathcal{R}^t}[2\cdot\kappa_{v^t} \mid v^1,\dots,v^{t-1},\mathcal{R}^1,\dots,\mathcal{R}^{t-1},\Xi^t]\end{aligned}$$

which cancels the expected change in c' each round. Thus, using the fact that both Φ_L and Φ_C do not increase when Ξ^t does not hold, we have that:

$$\mathop{\mathbb{E}}_{v^t,\mathcal{R}^t}[\Phi(t)-\Phi(t-1)+c'(\textsc{Alg}(t))-c'(\textsc{Alg}(t-1)) \mid v^1,\dots,v^{t-1},\mathcal{R}^1,\dots,\mathcal{R}^{t-1}] \leq 0$$

By repeatedly applying this inequality for all $t = 1,\dots,n$ we obtain:

$$\mathop{\mathbb{E}}_{v^t,\mathcal{R}^t}[\Phi(n)-\Phi(0)+c'(\textsc{Alg}(n))-c'(\textsc{Alg}(0))] \leq 0$$

Since $c'(\textsc{Alg}(0)) = 0$ and, by Lemma 4.2.4, $\Phi(0) = O(\beta \cdot \log mn)$ and $\Phi(n) \geq -\beta \cdot \log n$, this gives us:

$$\mathop{\mathbb{E}}_{v^t,\mathcal{R}^t}[c'(\textsc{Alg}(n))] \leq O(\beta \cdot \log mn)$$

Thus, combining the above we have:

$$\mathop{\mathbb{E}}_{v^t,\mathcal{R}^t}[c(\textsc{Alg}(n))] = \mathop{\mathbb{E}}_{v^t,\mathcal{R}^t}[c'(\textsc{Alg}(n))] + \mathop{\mathbb{E}}_{v^t,\mathcal{R}^t}[c''(\textsc{Alg}(n))] \leq O(\beta \cdot \log mn)$$

□

4.3 Random Order Covering IPs

In the last section, we saw how Algorithm 4.2 for Random-Order Set cover can be extended into Algorithm 4.2 for Random-Order Non-Metric Facility Location. It turns out that this approach can also be used to tackle two different types of Random-Order Online Integer Programmes, which significantly generalise Set Cover.

4.3.1 Covering Integer Programmes

As we mentioned in Chapter 1, a Covering Integer Programme (CIP) is defined as follows

$$\begin{aligned} \min \quad & \langle c, z\rangle \\ \text{s.t.} \quad & Az \geq \mathbf{1} \\ & z \in \mathbb{Z}^m_{\geq 0} \end{aligned} \tag{CIP}$$

where $A \in \mathbb{R}^{m\times n}_{\geq 0}$ is a matrix and $\mathbf{1}$ is a vector of n ones. If the constraints are given in the form $Az \geq b$ for some $b \in \mathbb{R}^n_{\geq 0}$, w.l.o.g. we divide the i-th row of A by b_i and consider that $b = \mathbf{1}$. Also, since $z \in \mathbb{Z}^m_{\geq 0}$, we may assume w.l.o.g. that $a_{ij} \in [0, 1]$ for any $i \in [n], j \in [m]$.

In the random-order version of this problem, the n constraints of (CIP) are revealed sequentially, in a random order. Whenever a constraint is revealed, if it is not satisfied, we must increase some variables of z in order to satisfy it, and we are not allowed to decrease variables. It is easy to see how, in the special case where $a_{ij} \in \{0, 1\}$, we recover the Set Cover problem.

This problem was examined in [34] along with Random-Order Set Cover. The algorithm presented is again very similar to Algorithm 4.2. We again assume we know, by "guess-and-double", a bound β such that $\text{LP}_{\text{OPT}} \leq \beta \leq 2 \cdot \text{LP}_{\text{OPT}}$. If z^t is the integer solution held by our algorithm at the end of round t, we define $d_i^t := \max(0, 1 - \langle a_i, z^t\rangle)$ to be the *undercoverage* of constraint i at the end of round t. We also define $\kappa_i^t := d_i^{t-1} \cdot \min_{k\in[m]} \frac{c_k}{a_{ik}}$ to be the minimum fractional cost of covering the current undercoverage of constraint i. Finally, for a vector y, denote the fractional remainder by $\tilde{y} := y - \lfloor y \rfloor$.

Algorithm 4.5 outputs a solution such that $Az \geq 1 - \gamma$ for a constant $\gamma = \frac{1}{e-1}$. This is done for technical reasons, and we can easily just buy $\lceil (1 - \gamma)^{-1} \rceil = 3$ copies of each column the algorithm buys to truly satisfy the constraints, incurring an additional factor of 3 in the cost.

Notice how, barring a couple of technical differences in lines 5 and 6, Algorithm 4.5 is the same as Algorithm 4.2 in the case where $a_{ij} \in \{0, 1\}$ and $z \in \{0, 1\}^m$. The analysis of this algorithm is also very similar. We denote by $U^t := \{i : d_i^t > \gamma\}$ the elements not covered to extent at least $1-\gamma$ by the end of round t, and we let $X_i^t = \langle a_i, x^t\rangle$. We define the

Algorithm 4.5: LearnOrCoverCIP

```
1 Let $m' \leftarrow |\{j : c_j \leq \beta\}|$
2 Initialise $x_j^0 \leftarrow \frac{\beta}{c_j \cdot m'} \cdot \mathbf{1}_{\{c_j \leq \beta\}}$ and $z^0 \leftarrow \mathbf{0}$
3 for $t \in [n]$ do
4     $i \leftarrow t$-th constraint in random order
5     if $d_i^{t-1} > \gamma$ then
6         Let $y := \kappa_i^t \cdot x^{t-1}/\beta$. For each column $j$, update $z_j^t \leftarrow z_j^{t-1} + \lfloor y_j \rfloor + \text{Ber}(\tilde{y}_j)$
7         if $\langle a_i, x^{t-1} \rangle < d_i^{t-1}$ then
8             For every $j$, update $x_j^t \leftarrow x_j^{t-1} \cdot \exp\{\kappa_i^t \cdot a_{ij}/c_j\}$
9             Let $Z^t = \langle c, x^t \rangle / \beta$ and normalise $x^t \leftarrow x^t / Z^t$
10        else
11            $x^t \leftarrow x^{t-1}$
12        Let $k^* = \arg\min_k \frac{c_k}{a_{ik}}$ and update $z_{k^*}^t \leftarrow z_{k^*}^t + \left\lceil \frac{d_i^{t-1}}{a_{ik^*}} \right\rceil$
```

potential $\Phi(t) = C_1 \cdot \Phi_L(t) + C_2 \cdot \Phi_C(t)$ exactly as we did in the analysis of Algorithm 4.2, and we similarly define the event Υ^t that for the constraint i^t arriving in round t we have $d_i^{t-1} > \gamma$. Then we can similarly prove the following lemmas for the initial potential and the change in the learning and covering components.

Lemma 4.3.1. *The initial potential is bounded as $\Phi(0) = O(\beta \cdot \log mn)$.*

Lemma 4.3.2. *For rounds t when Υ^t holds, the expected change in the learning potential is bounded by*

$$\mathop{\mathbb{E}}_{i^t, \mathcal{R}^t}[\Phi_L(t) - \Phi_L(t-1) \mid x^{t-1}, U^{t-1}, \Upsilon^t] \leq \mathop{\mathbb{E}}_{i \sim U^{t-1}}[(e-1)\kappa_i^t \min(X_i^{t-1}, d_i^{t-1}) - \kappa_i^t]$$

Lemma 4.3.3. *For rounds t when Υ^t holds, the expected change in the covering potential is bounded by*

$$\mathop{\mathbb{E}}_{i^t, \mathcal{R}^t}[\Phi_C(t) - \Phi_C(t-1) \mid x^{t-1}, U^{t-1}, \Upsilon^t] \leq -\alpha \cdot \mathop{\mathbb{E}}_{i' \sim U^{t-1}}[\kappa_{i'}^t \cdot \min(X_{i'}^{t-1}, d_{i'}^{t-1}))]$$

where α is a fixed constant.

As before, the three above lemmas imply the main theorem.

Theorem 4.3.4. *Algorithm 4.5 is $O(\log mn)$-competitive for Random-Order CIP.*

4.3.2 Set Multicover

Another type of online Integer Programme we will consider is the following one, which we will call Set Multicover

$$\begin{aligned} \min \quad & \langle \mathbf{1}, z \rangle \\ \text{s.t.} \quad & Az \geq b \\ & z \in \{0,1\}^m \end{aligned} \tag{LP-SMC}$$

This IP essentially corresponds to the Set Cover problem, where we impose the additional constraint that the i-th element must be covered at least b_i times. The random-order version of this problem was studied in [1], where again the ideas from Algorithm 4.2 were extended to design Algorithm 4.6. Let z^t be the algorithm's solution at the end of round t. We define $\mathcal{T}_i^t := \{j : z_j^{t-1} = 0, a_{ij} = 1\}$ to be the unbought sets at the beginning of round t that cover element i. We also define $X_i^{t-1} := \sum_{j\in\mathcal{T}_i^t} x_j^{t-1}$ to be the fractional coverage provided to element i by those unbought sets at the start of round t. As in the CIP case, we denote by $d_i^t := b_i - \langle a_i, z^{t-1}\rangle$ the integral undercoverage of element i at the beginning of round t.

Algorithm 4.6: LearnOrCoverSMC

```
Initialise $x_j^0 \leftarrow \beta/m$ for every $j$, and set $z_j^0 \leftarrow \mathbf{0}$
for $t \in [n]$ do
    $i \leftarrow t$-th constraint in random order
    if $i$ not covered on arrival then
        For each set $j$, sample $z_j^t \leftarrow \mathrm{Ber}(d_i^t \cdot x_j^{t-1}/\beta)$
        if $X_i^{t-1} \leq d_i^t$ then
            For each $j \in \mathcal{T}_i^t$, if $x_j^{t-1} \geq 1/e$, set $z_j^t \leftarrow 1$
            For each $j$, update $x_j^t \leftarrow x_j^{t-1} \cdot \exp\{\mathbf{1}_{\{j\in\mathcal{T}_i^t\}}\}$
            Let $Z^t := \langle \mathbf{1}, x^t\rangle/b$ and normalise $x^t \leftarrow x^t/Z^t$
        if $i$ still uncovered then
            $x^t \leftarrow x^{t-1}$ and $z^t \leftarrow z^{t-1}$
            For $d_i^t$-many arbitrary sets $j \in \mathcal{T}_i^t$, set $z_j^t \leftarrow 1$
    else
        $x^t \leftarrow x^{t-1}$ and $z^t \leftarrow z^{t-1}$
```

Again, notice how Algorithm 4.6 is very similar to Algorithm 4.2, with the interesting difference that the former deterministically buys any sets j that cover the arrived element and have a high enough x_j value. Also note that this algorithm samples more "aggressively" when it needs to buy more sets in order to satisfy a constraint, that is, when d_i^t is larger. The analysis of this algorithm is again similar to what we have encountered thus far, relying on the potential function $\Phi(t) = C_1 \cdot \Phi_L(t) + C_2 \cdot \Phi_C(t)$ and the following lemmas.

Lemma 4.3.5. *The initial potential is bounded as $\Phi(0) = O(\beta \log mn)$, and $\Phi(t) \geq -\beta \log n$ for all t.*

Lemma 4.3.6. *For rounds t when i^t arrives uncovered, the expected change in the learning potential is bounded by*

$$\mathop{\mathbb{E}}_{i^t,\mathcal{R}^t}[\Phi_L(t) - \Phi_L(t-1) \mid x^{t-1}, U^{t-1}, d_{i^t}^t > 0] \leq \mathop{\mathbb{E}}_{i\sim U^{t-1}}[(e-1)\cdot \min(X_i^{t-1}, d_i^t) - d_i^t]$$

When i^t is covered on arrival, $\Phi_L(t) - \Phi_L(t-1) \leq 0$.

Lemma 4.3.7. *For rounds t when i^t arrives uncovered, the expected change in the covering potential is bounded by*

$$\mathop{\mathbb{E}}_{i^t,\mathcal{R}^t}[\Phi_C(t) - \Phi_C(t-1) \mid x^{t-1}, U^{t-1}, d_{i^t}^t > 0] \leq -\alpha \cdot \mathop{\mathbb{E}}_{i\sim U^{t-1}}[\min(X_i^{t-1}, d_i^t)]$$

where α is a fixed constant. When i^t is covered on arrival, $\Phi_C(t) - \Phi_C(t-1) \leq 0$.

As before, the above three lemmas directly lead to the main theorem.

Theorem 4.3.8. *Algorithm 4.6 is $O(\log mn)$-competitive for Random-Order Set Multicover.*

4.4 AIPs in the Prophet Setting

So far, we have examined how a number of different Online Augmentable Integer Programmes (AIPs) can be tackled in the random-order setting, and how imposing random order is enough to allow for significantly improved guarantees compared to the fully adversarial setting. We will now see how these algorithms for the random-order setting can be used as black-boxes in order to design algorithms for the *prophet* setting, similarly to how we did in Section 3.3. First, we will need to recall the definition of AIPs, which we presented in Chapter 1.

Consider an integer programme defined by a vector of variables $z \in \mathbb{Z}^m$, a cost vector $c \in \mathbb{R}^m$ and a set of constrains V (each of the form $\langle a_j, z\rangle \geq b_j$). For any subset $V' \subseteq V$ of constraints, let $\textsc{Sols}(V') \subseteq \mathbb{Z}^m$ be the subset of solutions that are feasible to V'. Also, for any subset of constraints $V' \subseteq V$, any solution $z \in \textsc{Sols}(V')$ and any other subset of constrains $W \subseteq V$, define the *augmentation cost*:

$$\textsc{Aug}(W \mid z, V') \coloneqq \min_{w}\{\langle c, w\rangle : w + z \in \textsc{Sols}(V' \cup W)\}$$

or ∞ if no such w exists. Also let $\textsc{Backup}(W \mid z, V')$ be the minimiser of the above when it exists. The class of AIPs is defined as follows.

Definition 4.4.1 (AIPs). An *augmentable* integer programme (AIP) is one in which the augmentation costs are *monotone*, i.e. for any $V' \subseteq V'' \subseteq V$ and any $z' \leq z''$ such that $z' \in \textsc{Sols}(V')$ and $z'' \in \textsc{Sols}(V'')$, we have $\textsc{Aug}(W \mid z'', V'') \leq \textsc{Aug}(W \mid z', V')$ for any constraint set $W \subseteq V$.

In the online setting, the constraints of an AIP are revealed sequentially, we must cover each constraint upon arrival and we are only allowed to increase our variables z. Looking at the IP formulations of the problems we have considered so far ((LP-SC), (LP-FL), (CIP), (LP-SMC)), it is easy to verify that they are all contained within the family of AIPs.

In the *prophet* setting, there is a universe of constraints U for the AIP, and a set of independent distributions $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(n)}$ over U. On each round $t \in [n]$, a constraint is sampled according to $\mathcal{D}^{(t)}$ and presented to the algorithm. This means that we will not necessarily see the entire universe U. In the simple case of Set Cover, for example, each constraint corresponds to an element in the universe.

We will now present a result of [1], where it was demonstrated how, given an algorithm for the random-order version of an AIP and sample access to the distributions $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(n)}$, we can obtain an algorithm for the prophet version of the problem. Specifically, we have the following theorem.

Theorem 4.4.1. *Let $\mathcal{P}$ be a problem in the family of AIPs. If algorithm $\mathcal{A}$ is a Δ-competitive algorithm for the random-order version of $\mathcal{P}$, then there is a single-sample 2Δ-competitive algorithm $\mathcal{A}'$ for the prophet version of $\mathcal{P}$.*

The reduction described in the theorem is presented in Algorithm 4.7. Simply put, we simulate the random-order algorithm on one sample from each distribution, and we buy any "sets" that algorithm buys *a priori* (before the actual input sequence). Then, during the actual input sequence, we simply greedily cover any requests that are not satisfied upon arrival.

Algorithm 4.7: AIP Prophet to AIP Random-Order

```
1 Let $\textsc{MockRun} = \{\hat{v}^1, \dots, \hat{v}^n\}$ be one sample each from $\mathcal{D}^{(1)}, \dots, \mathcal{D}^{(n)}$
2 Pass a uniformly random permutation of $\textsc{MockRun}$ as input to $\mathcal{A}$
3 Let $\hat{z}$ be the output of $\mathcal{A}$
4 Initialise $z \leftarrow \hat{z}$
5 for $t \in [n]$ do
6     Draw constraint $v^t \sim \mathcal{D}^{(t)}$
7     if $v^t$ not satisfied by $z$ then
8         Update $z \leftarrow z + \textsc{Backup}(v^t \mid z, \textsc{MockRun} \cup \{v^1, \dots, v^{t-1}\})$
```

Proof. Let $\mathcal{A}$ be a Δ-competitive random-order algorithm. Define $\mathcal{A}'$ to be Algorithm 4.7. Clearly, $\mathcal{A}'$ is single-sample. The two sets of samples $v^1, \dots, v^n$ and $\hat{v}^1, \dots, \hat{v}^n$ are identically distributed, so

$$\textsc{Opt} := \mathbb{E}[c(\textsc{Opt}(v^1, \dots, v^n))] = \mathbb{E}[c(\textsc{Opt}(\hat{v}^1, \dots, \hat{v}^n))]$$

Thus the expected cost of the solution $\hat{z}$ bought by $\mathcal{A}$ is at most $\Delta \cdot \textsc{Opt}$ by the guarantee of $\mathcal{A}$. It remains to bound the cost of the backup purchases in the last line of $\mathcal{A}'$.

Consider a pair of requests $v^t, \hat{v}^t \sim \mathcal{D}^{(t)}$, where v^t is part of the actual input sequence and $\hat{v}$ is part of $\textsc{MockRun}$. We will argue that the expected augmentation cost of v^t is no more than the expected augmentation cost of $\hat{v}^t$ during the simulation of $\mathcal{A}$. Let $z(v^t)$ be the state of z at the beginning of round t when v^t arrives, and let $\hat{z}(\hat{v}^t)$ be the state of the solution of algorithm $\mathcal{A}$ at the beginning of the round when $\hat{v}^t$ arrives in the random order. Finally, let $\textsc{MockRun}_{<\hat{v}^t}$ be the set of requests of $\textsc{MockRun}$ that arrive before $\hat{v}^t$ in the random order. We have that:

$$\begin{aligned}\mathbb{E}\left[\textsc{Aug}\left(v^t \mid z(v^t)\textsc{MockRun} \cup \{v^1, \dots, v^{t-1}\}\right)\right] &\leq \mathbb{E}\left[\textsc{Aug}\left(v^t \mid \hat{z}(\hat{v}^t), \textsc{MockRun}_{<\hat{v}^t}\right)\right] \\ &= \mathbb{E}\left[\textsc{Aug}\left(\hat{v}^t \mid \hat{z}(\hat{v}^t), \textsc{MockRun}_{<\hat{v}^t}\right)\right]\end{aligned}$$

The inequality holds by the monotonicity of augmentation property in the definition of AIPs, since $z(v^t) \geq \hat{z}(\hat{v}^t)$. The equality holds because v^t and $\hat{v}^t$ are identically distributed. Now, notice that when $\hat{v}^\tau$ arrives during the simulation of $\mathcal{A}$, that algorithm must pay at least $\textsc{Aug}(\hat{v}^\tau \mid \hat{z}(\hat{v}^\tau), \textsc{MockRun}_{<\hat{v}^\tau})$ (recall that this is the cheapest way the solution $\hat{z}(\hat{v}^\tau)$ can be augmented to cover $\hat{v}^\tau$). Hence, summing the above inequality over t we obtain:

$$\begin{aligned}&\sum_{t=1}^{n} \mathbb{E}\left[\textsc{Aug}\left(v^t \mid z(v^t)\textsc{MockRun} \cup \{v^1, \dots, v^{t-1}\}\right)\right] \\ &\leq \sum_{t=1}^{n} \mathbb{E}\left[\textsc{Aug}\left(\hat{v}^t \mid \hat{z}(\hat{v}^t), \textsc{MockRun}_{<\hat{v}^t}\right)\right] \leq \mathbb{E}[c(\hat{z})] \leq \Delta \cdot \textsc{Opt}\end{aligned}$$

where we again used the competitiveness guarantee of $\mathcal{A}$. In total, we have that $\mathcal{A}'$ will pay at most $\Delta \cdot \textsc{Opt}$ on expectation when buying the solution of $\mathcal{A}$ and an additional $\Delta \cdot \textsc{Opt}$ to perform all augmentations, thus its total expected cost is at most $2\Delta \cdot \textsc{Opt}$. □

Therefore, the $O(\log mn)$-competitive algorithms that we have seen for the random-order versions of Set Cover, Metric and Non-Metric Facility Location, Covering Integer Programmes and Set Multicover can all be used as black-boxes in order to design $O(\log mn)$-competitive algorithms for the prophet versions of those problems as well. This reduction once again demonstrates how, in general terms, the prophet version of a problem is easier than the random order version.

4.5 Steiner Tree

Another classical combinatorial optimisation problem is the Steiner Tree problem. Here, we are given a metric space (V, d), which can be thought of as a complete graph with edge weights $d(u, v)$. We are also given a set $R \subseteq V$ of n vertices in that graph, and our goal is to pick a minimum-weight subset of edges in $\binom{V}{2}$, such that all vertices in R are connected. Note that, if we are given a complete graph whose edge weights do not induce a metric, we can simply take the metric closure of that graph and work with that instead. Any solution in this metric closure directly translates to a solution in the original, non-metric instance with the same approximation guarantees. Hence, there is no reason to consider the non-metric variant of Steiner Tree. It can easily be shown that simply outputting the minimum spanning tree over the vertices in R is a 2-approximation for the offline Steiner Tree problem, while better approximation guarantees are also possible.

In the online setting, the n requests in R are revealed one at a time. After the first request has arrived, whenever each subsequent request arrives, we must immediately buy a subset of edges in $\binom{V}{2}$ to connect it with the previous requests. In the fully adversarial setting, a simple greedy algorithm that connects each request to the previous ones in the cheapest way possible is $O(\log n)$ competitive.

It is easy to verify that Steiner Tree is indeed an AIP. However, unlike the AIPs we have discussed so far, imposing random order of arrivals *does not* significantly weaken the adversary compared to the fully-adversarial setting. In fact, there is an $\Omega(\log n)$ lower bound on the competitiveness of Steiner Tree not only for the fully adversarial case, but also for the random-order setting.

4.5.1 Failure in the Random-Order Setting

We will present a theorem due to [38] demonstrates how we can construct an instance of online Steiner Tree such that, even if the instance is presented in a random order, the $\Omega(\log n)$ lower bound of the fully adversarial setting still holds.

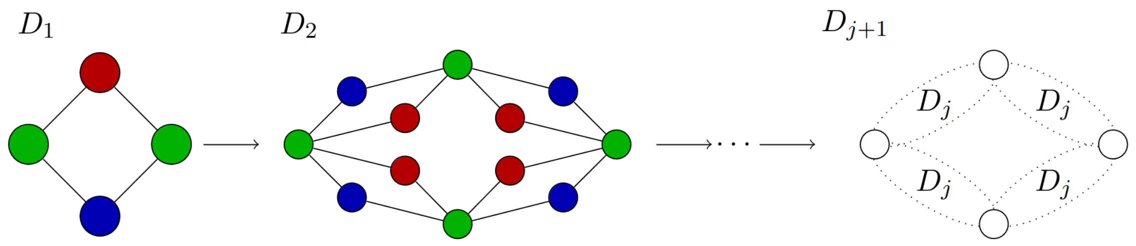
This construction uses two crucial properties of Steiner Tree. The first one is that duplicating requests does not change the cost of the optimal solution, but making many copies of a request makes it more likely that one of those copies will appear early in the random-order sequence. Hence, if we have a request sequence σ that is the worst-case for the fully adversarial setting, we can duplicate the i-th request C^{n-i} times. Then, if we apply a uniformly random permutation and remove all but the first copy of each unique request, the result will look close to the initial σ with high probability, so we will effectively recover the adversarial worst-case. Of course, this would increase the sequence length from n to $\approx C^n$, so the lower bound would be doubly logarithmic in the sequence length.

The second property of online Steiner Tree is that the worst-case example in the fully-adversarial setting consists of a sequence of $\log n$ *batches* of requests, where each batch contains $\approx 2^i$ requests, and the relative order of the requests within a batch does not matter. Thus, we can instead duplicate each request in the i-th batch $C^{\log n - i}$ times, and this will indeed give us the logarithmic lower bound we have claimed. Let us now present the argument formally.

Theorem 4.5.1. *No (randomised or deterministic) algorithm can attain an $o(\log n)$ competitive ratio for Random-Order Online Steiner Tree.*

Proof. Without loss of generality, we will assume that the algorithm is *lazy*, in the sense that, whenever a request arrives, the algorithm buys edges along a simple path that contains no requests other than at its endpoints (removing any further edges that the algorithm may choose will only decrease its cost).

We will construct an instance based on the so-called *diamond graphs*. The diamond graph D_j of level j is defined recursively as follows: D_0 is simply one edge with two endpoints, and for $j \geq 0$ the graph D_{j+1} is constructed from D_j by first creating an edge parallel to each edge in D_j and then subdividing each edge into two new edges by creating a new vertex for each. This is illustrated in the following figure.



Another way to view the above construction is that D_{j+1} is constructed by connecting 4 copies of D_j in a square shape. When we refer to a *path* in a diamond graph D_j, we will exclusively refer to simple paths that connect the leftmost to the rightmost vertex in D_j. Notice that each of those paths has length 2^j. Also note that, for any of those paths, there are exactly two choices for its middle vertex (e.g. in the above figure, for graph D_2, any path must cross one of the two middle green vertices). We will call those two vertices *top* and *bottom central vertices* for D_j. If a path in D_j crosses the top central vertex we will call it a *top path* in D_j, otherwise we will call it a *bottom path*.

We will define a graph and a sequence of requests of length at most n in this graph. It suffices to consider the case where n is a power of 16. Let $\ell = \log_4 n$, which is an integer. The graph we will consider for the rest of this construction is D_ℓ, and we will consider unit-cost edges.

As we mentioned, D_j is constructed by connecting 4 copies of D_{j-1}. Thus, D_ℓ can be seen as consisting of 4 subgraphs $D_{\ell-1}$, or of 16 subgraphs $D_{\ell-2}$ and so forth. Notice that each vertex in D_ℓ is a central vertex for some subgraph D_j of D_ℓ for some $j \leq \ell$. We will call this j the *level* of the vertex. Thus, D_ℓ contains 2 vertices of level ℓ, 8 vertices of level $\ell - 1$ and, in general, $2 \cdot 4^i$ vertices of level $\ell - i$.

We will now define the request sequence that we will use to show the lower bound. We first draw one path P in D_ℓ uniformly at random. Notice that P crosses 2^i subgraphs $D_{\ell-i}$ for each $i \in [\ell]$, thus it also crosses 2^i vertices of level $\ell - i$ for $i \in [\ell - 1]$. Each node of level $\ell - i$ in P will appear $4^{\ell-i}$ times in our request sequence. Note that the offline optimum is precisely this path P. By the above, the length of our request sequence is $\sum_{i\in[\ell]} 2^i \cdot 4^{\ell-i} = 4^\ell \sum_{i\in[\ell]} 2^{-i} \leq 2 \cdot 4^\ell = n$.

Having defined our problem instance, let us now analyse the performance of any algorithm on it. Since our problem instance is randomised, by Yao's Principle it suffices to consider a deterministic algorithm. Let $v_{j,b}$ be the b-th level-j vertex that appears on P from left to right. Also let $X_{j,b}$ be the cost that the algorithm incurs to connect this vertex. We will show the following lemma.

Lemma 4.5.2. $\mathbb{E}[X_{j,b}] \geq 2^{j-3}$ *for all* j *and* b.

Proof. By definition, $v_{j,b}$ is a central node of a subgraph D_j, where path P connects the leftmost and rightmost vertices of D_j. As we explained above, in this subgraph D_j path P crosses 2^i vertices of level $j - i$ for $i \in [j - 1]$. Since there are 4^{j-i} copies of each vertex of level $j-i$ in the request sequence, due to the uniformly random arrival order, the probability that a copy of $v_{j,b}$ will appear before any other vertex in this subgraph D_j is

$$\frac{4^j}{\sum_{i=0}^{j-1} 2^i 4^{j-i}} \geq \frac{1}{\sum_{i=0}^{\infty} \frac{1}{2}} = \frac{1}{2}$$

When this event holds, no requests will have appeared in this subgraph D_j before $v_{j,b}$ appears. Conditioned on this event, we have two cases:

- If the algorithm has not picked any edges in D_j, when $v_{j,b}$ arrives it will have to pay at least the cost of connecting $v_{j,b}$ to either the leftmost or the rightmost vertex of D_j. This cost is 2^{j-1}.

- If the (lazy, deterministic) algorithm has picked any edges in D_j, it will have done so to connect a request outside D_j, so it will have picked only a single path in D_j, either the top or the bottom path. Since P is chosen uniformly at random, with probability $1/2$ we will have that $v_{j,b}$ will not be connected by the algorithm when it appears (that is, $v_{j,b}$ will be a top central vertex when the algorithm has chosen the bottom path in D_j, or vice versa). In this case, the algorithm will again need to connect $v_{j,b}$ to either the leftmost or the rightmost vertex of D_j and pay 2^{j-1}.

Thus, combining all the above, we have that with probability at least $\frac{1}{2} \cdot \frac{1}{2}$ the algorithm will need to pay 2^{j-1} when $v_{j,b}$ arrives, so indeed we have $\mathbb{E}[X_{j,b}] \geq \frac{1}{2} \cdot \frac{1}{2} \cdot 2^{j-1} = 2^{j-3}$. □

Having shown the above lemma, we can now bound the total expected cost of the algorithm as follows

$$\mathbb{E}\left[\sum_{j=1}^{\ell} \sum_{b=1}^{2^{\ell-j}} X_{j,b}\right] = \sum_{j=1}^{\ell} \sum_{b=1}^{2^{\ell-j}} \mathbb{E}[X_{j,b}] \geq \sum_{j=1}^{\ell} 2^{\ell-j} 2^{j-3} = \frac{\ell}{8} \cdot 2^{\ell}$$

As we mentioned, the offline optimum is simply the path P, which costs 2^{ℓ}. Thus, we conclude that any algorithm will have to pay at least $\Omega(\ell) \cdot \textsc{Opt} = \Omega(\log n) \cdot \textsc{Opt}$ in expectation for the problem instance we have constructed. □

4.5.2 The Prophet Setting

We have seen how we cannot take advantage of the random arrival order in the Steiner Tree problem. However, it is significantly easier to tackle the problem in the prophet setting. Here, there are n independent distributions $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(n)}$ over the set V of vertices, and the request v^t at round t is drawn from $\mathcal{D}^{(t)}$. The following simple algorithm, due to [38], uses just one sample from each distribution in order to output an *a priori* spanning tree, and then greedily augments it when the actual requests arrive. This is the same logic we used in Algorithm 4.7 of the previous section.

Theorem 4.5.3. *Algorithm 4.8 is 4-competitive.*

Proof. Since the sets $v^1, \hat{v}^2, \hat{v}^3, \ldots, \hat{v}^n$ and $v^1, v^2, \ldots, \hat{v}^n$ are identically distributed, we have

$$\textsc{Opt} := \mathbb{E}[c(\textsc{Opt}(v^1, \ldots, v^n))] = \mathbb{E}[c(\textsc{Opt}(v^1, \hat{v}^2, \ldots, \hat{v}^n))]$$

Algorithm 4.8: Prophet Steiner Tree

```
1 Let $\hat{v}^2, \hat{v}^3, \dots, \hat{v}^n$ be one sample each from $\mathcal{D}^{(2)}, \mathcal{D}^{(3)}, \dots, \mathcal{D}^{(n)}$
2 Let $v_1$ be the first actual request, and let $S = \{v_1\} \cup \{\hat{v}^2, \dots, \hat{v}^n\}$
3 Find a minimum spanning tree $\widehat{T}$ connecting all vertices in $S$
4 Initialise $T \leftarrow \widehat{T}$
5 for $t = 2, \dots, n$ do
6     Draw $v^t \sim \mathcal{D}^{(t)}$
7     Add to $T$ the cheapest edge to connect $v^t$ to $S \cup \{v^2, \dots, v^{t-1}\}$
```

We know that the minimum spanning tree on S is a 2-approximation for the optimal Steiner Tree on S, thus $\mathbb{E}[c(\widehat{T})] \leq 2 \cdot \textsc{Opt}$. It remains to bound the cost of the greedy augmentations performed in the last line of the algorithm.

Consider that the tree $\widehat{T}$ is rooted at v_1. Let $w(\hat{v}^\tau)$ be the weight of the first edge of $\widehat{T}$ from vertex $\hat{v}^\tau$ to the root v^1. Clearly, $c(\widehat{T}) = \sum_\tau w(v^\tau)$. Now, the expected augmentation cost paid to connect v^t to $S \cup \{v^2, \dots, v^{t-1}\}$ is at most the expected cost to connect v^t to $S \setminus \{\hat{v}^t\}$. Since v^t and $\hat{v}^t$ are identically distributed, this is equal to the expected cost to connect $\hat{v}^t$ to $S \setminus \{\hat{v}^t\}$, which is equal to the expected distance of $\hat{v}^t$ from its closest neighbour in $\widehat{T}$ (since $\widehat{T}$ is an MST, it contains the min-weight edge from $\hat{v}^t$ to $S \setminus \{\hat{v}^t\}$). This is at most $\mathbb{E}[w(\hat{v}^t)]$. Thus, reusing the notation for AIPs, if $T(v^t)$ is the solution of our algorithm at the beginning of the round when v^t arrives, we have shown that

$$\mathbb{E}\big[\textsc{Aug}\big(v^t \mid T(v^t), S \cup \{v^2, \dots, v^{t-1}\}\big)\big] \leq \mathbb{E}\big[w(\hat{v}^t)\big]$$

Hence, summing over all t, we bound the total expected augmentation cost by $\sum_t w(v^t) = c(\widehat{T}) \leq 2 \cdot \textsc{Opt}$, so our overall expected cost is indeed at most $4 \cdot \textsc{Opt}$. □

Bibliography

[1] A. Gupta, G. Kehne, and R. Levin, “Set covering with our eyes wide shut”, in *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2024, pp. 4530–4553.

[2] N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor, “The online set cover problem”, *SIAM Journal on Computing*, vol. 39, no. 2, pp. 361–370, 2009.

[3] N. Buchbinder and J. Naor, “Online primal-dual algorithms for covering and packing”, *Mathematics of Operations Research*, vol. 34, no. 2, pp. 270–286, 2009.

[4] N. Buchbinder, A. Gupta, M. Molinaro, and J. Naor, “k-servers with a smile: Online algorithms via projections”, in *Proceedings of the 2019 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 98–116.

[5] A. Meyerson, “Online facility location”, in *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, 2001, pp. 426–431.

[6] D. Fotakis, “On the competitive ratio for online facility location”, *Algorithmica*, vol. 50, no. 1, pp. 1–57, 2008.

[7] A. Anagnostopoulos, R. Bent, E. Upfal, and P. V. Hentenryck, “A simple and deterministic competitive algorithm for online facility location”, *Information and Computation*, vol. 194, no. 2, pp. 175–202, 2004.

[8] D. Fotakis, “A primal-dual algorithm for online non-uniform facility location”, *Journal of Discrete Algorithms*, vol. 5, no. 1, pp. 141–148, 2007.

[9] M. Imase and B. M. Waxman, “Dynamic steiner tree problem”, *SIAM Journal on Discrete Mathematics*, vol. 4, no. 3, pp. 369–384, 1991.

[10] M. Manasse, L. McGeoch, and D. Sleator, “Competitive algorithms for on-line problems”, in *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, 1988, pp. 322–333.

[11] E. Koutsoupias and C. H. Papadimitriou, “On the k-server conjecture”, *Journal of the ACM*, vol. 42, no. 5, pp. 971–983, 1995.

[12] N. Bansal, N. Buchbinder, A. Madry, and J. Naor, “A polylogarithmic-competitive algorithm for the k-server problem”, *Journal of the ACM*, vol. 62, no. 5, pp. 1–49, 2015.

[13] S. Bubeck, C. Coester, and Y. Rabani, “The randomized k-server conjecture is false!”, in *Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC)*, 2023, pp. 581–594.

[14] A. Borodin, N. Linial, and M. E. Saks, “An optimal on-line algorithm for metrical task system”, *Journal of the ACM*, vol. 39, no. 4, pp. 745–763, 1992.

[15] A. Fiat and M. Mendel, “Better algorithms for unfair metrical task systems and applications”, *SIAM Journal on Computing*, vol. 32, no. 6, pp. 1403–1422, 2003.

[16] S. Bubeck, M. B. Cohen, J. R. Lee, and Y. T. Lee, “Metrical task systems on trees via mirror descent and unfair gluing”, *SIAM Journal on Computing*, vol. 50, no. 3, pp. 909–923, 2021.

[17] T. S. Ferguson, “Who solved the secretary problem?”, *Statistical Science*, vol. 4, no. 3, pp. 282–289, 1989.

[18] S. Agrawal, Z. Wang, and Y. Ye, “A dynamic near-optimal algorithm for online linear programming”, *Operations Research*, vol. 62, no. 4, pp. 876–890, 2014.

[19] R. Kleinberg, “A multiple-choice secretary algorithm with applications to online auctions”, in *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2005, pp. 630–631.

[20] M. Babaioff, N. Immorlica, and R. Kleinberg, “Matroids, secretary problems, and online mechanisms”, in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007, pp. 434–443.

[21] O. Lachish, “O(log log rank) competitive ratio for the matroid secretary problem”, in *IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS)*, 2014, pp. 326–335.

[22] M. Feldman, O. Svensson, and R. Zenklusen, “A simple o(log log(rank))-competitive algorithm for the matroid secretary problem”, in *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2015, pp. 1189–1201.

[23] N. Korula and M. Pál, “Algorithms for secretary problems on graphs and hypergraphs”, in *36th Internatilonal Colloquium on Automata, Languages and Programming (ICALP)*, 2009, pp. 508–520.

[24] J. A. Soto, A. Turkieltaub, and V. Verdugo, “Strong algorithms for the ordinal matroid secretary problem”, in *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2018, pp. 715–734.

[25] T. Kesselheim, K. Radke, A. Tönnis, and B. Vöcking, “An optimal online algorithm for weighted bipartite matching and extensions to combinatorial auctions”, in *ESA*, 2013, pp. 589–600.

[26] K. Chaudhuri, C. Daskalakis, R. Kleinberg, and H. Lin, “Online bipartite perfect matching with augmentations”, 2009, pp. 1044–1052.

[27] A. Rubinstein, J. Z. Wang, and S. M. Weinberg, “Optimal single-choice prophet inequalities from samples”, in *11th Innovations in Theoretical Computer Science Conference (ITCS)*, vol. 151, 2020, 60:1–60:10.

[28] M. T. Hajiaghayi, R. Kleinberg, and T. Sandholm, “Automated online mechanism design and prophet inequalities”, in *Proceedings of the 22nd National Conference on Artificial Intelligence - Volume 1*, 2007, pp. 58–65.

[29] S. Chawla, J. D. Hartline, D. L. Malec, and B. Sivan, “Multi-parameter mechanism design and sequential posted pricing”, in *Proceedings of the Forty-Second ACM Symposium on Theory of Computing (STOC)*, 2010, pp. 311–320.

[30] S. Alaei, M. Hajiaghayi, and V. Liaghat, “Online prophet-inequality matching with applications to ad allocation”, in *Proceedings of the 13th ACM Conference on Electronic Commerce (EC)*, 2012, pp. 18–35.

[31] P. D. Azar, R. Kleinberg, and S. M. Weinberg, "Prophet inequalities with limited information", in *Proceedings of the 2014 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1358–1377.

[32] R. Kleinberg and S. M. Weinberg, "Matroid prophet inequalities", in *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing (STOC)*, 2012, pp. 123–136.

[33] T. Ezra, M. Feldman, N. Gravin, and Z. G. Tang, "Online stochastic max-weight matching: Prophet inequality for vertex and edge arrival models", in *Proceedings of the 21st ACM Conference on Economics and Computation (EC)*, 2020, pp. 769–787.

[34] A. Gupta, G. Kehne, and R. Levin, "Random order online set cover is as easy as offline", in *Proceedings of the 62nd IEEE Symposium on Foundations of Computer Science (FOCS)*, 2022, pp. 1253–1264.

[35] S. Korman, "On the use of randomization in the online set cover problem", *Master's Thesis*, 2004.

[36] H. Kaplan, D. Naori, and D. Raz, "Almost tight bounds for online facility location in the random-order model", in *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2023, pp. 1523–1544.

[37] N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor, "A general approach to online network optimization problems", *ACM Transactions on Algorithms*, vol. 2, no. 4, pp. 640–660, 2006.

[38] N. Garg, A. Gupta, S. Leonardi, and P. Sankowski, "Stochastic analyses for online combinatorial optimization problems", 2008, pp. 942–951.

[39] D. D. Sleator and R. E. Tarjan, "Amortized efficiency of list update and paging rules", *Commun. ACM*, vol. 28, no. 2, pp. 202–208, 1985.

[40] R. M. Karp, U. V. Vazirani, and V. V. Vazirani, "An optimal algorithm for on-line bipartite matching", in *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing (STOC)*, 1990, pp. 352–358.

[41] G. Aggarwal, G. Goel, C. Karande, and A. Mehta, "Online vertex-weighted bipartite matching and single-bid budgeted allocations", in *Proceedings of the 2011 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2011, pp. 1253–1264.

[42] N. R. Devanur, K. Jain, and R. D. Kleinberg, "Randomized primal-dual analysis of ranking for online bipartite matching", in *Proceedings of the 2013 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA).* 2013, pp. 101–107.

[43] A. Fiat, R. M. Karp, M. Luby, L. A. McGeoch, D. D. Sleator, and N. E. Young, "Competitive paging algorithms", *Journal of Algorithms*, vol. 12, no. 4, pp. 685–699, 1991.

[44] E. F. Grove, M.-Y. Kao, P. Krishnan, and J. S. Vitter, "Online perfect matching and mobile computing", in *Algorithms and Data Structures*, 1995, pp. 194–205.

[45] A. Bernstein, J. Holm, and E. Rotenberg, "Online bipartite matching with amortized $O(log^2n)$ replacements", *Journal of the ACM*, vol. 66, no. 5, pp. 1–23, 2019.

[46] N. Buchbinder, A. Gupta, D. Hathcock, A. R. Karlin, and S. Sarkar, "Maintaining matroid intersections online", in *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2024, pp. 4283–4304.

[47] A. Gupta, K. Talwar, and U. Wieder, “Changing bases: Multistage optimization for matroids and matchings”, in *Automata, Languages, and Programming*, 2014, pp. 563–575.

[48] *Beyond the Worst-Case Analysis of Algorithms.* Cambridge University Press, 2021.

[49] C. Kenyon, “Best-fit bin-packing with random order”, in *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1996, pp. 359–364.

[50] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham, “Worst-case performance bounds for simple one-dimensional packing algorithms”, *SIAM Journal on Computing*, vol. 3, no. 4, pp. 299–325, 1974.

[51] T. Hill and R. Kertz, “A survey of prophet inequalities in optimal stopping theory”, *Contemporary Mathematics*, vol. 125, no. 1, pp. 191–207, 1992.

[52] J. P. Gilbert and F. Mosteller, “Recognizing the maximum of a sequence”, *Journal of the American Statistical Association*, vol. 61, no. 313, pp. 35–73, 1966.

[53] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg, “A knapsack secretary problem with applications”, in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, Jan. 2007, pp. 16–28.

[54] S. Albers and L. Ladewig, “New results for the k-secretary problem”, *Theoretical Computer Science*, vol. 863, pp. 102–119, 2021.

[55] M. Dinitz, “Recent advances on the matroid secretary problem”, *SIGACT News*, vol. 44, no. 2, pp. 126–142,

[56] M. Babaioff, M. Dinitz, A. Gupta, N. Immorlica, and K. Talwar, “Secretary problems: Weights and discounts”, in *Symposium on Discrete Algorithms (SODA)*, 2009, pp. 1245–1254.

[57] U. Krengel and L. Sucheston, “On semiamarts, amarts, and processes with finite value”, *Advances in Probability Related Topics*, vol. 4, pp. 197–266, 1978.

[58] E. Samuel-Cahn, “Comparison of threshold stop rules and maximum for independent nonnegative random variables”, *The Annals of Probability*, vol. 12, no. 4, pp. 1213–1216, 1984.

[59] M. Feldman, N. Gravin, and B. Lucier, “Combinatorial auctions via posted prices”, in *Proceedings of the 2015 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 123–135.

[60] M. Mahdian, Y. Ye, and J. Zhang, “Approximation algorithms for metric facility location problems”, *SIAM Journal on Computing*, vol. 36, no. 2, pp. 411–432, 2006.

Appendix A

Mathematical Tools

Theorem A.0.1 (Hoeffding's Inequality). *Let $X_1, \ldots, X_n$ be independent random variables such that $\mathbb{P}[a_i \leq X_i \leq b_i] = 1$ for all $i \in [n]$. Consider the sum of these random variables $S_n = \sum_{i \in [n]} X_i$. For all $t > 0$, the following bounds hold*

$$\mathbb{P}[S_n - \mathbb{E}[S_n] \geq t] \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right)$$
$$\mathbb{P}[|S_n - \mathbb{E}[S_n]| \geq t] \leq 2\exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right)$$

Theorem A.0.2 (Chernoff Bounds for independent Bernoulli random variables). *Suppose that $X_1, \ldots, X_n$ are independent random variables taking values in $\{0, 1\}$. Let X denote their sum and $\mu = \mathbb{E}[X]$. Then the following bounds hold*

$$\mathbb{P}[X \geq (1+\delta)\mu] \leq \exp\left(-\frac{\delta^2 \mu}{2+\delta}\right) \quad \forall \delta > 0$$
$$\mathbb{P}[X \leq (1-\delta)\mu] \leq \exp\left(-\frac{\delta^2 \mu}{2}\right) \quad \forall \delta \in [0, 1)$$
$$\mathbb{P}[|X - \mu| \geq \delta\mu] \leq 2\exp\left(-\frac{\delta^2 \mu}{3}\right) \quad \forall \delta \in [0, 1)$$

Definition A.0.1 (Matroids). A finite *matroid* $\mathcal{M} = (\mathcal{U}, \mathcal{I})$ is constructed from a finite ground set $\mathcal{U} \neq \emptyset$ and a family $\mathcal{I} \neq \emptyset$ of subsets of $\mathcal{U}$ called the *independent sets* of $\mathcal{U}$, such that the following two properties hold

- If $B \subseteq A \subseteq \mathcal{U}$ and $A \in \mathcal{I}$, then $B \in \mathcal{I}$. This is called the *hereditary* or *downward-closed* property.
- If $A, B \in \mathcal{I}$ and $|A| > |B|$, then there exists some element $x \in B \setminus A$ such that $A \cup \{x\} \in \mathcal{I}$. This is called the *exchange* property.

A maximal independent set of a matroid (that is, an independent set that becomes dependent upon adding any element to it), is called a *basis* of the matroid. The bases of a matroid completely characterise it. That is, we can equivalently define a matroid $\mathcal{M} = (\mathcal{U}, \mathcal{B})$ to be constructed by a ground set $\mathcal{U}$ and a set of bases $\mathcal{B} \neq \emptyset$ of the matroid, such that the following holds

- If $A, B \in \mathcal{B}$ and there exists an element $a \in \mathcal{A} \setminus \mathcal{B}$, then there exists an element $b \in B \setminus A$ such that $(A \setminus \{a\}) \cup \{b\} \in \mathcal{B}$. This is called the *basis exchange* property.

Any two bases of a matroid have the same number of elements. This number is called the *rank* of the matroid.

Some examples of interesting matroids include the following:

1. Consider a set of items $\mathcal{U}$ and an integer k. We can construct a matroid M by letting $\mathcal{U}$ be the ground set and defining a subset of $\mathcal{U}$ to be independent iff it contains at most k elements. The bases of this matroid are all subsets of exactly k elements of $\mathcal{U}$. This is called a *k-uniform matroid*.

2. Consider a set of items $\mathcal{U}$ and a collection of sets $U_1, \ldots, U_k$ that partition $\mathcal{U}$, i.e. $\bigcup U_i = \mathcal{U}$ and $U_i \cap U_j = \emptyset$ for all $i \neq j$. We can construct a matroid by letting $\mathcal{U}$ be the ground set and defining $S \subseteq \mathcal{U}$ to be independent iff $|S \cap U_i| \leq 1$ for all $i \in [k]$. The bases of this matroid are sets containing exactly one element from each U_i. This is called a *partition matroid*.

3. Consider a graph $G(V, E)$, where V is the vertex set and E the edge set. We can construct a matroid by letting E be the ground set and defining a subset of edges to be independent iff it contains no cycles. The bases of this matroid are all spanning forests of G. This is called a *graphic matroid*.

4. Consider a graph $G(V, E)$, and let $U, W \subseteq V$ be two distinct vertex sets ($U \cap W = \emptyset$). We can construct a matroid by letting U be the ground set and defining a subset $S \subseteq U$ to be independent iff there exist $|S|$ vertex-disjoint paths from W to U. This matroid is called a *gammoid*.

5. Consider a bipartite graph $G = (U, V; E)$. We can construct a matroid M by letting U be the ground set and defining a subset $S \subseteq U$ to be independent iff there is a bipartite matching between V and S. The bases of this matroid each correspond to a maximal matching in G. This is called a *transversal matroid*, and is a special case of a gammoid.