Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
Σχολή Θετικών Επιστημών
Τμήμα Μαθηματικών
Μεταπτυχιακό πρόγραμμα στη Λογική
και Θεωρία Αλγορίθμων και Υπολογισμού
$\mu\Pi\lambda\forall$

Υπολογιστική Σχεδίαση Παιγνίων

Διπλωματική Εργασία

Επιμέλεια: Νίκος Σαλαμάνος

Επιβλέπων καθηγητής: Ηλίας Κουτσουπιάς

Πρόλογος

Η περιοχή του Mechanism Design (Σχέδιο Μηχανισμού) ή Θεωρία Υλοποίησης (Implementation Theory) είναι υποπεριοχή της θεωρίας παιγνίων και των μικροοικονομικών και ασχολείται με την σχεδίαση πρωτοκόλλων για λογικούς (rational) παίχτες. Τα τελευταία χρόνια με την μεγάλη ανάπτυξη του internet προέκυψαν νέες εφαρμογές του δικτύου όπως το ηλεκτρονικό εμπόριο, οι on-line ψηφοφορίες, οι on-line πλειστηριασμοί κ.τ.λ. που δεν είχαν μελετηθεί στο παρελθόν. Έτσι ολοένα και περισσότερο παρουσιάζεται η ανάγκη για τον σχεδιασμό πρωτοκόλλων που θα χειρίζονται προβλήματα κατανεμημένων πληροφοριών στα οποία εμπλέκεται ένα μεγάλος αριθμός ιδιοτελών συμμετεχόντων.

Γενικά ένα πρόβλημα σχεδίου μηχανισμού μπορεί να περιγραφεί ως η επιλογή, από ένα σύνολο εφικτών παιγνίων, εκείνου του παιγνίου που θα δώσει τα αποτελέσματα που επιθυμεί ο σχεδιαστής του μηχανισμού (παιγνίου). Μας ενδιαφέρουν οι φιλαλήθεις μηχανισμοί, δηλαδή οι μηχανισμοί στους οποίους είναι πάντα προς όφελος των παιχτών να είναι ειλικρινείς ως προς τις πληροφορίες που αποκαλύπτουν στον μηχανισμό. Έτσι επικεντρώνουμε την μελέτη μας σε μία ειδική κατηγορία μηχανισμών τους VCG μηχανισμούς οι οποίοι είναι πάντα φιλαλήθεις.

Στην εργασία αυτή ασχοληθήκαμε με τη σχεδίαση μηχανισμών (παιγνίων) που υλοποιούν προβλήματα της θεωρητικής πληροφορικής.

Στο Κεφ. 1 δίνουμε τους βασικούς ορισμούς που αφορούν τους μηχανισμούς και ειδικότερα τους VCG μηχανισμούς με τους οποίους κυρίως θα ασχοληθούμε.

Στο Κεφ. 2 μελετούμε αλγοριθμικά σχέδια μηχανισμών και εφαρμογές τους. Στο πρώτο μέρος του κεφαλαίου αναλύουμε μηχανισμούς που υλοποιούν το task scheduling πρόβλημα. Στο δεύτερο μέρος μελετούμε τον υπολογισμό των VCG πληρωμών για το πρόβλημα του shortest path routing σε ένα δίκτυο επικοινωνίας.

Στο Κεφ. 3 αρχικά μελετούμε τον σχεδιασμό υπολογιστικά εφικτών VCG μηχανισμών για συνδυαστικούς πλειστηριασμούς και προβλήματα ελαχιστοποίησης κόστους που είναι προβλήματα υπολογιστικώς δύσκολα. Στη συνέχεια εξετάσουμε την εφαρμογή των μηχανισμών σε multicast transmissions σε δίκτυα επικοινωνίας, καθώς και σε on-line πλειστηριασμούς.

Ιούλιος 2002
Νίκος Σαλαμάνος

Περιεχόμενα

Πρόλογος **1**

Περιεχόμενα **2**

Κεφάλαιο 1 Mechanism Design **5**

1.1 Σχεδίαση Μηχανισμών **5**

1.1.1 Mechanism Design Προβλήματα 8

1.1.2 Ο Μηχανισμός 9

1.1.3 Vickrey-Groves-Clarke Μηχανισμοί 11

Κεφάλαιο 2 Αλγοριθμική Σχεδίαση Μηχανισμών **15**

2.1 Task Scheduling **15**

2.1.1 Το Πρόβλημα 15

2.1.2 Ο Μηχανισμός Min Work 17

2.1.3 Κάτω Φράγματα για το Task Scheduling 18

2.1.4 Πιθανοτικοί Μηχανισμοί 22

2.1.5 Μηχανισμοί με Επαλήθευση 25

2.1.6 Περιορισμένο Task Scheduling Πρόβλημα 31

2.1.7 Μηχανισμοί Πολυωνυμικού Χρόνου 34

2.2 Shortest Path **36**

2.2.1 Εισαγωγή 36

2.2.2 Το Μοντέλο 40

2.2.3 Path Graphs 41

2.2.4 Μη Κατευθυνόμενα Γραφήματα 44

2.2.5 Κατευθυνόμενα Δίκτυα 46

2.3 Συμπεράσματα, μελλοντικές κατευθύνσεις **49**

Κεφάλαιο 3 Υπολογιστικά Εφικτοί Μηχανισμοί **52**

3.1 Υπολογιστικά Εφικτοί Μηχανισμοί **52**

3.1.1 Εισαγωγή 52

3.1.2 Συνδυαστικοί Πλειστηριασμοί 54

3.1.3 Προβλήματα Ελαχιστοποίησης Κόστους 59

3.1.4 Υπολογιστικά Εφικτοί VCG Μηχανισμοί 63
3.1.5 Μέθοδοι Υλοποίησης 70

3.2 Multicast Transmissions 72
3.2.1 Εισαγωγή 72
3.2.2 Μοντέλο Δικτύου 74
3.2.3 Cost-Sharing Μηχανισμοί 74
3.2.4 Ο Marginal Cost Μηχανισμός 78
3.2.5 Ο Μηχανισμός Sharpley Value 85

3.3 Online Πλειστηριασμοί 87
3.3.1 Εισαγωγή 87
3.3.2 Το Μοντέλο 88
3.3.3 Supply Curves σε On-Line Πλειστηριασμούς 89
3.3.4 Διαιρέσιμα Αντικείμενα 93
3.3.5 Αδιαίρετα Αντικείμενα 95

3.4 Συμπεράσματα, μελλοντικές κατευθύνσεις 97

Βιβλιογραφία 100

Κεφάλαιο 1
Mechanism Design

Κεφάλαιο 1
Mechanism Design

1.1 Σχεδίαση Μηχανισμών

Η περιοχή του Mechanism Design (Σχέδιο Μηχανισμού) ή Θεωρία Υλοποίησης (Implementation Theory) είναι υποπεριοχή της θεωρίας παιγνίων και των μικροοικονομικών και ασχολείται με την σχεδίαση πρωτοκόλλων για λογικούς (rational) παίχτες. Ως λογικούς θεωρούμε τους παίχτες που επιλέγουν πάντα εκείνη την στρατηγική που θα μεγιστοποιήσει το προσωπικό τους όφελος, δηλαδή την καλύτερη στρατηγική για το παίγνιο που συμμετέχουν. Δεν θα εξετάσουμε την περίπτωση όπου οι παίχτες επιλέγουν σκόπιμα στρατηγικές ενάντια στην διεξαγωγή του παιγνίου αλλά υποθέτουμε ότι επιθυμούν την εύρυθμη διεξαγωγή του.

Γενικά ένα πρόβλημα σχεδίου μηχανισμού μπορεί να περιγραφεί ως η επιλογή, από ένα σύνολο εφικτών παιγνίων, εκείνου του παιγνίου που θα δώσει τα αποτελέσματα που επιθυμεί ο σχεδιαστής του μηχανισμού. Πιο συγκεκριμένα, ο σχεδιαστής πρέπει να σχεδιάσει ένα παίγνιο, στο οποίο συμμετέχει ένα συνήθως μεγάλο σύνολο ατόμων (παιχτών) και το οποίο να δίνει λύσεις σε κάποιο συγκεκριμένο υπαρκτό πρόβλημα. Τέτοιου είδους προβλήματα είναι οι πλειστηριασμοί με κλειστές προσφορές, η ανάθεση ενός συνόλου εργασιών με σκοπό τον μικρότερο χρόνο εκτέλεσης (το make-span), δρομολόγηση σε ένα δίκτυο με το μικρότερο κόστος, μετάδοση πληροφορίας ή εμπορικών αγαθών όπως π.χ. ταινίες μέσω του Internet και ο καταμερισμός του κόστους μετάδοσης, οι on-line πλειστηριασμοί κ.τ.λ. Το βασικό χαρακτηριστικό αυτών των προβλημάτων είναι ότι ο σχεδιαστής αγνοεί βασικές πληροφορίες για τους παίχτες οι οποίες του είναι απαραίτητες για να λύσει το πρόβλημα. Οι πληροφορίες αυτές αφορούν τους παίχτες είναι δηλαδή ιδιωτικές και με κανένα τρόπο ο σχεδιαστής δεν μπορεί εκ' των προτέρων να τις γνωρίζει. Συνεπώς η λύση του προβλήματος θα βασίζεται στις πληροφορίες που θα δώσουν οι ίδιοι οι παίχτες στον σχεδιαστή.

Παράδειγμα 1 (Ανάθεση Εργασιών): Για παράδειγμα έστω ότι ο σχεδιαστής πρέπει να αναθέσει ένα σύνολο εργασιών σε ένα σύνολο ατόμων και σκοπός του είναι να ελαχιστοποιήσει τον χρόνο αποπεράτωσης της τελευταίας εργασίας (το make-span). Ο σχεδιαστής για να δώσει λύση στο πρόβλημα πρέπει να γνωρίζει τους χρόνους εκτέλεσης των εργασιών από τα άτομα. Τους χρόνους αυτούς όμως δεν τους ξέρει. Αν τους γνώριζε θα μπορούσε να εφαρμόσει κάποιο off-line βέλτιστο ή προσεγγιστικό αλγόριθμο (το πρόβλημα είναι NP-hard) και να λύσει το πρόβλημα. Με σκοπό να αντιμετωπίσει την έλλειψη πληροφορίας βλέπει το πρόβλημα ως παίγνιο και τα άτομα ως παίχτες. Έτσι πρέπει να σχεδιάσει ένα παίγνιο μέσω του οποίου θα αναγκάσει τους παίχτες να του δηλώσουν τους χρόνους στους οποίους μπορούν να εκτελέσουν τις εργασίες. Μετά χρησιμοποιώντας έναν βέλτιστο αλγόριθμο (αλγόριθμο εξόδου) θα δώσει την

λύση στο πρόβλημα. Ένα τέτοιο παίγνιο θα το καλούμε μηχανισμό.

Πολλά είναι τα ερωτήματα που προκύπτουν από μία τέτοια προσέγγιση. Πως μπορεί ο σχεδιαστής να είναι σίγουρος ότι οι παίχτες είναι ειλικρινείς; Καθώς θεωρούμε ότι οι παίχτες είναι ιδιοτελείς, δηλαδή ενδιαφέρονται για το προσωπικό τους όφελος, υπάρχει περίπτωση να δηλώσουν ψευδείς χρόνους στον σχεδιαστή με σκοπό να αναλάβουν λιγότερες εργασίες ή και καμία. Έτσι μία βασική αρχή είναι ότι το παίγνιο θα πρέπει να είναι έτσι σχεδιασμένο που να αναγκάζει τους παίχτες να είναι φιλαλήθεις (truthful) και πάντα να δηλώνουν τους πραγματικούς χρόνους εκτέλεσης των εργασιών.

Οι παίχτες από την πλευρά τους μελετούν το παίγνιο που όρισε ο σχεδιαστής και προσπαθούν να βρουν την καλύτερη στρατηγική ώστε να μεγιστοποιήσουν το προσωπικό τους όφελος δηλαδή το συνολικό χρόνο που θα εργαστούν και το ποσό που θα πληρωθούν. Θεωρούμε ότι οι παίχτες έχουν άπειρη υπολογιστική δύναμη. Συνεπώς μπορούν να μελετήσουν πλήρως τον μηχανισμό καθώς και τα αποτελέσματα που θα δώσει ο αλγόριθμος εξόδου για κάθε πιθανή στρατηγική των παιχτών. Επιπλέον οι παίχτες θεωρούμε ότι είναι λογικοί και πάντα θα επιλέξουν την καλύτερη στρατηγική για το παίγνιο. Για να είναι οι παίχτες φιλαλήθεις πρέπει το παίγνιο να σχεδιαστεί έτσι ώστε οι παίχτες μελετώντας το να καταλήγουν στο συμπέρασμα ότι το όφελός τους μεγιστοποιείται μόνο αν είναι ειλικρινείς και σε καμία άλλη περίπτωση. Δηλαδή να είναι η στρατηγική της ειλικρίνειας η καλύτερη στρατηγική γι' αυτούς (κυρίαρχη στρατηγική-dominant strategy). Αν επιτευχθεί αυτό τότε το παίγνιο θα καταλήξει στο επιθυμητό σημείο ισορροπίας (equilibrium) όπου όλοι οι παίχτες θα έχουν επιλέξει την στρατηγική της ειλικρίνειας και θα παραμένουν σε αυτή. Ένας τέτοιος μηχανισμός θα καλείται φιλαλήθης. Ένας λογικός τρόπος για να επιτύχουμε κάτι τέτοιο είναι μέσω της συνάρτησης πληρωμών των παιχτών, όπου η πληρωμή που θα ορίζεται για κάθε παίχτη για τη συμμετοχή του στο παίγνιο δεν θα εξαρτάται από το τι δήλωσε ο ίδιος αλλά από τις δηλώσεις των άλλων παιχτών. Η συνάρτηση πληρωμών θα ορίζεται από τον μηχανισμό. Όπως θα δούμε στη συνέχεια μηχανισμοί με αυτή την λογική είναι οι Vickrey-Groves-Clarke Μηχανισμοί (VCG μηχανισμοί).

Παράδειγμα 2 (Vickrey Πλειστηριασμοί): Ένα κλασικό παράδειγμα VCG μηχανισμού είναι οι Vickrey πλειστηριασμοί. Έστω ότι έχουμε τον πλειστηριασμό ενός έργου τέχνης π.χ. ενός πίνακα, και ένα σύνολο πλειοδοτών που τον επιθυμούν. Οι πλειοδότες γνωρίζουν φυσικά το ποσό που είναι διατεθειμένοι να πληρώσουν αλλά ο πλειστηριαστής δεν το γνωρίζει. Ο πλειστηριασμός συνεπώς δεν είναι με ανοιχτές προσφορές. Ο πλειστηριαστής μπορεί να μάθει τις προσφορές των πλειοδοτών μόνο και αν τις αποκαλύψουν οι ίδιοι σε αυτόν. Για παράδειγμα μπορεί ο πλειστηριασμός να γίνεται μέσω του Internet και ο πλειστηριαστής να μην έχει οποιαδήποτε πληροφορία για τους πλειοδότες. Ο πωλητής λοιπόν πρέπει να σχεδιάσει ένα μηχανισμό (ένα παίγνιο) που να υλοποιεί το πρόβλημα αυτό και να δίνει την βέλτιστη λύση, που είναι η πώληση του πίνακα στην μέγιστη δυνατή τιμή. Ο μηχανισμός θα πρέπει να είναι σχεδιασμένος κατά τέτοιο τρόπο που να εγγυάται ότι οι πλειοδότες αναφέρουν το πραγματικό ποσό που είναι διατεθειμένοι να πληρώσουν. Είναι πιθανό το

ενδεχόμενο οι πλειοδότες να δηλώσουν ψευδές ποσό με σκοπό να πληρώσουν λιγότερο για τον πίνακα. Συνεπώς ο μηχανισμός θα πρέπει, όπως και στην περίπτωση της ανάθεσης εργασιών, να είναι φιλαλήθης. Πρέπει να είναι έτσι σχεδιασμένος που οι πλειοδότες μελετώντας τον να καταλήγουν στο συμπέρασμα ότι θα κερδίσουν μόνο αν αποκαλύψουν το πραγματικό ποσό που είναι διατεθειμένοι να πληρώσουν. Αυτή θα είναι η καλύτερη στρατηγική γι' αυτούς και αυτή τη στρατηγική θα ακολουθήσουν. Ένα τέτοιο αποτέλεσμα μπορεί να επιτευχθεί μέσω μίας συνάρτησης πληρωμής που θα καθορίζει το ποσό που θα πληρώσει ο τελικός αγοραστής ανεξάρτητα από το ποσό που δήλωσε ο ίδιος.

Ο πωλητής λοιπόν καλεί τους πλειοδότες να του δηλώσουν τα ποσά που είναι διατεθειμένοι να πληρώσουν με κλειστές προσφορές. Το ποσό που θα δηλώσει κάθε παίχτης θα το γνωρίζει μόνο ο πωλητής και κανείς άλλος. Ορίζει ότι αυτός που τελικά θα επιλεγεί και θα πάρει τον πίνακα, δεν θα πληρώσει το ποσό που δήλωσε άλλα το ποσό της αμέσως μικρότερης προσφοράς. Αυτή είναι η συνάρτηση πληρωμής του μηχανισμού. Κατ' αυτόν τον τρόπο κανένα παίχτη δεν τον συμφέρει να ψεύδεται. Αν ο παίχτης i κάνει μικρότερη προσφορά από αυτή που μπορεί, τότε είναι πιθανό να χάσει τον πίνακα και να ωφεληθεί κάποιος άλλος παίχτης j που τελικά θα πάρει τον πίνακα πληρώνοντας την τιμή που ψευδώς δήλωσε ο i. Αν ο παίχτης i δηλώσει ψευδώς ένα μεγαλύτερο ποσό τότε ίσως να πάρει τον πίνακα αλλά υπάρχει το ενδεχόμενο να πρέπει να πληρώσει ένα ποσό μεγαλύτερο από αυτό που μπορεί. Αν ο παίχτης δηλώσει το αληθές ποσό τότε ή θα πάρει τον πίνακα κερδίζοντας την διαφορά τιμής από τον προηγούμενο ή δεν θα τον πάρει καθόλου, επειδή υπήρχαν καλύτερες προσφορές τις οποίες δεν θα μπορούσε να συναγωνιστεί.

Ο μηχανισμός που παρουσιάσαμε είναι ένα τυπικό παράδειγμα VCG μηχανισμού. Για όλα τα προβλήματα που αναφέραμε προηγουμένως μπορούν να σχεδιαστούν VCG μηχανισμοί. Οι VCG μηχανισμοί είναι οι πιο αποτελεσματικοί μηχανισμοί διότι είναι πάντα φιλαλήθεις, γί αυτό και θα τους μελετήσουμε εκτενώς στα επόμενα κεφάλαια Με την χρήση τέτοιων μηχανισμών θα προσπαθήσουμε να υλοποιήσουμε προβλήματα που ανήκουν στην περιοχή της θεωρητικής πληροφορικής όπως είναι το shortest path και το task scheduling, καθώς και προβλήματα που εμφανίζονται σε δίκτυα επικοινωνίας..

Το κύριο αντικείμενο μελέτης της εργασίας αυτής είναι η υπολογιστική πλευρά των μηχανισμών. Ένας μηχανισμός μπορεί να υλοποιεί επιτυχώς ένα πρόβλημα αλλά υπολογιστικά να είναι ανέφικτος. Το πρόβλημα αυτό εμφανίζεται κυρίως όταν ο μηχανισμός υλοποιεί ένα πρόβλημα NP-complete. Για προβλήματα NP-complete και σχετικά μεγάλο αριθμό παιχτών, ο βέλτιστος αλγόριθμος εξόδου του μηχανισμού θα είναι εκθετικού χρόνου. Ένα βασικό ερώτημα είναι αν μπορούμε να χρησιμοποιήσουμε ένα προσεγγιστικό αλγόριθμο εξόδου. Αυτό το ερώτημα είναι πολύ σημαντικό διότι μπορεί να έχουμε σχεδιάσει κάποιο φιλαλήθη μηχανισμό που να υλοποιεί επιτυχώς κάποιο NP-complete πρόβλημα αλλά ο μηχανισμός στην πράξη να είναι ανεφάρμοστος. Θα δείξουμε ότι η χρήση προσεγγιστικών αλγορίθμων εξόδου οδηγεί σε μηχανισμούς που παρουσιάζουν ανώμαλη συμπεριφορά και δεν είναι φιλαλήθεις.

Τέλος η εφαρμογή της θεωρίας σχεδίασης μηχανισμών σε προβλήματα της

θεωρητικής πληροφορικής δεν είναι καθόλου τετριμμένη. Δύο είναι οι βασικοί λόγοι της δυσκολίας αυτής. Ο πρώτος έχει να κάνει με το ότι η θεωρία του mechanism design έχει αναπτυχθεί για προβλήματα των οικονομικών. Τα πρόβλημα αυτά έχουν συνήθως διαφορετικούς στόχους και υποθέσεις από εκείνα στην επιστήμη των υπολογιστών. Επίσης η θεωρία του mechanism design δεν ασχολείται με την υπολογιστική πλευρά των πρωτοκόλλων. Έτσι πολλές λύσεις μπορεί να είναι πρακτικά ανέφικτες στην υλοποίηση τους. Στη συνέχεια θα δούμε με ποιους τρόπους μπορούμε να ξεπεράσουμε αυτές τις δυσκολίες.

1.1.1 Mechanism Design Προβλήματα

Ένα πρόβλημα σχεδίου μηχανισμού (mechanism design problem) έχει δύο βασικά συστατικά: την αλγοριθμική περιγραφή της εξόδου (output) και την περιγραφή του τι επιθυμούν οι παίχτες που συμμετέχουν , η οποία δίνεται από τις utility συναρτήσεις (συναρτήσεις ωφελείας) πάνω στο σύνολο των πιθανών εξόδων (outputs).

Για παράδειγμα στο πρόβλημα ανάθεσης εργασιών που αναφέραμε προηγουμένως, η αλγοριθμική περιγραφή της εξόδου είναι ο βέλτιστος αλγόριθμος που θα χρησιμοποιεί ο μηχανισμός για να υπολογίσει την βέλτιστη ανάθεση των εργασιών στους παίχτες, αφού πρώτα έχει συλλέξει τις δηλώσεις των παιχτών όσων αφορά τους χρόνους εκτέλεσης των εργασιών. Η utility συνάρτηση κάθε παίχτη είναι η συνάρτηση του προσωπικού του οφέλους. Οι τιμές τις προκύπτουν από τον συνδυασμό των χρόνων στους οποίους μπορεί να εκτελέσει ο παίχτης κάθε πιθανό υποσύνολο των εργασιών, δηλαδή κάθε έξοδο (ανάθεση) του μηχανισμού, καθώς και από την πληρωμή του παίχτη για κάθε πιθανή έξοδο.

Ορισμός 1.1.1 (Πρόβλημα Σχεδίου Μηχανισμού) Ένα *Πρόβλημα Σχεδίου Μηχανισμού (Mechanism Design Problem)* ορίζεται από την περιγραφή της εξόδου και από το σύνολο των utilities των παιχτών. Συγκεκριμένα:

1. Έχουμε n παίχτες, και κάθε παίχτης i έχει στη διάθεση του μία προσωπική είσοδο (input) $t^i \in T^i$ που καλείται *τύπος (type)* του παίχτη, την οποία γνωρίζει μόνο ο ίδιος. Οτιδήποτε άλλο θεωρείται γνωστό σε όλους
2. Έχουμε το πεπερασμένο σύνολο O των επιτρεπτών εξόδων του μηχανισμού (outputs).
3. Η περιγραφή της εξόδου, απεικονίζει σε κάθε διάνυσμα τύπων $t = t^1 \ldots t^n$ ένα σύνολο από επιτρεπόμενες εξόδους $o \in O$.
4. Οι προτιμήσεις κάθε παίχτη i δίνονται από μια πραγματική συνάρτηση $v^i(t^i, o)$, που καλείται *αποτίμηση (valuation*) του παίχτη. Είναι ο προσδιορισμός (βάση κάποιου κοινού νομίσματος) της τιμής του από την έξοδο o, όταν ο τύπος του είναι ο t^i. Συγκεκριμένα αν η έξοδος του μηχανισμού είναι ο και ο μηχανισμός δώσει στον παίχτη p^i μονάδες (από το νόμισμα που έχει επιλεγεί) τότε η *utility* του παίχτη θα είναι $u^i = p^i + v^i(o, t^i)$. Αυτή την ποσότητα επιθυμεί να βελτιστοποιήσει ο παίχτης.

Παράδειγμα: Για το πρόβλημα της ανάθεσης εργασιών:

1. Ο τύπος κάθε παίχτη είναι οι ελάχιστοι χρόνοι στους οποίους μπορεί να εκτελέσει κάθε εργασία και τους οποίους γνωρίζει μόνο ο ίδιος.
2. Εφικτή έξοδος του μηχανισμού θα είναι οποιαδήποτε ανάθεση στους παίχτες, δηλαδή οποιαδήποτε διαμέριση των εργασιών.
3. Το πεπερασμένο σύνολο εξόδων Ο των επιτρεπτών εξόδων είναι όλες οι πιθανές διαμερίσεις των εργασιών.
4. Η περιγραφή της εξόδου είναι η αντικειμενική συνάρτηση του προβλήματος. Έτσι για κάθε διάνυσμα δηλούμενων χρόνων έχουμε ένα αντίστοιχο σύνολο αποδεκτών αναθέσεων.
5. Η αποτίμηση κάθε παίχτη, όπως θα δούμε στο Κεφ. 2, θα οριστεί ως η αρνητική τιμή του του συνολικού χρόνου που δαπανά για να εκτελέσει τις εργασίες της συγκεκριμένης ανάθεσης που επιλέχθηκε. Γι' αυτό και η αποτίμηση κάθε παίχτη εξαρτάται από την έξοδο του μηχανισμού, αλλά και από τους χρόνους που δήλωσε. Για διαφορετικές εξόδους έχουμε και διαφορετικές αποτιμήσεις.
6. Η συνάρτηση πληρωμής θα καθορίζεται από το σχέδιο του μηχανισμού.

Στα επόμενα κεφάλαια θα ασχοληθούμε κυρίως με προβλήματα βελτιστοποίησης. Τα προβλήματα αυτά χωρίζονται σε προβλήματα ελαχιστοποίησης και προβλήματα μεγιστοποίησης. Η έξοδος των προβλημάτων αυτών ορίζεται από την αντικειμενική συνάρτηση του προβλήματος. Στόχος μας είναι η βελτιστοποίηση της αντικειμενικής συνάρτησης του προβλήματος.

Θα ορίσουμε στη συνέχεια τα προβλήματα βελτιστοποίησης (ελαχιστοποίησης) σχεδίου μηχανισμού.

Ορισμός 1.1.2 (Πρόβλημα Βελτιστοποίησης Σχεδίου Μηχανισμού) (Mechanism Design Optimization Problem) Είναι ένα πρόβλημα σχεδίου μηχανισμού όπου η περιγραφή της εξόδου δίνεται από μία θετική πραγματική αντικειμενική συνάρτηση $g(o,t)$ και από ένα σύνολο εφικτών εξόδων (feasible outputs) F. Για την περίπτωση της βέλτιστης λύσης, απαιτούμε η έξοδος $o \in F$ να ελαχιστοποιεί το g.
Ανάλογος είναι ο ορισμός για προβλήματα μεγιστοποίησης.

Παράδειγμα: Ένα παράδειγμα είναι το πρόβλημα ανάθεσης εργασιών. Η έξοδος οποιουδήποτε μηχανισμό που υλοποιεί το πρόβλημα, θα είναι η αντικειμενική συνάρτηση g που θα δίνει make-span. Αυτή είναι και η περιγραφή της εξόδου του μηχανισμού. Το σύνολο των εφικτών λύσεων είναι όλες οι πιθανές αναθέσεις. Όταν αναζητούμε την βέλτιστη λύση, τότε ο αλγόριθμος εξόδου του μηχανισμού θα πρέπει να μας δίνει την ανάθεση που ελαχιστοποιεί το make-span.

1.1.2 Ο Μηχανισμός

Διαισθητικά ο μηχανισμός (mechanism) επιλύει ένα πρόβλημα με το να εξασφαλίζει ότι οι ζητούμενες έξοδοι προκύπτουν όταν οι παίχτες επιλέγουν την στρατηγική τους με σκοπό να μεγιστοποιήσουν τις προσωπικές τους utilities.

Αυτό προκύπτει από το ότι θεωρούμε ότι οι παίχτες είναι λογικοί και κατά συνέπεια θα ενδιαφέρονται να βελτιστοποιήσουν το προσωπικό τους όφελος. Ο σχεδιασμός του μηχανισμός πρέπει να στηρίζεται σε αυτό για να μπορεί να δίνει λογικά αποτελέσματα.

Σημείωση: Με α^{-i} θα συμβολίζουμε το $(\alpha^1,\ldots,\alpha^{i-1},\alpha^{i+1},\ldots,\alpha^n)$ και με (α^i,α^{-i}) θα συμβολίζουμε το $(\alpha^1,\ldots,\alpha^n)$.

Ορισμός 1.1.3 (Μηχανισμός) Ο μηχανισμός $m=(o,p)$ αποτελείται από δύο στοιχεία: Από την συνάρτηση εξόδου $o(\ )$ και από την n-άδα των πληρωμών (payments) $p^1(\ )\ldots p^n(\ )$. Συγκεκριμένα:

1. Ο μηχανισμός ορίζει για κάθε παίχτη i μία οικογένεια στρατηγικών A^i.Ο παίχτης μπορεί να επιλέξει οποιοδήποτε $\alpha^i \in A^i$.
2. Ο μηχανισμός αρχικά παρέχει την συνάρτηση εξόδου $o=o(\alpha^1\ldots\alpha^n)$.
3. Μετά καθορίζει την συνάρτηση πληρωμής $p^i=p^i(\alpha^1\ldots\alpha^n)$ για κάθε παίχτη.
4. Θα λέμε ότι ο μηχανισμός είναι ***υλοποίηση με κυρίαρχες στρατηγικές*** *(implementation with dominant strategies)* ή για συντομία ***υλοποίηση*** *(implementation)*, εάν :
   I. Για κάθε παίχτη i και κάθε t^i υπάρχει στρατηγική $\alpha^i \in A^i$, που θα καλούμε *κυρίαρχη (dominant)*, έτσι ώστε για όλες τις δυνατές στρατηγικές α^{-i} των άλλων παιχτών, το α^i μεγιστοποιεί την utility του παίχτη i.
   Συγκεκριμένα: αν για κάθε $\alpha'^i \in A^i$, ορίσουμε:
   $o=o(\alpha^i,\alpha^{-i}),\ o'=o(\alpha'^i,\alpha^{-i}),\ p^i=p^i(\alpha^i,\alpha^{-i}),\ p'^i=p^i(\alpha'^i,\alpha^{-i})$, τότε
   $\upsilon^i(t^i,o)+p^i \geq \upsilon^i(t^i,o')+p'^i$
   II. Για κάθε n-άδα $\alpha=(\alpha^1\ldots\alpha^n)$ κυρίαρχων στρατηγικών η έξοδος $o(\alpha)$ ικανοποιεί την περιγραφή της εξόδου.

Θα λέμε ότι ο μηχανισμός είναι ***πολυωνυμικού χρόνου*** αν οι συναρτήσεις εξόδου και πληρωμής είναι υπολογίσιμες σε πολυωνυμικό χρόνο.

Ένα παράδειγμα μηχανισμού με κυρίαρχες στρατηγικές αναφέραμε προηγουμένως για τους πλειστηριασμούς κατά Vickrey. Στο μηχανισμό που αναφέραμε οι παίχτες μεγιστοποιούν το όφελος τους μόνο αν δηλώσουν το πραγματικό ποσό που είναι διατεθειμένοι να πληρώσουν και σε καμία άλλη περίπτωση. Συνεπώς αυτή η στρατηγική είναι κυρίαρχη.

1.1.3 Vickrey-Groves-Clarke Μηχανισμοί

Ένα από τα σημαντικότερα αποτελέσματα στην περιοχή του mechanism design είναι οι Vickrey-Groves-Clarke Μηχανισμοί (VCG). Θα ξεκινήσουμε δίνοντας αρχικά κάποιους ορισμούς για να καταλήξουμε στον ορισμό των VCG μηχανισμών.

Ένας απλός τύπος μηχανισμού είναι εκείνος όπου η στρατηγική των παιχτών είναι απλά η αποκάλυψη των τύπων τους στον μηχανισμό.

Ορισμός 1.1.4 (Φιλαλήθης Υλοποίηση)
Θα λέμε ότι ένας μηχανισμός είναι *φιλαλήθης (truthful)* εάν:

1. Για όλα τα i και όλα τα t^i, $A^i = T^i$. Δηλαδή η στρατηγική των παιχτών είναι η αποκάλυψη των τύπων τους στο μηχανισμό. Ένας τέτοιος μηχανισμός καλείται ***μηχανισμός άμεσης αποκάλυψης*** *(direct revelation mechanism)* .
2. Η στρατηγική της *φιλαλήθειας* είναι κυρίαρχη στρατηγική. Δηλαδή η $\alpha^i = t^i$ ικανοποιεί τον ορισμό της κυρίαρχης στρατηγικής.

Δηλαδή σε ένα μηχανισμό άμεσης αποκάλυψης οι παίχτες έχουν μόνο μία στρατηγική, αποκαλύπτουν τις τιμές τους στον μηχανισμό. Ο μηχανισμός θα είναι φιλαλήθης όταν η στρατηγική της φιλαλήθειας είναι κυρίαρχη, δηλαδή δίνει στους παίχτες τα καλύτερα αποτελέσματα.

Μπορεί όμως να υπάρχουν και άλλες κυρίαρχες στρατηγικές για τους παίχτες. Θα ορίσουμε τώρα τους ισχυρά φιλαλήθεις μηχανισμούς. Σε αυτούς η φιλαλήθεια είναι η μόνη κυρίαρχη στρατηγική.

Ορισμός 1.1.5 (Ισχυρά Φιλαλήθης Μηχανισμός) Ένας μηχανισμός είναι *ισχυρά φιλαλήθης (strongly truthful),* εάν η φιλαλήθεια είναι η μοναδική κυρίαρχη στρατηγική.

Το επόμενο πόρισμα είναι πολύ σημαντικό και εκφράζει την λεγόμενη ***αρχή της αποκάλυψης.*** Σύμφωνα με την αρχή αυτή μπορούμε, χωρίς βλάβη της γενικότητας, να επικεντρώσουμε την μελέτη μας μόνο στις φιλαλήθεις υλοποιήσεις.

Πρόταση 1.1.6 Εάν υπάρχει μηχανισμός που υλοποιεί ένα πρόβλημα με κυρίαρχες στρατηγικές τότε υπάρχει και φιλαλήθης υλοποίηση για το πρόβλημα .

Ορισμός 1.1.7 Ένα πρόβλημα μεγιστοποίησης σχεδίου μηχανισμού (maximization mechanism design problem) θα καλείται ***ωφελιμιστικό*** *(utilitarian)* εάν η αντικειμενική συνάρτηση ικανοποιεί την σχέση:

$$g(o,t) = \sum_i v^i(t^i, o).$$

Παρατήρηση: Ο όρος ωφελιμιστικό έχει την εξής έννοια. Όπως αναφέραμε οι

τύποι t^i των παιχτών είναι οι τιμές τους για το πρόβλημα. Από τους τύπους των παιχτών καθορίζεται η έξοδος ο του μηχανισμού. Η αποτιμήσεις $\upsilon^i(t^i, o)$ των παιχτών είναι ουσιαστικά ένας ποσοτικός ο προσδιορισμός των προτιμήσεων τους και είναι συνάρτηση της εξόδου ο και των τύπων τους t^i. Όταν η αντικειμενική συνάρτηση του προβλήματος είναι το άθροισμα των αποτιμήσεων των παιχτών τότε αν επιτύχουμε την μεγιστοποίηση της g θα έχουμε και μεγιστοποίηση των προτιμήσεων των παιχτών.

Με βάση τα παραπάνω μπορούμε να δώσουμε τον ορισμό των VCG-μηχανισμών που είναι και οι ποιο σημαντικοί μηχανισμοί στην περιοχή του Mechanism Design και εφαρμόζονται σε ωφελιμιστικά προβλήματα σχεδίου μηχανισμού.

Ορισμός 1.1.8 (VCG Μηχανισμός)
Ένας μηχανισμός άμεσης αποκάλυψης $m = (o(t), p(t))$ θα ανήκει στην οικογένεια των VCG μηχανισμών εάν ισχύει:

1. $o(t) \in \arg\max_o \left(\sum_{i=1}^n \upsilon^i\left(t^i, o\right)\right)$.

2. $p^i(t) = \sum_{j \neq i} \upsilon^j\left(t^j, o(t)\right) + h^i\left(t^{-i}\right)$ όπου $h^i(\ )$ είναι αυθαίρετη συνάρτηση των t^{-i}.

Σχόλια:
1. Στον παραπάνω ορισμό η συνθήκη 1 υποδηλώνει ότι η έξοδος του μηχανισμού πάντα θα μεγιστοποιεί την αντικειμενική συνάρτηση του προβλήματος που είναι το άθροισμα των αποτιμήσεων των παιχτών.
2. Η συνθήκη 2 δηλώνει ότι η πληρωμή κάθε παίχτη i δεν εξαρτάται από την αποτίμηση του αλλά από τις αποτιμήσεις των άλλων παιχτών καθώς και από μία αυθαίρετη συνάρτηση $h^i(\ )$ η οποία έχει ως πεδίο ορισμού το σύνολο των τύπων των άλλων παιχτών, δηλαδή οι τιμές τις δεν εξαρτώνται από τον τύπο του i. Αυτή ακριβώς η μορφή των συναρτήσεων πληρωμής των VCG μηχανισμών είναι που δίνει το ζητούμενο αποτέλεσμα, δηλαδή φιλαλήθεις μηχανισμούς. Έτσι ο σχεδιαστής μπορεί να είναι σίγουρος ότι οι πληροφορίες που παίρνει από τους παίχτες είναι σωστές, ξεπερνώντας με επιτυχία την αρχική έλλειψη πληροφορίας.

Το επόμενο θεώρημα μας εγγυάται την φιλαλήθεια των VCG μηχανισμών.

Θεώρημα 1.1.9 [Groves 1973] Ο VCG μηχανισμός είναι φιλαλήθης.

Κατά συνέπεια ο VCG μηχανισμός μας δίνει λύσεις για κάθε ωφελιμιστικό πρόβλημα σχεδίου μηχανισμού.

Ορισμός 1.1.10 (*Ωφελιμιστικό με βάρη*) Ένα πρόβλημα μεγιστοποίησης

σχεδίου μηχανισμού θα καλείται *ωφελιμιστικό με βάρη (weighted utilitarian)* εάν υπάρχουν πραγματικοί αριθμοί $\beta^1,\ldots,\beta^n > 0$ έτσι ώστε η αντικειμενική συνάρτηση του προβλήματος να ικανοποιεί την σχέση:

$$g(o,t) = \sum_i \beta^i \cdot v^i\left(t^i, o\right).$$

Η έννοια του ωφελιμιστικού με βάρη είναι όμοια με τον ορισμό 1.17.

Ομοίως με τους VCG μηχανισμούς ορίζονται και οι VCG μηχανισμοί με βάρη.

Ορισμός 1.1.11 (VCG μηχανισμοί με βάρη)
Ένας μηχανισμός άμεσης αποκάλυψης θα ανήκει στην οικογένεια των *VCG μηχανισμών με βάρη*, εάν ισχύει :

1. $o(t) \in \arg\max_o(g(o,t))$

2. $p^i(t) = \frac{1}{\beta^i} \cdot \sum_{j\neq i} \beta^j \cdot v^j\left(t^j, o(t) + h^i\left(t^{-i}\right)\right)$ όπου $h^i(\ )$ είναι αυθαίρετη συνάρτηση των t^{-i}.

Η σημασία των συνθηκών 1 και 2 στον ορισμό αυτό είναι όμοια με του ορισμού 1.1.8

Τέλος αποδεικνύεται το ακόλουθο θεώρημα που μας εγγυάται την φιλαλήθεια των VCG μηχανισμών με βάρη.

Θεώρημα 1.1.12 Ένας VCG μηχανισμός με βάρη είναι φιλαλήθης.

Παρατήρηση: Η συνάρτηση εξόδου ενός VCG μηχανισμού πρέπει να μεγιστοποιεί την αντικειμενική συνάρτηση του προβλήματος. Σε πολλές περιπτώσεις αυτό κάνει τον μηχανισμό υπολογιστικά δύσκολο κυρίως όταν το πρόβλημα είναι NP-complete. Έτσι ένα λογικό ερώτημα είναι το κατά πόσο ένας VCG μηχανισμός παραμένει φιλαλήθης αν αντικαταστήσουμε τον βέλτιστο αλγόριθμο του μηχανισμού με κάποιο προσεγγιστικό. Όπως θα δούμε σε επόμενο κεφάλαιο, μία τέτοια αντικατάσταση οδηγεί σε μηχανισμούς με ανώμαλη συμπεριφορά.

Κεφάλαιο 2
Αλγοριθμική Σχεδίαση Μηχανισμών

Κεφάλαιο 2
Αλγοριθμική Σχεδίαση Μηχανισμών

2.1 Task Scheduling

Στις επόμενες ενότητες θα αναλύσουμε το πρόβλημα Task Scheduling με την χρήση μηχανισμών. Η ανάλυση βασίζεται στο άρθρο [NR99] των Noam Nisan και Amir Ronen καθώς και σε στοιχεία των [N99] και [R00].

2.1.1 Το Πρόβλημα

Στη συνέχεια θα μελετήσουμε το πρόβλημα ανάθεσης εργασιών με τη χρήση μηχανισμών.

Ορισμός 2.1.1 (Πρόβλημα Ανάθεσης Εργασιών) (Task Allocation Problem)
Έστω k εργασίες που πρέπει να ανατεθούν σε n παίχτες. Ο τύπος κάθε παίχτη i για την εργασία j, είναι ο ελάχιστος χρόνος t_j^i που χρειάζεται για να εκτελέσει την εργασία αυτή. Στόχος μας είναι να ελαχιστοποιήσουμε τον χρόνο αποπεράτωσης της τελευταίας ανάθεσης (make-span). Η αποτίμηση (valuation) του παίχτη i θα είναι η αρνητική τιμή του συνολικού χρόνου που δαπανά για να εκτελέσει τις εργασίες που του έχουν ανατεθεί.
Συγκεκριμένα:

- Οι εφικτές έξοδοι (outputs) του μηχανισμού είναι όλες οι διαμερίσεις $x = x^1 \ldots x^n$ των εργασιών στους παίχτες, όπου x^i είναι το σύνολο των εργασιών που έχουν ανατεθεί στον παίχτη i.
- Η αντικειμενική συνάρτηση είναι η $g(x,t) = \max_i \sum_{j \in x^i} t_j^i$
- Η αποτίμηση του παίχτη i είναι $v^i\left(x, t^i\right) = -\sum_{j \in x^i} t_j^i$

Ας δούμε ένα παράδειγμα ανάθεσης εργασιών.

Παράδειγμα: Έστω ότι έχουμε 2 παίχτες $\{A_1, A_2\}$ στους οποίους θέλουμε να αναθέσουμε 3 εργασίες $\{j_1, j_2, j_3\}$. Στον πίνακα 2.1.1 δίνουμε τους χρόνους εκτέλεσης των εργασιών.

	j_1	j_2	j_3
A_1	20	40	50
A_2	70	30	60

Πίνακας 2.1.1

Οι εφικτές έξοδοι του προβλήματος είναι όλες οι πιθανές αναθέσεις των εργασιών στους 2 παίχτες. Ας υποθέσουμε ότι ο μηχανισμός γνωρίζει τους πραγματικούς

χρόνους εκτέλεσης των εργασιών. Ο μηχανισμός με βάση τους χρόνους εκτέλεσης υπολογίζει για κάθε δυνατή ανάθεση το make-span μέσω της συνάρτησης g. Από τα αποτελέσματα της g επιλέγει εκείνη την ανάθεση για την οποία η g δίνει την ελάχιστη τιμή. Αυτή θα είναι και η τελική ανάθεση των εργασιών στους παίχτες. Για το παράδειγμα μας η καλύτερη ανάθεση είναι η A_1: $\{j_1, j_2\}$, A_2: $\{j_3\}$ με make-span 60.

Παρατηρήσεις:

1. Στον παραπάνω ορισμό η αντικειμενική συνάρτηση g μας δίνει το make-span για κάθε ανάθεση. Το make-span είναι το άθροισμα των χρόνων του παίχτη που θα τελειώσει τελευταίος. Στόχος μας είναι να βρούμε εκείνη την ανάθεση που θα ελαχιστοποιεί το άθροισμα αυτό.

2. Όπως αναφέραμε, ορίζουμε την αποτίμηση κάθε παίχτη ως την αρνητική τιμή του συνολικού χρόνου που δαπανά για να εκτελέσει τις εργασίες που του έχουν ανατεθεί. Ορίζουμε έτσι τις αποτιμήσεις διότι η utility του παίχτη i είναι $u^i = p^i + \upsilon^i\left(x, t^i\right)$ συνεπώς $u^i = p^i - \sum_{j \in x^i} t^i_j$, όπου p είναι η συνάρτηση πληρωμών του μηχανισμού. Η utility εκφράζει το όφελος του παίχτη από την έξοδο x. Οι παίχτες είναι ιδιοτελείς συνεπώς επιθυμούν να εργαστούν λίγο και να πληρωθούν πολύ. Ο ορισμός αυτός για τις αποτιμήσεις θα μας βοηθήσει να σχεδιάσουμε φιλαλήθεις μηχανισμούς.

3. Μια σημαντική παρατήρηση είναι ότι στο προηγούμενο παράδειγμα υποθέσαμε ότι ο μηχανισμός γνωρίζει τους χρόνους εκτέλεσης των εργασιών. Κάτι τέτοιο όμως δεν είναι δεδομένο εκ' των προτέρων. Αν ήταν τότε δεν θα χρειαζόταν κάποιος μηχανισμός για να βρούμε την βέλτιστη ανάθεση. Απλά θα χρησιμοποιούσαμε κάποιο βέλτιστο αλγόριθμο για το task scheduling. Θεωρούμε ότι κάθε παίχτης και μόνο αυτός γνωρίζει τους χρόνους στους οποίους μπορεί να εκτελέσει τις εργασίες. Ο βασικός λόγος σχεδίασης του μηχανισμού είναι ότι ο σχεδιαστής δεν γνωρίζει τους χρόνους αυτούς. Οι παίχτες θεωρούνται ιδιοτελείς και είναι πιθανό να δηλώσουν ψευδείς χρόνους για να αναλάβουν λιγότερες εργασίες. Ο σχεδιαστής πρέπει να επιλέξει εκείνο το μηχανισμό που θα «αναγκάζει» τους παίχτες να του αποκαλύπτουν τους πραγματικούς χρόνους εκτέλεσης. Θέλουμε συνεπώς ένα φιλαλήθη μηχανισμό για το πρόβλημα. Από τη στιγμή που ο μηχανισμός πάρει τους χρόνους εκτέλεσης, εφαρμόζει κάποιο βέλτιστο αλγόριθμο που θα του δώσει την βέλτιστη ανάθεση.

4. Μία άλλη παρατήρηση είναι ότι το task scheduling είναι πρόβλημα NP- hard και με τα μέχρι τώρα δεδομένα οι βέλτιστοι αλγόριθμοι γι' αυτό, για μεγάλο αριθμό παιχτών, είναι εκθετικού χρόνου. Άρα κάθε μηχανισμός φιλαλήθης ή όχι που χρησιμοποιεί ως αλγόριθμο εξόδου κάποιο βέλτιστο αλγόριθμο εξόδου θα χρειάζεται εκθετικό χρόνο για τον υπολογισμό της βέλτιστης ανάθεσης.

Για τη συνέχεια θα επικεντρώσουμε την μελέτη μας στους φιλαλήθεις μηχανισμούς.

Σημείωση: Με $m = (x, p)$ θα συμβολίζουμε κάποιο μηχανισμό άμεσης αποκάλυψης για το task scheduling πρόβλημα, όπου $x = x(t)$ είναι ο αλγόριθμος

της ανάθεσης των εργασιών και $p = p(t)$ οι πληρωμές που ορίζει ο μηχανισμός στους παίχτες.

2.1.2 Ο Μηχανισμός Min Work

Ένας απλός τρόπος για να έχουμε προσεγγιστική λύση στο task scheduling είναι η ελαχιστοποίηση του συνολικού χρόνου εργασίας. Δηλαδή δεν αναζητούμε πλέον την ανάθεση που δίνει το μικρότερο make-span, αλλά εκείνη που ελαχιστοποιεί τον συνολικό χρόνο εργασίας όλων των παιχτών.

Με βάση την μέθοδο αυτή θα σχεδιάσουμε ένα προσεγγιστικό μηχανισμό για το πρόβλημα. Δηλαδή ο μηχανισμός που θα σχεδιάσουμε δεν θα μας δίνει πλέον την βέλτιστη ανάθεση για κάθε στιγμιότυπο του προβλήματος. Για κάθε στιγμιότυπο όμως, η ανάθεση που θα μας δίνει θα είναι το πολύ κατά κάποιο σταθερό παράγοντα c χειρότερη από την βέλτιστη.

Ορισμός 2.1.2 (Ο Μηχανισμός Min Work)

- **Ανάθεση:** κάθε εργασία ανατίθεται στον παίχτη που είναι ικανός να την εκτελέσει στον μικρότερο χρόνο.
- **Πληρωμές:** η πληρωμή κάθε παίχτη ορίζεται ως :
  $p^i(t) = \sum_{j \in x^i(t)} \min_{i' \neq i} \left(t_j^{i'} \right)$
  Δηλαδή ο παίχτης i για κάθε εργασία που του ανατίθεται θα πληρωθεί ποσό ίσο με τον χρόνο του δεύτερου καλύτερου παίχτη για αυτήν την εργασία.

Στη συνέχεια θα αποδείξουμε ότι ο μηχανισμός Min Work είναι n-προσεγγιστικός για το task scheduling πρόβλημα, όπου n είναι ο αριθμός των παιχτών. Δηλαδή θα δείξουμε ότι για κάθε στιγμιότυπο του προβλήματος, ο Min Work δίνει αναθέσεις των οποίων το make-span είναι στη χειρότερη περίπτωση n φορές μεγαλύτερο από το βέλτιστο.

Θεώρημα 2.1.3 Ο μηχανισμός Min Work είναι ισχυρά φιλαλήθης και n-προσεγγιστικός για το task scheduling πρόβλημα.

Απόδειξη: Η απόδειξη βασίζεται στους επόμενους δύο ισχυρισμούς.

Ισχυρισμός 1 Ο μηχανισμός Min Work είναι ισχυρά φιλαλήθης.

Απόδειξη: Πρώτα θα δείξουμε ότι ο Min Work ανήκει στους VCG μηχανισμούς. Έτσι ο Min Work θα είναι φιλαλήθης, αφού κάθε VCG μηχανισμός είναι φιλαλήθης.

Η έξοδος είναι η ανάθεση που μεγιστοποιεί την συνάρτηση $\sum_{i=1}^{n} v^i\left(t^i, x\right)$, καθώς οι αποτιμήσεις των παιχτών είναι η αρνητική τιμή του συνολικού χρόνου που εργάζονται και η ανάθεση ελαχιστοποιεί το συνολικό χρόνο εργασίας.

Έστω ότι h^i είναι το $\sum_{j=1}^{k} \min_{i' \neq i} t_j^{i'}$, τότε το $\sum_{i' \neq i} v^{i'}\left(t^{i'}, x\right) + h^{-i}$ είναι

ακριβώς η συνάρτηση πληρωμών του μηχανισμού και έχει την μορφή των VCG μηχανισμών. Άρα ο ο Min Work ανήκει στους VCG μηχανισμούς.

Θα δείξουμε τώρα ότι η φιλαλήθεια είναι η μοναδική κυρίαρχη στρατηγική. Θα το δείξουμε για την περίπτωση που έχουμε μία εργασία, η περίπτωση $k > 1$ αποδεικνύεται ομοίως. Έστω d οι δηλώσεις των παιχτών και t οι αληθείς τύποι τους. Θεωρούμε την περίπτωση όπου $d^i \neq t^i$ $(i = 1,2)$. Εάν $d^i > t^i$ τότε για d^{3-i} ώστε $d^i > d^{3-i} > t^i$, η utility του παίχτη i είναι $t^i - d^i < 0$, αντί για 0 που είναι στην περίπτωση της φιλαλήθειας. Όμοίως εργαζόμαστε για την περίπτωση που $d^i < t^i$. ■

Ισχυρισμός 2 Ο Min Work είναι n-προσεγγιστικός για το task scheduling πρόβλημα.

Απόδειξη: Έστω opt(t) η βέλτιστη ανάθεση. Το ζητούμενο προκύπτει άμεσα από τις σχέσεις $g(x(t),t) \le \sum_{j=1}^{k} \min_i t_j^i$ και $g(opt(t),t) \ge \frac{1}{n} \cdot \sum_{j=1}^{k} \min_i t_j^i$. ■

Με βάση τα παραπάνω η απόδειξη του θεωρήματος 2.1.3 έχει ολοκληρωθεί. ■

Παράδειγμα: Στο παράδειγμα του πίνακα 2.1.1, είχαμε 2 παίχτες στους οποίους έπρεπε να αναθέσουμε 3 εργασίες. Η βέλτιστη ανάθεση ήταν η A_1: $\{j_1, j_2\}$, A_2: $\{j_3\}$ με make-span 60. Αν στο παράδειγμα αυτό εφαρμόσουμε τον Min Work, τότε θα πάρουμε την ανάθεση A_1: $\{j_1, j_3\}$, A_2: $\{j_2\}$ με make-span 70. Το θεώρημα 2.1.3 εγγυάται ότι το make-span της ανάθεσης του Min Work θα είναι το πολύ διπλάσιο του βέλτιστου make-span

2.1.3 Κάτω Φράγματα για το Task Scheduling

Στην ενότητα 1.1.3 αναφέραμε ότι εάν υπάρχει μηχανισμός που υλοποιεί ένα πρόβλημα με κυρίαρχες στρατηγικές τότε υπάρχει και φιλαλήθης υλοποίηση για το πρόβλημα (πρόταση 1.1.6). Συνεπώς αρκεί να αποδείξουμε ένα κάτω φράγμα γενικά για τις φιλαλήθεις υλοποιήσεις του task scheduling.

Για τη συνέχεια θα θεωρούμε ότι ο $m = (x, p)$ είναι φιλαλήθης μηχανισμός για το task scheduling.

Πρόταση 2.1.4 (Ανεξαρτησία) Έστω t_1 και t_2 δύο διανύσματα τύπων και έστω ένας παίχτης i .Εάν $t_1^{-i} = t_2^{-i}$ και $x^i(t_1) = x^i(t_2)$ τότε $p^i(t_1) = p^i(t_2)$.

Απόδειξη: Υποθέτουμε ότι $p^i(t_1) < p^i(t_2)$, τότε εάν ο τύπος του παίχτη i είναι t_1^i είναι καλύτερο γι' αυτόν να δηλώσει ψευδώς t_2^i. Καταλήγουμε έτσι σε αντίφαση διότι ο μηχανισμός είναι φιλαλήθης. ■

Παρατήρηση: Από την πρόταση αυτή συμπεραίνουμε ότι όταν οι τύποι των

υπολοίπων παιχτών και η ανάθεση έχουν καθοριστεί, τότε η πληρωμή που προσφέρεται σε ένα παίχτη δεν εξαρτάται από τη δήλωση για τον τύπο του.

Η πληρωμή κάθε παίχτη μπορεί να αναπαρασταθεί από την ακόλουθη συνάρτηση.

Ορισμός 2.1.5 Έστω t διάνυσμα τύπων και i παίχτης. Για ένα σύνολο εργασιών X ορίζουμε την πληρωμή που προσφέρεται στον i λόγω του X:

$$p^i\left(X, t^{-i}\right) = \begin{cases} p^i\left(t'^i, t^{-i}\right) & \text{εάν υπάρχει } t'^i \text{ έτσι ώστε } x^i\left(t'^i, t^{-i}\right) = X \\ 0 & \text{αλλιώς} \end{cases}$$

Ερμηνεία: Η παραπάνω συνάρτηση έχει την εξής ερμηνεία. Έστω ένα διάνυσμα τύπων t ένας παίχτης i και κάποιο σύνολο εργασιών X. Αν υπάρχει κάποιος τύπος t'^i για τον i έτσι ώστε η ανάθεση που προκύπτει για τον i να είναι X τότε η πληρωμή του θα είναι $p^i\left(t'^i, t^{-i}\right)$, αν δεν υπάρχει τέτοιος τύπος τότε η πληρωμή θα είναι 0. Η συνάρτηση πληρωμής του i αναφέρεται σε δεδομένο διάνυσμα τύπων των υπολοίπων παιχτών και σε συγκεκριμένο σύνολο εργασιών X. Είναι ένας γενικός τρόπος για να περιγράψουμε τις πληρωμές των παιχτών. Μας δίνει την πληρωμή που αντιστοιχεί στον i αν τελικά υπάρχει τύπος γι' αυτόν ώστε να μπορεί να πάρει το X. Συνήθως η είναι πιο πρακτικό να περιγράφουμε ένα μηχανισμό με τις τιμές των πληρωμών που δίνει παρά με την συνάρτηση πληρωμής.

Σημείωση: Έστω παίχτης i, t^i ο τύπος του και X σύνολο εργασιών. Με $t^i(X) = \sum_{j \in X} t^i_j$ ορίζουμε τον χρόνο που χρειάζεται ο i για να εκτελέσει όλες τις εργασίες του X.

Αποδεικνύεται η ακόλουθη πρόταση:

Πρόταση 2.1.6 (Μεγιστοποίηση) Για κάθε διάνυσμα τύπων και κάθε παίχτη i :

$$x^i(t) \in \arg\max_{X \subseteq \{1,\ldots,k\}} \left(p^i\left(X, t^{-i}\right) - t^i(X)\right)$$

Ερμηνεία: Η ερμηνεία της πρότασης αυτής είναι ότι η ανάθεση του μηχανισμού $x^i(t)$, θα πρέπει να μεγιστοποιεί το όφελος του παίχτη i δηλαδή την utility του. Αν αυτό δεν συμβαίνει τότε ο παίχτης μπορεί να το επιδιώξει ψευδόμενος. Όπως έχουμε αναφέρει η utility του παίχτη i είναι $p^i\left(x^i, t^{-i}\right) - t^i\left(x^i\right)$, (όπου $t^i\left(x^i\right) = t^i(X) = \sum_{j \in X} t^i_j$).

Στη συνέχεια αποδεικνύουμε το βασικό θεώρημα αυτής της υποενότητας. Θα δείξουμε ότι δεν υπάρχει προσεγγιστικός μηχανισμός για το task scheduling πρόβλημα που να επιτυγχάνει προσέγγιση καλύτερη από 2.

Θεώρημα 2.1.7 Δεν υπάρχει μηχανισμός για το task scheduling που να είναι c-προσεγγιστική υλοποίηση, για κάθε $c < 2$.

Απόδειξη: Αρχικά αποδεικνύεται το ακόλουθο λήμμα.

Σημείωση: Έστω παίχτης i, διάνυσμα τύπων t και A και B δύο ξένα σύνολα εργασιών. Ορίζουμε την ***διαφορά τιμής*** $\Delta^i(A,B)$ ως $p^i\left(A \cup B, t^{-i}\right) - p^i\left(A, t^{-i}\right)$. Η διαφορά τιμής απλά μας δίνει την διαφορά στα ποσά που θα πληρωθεί ο παίχτης i όταν του ανατεθούν τα σύνολα εργασιών $A \cup B$ και A.

Λήμμα 2.1.8 Έστω διάνυσμα τύπων t και έστω $X = x^i(t)$. Για κάθε σύνολο εργασιών $D \neq X$ ισχύουν οι ακόλουθες ανισότητες:

1. Εάν $D \subset X$ τότε $\Delta^i(D, X-D) \geq t^i(X-D)$
2. Εάν $D \supset X$ τότε $\Delta^i(X, D-X) \leq t^i(D-X)$
3. Αλλιώς, έστω $L = D \bigcap X$ τότε
   $\Delta^I(L, X-L) - t^i(X-L) \geq \Delta^i(L, D-L) - t^i(D-L)$

Επιπλέον εάν το σύνολο Y των εργασιών ικανοποιεί τις ανισότητες για όλα τα D τότε $Y = X = x^i(t)$.

Απόδειξη: Το ότι οι παραπάνω ανισότητες ισχύουν για το $x^i(t)$ προκύπτει άμεσα από την πρόταση 2.1.6 και τον ορισμό της utility ως $u^i = p^i\left(x^i(t), t^{-i}\right) - t^i\left(x^i(t)\right)$. Όταν οι ανισότητες είναι αυστηρές τότε το X είναι το μοναδικό σύνολο εργασιών που μεγιστοποιεί την utility του παίχτη i. ■

Θα αποδείξουμε το θεώρημα για την περίπτωση των δύο παιχτών, $n = 2$. Την περίπτωση $n > 2$ μπορούμε να την ανάγουμε σε αυτή των δύο παιχτών αν θεωρήσουμε ότι οι υπόλοιποι παίχτες είναι πολύ πιο αργοί από τους 1 και 2.

Σημείωση: Έστω t διάνυσμα τύπων , παίχτης i και σύνολο εργασιών X. Αν $\alpha > 0$ είναι ένας πραγματικός αριθμός, τότε με $\hat{t} = t\left(X \xrightarrow{i} \alpha\right)$ θα καλούμε τον τύπο για τον οποίο : $\hat{t}_j^{i'} = \begin{cases} \alpha & \text{εάν } i' = i \text{ και } j \in X \\ t_j^{i'} & \text{αλλιώς} \end{cases}$

Δηλαδή στο διάνυσμα τύπων t αντικαθιστούμε με α τους χρόνους του παίχτη i για τις εργασίες του συνόλου X.
Ομοίως ορίζουμε με $\hat{t} = t\left(X^1 \xrightarrow{i_1} \alpha_1, X^2 \xrightarrow{i_2} \alpha_2, \ldots\right)$ το αποτέλεσμα της ακολουθίας των παραπάνω μετασχηματισμών.

Έστω $k \geq 3$ και t διάνυσμα τύπων με $t_j^i = 1$ για κάθε παίχτη i και κάθε εργασία j. Υποθέτουμε ότι $\left|x^1(t)\right| \leq \left|x^2(t)\right|$, έστω $x = x^1(t)$ και $\bar{x}$ το συμπλήρωμα του.

Ισχυρισμός Έστω $0 < \varepsilon < 1$, $\hat{t} = t\left(x \xrightarrow{1} \varepsilon, \overline{x} \xrightarrow{1} 1+\varepsilon\right)$. Τότε $x(\hat{t}) = x(t)$.

Απόδειξη: Επειδή $n = 2$ αρκεί να δείξουμε ότι $x^1(\hat{t}) = x^1(t)$. Καθώς ο τύπος του παίχτη 2 δεν έχει αλλάξει, η πληρωμή του παίχτη 1 παραμένει η ίδια . Για τον τύπο t, το $x^1(t)$ ικανοποιεί τις ανισότητες του λήμματος 2.1.8. Όταν ο τύπος γίνει $\hat{t}$, οι ανισότητες είναι αυστηρές και κατά συνέπεια η ανάθεση δεν αλλάζει. ■

Αν το $|x^2(t)|$ είναι άρτιος, τότε το κάτω φράγμα προκύπτει αμέσως καθώς

$$g(x(\hat{t}), \hat{t}) = |x^2(\hat{t})| = |x^2(t)| \text{ και } g(opt(\hat{t}), \hat{t}) \le \frac{1}{2} \cdot |x^2| + k \cdot \varepsilon$$

Αν το $|x^2(t)|$ είναι περιττός τότε πρέπει $|x^2(t)| \ge 3$. Επιλέγουμε αυθαίρετα μια εργασία $j \in x^2(t)$ και θεωρούμε τον τύπο $\hat{t}\left(\{j\} \xrightarrow{2} \varepsilon\right)$ που μας δίνει την ίδια ανάθεση. ■

Το κάτω φράγμα είναι σφιχτό (tight) για την περίπτωση των δύο παιχτών. Αυτό προκύπτει από το προηγούμενο θεώρημα και από το άνω φράγμα που αποδείξαμε για τον μηχανισμό Min Work. Οι Noam Nisan και Amir Ronen στο [NR99] κάνουν την εικασία ότι και για την γενική περίπτωση το πάνω φράγμα είναι σφιχτό (tight).

Εικασία 2.1.9 [NR99] Δεν υπάρχει c-προσεγγιστικός μηχανισμός για το task scheduling με n παίχτες για οποιοδήποτε $c < n$.

Αν η εικασία αυτή είναι σωστή τότε ο μηχανισμός Min Work θα επιτυγχάνει την καλύτερη δυνατή προσέγγιση για το task scheduling καθώς είναι n-προσεγγιστικός. Στην συνέχεια θα δείξουμε την εικασία για δύο ειδικές περιπτώσεις.

Ειδικές Περιπτώσεις

Ορισμός 2.1.10 Ένας μηχανισμός θα καλείται *προσθετικός (additive)*, αν για κάθε παίχτη i, διάνυσμα τύπων t και σύνολο εργασιών X ισχύει:

$$p^i(X, t^{-i}) = \sum_{j \in X} p^i(\{j\}, t^{-i})$$

Δηλαδή η πληρωμή του παίχτη i προκύπτει από το άθροισμα των πληρωμών για κάθε εργασία j που ανέλαβε από το σύνολο X.

Μπορεί να αποδειχθεί το ακόλουθο θεώρημα.

Θεώρημα 2.1.11 Δεν υπάρχει προσθετικός μηχανισμός που να λύνει το c-προσεγγιστικό πρόβλημα για οποιοδήποτε $c < n$.

Συνεπώς η εικασία 2.1.9 ισχύει για τους προσθετικούς μηχανισμούς.

Ορισμός 2.1.12 Ένας μηχανισμός θα καλείται *τοπικός (local)*, εάν για κάθε παίχτη i, διάνυσμα τύπων t και σύνολο εργασιών X, η πληρωμή $p^i(X, t^{-i})$ εξαρτάται μόνο από τις τιμές των παιχτών για τις εργασίες στο X. Δηλαδή από το $\{t_j^{l\neq i} \mid j \in X\}$.

Μπορεί να αποδειχθεί το ακόλουθο θεώρημα.

Θεώρημα 2.1.13 Δεν υπάρχει τοπικός μηχανισμός που να λύνει το c-προσεγγιστικό πρόβλημα για οποιοδήποτε $c < n$.

Κατά συνέπεια η εικασία 2.1.9 ισχύει και για τους τοπικούς μηχανισμούς

2.1.4 Πιθανοτικοί Μηχανισμοί

Στην ενότητα 2.1.3 δείξαμε ότι δεν υπάρχει μηχανισμός για το task scheduling πρόβλημα που να είναι καλύτερος από 2-προσεγγιστικός. Στην συνέχεια θα δείξουμε ότι με την σχεδίαση πιθανοτικών μηχανισμών μπορούμε να επιτύχουμε καλύτερο κάτω φράγμα. Το πιθανοτικό μοντέλο διατηρεί την απαίτηση για κυρίαρχες στρατηγικές. Θεωρούμε ότι οι παίχτες επιλέγουν τις στρατηγικές τους χωρίς να γνωρίζουν τα αποτελέσματα από τις πιθανοτικές ρίψεις του μηχανισμού. Ο μηχανισμός θα πρέπει να είναι έτσι σχεδιασμένος ώστε η στρατηγική των παιχτών να είναι κυρίαρχη για όλες τις πιθανές ρίψεις.

Ορισμός 2.1.14 (Πιθανοτικός Μηχανισμός) Ένας *πιθανοτικός μηχανισμός* (randomized mechanism) είναι μία πιθανοτική κατανομή πάνω στην οικογένεια μηχανισμών $\{m_r \mid r \in I\}$, όπου όλοι έχουν το ίδιο σύνολο στρατηγικών και το ίδιο σύνολο πιθανών εξόδων.

Η έξοδος ενός τέτοιου μηχανισμού είναι μία πιθανοτική κατανομή πάνω στις εξόδους και τις πληρωμές των μηχανισμών της οικογένειας. Η περιγραφή του προβλήματος πρέπει να καθορίζει ποια κατανομή εξόδου απαιτείται. Για την περίπτωση προβλημάτων βελτιστοποίησης, η αντικειμενική συνάρτηση είναι $g(\alpha, t) = E_{r\in I}(g(o_{m_r}(\alpha), t))$, δηλαδή είναι η αναμενόμενη μέση τιμή για όλες τις αντικειμενικές συναρτήσεις των μηχανισμών της οικογένειας.

Θα ορίσουμε τώρα την έννοια της κυρίαρχης στρατηγικής για τους πιθανοτικούς μηχανισμούς.

Ορισμός 2.1.15 (Καθολικά Κυρίαρχη Στρατηγική)

- Η στρατηγική α^i για τον παίχτη i θα καλείται ***καθολικά κυρίαρχη στρατηγική*** *(universally dominant strategy)* ή απλά ***κυρίαρχη***, εάν είναι κυρίαρχη στρατηγική για κάθε μηχανισμό που υποστηρίζει τον πιθανοτικό

μηχανισμό.

- Ένας πιθανοτικός μηχανισμός θα καλείται ***καθολικά φιλαλήθης*** *(universally truthful)* ή απλά ***φιλαλήθης*** εάν η φιλαλήθεια είναι κυρίαρχη στρατηγική.
- Θα καλείται ***ισχυρά φιλαλήθης*** *(strongly truthful)* εάν η φιλαλήθεια είναι η μοναδική κυρίαρχη στρατηγική.

Θα παρουσιάσουμε ένα ισχυρά φιλαλήθη πιθανοτικό μηχανισμό για το task scheduling που δίνει καλύτερα αποτελέσματα από το ντετερμινιστικό κάτω φράγμα. Ο πιθανοτικός μηχανισμός θα είναι μία κατανομή των ντετερμινιστικών *biased (προκατειλημμένων) min work* μηχανισμών που ορίζονται στην εικόνα 2.1.1. Τα χαρακτηριστικά της κατανομής αυτής θα τα δούμε στη συνέχεια. Στην εικόνα 2.1.1 δίνουμε τον biased min work μηχανισμό για 2 παίχτες.

Εικόνα 2.1.1: Ο Biased Min Work Μηχανισμός για δύο παίχτες

Παράμετροι: Ένας πραγματικός αριθμός $\beta \geq 1$ και ένα διάνυσμα $s \in \{1,2\}^k$.

Είσοδος: Το διάνυσμα των δηλωμένων τύπων $t = (t^1, t^2)$.

Έξοδος: Μία ανάθεση $x = (x^1, x^2)$ και μία πληρωμή $p = (p^1, p^2)$.

Μηχανισμός:

$x^1 \leftarrow \varnothing ; x^2 \leftarrow \varnothing ; p^1 \leftarrow 0 ; p^2 \leftarrow 0.$

Για κάθε εργασία $j = 1 \ldots k$

Έστω $i = s_j$ και $i' = 3 - i$

Εάν $t_j^i \leq \beta \cdot t_j^{i'}$

Τότε $x^i \leftarrow x^i \cup \{j\} ; p^i \leftarrow p^i + \beta \cdot t_j^{i'}$

Αλλιώς $x^{i'} \leftarrow x^{i'} \cup \{j\} ; p^{i'} \leftarrow p^{i'} + \beta^{-1} \cdot t_j^i$

Παρατηρούμε ότι ο μηχανισμός αυτός είναι προκατειλημένος (biased) λόγω της μεθόδου με την οποία υπολογίζει τις πληρωμές.
Αν $t_j^i \leq \beta \cdot t_j^{i'}$ τότε $p^i \leftarrow p^i + \beta \cdot t_j^{i'}$, ενώ αν $t_j^i \geq \beta \cdot t_j^{i'}$ τότε $p^{i'} \leftarrow p^{i'} + \beta^{-1} \cdot t_j^i$.

Παράδειγμα: Θα εφαρμόσουμε τον Biased Min Work στο παράδειγμα του πίνακα 2.1.2. Έστω ότι $\beta = 2$ και $s = \{1,1,2\}$.

	j_1	j_2	j_3
A_1	30	70	60
A_2	50	20	80

Πίνακας 2.1.2

- Αρχικά ο αλγόριθμος αρχικοποιεί με $x^1 = \varnothing$, $x^2 = \varnothing$ και $p^1 = 0$, $p^2 = 0$.
- Μετά χρησιμοποιεί το διάνυσμα s.
- Ξεκινά με την εργασία j_1, $i = 1 = s_1$ και $i' = 3 - 1 = 2$. Μετά συγκρίνει τις τιμές $t_1^1 = 30$ και $\beta \cdot t_1^2 = 2 \cdot 50 = 100$. Επειδή $t_1^1 \le \beta \cdot t_1^2$ κάνει την ανάθεση $x^1 = \{ j_1 \}$ και ορίζει $p^1 = 2 \cdot t_1^2 = 2 \cdot 50 = 100$.
- Συνεχίζει με την εργασία j_2. Έχουμε $i = 1 = s_2$ και $i' = 3 - 1 = 2$. Όμοια με πριν συγκρίνει τις τιμές $t_2^1 = 70$ και $\beta \cdot t_2^2 = 2 \cdot 20 = 40$. Επειδή $t_2^1 \ge \beta \cdot t_2^2$ κάνει την ανάθεση $x^2 = \{ j_2 \}$ και ορίζει $p^2 = 2^{-1} \cdot t_2^1 = 2^{-1} \cdot 70 = 35$.
- Τέλος για την εργασία j_3, $2 = s_3$ και $i' = 3 - 2 = 1$. Επειδή $t_3^2 = 80 \le 2 \cdot t_3^1 = 120$ κάνει την ανάθεση $x^2 = \{ j_2, j_3 \}$ και ορίζει $p^2 = 35 + 2 \cdot t_3^1 = 35 + 2 \cdot 60 = 155$.
- Η τελική ανάθεση είναι $x^1 = \{ j_1 \}$ και $x^2 = \{ j_2, j_3 \}$ και οι πληρωμές $p^1 = 100$ και $p^2 = 155$.

Λήμμα 2.1.16 Για όλες τις τιμές των παραμέτρων β και s, ο biased min work μηχανισμός είναι ισχυρά φιλαλήθης.

Απόδειξη: Καθώς η συνολική utility κάθε παίχτη μπορεί να καθοριστεί ως το άθροισμα των utility από κάθε βήμα αρκεί να θεωρήσουμε την περίπτωση $k = 1$. Σε αυτή ο μηχανισμός είναι ισοδύναμος με τον weighted VCG μηχανισμό (Κεφ.1) με βάρη $\{1, \beta\}$ ή $\{\beta, 1\}$ ανάλογα με το s_j. ■

Ορισμός 2.1.17 (ΟΠιθανοτικός Biased Min Work Μηχανισμός)
Ο *πιθανοτικός biased min work μηχανισμός* είναι η κατανομή των *biased min work μηχανισμών* για $\beta = 4/3$ και για ομοιόμορφη κατανομή των $s \in \{1,2\}^k$.

Παρατήρηση: Ο μηχανισμός biased min work είναι ντετερμινιστικός και κάθε φορά οι τιμές των β και s θα είναι καθορισμένες. Για διαφορετικά διανύσματα s (ή και για διαφορετικά β) έχουμε και διαφορετικούς biased min work μηχανισμούς. Αν οι τιμές του s αλλάζουν σύμφωνα με την ομοιόμορφη κατανομή και $\beta = 4/3$, τότε έχουμε τον πιθανοτικό biased min work μηχανισμό που αποτελεί ομοιόμορφη κατανομή των biased min work μηχανισμών που έχουν τιμές του s στο χώρο $\{1,2\}^k$. Αυτή είναι και η οικογένεια των μηχανισμών που υποστηρίζει τον πιθανοτικό biased min work μηχανισμό.

Αποδεικνύεται ότι ο πιθανοτικός diased min work μηχανισμός είναι πολυωνιμικού χρόνου, δηλαδή οι αναθέσεις και οι πληρωμές υπολογίζονται σε πολυωνιμικό χρόνο. Επίσης είναι 7/4 προσεγγιστικός για το task-scheduling. Ο μηχανισμός αυτός είναι ισχυρά φιλαλήθης κάτι που προκύπτει άμεσα από το λήμμα 2.1.16.

Θεώρημα 2.1.18 Ο πιθανοτικός diased min work μηχανισμός είναι 7/4-

προσεγγιστική, ισχυρά φιλαλήθης και πολυωνιμικού χρόνου υλοποίηση του task scheduling προβλήματος με δύο παίχτες.

2.1.5 Μηχανισμοί με Επαλήθευση

Στο μοντέλο του σχεδίου μηχανισμών που είδαμε μέχρι τώρα, οι παίχτες μπορούν να επιλέγουν οποιαδήποτε στρατηγική επιθυμούν ανεξάρτητα από τον τύπο τους. Οι παίχτες κάνουν μία δήλωση στο μηχανισμό η οποία αναφέρεται στις τιμές του τύπου τους και με βάση τις δηλώσεις των παιχτών ο μηχανισμός δίνει μία έξοδο που είναι η λύση του προβλήματος. Οι παιχτών μπορεί να δώσουν ψευδή στοιχεία στο μηχανισμό κάτι που ο μηχανισμός δεν μπορεί να ελέγξει. Για να αντιμετωπίσει το πρόβλημα αυτό ο σχεδιαστής προσπαθεί να σχεδιάσει ένα φιλαλήθη μηχανισμό. Αν ο μηχανισμός είναι φιλαλήθης τότε οι παίχτες θα έχουν το μεγαλύτερο όφελος μόνο αν είναι ειλικρινείς.

Παρατηρούμε ότι στο μοντέλο αυτό δεν υπάρχει κανένα στάδιο επαλήθευσης των δηλώσεων των παιχτών. Ο μηχανισμός συλλέγει τις απαιτούμενες πληροφορίες από τους παίχτες, καθορίζει την έξοδο και τις πληρωμές αλλά μετά δεν κάνει κανένα έλεγχο για το αν οι παίχτες τήρησαν τα συμφωνηθέντα. Ακόμα και σε ένα φιλαλήθη μηχανισμό μπορεί τελικά οι παίχτες όταν εκτελέσουν την προκύπτουσα έξοδο να μην ακολουθήσουν την (αληθή) δήλωση τους.

Στη συνέχεια θα παρουσιάσουμε μία «φυσική» τροποποίηση του προηγούμενου μοντέλου και θα ορίσουμε τους μηχανισμούς με επαλήθευση. Στους μηχανισμούς αυτούς διακρίνουμε δύο στάδια: *το στάδιο της δήλωσης* όπου οι παίχτες «συζητούν» με το μηχανισμό και το αποτέλεσμά του είναι μία απόφαση (π.χ. ανάθεση) και *το στάδιο της εκτέλεσης*, όπου οι παίχτες εκτελούν τη συμφωνηθείσα έξοδο. Οι πληρωμές δίνονται από το μηχανισμό μετά το στάδιο της εκτέλεσης. Διαισθητικά μπορούμε να φανταστούμε τη φάση της εκτέλεσης ως το στάδιο στο οποίο ο μηχανισμός επαληθεύει τις δηλώσεις των παιχτών και τους τιμωρεί εάν ψεύδονται.

Ορισμός 2.1.19 (Μηχανισμός με Επαλήθευση)

1. Η στρατηγική α^i του παίχτη i αποτελείται από δύο μέρη, την δήλωση d^i και την εκτέλεση e^i.
2. Η δήλωση d^i, επιλέγεται από τον παίχτη i και βασίζεται στον τύπο του t^i κατά αυθαίρετο τρόπο.
3. Η απόφαση k του μηχανισμού πρέπει να είναι μία συνάρτηση των δηλώσεων $d^1,\ldots,d^n$.
4. Η εκτέλεση e^i ενός παίχτη μπορεί να εξαρτάται από το t^i αλλά και από το k. Η περιγραφή του προβλήματος καθορίζει για κάθε t^i τα πιθανά $e^i(\ )$ που μπορεί να επιλέξει ο παίχτης με τύπο t^i.
5. Η έξοδος του μηχανισμού είναι το αποτέλεσμα της απόφασης k και των εκτελέσεων $e^1(k),\ldots,e^n(k)$ των παιχτών. Η συνάρτηση εξόδου $o(k,e)$ είναι μέρος της περιγραφής του προβλήματος.

6. Η έξοδος ο, καθορίζει την αντικειμενική συνάρτηση $g(o,t)$ και τις αποτιμήσεις (valuations) $v^i(t^i,o)$ των παιχτών.
7. Η πληρωμή p^i που ορίζει ο μηχανισμός εξαρτάται από τις δηλώσεις $d^1,\ldots,d^n$ και από τις εκτελέσεις $e^1(k),\ldots,e^n(k)$.

Σχόλια:
1. Η στρατηγική των παιχτών αποτελείται από τη δήλωση και την εκτέλεση. Δηλαδή οι παίχτες αποφασίζουν εκτός από τον τύπο που θα ανακοινώσουν στο μηχανισμό και τον τύπο που θα ακολουθήσουν για στο στάδιο της εκτέλεσης. Για παράδειγμα στο πρόβλημα ανάθεσης εργασιών ο παίχτης αποφασίζει τι χρόνο εκτέλεσης θα δηλώσει για την εργασία j αλλά επίσης και σε τι χρόνο τελικά θα την εκτελέσει.
2. Η απόφαση k αντιστοιχεί στην έξοδο του μηχανισμού στο κλασικό μοντέλο. Για παράδειγμα για να αποφασίσει μία ανάθεση εργασιών ο μηχανισμός αναγκαστικά θα στηριχθεί στις πληροφορίες που θα του δώσουν οι παίχτες.
3. Στους μηχανισμούς με επαλήθευση η έξοδος είναι συνάρτηση της απόφασης και της εκτέλεσης. Δηλαδή ο μηχανισμός δίνει ως έξοδο το τι πρέπει να κάνουν οι παίχτες καθώς και το τι πραγματικά έκαναν.
4. Είναι σημαντικό να παρατηρήσουμε ότι οι παίχτες θα πληρωθούν με βάση την εκτέλεση της απόφασης και όχι με βάση τη δήλωση τους. Έτσι το μοντέλο αυτό είναι πιο ρεαλιστικό διότι οι παίχτες δεν μπορούν να εξαπατήσουν τον μηχανισμό καθώς θα πληρωθούν με βάση την πραγματική τους εργασία.

Ορισμός 2.1.20 (Φιλαλήθης μηχανισμός με επαλήθευση)
- Ένας μηχανισμός με επαλήθευση θα καλείται ***φιλαλήθης*** *(truthful)* εάν:
  1. Οι δηλώσεις των παιχτών είναι οι τύποι τους.
  2. Για κάθε παίχτη i με τύπο t^i, υπάρχει κυρίαρχη στρατηγική της μορφής $\alpha^i = (t^i, e^i(\ ))$.
- Ο μηχανισμός θα καλείται ***ισχυρά φιλαλήθης*** *(strongly truthful)* αν η παραπάνω στρατηγική είναι η μόνη κυρίαρχη στρατηγική

Σχόλια: Παρατηρούμε ότι στη κυρίαρχη στρατηγική των παιχτών υπεισέρχεται και η εκτέλεση τους. Ένας μηχανισμός θα είναι φιλαλήθης αν οι παίχτες έχουν το μεγαλύτερο όφελος όταν είναι ειλικρινείς στο μηχανισμό και επίσης υπάρχει κάποια εκτέλεση που την θεωρούν την καλύτερη γι' αυτούς. Ο μηχανισμός θα «εξαναγκάζει» τους παίχτες να μείνουν πιστοί στη δήλωση τους μέσω της συνάρτησης πληρωμής που θα εξαρτάται και από τις εκτελέσεις. Προσθέτοντας την αρχή της αποκάλυψης (πρόταση 1.1.6) στο στάδιο της δήλωσης μπορούμε να περιορίσουμε την μελέτη μας στους μηχανισμούς όπου οι παίχτες απλώς καλούνται να αποκαλύψουν τους τύπους τους.

Σημείωση: Θα συμβολίζουμε έναν μηχανισμό με επαλήθευση με το ζεύγος $m = (k(d), p(d,e))$, όπου $k(\ )$ είναι η απόφαση και $p(\ )$ η συνάρτηση πληρωμής.

Ορισμός 2.1.21 (Task Scheduling Με Επαλήθευση) Ο ορισμός του προβλήματος είναι ο ίδιος με το πρόβλημα ανάθεσης εργασιών (2.1.1) εκτός από το ότι ο μηχανισμός μπορεί να πληρώσει τους παίχτες μετά την εκτέλεση των εργασιών. Υποθέτουμε συνεπώς ότι οι χρόνοι στους οποίους εκτελούνται οι εργασίες είναι γνωστοί στο μηχανισμό.
Συγκεκριμένα:

1. Η εφικτή (feasible) έξοδος του μηχανισμού ορίζεται από το ζεύγος $(x, \tilde{t})$, όπου $x = x^1, \ldots, x^n$ είναι η ανάθεση των εργασιών στους παίχτες και $\tilde{t} = \tilde{t}_1, \ldots, \tilde{t}_k$ οι πραγματικοί χρόνοι εκτέλεσης των εργασιών. Εάν $j \in x^i(t)$ τότε πρέπει $\tilde{t}_j \geq t_j^i$, όπου t_j^i είναι ο χρόνος που δήλωσε ο παίχτης i ότι μπορεί να εκτελέσει την εργασία j.
2. Η στρατηγική ενός παίχτη αποτελείται από δύο μέρη: την δήλωση του τύπου του και την εκτέλεση της εργασίας που του ανατέθηκε. Ο παίχτης μπορεί να ψεύδεται ή να εκτελεί οποιαδήποτε εργασία του ανατίθεται σε χρόνο $\tilde{t} \geq t_j^i$.
3. Η αντικειμενική συνάρτηση δίνεται από τον τύπο: $g(x, \tilde{t}) = \max_i \sum_{j \in x^i} \tilde{t}_j$ (το make span).
4. Η αποτίμηση (valuation) του παίχτη είναι $v^i(x, \tilde{t}) = -\sum_{j \in x^i} \tilde{t}_j$.
5. Ο μηχανισμός είναι το ζεύγος (x, p) όπου $x(t) = x^1(t), \ldots, x^n(t)$ είναι η συνάρτηση ανάθεσης και $p(t, \tilde{t}) = p^1(t, \tilde{t}), \ldots, p^n(t, \tilde{t})$ είναι η πληρωμή.

Παρατηρήσεις:
1. Στο πρόβλημα ανάθεσης εργασιών που μελετήσαμε στη 2.1.1 οι παίχτες δήλωναν στο μηχανισμό τους χρόνους στους οποίους μπορούσαν να εκτελέσουν τις εργασίες. Με βάση τους χρόνους αυτούς ο μηχανισμός καθόριζε τις εργασίες που θα αναλάμβανε κάθε παίχτης και πόσο θα πληρωνόταν. Δεν γίνονταν κάποια επαλήθευση για το αν τελικά οι παίχτες εκτελούσαν οι παίχτες τις εργασίες στους χρόνους που είχαν δηλώσει. Δεν υπήρχε συνεπώς κάποιο στάδιο κατά το οποίο ο μηχανισμός να μετρά τους χρόνους εκτέλεσης, έτσι δεν μπορούμε να θεωρήσουμε ότι τους γνωρίζει. Γι' αυτό και η αντικειμενική συνάρτηση και οι πληρωμές υπολογίζονταν μόνο με βάση τους χρόνους που δήλωναν οι παίχτες.
2. Σε ένα μηχανισμό που υλοποιεί το task scheduling με επαλήθευση, αρχικά οι παίχτες θα δηλώσουν τους χρόνους t_j^i που μπορούν να εκτελέσουν τις εργασίες. Με βάση αυτούς τους χρόνους ο μηχανισμός θα καθορίσει την ανάθεση των εργασιών χρησιμοποιώντας την συνάρτηση ανάθεσης x. Στη συνέχεια οι παίχτες θα εκτελέσουν τις εργασίες και ο μηχανισμός θα μετρήσει τους χρόνους εκτέλεσης $\tilde{t} = \tilde{t}_1, \ldots, \tilde{t}_k$. Δεν θεωρείται δεδομένο ότι οι παίχτες είναι συνεπείς με την δήλωση τους, δηλαδή ότι εκτελούν τις εργασίες στους χρόνους που δήλωσαν ότι μπορούν. Μπορεί οι παίχτες να εκτελέσουν τις εργασίες σε μεγαλύτερο χρόνο από αυτόν που δήλωσαν, κατά συνέπεια η φιλαλήθεια τους πρέπει να επαληθευθεί. Η αντικειμενική συνάρτηση, δηλαδή το make-span, θα υπολογιστεί

από τους πραγματικούς χρόνους εκτέλεσης των παιχτών και όχι από τους χρόνους που δήλωσαν. Η αποτίμηση $v^i(x,\tilde{t}) = -\sum_{j \in x^i} \tilde{t}_j$ κάθε παίχτη υπολογίζεται και αυτή από τους πραγματικούς χρόνους εκτέλεσης Η συνάρτηση πληρωμής, που ουσιαστικά θα καθορίζει τον μηχανισμό, θα εξαρτάται από τους πραγματικούς και από τους δηλούμενους χρόνους. Ο στόχος του μηχανισμού είναι να βρει εκείνη την ανάθεση που με βάση τους πραγματικούς χρόνους ελαχιστοποιεί το make-span. Γι' αυτό και η έξοδος του μηχανισμού είναι το ζεύγος $(x,\tilde{t})$.

Στη συνέχεια θα αναφερθούμε στον ***Μηχανισμό Αποζημίωσης και Πριμ***, ένα μηχανισμό επαλήθευσης για το task scheduling με επαλήθευση,. Ο μηχανισμός αυτός αποτελείται από ένα βέλτιστο αλγόριθμο ανάθεσης και από μία προσεκτικά σχεδιασμένη συνάρτηση πληρωμής. Η συνάρτηση πληρωμής είναι το άθροισμα δύο όρων, της αποζημίωσης (compensation) και του πριμ (bonus).

Ορισμός 2.1.21 (Αποζημίωση) Ορίζουμε τη *συνάρτηση αποζημίωσης* του παίχτη i ως:

$$c^i(t,\tilde{t}) = \sum_{j \in x^i(t)} \tilde{t}_j$$

Δηλαδή η συνάρτηση αποζημίωσης υπολογίζει για κάθε παίχτη το άθροισμα των πραγματικών χρόνων στους οποίους εκτέλεσε τις εργασίες που του ανατέθηκαν. Υπενθυμίζουμε ότι η ανάθεση καθορίζεται από τους χρόνους που δήλωσαν οι παίχτες. Γι' αυτό και η αποζημίωση είναι συνάρτηση των t και $\tilde{t}$.

Ορισμός 2.1.22 (Διάνυσμα Διορθωμένου Χρόνου)
Έστω παίχτης i και x μια ανάθεση. Δοθέντος της δήλωσης t των παιχτών και του διανύσματος των πραγματικών χρόνων $\tilde{t}$, ορίζουμε το *διάνυσμα διορθωμένου χρόνου (corrected time vector)* για τον παίχτη i ως :

$$corr^i(x,t,\tilde{t}) = \begin{cases} \tilde{t}_j & \text{εάν } j \in x^i \\ t^l_j & \text{εάν } j \in x^l \text{ και } l \neq i \end{cases}$$

Ορίζουμε με $corr^*(x,t)$ του x και t, το μοναδικό διάνυσμα που ικανοποιεί την $corr^*(x,t)_j = t^i_j$ για όλα τα i και $j \in x^i$.

Ερμηνεία: Έστω ανάθεση εργασιών x και παίχτης i. Το διάνυσμα διορθωμένου χρόνου για τον i θα έχει μήκος k (ο αριθμός των εργασιών) και θα αποτελείται a) από τους χρόνους εκτέλεσης που δήλωσαν οι παίχτες για τις εργασίες που τους ανατέθηκαν b) από τους πραγματικούς χρόνους του παίχτη i στους οποίους εκτέλεσε τις εργασίες που του ανατέθηκαν. Γι' αυτό και ονομάζεται διάνυσμα διορθωμένου χρόνου, διότι περιέχει τους πραγματικούς χρόνους εκτέλεσης του παίχτη i και όχι τους χρόνους που αρχικά δήλωσε.

Ορισμός 2.1.23 (Πριμ) Η συνάρτηση

$$b^i(t,\tilde{t}) = -g\left(x(t), corr^i(x(t),t,\tilde{t})\right)$$

καλείται *συνάρτηση πριμ* του παίχτη i.

Παρατηρήσεις: Το πριμ για τον i, είναι η αρνητική τιμή της αντικειμενικής συνάρτησης με βάση την ανάθεση x και τους χρόνους που περιέχονται στο διάνυσμα διορθωμένου χρόνου του i. Ο μηχανισμός υπολογίζει την βέλτιστη ανάθεση των εργασιών με βάση τους χρόνους που του δήλωσαν οι παίχτες. Αν οι παίχτες είναι ειλικρινείς τότε υπολογίζει την βέλτιστη ανάθεση των εργασιών. Έτσι αν οι παίχτες εκτελέσουν τις εργασίες στους χρόνους που δήλωσαν τότε θα επιτευχθεί το μικρότερο make-span. Όμως σε ένα μηχανισμό με επαλήθευση, όπως αναφέραμε στον ορισμό 2.1.21, μπορεί τελικά οι παίχτες να εκτελέσουν οποιαδήποτε εργασία σε μεγαλύτερο χρόνο από αυτόν που δήλωσαν. Τότε όμως το make-span θα είναι μεγαλύτερο του βέλτιστου. Η συνάρτηση πριμ υπολογίζει το make-span με βάση τους δηλωμένους χρόνους των άλλων παιχτών και τους πραγματικούς για τον i. Έτσι έχουμε κατά κάποιο τρόπο μία μέτρηση του κατά πόσο επιβάρυνε ή όχι ο παίχτης i , μετά την εκτέλεση των εργασιών του, το βέλτιστο make-span.

Στη συνέχεια θα ορίσουμε τον μηχανισμό αποζημίωσης και πριμ, χρησιμοποιώντας τις συναρτήσεις αποζημίωσης και πριμ.

Ορισμός 2.1.24 (Ο Μηχανισμός Αποζημίωσης και Πριμ)
Ο *Μηχανισμός Αποζημίωσης και Πριμ (Compensation and Bonus Mechanism)* αποτελείται από έναν βέλτιστο αλγόριθμο ανάθεσης και με συνάρτηση πληρωμής:

$$p^i(t,\tilde{t}) = c^i(t,\tilde{t}) + b^i(t,\tilde{t}).$$

Παρατηρήσεις: Η πληρωμή που ορίζει ο μηχανισμός για τον i είναι απλά το άθροισμα της αποζημίωσης και του πριμ. Όπως έχουμε δει μέχρι τώρα η φιλαλήθεια ενός μηχανισμού βασίζεται κυρίως στο σωστό σχεδιασμό της συνάρτησης πληρωμών. Οι παίχτες μελετώντας την συνάρτηση πληρωμής θα πρέπει να καταλήγουν στο συμπέρασμα ότι μόνο αν είναι ειλικρινείς θα έχουν το μέγιστο κέρδος. Σε ένα μηχανισμό επαλήθευσης θέλουμε οι παίχτες να είναι φιλαλήθεις αλλά επίσης κατά την εκτέλεση των εργασιών να παραμείνουν πιστοί στη αληθή δήλωση τους. Γι' αυτό το σκοπό ορίζουμε την πληρωμή του i ως το άθροισμα της αποζημίωσης και του πριμ. Η utility του i είναι $u^i = p^i(t,\tilde{t}) + v^i(x,\tilde{t})$ με $v^i(x,\tilde{t}) = -\sum_{j\in x^i}\tilde{t}_j$ και εκφράζει το όφελος του παίχτη. Επίσης $c^i(t,\tilde{t}) = \sum_{j\in x^i(t)}\tilde{t}_j$. Συνεπώς επειδή $p^i(t,\tilde{t}) = c^i(t,\tilde{t}) + b^i(t,\tilde{t})$, η utility του i ισούται με το πριμ του. Άρα κάθε παίχτης θα προσπαθήσει να μεγιστοποιήσει το πριμ του κάτι που θα συμβεί μόνο αν μείνει πιστός στην αρχική φιλαλήθη δήλωση του. Υπενθυμίζουμε ότι το πριμ είναι η αρνητική τιμή του $g\left(x(t), corr^i(x(t),t,\tilde{t})\right)$.

Με βάση τα παραπάνω μπορούμε να αποδείξουμε το επόμενο θεώρημα.

Θεώρημα 2.1.25 Ο Μηχανισμός Αποζημίωσης και Πριμ είναι ισχυρά φιλαλήθης υλοποίηση του task scheduling προβλήματος.

Απόδειξη: Θα δείξουμε ότι για κάθε παίχτη η μοναδική κυρίαρχη στρατηγική είναι η αποκάλυψη του αληθινού τύπου του και η εκτέλεση των εργασιών του στον ελάχιστο χρόνο.

Έστω παίχτης i, t^i ο τύπος του και έστω d^{-i} οι δηλώσεις των άλλων παιχτών. Σημειώνουμε ότι η ανάθεση και το πριμ που δίνονται στον i εξαρτώνται από το d^{-i} αλλά όχι από τους πραγματικούς χρόνους εκτέλεσης των άλλων παιχτών. Έστω $t = (d^{-i}, t^i)$. Όπως είπαμε η utility του παίχτη ισούται με το πριμ του και κατά συνέπεια για κάθε ανάθεση x το πριμ του μεγιστοποιείται όταν εκτελέσει την ανάθεσή του στον ελάχιστο χρόνο. Αρκεί λοιπόν να δείξουμε ότι $-g(x(t), corr^*(x(t), t)) \geq -g(x(t'^i, d^{-i}), corr^*(x(t'^i, d^{-i}), t))$ για κάθε t'^i. Αυτό όμως προκύπτει άμεσα από το γεγονός ότι ο αλγόριθμός ανάθεσης είναι βέλτιστος. Συνεπώς αν ο παίχτης δεν ακολουθήσει αυτή τη στρατηγική τότε υπάρχει περίπτωση να αυξήσει το make-span και έτσι να μειώσει το πριμ του. Κατά συνέπεια η παραπάνω στρατηγική είναι η μοναδική κυρίαρχη.

Όταν όλοι οι παίχτες ακολουθούν την κυρίαρχη στρατηγική τότε επιτυγχάνεται το καλύτερο δυνατό make-span, λόγω του ότι ο αλγόριθμος ανάθεσης είναι ο βέλτιστος. ■

Παράδειγμα: Θεωρούμε τον πίνακα 2.1.3.

	j_1	j_2	j_3
A_1	**10**	30	**45**
A_2	100	**60**	100

Πίνακας 2.1.3: Πίνακας τύπων για δύο παίχτες

Θεωρούμε ότι και οι δύο παίχτες είναι φιλαλήθεις. Η βέλτιστη ανάθεση σε αυτή την περίπτωση είναι $\{\{j_1, j_3\}\{j_2\}\}$ και το make-span 60. Έτσι το πριμ που δίνεται σε κάθε παίχτη είναι –60. Ας θεωρήσουμε την περίπτωση στην οποία ο παίχτης 1 προσπαθεί να απαλλαγεί από την j_3 και δηλώνει t_3^1 ως 200. Τώρα το βέλτιστο make-span γίνεται 100 και το πριμ κάθε παίχτη είναι –100.Ομοίως αν ο παίχτης 1 προσπαθήσει να πάρει την j_2 δηλώνοντας π.χ. $t_2^1 = 4$, τότε το πριμ γίνεται –85. Επίσης αν ο παίχτης καθυστερήσει να εκτελέσει τις εργασίες του και κάνει 100 μονάδες χρόνου αντί για 55 τότε το πριμ του από –60 γίνεται –100.

Στη συνέχεια θα γενικεύσουμε τον προηγούμενο μηχανισμό και θα σχεδιάσουμε τον Γενικευμένο Μηχανισμό Αποζημίωσης και Πριμ

Ορισμός 2.1.26 (Γενικευμένη Αποζημίωση)
Η συνάρτηση $c^i(t,\tilde{t})$ καλείται *γενικευμένη αποζημίωση* για τον παίχτη i εάν:

1. $c^i(t,\tilde{t}) \le \sum_{j \in x^i(t)} \tilde{t}_j$ για όλα τα t και $\tilde{t}$.
2. Η ισότητα ισχύει στην περίπτωση που ο παίχτης είναι φιλαλήθης.

Ορισμός 2.1.27 (Γενικευμένο Πριμ) Έστω $m^i(t^{-i}, w)$ οποιαδήποτε θετική πραγματική συνάρτηση η οποία είναι γνησίως αύξουσα συνάρτηση ως προς w. Η συνάρτηση $b^i(t,\tilde{t}) = m^i\left(t^{-i}, -g\left(x(t), corr^i(t,\tilde{t})\right)\right)$ καλείται *γενικευμένο πριμ* (generalized bonus) για τον παίχτη i.

Παρατηρούμε ότι καθώς η m^i είναι γνησίως αύξουσα ως προς w για μεγαλύτερες τιμές του $-g\left(x(t), corr^i(t,\tilde{t})\right)$ δηλαδή για μικρότερα make-span θα δίνει μεγαλύτερες τιμές του γενικευμένου πριμ. Συνεπώς και το γενικευμένο πριμ συμπεριφέρεται όμοια με το πριμ που ορίσαμε προηγουμένως.

Ορισμός 2.1.28 (Ο Γενικευμένος Μηχανισμός Αποζημίωσης και Πριμ)
Ο *Γενικευμένος Μηχανισμός Αποζημίωσης και Πριμ* είναι το ζεύγος $m = (o,p)$ όπου :

- $o(\ )$ είναι ένας βέλτιστος αλγόριθμος ανάθεσης.
- $p^i(t,\tilde{t}) = c^i(t,\tilde{t}) + b^i(t,\tilde{t})$, όπου $c^i(\ )$ και $b^i(\ )$ είναι αντίστοιχα οι γενικευμένες συναρτήσεις αποζημίωσης και πριμ.

Όμοια με το θεώρημα 2.1.25 αποδεικνύεται ότι:

Θεώρημα 2.1.29 Ο Γενικευμένος Μηχανισμός Αποζημίωσης και Πριμ είναι ισχυρά φιλαλήθης υλοποίηση του task scheduling προβλήματος.

2.1.6 Περιορισμένο Task Scheduling Πρόβλημα

Στην ενότητα αυτή θα επεκτείνουμε το μοντέλο των μηχανισμών με επαλήθευση. Θα εισάγουμε στο μοντέλο αυτό δύο πρόσθετες απαιτήσεις: a) περιορισμούς συμμετοχής και b) όρια προϋπολογισμού. Θα δείξουμε ότι υπάρχει και στο μοντέλο αυτό φιλαλήθης μηχανισμός για το task scheduling.

Ορισμός 2.1.30 (Περιορισμοί Συμμετοχής) Θα λέμε ότι ένας μηχανισμός ικανοποιεί *περιορισμούς συμμετοχής (participation constraints)*, εάν ισχύει ότι όταν ο παίχτης είναι φιλαλήθης τότε η utility του δεν είναι αρνητική. Συγκεκριμένα, για κάθε t και $\tilde{t}$ και για κάθε παίχτη i , εάν $\tilde{t}_j = t_j$ για κάθε $j \in x^i(t)$, τότε $p^i(t,\tilde{t}) + v^i(x(t),\tilde{t}) \ge 0$.

Οι περιορισμοί συμμετοχής μας εγγυώνται ότι ένας παίχτης θα έχει πάντα κέρδος με το να είναι φιλαλήθης.

Ορισμός 2.1.31 (Συνεισφορά) Ορίζουμε την ***συνεισφορά*** *(contribution)* του παίχτη i ως:

$$\mathrm{cont}^{i}\left(t^{-i},\tilde{t}\right)=g^{-i}\left(t^{-i}\right)-g\left(x(t),\mathrm{corr}^{i}\left(x(t),t,\tilde{t}\right)\right).$$

Όπου $g^{-i}\left(t^{-i}\right)$ είναι το βέλτιστο make-span από τις αναθέσεις που δεν περιλαμβάνουν τον παίχτη i.

Σχόλια: Η συνεισφορά ουσιαστικά υπολογίζει την συμμετοχή του παίχτη i στο make-span. Για να βρούμε την συνεισφορά του παίχτη i πρώτα υπολογίζουμε το βέλτιστο make-span από τους χρόνους που δήλωσαν οι υπόλοιποι παίχτες και χωρίς την συμμετοχή του παίχτη i. Μετά υπολογίζουμε το πριμ του παίχτη i, $-g\left(x(t),\mathrm{corr}^{i}\left(x(t),t,\tilde{t}\right)\right)$. Η συνεισφορά του i θα προκύψει από το άθροισμα των παραπάνω τιμών. Αν η συνεισφορά έχει θετική τιμή τότε ο παίχτης με τη συμμετοχή του βοηθά στην επίτευξη καλύτερου make-span. Παρατηρούμε ότι για τον υπολογισμό της συμμετοχής του i λαμβάνουμε υπόψη τους πραγματικούς χρόνους εκτέλεσης του i.

Θεώρημα 2.1.32 Υπάρχει ισχυρά φιλαλήθης μηχανισμός για το task scheduling πρόβλημα που ικανοποιεί περιορισμούς συμμετοχής.

Απόδειξη: Ένας Γενικευμένος Μηχανισμός Αποζημίωσης και Πριμ στον οποίο το πριμ κάθε παίχτη ισούται με την συνεισφορά του θα ικανοποιεί τους περιορισμούς συμμετοχής. Όπως δείξαμε στο θεώρημα 2.1.29 ο Γενικευμένος Μηχανισμός Αποζημίωσης και Πριμ είναι ισχυρά φιλαλήθης μηχανισμός για το task scheduling πρόβλημα. ■

Μέχρι τώρα δεν έχουμε θέσει κάποιο όριο στο χρηματικό ποσό που ο μηχανισμός μπορεί να πληρώσει στους παίχτες. Αν όμως θεωρήσουμε μηχανισμούς με όριο στον προϋπολογισμό τους, τότε υπάρχει το ενδεχόμενο να μην μπορεί ο μηχανισμός να κάνει καμία ανάθεση στους παίχτες, για κάποιο στιγμιότυπο του προβλήματος. Γι' αυτό το λόγο θα αλλάξουμε τον ορισμό του προβλήματος.

Ορισμός 2.1.33 (Περιορισμένο Task Scheduling πρόβλημα) Το πρόβλημα ορίζεται ομοίως με το task scheduling με επαλήθευση, με τη διαφορά ότι τώρα υπάρχει μία επιπλέον πιθανή έξοδος $\perp$ που αντιστοιχεί στην περίπτωση όπου δεν γίνεται καθόλου ανάθεση, έτσι $v^{i}(\perp)=0$ και $g(\perp)=\infty$.

Αν θεωρήσουμε τον μηχανισμό ως ένα αλγόριθμο, τότε με την έξοδο $\perp$ του δίνουμε την δυνατότητα να τερματίζει πάντα, ακόμα και όταν δεν μπορεί να κάνει κάποια ανάθεση.

Ορισμός 2.1.34 (Περιορισμένος Προϋπολογισμός)

1. Θα λέμε ότι ένας μηχανισμός $m(o,p)$ έχει *περιορισμένο προϋπολογισμό b*, αν για κάθε n-άδα στρατηγικών $s=s^{1},\ldots,s^{n}$, $\sum_{i}p^{i}(s)\le b$.

2. Θα καλούμε έναν μηχανισμό άμεσης αποκάλυψης *b-περιορισμένο (b-constrained)* εάν ικανοποιεί τους περιορισμούς συμμετοχής και έχει περιορισμένο προϋπολογισμό b.

Στον ορισμό η συνθήκη 1 εκφράζει το γεγονός ότι όποια και αν είναι η στρατηγική των παιχτών ο μηχανισμός δεν μπορεί να πληρώσει στους παίχτες συνολικό ποσό μεγαλύτερο του b. Το b είναι το μέγιστο χρηματικό ποσό που μπορεί να διαθέσει.

Σημείωση: Αν $Y(t)$ είναι η ανάθεση ενός b-περιορισμένου μηχανισμού τότε πρέπει να ισχύει $\sum_i \sum_{j \in Y^i} t_j^i \le b$. Αυτό προκύπτει από τις συναρτήσεις πληρωμής οι οποίες υπολογίζουν τις πληρωμές με βάση τους χρόνους των παιχτών.

Ορισμός 2.1.35 Έστω $m = (o, p)$ και $m' = (o', p')$ δύο μηχανισμοί για το ίδιο πρόβλημα ελαχιστοποίησης. Θα λέμε ότι *ο m είναι τόσο καλός όσο ο m΄*, εάν για οποιασδήποτε κυρίαρχες στρατηγικές $d = d^1, \ldots, d^n$ των παιχτών στον m και $s = s^1, \ldots, s^n$ στον m΄ ισχύει, $g(o(d)) \le g(o'(s))$.

Δηλαδή ο m δεν θα δίνει ποτέ χειρότερα αποτελέσματα από τον m΄.

Θεώρημα 2.1.36 Έστω $b, h > 0$. Τότε υπάρχει φιλαλήθης μηχανισμός m για το περιορισμένο task scheduling πρόβλημα έτσι ώστε:

1. m είναι $(b + h)$ – περιορισμένος.
2. Εάν ο m΄ είναι b-περιορισμένος, τότε ο m *είναι τόσο καλός όσο* ο m΄.

Απόδειξη: Θεωρούμε τον ακόλουθο μηχανισμό $m = (o, p)$:

- Ο αλγόριθμος εξόδου $o(t)$ ψάχνει μια βέλτιστη ανάθεση μεταξύ των αναθέσεων y έτσι ώστε $\sum_i \sum_{j \in y^i} t_j^i \le b$ ή $\perp$ αν δεν υπάρχει ανάθεση.
- Έστω $m(\ )$ μία γνησίως αύξουσα μονότονη συνάρτηση για την οποία $0 \le m(\ ) \le h/n$. Ορίζουμε την συνεισφορά $cont^i$ του παίχτη όπως στο θεώρημα 2.1.32, εκτός από το ότι ο αλγόριθμος και η αντικειμενική συνάρτηση είναι διαφορετικές. Ορίζουμε το πριμ $b^i(\ )$ ως $m(cont^i(\ ))$ και την συνάρτηση πληρωμής ως $p^i = c^i(\ ) + b^i(\ )$, όπου $c(\ )$ είναι η συνάρτηση αποζημίωσης. Η πληρωμή ορίζεται να είναι 0 όταν η έξοδος είναι $\perp$.

Η συνολική αποζημίωση είναι φραγμένη από το b και το συνολικό πριμ από το h, κατά συνέπεια ο προϋπολογισμός είναι φραγμένος από το $b + h$. Όπως και στο θεώρημα 2.1.25 μπορούμε να δείξουμε ότι ο μηχανισμός είναι φιλαλήθης και ότι η μοναδική κυρίαρχη για τον παίχτη i είναι να αποκαλύψει την αλήθεια (όταν

$t^i_j \geq b$ ο παίχτης μπορεί να δηλώσει οποιοδήποτε $d^i_j \geq b$) και να εκτελέσει τις εργασίες του όσο πιο αποδοτικά μπορεί.

Υπενθυμίζουμε ότι κάθε b-περιορισμένος μηχανισμός m΄ πρέπει να επιλέξει ανάθεση z ώστε : $\sum_i \sum_{j \in z^i} t^i_j \leq b$. ■

2.1.7 Μηχανισμοί Πολυωνυμικού Χρόνου

Ο Μηχανισμός Αποζημίωσης και Πριμ είναι βέλτιστος αλλά υπολογιστικά δύσκολος, διότι ο αλγόριθμος ανάθεσης που χρησιμοποιεί είναι μεν βέλτιστος αλλά εκθετικού χρόνου. Υπενθυμίζουμε ότι το task scheduling είναι πρόβλημα NP-hard και κατά συνέπεια όλοι οι γνωστοί βέλτιστοι αλγόριθμοι γι' αυτό είναι εκθετικού χρόνου. Στην ενότητα αυτή θα μελετήσουμε μηχανισμούς αποζημίωσης και πριμ που δεν χρησιμοποιούν βέλτιστο αλγόριθμο ανάθεσης αλλά κάποιο πολυωνιμικό προσεγγιστικό αλγόριθμο. Θα δείξουμε ότι η αντικατάσταση του βέλτιστου αλγορίθμου με έναν προσεγγιστικό πολυωνιμικού χρόνου δεν οδηγεί πάντα σε φιλαλήθη μηχανισμό. Έτσι θα ορίσουμε μία υποπερίπτωση του προβλήματος στο οποίο ο αριθμός των παικτών είναι σταθερά καθορισμένος και οι χρόνοι εκτέλεσης t^i_j είναι φραγμένοι. Το πρόβλημα παρόλο τους περιορισμούς παραμένει NP-hard και τα κάτω φράγματα που δείξαμε στην ενότητα 2.1.3 ισχύουν και γι' αυτό. Τέλος θα δείξουμε ότι για κάθε $\varepsilon > 0$ μπορούμε να έχουμε ένα πολυωνιμικό 1+ε προσεγγιστικό μηχανισμό για το πρόβλημα.

Ορισμός 2.1.37 Έστω x( ) ένας αλγόριθμος ανάθεσης. Ορίζουμε τον *Μηχανισμό Αποζημίωσης και Πριμ που βασίζεται στον x* με όμοιο τρόπο με τον Μηχανισμό Αποζημίωση και Πριμ, εκτός από το ότι αντί για τον βέλτιστο αλγόριθμο ανάθεσης έχουμε τον x( ).

Το επόμενο θεώρημα είναι πολύ σημαντικό για την σχεδίαση πολυωνιμικών μηχανισμών.

Θεώρημα 2.1.38 Έστω x( ) ένας προσεγγιστικός αλγόριθμος για το task scheduling. Έστω $m = (x, p)$ είναι Μηχανισμός Αποζημίωσης και Πριμ που βασίζεται στον x( ). Τότε ο m δεν είναι φιλαλήθης.

Απόδειξη: Υποθέτουμε ότι ο m είναι φιλαλήθης. Για κάποια ανάθεση y και τύπο t έστω $g(y,t)$ δηλώνει το make-span, $\max_i \sum_{j \in y^i} t^i_j$ και opt(t) δηλώνει τον βέλτιστο αλγόριθμο ανάθεσης. Έστω t τύπος για τον οποίο ισχύει $g(opt(t),t) < g(x(t),t)$ και έστω t'^1 ο τύπος του παίχτη 1 για τον οποίο:

$t'^1_j = \begin{cases} t^1_j & \text{εάν } j \in opt^1(t) \\ \infty & \text{αλλιώς} \end{cases}$ όπου το ∞ αντιπροσωπεύει μια αυθαίρετα μεγάλη τιμή.

Ισχυρισμός Έστω $t' = t'^1, t^2, \ldots, t^n$. Τότε $g(x(t'), t') \geq g(x(t), t)$.

Απόδειξη: Έστω ότι δεν ισχύει. Τότε αν ο τύπος του παίχτη 1 είναι t^1 ,θα ήταν ωφέλιμο γι' αυτόν να υποκριθεί ότι είναι t'^1. Αυτό όμως έρχεται σε αντίθεση με την υπόθεση ότι ο μηχανισμός είναι φιλαλήθης. ■

Πόρισμα 2.1.39 Έστω s τύπος για τον οποίο:

$$s_j^i = \begin{cases} t_j^i & \text{εάν } j \in opt^i(t) \\ \infty & \text{αλλιώς} \end{cases}$$

Τότε

$$g(x(s), s) \geq g(x(t), t).$$

■

Καθώς $g(x(s), s) \geq g(x(t), t) > g(opt(t), t) = g(opt(s), s)$, έχουμε ότι $x(s) \neq opt(s)$. Τότε θα υπάρχει παίχτης που του ανατίθεται άπειρη εργασία, άτοπο καθώς ο x( ) είναι προσεγγιστικός. ■

Το θεώρημα αυτό μας δίνει ένα πολύ σημαντικό αποτέλεσμα. Δηλώνει ότι δεν μπορούμε εύκολα να σχεδιάσουμε κάποιο πολυωνιμικό μηχανισμό που να υλοποιεί το task scheduling. Δεν μπορούμε απλά να αντικαταστήσουμε τον βέλτιστο αλγόριθμο ανάθεσης με κάποιο προσεγγιστικό στον μηχανισμό Μηχανισμός Αποζημίωσης και Πριμ. Κάτι τέτοιο θα έδινε μηχανισμό που δεν είναι πλέον φιλαλήθης και συνεπώς δεν μπορεί να χρησιμοποιηθεί. Θα προσπαθήσουμε στη συνέχεια να ξεπεράσουμε την δυσκολία που εκφράζει το θεώρημα 2.1.38 με το να αλλάξουμε τον ορισμό του προβλήματος.

Ορισμός 2.1.40 (Φραγμένο Task Scheduling Πρόβλημα)
Ο ορισμός του προβλήματος είναι όμοιος με τον ορισμό του task scheduling με επαλήθευση με τη διαφορά ότι ο αριθμός των παιχτών είναι σταθερός και υπάρχουν σταθερές $b > \alpha > 0$ έτσι ώστε: για κάθε i, j $\alpha \leq t_j^i \leq b$.

Δηλαδή ο αριθμός πλέον των παιχτών είναι σταθερός και θα δίνεται μαζί με τον ορισμό του προβλήματος.. Επίσης οι χρόνοι των παιχτών είναι φραγμένοι από τους αριθμούς α,b.

Ένας 1+ε –προσεγγιστικός αλγόριθμος για το φραγμένο task scheduling πρόβλημα είναι ο ***αλγόριθμος στρογγυλοποίησης***. Ο αλγόριθμος αυτός λειτουργεί ως εξής: Οι είσοδοι t_j^i αρχικά στρογγυλοποιούνται σε ακέραιους πολλαπλάσια του δ. Το δ είναι παράμετρος που εξαρτάται από τα α και ε. Μετά επιλύεται αυτό το (στρογγυλοποιημένο) πρόβλημα χρησιμοποιώντας δυναμικό προγραμματισμό, σε πολυωνιμικό χρόνο. Αποδεικνύεται ότι η λύση του στρογγυλοποιημένου προβλήματος είναι 1+ε προσεγγιστική του αρχικού προβλήματος.

Στη συνέχεια θα σχεδιάσουμε έναν μηχανισμό χρησιμοποιώντας τον

αλγόριθμο στρογγυλοποίησης. Θα συνδέσουμε τον αλγόριθμο αυτό με κατάλληλη συνάρτηση πληρωμής, χρησιμοποιώντας τους ακριβείς χρόνους για την συνάρτηση αποζημίωσης, αλλά τους στρογγυλοποιημένους για την συνάρτηση πριμ.

Σημείωση: Έστω διάνυσμα t, με $\hat{t}$ θα καλούμε το διάνυσμα όπου όλα τα στοιχεία του έχουν στρογγυλοποιηθεί σε ακέραιο πολλαπλάσιο του δ. Θα δηλώνουμε επίσης με $\hat{g}(x,\tilde{t}) = g\left(x,\hat{\tilde{t}}\right)$, όπου g είναι η αντικειμενική συνάρτηση.

Ορισμός 2.1.41 (Ο Μηχανισμός Στρογγυλοποίησης)
Ο *Μηχανισμός Στρογγυλοποίησης (Rounding Mechanism)* ορίζεται ως εξής:

- Ο αλγόριθμος ανάθεσης είναι ο αλγόριθμος στρογγυλοποίησης
- Η συνάρτηση πληρωμής δίνεται από τη σχέση:

$$p^i(t,\tilde{t}) = c^i(t,\tilde{t}) + b^i(t,\tilde{t})$$

$$\text{όπου } c^i(t,\tilde{t}) = \sum_{j \in x^i(t)} \tilde{t}_j \quad \text{και} \quad b^i(t,\tilde{t}) = -\hat{g}\left(x(\hat{t}), \hat{corr}^i(x(t),t,\tilde{t})\right)$$

Ο μηχανισμός στρογγυλοποίησης αποζημιώνει τους παίχτες βάση της εργασίας τους, αλλά υπολογίζει το πριμ με βάση τους στρογγυλοποιημένους χρόνους δήλωσης και εκτέλεσης.

Αποδεικνύεται το ακόλουθο θεώρημα :

Θεώρημα 2.1.42 Για κάθε σταθερά $\varepsilon > 0$ ο μηχανισμός στρογγυλοποίησης είναι πολυωνιμικός μηχανισμός με επαλήθευση. Επίσης είναι φιλαλήθης 1+ε προσεγγιστική υλοποίηση του φραγμένου task scheduling προβλήματος.

2.2 Shortest Path

2.2.1 Εισαγωγή

Όπως είδαμε και στην ενότητα 2.1 για το task scheduling, για να λύσουμε ένα αλγοριθμικό πρόβλημα στο οποίο έχουμε ελλιπή πληροφορία για τους παίχτες που συμμετέχουν, προσπαθούμε να σχεδιάσουμε ένα VCG μηχανισμό που να υλοποιεί το πρόβλημα αυτό. Επιλέγουμε τους VCG μηχανισμούς διότι είναι πάντα φιλαλήθεις. Η φιλαλήθεια είναι απαραίτητο χαρακτηριστικό του μηχανισμού που θα σχεδιάσουμε διότι μας εγγυάται ότι οι πληροφορίες που μας δηλώνουν οι παίχτες είναι αληθείς. Στην ενότητα αυτή θα μελετήσουμε το πρόβλημα της δρομολόγησης ελάχιστου μονοπατιού (shortest path routing) σε ένα δίκτυο επικοινωνίας..

Μια εκτενής μελέτη του προβλήματος έχει γίνει στο άρθρο [HS01] από τους John Hershberger και Sudhash Suri. Τα αποτελέσματα του άρθρου αυτού θα παρουσιάσουμε στην ενότητα που ακολουθεί.

Έστω ότι έχουμε ένα δίκτυο επικοινωνίας G και ένα σύνολο ατόμων στα οποία ανήκουν τα links του δικτύου. Το δίκτυο μοντελοποιείται από ένα κατευθυνόμενο γράφο G όπου κάθε ακμή του ανήκει σε ένα παίχτη. Οι παίχτες θεωρούνται λογικοί και ιδιοτελείς, δηλαδή ενδιαφέρονται να μεγιστοποιήσουν το προσωπικό τους κέρδος με τον αποδοτικότερο τρόπο. Κάθε παίχτης i έχει μία προσωπική πληροφορία $t^i \geq 0$ (που είναι ο τύπος του) και αντιστοιχεί στο κόστος για την αποστολή ενός μηνύματος μέσω της ακμής του. Το κόστος κάθε link είναι γνωστό μόνο στον παίχτη στον οποίο ανήκει. Στο G θεωρούμε δύο συγκεκριμένους κόμβους x , y και σκοπός μας είναι να βρούμε το μονοπάτι με το ελάχιστο κόστος (shortest path) από το x στο y, με σκοπό να στείλουμε ένα μήνυμα από το x στο y.

Η ανάγκη σχεδίασης ενός μηχανισμού που να υλοποιεί το πρόβλημα αυτό πηγάζει από το γεγονός ότι ο αποστολέας του μηνύματος δεν γνωρίζει τα κόστη των ακμών του γράφου. Αν τα γνώριζε τότε ο μηχανισμός δεν θα χρειαζόταν. Απλά θα χρησιμοποιούσε κάποιο αλγόριθμο που υπολογίζει το shortest path σε ένα γράφο. Πρέπει κατά συνέπεια να σχεδιάσει ένα φιλαλήθη μηχανισμό που να υλοποιεί το πρόβλημα και μέσω του μηχανισμού να εκμαιεύσει από τους παίχτες (τους ιδιοκτήτες των ακμών) τα κόστη των ακμών τους.

Θα αναφέρουμε τώρα το γενικό σχήμα των μηχανισμών που υλοποιούν το πρόβλημα της shortest path δρομολόγησης σε ένα δίκτυο μεταξύ δύο κόμβων x και y.

- Το διάνυσμα τύπων των παιχτών είναι $t = (t^1, t^2, \ldots, t^n)$. Για το πρόβλημα της δρομολόγησης σε ένα δίκτυο, θεωρούμε ότι κάθε παίχτης, συσχετίζεται με μία ακμή e και έχει τύπο t^e που είναι το κόστος της ακμής. Το κόστος αυτό το γνωρίζει μόνο ο ίδιος. Οι παίχτες καλούνται να δώσουν στο μηχανισμό τις τιμές των τύπων τους.
- Δοθέντος ενός διανύσματος τύπων, ο αλγόριθμος εξόδου του μηχανισμού υπολογίζει την έξοδο $o \in O$, με βάση το διάνυσμα τύπων των παιχτών. Οι εφικτές έξοδοι του μηχανισμού είναι όλα τα δυνατά μονοπάτια από το x στο y. Στόχος του μηχανισμού είναι να εντοπίσει το μονοπάτι ελαχίστου κόστους από το x στο y.
- Κάθε παίχτης έχει μία αποτίμηση (valuation) υ( ) που σχετίζεται με κάθε έξοδο o του μηχανισμού. Η συνάρτηση αποτίμησης $v^i(t^i, o)$ ποσοτικοποιεί την προτίμηση του παίχτη i για την έξοδο o όταν ο τύπος του είναι t^i. Έστω P ένα μονοπάτι που προέκυψε ως έξοδος από τον μηχανισμό. Εάν η ακμή e δεν ανήκει στο P, τότε η αποτίμηση του παίχτη e είναι 0, αλλιώς η αποτίμηση του παίχτη είναι $-t^e$.
- Επίσης ο μηχανισμός καθορίζει για κάθε παίχτη i την πληρωμή του p^i. Η συνάρτηση πληρωμών του μηχανισμού αποτελεί μέρος του σχεδίου του και ουσιαστικά καθορίζει τον μηχανισμό.
- Η utility του παίχτη i είναι $u^i = p^i + v^i(t^i, o)$ και εκφράζει το όφελος του παίχτη από την έξοδο o. Οι παίχτες θεωρούνται λογικοί και ιδιοτελείς. Κατά συνέπεια κάθε παίχτης επιθυμεί να βελτιστοποιήσει την utility του με τον καλύτερο τρόπο χωρίς να συνεργαστεί με άλλους παίχτες.

Από τους μηχανισμούς που έχουν το παραπάνω γενικό σχήμα μας ενδιαφέρουν οι VCG μηχανισμοί που είναι πάντα φιλαλήθεις. Όπως ήδη έχουμε αναφέρει στο Κεφ. 1, ένας μηχανισμός ανήκει στην οικογένεια VCG εάν:

1. Επιλέγει μια έξοδο που μεγιστοποιεί την αποτίμηση: $o(t) = \arg\max_o \sum_{i=1}^{n} v^i(t^i, o)$
2. Οι συναρτήσεις πληρωμής είναι: $p^i(t) = \sum_{j \neq i} v^j(t^j, o(t)) + h^i(t^{-i})$, όπου h^i είναι μία αυθαίρετη συνάρτηση των $t^{-i} = (t^1, \ldots, t^{i-1}, t^{i+1}, \ldots, t^n)$.

Παρατηρήσεις:

1. Στον παραπάνω ορισμό η συνθήκη 1 δηλώνει ότι σε ένα VCG μηχανισμό η έξοδος θα μεγιστοποιεί το άθροισμα των αποτιμήσεων των παιχτών. Το άθροισμα αυτό είναι η αντικειμενική συνάρτηση του προβλήματος. Η έξοδος λοιπόν του μηχανισμού θα πρέπει να μεγιστοποιεί την αντικειμενική συνάρτηση. Για το πρόβλημα μας, η αποτίμηση κάθε παίχτη που η ακμή του ανήκει στο shortest path ισούται με την αρνητική τιμή του κόστους της ακμής του. Η αντικειμενική συνάρτηση του προβλήματος ορίζεται ως το άθροισμα των αποτιμήσεων των παιχτών. Στόχος του μηχανισμού είναι να βρει το ελάχιστο μονοπάτι μεταξύ των x και y. Καθώς η αντικειμενική συνάρτηση είναι άθροισμα αρνητικών τιμών, το ελάχιστο μονοπάτι θα την μεγιστοποιεί.

2. Στη συνθήκη 2 ο πρώτος όρος της συνάρτησης πληρωμής είναι ίσος με το άθροισμα των αποτιμήσεων των υπολοίπων παιχτών για την έξοδο o. Ο δεύτερος όρος εξαρτάται από τις αποτιμήσεις των υπόλοιπων παιχτών όταν αγνοήσουμε τον παίχτη i. Θεωρούμε ότι η πληρωμή είναι από τους παίχτες προς το σύστημα, όταν οι παίχτες αγοράζουν και από το σύστημα στους παίχτες, όταν οι παίχτες πουλούν.

Θα ορίσουμε τώρα την συνάρτηση πληρωμής για το πρόβλημα που μελετούμε με βάση το VCG σχήμα. Όπως αναφέραμε το αποτέλεσμα που δίνει την μέγιστη συνολική αποτίμηση είναι το shortest path από το x στο y. Ο όρος $\sum_{j \neq i} v^j(t^j, o(t))$ στη συνάρτηση πληρωμής είναι ίσος με $-d(x, y; G|_{e=0})$, δηλαδή η αρνητική τιμή του κόστους του βέλτιστου μονοπατιού όταν στο κόστος αυτό δεν συμπεριλάβουμε το κόστος του i. Για τον δεύτερο όρο $h^i(t^{-i})$, θα χρησιμοποιήσουμε το $d(x, y; G \setminus e)$, που είναι το κόστος του βέλτιστου μονοπατιού αν δεν υπάρχει ο παίχτης e στο γράφο.

Η πληρωμή για την ακμή e ορίζεται ως:

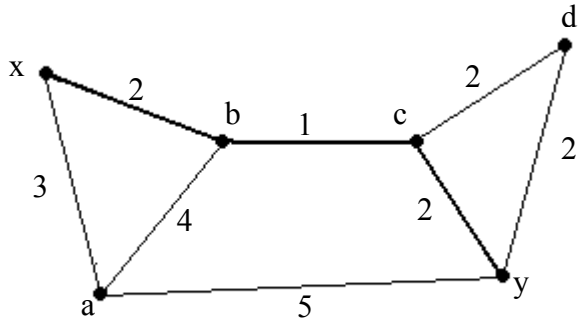
$$p^e = \begin{cases} d(x, y; G \setminus e) - d(x, y; G|_{e=0}) & \text{εάν e ανήκει στο shortest path} \\ 0 & \text{αλλιώς} \end{cases} \quad (2.2.1)$$

Παρατηρούμε ότι η πληρωμή κάθε παίχτη δεν εξαρτάται από το κόστος που

δήλωσε για την ακμή του. Η συνάρτηση 2.2.1 ανήκει στην VCG κλάση.

Παρατηρήσεις: Το πρόβλημα εύρεσης των ελάχιστων μονοπατιών σε ένα γράφο είναι πολυωνιμικού χρόνου. Σε ένα γράφο $G = (V, E)$ με $|V| = n$ και $|E| = m$ και θετικά βάρη στις ακμές του, μπορούμε με τον αλγόριθμο του Dijkstra να βρούμε το shortest path από το x στο y σε χρόνο $O(n \log n + m)$. Αν το γράφημα έχει και αρνητικά κόστη αλλά δεν έχει κύκλους αρνητικού κόστους, τότε με τον αλγόριθμο Bellman-Ford υπολογίζουμε το shortest path σε χρόνο $O(nm)$. Με σκοπό να υπολογίσουμε τις πληρωμές για κάθε παίχτη e, παρατηρούμε ότι ο όρος $d(x, y; G|_{e=0})$ μπορεί να υπολογιστεί από το ελάχιστο μονοπάτι $d(x, y; G)$, απλά αφαιρώντας το κόστος του e. Ο όρος $d(x, y; G \setminus e)$ δεν είναι τόσο απλό για να υπολογιστεί, χρειάζεται να αφαιρέσουμε την ακμή e από το γράφημα κάτι που θα αλλάξει το shortest path και το μήκος του. Μία απλή μέθοδος είναι να τρέξουμε ξανά τον αλγόριθμο του shortest path στο $G \setminus e$ για κάθε e στο shortest path. Επειδή στο shortest path μεταξύ του x και y υπάρχουν το πολύ n-1 ακμές, μπορούμε με αυτή τη μέθοδο, να υπολογίσουμε τις πληρωμές για όλους τους παίχτες σε χρόνο $O(n^2 \log n + nm)$ (για δίκτυο με θετικά βάρη) ή σε $O(n^2 m)$ (για δίκτυο με αρνητικά βάρη). Άρα ένας τέτοιος μηχανισμός θα είναι πολυωνιμικού χρόνου. Σε αυτή την ενότητα θα δείξουμε ότι μπορούμε να επιτύχουμε κάτι καλύτερο, ότι τελικά μπορούμε να υπολογίσουμε όλες τις πληρωμές σε χρόνο ασυμπτωτικά ίσο με τον χρόνο υπολογισμού ενός shortest path.

Παράδειγμα: Ας δούμε τώρα ένα παράδειγμα για τον προηγούμενο VCG μηχανισμό. Έστω ο μη κατευθυνόμενος γράφος της εικόνας 2.2.1. Θέλουμε να βρούμε το ελάχιστο μονοπάτι που συνδέει το x και y



Εικόνα 2.2.1 Το γραμμοσκιασμένο μονοπάτι είναι το shortest path.

Έστω κάποιος φιλαλήθης μηχανισμός που υλοποιεί το πρόβλημα. Ο μηχανισμός αρχικά ζητά από τους παίχτες να του δηλώσουν τα κόστη των ακμών τους. Επειδή ο μηχανισμός είναι φιλαλήθης οι παίχτες θα δηλώσουν τα πραγματικά κόστη των ακμών. Στην ανάλυση που ακολουθεί ταυτίζουμε τους παίχτες με τις ακμές τους. Στη συνέχεια βάση αυτών των δηλώσεων ο μηχανισμός εφαρμόζει τον αλγόριθμο εξόδου (τον αλγόριθμο του Dijkstra). Έτσι βρίσκει το

ελάχιστο μονοπάτι {(x,b), (b,c), (c,y)} που έχει κόστος 5.

Μετά υπολογίζει τις πληρωμές των παιχτών με την συνάρτηση 2.2.1. Για να βρει την πληρωμή της (x,b) υπολογίζει πρώτα τον όρο $d(x,y;G\setminus(x,b))$, δηλαδή το κόστος του ελάχιστου μονοπατιού αν δεν υπήρχε η ακμή (x,b) στο γράφο. Αυτό είναι το {(x,a), (a,y)} με κόστος 8. Στη συνέχεια υπολογίζει τον όρο $d(x,y;G|_{(x,b)=0})$ δηλαδή το κόστος του ελάχιστου μονοπατιού χωρίς το κόστος της ακμής (x,b). Το κόστος αυτό είναι 3. Κατά συνέπεια η πληρωμή της (x,b) είναι :

$$p^{(x,b)} = d(x,y;G\setminus(x,b)) - d(x,y;G|_{(x,b)=0}) = 8-3 = 5 .$$

Για την πληρωμή της (b,c) ο μηχανισμός εργάζεται ομοίως. Το ελάχιστο μονοπάτι χωρίς την ακμή (b,c) είναι το {(x,a), (a,y)}. Έτσι

$$p^{(b,c)} = d(x,y;G\setminus(b,c)) - d(x,y;G|_{(b,c)=0}) = 8-4 = 4 .$$

Το ελάχιστο μονοπάτι χωρίς την ακμή (c,y) είναι το {(x,b), (b,c), (c,d), (d,y)} κόστους 7. Άρα

$$p^{(c,y)} = d(x,y;G\setminus(c,y)) - d(x,y;G|_{(c,y)=0}) = 7-3 = 4$$

2.2.2 Το Μοντέλο

Θεωρούμε ότι το δίκτυο μοντελοποιείται από ένα γράφο $G=(V,E)$, με $|V|=n$ και $|E|=m$. Κάθε ακμή $e\in G$ έχει κόστος $c(e)$. Θα μελετήσουμε κατευθυνόμενα και μη κατευθυνόμενα γραφήματα. Θα προτείνουμε αρχικά έναν αλγόριθμο για μη κατευθυνόμενα γραφήματα διότι η ανάλυση τους είναι απλούστερη. Θεωρούμε ότι κάθε ζεύγος κόμβων u και υ συνδέεται το πολύ με μία ακμή. Αυτός ο περιορισμός πάντως δεν είναι απαραίτητος για τον αλγόριθμο που θα παρουσιάσουμε, ο αλγόριθμος λειτουργεί το ίδιο καλά και για γράφο του οποίου τα ζεύγη κόμβων συνδέονται με πολλές παράλληλες ακμές .

Ένα μονοπάτι στο G είναι μία ακολουθία ακμών, όπου συνδεόμενες ακμές μοιράζονται ένα κοινό κόμβο και κάθε κόμβος να ανήκει το πολύ σε δύο ακμές του μονοπατιού. Το συνολικό κόστος ενός μονοπατιού στο G είναι το άθροισμα από τα κόστη των ακμών του μονοπατιού.

Το shortest path μεταξύ δύο κόμβων α και b ορίζεται ως $path(\alpha,b)$, είναι το μονοπάτι που συνδέει τους α , b και έχει το ελάχιστο κόστος. Η απόσταση μεταξύ α και b ορίζεται ως $d(\alpha,.b)$ και είναι το μήκος του path(α,b). Αν δεν υπάρχει μονοπάτι θεωρούμε την απόσταση αυτή άπειρη.

Σημείωση: Για ευκολία θεωρούμε ότι τα μήκη όλων των μονοπατιών είναι διαφορετικά , έτσι το shortest path μεταξύ δύο κόμβων θα είναι μοναδικό.

Θεωρούμε στο γράφημα δύο συγκεκριμένους κόμβους x και y που θα καλούμε πηγή (source) και στόχος (target). Το shortest path που τους συνδέει είναι το $path(x,y)=(\upsilon_1,\upsilon_2,\ldots,\upsilon_k)$, όπου $\upsilon_1=x$ και $\upsilon_k=y$. Όπως αναφέραμε προηγουμένως θέλουμε να υπολογίσουμε για κάθε $i\in\{1,\ldots,k-1\}$ το μήκος του

shortest path από το x στο y αν δεν χρησιμοποιήσουμε την ακμή $e_i = (\upsilon_i, \upsilon_{i+1})$. Την απόσταση αυτή θα καλούμε *x-y απόσταση παραλειπόμενης της* e_i, είναι ο όρος $d(x, y; G \setminus e_i)$ στη συνάρτησης πληρωμής 2.2.1. Έστω το shortest path $path(\alpha, b)$ μεταξύ των κόμβων α και b. Ορίζουμε ως $path(\alpha, b; G \setminus e)$ και $d(\alpha, b; G \setminus e)$, το shortest path και την ελάχιστη απόσταση όταν έχει απομακρυνθεί η ακμή e από το γράφημα G. Η απόσταση $d(\alpha, b)$ στο πλήρες γράφημα είναι συντομογραφία του $d(\alpha, d; G)$, όπως $path(\alpha, b)$ είναι συντομογραφία του $path(\alpha, b; G)$.

Μπορούμε να ορίσουμε το shortest path από το x στο y με βάση ένα σύνολο πλευρών που διασχίζουν ένα cut. Θεωρούμε μία οποιαδήποτε διαμέριση του συνόλου V των κόμβων σε δύο σύνολα V_x και V_y έτσι ώστε $x \in V_x$ και $y \in V_y$. Το σύνολο των κόμβων που διασχίζουν το cut ορίζεται ως $E(V_x, V_y)$. Κάθε ακμή $(u, \upsilon) \in E(V_x, V_y)$ έχει $u \in V_x$ και $\upsilon \in V_y$. Κάθε μονοπάτι από το x στο y πρέπει να περιέχει τουλάχιστον μία ακμή από το $E(V_x, V_y)$. Συνεπώς μπορούμε να διατυπώσουμε το πρόβλημα ως έξης:
Για κάθε $e_i = (\upsilon_i, \upsilon_{i+1})$ και κάποιο cut (V_x, V_y) που πιθανώς εξαρτάται από το e_i, η x-y απόσταση παραλειπόμενης της e_i υπολογίζεται από τη σχέση:

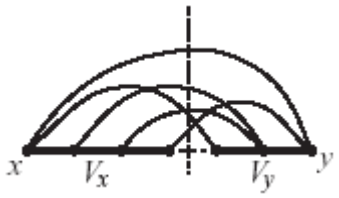
$$d(x, y; G \setminus e_i) = \min_{\substack{(u,\upsilon) \in E(V_x, V_y) \\ (u,\upsilon) \neq e_i}} (d(x, u; G \setminus e_i) + c(u, \upsilon) + d(\upsilon, y; G \setminus e_i)) \qquad (2.2.2)$$

Δηλαδή χωρίζουμε το shortest path, όταν έχει παραλειφθεί η e_i, σε τρία μέρη. Πρέπει να βρούμε εκείνη την ακμή (u,υ) στο cut $E(V_x, V_y)$ που θα ελαχιστοποιεί το παραπάνω άθροισμα. Την σχέση αυτή θα την χρησιμοποιήσουμε στις επόμενες ενότητες για την ανάλυση του προβλήματος .

2.2.3 Path Graphs

Αρχικά θα μελετήσουμε την ειδική περίπτωση όπου το $path(x, y)$ περιλαμβάνει όλους τους κόμβους του γραφήματος. Θα ορίζουμε το cut (V_x, V_y) βάση της διαμέρισης του $path(x, y) = (\upsilon_1, \upsilon_2, \ldots, \upsilon_n)$ όταν αφαιρέσουμε την ακμή $e_i = (\upsilon_i, \upsilon_{i+1})$. Επιλέγουμε $V_x = \{\upsilon_1, \ldots, \upsilon_i\}$ και $V_y = \{\upsilon_{i+1}, \ldots, \upsilon_n\}$. Για ευκολία ορίζουμε $E_i = E(V_x, V_y)$ για την ακμή e_i (εικόνα 2.2.2).

Έστω ότι πρόκειται να απομακρύνουμε την ακμή $e_i = (\upsilon_i, \upsilon_{i+1})$. Θεωρούμε μία ακμή $(\upsilon, u) \in E_i$ του cut. Επειδή το $path(x, u)$ περιέχεται στο V_x έχουμε $d(x, u) = d(x, u; G \setminus e_i)$. Ομοίως, επειδή το $path(\upsilon, y)$ περιέχεται στο V_y, $d(\upsilon, y) = d(\upsilon, y; G \setminus e_i)$. Οι αποστάσεις είναι τα μήκη των υπομονοπατιών του $path(x, y)$ που συνδέουν κάθε σημείο με το x και y .



Εικόνα 2.2.2: Οι ακμές του Ei διασχίζουν την διακεκομμένη γραμμή

Έτσι από τη σχέση (2.2.2) έχουμε:

$$d(x,y;G\setminus e_i)=\min_{\substack{(u,\upsilon)\in E_i\\(u,\upsilon)\neq e_i}} d(x,u)+c(u,\upsilon)+d(\upsilon,y) \qquad (2.2.3)$$

Για να υπολογίσουμε τα shortest paths όταν παραλείπουμε τις ακμές του $path(x,y)$, υπολογίζουμε διαδοχικά την (2.2.3) για κάθε $e_i=(\upsilon_i,\upsilon_{i+1})$ από $i=1$ μέχρι n-1. Για δοθέν i , ελαχιστοποιούμε την ποσότητα $d(x,u)+c(u,\upsilon)+d(\upsilon,y)$ για όλα τα $(u,\upsilon)\in E_i\setminus e_i$. Η διαφορά μεταξύ E_i και E_{i+1} είναι εύκολο να υπολογιστεί. Αποτελείται από εκείνες τις ακμές που έχουν άκρο τον υ_{i+1}. Για να παράγουμε το E_{i+1} από το E_i , προσθέτουμε στο E_i τις ακμές που έχουν ως αριστερό άκρο τον υ_{i+1} και απομακρύνουμε από το E_i εκείνες που έχουν δεξιό άκρο τον υ_{i+1}. Με left(e) και right(e) θα καλούμε τους δείκτες των άκρων της e στο path(x,y), με $left(e)<right(e)$.

Με βάση τα παραπάνω έχουμε τον ακόλουθο αλγόριθμο:

Path Algorithm

1. Έστω Lκαι R πίνακες με k στοιχεία (τα στοιχεία τους είναι σύνολα ακμών), αρχικά κενοί. Έστω Q ουρά προτεραιότητας (priority queue) με στοιχεία ζεύγη (βάρη , ακμές) που εισάγονται με βάση το βάρος. Η Q είναι αρχικά κενή.

2. Για κάθε $e\in E\setminus path(x,y)$
   εάν $left(e)<right(e)$ εισήγαγε την e στους $L[left(e)]$ και $R[right(e)]$.

3. Από $i=1$ μέχρι k-1

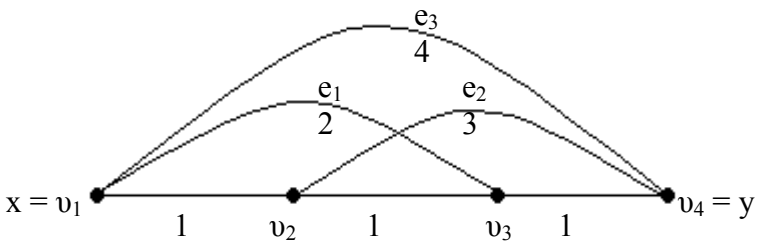
   a) Για κάθε $e=(u,\upsilon)\in L[i]$
   Εισήγαγε (w,e) στην Q , με $w=d(x,u;G\setminus e_i)+c(u,\upsilon)+d(\upsilon,y;G\setminus e_i)$
   b) Απομάκρυνε από το Q όλα τα ζεύγη (w,e) με $e\in R[i]$.
   c) Ενημέρωσε το ελάχιστο βάρος στην Q ως την απόσταση x-y παραλειπόμενης της e_i.

Ανάλυση: Ουσιαστικά ο αλγόριθμος δοκιμάζει για κάθε ακμή του shortest path που αφαιρεί όλες τις ακμές του αντίστοιχου cut, εφαρμόζει δηλαδή την

ελαχιστοποίηση της σχέσης 2.2.3. Οι δείκτες left(e) και right(e) απλά τον βοηθούν να βρίσκει αποδοτικά τις ακμές των διαδοχικών cut. Η ουρά προτεραιότητας είναι μια δομή δεδομένων που του επιτρέπει να εντοπίζει για κάθε ακμή e_i που αφαιρούμε το shortest path στο γράφο που έμεινε όταν αφαιρέσαμε την ακμή αυτή.

Στο βήμα-i η Q περιέχει τις ακμές του $E_i \setminus e_i$, έτσι η απόσταση x-y παραλειπόμενης της e_i είναι σωστά υπολογισμένη. Στον αλγόριθμο μόνο η ουρά προτεραιότητας δεν έχει σταθερό χρόνο. Μία απλή εφαρμογή της ουράς έχει χρόνο $O(m \log m)$. Συνεπώς ο συνολικός χρόνος για τον υπολογισμό των πληρωμών όλων των παιχτών θα είναι ασυμπτωτικά ίσος με τον χρόνο υπολογισμού του shortest path στο γράφο.

Παράδειγμα: Θα εφαρμόσουμε τον Path Algorithm στον επόμενο γράφο



Οι ακμές που ανήκουν στο $E \setminus path(x, y)$ είναι οι e_1, e_2, e_3 με βάρη 2, 3, 4 αντίστοιχα. Ο αλγόριθμος στο βήμα 2 υπολογίζει τα στοιχεία των L και R.
Έχουμε:

$e_1 = (\upsilon_1, \upsilon_3)$ άρα $left(e_1) = 1$ και $right(e_1) = 3$.

$e_2 = (\upsilon_2, \upsilon_4)$ άρα $left(e_2) = 2$ και $right(e_2) = 4$.

$e_3 = (\upsilon_1, \upsilon_4)$ άρα $left(e_3) = 1$ και $right(e_3) = 4$.

Έτσι έχουμε: $L = [\{e_1, e_3\}, \{e_2\}, \varnothing, \varnothing]$ και $R = [\varnothing, \varnothing, \{e_1\}, \{e_2, e_3\}]$.

Μετά εκτελεί το βήμα 3 για i = 1,2 .

Για $i = 1$ εξετάζει τις ακμές που ανήκουν στο L[1].

Ξεκινά με την $e_1 = (\upsilon_1, \upsilon_3)$ και υπολογίζει το w. Για την ακμή αυτή

$$w = d(x, \upsilon_1; G \setminus (\upsilon_1, \upsilon_2)) + c(\upsilon_1, \upsilon_3) + d(\upsilon_3, y; G \setminus (\upsilon_1, \upsilon_2)) = 0 + 2 + 1 = 3$$

Ο πρώτος όρος είναι ίσος με 0 διότι τα x και υ_1 ταυτίζονται. Ο δεύτερος όρος είναι το βάρος της e_1 που είναι ίσο με 2. Ο τρίτος όρος είναι το βάρος της (υ_3, y) δηλαδή 1. Άρα για την e_1 το w είναι 3. Εισάγει λοιπόν στη Q (που ήταν κενή) το ζεύγος $(3, e_1)$.
Επαναλαμβάνει την ίδια διαδικασία για την e_3. Προφανώς το w ισούται με το

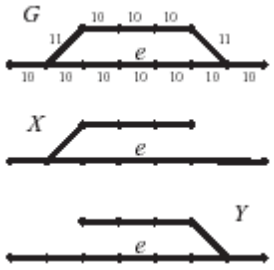
βάρος της e_3 αφού συνδέει απευθείας τους x και y. Άρα εισάγει στην Q το ζεύγος $(4, e_3)$.
Μετά αφαιρεί από την Q όλα τα ζεύγη (w, e) με $e \in R[1]$. Αλλά $R[1] = \varnothing$. Για $i = 1$ το ελάχιστο βάρος στην Q είναι 3. Αυτή είναι η απόσταση x-y παραλειπόμενης της (υ_1, υ_2).

Μετά εργάζεται ομοίως.

2.2.4 Μη Κατευθυνόμενα Γραφήματα

Γενικά σε ένα μη κατευθυνόμενο γράφημα δεν ισχύει πάντα ότι όλοι οι κόμβοι ανήκουν στο $path(x, y)$. Αυτό έχει ως αποτέλεσμα η δομή των shortest paths από το x προς τους άλλους κόμβους να μην έχει σχέση με την δομή των shortest paths από τους κόμβους προς το y. Το shortest path tree με πηγή (source) το x είναι η ένωση όλων των shortest paths από το x προς τους άλλους κόμβους του V. Από την υπόθεση της μοναδικότητας των shortest paths προκύπτει ότι η ένωση των shortest paths είναι δέντρο. Κάθε κόμβος υ έχει ένα μοναδικό πατέρα u στο δέντρο. Το $path(x, \upsilon)$ προκύπτει από την αλληλουχία των $path(x, u)$ με την ακμή (u, υ). Θα καλούμε με X το shortest path tree με πηγή το x. Ομοίως με Y το shortest path tree με απόληξη (sink) το y.

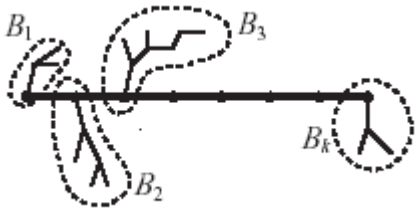
Επειδή έχουμε υποθέσει ότι το G είναι μη κατευθυνόμενο, το Y θα είναι όμοιο με το shortest path tree με πηγή το y (σε κατευθυνόμενα γραφήματα αυτό δεν ισχύει). Συνεπώς το Y θα είναι η ένωση όλων των shortest paths από τους κόμβους του V με κατάληξη το y. Τα shortest path trees X και Y μπορούν να υπολογιστούν με τον αλγόριθμο του Dijkstra σε χρόνο $O(n \log n + m)$ χρησιμοποιώντας πολύπλοκη δομή δεδομένων ή σε χρόνο $O(m \log n)$ χρησιμοποιώντας μια απλή δομή δεδομένων .



Εικόνα 2.2.3 : Τα X και Y έχουν διαφορετική δομή

Για την περίπτωση των path graphs, όπως είδαμε στην προηγούμενη ενότητα, έχουμε $X = Y = path(x, y)$, έτσι η απομάκρυνση μιας ακμής e χωρίζει τα X και Y σε όμοια μέρη. Γενικά για μη κατευθυνόμενα γραφήματα αυτό δεν ισχύει. Ένα παράδειγμα φαίνεται στην εικόνα 2.2.3. Έχουμε το γράφημα G και τα X και Y, παρατηρούμε ότι τα X και Y είναι διαφορετικά. Όταν αφαιρέσουμε την e, τότε

οι κόμβοι στον πάνω κλάδο του γράφου πρόσκεινται στα X \ e και Y \ e που είναι διαφορετικά. Για να υπολογίσουμε την απόσταση x-y όταν αφαιρέσουμε την ακμή $e_i = (\upsilon_i, \upsilon_{i+1})$ από το $\text{path}(x, y) = (\upsilon_1, \ldots, \upsilon_k)$, ορίζουμε το $\text{cut}(V_x, V_y)$ που βασίζεται στο shortest path tree X. Επειδή $\text{path}(x, y)$ περιέχεται στο X η απομάκρυνση της e_i χωρίζει το X σε δύο μέρη, επιλέγουμε το V_x βάση της συνιστώσας που περιέχει το x. Το συμπλήρωμα του V_x είναι το V_y. Η κατασκευή των V_x και V_y γίνεται ως εξής: Επιτρέπουμε σε κόμβους του V να δημιουργούν blocks με βάση τη θέση τους στο shortest path tree X. Αν απαλείψουμε όλες τις ακμές του path (x,y) από το X οι κόμβοι που συνδέονται με το υ_i στο εναπομείναν δάσος, σχηματίζουν το block B_i. Εάν $u \in B_i$ τότε ορίζουμε $\text{block}(u) = i$. Συνεπώς για δοθείσα ακμή $e_i = (\upsilon_i, \upsilon_{i+1}) \in \text{path}(x, y)$ έχουμε $V_x = \bigcup_{j=1}^{i} B_j$ και $V_y = \bigcup_{j=i+1}^{k} B_j$ (Εικόνα 2.2.4).



Εικόνα 2.2.4: Απομακρύνοντας το path(x, y) από το X ορίζονται τα blocks B_i

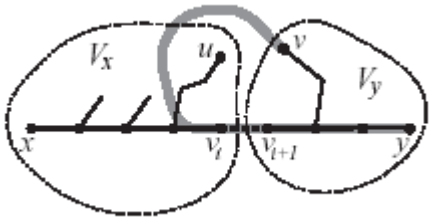
Έστω τώρα ότι θέλουμε να υπολογίσουμε την απόσταση x-y παραλειπόμενης της e_i σύμφωνα με την σχέση:

$$d(x, y; G \setminus e_i) = \min_{\substack{(u,\upsilon) \in E(V_x, V_y) \\ (u,\upsilon) \neq e_i}} (d(x, u; G \setminus e_i) + c(u, \upsilon) + d(\upsilon, y; G \setminus e_i)).$$

Από τον ορισμό, το path(x,u) περιέχεται στο V_x για όλα τα $u \in V_x$, έτσι ισχύει $d(x, u) = d(x, u; G \setminus e_i)$. Η αφαίρεση της ακμής e_i επιφέρει μια διαμέριση στο X που όμως είναι διαφορετική από την αντίστοιχη διαμέριση του Y (εικόνα 2.2.3), έτσι δεν είναι προφανές ότι το path(υ,y) περιλαμβάνεται στο V_y για όλα τα $\upsilon \in V_y$. Ωστόσο αποδεικνύεται ότι το path(υ,y) δεν χρησιμοποιεί την e_i.

Λήμμα 2.2.1 Έστω υ κόμβος του $V_y = \bigcup_{j=i+1}^{k} B_j$ για κάποια $e_i = (\upsilon_i, \upsilon_{i+1})$. Τότε ισχύει $d(\upsilon, y) = d(\upsilon, y; G \setminus e_i)$.

Απόδειξη: Έστω ότι το path(υ,y) χρησιμοποιεί την ακμή e_i. Τότε θα πρέπει να διασχίζει την e_i με κατεύθυνση από το υ_i στο υ_{i+1}, επειδή το shortest path από το υ_{i+1} στο y περιέχεται όλο στο V_y και δεν διέρχεται από την e_i. Τότε το path(υ,y) είναι η αλληλουχία του path(υ,υ_{i+1}) με το path(υ_{i+1},y). Το πρώτο υπομονοπάτι περιέχει τον κόμβο υ_i (που είναι κόμβος του V_x) (εικόνα 2.2.5).



Εικόνα 2.2.5: Αντίθετα με το γραμμοσκιαζμένο μονοπάτι, το path(υ,y) δεν περιλαμβάνει την e_i επειδή το path $(\upsilon_{i+1},\upsilon)$ περιλαμβάνεται στο V_y.

Από το shortest path tree X βλέπουμε ότι το $\text{path}(\upsilon_{i+1},\upsilon)$ περιέχεται πλήρως στο V_y επειδή $\upsilon \in V_y$. Έτσι καθώς το G είναι μη κατευθυνόμενο το $\text{path}(\upsilon,\upsilon_{i+1})$ είναι το αντιστροφό του $\text{path}(\upsilon_{i+1},\upsilon)$. Αλλά το ένα περιέχει το υ_i και το άλλο όχι άρα το $\text{path}(\upsilon,y)$ δεν περιέχει την e_i. ■

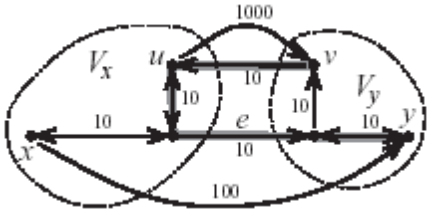
Έχουμε σχεδόν ανάγει την γενική περίπτωση των μη κατευθυνόμενων γράφων σε εκείνη όπου όλοι οι κόμβοι πρόσκεινται στο $\text{path}(x,y)$. Για κάποια ακμή $e=(u,\upsilon)\notin \text{path}(x,y)$ ορίζουμε $\text{left}(e)=\text{block}(u)$ και $\text{right}(e)=\text{block}(\upsilon)$, υποθέτοντας ότι $\text{block}(u)\le \text{block}(\upsilon)$. Αυτό μας εξασφαλίζει ότι $e\in E_i$ ανν $\text{left}(e)\le i<\text{right}(e)$ και έτσι μπορούμε να εφαρμόσουμε τον Path Algorithm. Για κάθε ακμή $(u,\upsilon)\in E_i$, η απόσταση $d(x,u)$ υπολογίζεται χρησιμοποιώντας το shortest path tree X και η απόσταση $d(\upsilon,y)$ υπολογίζεται χρησιμοποιώντας το Y.

Συνεπώς και για την γενική περίπτωση των μη κατευθυνόμενων γράφων ο συνολικός χρόνος για τον υπολογισμό των πληρωμών όλων των παιχτών θα είναι ασυμπτωτικά ίσος με τον χρόνο υπολογισμού του shortest path στο γράφο.

2.2.5 Κατευθυνόμενα Δίκτυα

Όταν το γράφημα G είναι κατευθυνόμενο το πρόβλημα είναι πιο περίπλοκο. Για παράδειγμα το shortest path με πηγή το s είναι διαφορετικό από το shortest path με απόληξη το s. Για να πάρουμε το shortest path tree Y με απόληξη y, πρέπει να αντιστρέψουμε τον προσανατολισμό κάθε ακμής του E και να υπολογίζουμε το shortest path tree με πηγή y στο τροποποιημένο δέντρο. Αν το G δεν έχει αρνητικά κόστη μπορούμε να υπολογίσουμε το shortest path tree σε χρόνο $O(n\log n+m)$ με τον αλγόριθμο του Dijkstra όπως και στα μη κατευθυνόμενα γραφήματα.. Αν το γράφημα έχει και αρνητικά κόστη αλλά δεν έχει κύκλους αρνητικού κόστους, τότε χρησιμοποιούμε τον αλγόριθμο Bellman-Ford που υπολογίζει το shortest path tree σε χρόνο $O(nm)$.

Ακόμα και αν έχουμε υπολογίσει τα shortest paths trees X και Y, ο υπολογισμός της απόστασης x-y παραλειπόμενης διαδοχικά κάθε ακμής του $\text{path}(x,y)$, είναι πιο πολύπλοκος από την περίπτωση των μη κατευθυνόμενων γραφημάτων.

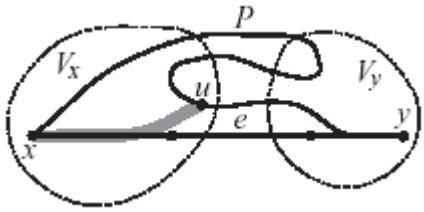


Εικόνα 2.2.6 : Το Λήμμα 2.2.1 δεν αληθεύει για κατευθυνόμενουςγράφους.

Το κύριο πρόβλημα είναι ότι το λήμμα 2.2.1 δεν ισχύει για κατευθυνόμενους γράφους. Ένα παράδειγμα δίνεται στην εικόνα 2.2.6, όπου ο κόμβος υ ανήκει στο τμήμα του $X \setminus e$ που περιέχει το y, αλλά το shortest path $path(\upsilon, y)$ περιέχει την ακμή e. Παρόλα αυτά μπορούμε να ξεπεράσουμε αυτή την δυσκολία. Από το ακόλουθο λήμμα προκύπτει το συμπέρασμα ότι δεν χρειάζεται στην σχέση 2.2.2 να ελαχιστοποιήσουμε πάνω σε όλες τις ακμές του $E(V_x, V_y)$.

Λήμμα 2.2.2 Έστω V_x και V_y οι συνιστώσες του X που προκύπτουν όταν απομακρύνουμε την ακμή $e \in path(x, y)$. Τότε το $path(x, y; G \setminus e)$ περιέχει ακριβώς μία ακμή του $E(V_x, V_y)$.

Απόδειξη: Έστω τυχαίο μονοπάτι P στο G \ e που συνδέει τα x και y και u ο τελευταίος κόμβος του P στο V_x (το μονοπάτι P μπορεί να διέρχεται από το V_x στο V_y πολλές φορές , εμείς επιλέγουμε την τελευταία διάσχιση (εικόνα 2.2.7).



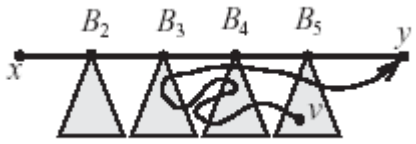
Εικόνα 2.2.7: Η u είναι ο τελευταίος κόμβος του P στο V_X. Το γραμμοσκιαζμένο path(x,υ) περιέχεται στο V_X.

Το shortest path από το x στο u στο G περιέχεται στο V_x, καθώς το V_x είναι υποδέντρο του X περιέχει τα x και u. Έτσι $path(x, u) = path(x, u; G \setminus e)$ και μπορούμε να μικρύνουμε το P αντικαθιστώντας το τμήμα του P μέχρι το u με το $path(x, u)$. Η μόνη ακμή του $E(V_x, V_y)$ σε αυτό το shortest path είναι αυτή που έπεται αμέσως μετά το u. Προκύπτει ότι το shortest path από το x στο y στο G \ e πρέπει να περιέχει ακριβώς μία ακμή του $E(V_x, V_y)$. ■

Πόρισμα αυτού του λήμματος είναι ότι εάν (u,υ) είναι η μοναδική ακμή του $E(V_x, V_y)$ στο $path(x, y; G \setminus e)$, τότε $path(\upsilon, y; G \setminus e) = path(\upsilon, y)$. Έτσι προκύπτει ότι για την ελαχιστοποίηση στη σχέση 2.2.2 δεν χρειάζεται να εξετάσουμε καμία ακμή του $E(V_x, V_y)$ τέτοια που το $path(\upsilon, y)$ περιέχει οποιαδήποτε κόμβο του V_x. Συνεπώς το ελαχιστοποιημένο σύνολο εξισώσεων της 2.2.2 μετασχηματίζεται στο:

$$d(x, y; G \setminus e) = \min_{\substack{(u,\upsilon)\in E(V_x, V_y) \\ (u,\upsilon)\neq e \\ path(\upsilon,y)\cap V_x = \varnothing}} d(x, u; G \setminus e) + c(u, \upsilon) + d(\upsilon, y; G \setminus e) \quad (2.2.4)$$

Για να απορρίψουμε τις ακμές που παραβιάζουν την συνθήκη στο $path(\upsilon, y)$, βάζουμε ετικέτες σε κάθε κόμβο $\upsilon \in V$ σύμφωνα με το χαμηλότερο δεικτοδοτημένο block του X από το οποίο περνάει το $path(\upsilon, y)$. Ορίζουμε ως minblock(υ) το μικρότερο i για το οποίο το path(υ,y) περιέχει ένα κόμβο του block B_i. Έτσι $\min block(\upsilon) = \min_{w\in path(\upsilon,y)} block(w)$ (εικόνα 2.2.8).



Εικόνα 2.2.8 : block(υ)=5 , αλλά minblock(υ)=3.

Μπορούμε να υπολογίσουμε το minblock(u) για όλους τους κόμβους υ σε χρόνο O(n) με μία preorder διάσχιση του Y ξεκινώντας από το y. Για κάθε ακμή (u,υ) το minblock(u) είναι min(block(u),minblock(υ)) και η preorder διάσχιση επισκέπτεται τον υ πριν από τον u.

Για μια κατευθυνόμενη ακμή $e = (u, \upsilon) \notin path(x, y)$, ορίζουμε $left(e) = block(u)$ και $right(e) = minblock(\upsilon)$. Αυτοί οι ορισμοί των left(e) και right(e) μας εγγυώνται ότι η e ανήκει στο ελαχιστοποιημένο σύνολο της σχέσης 2.2.4 για e_i ανν $left(e) \le i \le right(e)$ και έτσι μπορούμε να εφαρμόσουμε τον Path Algorithm. Οι αποστάσεις $d(x, y)$ και $d(\upsilon, y)$ υπολογίζονται από τα shortest path trees X και Y.

Από τα παραπάνω προκύπτει το ακόλουθο θεώρημα

Θεώρημα 2.2.3 Δοθέντος ενός κατευθυνόμενου γράφου G με m ακμές και ενός ζεύγους κόμβων (x,y), μπορούμε να υπολογίσουμε την απόσταση $d(x, y; G \setminus e)$ για κάθε ακμή $e \in path(x, y)$ σε συνολικό χρόνο O(m) συν το χρόνο υπολογισμού του shortest path tree στο G .

Αυτό το θεώρημα μας επιτρέπει να υπολογίσουμε τις VCG πληρωμές για όλους τους παίχτες που βρίσκονται στις ακμές του shortest path ενός δικτύου, σε

χρόνο ασυπτωτικά ίσο με το χρόνο υπολογισμού του shortest path του δικτύου.

2.3 Συμπεράσματα, μελλοντικές κατευθύνσεις

Στο κεφάλαιο αυτό μελετήσαμε τον σχεδιασμό μηχανισμών για δύο αλγοριθμικά προβλήματα το task scheduling και το shortest path.

Στην ενότητα 2.1 μελετήσαμε το task scheduling πρόβλημα με δύο μοντέλα: με το βασικό μοντέλο μηχανισμών που ορίσαμε στο κεφάλαιο 1 και με το μοντέλο των μηχανισμών με επαλήθευση. Με το πρώτο μοντέλο δείξαμε ότι ο καλύτερος προσεγγιστικός μηχανισμός για το task scheduling θα είναι 2-προσεγγιστικός. Επίσης χρησιμοποιώντας πιθανοτικούς μηχανισμούς αποδείξαμε ότι μπορούμε να επιτύχουμε προσέγγιση καλύτερη από 2. Συγκεκριμένα δείξαμε ότι ο πιθανοτικός diased min work μηχανισμός είναι $7/4$-προσεγγιστικός.

Με το μοντέλο των μηχανισμών με επαλήθευση παρουσιάσαμε πολλούς καινούργιους μηχανισμούς, βέλτιστους, περιορισμένους βέλτιστους και πολυωνιμικούς προσεγγιστικούς μηχανισμούς. Μελετήσαμε εκτενώς τον Μηχανισμός Αποζημίωσης και Πριμ καθώς και το περιορισμένο task scheduling πρόβλημα. Ένα από τα βασικά συμπεράσματα στα οποία καταλήξαμε είναι ότι αν αντικαταστήσουμε στο Μηχανισμό Αποζημίωσης και Πριμ τον βέλτιστο αλγόριθμο εξόδου με κάποιο προσεγγιστικό τότε ο μηχανισμός που προκύπτει δεν είναι πλέον φιλαλήθης.

Πολλά είναι τα ερωτήματα που προκύπτουν από την μελέτη αυτή. Ένα θέμα προς μελέτη είναι το μεγάλο διάστημα μεταξύ του πάνω και κάτω φράγματος για το task scheduling χωρίς επαλήθευση. Επίσης θα μπορούσαν να υπάρξουν μοντέλα με διαφορετικές υποθέσεις όπως για παράδειγμα τα επαναλαμβανόμενα παίγνια. Μία άλλη παρατήρηση είναι ότι είδαμε τον μηχανισμό σαν ένα μαύρο κουτί και δεν ασχοληθήκαμε με τον κατανεμημένο τρόπο με τον οποίο εκτελείται η λειτουργία του. Ένα σύνολο προβλημάτων προκύπτει όταν προσπαθήσουμε να αναλύσουμε τα βήματα με τα οποία, με κατανεμημένο τρόπο, υλοποιείται ο μηχανισμός.

Στην ενότητα 2.2 επικεντρωθήκαμε στο αλγοριθμικό μέρος του υπολογισμού των VCG πληρωμών για το πρόβλημα του shortest path routing σε ένα δίκτυο επικοινωνίας (π.χ. το internet), όπου ένας σύνολο ιδιοτελών παιχτών κατέχει ένα μέρος του δικτύου. Μία απλοϊκή συνάρτηση πληρωμής για n παίχτες είδαμε ότι απαιτεί τον υπολογισμό $n+1$ shortest paths. Η μέθοδος που παρουσιάσαμε είναι σχετικά απλή και επιτυγχάνει τον υπολογισμό των πληρωμών σε χρόνο ασυμπτωτικά ίσο με τον χρόνο υπολογισμού ενός shortest path.

Ακόμα και αν για κάποιο πρόβλημα ο υπολογισμός των πληρωμών απαιτεί πολυωνιμικό χρόνο, όπως στο πρόβλημα που εξετάσαμε, είναι πολύ σημαντικό να μπορούμε να βελτιώσουμε τον υπολογισμό $n+1$ προβλημάτων βελτιστοποίησης σε χρόνο πολύ μικρότερου αριθμού προβλημάτων. Για παράδειγμα ας θεωρήσουμε το πρόβλημα του πλειστηριασμού εργασιών, όπου n-ιδιοτελείς παίχτες πλειοδοτούν για ένα σύνολο n εργασιών, με $c(i,j)$ να είναι η προσφορά του παίχτη i για την εκτέλεση της εργασίας j. Η βέλτιστη ανάθεση εργασιών απαιτεί τον υπολογισμό του min-cost bipartite matching. Καθώς οι παίχτες είναι

ιδιοτελείς θα πρέπει να χρησιμοποιήσουμε ένα VCG μηχανισμό για να έχουμε ειλικρινείς προσφορές. Για να υπολογίσουμε τις VCG πληρωμές για όλα τα ζεύγη παίχτης-εργασία πρέπει να λύσουμε n διαφορετικά min-cost bipartite matching προβλήματα. Έτσι προκύπτει το ερώτημα του κατά πόσο μπορούμε να βελτιώσουμε τους παραπάνω υπολογισμούς και να επιτύχουμε αποτελέσματα όμοια με το πρόβλημα της δρομολόγησης ελάχιστου μονοπατιού.

Κεφάλαιο 3
Υπολογιστικά Εφικτοί Μηχανισμοί

Κεφάλαιο 3
Υπολογιστικά Εφικτοί Μηχανισμοί

3.1 Υπολογιστικά Εφικτοί Μηχανισμοί

Ένα από τα σημαντικότερα αποτελέσματα στη περιοχή του mechanism design είναι η οικογένεια των VCG μηχανισμών τους οποίους ορίσαμε στο Κεφ. 1.

Η σπουδαιότητα των VCG μηχανισμών έγκειται στο ότι είναι φιλαλήθεις. Οι φιλαλήθεις μηχανισμοί ονομάζονται και incentive compatible ή strategy proof. Η φιλαλήθεια είναι απαραίτητο χαρακτηριστικό σε κάθε μηχανισμό που σχεδιάζουμε για να υλοποιεί κάποιο πρόβλημα. Η φιλαλήθεια μας εγγυάται ότι οι πληροφορίες που δίνουν οι παίχτες στο μηχανισμό είναι αληθείς. Κατά το σχεδιασμό όμως ενός μηχανισμού εκτός από τη φιλαλήθεια μας ενδιαφέρουν και οι υπολογιστικές απαιτήσεις του μηχανισμού. Ένας φιλαλήθης μηχανισμός που χρειάζεται εκθετικό χρόνο για να υπολογίσει την έξοδο που ζητούμε θα είναι στην πράξη ανεφάρμοστος.

Όταν υλοποιούμε κάποιο υπολογιστικά δύσκολο πρόβλημα με VCG μηχανισμούς τότε ο υπολογισμός της βέλτιστης εξόδου θα είναι υπολογιστικά δύσκολος. Αυτό οφείλεται στο ότι οποιοσδήποτε μηχανισμός υλοποιεί το πρόβλημα έχει ένα αλγόριθμο εξόδου που είναι ουσιαστικά αλγόριθμος του αντίστοιχου off-line προβλήματος. Αν απαιτούμε ο μηχανισμός να μας δίνει την βέλτιστη λύση τότε ο αλγόριθμος εξόδου θα πρέπει να είναι κάποιος βέλτιστος αλγόριθμος του προβλήματος. Συνεπώς αν το πρόβλημα είναι NP-complete τότε οποιοσδήποτε βέλτιστος αλγόριθμος εξόδου θα είναι εκθετικού χρόνου. Για τα προβλήματα αυτά δεν είναι γνωστό αν υπάρχουν αλγόριθμοι πολυωνιμικού χρόνου.

Μία λύση είναι να αντικαταστήσουμε το βέλτιστο αλγόριθμο του μηχανισμού με κάποιο προσεγγιστικό. Όπως είδαμε όμως και στο κεφ. 2 για το task scheduling, ο μηχανισμός που προκύπτει μπορεί πλέον να μην είναι φιλαλήθης. Στην ενότητα αυτή θα μελετήσουμε ακριβώς αυτό το πρόβλημα. Θα εξετάσουμε την περίπτωση των συνδυαστικών πλειστηριασμών και των προβλημάτων ελαχιστοποίησης κόστους. Θα δείξουμε ότι χρησιμοποιώντας προσεγγιστικούς αλγορίθμους εξόδου αντί για βέλτιστους προκύπτουν μηχανισμοί που δεν είναι φιλαλήθεις και θα εξετάσουμε τρόπους με τους οποίους μπορούμε να αντιμετωπίσουμε αυτό το πρόβλημα.

Τα αποτελέσματα και η ανάλυση που παρουσιάζονται στην ενότητα προέρχονται από το άρθρο [NR00] των Noam Nisan και Amir Ronen.

3.1.1 Εισαγωγή

Ας θυμηθούμε μερικές βασικές έννοιες.

Ορισμός 3.1.1 (Ωφελιμιστικό πρόβλημα σχεδίου μηχανισμού)
Σε ένα πρόβλημα σχεδίου μηχανισμού έχουμε:

1. Το πεπερασμένο σύνολο O των επιτρεπτών εξόδων του μηχανισμού.
2. Κάθε παίχτης i, έχει μία πραγματική συνάρτηση $v^i(o)$, $o \in O$, που καλείται αποτίμηση ή τύπος και την γνωρίζει μόνο αυτός. Ο χώρος V^i όλων των πιθανών συναρτήσεων αποτίμησης, καλείται χώρος των τύπων του παίχτη.
3. Εάν η έξοδος του μηχανισμού είναι o και η πληρωμή του παίχτη είναι p^i, τότε η utility του παίχτη είναι: $u^i = v^i(o) + p^i$. Αυτή την ποσότητα επιθυμεί να βελτιστοποιήσει ο παίχτης.
4. Σκοπός του μηχανισμού είναι να επιλέξει εκείνη την έξοδο $o \in O$ που μεγιστοποιεί το συνολικό όφελος, $g(v,o) = \sum_i v^i(o)$.

Τα προβλήματα αυτά καλούνται ωφελιμιστικά διότι και ο μηχανισμός και οι παίχτες επιθυμούν να μεγιστοποιήσουν το όφελος τους.

Ένας μηχανισμός είναι το ζεύγος (k,p), όπου k είναι η συνάρτηση εξόδου και p η συνάρτηση πληρωμής. Η k δέχεται ως είσοδο ένα διάνυσμα $w = (w^1,\ldots,w^n)$ που είναι οι αποτιμήσεις που δήλωσαν οι παίχτες και δίνει μία έξοδο $k(w) \in O$.

Η συνάρτηση εξόδου $p(w) = (p^1(w),\ldots,p^n(w))$ δίνει ένα πραγματικό διάνυσμα που αποτελείται από τις πληρωμές των παιχτών.

Ορισμός 3.1.2 (VCG–μηχανισμός) Ένας μηχανισμός $m = (k,p)$ ανήκει στην οικογένεια των VCG μηχανισμών αν:

- Η συνάρτηση εξόδου k(w), μεγιστοποιεί το συνολικό όφελος με βάση το διάνυσμα αποτιμήσεων w (που δήλωσαν οι παίχτες).
- Η συνάρτηση πληρωμής υπολογίζεται σύμφωνα με την σχέση: $p^i(w) = \sum_{j \neq i} w^j(k(w)) + h^i(w^{-i})$, όπου $h^i(\ )$ είναι αυθαίρετη συνάρτηση των w^{-i}.

Όπως έχουμε ήδη αναφέρει στο κεφ.1 κάθε VCG μηχανισμός είναι φιλαλήθης.

Σε πάρα πολλές εφαρμογές, η εύρεση της εξόδου k(w) που μεγιστοποιεί το συνολικό όφελος είναι υπολογιστικά πολύ δύσκολη. Στις επόμενες ενότητες θα ασχοληθούμε με μηχανισμούς στους οποίους ο βέλτιστος αλγόριθμος έχει αντικατασταθεί από αλγόριθμο που δεν είναι βέλτιστος αλλά είναι υπολογιστικά εφικτός.

Ορισμός 3.1.3 (VCG-Based Μηχανισμός) Έστω k(w) αλγόριθμος που αντιστοιχεί τις δηλώσεις των τύπων με τις αποδεκτές εξόδους. Θα καλούμε τον μηχανισμό $m = (k(w), p(w))$ ***VCG μηχανισμό που βασίζεται στον k***, αν η p

υπολογίζεται σύμφωνα με τον VCG τύπο: $p^i(w) = \sum_{j \neq i} w^j(k(w)) + h^{-i}\left(w^{-i}\right)$, όπου $h^i(\ )$ είναι αυθαίρετη συνάρτηση των w^{-i}.

Δηλαδή απλά αντικαταστήσαμε τον βέλτιστο αλγόριθμο εξόδου του VCG μηχανισμού με τον αλγόριθμο k.

Στην επιστήμη των υπολογιστών γίνεται μία βασική διάκριση των προβλημάτων ανάλογα την υπολογιστική τους δυσκολία σε δύο κύριες κλάσεις την κλάση P και την κλάση NP. Η κλάση P περιλαμβάνει τα προβλήματα που μπορούν να λυθούν σε πολυωνιμικό χρόνο σε σχέση με το μέγεθος της εισόδου τους. Η κλάση των NP-complete περιέχει προβλήματα που με τα μέχρι τώρα δεδομένα είναι υπολογιστικά δύσκολα. Μέχρι τώρα δεν έχει βρεθεί πολυωνιμικός αλγόριθμος για τα NP-complete προβλήματα. Δυστυχώς τα περισσότερα προβλήματα που παρουσιάζουν ενδιαφέρον είναι NP-complete. Όμως για τα περισσότερα από αυτά είναι διαθέσιμοι προσεγγιστικοί αλγόριθμοι πολυωνιμικού χρόνου.

Όσον αφορά τους μηχανισμούς έχουμε τους ακόλουθους ορισμούς.

Ορισμός 3.1.4 Ένας μηχανισμός (k,p) θα καλείται *πολυωνιμικού χρόνου* αν οι k(w) και p(w) υπολογίζονται σε πολυωνιμικό χρόνο.

Σημειώνουμε ότι ένας VCG-Based Μηχανισμός είναι πολυωνιμικός αν ο αλγόριθμος εξόδου είναι πολυωνιμικός. Θα καλούμε τους πολυωνιμικούς αλγορίθμους και μηχανισμούς ως ***υπολογιστικά εφικτούς***.

Ορισμός 3.1.5 Ένα πρόβλημα σχεδίου μηχανισμού θα καλείται NP-complete αν ο υπολογισμός της εξόδου που μεγιστοποιεί το συνολικό όφελος είναι πρόβλημα NP-complete.

3.1.2 Συνδυαστικοί Πλειστηριασμοί

Συνδυαστικοί Πλειστηριασμοί
Το πρόβλημα: Ένας πωλητής επιθυμεί να πουλήσει ένα σύνολο αντικειμένων S σε ένα σύνολο παιχτών. Κάθε παίχτης i για κάθε υποσύνολο $s \subseteq S$ των αντικειμένων έχει ένα αριθμό $v^i(s)$ που αντιπροσωπεύει πόσο αξίζει το s γι' αυτόν. Τον αριθμό $v^i(s)$ τον γνωρίζει μόνο ο παίχτης i.

Κάνουμε τις εξής υποθέσεις για τους παίχτες:
Υποθέσεις:

1. Η αποτίμηση (valuation) κάθε παίχτη εξαρτάται μόνο από τα αντικείμενα που του έχουν ανατεθεί. Έτσι το $\{v^i(s) \mid s \subseteq S\}$ είναι η αποτίμηση του παίχτη i.
2. **(Ελεύθερη διάθεση)** Τα αντικείμενα δεν έχουν αρνητικές αποτιμήσεις.

Έτσι αν $s \subseteq t$ τότε $\upsilon^i(s) \le \upsilon^i(t)$, επίσης $\upsilon^i(\varnothing) = 0$. Δηλαδή τα αντικείμενα ή τα θέλουν οι παίχτες ή όχι. Δεν μπορούμε να αναγκάσουμε τους παίχτες να πάρουν αντικείμενα που δεν θέλουν, γι' αυτό και δεν έχουν αρνητικές τιμές..

Παρατηρήσεις:

1. Από τον ορισμό του προβλήματος μπορεί να έχουμε:

$\upsilon^i(S \bigcup T) \ge \upsilon^i(S) + \upsilon^i(T)$ ή και $\upsilon^i(S \bigcup T) \le \upsilon^i(S) + \upsilon^i(T)$ (όπου S, T ξένα).

Για παράδειγμα ένας παίχτης μπορεί να είναι πρόθυμος να πληρώσει 300 ευρώ για T.V, 200 ευρώ για VCR αλλά 600 ευρώ και για τα δύο μαζί ή 300 ευρώ για δύο VCR.

2. Επίσης αν για παίχτη i και σύνολο αντικειμένων s^i η πληρωμή είναι p^i τότε η η utility του είναι $p^i + \upsilon^i(s)$. Επισημαίνουμε ότι η πληρωμή δεν είναι θετική. Η utility είναι η ποσότητα που ο παίχτης θέλει να βελτιστοποιήσει και εκφράζει το όφελος του παίχτη από την αγορά του s. Αυτό προκύπτει από το ότι κάθε παίχτης προσπαθεί να αποκτήσει τα αντικείμενα που επιθυμεί περισσότερο σε όσο το δυνατό μικρότερη τιμή. Για παράδειγμα ένας παίχτης προτιμά να αγοράσει για 600 ευρώ ένα VCR αποτιμημένο για 1000 ευρώ (κερδίζοντας έτσι 400 ευρώ), παρά να αγοράσει για 1250 ευρώ ένα VCR αποτιμημένο για 1500 ευρώ.

Σχόλια:

1. Σε ένα VCG μηχανισμό για τέτοιου είδους πλειστηριασμό οι συμμετέχοντες αρχικά καλούνται να αποκαλύψουν στον μηχανισμό τις συναρτήσεις αποτίμησης. Μετά ο μηχανισμός με βάση τις δηλώσεις των παιχτών υπολογίζει μία ανάθεση s που μεγιστοποιεί το συνολικό όφελος. Η πληρωμή για κάθε παίχτη υπολογίζεται από τη συνάρτηση πληρωμών που είναι τύπου VCG. Επειδή κάθε VCG μηχανισμός είναι φιλαλήθης, η utility $u^i = p^i + \upsilon^i(s^i)$ κάθε παίχτη μεγιστοποιείται όταν αποκαλύψει στον μηχανισμό την αληθή αποτίμηση του. Όταν όλοι παίχτες είναι ειλικρινείς, τότε ο μηχανισμός μεγιστοποιεί το συνολικό όφελος.

2. Ας εξετάσουμε τώρα το υπολογιστικό μέρος ενός τέτοιου μηχανισμού. Αφού οι παίχτες αποκαλύψουν τους τύπους τους, ο μηχανισμός πρέπει να επιλέξει από όλες τις πιθανές αναθέσεις εκείνη που μεγιστοποιεί το συνολικό όφελος. Το πρόβλημα όμως των συνδυαστικών πλειστηριασμών είναι NP-complete. Άρα ο υπολογισμός της βέλτιστης εξόδου θα χρειαστεί εκθετικό χρόνο. Κατά συνέπεια ένας τέτοιος μηχανισμός είναι υπολογιστικά ανέφικτος έκτος και αν ο αριθμός των παιχτών είναι μικρός.

Στη συνέχεια θα μελετήσουμε τους περιορισμούς των φιλαληθών VCG-based μηχανισμών. Θα δείξουμε ότι κάθε φιλαλήθης VCG-based μηχανισμός για συνδυαστικούς πλειστηριασμούς που δεν είναι βέλτιστος, παρουσιάζει ανώμαλη συμπεριφορά.

Ορισμός 3.1.6 Έστω αλγόριθμός k(w) που αντιστοιχεί τις δηλώσεις των τύπων σε επιτρεπόμενες εξόδους. Έστω υπόχωρος V΄ του χώρου όλων των δυνατών

τύπων $V \stackrel{df}{=} \prod_{i=1}^{n} V^i$. Με **O** θα δηλώνουμε το πεδίο τιμών του k στο V΄, $\mathbf{O} = \{k(w) \mid w \in V'\}$. Θα λέμε ότι ο *k είναι maximal στο πεδίο τιμών του στο V΄*, αν για κάθε τύπο $w \in V'$, το k(w) μεγιστοποιεί το g στο **O**, όπου g είναι το συνολικό όφελος. Θα λέμε ότι ο *k είναι maximal στο πεδίο τιμών του* εάν είναι maximal στο πεδίο τιμών του στο V.

Παρατηρήσεις: Όπως έχουμε ήδη αναφέρει ο αλγόριθμος εξόδου ενός VCG μηχανισμού είναι κάποιος βέλτιστος αλγόριθμος του προβλήματος. Ένας βέλτιστος αλγόριθμος θα δίνει έξοδο που θα μεγιστοποιεί το g, που είναι η αντικειμενική συνάρτηση του προβλήματος, για οποιοδήποτε διάνυσμα τύπων. Στόχος μας είναι να μελετήσουμε τους μηχανισμούς όταν αντικαταστήσουμε τον βέλτιστο αλγόριθμο εξόδου με τον αλγόριθμο k. Με τον παραπάνω ορισμό προσπαθούμε να κατηγοριοποιήσουμε τους αλγόριθμους k. Περιορίζουμε έτσι το πεδίο ορισμού τους στο V΄ και εξετάζουμε αν οι έξοδοι που δίνει ο k για τους τύπους στο V΄ μεγιστοποιούν το g.

Πρόταση 3.1.7 Ένας VCG-based μηχανισμός με αλγόριθμο εξόδου maximal στο πεδίο τιμών του, είναι φιλαλήθης.

Απόδειξη: Ένας τέτοιος μηχανισμός είναι VCG μηχανισμός όπου το σύνολο των αποδεκτών εξόδων είναι το πεδίο τιμών του αλγορίθμου εξόδου. Συνεπώς ο μηχανισμός θα είναι φιλαλήθης. ■

Σημείωση: Θα συμβολίζουμε με $\tilde{V}$ τον χώρο όλων των τύπων $\upsilon = (\upsilon^1, \cdots, \upsilon^n)$ για τους οποίους για οποιεσδήποτε διαφορετικές αναθέσεις x και y να ισχύει $g(\upsilon, x) \neq g(\upsilon, y)$, όπου g( ) είναι το συνολικό όφελος. Είναι εύκολο να δούμε ότι το $\tilde{V}$ περιέχει σχεδόν όλους τους τύπους.

Θεώρημα 3.1.8 Εάν ένας VCG-based μηχανισμός για συνδυαστικούς πλειστηριασμούς είναι φιλαλήθης, τότε ο αλγόριθμος εξόδου είναι maximal στο πεδίο τιμών του στο $\tilde{V}$.

Απόδειξη: Έστω ότι ο $m = (k, p)$ είναι φιλαλήθης, αλλά ο k όμως δεν είναι maximal στο πεδίο τιμών του στο $\tilde{V}$. Χωρίς βλάβη της γενικότητας υποθέτουμε ότι $p^i(w) = \sum_{j \neq i} w^j(k(w))$ (δηλαδή στον VCG τύπο της p^i θέτουμε την $h^i(\,)$ ίση με 0). Η utility του φιλαλήθη παίχτη i, όταν οι δηλώσεις των άλλων παιχτών είναι w^{-i}, είναι ίση με :

$$u^i = \upsilon^i + p^i(w) = \upsilon^i + \sum_{j \neq i} w^j(k(w)) = \sum_i \upsilon^i = g\left(\left(\upsilon^i, w^{-i}\right), k\left(\upsilon^i, w^{-i}\right)\right).$$

Έστω **O** συμβολίζει το πεδίο τιμών του k στο $\tilde{V}$ και έστω $\upsilon \in \tilde{V}$ τύπος για τον οποίο το k(υ) δεν είναι βέλτιστο στο **O**. Έστω $y = opt_{\mathbf{O}}(\upsilon)$. Θεωρούμε ότι η βέλτιστη ανάθεση στο **O,** αναθέτει όλα τα αντικείμενα στους παίχτες, δηλαδή ότι

για την $y = (y^1, \ldots, y^n)$ έχουμε $\bigcup_i y^i = S$. Αυτό είναι εφικτό λόγω της υπόθεσης της ελεύθερης διάθεσης που κάναμε για το πρόβλημα. Τέλος έστω τύπος $w \in \tilde{V}$ για τον οποίο $y = k(w)$.

Ορίζουμε τον τύπο z ως: $z^i(s) = \begin{cases} \upsilon^i(s) & \text{εάν } s \not\supseteq y^i \\ \alpha & \text{εάν } s \supseteq y^i \end{cases}$

όπου το α αντιπροσωπεύει κάποιο αρκετά μεγάλο αριθμό.
Δηλαδή κάθε παίχτης i επιθυμεί πολύ το σύνολο y^i. Υποθέτουμε ότι $z \in \tilde{V}$

Λήμμα 3.1.9 $y = k(z)$

Απόδειξη: Θεωρούμε την ακολουθία διανυσμάτων τύπων:

$$w_0 = (w^1, \ldots, w^n)$$
$$w_1 = (z^1, w^2, \ldots, w^n)$$
$$\vdots$$
$$w_n = (z^1, \ldots, z^n)$$

Υποθέτουμε ότι $w_j \in \tilde{V}$

Ισχυρισμός 1 $k(w_1) = y$

Απόδειξη: Έστω ότι ο ισχυρισμός δεν ισχύει. Από τον ορισμό του $\tilde{V}$ παίρνουμε ότι $g(w_1, k(w_1)) \neq g(w_1, y)$. Θεωρούμε την περίπτωση όπου ο τύπος του παίχτη 1 είναι z^1 και των υπόλοιπων παιχτών $w^2, \ldots, w^n$. Δηλώνοντας w^1, ο παίχτης 1 μπορεί να αναγκάζει τον αλγόριθμο να αποφασίσει στο y. Κατά συνέπεια πρέπει $g(w_1, k(w_1)) > g(w_1, y)$. Συγκεκριμένα αφού το α είναι μεγάλο πρέπει $k^1(w^1) \supseteq y^1$. Έτσι έχουμε $\alpha + \sum_{j=2}^n w^j(k(w_1)) > \alpha + \sum_{j=2}^n w^j(y)$. Από την συνθήκη της ελεύθερης διάθεσης έχουμε $w^1(k^1(w^1)) > w^1(y)$ και έτσι παίρνουμε ότι $w^1(k^1(w^1)) + \sum_{j=2}^n w^j(k(w_1)) > w^1(y) + \sum_{j=2}^n w^j(y)$.
Δηλαδή $g(w_0, k(w^1)) > g(w_0, y)$. Από τα παραπάνω προκύπτει ότι όταν ο τύπος του παίχτη 1 είναι w^1, τότε είναι καλύτερο γι' αυτόν να δηλώσει z^1. Αυτό όμως είναι άτοπο διότι ο μηχανισμός είναι φιλαλήθης. ■

Θεωρούμε τώρα την ακολουθία διανυσμάτων τύπων:

$$\upsilon_0 = \left(\upsilon^1, \ldots, \upsilon^n\right)$$
$$\upsilon_1 = \left(z^1, \upsilon^2, \ldots, \upsilon^n\right)$$
$$\vdots$$
$$\upsilon_n = \left(z^1, \ldots, z^n\right)$$

Ισχυρισμός 2 Για όλα τα υ_j, το y μεγιστοποιεί το g στο **O**.

Απόδειξη: Το y μεγιστοποιεί το g για το υ_0. Βάση της υπόθεσης της ελεύθερης διάθεσης μία βέλτιστη ανάθεση για υ_j κάνει την ανάθεση y^i για κάθε παίχτη $i \le j$. Θα πρέπει όμως να αναθέσει βέλτιστα και τα υπόλοιπα αντικείμενα στους υπόλοιπους παίχτες. Αυτό γίνεται σύμφωνα με την y. ■

Ισχυρισμός 3 $k(\upsilon_{n-1}) = y$

Απόδειξη: Αν δεν ίσχυε τότε όταν ο τύπος του n παίχτη είναι υ^n τότε είναι καλύτερο γι' αυτόν να δηλώσει z^n και να έχει έτσι $g(\upsilon_{n-1}, y)$. Καταλήγουμε όμως σε αντίφαση διότι ο μηχανισμός είναι φιλαλήθης. ■

Τελικά έχουμε ότι $k(\upsilon_0) = y$. Άτοπο. Και το θεώρημα έχει αποδειχθεί. ■

Πόρισμα 3.1.10 Θεωρούμε ένα VCG-based μηχανισμό για συνδυαστικούς πλειστηριασμούς με αλγόριθμο εξόδου k. Εάν ο μηχανισμός είναι φιλαλήθης, τότε υπάρχει αλγόριθμος εξόδου $\tilde{k}$, maximal στο πεδίο τιμών του, έτσι ώστε για κάθε υ, $g(\upsilon, k(\upsilon)) = g\left(\upsilon, \tilde{k}(\upsilon)\right)$.

Στη συνέχεια θα δείξουμε ότι οι φιλαλήθεις VCG-based μηχανισμοί που δεν είναι βέλτιστοι παρουσιάζουν ανώμαλη συμπεριφορά.

Ορισμός 3.1.11 (Λογικός μηχανισμός) Ένας μηχανισμός για συνδυαστικούς πλειστηριασμούς καλείται *λογικός (reasonable)*, εάν κάθε φορά που υπάρχει αντικείμενο j και παίχτης i για τους οποίους:

- Για όλα τα S, εάν $j \notin S$ τότε $\upsilon^i(S \cup \{j\}) > \upsilon^i(S)$.
- Για κάθε παίχτη $l \neq i$, $\upsilon^l(S \cup \{j\}) = \upsilon^i(S)$.

τότε ο μηχανισμός αναθέτει το j στον παίχτη i.

Παρατήρηση: Οι μηχανισμοί αυτοί καλούνται λογικοί διότι όταν κάποιος από τους παίχτες επιθυμεί ένα αντικείμενο περισσότερο από τους άλλους, τότε ο μηχανισμός θα του το δώσει.

Θεώρημα 3.1.12 Κάθε φιλαλήθης VCG-based μηχανισμός για συνδυαστικούς

πλειστηριασμούς που δεν είναι βέλτιστος, δεν είναι λογικός.

Απόδειξη: Από το Θεώρημα 3.1.8, έστω $S = (S^1, \ldots, S^n)$ μία ανάθεση που δεν είναι στο πεδίο τιμών του αλγόριθμου εξόδου. Ορίζουμε ένα τύπο υ ως $\upsilon^i(X) = |X \bigcap S^i|$. Προφανώς κάθε αντικείμενο j το επιθυμεί περισσότερο κάποιος παίχτης. Καθώς η ανάθεση του μηχανισμού δεν είναι η S, τότε θα υπάρχει τουλάχιστον ένα αντικείμενο που δεν θα ανατεθεί στον σωστό παίχτη. ■

Σχόλια: Όπως αναφέραμε προηγουμένως ένας μηχανισμός είναι λογικός όταν δίνει τελικά στους παίχτες τα αντικείμενα που επιθυμούν περισσότερο. Από το προηγούμενο θεώρημα όμως προκύπτει ότι αν ο αλγόριθμος του VCG μηχανισμού δεν είναι βέλτιστος τότε ο μηχανισμός δεν είναι λογικός. Αυτό σημαίνει ότι ο αλγόριθμος θα παρουσιάζει ανώμαλη συμπεριφορά.

3.1.3 Προβλήματα Ελαχιστοποίησης Κόστους

Θα δείξουμε ότι για πολλά προβλήματα ελαχιστοποίησης κόστους (cost minimization problems) κάθε φιλαλήθης VCG-based μηχανισμός είναι είτε βέλτιστος είτε δίνει αποτελέσματα αυθαίρετα μακριά από το βέλτιστο. Ας δούμε ένα παράδειγμα τέτοιου προβλήματος.

Multicast transmissions: Έχουμε ένα δίκτυο επικοινωνίας και ένα σύνολο από χρήστες του δικτύου στους οποίους ανήκουν τα links του δικτύου. Σκοπός μας είναι να στείλουμε από ένα κόμβο του δικτύου ένα μήνυμα σε πολλούς παραλήπτες και αναζητούμε την διαδρομή με το μικρότερο κόστος.

Το δίκτυο μοντελοποιείται από ένα κατευθυνόμενο γράφημα G = (V,E) όπου κάθε ακμή e ανήκει σε ένα παίχτη. Το κόστος t_e της αποστολής κάποιου μηνύματος μέσω της ακμής e είναι γνωστό μόνο στον κάτοχο της. Δοθέντος μιας πηγής (source) $s \in V$ και ενός συνόλου σημείων τερματισμού (terminals) $T \subseteq V$ ο μηχανισμός πρέπει να επιλέξει ένα υποδέντρο με ρίζα το s που να καλύπτει όλα τα σημεία τερματισμού. Το μήνυμα θα μεταδοθεί μέσω αυτού του δέντρου. Θεωρούμε ότι κανένας παίχτης δεν κατέχει ένα cut στο δίκτυο.

Σκοπός του μηχανισμού είναι να επιλέξει εκείνο το δέντρο R που ελαχιστοποιεί το συνολικό κόστος $\sum_{e \in R} t_e$, δηλαδή το δέντρο που ελαχιστοποιεί την αντικειμενική συνάρτηση g του προβλήματος που δίνει το κόστος της μετάδοσης.

Ο στόχος κάθε παίχτη είναι να μεγιστοποιήσει το δικό του κέρδος: $p^i - \sum_{(e \in R \text{ ανήκει στον i})} t_e$. Ο όρος $-\sum_{(e \in R \text{ ανήκει στον i})} t_e$ αντιπροσωπεύει την συνολική αποτίμηση του παίχτη i. Είναι η αρνητική τιμή του αθροίσματος από τα κόστη των ακμών που ανήκουν στον i και βρίσκονται στο επιλεγμένο δέντρο R.

Το πρόβλημα αυτό θα το μελετήσουμε αναλυτικά στην ενότητα 3.2 με διαφορετικό όμως μοντέλο. Τώρα θα γενικεύσουμε το παράδειγμα αυτό.

Ορισμός 3.1.13 (Cost minimization allocation problem)
Ένα *cost minimization allocation problem (CMAP)* (πρόβλημα ελαχιστοποίησης κόστους ανάθεσης) είναι ένα πρόβλημα σχεδίου μηχανισμού που περιγράφεται ως εξής:

1. Ο τύπος κάθε παίχτη i δίνεται από ένα διάνυσμα $\left(\upsilon_1^i,\ldots,\upsilon_{m_i}^i\right)$. Θέτουμε $m=\sum_i m_i$. Στο παράδειγμα των multicast transmissions το υ_j^i αντιστοιχεί στο $-t_e$, δηλαδή η αποτίμηση κάθε παίχτη εξαρτάται από τα κόστη των ακμών του που βρίσκονται στο R. Το m είναι ο αριθμός των ακμών του i στο R.
2. Κάθε έξοδος ορίζεται από ένα διάνυσμα $x=\left(x_1^1,\ldots,x_{m_1}^1,\ldots,x_1^n,\ldots,x_{m_n}^n\right)\in\{0,1\}^m$. Με x^i θα συμβολίζουμε το $\left(x_1^i,\ldots,x_{m_i}^i\right)$. Στο προηγούμενο παράδειγμα ο μηχανισμός που θα το υλοποιεί θα δώσει ως έξοδο ένα δέντρο R. Το δέντρο αυτό θα περιγράφεται από ένα διάνυσμα x με τιμές 0 και 1. Οι τιμές αυτές δηλώνουν το αν η ακμή ενός παίχτη ανήκει ή όχι στο R. Τα $1,\ldots m_i$ αντιπροσωπεύουν τις ακμές που κατέχει ο i. Αν στο x έχουμε π.χ. $x_{m_1}^1=0$ τότε η ακμή m_1 του παίχτη 1 δεν ανήκει στο R.
3. Εάν $\upsilon^i=\left(\upsilon_1^i,\ldots,\upsilon_{m^i}^i\right)$ είναι τύπος του παίχτη i και $w^i\le\upsilon^i$ (η ανισότητα αναφέρεται στα διανύσματα), τότε το διάνυσμα w^i είναι επίσης τύπος. Δηλαδή δεν υπάρχει κάποιο φράγμα για τους τύπους των παιχτών. Συνεπώς στο προηγούμενο παράδειγμα τα κόστη των ακμών των παιχτών μπορεί να έχουν οποιαδήποτε τιμή.
4. ***Ανεξαρτησία***. Κάθε αποτίμηση (valuation) υ^i εξαρτάται μόνο από τα bits x^i του i . Η αποτίμηση κάθε παίχτη εξαρτάται από την έξοδο του μηχανισμού και μόνο από το τμήμα της εξόδου που τον αφορά. Για το προηγούμενο παράδειγμα η αποτίμηση του παίχτη i εξαρτάται από το δέντρο R που δίνει ως έξοδο ο μηχανισμός. Εξαρτάται όμως μόνο από τις ακμές του που ανήκουν δέντρου.
5. ***Μονοτονία***. Αν για όλα τα j, $w_j^i\le\upsilon_j^i$ τότε για κάθε έξοδο x ισχύει, $w^i\left(x^i\right)\le\upsilon^i\left(x^i\right)$. Αν στο προηγούμενο παράδειγμα τα κόστη όλων των ακμών του i ήταν μεγαλύτερα τότε επειδή το υ_j^i αντιστοιχεί στο $-t_e$ θα είχαμε $w_j^i\le\upsilon_j^i$. Επίσης επειδή η αποτιμησή ορίζεται ως $\upsilon^i=-\sum_{(e\in R\ \text{ανήκει στον i})}t_e$, θα ίσχυε $w^i\left(x^i\right)\le\upsilon^i\left(x^i\right)$ για κάθε έξοδο x.
6. ***Συνθήκη επιβολής***. Για κάθε τύπο υ, δεκτή έξοδο x και πραγματικό αριθμό α, ορίζουμε τον τύπο υ[α] ως: $\upsilon[\alpha]_j^i=\begin{cases}\upsilon_j^i & \text{εαν } x_j^i=1\\ \alpha & \text{αλλιως}\end{cases}$

   Η συνθήκη επιβολής ικανοποιείται αν για κάθε αποδεκτή έξοδο $y\ne x$

έχουμε, $\lim_{\alpha\to-\infty} g(t(\alpha), y) = -\infty$.

Για το προηγούμενο παράδειγμα κάθε τύπος υ αποτελείται τους τύπους $\upsilon^i = (\upsilon^i_1, \ldots, \upsilon^i_{m^i})$ των παιχτών που είναι τα κόστη των ακμών τους. Για κάθε εφικτή έξοδο, δηλαδή για κάθε δέντρο R, ο τύπος $\upsilon[\alpha]$ υπολογίζεται ως εξής: Για κάθε παίχτη i για να πάρουμε τα $\upsilon[\alpha]^i_j$ κρατάμε στο $\upsilon^i = (\upsilon^i_1, \ldots, \upsilon^i_{m^i})$ τα υ^i_j για τα οποία $x^i_j = 1$, δηλαδή κρατούμε τα κόστη για τις ακμές που ανήκουν στο R , τα υπόλοιπα τα αντικαθιστούμε με α. Έτσι παράγουμε το $\upsilon[\alpha]^i$. Αν το $\alpha \to -\infty$ τότε για κάθε δέντρο R΄ διαφορετικό του R η αντικειμενική συνάρτηση g, δηλαδή το κόστος του δέντρου αυτού με βάση τους νέους τύπους θα τείνει στο $-\infty$. Αυτό είναι προφανές διότι στο R΄ θα ανήκουν ακμές με κόστη α (βάση των νέων τύπων $\upsilon[\alpha]^i$) και $\alpha \to -\infty$. Άρα για το μοντέλο των multicast transmissions που ορίσαμε η συνθήκη επιβολής ικανοποιείται.

Σημείωση: Για τύπο υ, καλούμε με $g_{opt}(\upsilon)$ την βέλτιστη τιμή της g. Δηλαδή είναι η τιμή της g που παίρνουμε χρησιμοποιώντας ένα βέλτιστο αλγόριθμο εξόδου. Θα καλούμε με $g_k(\upsilon)$ την $g(\upsilon, k(\upsilon))$, δηλαδή είναι η τιμή της g που προκύπτει με βάση τον αλγόριθμο εξόδου k..

Ορισμός 3.1.14 (Εκφυλισμένος αλγόριθμος)
Ένας αλγόριθμος εξόδου k θα καλείται *εκφυλισμένος (degenerate)*, εάν ο λόγος $r_k(\upsilon) = \dfrac{g_k(\upsilon) - g_{opt}(\upsilon)}{|g_{opt}(\upsilon)| + 1}$ δεν είναι φραγμένος. Αυτό σημαίνει ότι υπάρχει υ για το οποίο το $r_k(\upsilon)$ είναι αυθαίρετα μεγάλο.

Παρατήρηση: Αν ένας αλγόριθμος είναι εκφυλισμένος θα δίνει εξόδους που είναι αυθαίρετα μακριά από το βέλτιστο. Κατά συνέπεια ένας μηχανισμός με εκφυλισμένο αλγόριθμο εξόδου θα είναι αναξιόπιστος. Παρατηρούμε ότι από τον ορισμό του $r_k(\upsilon)$ παίρνουμε τη σχέση $r_k(\upsilon)|g_{opt}(\upsilon)| + r_k(\upsilon) = g_k(\upsilon) - g_{opt}(\upsilon)$, συνεπώς ένας εκφυλισμένος αλγόριθμος είναι αυθαίρετα μακριά από το βέλτιστο και προσθετικά και πολλαπλασιαστικά.

Θεώρημα 3.1.15 Εάν ένας VCG-based μηχανισμός για κάποιο CMAP είναι φιλαλήθης τότε ο αλγόριθμος εξόδου του είναι ή βέλτιστος ή εκφυλισμένος.

Απόδειξη: Έστω $m = (k, p)$ ένας φιλαλήθης VCG-based μηχανισμός για CMAP. Υποθέτουμε ότι $p^i(w) = \sum_{j\neq i} w^j(k(w))$. Έστω υ τύπος για τον οποίο το k(υ) δεν είναι βέλτιστο και $y = opt(\upsilon)$ είναι η βέλτιστη έξοδος.

Ορίζουμε τον τύπο z ως : $z_j^i = \begin{cases} \upsilon_j^i & \text{εαν } y_j^i=1 \\ -\alpha & \text{αλλιως} \end{cases}$ όπου το α είναι αυθαίρετα μεγάλο. Ο τύπος z περιέχει τις αποτιμήσεις των παιχτών που ανήκουν στη βέλτιστη έξοδο y και τις υπόλοιπες τις αντικαθιστά με –α.

Θεωρούμε την ακολουθία τύπων:

$$\upsilon_0 = \left(\upsilon^1, \ldots, \upsilon^n\right)$$
$$\upsilon_1 = \left(z^1, \upsilon^2, \ldots, \upsilon^n\right)$$
$$\vdots$$
$$\upsilon_n = \left(z^1, \ldots, z^n\right)$$

Ισχυρισμός 1 Για όλα τα $j,\ y = \mathrm{opt}\left(\upsilon_j\right)$

Απόδειξη: Το y είναι βέλτιστο για τον υ_0 από τον ορισμό του y. Από την συνθήκη της ανεξαρτησίας στον ορισμού του CMAP και τον ορισμό του z έχουμε ότι $g\left(\upsilon_j, y\right) = g\left(\upsilon_0, y\right)$. Από την συνθήκη της μονοτονίας και τον ορισμό του z έχουμε $g\left(\upsilon_j, x\right) \le g\left(\upsilon_0, x\right)$ για όλα τα x. Άρα ο ισχυρισμός είναι αληθής. ■

Ισχυρισμός 2 $g\left(\upsilon_1, k\left(\upsilon_1\right)\right) < g\left(\upsilon_1, y\right)$

Απόδειξη: Αν δεν ίσχυε τότε, $g\left(\upsilon_1, k\left(\upsilon_1\right)\right) = g\left(\upsilon_1, y\right) = g\left(\upsilon_0, y\right) > g\left(\upsilon_0, k\left(\upsilon_0\right)\right)$. Συνεπώς , όταν ο τύπος του παίχτη 1 είναι υ^1 και οι δηλώσεις των άλλων παιχτών είναι $\upsilon^2, \ldots, \upsilon^n$, είναι καλύτερο για τον παίχτη 1 να δηλώσει z^1. Καταλήγουμε όμως σε άτοπο διότι ο μηχανισμός είναι φιλαλήθης. ■

Τελικά παίρνουμε ότι $g\left(z, k\left(z\right)\right) < g\left(z, y\right) = g\left(\upsilon_0, y\right)$. Από την συνθήκη επιβολής έχουμε ότι όταν $\alpha \to \infty$ τότε $g\left(z, k\left(z\right)\right) \to -\infty$. Και το θεώρημα έχει αποδειχθεί.

■

Παρατηρήσεις: Το θεώρημα αυτό είναι πολύ σημαντικό. Ουσιαστικά μας λέει ότι δεν μπορούμε να σχεδιάσουμε ένα φιλαλήθη VCG-based μηχανισμό για κάποιο CMAP με αλγόριθμο εξόδου που δεν είναι βέλτιστος. Μόνο αν ο αλγόριθμος εξόδου είναι βέλτιστος θα έχουμε εγγύηση για την αξιοπιστία των αποτελεσμάτων. Αν ο μηχανισμός είναι φιλαλήθης αλλά ο αλγόριθμος εξόδου δεν είναι βέλτιστος τότε ο αλγόριθμος αυτός θα είναι εκφυλισμένος δηλαδή ο μηχανισμός έχει ανώμαλη συμπεριφορά. Όταν ένα CMAP είναι NP-complete τότε ένας βέλτιστος αλγόριθμος γι' αυτό το πρόβλημα θα είναι υπολογιστικά πολύ δύσκολος. Αλλά οποιοσδήποτε υπολογιστικά εφικτός και φιλαλήθης VCG-based μηχανισμός θα δίνει ανώμαλα αποτελέσματα και δεν θα είναι επιτυχής υλοποίηση του προβλήματος.

3.1.4 Υπολογιστικά Εφικτοί VCG Μηχανισμοί

Μέχρι σήμερα οι VCG μηχανισμοί είναι η μόνη γενική μέθοδος για να σχεδιάσουμε φιλαλήθεις μηχανισμούς. Από τα προηγούμενα αποτελέσματα όμως φαίνεται ότι είναι πολύ δύσκολο να σχεδιάσουμε υπολογιστικά εφικτούς φιλαλήθεις μηχανισμούς.

Στη συνέχεια θα αναπτύξουμε εφικτούς μηχανισμούς που θα δίνουν λύσεις στο ίδιο πνεύμα με τους φιλαλήθεις. Θα ορίσουμε ξανά την έννοια της κυρίαρχης στρατηγικής, υποθέτοντας ότι οι παίχτες έχουν κάποιους υπολογιστικούς περιορισμούς στο να επιλέξουν την καλύτερη στρατηγική.

Στο μοντέλο των VCG μηχανισμών που μελετήσαμε θεωρούμε ότι οι παίχτες είναι λογικοί και ιδιοτελείς. Το ότι οι παίχτες θεωρούνται λογικοί έχει την έννοια ότι πάντα θα επιλέγουν την καλύτερη στρατηγική για το παίγνιο που συμμετέχουν. Για να επιλέξουν όμως την καλύτερη στρατηγική θα πρέπει να θεωρήσουμε ότι δεν έχουν υπολογιστικούς περιορισμούς. Μελετούν τα χαρακτηριστικά του μηχανισμού, τον αλγόριθμο εξόδου και όλα τα αποτελέσματα από όλες τις δυνατές στρατηγικές και επιλέγουν εκείνη που θα μεγιστοποίηση το κέρδος τους. Έτσι αν ο μηχανισμός είναι VCG οι παίχτες μπορούν να συμπεράνουν ότι η καλύτερη στρατηγική είναι η φιλαλήθεια. Όταν όμως ένα πρόβλημα είναι υπολογιστικά δύσκολο τότε θα είναι δύσκολο και για τους παίχτες να υπολογίσουν, μέσα σε κάποιο λογικό χρονικό διάστημα, την καλύτερη ενέργεια από τον χώρο όλων των δυνατών ενεργειών.

Με σκοπό να σχεδιάσουμε υπολογιστικά εφικτούς μηχανισμούς που δεν θα έχουν όμως ανώμαλη συμπεριφορά, θα θεωρήσουμε ένα πιο ρεαλιστικό μοντέλο στο οποίο η υπολογιστική ισχύς των παιχτών θα είναι περιορισμένη.

Υποθέτουμε ότι οι παίχτες επιλέγουν τις ενέργειες τους σύμφωνα με κάποια γνώση που έχουν από την αρχή του παιγνίου. Η γνώση αυτή μπορεί να αντιπροσωπεύει τους υπολογιστικούς περιορισμούς του παίχτη ή κάποιους περιορισμούς στην πληροφόρηση ή στην συμπεριφορά του. Βάση αυτών των περιορισμών θα ορίσουμε την εφικτή κυρίαρχη στρατηγική. Μία ενέργεια θα είναι εφικτή κυρίαρχη για τον παίχτη i αν για οποιαδήποτε κατάσταση δεν μπορεί να *αντιληφθεί* καλύτερη ενέργεια από αυτή. Ο παίχτης υπολογίζει την εφικτή κυρίαρχη ενέργεια μέσα σε κάποιο λογικό χρονικό διάστημα, έχει δηλαδή κάποιους χρονικούς περιορισμούς.

Σημείωση: Θα καλούμε με A^i τον χώρο ενεργειών του παίχτη i. Δοθείσας μιας n-άδας ενεργειών $\alpha = (\alpha^1, \ldots, \alpha^n)$ επιλεγμένων από τους παίχτες, θα καλούμε με $u^i(\alpha)$ την utility του παίχτη i.

Ορισμός 3.1.16 (Στρατηγική γνώση) *Στρατηγική γνώση (Strategic knowledge)* ή για συντομία *γνώση (knowledge)* του παίχτη i, είναι μια μερική συνάρτηση $b^i : A^{-i} \to A^i$.

Ερμηνεία: Γνώση είναι μια συνάρτηση με την οποία ο παίχτης περιγράφει πως θα έπρεπε να αντιδράσει σε οποιαδήποτε κατάσταση. Η σημασιολογία του

$\alpha^i = b^i(\alpha^{-i})$ είναι: «όταν οι ενέργειες των άλλων παιχτών είναι α^{-i} τότε η καλύτερη ενέργεια για μένα είναι η α^i ». Θεωρούμε ότι κάθε παίχτης υπολογίζει την γνώση του $b^i(\alpha^{-i})$ μέσα σε κάποιο λογικό χρονικό διάστημα.

Ορισμός 3.1.17 (Εφικτή καλύτερη αντίδραση)
Μία ενέργεια α^i του παίχτη i θα καλείται *εφικτή καλύτερη αντίδραση (feasible best response) στο* α^{-i}, αν το α^{-i} δεν ανήκει στο πεδίο ορισμού της γνώσης του b^i ή αν έχουμε $u^i(b^i(\alpha^{-i}), \alpha^{-i}) \leq u^i(\alpha)$.

Δηλαδή, μπορεί άλλες ενέργειες να είναι καλύτερες εναντίον των α^{-i} αλλά όταν ο παίχτης επέλεξε την α^i δεν τις γνώριζε. Επίσης μπορούμε να παρατηρήσουμε ότι αν το α^{-i} δεν είναι στο πεδίο ορισμού της b^i τότε κάθε ενέργεια α^i είναι καλύτερη εφικτή αντίδραση στο α^{-i}.

Ορισμός 3.1.18 (Εφικτή κυρίαρχη ενέργεια)
Μία ενέργεια α^i ενός παίχτη i θα καλείται *εφικτή κυρίαρχη ενέργεια (feasibly dominant action)* εάν είναι εφικτή καλύτερη αντίδραση ενάντια σε κάθε α^{-i}.Θα συμβολίζουμε μια τέτοια ενέργεια με fdα.

Προφανώς μια κυρίαρχη ενέργεια είναι και εφικτή κυρίαρχη.

Παρόλο που οι VCG-based μηχανισμοί μπορεί να μην είναι φιλαλήθεις, είναι εύκολο να δούμε ότι ο μόνος λόγος για τον οποίο ένας παίχτης μπορεί να ψεύδεται για τον τύπο του, είναι για να βοηθήσει τον αλγόριθμο να βελτιώσει το συνολικό αποτέλεσμα. Τα προβλήματα που μελετούμε είναι ωφελιμιστικά. Σε αυτά η αντικειμενική συνάρτηση του προβλήματος είναι το άθροισμα των αποτιμήσεων των παιχτών. Ένας παίχτης ψεύδεται με σκοπό αυξήσει το προσωπικό του όφελος δηλαδή την utility του. Όμως η utility κάθε παίχτη εξαρτάται από την έξοδο του μηχανισμού δηλαδή από το συνολικό αποτέλεσμα. Άρα κάθε παίχτης αναγκαστικά θα ενδιαφέρεται για το αποτέλεσμα του μηχανισμού. Όταν όμως ένας παίχτης πιστεύει ότι σύμφωνα με τις δηλώσεις των άλλων παιχτών τον συμφέρει να ψεύδεται, μπορεί τελικά να κάνει λάθος και συνεπώς να μειώσει το τελικό αποτέλεσμα του μηχανισμού.

Για να αποφύγουμε τέτοια φαινόμενα θα ορίσουμε την *συνάρτηση προσφυγής l*. Η σημασιολογία της προσφυγής l είναι: «όταν ο τύπος των παιχτών είναι $\upsilon = (\upsilon^1, \ldots, \upsilon^n)$ τότε πιστεύω ότι ο αλγόριθμος εξόδου k θα δώσει καλύτερα αποτελέσματα για την είσοδο l(υ) αντί για την υ». Δοθέντος των δηλώσεων w των παιχτών, ο μηχανισμός υπολογίζει το k(w) καθώς και τα αποτελέσματα $k(l^1(w)), \ldots, k(l^n(w))$ απ' όλες τις προσφυγές των παιχτών και τελικά επιλέγει την καλύτερη έξοδο από αυτά τα αποτελέσματα.

Η βασική ιδέα είναι ότι αντί ο παίχτης να δηλώσει ψευδή τύπο, είναι καλύτερα να είναι ειλικρινής και απλά να ζητήσει του μηχανισμού να ελέγξει κατά πόσο μία

ψευδή δήλωση μπορεί να δώσει καλύτερα αποτελέσματα. Αν οι παίχτες είναι φιλαλήθεις τότε η έξοδος που επιλέγει ο μηχανισμός θα είναι τουλάχιστον τόσο καλή όσο και η k(υ).

Ορισμός 3.1.19 (Συνάρτηση προσφυγής) Έστω $V = \prod_i V^i$ δηλώνει τον χώρο των τύπων των παιχτών. *Προσφυγή (appeal)* είναι μία συνάρτηση $l: V \to V$.

Θα ορίσουμε τώρα τον μηχανισμό δεύτερης ευκαιρίας.

Ορισμός 3.1.20 (Ο Μηχανισμός Δεύτερης Ευκαιρίας)
Δοθέντος ενός αλγορίθμου εξόδου k, *ο μηχανισμός δεύτερης ευκαιρίας που βασίζεται στον k (second chance mechanism)* είναι το ακόλουθο παίγνιο:

1. Κάθε παίχτης στέλνει στον μηχανισμό μία δήλωση του τύπου του w^i και μία συνάρτηση προσφυγής l^i.

2. Έστω $w = (w^1, \ldots, w^n)$ οι δηλώσεις των παιχτών. Ο μηχανισμός υπολογίζει τα $k(w), k(l^1(w)), \ldots, k(l^n(w))$ και επιλέγει από αυτές τις εξόδους εκείνη που μεγιστοποιεί το συνολικό όφελος.. Με άλλα λόγια ο μηχανισμός δοκιμάζει όλες τις προσφυγές (appeals) και επιλέγει εκείνη που δίνει το καλύτερο αποτέλεσμα.

3. Έστω ô η έξοδος που έχει επιλεγεί. Ο μηχανισμός υπολογίζει τις πληρωμές σύμφωνα με τον VCG τύπο: $p^i = \sum_{j \neq i} w^j(\hat{o}) + h^i(w^{-i})$, όπου $h^i(w^{-i})$ είναι μια αυθαίρετη συνάρτηση και είναι ανεξάρτητη από τις προσφυγές των παιχτών.

4. *Ενέργεια (action)* του παίχτη i είναι το ζεύγος (w^i, l^i), όπου w^i είναι ο τύπος που δηλώθηκε και l^i είναι η συνάρτηση προσφυγής.

Ορισμός 3.1.21 (Φιλαλήθης Ενέργεια / Εφικτά Φιλαλήθης Μηχανισμός)
- Μία ενέργεια $\alpha^i = (w^i, l^i)$ θα καλείται *φιλαλήθης* εάν $w^i = \upsilon^i$. Δηλαδή οι παίχτες αποκαλύπτουν στον μηχανισμό τους αληθινούς τους τύπους.
- Ένας μηχανισμός θα καλείται *εφικτά φιλαλήθης (feasibly truthful)* εάν η φιλαλήθης ενέργεια είναι κυρίαρχη για όλους τους παίχτες.

Αποδεικνύεται ότι:

Πρόταση 3.1.22 Θεωρούμε ένα μηχανισμό δεύτερης ευκαιρίας με αλγόριθμο εξόδου k. Εάν όλοι οι παίχτες είναι φιλαλήθεις για κάθε τύπο $\upsilon = (\upsilon^1, \ldots, \upsilon^n)$, τότε η έξοδος που επιλέγει ο μηχανισμός είναι τουλάχιστον το ίδιο καλή με την

k(υ).

Παρατηρήσεις: Όταν οι παίχτες είναι φιλαλήθεις τότε ο μηχανισμός είναι τουλάχιστον τόσο καλός όσο και ο αλγόριθμος εξόδου. Αυτό έχει την έννοια ότι δεν υπάρχει περίπτωση οι προσφυγές των παιχτών να επιδράσουν αρνητικά στο τελικό αποτέλεσμα του μηχανισμού. Ο μηχανισμός συγκρίνει την έξοδο k(υ) με τις $k\left(l^1(\upsilon)\right),\ldots,k\left(l^n(\upsilon)\right)$. Αν οι προσφυγές των παιχτών δεν οδηγούν σε καλύτερο αποτέλεσμα τότε ο μηχανισμός θα επιλέξει την έξοδο k(υ) . Συνεπώς αν ο αλγόριθμος εξόδου είναι c-προσεγγιστικός τότε την ίδια προσέγγιση θα επιτυγχάνει και ο μηχανισμός.

Παράδειγμα: Ας θεωρήσουμε ένα παίχτη i που συμμετέχει σε κάποιο πλειστηριασμό κατοικιών. Έστω ότι ενδιαφέρεται για δύο συγκεκριμένα γειτονικά διαμερίσματα, για τα οποία προτίθεται να πληρώσει 200.000 ευρώ και για τα δύο ή 60.000 ευρώ για το καθένα ξεχωριστά. Ο τύπος του i είναι $\upsilon^i = \{200.000, 60.000, 60.000\}$. Μετά από κάποια έρευνα που έκανε ο i διαπίστωσε ότι σε πολλές περιπτώσεις η δήλωση $w^i = \{200.000, 0, 0\}$ οδηγεί τον αλγόριθμο σε καλύτερα αποτελέσματα.

Σε ένα VCG-based μηχανισμό ο i μπορεί να δήλωνε w^i αντί υ^i. Έτσι σε κάποιες περιπτώσεις, ανάλογα και με τις δηλώσεις των άλλων παιχτών, θα βελτιώνονταν το συνολικό όφελος, άρα και το όφελος του i, ενώ σε κάποιες άλλες μπορεί να συνέβαινε το αντίθετο.

Σε ένα μηχανισμό δεύτερης ευκαιρίας ο i θα δηλώσει τον αληθή τύπο του υ^i και θα ορίσει την προσφυγή $l^i\left(\upsilon^i, w^{-i}\right)$ ως $\left(w^i, w^{-i}\right)$. Έτσι για τις περιπτώσεις όπου η δήλωση w^i βελτιώνει το συνολικό όφελος ο μηχανισμός θα επιλέξει την $k\left(l^i(\upsilon)\right)$. Για τις περιπτώσεις όπου η ψευδείς δηλώσεις μειώνουν το συνολικό όφελος, ο μηχανισμός θα επιλέξει την k(υ).

Στη συνέχεια θα δείξουμε ότι κάτω από κάποιες προϋποθέσεις υπάρχει για τους παίχτες υπολογιστικά εφικτή κυρίαρχη φιλαλήθης ενέργεια.

Παρατηρήσεις:
1. Αν δεν επιβληθούν χρονικοί περιορισμοί στις συναρτήσεις προσφυγής τότε οι παίχτες θα έχουν κυρίαρχες ενέργειες (στρατηγικές). Έστω παίχτης i που συμμετέχει σε ένα μηχανισμό δεύτερης ευκαιρίας, όπου k είναι ο αλγόριθμος εξόδου. Δοθέντος των δηλώσεων των τύπων w^{-i} των άλλων παιχτών, υπάρχει διάνυσμα τύπων $\hat{w} \stackrel{df}{=} l^i\left(w^{-i}\right)$ που μεγιστοποιεί το $g\left(\left(\upsilon^i, w^{-i}\right), k(\hat{w})\right)$. Είναι εύκολο να δούμε ότι η $\left(\upsilon^i, l^i\right)$ είναι κυρίαρχη και κατά συνέπεια εφικτή κυρίαρχη ενέργεια για τον παίχτη i. Παρόλα αυτά αν το πρόβλημα είναι πολύ δύσκολο στον υπολογισμό του τότε δύσκολος θα είναι και ο υπολογισμός της l^i.
2. Όταν στους παίχτες επιβάλλεται να υποβάλουν μόνο υπολογιστικά

περιορισμένες συναρτήσεις προσφυγής, τότε δεν είναι προφανής η ύπαρξη εφικτής φιλαλήθης κυρίαρχης ενέργειας. Αν οι συναρτήσεις προσφυγής είναι υπολογιστικά δύσκολές τότε ο μηχανισμός δεύτερης ευκαιρίας δεν θα είναι πολυωνιμικού χρόνου ακόμα και αν ο αλγόριθμος εξόδου k είναι πολυωνιμικός. Αυτό προκύπτει από το γεγονός ότι ο μηχανισμός πρέπει να επιλέξει την καλύτερη έξοδο από τις $k(w), k(l^1(w)), \ldots, k(l^n(w))$. Συνεπώς θα πρέπει να προηγηθεί ο υπολογισμός των $l^1(w), \ldots, l^n(w)$. Αν όμως ο υπολογισμός τους είναι εκθετικού χρόνου τότε εκθετικού χρόνου θα είναι και ο μηχανισμός ακόμα και αν ο k είναι πολυωνιμικός. Η απαίτηση λοιπόν να είναι οι συναρτήσεις προσφηγής να είναι υπολογιστικά εφικτές είναι λογική.

Με βάση τις προηγούμενες παρατηρήσεις πρέπει να δείξουμε την ύπαρξη εφικτής φιλαλήθης κυρίαρχης ενέργειας. ακόμα και αν περιορίσουμε την γνώση των παιχτών. Θα επικεντρωθούμε σε δύο τύπους *γνώσης*:
I. Ο πρώτος τύπος είναι αυτός που αποκτά ο παίχτης απλώς εξερευνώντας τον αλγόριθμο εξόδου. Ένας τέτοιος παίχτης αδιαφορεί για τις ενδεχόμενες προσφυγές των άλλων παιχτών. Ουσιαστικά η γνώση του είναι γνώση για τον αλγόριθμο. Μελετά συνεπώς ποια θα είναι τα αποτελέσματα του αλγορίθμου αν κάνει δηλώσεις διαφορετικές από τον αληθή τύπο του. Για τους άλλους παίχτες χρησιμοποιεί ως δεδομένα μόνο τις δηλώσεις τους. Οι παίχτες έχουν τέτοιο είδος γνώσης όταν ο χώρος των πιθανών προσφυγών των άλλων παιχτών είναι πολύ μεγάλος και πολύπλοκος για τις δυνατότητες τους και έτσι δεν μπορούν να τον μελετήσουν.
II. Ο δεύτερος τύπος γνώσης είναι εκείνος που αποκτούν οι παίχτες όταν, μαζί με τον αλγόριθμο εξόδου, εξετάζουν και μία μικρή οικογένεια ενδεχόμενων προσφυγών των άλλων παιχτών. «Τρέχουν» συνεπώς τον αλγόριθμο εξόδου για τις δικές τους προσφυγές αλλά και για κάποιες πιθανές προσφυγές των άλλων παιχτών.

Θα δείξουμε και για τους δύο τύπους γνώσης, υπάρχει για τους παίχτες υπολογιστικά εφικτή κυρίαρχη φιλαλήθης ενέργεια. Ξεκινούμε με τους ορισμούς των υπολογιστικά περιορισμένων αλγορίθμων και μηχανισμών.

Ορισμός 3.1.23 (Αλγόριθμος βαθμού d) Ένας αλγόριθμος εξόδου είναι *βαθμού d*, εάν ο χρόνος λειτουργίας του είναι φραγμένος από ένα πολυώνυμο βαθμού d ως προς τον αριθμό των παιχτών.

Δηλαδή ένας αλγόριθμος βαθμού d είναι πολυωνιμικός. Με όμοιο τρόπο μπορούμε να ορίσουμε συναρτήσεις προσφυγής, πράξεις και μηχανισμούς βαθμού d.

Αποδεικνύεται ότι:

Πρόταση 3.1.24 Εάν ο αλγόριθμος εξόδου ενός μηχανισμού δεύτερης ευκαιρίας είναι βαθμού d, τότε ο μηχανισμός είναι βαθμού $d+1$.

Στη συνέχεια δίνουμε τον ορισμό του $1^{ου}$ τύπου γνώσης (που αναφέραμε προηγουμένως) που αποκτά ένας παίχτης όταν απλώς εξερευνά τον αλγόριθμο εξόδου του μηχανισμού.

Ορισμός 3.1.25 (γνώση ανεξάρτητη προσφυγής)

- Η γνώση b^i θα καλείται *γνώση ανεξάρτητη προσφυγής (appeal independent knowledge),* εάν είναι της μορφής: $b^i : V^{-i} \to A^i$.
- Τέτοιου είδους γνώση θα καλείται *γνώση βασισμένη σε δήλωση (declaration based knowledge)* εάν είναι της μορφής: $b^i : V^{-i} \to V^i$.

Παρατηρήσεις:

1. Γενικά η συνάρτηση γνώσης ορίζεται ως $b^i : A^{-i} \to A^i$, έχει πεδίο ορισμού το χώρο των πιθανών ενεργειών των υπολοίπων παιχτών. Σε ένα μηχανισμό δεύτερης ευκαιρίας οι ενέργειες των παιχτών είναι τα ζεύγη (w^i, l^i). Η γνώση ανεξάρτητη προσφυγής για τον παίχτη i έχει ως πεδίο ορισμού μόνο το χώρο δηλώσεων των υπολοίπων παιχτών. Δηλαδή ο i μελετά τις πιθανές δηλώσεις w^{-i} των άλλων παιχτών και υπολογίζει την ενέργεια $(w^i, l^i) = b^i(w^{-i})$ χωρίς να λάβει υπόψη του τις προσφυγές των υπολοίπων παιχτών. Η σημασιολογία του $(w^i, l^i) = b^i(w^{-i})$ είναι: «όταν οι άλλοι δηλώνουν w^{-i} θα δηλώνω w^i και θα υποβάλλω την προσφυγή l^i.

2. Η σημασιολογία της γνώσης βασισμένης σε δήλωση είναι: «όταν οι άλλοι δηλώνουν w^{-i}, τότε θα δηλώνω $b^i(w^{-i})$».

Θεώρημα 3.1.26 Εάν για τον παίχτη i η b^i είναι γνώση βασισμένη σε δήλωση, τότε η (υ^i, b^i) είναι εφικτή κυρίαρχη ενέργεια για τον παίχτη i.

Απόδειξη: Έστω ότι η (υ^i, b^i) δεν είναι εφικτή κυρίαρχη. Τότε θα υπάρχει ένα διάνυσμα ενεργειών α^{-i} των υπόλοιπων παιχτών έτσι ώστε $(w^i, \varnothing) \overset{df}{=} b^i(\alpha^{-i})$ είναι καλύτερο για τον παίχτη από το να παίξει (υ^i, b^i). Επισημαίνουμε ότι η προσφυγή του i είναι πάντα κενή. Καθώς η γνώση b^i είναι βασισμένη σε δήλωση μπορούμε να θεωρήσουμε ότι και οι προσφυγές των άλλων παιχτών είναι κενές. Θα καλούμε με $(w^{-i}, \varnothing)$ το α^{-i}. Θεωρούμε την περίπτωση όπου η ενέργεια του παίχτη είναι (υ^i, b^i). Εάν ο μηχανισμός απορρίψει την προσφυγή και επιλέξει τον τύπο (υ^i, w^{-i}) τότε $g((\upsilon^i, w^{-i}), k(\upsilon^i, w^{-i})) \geq g((\upsilon^i, w^{-i}), k(w^i, w^{-i}))$.Ο παίχτης δηλώνοντας w^i αναγκάζει τον μηχανισμό να επιλέξει την $k(w^i, w^{-i})$ και η utility του παίχτη μειώνεται. Εάν ο μηχανισμός αποδεχθεί την προσφυγή και επιλέξει τον τύπο (w^i, w^{-i}) τότε κάνει το ίδιο όταν ο παίχτης δηλώσει w^i. Άρα τελικά η

(υ^i, b^i) είναι εφικτή κυρίαρχη ενέργεια ■

Θεώρημα 3.1.27 Εάν ο αλγόριθμος εξόδου είναι βαθμού d και ο παίχτης έχει γνώση ανεξάρτητη προσφυγής βαθμού d, τότε υπάρχει φιλαλήθης κυρίαρχη ενέργεια βαθμού d για τον παίχτη.

Απόδειξη: Έστω b^i γνώση βαθμού d του παίχτη i. Ορίζουμε την προσφυγή l^i ως εξής: Δοθέντος των w^{-i}, έστω $(w^i, \psi^i) = b^i(w^{-i})$. Η προσφυγή που ορίζουμε υπολογίζει τα $k(w^i, w^{-i})$ και $k(\psi(w^i, w^{-i}))$ και επιλέγει το καλύτερο αποτέλεσμα σύμφωνα με το (υ^i, w^{-i}). Προφανώς η l^i είναι εφικτή κυρίαρχη βαθμού d. ■

Τώρα θα εξετάσουμε τον 2^o τύπο γνώσης όπου ο παίχτης, εκτός από τον αλγόριθμο εξόδου, εξετάζει και ένα μικρό μέρος από τις προσφυγές των άλλων παιχτών.

Ορισμός 3.1.28 (d-obtainable γνώση) Η γνώση b^i θα καλείται *d-obtainable* (*d-αποκτούμενη)* εάν ισχύει:

1. Η b^i είναι βαθμού d.
2. Κάθε συνάρτηση προσφυγής που εμφανίζεται στο πεδίο ορισμού ή στο πεδίο τιμών της b^i, είναι βαθμού d.
3. Υπάρχουν το πολύ n^d συναρτήσεις προσφυγής που εμφανίζονται στο πεδίο ορισμού ή στο πεδίο τιμών της b^i. Επιπλέον υπάρχουν υπάρχει μία αντιπροσωπευτική οικογένεια L^i με όχι περισσότερα από $n^d(n-1)$ ακολουθίες προσφυγών, έτσι ώστε για κάθε ακολουθία φ^{-i} που εμφανίζεται στο πεδίο ορισμού της b^i, να υπάρχει $\psi^i \in L^i$ ώστε για όλα τα w^{-i}, $b^i(w^{-i}, \varphi^{-i}) = b^i(w^{-i}, \psi^{-i})$.

Παρατηρήσεις:
1. Στον παραπάνω ορισμό η δεύτερη συνθήκη δικαιολογείται από την υπόθεση ότι ο παίχτης δεν μπορεί να εξετάσει συναρτήσεις που δεν μπορεί να υπολογίσει. Συνεπώς αφού η γνώση του είναι βαθμού d τότε και κάθε συνάρτηση προσφυγής που μελετά ή παράγει θα πρέπει να είναι επίσης βαθμού d.
2. Στην τρίτη συνθήκη η αντιπροσωπευτική οικογένεια αντιπροσωπεύει όλες τις περιπτώσεις που έχει εξετάσει ή έχει παράγει ο παίχτης. Η υπόθεση είναι ότι ο παίχτης δεν μπορεί να βγάλει συμπεράσματα για συναρτήσεις ή ακολουθίες που δεν έχει ελέγξει τουλάχιστον μία φορά. Υποθέτουμε επίσης ότι οι παίχτες μπορούν να εξετάσουν μόνο κάποιο λογικό αριθμό περιπτώσεων.

Μπορεί να αποδειχθεί το ακόλουθο θεώρημα.

Θεώρημα 3.1.29 Εάν ο αλγόριθμος εξόδου είναι βαθμού d και ο παίχτης έχει d-obtainable γνώση, τότε έχει μια εφικτή φιλαλήθης κυρίαρχη ενέργεια βαθμού $3 \cdot d$.

Σχόλια: Από τα παραπάνω θεωρήματα προκύπτει το συμπέρασμα ότι και για τους δύο τύπους γνώσης υπάρχει σε μηχανισμούς βαθμού d εφικτή φιλαλήθης κυρίαρχη ενέργεια για τους παίχτες. Συνεπώς όταν οι παίχτες αποκτούν την γνώση τους με κάποιο από τους παραπάνω τρόπους και οι χρονικοί περιορισμοί για τις συναρτήσεις προσφυγής δεν είναι πολύ μικροί, τότε παρακινούνται να αποκαλύψουν τους αληθούς τύπους τους στον μηχανισμό. Άρα τέτοιου είδους μηχανισμοί συμπεριφέρονται κατά τρόπο όμοιο με τους κλασικούς φιλαλήθεις μηχανισμούς. Επίσης καθώς είναι βαθμού d δηλαδή πολυωνιμικού χρόνου είναι υπολογιστικά εφικτοί.

3.1.5 Μέθοδοι Υλοποίησης

Θεωρούμε δύο φυσικούς τρόπους για την υλοποίηση των συναρτήσεων προσφυγής.

Ο πρώτος τρόπος είναι να αφήσουμε τους παίχτες να υπολογίσουν οι ίδιοι τις προσφυγές τους και να δώσουν τα αποτελέσματα στον μηχανισμό στον δεύτερο γύρο. Σε αυτή την μέθοδο ο τύπος κάθε παίχτη θα πρέπει να είναι γνωστός στους άλλους ώστε να μπορούν να εκτελέσουν τους υπολογισμούς.. Κάτι τέτοιο όμως δεν είναι γενικά επιθυμητό για τις περισσότερες εφαρμογές.

Ο δεύτερος τρόπος είναι να εφοδιάσουμε τους παίχτες με μία γλώσσα για την περιγραφή των προσφυγών τους και να εκτελεστεί ο υπολογισμός από την μηχανή του μηχανισμού. Και στις δύο μεθόδους μπορούμε να επιβάλουμε χρονικά όρια στους παίχτες ή να τους ζητούμε να πληρώνουν τον έξτρα χρόνο υπολογισμού.

Ένα άλλο ζήτημα είναι οι *περιορισμοί συμμετοχής*. Συνήθως επιθυμούμε η utility ενός φιλαλήθη παίχτη να είναι πάντα θετική. Αυτή η ιδιότητα καλείται συνήθως περιορισμός συμμετοχής. Στη συνέχεια θα σχεδιάσουμε ένα μηχανισμό δεύτερης ευκαιρίας που θα ικανοποιεί αυτή την ιδιότητα.

Υποθέτουμε ότι για κάθε παίχτη i υπάρχει τύπος $\underline{\upsilon}^i$ έτσι ώστε για κάθε $\upsilon = (\upsilon^1, \ldots, \upsilon^n)$, να ισχύει $g_{opt}\left((\underline{\upsilon}^i, \upsilon^{-i})\right) \le g_{opt}(\upsilon)$. Για παράδειγμα στους συνδυαστικούς πλειστηριασμούς ο τύπος αυτός ορίζεται ως $\underline{\upsilon}^i(s) = 0$ για κάθε συνδυασμό s από αντικείμενα.

Ο Clarke μηχανισμός είναι VCG μηχανισμός με πληρωμές:

$$p^i(w) = \sum_{j \neq i} w^j(opt(w)) - g_{opt}\left(\underline{\upsilon}^i, w^{-i}\right).$$

Υπενθυμίζουμε ότι συνάρτηση πληρωμής του VCG μηχανισμού έχει την μορφή $p^i(w) = \sum_{j \neq i} w^j(k(w)) + h^i\left(w^{-i}\right)$, όπου $h^i(\ )$ είναι μία αυθαίρετη συνάρτηση. Παρατηρούμε ότι η συνάρτηση πληρωμής του Clarke είναι της κλάσης VCG όπου η αυθαίρετη συνάρτηση $h^i(\ )$ είναι η $-g_{opt}\left(\underline{\upsilon}^i, w^{-i}\right)$.

Αποδεικνύεται ότι ο μηχανισμός αυτός ικανοποιεί τους περιορισμούς συμμετοχής. Όταν ο βέλτιστος αλγόριθμος του Clarke μηχανισμού αντικατασταθεί από άλλο που δεν είναι βέλτιστος τότε το αποτέλεσμα του αλγορίθμου μπορεί να βελτιωθεί αν αντικατασταθεί το w^i με το $\underline{\upsilon}^i$. Συνεπώς οι περιορισμοί συμμετοχής μπορεί πλέον να μην ικανοποιούνται. Αυτό όμως μπορεί να διορθωθεί.

Ορισμός 3.1.30 Θα λέμε ότι ένας αλγόριθμος εξόδου k ικανοποιεί τους περιορισμούς συμμετοχής εάν για κάθε τύπο $\upsilon = (\upsilon^1, \ldots, \upsilon^n)$ και παίχτη i ισχύει $g_k(\upsilon) \geq g_k((\underline{\upsilon}^i, \upsilon^{-i}))$.

Αποδεικνύεται η ακόλουθη πρόταση.

Πρόταση 3.1.31 Εάν ο αλγόριθμος εξόδου k είναι βαθμού d, τότε υπάρχει αλγόριθμος εξόδου $\tilde{k}$ βαθμού $d+1$ που ικανοποιεί τους περιορισμούς συμμετοχής, έτσι ώστε για κάθε τύπο υ να ισχύει $g_{\tilde{k}}(\upsilon) \geq g_k(\upsilon)$.

Δοθέντος ενός αλγορίθμου $\tilde{k}$ που ικανοποιεί τους περιορισμούς συμμετοχής ορίζουμε την συνάρτηση πληρωμής του μηχανισμού δεύτερης ευκαιρίας που βασίζεται στον $\tilde{k}$ ως $p^i = \sum_{j \neq i} w^j(o) - g_{opt}(\underline{\upsilon}^i, w^{-i})$, όπου w είναι οι δηλώσεις των παιχτών και ο η επιλεγόμενη έξοδος. Καθώς $g(w,o) \geq g_{\tilde{k}}(w)$ οι περιορισμοί συμμετοχής θα ικανοποιούνται.

Μπορούμε τώρα να παραθέσουμε τα βήματα που πρέπει να ακολουθηθούν για τον σχεδιασμό ενός μηχανισμού με βάση την συνολική μέθοδο που παρουσιάσαμε.

- Αρχικά ο σχεδιαστής πρέπει να κατασκευάσει ένα αλγόριθμο k για το πρόβλημα βελτιστοποίησης που μελετά. Αυτό μπορεί να γίνει χωρίς περιορισμούς όσον αφορά την γενική συμπεριφορά του αλγορίθμου.
- Μετά μπορεί να μετασχηματίσει τον k σε αλγόριθμο $\tilde{k}$ που ικανοποιεί τους περιορισμούς συμμετοχής.
- Το επόμενο βήμα είναι να προσδιορίσει τους χρονικούς περιορισμούς για τις προσφυγές, ή να σχεδιάσει μία γλώσσα για τις προσφυγές που θα εκφράζει την γνώση των παιχτών για τον αλγόριθμο.
- Μετά ο σχεδιαστής θα δώσει τον μηχανισμό στους παίχτες για να πειραματιστούν και να μάθουν τον $\tilde{k}$.
- Τέλος όταν οι συμμετέχοντες είναι έτοιμοι τότε μπορεί να εκτελεστεί ο μηχανισμός δεύτερης ευκαιρίας.

3.2 Multicast Transmissions

Στην ενότητα αυτή θα μελετήσουμε μηχανισμούς που μας επιτρέπουν να μοιράσουμε το κόστος των multicast transmissions οι οποίες λαμβάνουν χώρα σε ένα δίκτυο επικοινωνίας. Η ανάλυση και τα αποτελέσματα που παρουσιάζονται βασίζονται στο άρθρο [FPS00] των J. Feigenbaum, C. Papadimitriou και S. Shenker.

3.2.1 Εισαγωγή

Ο κλασσικός τρόπος για να διατρέξουμε το Internet είναι το unicast routing. Με τη μέθοδο αυτή κάθε μήνυμα που αποστέλλεται από την πηγή (source) λαμβάνεται από ένα παραλήπτη. Έτσι για να στείλει η πηγή το ίδιο μήνυμα σε διαφορετικούς παραλήπτες στέλνει ξεχωριστά σε κάθε παραλήπτη αντίγραφα του μηνύματος. Αυτό έχει ως αποτέλεσμα από τα links (συνδέσμους) του δικτύου που είναι κοντά στην πηγή να διέρχονται πολλά όμοια μηνύματα και να επιβαρύνεται το δίκτυο με περιττή πληροφορία.

Ένας άλλος τρόπος για να διατρέξουμε το δίκτυο είναι το multicast routing. Με τη μέθοδο αυτή το δίκτυο μοντελοποιείται από ένα γράφο G στους κόμβους του οποίου θεωρούμε ότι βρίσκονται οι παραλήπτες του μηνύματος. Στη συνέχεια βρίσκουμε ένα δέντρο στο γράφο το οποίο συνδέει την πηγή με όλους τους παραλήπτες. Η πηγή στέλνει αρχικά ένα αντίγραφο του μηνύματος στα παιδία της βάση του δέντρου που έχει επιλεγεί. Στη συνέχεια οι κόμβοι αυτοί αντιγράφουν το μήνυμα και στέλνουν από ένα αντίγραφο στους γείτονες τους στο δέντρο, δηλαδή συμπεριφέρονται σαν πηγή για τους επόμενους παραλήπτες. Έτσι από κανένα link του δικτύου δεν διέρχονται πολλαπλά αντίγραφα του ίδιου μηνύματος.

Η μέθοδος του multicast routing διευκολύνει την μετάδοση πληροφοριών με μεγάλη ζήτηση, όπως μουσική και ταινίες, για τις οποίες ο αριθμός των παραληπτών θα είναι πολύ μεγάλος. Παρόλα αυτά όμως το κόστος μετάδοσης τέτοιας πληροφορίας σε μεγάλο αριθμό αποδεκτών μπορεί να παραμένει υψηλό. Είναι λοιπόν αναγκαίο να βρούμε ένα τρόπο κατανομής του κόστους μετάδοσης που να εξαρτάται από το πόσο μακριά από την πηγή βρίσκονται οι παραλήπτες του μηνύματος.

Ένας cost-sharing μηχανισμός κάνει ακριβώς αυτό, καθορίζει ποιοι χρήστες έλαβαν την multicast μετάδοση και πόσο πρέπει να χρεωθούν. Έτσι:

- Με $x_i \geq 0$ θα ορίζουμε το ποσό με το οποίο έχει χρεωθεί ο χρήστης i.
- Με σ_i θα δηλώνουμε αν έλαβε (ή όχι) την μετάδοση. Αν ο χρήστης έχει λάβει την μετάδοση τότε $\sigma_i = 1$, αν δεν την έλαβε τότε $\sigma_i = 0$.

Ας θεωρήσουμε την μετάδοση μίας ταινίας για παράδειγμα την μετάδοση κάποιας ταινίας στο δίκτυο. Η utility του χρήστη i για την ταινία είναι u_i και εκφράζει το πόσο θέλει ο χρήστης i να δεί την ταινία. Το όφελος του χρήστη, αφού είδε την ταινία και πλήρωσε γι' αυτή, είναι $w_i = \sigma_i u_i - x_i$. Τα διανύσματα σ και x για όλους τους χρήστες είναι συναρτήσεις του διανύσματος utility u όλων των χρηστών. Ένας cost-sharing μηχανισμός ορίζεται από τις συναρτήσεις x(u)

και σ(u). Θεωρούμε ότι οι τιμές των u_i δεν είναι γνωστές στο δίκτυο καθώς είναι προσωπικές πληροφορίες των χρηστών και όχι του δικτύου. Έτσι ο μηχανισμός αναγκαστικά θα βασίζεται στις τιμές που θα του δηλώσουν οι χρήστες, η ειλικρίνεια των οποίων δεν θεωρείται δεδομένη. Υποθέτουμε επίσης ότι οι χρήστες δεν συνεργάζονται μεταξύ τους. Οι χρήστες έχουν ως στόχο να μεγιστοποιήσουν το προσωπικό τους όφελος w_i.

Επειδή υπάρχει περίπτωση οι χρήστες να ψεύδονται, το δίκτυο πρέπει να χρησιμοποιήσει ένα *φιλαλήθη (strategyproof) cost-sharing μηχανισμό*. Όπως έχουμε ήδη αναφέρει σε προηγούμενες ενότητες ένας μηχανισμός, είναι *φιλαλήθης (strategyproof)* εάν η κυρίαρχη στρατηγική των χρηστών είναι η αποκάλυψη των αληθών τιμών των u_i. Δηλαδή οι χρήστες του δικτύου θα έχουν το μέγιστο κέρδος μόνο όταν είναι ειλικρινείς στο μηχανισμό. Σε ένα τέτοιο μηχανισμό ισχύει: $w_i(u) \geq w_i\left(u|^i \upsilon\right)$ για κάθε υ_i και για όλα τα i, δηλαδή το όφελος του χρήστη είναι το μέγιστο μόνο όταν αποκαλύψει την αληθή τιμή του u_i.

Σημείωση: Θα χρησιμοποιούμε τον συμβολισμό $\left(u|^i \upsilon_i\right)_j = u_j$ για $j \neq i$ που μας δίνει την συνιστώσα u_j του χρήστη j στο διάνυσμα u στο οποίο αντικαταστήσαμε το u_i με υ_i. Ομοίως $\left(u|^i \upsilon_i\right)_i = \upsilon_i$.

Με βάση τα παραπάνω, όπως θα δείξουμε στη συνέχεια, θα οδηγηθούμε σε δύο φυσικούς cost-sharing μηχανισμούς: τον Marginal Cost (MC) και τον Shapley Value (SH).

Όπως αναφέραμε προηγουμένως, ένας cost-sharing μηχανισμός είναι το ζεύγος των x(u) και σ(u). Ο υπολογισμός των x(u) και σ(u) όμως δεν είναι πάντα εύκολος επειδή τα u_i μεταδίδονται μέσα στο δίκτυο και τα αποτελέσματα των $x_i(u)$ και $\sigma_i(u)$ πρέπει να αποσταλούν στους χρήστες. Οι τιμές των u_i πρέπει να σταλούν από τους χρήστες στο μηχανισμό καθώς ο μηχανισμός δεν τις γνωρίζει. Επίσης οι τιμές των $x_i(u)$ και $\sigma_i(u)$ πρέπει να αποσταλούν στους χρήστες για να γνωρίζουν αν θα πάρουν το μήνυμα και πόσο πρέπει να πληρώσουν γι' την μετάδοση του. Αναγκαστικά συνεπώς θα υπάρξει κάποια ανταλλαγή μηνυμάτων μεταξύ των παραληπτών και του μηχανισμού. Γι' αυτό το λόγο, δοθέντος ενός cost-sharing μηχανισμού, είναι αναγκαίο να οριστεί ο κατανεμημένος cost-sharing αλγόριθμος ή πρωτόκολλο που θα υλοποιεί τον μηχανισμό. Δηλαδή πρέπει να καθοριστεί η διαδικασία που θα ακολουθηθεί για την ανταλλαγή των μηνυμάτων που αναφέραμε ώστε να μπορεί να γίνει ο υπολογισμός του κόστους μετάδοσης.

Στις επόμενες υποενότητες θα μελετήσουμε δύο συγκεκριμένους cost-sharing μηχανισμούς τον Marginal Cost (MC) και τον Shapley Value (SH). Θα μας απασχολήσει κυρίως η υπολογιστική πολυπλοκότητα των αλγορίθμων που υλοποιούν τους MC και SH μηχανισμούς. Θα δείξουμε ότι κάθε cost-sharing αλγόριθμος που υλοποιεί τον SH μηχανισμό, πρέπει να στείλει γραμμικό αριθμό μηνυμάτων σε καθ' ένα από τα γραμμικού αριθμού links, συνολικά

δευτεροβάθμιο αριθμό μηνυμάτων. Αντιθέτως για τον MC μηχανισμό υπάρχει αλγόριθμος που τον υλοποιεί, χρησιμοποιώντας μόνο δύο μηνύματα για κάθε link

3.2.2 Μοντέλο Δικτύου

Θεωρούμε ένα σύνολο χρηστών P, ένα σύνολο κόμβων N και ένα σύνολο L από links δύο κατευθύνσεων (bi-directional). Κάθε χρήστης βρίσκεται σε κάποιο κόμβο του δικτύου $\alpha \in N$. Δοθέντος ενός συνόλου $R \subseteq P$ από παραλήπτες και μιας πηγής α_s, εφαρμόζουμε τη μέθοδο του multicast routing και κατασκευάζουμε ένα δέντρο (multicast tree) $T(R) \subseteq L$ με ρίζα το α_s. Το $T(R)$ συνδέει το α_s με όλους τους κόμβους στους οποίους βρίσκονται οι παραλήπτες του R. Θεωρούμε για κάθε χρήστη i ότι υπάρχει ένα σταθερά καθορισμένο (από το multicast routing) μονοπάτι T(i) που τον συνδέει με την πηγή α_s. Έτσι για κάθε σύνολο R από παραλήπτες, το δέντρο της μετάδοσης είναι απλώς η ένωση όλων των σταθερών μονοπατιών, $T(R) = \bigcup_{i \in R} T(i)$.

Αυτή η προσέγγιση είναι σχετικά απλή στην υλοποίηση της και δίνει σταθερά multicast trees, αποκλείει όμως την χρήση πιο θεωρητικών κατασκευών όπως είναι τα Steiner trees. Επίσης αυτή η μορφή του multicast routing μπορεί να οδηγήσει σε δέντρα που δεν είναι βέλτιστα. Σε αυτή την κλάση δέντρων, το μονοπάτι μετάδοσης από την πηγή προς κάποιο συγκεκριμένο χρήστη εξαρτάται από το ποιοι άλλοι χρήστες είναι παρόντες, στα Steiner trees δεν συμβαίνει αυτό.

Το κόστος κάθε link $l \in L$ για την μετάδοση ενός μηνύματος μέσω αυτού θα είναι, $c(l) \geq 0$ και είναι γνωστό στους κόμβους των άκρων του. Το κόστος του δέντρου $T(R)$ για την προσέγγιση ενός συνόλου παραληπτών R είναι: $c(T(R))$ και το συνολικό όφελος (net worth) ορίζεται ως: $NW(R) = u_R - c(T(R))$ όπου $u_R = \sum_{i \in R} u_i$ και $c(T(R)) = \sum_{l \in T(R)} c(l)$.

Πρέπει να σημειώσουμε ότι το συνολικό όφελος δεν είναι το άθροισμα των ατομικών οφελών και δεν εξαρτάται τα μερίδια κόστους για τους παραλήπτες. Το συνολικό όφελος μετρά το συνολικό κέρδος από την εκτέλεση της multicast μετάδοσης. Τα μερίδια κόστους απλά μεταδίδονται από το δίκτυο στους παραλήπτες και δεν αλλάζουν το συνολικό όφελος από την μετάδοση. Το γεγονός ότι $T(R) = \bigcup_{i \in R} T(i)$ εγγυάται ότι το κόστος $c(T(R))$ είναι submodular συνάρτηση του συνόλου R που δεν είναι φθίνουσα. Κατά συνέπεια:
$c(T(R+i)) \geq c(T(R))$ και
$c(T(R_1)) + c(T(R_2)) \geq c(T(R_1 \cup R_2)) + c(T(R_1 \cap R_2))$, για όλα τα $R, R_1, R_2 \subseteq P$.

3.2.3 Cost-Sharing Μηχανισμοί

Οι cost-sharing μηχανισμοί ορίζονται από τις συναρτήσεις $x_i(u)$ και $\sigma_i(u)$. Για δοθέν cost-sharing μηχανισμό, ορίζουμε το σύνολο των παραληπτών

$R(u) = \{i \in P \mid \sigma_i(u) = 1\}$, δηλαδή το σύνολο των χρηστών που λαμβάνουν την μετάδοση για δοθέν διάνυσμα utility u. Με $W(u) \equiv NW(R(u))$ συμβολίζουμε το συνολικό όφελος που προκύπτει από τον μηχανισμό για διάνυσμα utility u. Θεωρούμε μόνο strategyproof cost-sharing μηχανισμούς , έτσι θα ισχύει :

$$w_i(u) \geq w_i\left(u \mid^i \upsilon_i\right) \quad \text{για όλα τα u, i και } \upsilon_i.$$

Δηλαδή οι χρήστες του δικτύου θα έχουν το μέγιστο κέρδος μόνο όταν είναι ειλικρινείς στο μηχανισμό. Μας ενδιαφέρουν οι φιλαλήθεις μηχανισμοί διότι είναι η μόνη μέθοδος που μας εγγυάται ότι οι χρήστες θα αποκαλύψουν τις αληθείς τιμές τους στο μηχανισμό.

Υπάρχουν όμως και κάποιοι οικονομικοί και υπολογιστικοί περιορισμοί που θα θέλαμε να ικανοποιούνται. Έτσι προκύπτουν οι επόμενες συνθήκες :

- **No Positive Transfers (NPT):** $x_i(u) \geq 0$. Δηλαδή θεωρούμε ότι οι χρήστες πάντα πληρώνουν για να συμμετέχουν στην μεταβίβαση. Δεν υπάρχουν παραλήπτες που να πληρώνονται για να δεχθούν την μετάδοση.
- **Voluntary Participation (VP) (Εθελοντική συμμετοχή):** $w_i(u) \geq 0$. Οι χρήστες έχουν το δικαίωμα να μην λάβουν την μετάδοση και να μην χρεωθούν. Τότε το ατομικό τους όφελος θα είναι 0. Έτσι έχουμε $x_i = 0$ όταν $\sigma_i = 0$.
- **Consumer Sovereignty (CS) (Ο καταναλωτής η ανώτατη αρχή):** Για όλα τα u, $\sigma\left(u \mid^i \upsilon_i\right) = 1$ για αρκετά μεγάλο υ_i. Ο cost-sharing αλγόριθμος δεν μπορεί αυθαίρετα να αποκλείσει κάποιο χρήστη. Το δίκτυο είναι υποχρεωμένο να επιτρέψει στους χρήστες να λάβουν την μετάδοση αν είναι πρόθυμοι να πληρώσουν σημαντικά μεγάλο κόστος.
- **Budget-balance (Ισορροπία προϋπολογισμού):** $\sum_{i \in P} x_i = c(T(R(u)))$. Με αυτή την συνθήκη εξασφαλίζουμε ότι τα χρήματα που προέρχονται από τους παραλήπτες καλύπτουν το κόστος της μετάδοσης.
- **Efficiency (Αποδοτικότητα):** $NW(R(u)) \geq NW(R)$ για όλα τα $R \subseteq P$. Με την συνθήκη αυτή απαιτούμε το σύνολο των παραληπτών να μεγιστοποιεί το συνολικό κέρδος του δικτύου. Το σύνολο που μεγιστοποιεί το NW(R) καλείται *αποδοτικό σύνολο*.

Σημείωση: Πρέπει να αναφέρουμε ότι δεν υπάρχει strategyproof cost-sharing μηχανισμός που να είναι ταυτόχρονα budget-balanced και αποδοτικός (efficient). Υπάρχει ένας μόνο strategyproof cost-sharing μηχανισμός που ικανοποιεί τις συνθήκες NPT και VP και είναι αποδοτικός, ο Marginal Cost (MC). Ο MC είναι ειδική περίπτωση της γενικής κλάσης των Vickrey-Clarke-Groves μηχανισμών.

Marginal Cost μηχανισμός (MC): Έστω $R^*(u)$ το μεγαλύτερο αποδοτικό σύνολο. Το σύνολο αυτό είναι καλά ορισμένο επειδή η συνάρτηση κόστους είναι submodular και μας εγγυάται ότι η ένωση δύο αποδοτικών συνόλων είναι

αποδοτικό σύνολο. Θέτουμε $\sigma_i = 1$ για όλα τα $i \in R^*(u)$ και $\sigma_i = 0$ για όλα τα $i \notin R^*(u)$. Έτσι $W(u) = NW\left(R^*(u)\right)$. Το μερίδιο κόστους για τον χρήστη i (δηλαδή το ποσό που πρέπει να πληρώσει στο μηχανισμό) δίνεται από τη σχέση

$$x_i = u_i \sigma_i(u) - \left(W(u) - W\left(u|^i 0\right)\right) \qquad (3.2.1)$$

Παρατηρήσεις: Η σχέση 3.2.1 έχει την εξής ερμηνεία. Αν ένας χρήστης δεν ανήκει στο $R^*(u)$ τότε δεν θα λάβει την μετάδοση και συνεπώς $x_i = 0$. Αν ο i λάβει την μετάδοση τότε $\sigma_i(u) = 1$. Σε αυτή την περίπτωση όπως προκύπτει από την 3.2.1 ο μηχανισμός δίνει στον i ένα κέρδος $w_i = W(u) - W\left(u|^i 0\right)$. Ο όρος $W(u) - W\left(u|^i 0\right)$ αντιπροσωπεύει την συνεισφορά του χρήστη i στο συνολικό όφελος , δηλαδή πόσο οφελείται ο μηχανισμός από την συμμετοχή του i. Είναι η διαφορά του συνολικού οφέλους όταν ο i συμμετέχει στο $R^*(u)$ από το συνολικό όφελος όταν αγνοήσουμε τον χρήστη i. Ο μηχανισμός MC είναι αποδοτικός αλλά δεν είναι budget-balance.

Budget-Balance Μηχανισμοί: Ενώ η απαίτηση για αποδοτικότητα μας έδωσε ένα μόνο φυσικό cost-sharing μηχανισμό, η συνθήκη budget-balance μας δίνει πολλούς πιθανούς μηχανισμούς. Η συνθήκη strategyproofness (η φιλαλήθεια είναι κυρίαρχη στρατηγική), δηλαδή ότι κανένας χρήστης δεν μπορεί να αυξήσει το κέρδος του με το να δηλώσει ψευδή utility, είναι πολύ ισχυρότερη και καθορίζει πλήρως τους πιθανούς cost-sharing μηχανισμούς.

Κάθε budget-balance μηχανισμός ορίζεται από μία συνάρτηση $f : 2^P \mapsto \mathbb{R}_{\geq 0}^{|P|}$ για την οποία:

$$\sum_i f_i(R) = c(T(R)) \text{ και } f_i(R + j) \leq f_i(R) \text{ για όλα τα } i, j \in P.$$

Η f είναι έχει ως πεδίο ορισμού όλα τα πιθανά υποσύνολα του P , δηλαδή όλα τα δυνατά υποσύνολα χρηστών $R \subseteq P$. Η f δίνει ως τιμές από τον χώρο $\mathbb{R}_{\geq 0}^{|P|}$ και είναι τα διανύσματα των πληρωμών (τα μερίδια κόστους) των παραληπτων. Επίσης η συνθήκη $\sum_i f_i(R) = c(T(R))$ αντιπροσωπεύει την απαίτηση budget-balance. Η συνθήκη $f_i(R + j) \leq f_i(R)$ ουσιαστικά εκφράζει ότι για μεγαλύτερο αριθμό παραληπτών θα πρέπει τα μερίδια κόστους στους παραλήπτες να είναι μικρότερα.

Μπορούμε να χρησιμοποιήσουμε την συνάρτηση f για να ορίσουμε με επαναληπτικό τρόπο τις x(u) και σ(u).

Αρχικά $\sigma_i^{(1)}(u) = 1$.

Στο κάθε βήμα k έχουμε: $R^{(k)}(u) = \left\{i \mid \sigma_i^{(k-1)}(u) = 1\right\}$, $x_i^{(k)}(u) = f_i\left(R^{(k)}(u)\right)$ και $\sigma_i^{(k)}(u) = 1$ εάν $u_i \geq x_i^{(k)}(u)$ και $\sigma_i^{(k)}(u) = 0$ εάν $u_i \leq x_i^{(k)}(u)$.

Τα σύνολα $R^{(k)}$ σχηματίζουν μία μονότονη ακολουθία, $R^{(k)} \subseteq R^{(k-1)}$

συγκλίνοντας μετά από πεπερασμένο αριθμό βημάτων σε κάποιο σύνολο $\hat{R}(u)$. Οι τιμές των x(u) και σ(u) στις οποίες καταλήγουμε, ορίζουν τον cost-sharing μηχανισμό και το σύνολο $\hat{R}(u)$ ορίζει το σύνολο των παραληπτών.

Σχόλια: Αρχικά το R περιέχει όλους τους χρήστες . Σε κάθε βήμα k το σύνολο των παραληπτών ορίζεται ως $R^{(k)}(u) = \{i \mid \sigma_i^{(k-1)}(u) = 1\}$ και οι πληρωμές $x_i^{(k)}(u)$ προκύπτουν από την f για το $R^{(k)}(u)$. Οι τιμές $\sigma_i^{(k)}(u)$ που θα καθορίσουν τους παραλήπτες του k+1 βήματος προκύπτουν συγκρίνοντας τις τιμές u_i και $x_i^{(k)}(u)$. Αν $u_i \geq x_i^{(k)}(u)$ τότε $\sigma_i^{(k)}(u) = 1$ δηλαδή ένα χρήστης θα παραμείνει παραλήπτης και για το k+1 βήμα αν έχει δηλώσει utility μεγαλύτερη από την πληρωμή που προέκυψε στο βήμα k. Αυτή η συνθήκη είναι απαραίτητη ώστε τελικά ο μηχανισμός να είναι budget-balance δηλαδή $\sum_i f_i(R) = c(T(R))$.

Μπορούμε τώρα να ορίσουμε τον μηχανισμό Shapley Value.

Shapley Value μηχανισμός (SH): Ο μηχανισμός αυτός χρησιμοποιεί ως συνάρτηση f την συνάρτηση Shapley Value. Η συνάρτηση αυτή για το cost-sharing πρόβλημα σε ένα δίκτυο ορίζεται από τον τύπο:

$$f_i(R) = \sum_{\tilde{R} \subseteq R - i} \frac{|\tilde{R}|!(|R| - |\tilde{R}| - 1)!}{|R|!} \left[c\left(T\left(\tilde{R} \cup i\right)\right) - c\left(T\left(\tilde{R}\right)\right) \right]$$

Η γενική ερμηνεία του παραπάνω τύπου είναι ότι το κόστος ενός link l μοιράζεται εξίσου σε όλους τους παραλήπτες που βρίσκονται κάτω από το l.

Σημείωση: Κανένας από τους budget-balanced μηχανισμούς δεν μεγιστοποιεί το ολικό όφελος, δηλαδή οι μηχανισμοί αυτοί δεν είναι αποδοτικοί.. Παρόλα αυτά ο Shapely Value ελαχιστοποιεί την απώλεια κέρδους για την χειρότερη περίπτωση. Η ποσότητα $Max_u [NW(R^*(u) - NW(R(u)))]$ που προκύπτει από τον Shapely Value, είναι σημαντικά μικρότερη από τις αντίστοιχες ποσότητες οποιουδήποτε άλλου μηχανισμού της κλάσης αυτής. Αν επιμείνουμε στην συνθήκη budget-balance, τότε είναι λογικό να επιλέξουμε εκείνο το μηχανισμό που ελαχιστοποιεί την απώλεια σε αποδοτικότητα. Έτσι ο Shapely Value είναι η λογικότερη επιλογή από την κλάση των budget-balance cost-sharing μηχανισμών.

Για να είναι ένας cost-sharing μηχανισμός υλοποιήσιμος, πρέπει η επικοινωνία και οι τοπικοί υπολογισμοί που απαιτεί να είναι σχετικά περιορισμένοι. Η κατανομή του κόστους στους χρήστες δεν είναι ο λόγος ύπαρξης του δικτύου, απλά υποστηρίζει με ένα οικονομικά βιώσιμο τρόπο τον κύριο στόχο που είναι η μετάδοση του υλικού στους παραλήπτες. Έτσι είναι απαραίτητη η ανάλυση της πολυπλοκότητας των αλγορίθμων που υλοποιούν τον καταμερισμό του κόστους στους χρήστες.

Ένα στιγμιότυπο του cost-sharing προβλήματος έχει μέγεθος $n+p+m$ όπου $n=|T(P)|$, $p=|P|$ και m είναι το συνολικό μέγεθος της εισόδου $\{c(l)\}_{l\in L} \cup \{u_i\}_{i\in P}$. Το ιδανικό είναι ο συνολικός αριθμός μηνυμάτων που ανταλλάσσει ο cost-sharing αλγόριθμος με τους χρήστες να είναι O(n), και ο μέγιστος αριθμός μηνυμάτων που στέλνει στο link l, για όλα τα $l \in L$, να είναι O(1).

Στις επόμενες υποενότητες μελετούμε κυρίως το κόστος επικοινωνίας των αλγορίθμων παρά το τοπικό κόστος υπολογισμού. Υποθέτουμε ότι τα μηνύματα πάντα φθάνουν στους αποδέκτες τους μέσα σε μικρό χρονικό διάστημα. Επίσης θεωρούμε ότι τα μηνύματα έχουν λογικό μέγεθος. Η τελευταία υπόθεση είναι σημαντική. Για παράδειγμα δείχνουμε μία ανεπιτυχή υλοποίηση ενός μηχανισμού πολυωνιμικού χρόνου. Κάθε κόμβος αφού λάβει ένα μήνυμα από κάθε παιδί του, συγκεντρώνει τα μηνύματα που έλαβε, όλες τις utility των παιχτών που βρίσκονται σε αυτόν καθώς και το κόστος του link που τον συνδέει με τον πατέρα του και στέλνει το αποτέλεσμα στον πατέρα του. Όταν η ρίζα α_s του T(P) λάβει όλες τις utilities και όλα τα κόστη, τότε υπολογίζει τις σ και x και τις στέλνει στα παιδιά της. Αυτός ο αλγόριθμος χρειάζεται O(1) μηνύματα για κάθε link, αλλά το μέγιστο μέγεθος ενός μηνύματος μπορεί να είναι Ω(m), που είναι πολύ μεγάλο.

3.2.4 Ο Marginal Cost Μηχανισμός

Θα εξετάσουμε τώρα το κόστος επικοινωνίας του αλγορίθμου που υλοποιεί τον μηχανισμό Marginal Cost. Θα αποδείξουμε ότι υπάρχει αλγόριθμος που υλοποιεί τον μηχανισμό και είναι βέλτιστος ως προς τον αριθμό των μηνυμάτων που ανταλλάσσει.

Θεώρημα 3.2.1 Ο MC μηχανισμός χρειάζεται ακριβώς δύο μηνύματα για κάθε link. Υπάρχει αλγόριθμος που υπολογίζει τα μερίδια κόστους εκτελώντας μία διάσχιση του T(P) από τον πυθμένα προς τα πάνω (buttom-up), ακολουθούμενη από μια διάσχιση από την κορυφή προς τα κάτω (top-down). Ο αλγόριθμος αυτός είναι βέλτιστος σε σχέση με τον αριθμό των μηνυμάτων που ανταλλάσει.

Απόδειξη:

Συμβολισμοί: Θα συμβολίζουμε με:

- $\mathbf{u^{\alpha}}$ το άθροισμα των utilities των παιχτών που βρίσκονται στον κόμβο α.
- $\mathbf{c^{\alpha}}$ το κόστος του link ακριβώς πάνω από τον κόμβο α στο δέντρο T(P).
- **Ch(a)** τους κόμβους που είναι παιδιά του α στο T(P).
- **res(α)** το σύνολο των χρηστών στον κόμβο α.
- $\mathbf{T^{\alpha}(P)}$ την ένωση του υποδέντρου με ρίζα τον κόμβο α και του link που συνδέει τον α με τον πατέρα του p(α).

Μπορούμε να υπολογίσουμε το όφελος $W^{\alpha}(u)$ του υποδέντρου $T^{\alpha}(P)$ με ρίζα τον κόμβο α, που ισούται με την διαφορά της συνολικής utility μείον το κόστος με :

$$W^{\alpha}(u) = u^{\alpha} + \left(\sum_{\beta \in Ch(\alpha) | W^{\beta}(u) \geq 0} W^{\beta}(u) \right) - c^{\alpha}.$$

Παρατήρηση: Στον προηγούμενο τύπο το όφελος $W^{\alpha}(u)$ το εκφράζουμε με μία αναδρομική σχέση. Είναι το άθροισμα του όρου $u^{\alpha} - c^{\alpha}$ για το link πάνω από τον α συν το άθροισμα των οφελών $W^{\beta}(u)$ των υποδέντρων $T^{\beta}(P)$ με $W^{\beta}(u) \geq 0$, όπου β είναι τα παιδιά του κόμβου α.

Bottom-Up Διάσχιση: Οι τιμές $W^{\alpha}(u)$ υπολογίζονται με την bottom-up διάσχιση που περιγράφεται στην εικόνα 3.2.1. Θα έχουμε $\sigma(i) = 1$, δηλαδή ο i θα λάβει την μετάδοση, εάν $W^{\alpha}(u) \geq 0$ για όλους τους κόμβους α στο μονοπάτι από τον χρήστη i προς την ρίζα.

Εικόνα 3.2.1
Bottom-Up Διάσχιση: Υπολογισμός τιμών οφέλους

Για τον κόμβο $\alpha \in V(P)$
  Αφού ληφθεί το μήνυμα A^{β} από κάθε παιδί του $\beta \in Ch(\alpha)$
    $W^{\alpha} \leftarrow u^{\alpha} + \left(\sum_{\beta \in Ch(\alpha)} A^{\beta} \right) - c^{\alpha}$
    Εάν $W^{\alpha} \geq 0$ τότε
    {
      $\sigma_i \leftarrow 1$ για όλα τα $i \in res(\alpha)$
      Στείλε το W^{α} στον πατέρα $p(\alpha)$
    }
    Αλλιώς
    {
      $\sigma_i \leftarrow 0$ για όλα τα $i \in res(\alpha)$
      Στείλε 0 στον πατέρα $p(\alpha)$
    }

Ανάλυση: Στη bottom-up διάσχιση κάθε κόμβος α λαμβάνει ένα μήνυμα A^{β} από κάθε παιδί του $\beta \in Ch(\alpha)$. Το A^{β} είναι το όφελος $W^{\beta}(u)$ του υποδέντρου $T^{\beta}(P)$ όπου β είναι τα παιδιά του κόμβου α. Στη συνέχεια υπολογίζεται το όφελος W^{α} του υποδέντρου $T^{\alpha}(P)$ δηλαδή ο όρος $u^{\alpha} + \left(\sum_{\beta \in Ch(\alpha)} A^{\beta} \right) - c^{\alpha}$. Αν $W^{\alpha} \geq 0$ τότε όλοι οι χρήστες που κατοικούν στον α θα λάβουν την μετάδοση, δηλαδή $\sigma_i = 1$, για όλα τα $i \in res(\alpha)$. Υπενθυμίζουμε ότι τα $\sigma_i(u)$ είναι τα bits που υποδεικνύουν εάν ο χρήστης i ανήκει ή όχι στο αποδοτικό σύνολο $R^*(u)$.

Αν $W^{\alpha} < 0$ τότε οι χρήστες του α δεν θα λάβουν την μετάδοση. Η βασική ιδέα είναι ότι τελικά θέλουμε να βρούμε το μεγαλύτερο αποδοτικό σύνολο παραληπτών $R^*(u)$. Έτσι πρέπει να αποκλειστούν υποδέντρα με αρνητικό όφελος. Η bottom-up διάσχιση τελικά εφαρμόζεται σε όλους τους κόμβους του T(P).

Πρέπει να παρατηρήσουμε ότι κάποιες από τις τιμές $\sigma_i = 1$ που ορίζονται με την bottom-up διάσχιση μπορεί τελικά να είναι λανθασμένες. Αυτό συμβαίνει διότι οι τιμές των σ_i δεν ενημερώνονται εκ' νέου με βάση τα αποτελέσματα των επόμενων υπολογισμών. Για παράδειγμα έστω κόμβος α για τον οποίο προέκυψε $W^{\alpha} \geq 0$, τότε $\sigma_i = 1$, για όλα τα $i \in res(\alpha)$. Ο επόμενος υπολογισμός της bottom-up διάσχισης θα γίνει στον κόμβο p(α), τον πατέρα του α. Αν $W^{p(\alpha)}(u) < 0$ οι χρήστες του p(α) θα αποκλειστούν αλλά τότε θα αποκλειστούν και όλοι οι κόμβοι του υποδέντρου του T(P) με ρίζα τον α. Συνεπώς οι τιμές $\sigma_i = 1$ για τα $i \in res(\alpha)$ ήταν λανθασμένες. Όμως για οποιοδήποτε κόμβο οι τιμές όμως $\sigma_i = 0$ θα είναι πάντα σωστές. Οι τιμές $\sigma_i = 1$ που υπολογίζονται με την bottom-up διάσχιση θα θεωρούνται προσωρινές.

Αφού υπολογιστούν τα $W^{\alpha}(u)$, οι ακριβείς τιμές των $\sigma_i(u)$ θα παραχθούν με την top-down διάσχιση που θα περιγράψουμε στη συνέχεια.
Το μερίδιο κόστους $x_i(u)$ του χρήστη i υπολογίζεται από την εξίσωση

$$x_i = u_i \sigma_i(u) - \left(W(u) - W\left(u|^i 0\right)\right).$$

Για να υπολογιστεί το x_i πρέπει να υπολογιστεί το $W\left(u|^i 0\right)$ που είναι το όφελος του δικτύου όταν παραλειφθεί ο i. Ας δούμε πως μπορούμε να κάνουμε τον υπολογισμό αυτό. Για κάθε κόμβο α και χρήστη $i \in R^*(u)$ στον α, έστω $y_i(u)$ το μικρότερο $W^{\beta}(u)$ των κόμβων β στο μονοπάτι από το α προς την ρίζα (μπορεί το μικρότερο όφελος να εμφανίζεται στον α). Τότε έχουμε δύο περιπτώσεις:

1. Εάν $u_i \leq y_i(u)$, τότε η δομή του δέντρου για το $R^*\left(u|^i 0\right)$ είναι ίδια με του δέντρου για το $R^*(u)$. Έτσι η διαφορά $W(u) - W\left(u|^i 0\right)$ είναι ίση με u_i διότι όταν παραλείπουμε τον χρήστη i το $W^{\alpha}(u)$ μειώνεται κατά u_i. Υπενθυμίζουμε ότι $W(u) \equiv NW(R(u))$ και $NW(R) = u_R - c(T(R))$. Κατά συνέπεια ο χρήστης i πρέπει να πληρώσει $x_i(u) = u_i - \left(W(u) - W\left(u|^i 0\right)\right) = 0$.

2. Εάν $u_i > y_i(u)$, παραλείποντας τον χρήστη i καταλήγουμε στην αφαίρεση από το $R^*(u)$ του υποδέντρου με συνολικό όφελος $y_i(u)$. Έτσι ο χρήστης i πρέπει να πληρώσει $x_i(u) = u_i - y_i(u)$. Πιο συγκεκριμένα θα υπάρχει κάποιος πρόγονος α΄ του α, χαμηλότερα από το β, έτσι ώστε

παραλείποντας τον χρήστη i το $W^{\alpha'}(u)$ να γίνεται αρνητικό και το $T^{\alpha'}(P)$ να παραλείπεται από το $R^*(u)$. Παραλείποντας το $T^{\alpha'}(P)$ μπορεί να καταλήξουμε σε αρνητικό όφελος για κάποιο πρόγονο α΄΄ του α΄ και τελικά να παραλειφθεί το $T^{\alpha''}(P)$ κ.τ.λ.π. Αυτή η αλυσιδωτή αντίδραση σταματά μετά την απομάκρυνση του υποδέντρου ελάχιστου-οφέλους $T^{\beta}(P)$.

Top-Down Διάσχιση: Με την top-down διάσχιση υπολογίζουμε τις τιμές των σ_i και x_i. Στη top-down διάσχιση θεωρούμε ότι κάθε κόμβος α γνωρίζει τα στοιχεία από την bottom-up διάσχιση που προηγήθηκε, δηλαδή τα μηνύματα που έλαβε από τα παιδιά του, το μήνυμα που έστειλε στον πατέρα του και τις τιμές σ_i για τους χρήστες που βρίσκονται σε αυτόν.

Εικόνα 3.2.2
Top-Down Διάσχιση: Υπολογισμός συμμετοχής και μεριδίων κόστους

Αρχικοποίηση: Η ρίζα α_s στέλνει το W^{α_s} σε κάθε παιδί της.
Για κάθε $\alpha \in V(P) - \{\alpha_s\}$
Αφού ληφθεί το μήνυμα A από τον πατέρα p(α)
// Περίπτωση 1: $T^{\alpha}(P) \bigcap T(R^*(u)) = \varnothing$
// Θέσε τα ορθά σ_i στον α και διέδωσε καμία-συμμετοχή προς τα κάτω.
Εάν $\sigma_i = 0$, για όλα τα $i \in res(\alpha)$, ή $A < 0$ τότε
{
$x_i \leftarrow 0$ και $\sigma_i \leftarrow 0$ για όλα τα $i \in res(\alpha)$
στείλε -1 στον β για όλα τα $\beta \in Ch(\alpha)$
}
// Περίπτωση 2: $T^{\alpha}(P) \bigcap T(R^*(u)) \neq \varnothing$
//Υπολόγισε τα μερίδια κόστους και διέδωσε την ελάχιστη τιμή οφέλους προς τα κάτω.
Αλλιώς
{
$A \leftarrow \min(A, W^{\alpha})$
Για κάθε $i \in res(\alpha)$
$u_i \le A$, τότε $x_i \leftarrow 0$, αλλιώς $x_i \leftarrow u_i - A$
Για κάθε $\beta \in Ch(\alpha)$
Στείλε το A στο β
}

Μερικές από τις τιμές σ_i μπορεί εσφαλμένα να έχουν τεθεί 1 αλλά αυτό είναι προσωρινό και θα διορθωθούν από την top-down διάσχιση. Κατά την διάσχιση

αυτή πρέπει να μεταφερθεί αρκετή πληροφορία ώστε να μπορούν οι κόμβοι να υπολογίσουν τα μερίδια κόστους χρησιμοποιώντας την εξίσωση 3.2.1 και να διορθώσουν τις λανθασμένες τιμές των σ_i.

Ανάλυση: Κάθε κόμβος α λαμβάνει από τον πατέρα του p(α) ένα μήνυμα A που θα δούμε στη συνέχεια ποιες είναι οι τιμές του. Το A είναι απαραίτητο για τον υπολογισμό των x_i και σ_i.
Για τον κόμβο α υπάρχουν δύο περιπτώσεις.

- Περίπτωση 1: $T^{\alpha}(P) \bigcap T(R^{*}(u)) = \varnothing$, δηλαδή κανένας κόμβος του $T^{\alpha}(P)$, (της ένωσης του υποδέντρου με ρίζα τον κόμβο α και του link που συνδέει τον α με τον πατέρα του p(α)) δεν θα λάβει την μετάδοση. Η περίπτωση αυτή εμφανίζεται όταν $\sigma_i = 0$, για όλα τα $i \in res(\alpha)$, ή όταν $A < 0$, όπου A είναι η τιμή του $W^{p(\alpha)}(u)$. Τότε για όλους τους χρήστες στον α τα σ_i και x_i είναι μηδέν αφού δεν μπορούν να συμμετέχουν στη μετάδοση. Συνεπώς η μόνη εργασία για τον α είναι να θέσει τις κατάλληλες τιμές στα σ_i και να μεταδώσει ότι δεν υπάρχει προς τα κάτω άλλη συμμετοχή. Αν Γι' αυτό και το μήνυμα που στέλνει ο α στα παιδιά του είναι –1. Το μήνυμα είναι –1 για να αποκλείσει και τα παιδιά του από την μετάδοση.

- Περίπτωση 2: $T^{\alpha}(P) \bigcap T(R^{*}(u)) \neq \varnothing$ τότε υπάρχουν κόμβοι του $T^{\alpha}(P)$ που συμμετέχουν στο δέντρο μετάδοσης. Τώρα ο α πρέπει να υπολογίσει τα μερίδια κόστους για τους χρήστες που μένουν σε αυτόν με βάση την σχέση $x_i = u_i \sigma_i(u) - \left(W(u) - W\left(u \mid^{i} 0\right)\right)$. Όπως αναφέραμε προηγουμένως για να υπολογίσει γρήγορα την σχέση αυτή πρέπει να βρεί το $y_i(u)$ που είναι το μικρότερο $W^{\beta}(u)$ των κόμβων β στο μονοπάτι από το α προς την ρίζα. Η τιμή του $y_i(u)$ είναι χρήσιμη και για τα παιδιά του άρα θα πρέπει να τους την μεταδόσει. Συνεπώς ο α υπολογίζει τα μερίδια κόστους με βάση το A και ορίζει ως μήνυμα για τα παιδιά του την ελάχιστη τιμή των A και W^{α}.

Στην πράξη πρέπει να περάσουν δύο μηνύματα από κάθε link του T(P) για υπολογιστεί σωστά ο καταμερισμός του κόστους, συνεπώς υπάρχει αλγόριθμος που υλοποιεί τον MC μηχανισμό και είναι βέλτιστος ως προς τον αριθμό των μηνυμάτων που ανταλλάσσει. ■

Το μοντέλο που ορίσαμε για την διάσχιση ενός δικτύου, απαιτεί όλα τα δέντρα να είναι υποδέντρα του δέντρου που συνδέει την πηγή με όλους τους χρήστες. Το μοντέλο αυτό είναι απλό και σταθερό σε βελτιστοποίηση. Υπάρχουν όμως και άλλα περιβάλλοντα όπου η βελτιστοποίηση είναι ο κύριος στόχος. Ένα παράδειγμα είναι η εγκατάσταση γραμμών επικοινωνίας για να συνδεθούν κάποια ινστιτούτα μεταξύ τους. Μετά από μία εγκατάσταση, η κατάσταση είναι σταθερή και κατά συνέπεια το κύριο πρόβλημα δεν είναι η σταθερότητα, αλλά η

ελαχιστοποίηση του κόστους. Μέσα σε αυτό το πλαίσιο δεν υπάρχει η έννοια του παίχτη που βρίσκεται σε κάποιο κόμβο. Κάθε ινστιτούτο αντιστοιχεί σε ένα κόμβο και έχει μία τιμή (utility), ο κόμβος είτε θα συνδεθεί με το δέντρο διανομής, είτε όχι.

Μέσα σε αυτό το πλαίσιο θα μελετήσουμε μία γενίκευση του MC μηχανισμού. Το σύνολο N των κόμβων και το σύνολο L των links σχηματίζουν ένα κατευθυνόμενο γράφο. Η πηγή s είναι στοιχείο του N. Κάθε υποδέντρο T του (N, L) με ρίζα το s είναι δέντρο μετάδοσης διότι μέσα σε αυτό το πλαίσιο ταυτίζουμε τους κόμβους με τους χρήστες. Το T ορίζει το σύνολο V(T) των παραληπτών και το συνολικό όφελος είναι η διαφορά του αθροίσματος των utility των κόμβων του T και του αθροίσματος από τα κόστη των link του T. Το ερώτημα που προκύπτει είναι εάν είναι υπολογιστικά εφικτός ο εντοπισμός του δέντρου μετάδοσης με το μέγιστο όφελος.

Πιο αυστηρά, έστω T (R) το σύνολο όλων των δέντρων που συνδέουν την πηγή με το σύνολο R των αποδεκτών. Δοθέντος του συνόλου R επιλέγουμε το δέντρο $T \in T(R)$ που ελαχιστοποιεί το c(T), ορίζουμε $C(R) = \min_{T \in T(R)} c(T)$. Το αποδοτικό σύνολο $R^* \subseteq N$ είναι εκείνο που μεγιστοποιεί το $u_R - C(R)$. Τώρα το μεγαλύτερο αποδοτικό σύνολο δεν είναι πλέον μοναδικό, διότι η συνάρτηση κόστους C(R) δεν είναι submodular.

Έστω $W(N, L, c, u, r) = \max_{R \subseteq N - \{r\}} [u_R - C(R)]$, όπου r είναι η ρίζα , η βέλτιστη αποδοτικότητα. Για δοθείσα συλλογή αποδοτικών συνόλων $R^*(u)$, η εξίσωση $x_i = u_i \sigma_i(u) - \left(W(u) - W\left(u|^i 0\right)\right)$ ορίζει τα μερίδια κόστους για τον MC strategyproof μηχανισμό ικανοποιώντας τις συνθήκες NPT, VP και CS. Ο υπολογισμός του W( ) είναι NP-hard διότι είναι ισοδύναμος με την net-worth maximization εκδοχή του Prize Collecting Steiner Tree προβλήματος. Θα δείξουμε ότι η προσέγγιση του W( ) για οποιοδήποτε σταθερό παράγοντα είναι NP-hard ακόμα και για φραγμένου βαθμού γραφήματα.

Σημείωση: Θα λέμε ότι ο αλγόριθμος A *προσεγγίζει το f εντός λόγου* ε $(0 < \varepsilon < 1)$ αν για όλα τα x, $\varepsilon \le A(x)/f(x) \le 1/\varepsilon$. Δηλαδή για όλα τα στιγμιότυπα x του προβλήματος ισχύει $\varepsilon \cdot f(x) \le A(x) \le f(x)/\varepsilon$. Το f είναι η συνάρτηση που περιγράφει το πρόβλημα και δίνει τις βέλτιστες τιμές.

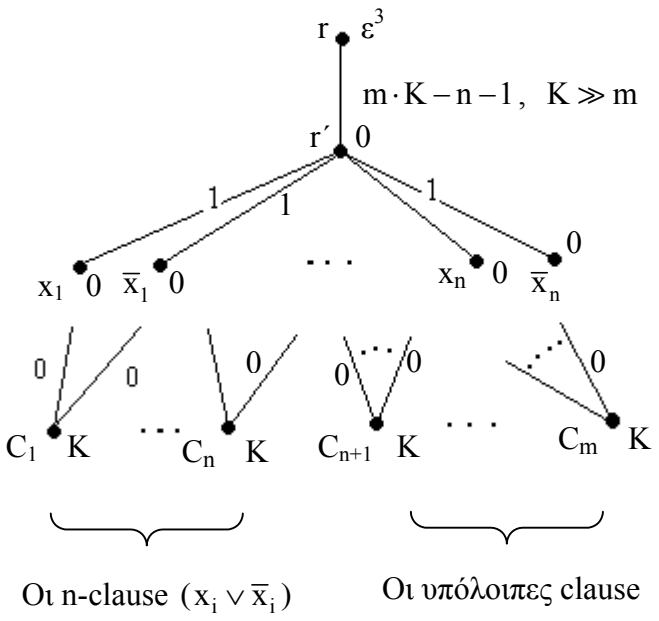
Θεώρημα 3.2.2 Έστω γράφημα $G = (N, L)$, $r \in N$ ένας κόμβος-ρίζα, $c : L \to \mathbb{R}^{\ge 0}$ η συνάρτηση κόστους των links και $u : N \to \mathbb{R}^{\ge 0}$ η συνάρτηση utility των κόμβων. Για $R \subseteq N - \{r\}$, έστω C(R) είναι το κόστος του ελαχίστου κόστους δέντρου το οποίο περιέχει τα r και R. Τότε για κάθε σταθερά ε, με $0 < \varepsilon < 1$, η προσέγγιση εντός λόγου ε της συνάρτησης

$$W(N, L, c, u, r) = \max_{R \subseteq N - \{r\}} [u_R - C(R)]$$

είναι πρόβλημα NP-hard. Επιπλέον το πρόβλημα παραμένει NP-hard ακόμα και αν το (N,L) είναι γράφημα φραγμένου βαθμού.

Απόδειξη: Θα εφαρμόσουμε την ακόλουθη αναγωγή. Έστω $(x_1,\ldots x_n, C_1,\ldots C_m)$ ένα στιγμιότυπο του SAT. Χωρίς βλάβη της γενικότητας θεωρούμε ότι ο τύπος περιέχει μία clause (φράση) $(x_i \vee \overline{x}_i)$ για όλα τα $1 \le i \le n$.
Το ισοδύναμο γράφημα έχει τέσσερα επίπεδα (εικόνα 3.2.3):

- Το **πρώτο επίπεδο** περιέχει την ρίζα r, με utility ε^3.
- Το **δεύτερο επίπεδο** περιέχει τον κόμβο r΄, με utility μηδέν, που συνδέεται με τον r με link κόστους $m \cdot K - n - 1$, όπου $K \gg m$.
- Το **τρίτο επίπεδο** περιέχει 2n κόμβους, ένα για κάθε x_i και $\overline{x}_i$,. Όλοι έχουν utility μηδέν και όλοι συνδέονται με τον r΄ με links μοναδιαίου κόστους.
- Τέλος το **τέταρτο επίπεδο** περιέχει m κόμβους, έναν για κάθε clause. Κάθε κόμβος έχει utility K. Καθένας από αυτούς συνδέεται με τους κόμβους του $3^{ου}$ επιπέδου που αντιστοιχούν στα λεκτικά (literals) που περιέχει, μέσω μηδενικού κόστους links.



Εικόνα 3.2.3: Ο Ισοδύναμος γράφος των τεσσάρων επιπέδων.

Κάθε γράφημα αυτής της μορφής έχει μια τετριμμένη λύση με όφελος ε^3 που περιέχει μόνο την ρίζα r. Οποιαδήποτε λύση με όφελος μεγαλύτερο του ε^3 πρέπει οπωσδήποτε να περιλαμβάνει το link (r, r'). Αυτό όμως αναγκαστικά οδηγεί στο να περιληφθούν όλοι οι κόμβοι του $4^{ου}$ επιπέδου ώστε να μπορούμε να έχουμε όφελος μεγαλύτερο του ε^3. Όμως θα πρέπει να πάρουμε το πολύ n links από το r΄

προς τους κόμβους του $3^{ου}$ επιπέδου διότι με $n+1$ links του $3^{ου}$ επιπέδου έχουμε όφελος $W = u_R - C(R) = (\varepsilon^2 + m \cdot K) - ((m \cdot K - n - 1) + n + 1) = \varepsilon^2$.

Η λύση θα πρέπει να περιέχει είτε ένα από τους x_i-κόμβους είτε ένα από τους $\overline{x}_i$-κόμβους, για κάθε i, ώστε να φθάσουμε στον κόμβο του $4^{ου}$ επιπέδου που αντιστοιχεί στην clause $(x_i \vee \overline{x}_i)$. Συνεπώς μία λύση που δίνει όφελος μεγαλύτερο του ε^3 θα αντιστοιχεί σε ανάθεση που ικανοποιεί τον τύπο. Για την ακρίβεια τέτοια λύση θα δίνει όφελος ακριβώς $1+\varepsilon^3$. Μη ικανοποιήσιμοι τύποι αντιστοιχούν σε γράφους με μέγιστο όφελος ε^3. Γι' αυτούς ένας ε-προσεγγιστικός αλγόριθμος για το W( ) θα δίνει τιμές μικρότερες ή ίσες του ε^2. Ικανοποιήσιμοι τύποι αντιστοιχούν σε γράφους με μέγιστο όφελος $1+\varepsilon^3$, για τους οποίους ένας ε-προσεγγιστικός αλγόριθμος θα δίνει τιμές μεγαλύτερες ή ίσες του $\varepsilon(1+\varepsilon^3) > \varepsilon$. Κατά συνέπεια η ε-προσέγγιση του W( ) είναι NP-hard.

Για να δείξουμε ότι το πρόβλημα παραμένει NP-hard και για φραγμένου βαθμού γράφους θα τροποποιήσουμε την αναγωγή. Πρώτα αντικαθιστούμε τα 2n links που συνδέουν τον r΄με τους κόμβους-λεκτικών του $3^{ου}$ επιπέδου, με ένα δυαδικό δέντρο του οποίου η ρίζα είναι ο r΄και τα φύλλα του είναι οι 2n κόμβοι των λεκτικών. Όλοι οι κόμβοι σε αυτό το δυαδικό δέντρο έχουν utility μηδέν και όλα τα links έχουν κόστος μηδέν, εκτός από τα 2n links που συνδέουν τους κόμβους-λεκτικών με τους γονείς τους. Κάθε ένα από αυτά έχει κόστος 1. Μετά θεωρούμε, χωρίς βλάβη της γενικότητας, ότι κάθε clause C_j είναι ή διάζευξη τριών λεκτικών ή της μορφής $(x_i \vee \overline{x}_i)$. Για κάθε λεκτικό y χρησιμοποιούμε ένα δυαδικό δέντρο που συνδέει τους κόμβους λεκτικών που αντιστοιχούν στο y με όλους τους κόμβους που αντιστοιχούν στις clause που εμφανίζεται το y. Όλα τα links σε αυτά τα δέντρα έχουν κόστος μηδέν και όλοι οι κόμβοι έχουν utility μηδέν εκτός από τους κόμβους των clause που έχουν utility K. ■

3.2.5 Ο Μηχανισμός Sharpley Value

Ο καταμερισμός του κόστους σε ένα SH μηχανισμό μπορεί να υπολογιστεί από ένα brute-force αλγόριθμο που στην χειρότερη περίπτωση ανταλλάσσει $\Theta(n \cdot p)$ μηνύματα, με τουλάχιστον p μηνύματα πάνω σε συγκεκριμένα links, όπου p είναι ο συνολικός αριθμός των χρηστών. Οι J. Feigenbaum, C. Papadimitriou και S. Shenker στο άρθρο τους [FPS00] κάνουν την εικασία ότι αυτή η πολυπλοκότητα είναι η καλύτερη δυνατή. Αρχικά θα παρουσιάσουμε τον brute-force αλγόριθμο.

Η απλούστερη περίπτωση του SH cost-share προβλήματος είναι εκείνη όπου όλα τα u_i είναι αρκετά μεγάλα και έτσι όλοι οι χρήστες του P λαμβάνουν την μετάδοση. Για παράδειγμα θα αρκούσε αν $u_i > c(T(P))$ για όλα τα i. Για την περίπτωση αυτή τα μερίδια κόστους του SH μπορεί να υπολογιστούν ως εξής.

- Αρχικά εκτελούμε μία bottom-up διάσχιση στο δέντρο $T(P)$ η οποία καθορίζει για κάθε κόμβο α, τον αριθμό των χρηστών p_α στο υποδέντρο με ρίζα το α.
- Μετά εκτελούμε μία top-down διάσχιση κατά την οποία η ρίζα αρχικοποιεί

στέλνοντας τον αριθμό $\mathrm{md} = 0$ στα παιδιά της. Ο κόμβος α αφού λάβει το μήνυμα md, υπολογίζει το $\mathrm{md}' = \left(\frac{c(l)}{p_\alpha} \right) + \mathrm{md}$, όπου l είναι το link μεταξύ του α και του πατέρα του. Μετά δίνει το μερίδιο κόστους md΄σε κάθε χρήστη που βρίσκεται σε αυτόν και στέλνει το md΄σε κάθε παιδί του. Έτσι κάθε χρήστης καταλήγει να πληρώνει ένα κλάσμα του κόστους από κάθε link του μονοπατιού που τον συνδέει με την πηγή. Το κλάσμα του κόστους για κάθε link εξαρτάται από τον αριθμό των χρηστών που μοιράζονται το link.

Για την γενική περίπτωση, η εφαρμογή του brute-force αλγορίθμου θα γίνει με ένα επαναληπτικό τρόπο.

- Αρχικά ξεκινούμε με $R = P$ και υπολογίζουμε τα μερίδια κόστους όπως στην προηγούμενη περίπτωση. Για την γενική περίπτωση δεν μπορούμε να υποθέσουμε ότι $u_i \geq \mathrm{md}'$ για όλα τα i. Συνεπώς υπάρχει περίπτωση κάποιοι από τους χρήστες να μην επιθυμούν να λάβουν την μετάδοση.
- Έτσι μετά από μία επανάληψη ενημερώνουμε το R αφαιρώντας όλους τους χρήστες i για τους οποίους $u_i < \mathrm{md}'$ και επαναλαμβάνουμε.
- Ο αλγόριθμός τερματίζει όταν δεν υπάρχουν χρήστες που πρέπει να αφαιρεθούν.

Στη χειρότερη περίπτωση θα πρέπει να αφαιρεθούν όλοι οι χρήστες με ένα χρήστη να αφαιρείται σε κάθε επανάληψη. Τότε θα έχουμε p επαναλήψεις με $\Omega(n \cdot p)$ συνολικό αριθμό μηνυμάτων.

Θεώρημα 3.2.3 Ο brute-force αλγόριθμος υλοποιεί τον SH μηχανισμό ανταλλάσσοντας $\Theta(n \cdot p)$ μηνύματα.

Στη συνέχεια θα δείξουμε ότι ο brute-force αλγόριθμος είναι βέλτιστος και ότι η δευτεροβάθμια εξάρτηση είναι έμφυτη στο μηχανισμό SH.

Κάτω Φράγμα

Μπορεί να αποδειχθεί ένα κάτω φράγμα στον αριθμό των μηνυμάτων του SH για την κλάση των κατανεμημένων αλγορίθμων. Οι αλγόριθμοι αυτοί αρχικά θέτουν πραγματικές μεταβλητές στους κόμβους και μετά αρχίζουν να ανταλλάσσουν μηνύματα μεταξύ των κόμβων και ταυτόχρονα κάνουν κάποιους τοπικούς υπολογισμούς. Κάθε μήνυμα είναι ένας πραγματικός αριθμός, ένας γραμμικός συνδυασμός των μεταβλητών του κόμβου-αποστολέα και των μηνυμάτων που ελήφθησαν από τον κόμβο-αποστολέα στα προηγούμενα βήματα. Εάν οι συντελεστές που υπάρχουν σε αυτό το γραμμικό συνδυασμό είναι σταθεροί και καθορισμένοι εκ' των προτέρων τότε ο αλγόριθμος καλείται *oblivious linear distributed αλγόριθμος*.

Μία πιο γενική κατηγορία αλγορίθμων είναι εκείνη στην οποία το επόμενο μήνυμα που πρόκειται να σταλεί από ένα κόμβο, είναι ένα δέντρο απόφασης, με

συγκρίσεις ανάμεσα στους γραμμικούς συνδυασμούς των τοπικών μεταβλητών και στα μηνύματα που έχουν ήδη ληφθεί. Τα φύλλα αυτού του δέντρου υπολογίζουν γραμμικούς συνδυασμούς των τοπικών μεταβλητών και των μηνυμάτων. Αυτοί οι αλγόριθμοι καλούνται *non-oblivious linear distributed* ή απλά *linear distributed*. Οι αλγόριθμοι που αναφέραμε μέχρι τώρα ανήκουν στην κλάση αυτή.

Μπορεί να αποδειχθεί το επόμενο θεώρημα από το οποίο συμπεραίνουμε ο brute-force αλγόριθμος είναι ουσιαστικά βέλτιστος ως προς τον αριθμό των μηνυμάτων που ανταλλάσει. Μπορεί να αποδειχθεί το ακόλουθο θεώρημα.

Θεώρημα 3.2.4 Υπάρχει μία άπειρη οικογένεια multicast υπολογισμών, με n κόμβους και O(n) χρήστες, έτσι ώστε κάθε linear distributed αλγόριθμος που υλοποιεί τον SH μηχανισμό να απαιτεί την ανταλλαγή $\Omega(n^2)$ μηνυμάτων.

3.3 Online Πλειστηριασμοί

Στην ενότητα που ακολουθεί θα μελετήσουμε μία ειδική κατηγορία πλειστηριασμών τους on-line πλειστηριασμούς. Η ανάλυση βασίζεται στο άρθρο [LN00] των Ron Lavi και Noam Nisan.

3.3.1 Εισαγωγή

Στους κλασικούς πλειστηριασμούς έχουμε ένα αριθμό υποψήφιων αγοραστών (πλειοδότες) που επιθυμούν να αγοράσουν το ίδιο αντικείμενο ή ένα σύνολο αντικειμένων από τον πλειστηριαστή. Οι πλειοδότες βρίσκονται συνήθως στον ίδιο χώρο με τον πλειστηριαστή και οι προσφορές που κάνουν είναι ανοιχτές, δηλαδή κάθε πλειοδότης έχει γνώση για τις προσφορές των υπολοίπων. Ο στόχος του πλειστηριαστή είναι να πουλήσει το αντικείμενο αυτό σε όσο το δυνατό υψηλότερη τιμή. Οι πλειοδότες του ανακοινώνουν πρώτα τις προσφορές τους και εκείνος πουλά το αντικείμενο σε εκείνον με την υψηλότερη προσφορά.

Εκτός όμως από τους κλασικούς πλειστηριασμούς υπάρχουν και άλλα είδη πλειστηριασμών. Ένα παράδειγμα είναι οι Vickrey πλειστηριασμοί στους οποίους αναφερθήκαμε στο κεφ.1, καθώς και οι συνδυαστικοί πλειστηριασμοί που μελετήσαμε στην ενότητα 3.1.2. Στην ενότητα αυτή θα εξετάσουμε ένα διαφορετικό είδος πλειστηριασμών, τους on-line πλειστηριασμούς.

Τα τελευταία χρόνια έχουν προκύψει νέες εφαρμογές των πλειστηριασμών, κυρίως μέσω του internet, σε αυτούς ο αριθμός των πλειοδοτών είναι πολύ μεγάλος και δεν συμμετέχουν όλοι ταυτόχρονα στον πλειστηριασμό. Στην ενότητα αυτή θα μελετήσουμε πλειστηριασμούς όπου οι πλειοδότες φθάνουν σε διαφορετικές χρονικές στιγμές και ο μηχανισμός του πλειστηριασμού θα πρέπει να αποφασίσει για κάθε προσφορά την στιγμή που γίνεται. Στους κλασσικούς πλειστηριασμούς ο πωλητής λαμβάνει πρώτα όλες τις προσφορές και μετά αποφασίζει σε ποιους θα πωλήσει τα αντικείμενα. Επίσης οι συμμετέχοντες είναι διατεθειμένοι να περιμένουν για κάποιο χρονικό διάστημα ώστε να συγκεντρωθούν οι προσφορές και να αποφασιστεί η πώληση. Στους on-line

πλειστηριασμούς όμως οι συμμετέχοντες απαιτούν μία άμεση απάντηση στην προσφορά τους και δεν μπορούν να περιμένουν.

Στο μοντέλο που θα μελετήσουμε, k διαφορετικά αντικείμενα πρόκειται να πωληθούν σε μία δημοπρασία. Κάθε πλειοδότης έχει μία αποτίμηση για κάθε υποσύνολο των αντικειμένων, την όποία γνωρίζει μόνο αυτός και αντιπροσωπεύει την μεγαλύτερη προσφορά που είναι διατεθειμένος να κάνει. Ο πλειοδότης μαθαίνει την αποτίμηση κάποια συγκεκριμένη στιγμή και πρέπει αμέσως να κάνει την προσφορά του. Ο μηχανισμός του πλειστηριασμού μόλις πάρει την προσφορά πρέπει να αποφασίσει άμεσα (χωρίς να ξέρει τις μελλοντικές προσφορές) πόσα αντικείμενα θα δώσει στον πλειοδότη και σε ποια τιμή. Οι πλειστηριασμοί αυτοί καλούνται on-line (άμεσοι).

3.3.2 Το Μοντέλο

Θεωρούμε τον πλειστηριασμό k διαφορετικών και αδιαίρετων αντικειμένων, σε ένα σύνολο παιχτών. Διακρίνουμε την περίπτωση αυτή από εκείνη όπου το k είναι πολύ μεγάλο και έτσι μπορούμε να θεωρήσουμε ότι έχουμε μία ενιαία ποσότητα Q από ένα προϊόν που μπορεί να διαιρεθεί.

- Κάθε παίχτης έχει κάποιο θετικό όφελος από την απόκτηση κάποιας ποσότητας αντικειμένων. Το όφελος αυτό το γνωρίζει μόνο ο παίχτης. Θα καλούμε με $\upsilon_i(q)$ την *περιθωριακή αποτίμηση (marginal valuation)* του παίχτη i που αντιπροσωπεύει το πρόσθετο όφελος του παίχτη i από την απόκτηση του q-οστού αντικειμένου. Η $\upsilon_i(q)$ αντιπροσωπεύει το πόσο θέλει το q-οστό αντικείμενο ο παίχτης i. Η συνολική αποτίμηση για q αντικείμενα είναι $\sum_{j=1}^{q} \upsilon_i(j)$. Κάθε παίχτης i έχει μία *συνάρτηση περιθωριακής αποτίμησης (marginal valuation function)* που δεν είναι αύξουσα, για την οποία ισχύει: $\forall i, q : \upsilon_i(q+1) \le \upsilon_i(q)$.
- Όταν ο παίχτης i λάβει q αντικείμενα θα πληρώσει γι' αυτά P_i και η utility του θα είναι $U_i(q, P_i) = \sum_{j=1}^{q} \upsilon_i(j) - P_i$. Στόχος κάθε παίχτη είναι η μεγιστοποίηση αυτής της ποσότητας.
- Το on-line παίγνιο έχει την ακόλουθη δομή. Αρχικά το σύνολο των παιχτών δεν είναι γνωστό στο πλειστηριαστή και κανένας παίχτης δεν γνωρίζει την αποτίμηση του. Σε κάποιο χρόνο t_i, παίχτης i μαθαίνει την αποτίμησή του και πρέπει άμεσα να κάνει την προσφορά του. Επικεντρώνουμε την μελέτη μας στους μηχανισμούς άμεσης αποκάλυψης (direct revelation mechanisms), όπου οι παίχτες απλά δηλώνουν τις συναρτήσεις της περιθωριακής αποτίμησης. Η προσφορά (bid) του παίχτη i περιγράφεται από μία συνάρτηση $b_i(q)$, που δεν είναι αύξουσα, της μορφής $b_i : [1 \ldots k] \to \mathbb{R}$,. Οποιοσδήποτε παίχτης μπορεί να ψεύδεται, δηλώνοντας κάποιο $b_i(q) \neq \upsilon_i(q)$, με σκοπό να αυξήσει την προσωπική του utility. Ο πλειστηριαστής πρέπει να απαντήσει αμέσως στην προσφορά του παίχτη i πριν να γίνει η επόμενη προσφορά. Η απάντηση του πλειστηριαστή είναι η ποσότητα που θα πουλήσει στον παίχτη και σε ποια

τιμή. Αν στον παίχτη δεν δοθεί κάποια ποσότητα, τότε η συνολική πληρωμή είναι μηδέν. Το παίγνιο σταματά όταν ο πλειστηριαστής πουλήσει όλα τα αντικείμενα ή όταν και ο τελευταίος παίχτης ανακοινώσει την προσφορά του.

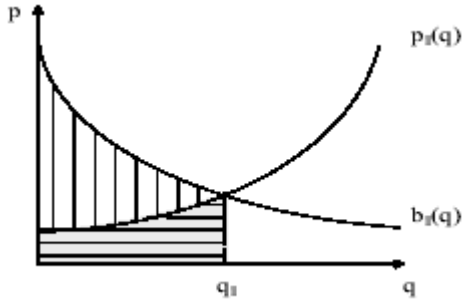
- Θα μελετήσουμε μηχανισμούς που είναι φιλαλήθεις υλοποιήσεις με κυρίαρχες στρατηγικές (φιλαλήθεις μηχανισμοί, Κεφ.1) και καλούνται φιλαλήθεις μηχανισμοί (incentive compatible ή strategyproof). Η στρατηγική (δηλαδή η προσφορά) $b_i(q)$ του παίχτη i θα καλείται *κυρίαρχη* εάν για κάθε άλλη προσφορά $\tilde{b}_i(q)$ και για οπιαδήποτε ακολουθία από προηγούμενες ή μελλοντικές προσφορές των άλλων παιχτών να ισχύει: $U_i(q_i, P_i) \geq U_i(\tilde{q}_i, \tilde{P}_i)$. Με q_i και $\tilde{q}_i$ συμβολίζουμε τις ποσότητες που πωλούνται στον παίχτη i όταν δηλώσει $b_i(q)$, και $\tilde{b}_i(q)$ αντίστοιχα και με P_i και $\tilde{P}_i$ τις πληρωμές που αντιστοιχούν στις ποσότητες αυτές. Δηλαδή για οποιαδήποτε ακολουθία προσφορών των άλλων παιχτών, η utility του παίχτη i θα μεγιστοποιείται αν κάνει την προσφορά $b_i(q)$ (όταν είναι φιλαλήθης). Ένας μηχανισμός άμεσης αποκάλυψης είναι *φιλαλήθης (incentive compatible ή strategyproof),* εάν η στρατηγική της αποκάλυψης της αληθούς αποτίμησης εκ' μέρους των παιχτών είναι κυρίαρχη.

3.3.3 Supply Curves σε On-Line Πλειστηριασμούς

Ορισμός 3.3.1 (Supply Curves) Θα λέμε ότι ένας on-line πλειστηριασμός «*βασίζεται σε εφαρμογή καμπυλών» (based on supply curves)* εάν πριν λάβει την i-οστή προσφορά καθορίζει μία συνάρτηση (supply curve) $p_i(q)$ βάση των προηγούμενων προσφορών και

1. Η ποσότητα q_i που πωλείται στον πλειοδότη i είναι εκείνη η ποσότητα q που μεγιστοποιεί το άθροισμα $\sum_{j=1}^{q}(b_i(j) - p_i(j))$, δηλαδή την utility του πλειοδότη.
2. Η τιμή που πρέπει να πληρώσει ο παίχτης i είναι $\sum_{j=1}^{q} p_i(j)$.

Ένας απλός τύπος εφαρμογής καμπυλών είναι εκείνος όπου κάθε εφαρμοζόμενη καμπύλη $p_i(q)$ δεν είναι φθίνουσα. Για μία τέτοια καμπύλη η ποσότητα q_i είναι η μεγαλύτερη ποσότητα q για την οποία $b_i(q) \geq p_i(q)$. Για την περίπτωση διαιρετών αγαθών ο παραπάνω τύπος μπορεί να οριστεί με όμοιο τρόπο. Η εφαρμοζόμενη καμπύλη $p_i(q)$ θα είναι μία πραγματική και όχι φθίνουσα συνάρτηση, η ποσότητα q_i καθορίζεται όπως πριν και η πληρωμή είναι $\int_0^{q_i} p_i(q)\,dq$. Εάν επιπλέον οι $b_i(q)$ και $p_i(q)$ είναι συνεχείς τότε η q_i είναι η μοναδική λύση της $b_i(q) = p_i(q)$



Εικόνα 3.3.1 : Παράδειγμα εφαρμογής καμπυλών

Παράδειγμα: Ένα παράδειγμα έχουμε στην εικόνα 3.3.1, όπου $p_1(q)$ είναι η εφαρμοζόμενη καμπύλη και $b_1(q)$ είναι η προσφορά. Σύμφωνα με τον ορισμό 3.3.1, η ποσότητα που θα λάβει ο παίχτης είναι ίση με q_1 και η συνολική πληρωμή είναι η περιοχή, που σημειώνεται με οριζόντιες γραμμές, κάτω από την $p_1(q)$. Η αποτίμηση του παίχτη για την ποσότητα q_1 είναι η περιοχή κάτω από την $b_1(q)$. Η utility του παίχτη που προκύπτει είναι η περιοχή μεταξύ της $b_1(q)$ και της $p_1(q)$ που σημειώνεται με τις κάθετες γραμμές. Μετά την πώληση ο μηχανισμός συνεχίζει με τον επόμενο παίχτη χρησιμοποιώντας μία νέα καμπύλη $p_2(q)$.

Σχόλια:

1. Στην ανάλυση που κάνουμε όταν αναφερόμαστε στον on-line πλειστηριασμό ουσιαστικά εννοούμε τον μηχανισμό που τον υλοποιεί. Η χρήση μηχανισμού για την λύση του προβλήματος είναι αναγκαία διότι ο πλειστηριαστής δεν γνωρίζει τις αληθείς αποτιμήσεις των πλειοδοτών. Τελικά ο στόχος του πλειστηριαστή θα είναι ο σχεδιασμός ενός φιλαλήθη μηχανισμού που θα υλοποιεί το πρόβλημα και θα του παρέχει τις αληθείς αποτιμήσεις των παιχτών. Εκτός όμως από το έλλειμμα πληροφορίας πρέπει να αντιμετωπίσει και το γεγονός ότι το πρόβλημα είναι on-line. Ο πλειστηριαστής όχι μόνο δεν γνωρίζει τις αποτιμήσεις των παιχτών αλλά δεν ξέρει ούτε ποιοι ή πόσοι είναι οι παίχτες. Ο μηχανισμός που θα σχεδιάσει θα πρέπει να είναι κατά κάποιο δυναμικό τρόπο φιλαλήθης δηλαδή να παραμένει φιλαλήθης συνεχώς κατά την εξέλιξη του πλειστηριασμού. Αυτό το πρόβλημα έρχονται να λύσουν οι συναρτήσεις $p_i(q)$. Η $p_i(q)$ ορίζεται κάθε φορά από τις προσφορές που έχουν γίνει μέχρι τη στιγμή που εμφανίστηκε ο i, έτσι για κάθε παίχτη i σχεδιάζεται και διαφορετική $p_i(q)$.

2. Η συνθήκη 1 του ορισμού θα μας οδηγήσει , όπως θα δείξουμε στη συνέχεια, σε φιλαλήθεις μηχανισμούς. Αυτό είναι αναμενόμενο διότι κάθε παίχτης έχει ως στόχο να μεγιστοποιήσει την utility του. Άρα θα είναι ειλικρινής όταν ο μηχανισμό του προσφέρει την δυνατότητα αυτή. Αυτό μας το παρέχει η συνθήκη-1.

Θεώρημα 3.3.2 Ένας on-line πλειστηριασμός είναι φιλαλήθης ανν βασίζεται σε εφαρμογή καμπυλών.

Απόδειξη: Θα αποδείξουμε τα ακόλουθα λήμματα.

Λήμμα 3.3.3 Ένας on-line πλειστηριασμός που βασίζεται σε εφαρμογή καμπύλων είναι φιλαλήθης.

Απόδειξη: Η utility του παίχτη i όταν λάβει κάποια ποσότητα q είναι $U_i(q) = \sum_{j=1}^{q}(\upsilon_i(j) - p_i(j))$. Έστω $b_i(q) \neq \upsilon_i(q)$ κάποια προσφορά, για την οποία η ποσότητα που θα πουληθεί είναι $\tilde{q}_i$. Επίσης έστω q_i η ποσότητα που θα πωληθεί στον παίχτη αν είναι ειλικρινής και δηλώσει $\upsilon_i(q)$. Τότε $U_i(q_i) \geq U_i(\tilde{q}_i)$ από την συνθήκη 1 του ορισμού 3.3.1 διότι όταν η προσφορά είναι αληθής τότε ο όρος που μεγιστοποιείται είναι ίσος με $U_i(q)$. Άρα ο πλειστηριασμός είναι φιλαλήθης. ■

Λήμμα 3.3.4 Κάθε φιλαλήθης on-line πλειστηριασμός βασίζεται σε εφαρμογή καμπυλών.

Απόδειξη: Κάνουμε την υπόθεση ότι σε κάθε φιλαλήθης on-line πλειστηριασμό Α, η συνολική πληρωμή του παίχτη i καθορίζεται κατά μοναδικό τρόπο από την συνολική ποσότητα που πουλήθηκε σε αυτόν και από τις προηγούμενες προσφορές. Αν δεν ισχύει η υπόθεση τότε θα υπάρχουν δύο διαφορετικές προσφορές $\upsilon(q)$ και $\tilde{\upsilon}(q)$ για τις οποίες η ποσότητα που θα πωληθεί να είναι η ίδια, αλλά η συνολική πληρωμή διαφορετική. Έστω P η συνολική πληρωμή όταν ο παίχτης δηλώσει $\upsilon(q)$, και $\tilde{P}$ η συνολική πληρωμή όταν δηλώσει $\tilde{\upsilon}(q)$. Υποθέτουμε ότι $P < \tilde{P}$. Τότε όμως ο παίχτης με αποτίμηση $\tilde{\upsilon}(q)$ θα αυξήσει την utility του αν δηλώσει $\upsilon(q)$(έτσι θα λάβει την ίδια ποσότητα αντικειμένων και θα πληρώσει συνολικά λιγότερο), δηλαδή θα κερδίσει ψευδόμενος. Καταλήγουμε συνεπώς σε αντίφαση, διότι ο Α είναι φιλαλήθης.

Την συνολική πληρωμή του παίχτη i θα την συμβολίζουμε με $P_i(q)$. Παρατηρούμε ότι σύμφωνα με τον ορισμό 3.3.1, η $p_i(q) = P_i(q) - P_i(q-1)$ είναι η εφαρμοζόμενη καμπύλη (η συνολική πληρωμή και η συνολική ποσότητα καθορίζονται σύμφωνα με τις συνθήκες του ορισμού). Η συνολική πληρωμή ικανοποιεί τον ορισμό 3.3.1 διότι το $p_i(q)$ προκύπτει από το $P_i(q)$ και $P_i(q) = P_i(0) + \sum_{j=1}^{q} p_i(q)$, όπου $P_i(0) = 0$. Για την συνολική ποσότητα τώρα, έστω q_i η ποσότητα που μεγιστοποιεί το $U_i(q) = \sum_{j=1}^{q} \upsilon_i(q) - P_i(q)$ (ο όρος αυτός είναι ίσος με τον όρο στη συνθήκη 1 του ορισμού 3.3.1). Έστω $b(q)$ κάποια προσφορά για την οποία ο Α πουλά την ποσότητα q_i. Αν τώρα ο Α για την αληθή προσφορά $\upsilon_i(q)$ πουλήσει την ποσότητα $\tilde{q}_i \neq q_i$, τότε ο παίχτης i θα

αύξανε την utility του δηλώνοντας $b(q)$, άτοπο , διότι ο Α είναι φιλαλήθης. ■
Από τα παραπάνω λήμματα το θεώρημα 3.3.2 προκύπτει άμεσα. ■

Μία ενδιαφέρουσα περίπτωση έχουμε όταν οι αποτιμήσεις των παιχτών είναι σταθερές περιθωριακές αποτιμήσεις. Σε αυτή την περίπτωση οι περιθωριακές αποτιμήσεις είναι του τύπου $\upsilon_i(q) = \upsilon_i$ για όλα τα i, q. Δηλαδή κάθε παίχτης κάνει την ίδια προσφορά για όλα τα αντικείμενα. Γι' αυτή την περίπτωση μπορούμε να χαρακτηρίσουμε με μεγαλύτερη ακρίβεια τις εφαρμοζόμενες καμπύλες. Αποδεικνύεται το ακόλουθο λήμμα.

Λήμμα 3.3.5 Υποθέτουμε ότι όλες οι περιθωριακές αποτιμήσεις είναι της μορφής $\upsilon_i(q) = \upsilon_i$. Τότε κάθε φιλαλήθης on-line πλειστηριασμός βασίζεται σε εφαρμοσμένες καμπύλες που δεν είναι φθίνουσες.

Στο γενικό μοντέλο που περιγράψαμε δεν υπάρχει κάποια σχέση που να συνδέει τις διαφορετικές εφαρμοζόμενες καμπύλες που ορίζονται κατά την διάρκεια ενός πλειστηριασμού. Μία ενδιαφέρουσα και χρήσιμη κατασκευή είναι εκείνη όπου όλες οι εφαρμοζόμενες καμπύλες προκύπτουν από κάποια ολική εφαρμοζόμενη καμπύλη.

Ορισμός 3.3.6 (Ολική Εφαρμοζόμενη Καμπύλη)
Ένας on-line πλειστηριασμός θα «*βασίζεται σε ολική εφαρμοζόμενη καμπύλη* p(q)» *(global supply curve)* εάν βασίζεται σε εφαρμοζόμενες καμπύλες και αν $p_i(q) = p\left(q + \sum_{j=1}^{i-1} q_j\right)$, όπου q_j είναι η ποσότητα που πωλείται στον j-οστό πλειοδότη.

Ερμηνεία: Η i-οστή εφαρμοζόμενη καμπύλη είναι η μετατόπιση προς τα αριστερά κατά q_{i-1} της (i-1)-οστής εφαρμοζόμενης καμπύλης. Έτσι ο i-οστός πλειοδότης θα λάβει την ποσότητα σύμφωνα με την εφαρμοζόμενη καμπύλη $p_1(q)$ μείον την ποσότητα που έχει πουληθεί προηγουμένως.

Θα δώσουμε τώρα τους ορισμούς του εσόδου και της κοινωνικής αποδοτικότητας. Για την worst-case ανάλυση θεωρούμε ότι όλες οι περιθωριακές αποτιμήσεις προέρχονται από κάποιο διάστημα $[\underline{p}, \overline{p}]$ χωρίς να υποθέτουμε κάποια κατανομή σε αυτές. Θεωρούμε ότι $\underline{p}$ είναι η τιμή εκκίνησης (που ορίζει ο πλειστηριαστής) με $\underline{p} > 0$.

Ορισμός 3.3.7 (Έσοδο) Το *έσοδο (revenue)* του πλειστηριασμού Α για ακολουθία αποτιμήσεων σ ορίζεται ως $R_A(\sigma)$. Είναι η προκύπτουσα utility του πλειστηριαστή, δηλαδή η συνολική πληρωμή που έλαβε συν την αποτίμηση της ποσότητας που δεν πούλησε. Συγκεκριμένα, έστω q_i η ποσότητα που πουλήθηκε στον i-οστό παίχτη της σ και P_i η συνολική τιμή που πλήρωσε ο i-οστός παίχτης ,

τότε: $$R_A(\sigma) = \sum_i P_i + \underline{p}\left(k - \sum_i q_i\right)$$

Το έσοδο εκφράζει το συνολικό κέρδος του πλειστηριαστή από τον πλειστηριασμό. Για την ποσότητα που δεν πούλησε κρατούμε την τιμή εκκίνησης που εκφράζει την μικρότερη τιμή που ορίζει ο πλειστηριαστής για αυτή την ποσότητα.

Ορισμός 3.3.8 (Κοινωνική αποδοτικότητα) Η *κοινωνική αποδοτικότητα (social efficiency)* του πλειστηριασμού Α για ακολουθία αποτιμήσεων σ, ορίζεται ως $E_A(\sigma)$ και είναι το άθροισμα όλων των utilities που προκύπτουν συμπεριλαμβανομένου και του πλειστηριαστή. Επίσης είναι ίσο με το άθροισμα των αποτιμήσεων των παιχτών για τις ποσότητες που έλαβαν (συμπεριλαμβανομένου και του πλειστηριαστή). Έχουμε:

$$E_A(\sigma) = \sum_i \sum_{j=1}^{q_i} \upsilon_i(j) + \underline{p}\left(k - \sum_i q_i\right)$$

Η utility κάθε παίχτη είναι $U_i(q, P_i) = \sum_{j=1}^{q} \upsilon_i(j) - P_i$ και του πλειστηριαστή είναι $R_A(\sigma) = \sum_i P_i + \underline{p}\left(k - \sum_i q_i\right)$. Το άθροισμα των σχέσεων αυτών μας δίνει τη σχέση της κοινωνικής αποδοτικότητας.

Για τη συνέχεια η μελέτη μας θα γίνει με competitive ανάλυση. Θα συγκρίνουμε το έσοδο και την κοινωνική αποδοτικότητα που προκύπτει από ένα on-line πλειστηριασμό με εκείνα που προκύπτουν από ένα off-line Vickrey πλειστηριασμό. Η σύγκριση με ένα off-line Vickrey πλειστηριασμό γίνεται διότι είναι ο μόνος πλειστηριασμός με κυρίαρχες στρατηγικές. Επίσης ο Vickrey πλειστηριασμός είναι πάντα βέλτιστος όσον αφορά την κοινωνική αποδοτικότητα.

Ορισμός 3.3.9 (Ανταγωνιστικότητα) (Competitiveness)

1. Ο on-line πλειστηριασμός Α είναι *c-ανταγωνιστικός όσον αφορά το έσοδο* εάν για κάθε ακολουθία αποτιμήσεων σ, $R_A(\sigma) \geq R_{vic}(\sigma)/c$.
2. Ο on-line πλειστηριασμός Α είναι *c-ανταγωνιστικός όσον αφορά την κοινωνική αποδοτικότητα*, εάν για κάθε ακολουθία αποτιμήσεων σ, $E_A(\sigma) \geq E_{vic}(\sigma)/c$.

Με τους παραπάνω ορισμούς εννοούμε ότι ένας c-ανταγωνιστικός πλειστηριασμός Α δεν θα δίνει ποτέ χειρότερα αποτελέσματα από το 1/c των αντίστοιχων αποτελεσμάτων του off-line Vickrey πλειστηριασμού.

3.3.4 Διαιρέσιμα Αντικείμενα

Στη συνέχεια θα εξετάσουμε την περίπτωση όπου έχουμε ένα διαιρέσιμο αντικείμενο (εμπόρευμα). Χωρίς βλάβη της γενικότητας, θεωρούμε ότι $Q = 1$. Θα περιγράψουμε μία ολική εφαρμοζόμενη καμπύλη που είναι $\Theta\left(\log\left(\overline{p}/\underline{p}\right)\right)$-

ανταγωνιστική όσον αφορά το έσοδο και την κοινωνική αποδοτικότητα. Υπενθυμίζουμε ότι όλες οι περιθωριακές αποτιμήσεις προέρχονται από κάποιο διάστημα $[\underline{p},\overline{p}]$ όπου $\underline{p}$ είναι η τιμή εκκίνησης (που ορίζει ο πλειστηριαστής) με $\underline{p}>0$.

Έστω c η μοναδική λύση της εξίσωσης:

$$c=\ln\frac{(\overline{p}/\underline{p})-1}{c-1}. \qquad (3.3.1)$$

Μπορεί να αποδειχθεί ότι $c=\Theta\left(\log\left(\overline{p}/\underline{p}\right)\right)$. Για παράδειγμα αν $\left(\overline{p}/\underline{p}\right)=2$ τότε $c=1.28$, αν $\left(\overline{p}/\underline{p}\right)=8$ τότε $c=1.97$.

Ορισμός 3.3.10 (Ανταγωνιστικός On-line Πλειστηριασμός)

- Ορίζουμε την *Ανταγωνιστική Εφαρμοζόμενη Καμπύλη (Competitive Supply Curve)* ως:

$$p(q)=\underline{p}\left(1+(c-1)e^{c\cdot q}\right). \qquad (3.3.2)$$

- Ο *Ανταγωνιστικός On-line Πλειστηριασμός (Competitive On-line Auction)* έχει την Ανταγωνιστική Εφαρμοζόμενη Καμπύλη ως ολική εφαρμοζόμενη καμπύλη.

Έστω $q(p)=p^{-1}(q)$ η αντίστροφη συνάρτηση της p(q) και $r(p)=\int_0^{q(p)}p(x)dx$. Μπορούμε να δείξουμε ότι η q(p) είναι η συνολική ποσότητα που πωλείται από τον Ανταγωνιστικό On-line Πλειστηριασμό όταν η τελευταία αποτίμηση τέμνει την τελευταία εφαρμοζόμενη καμπύλη στην τιμή p. Τότε r(p) είναι η συνολική του πλειστηριασμού για αυτή την ακολουθία.

Αποδεικνύονται τα ακόλουθα λήμματα.

Λήμμα 3.3.11 (El-Yaniv,Fiat,Karp και Turpin)
Για τις συναρτήσεις q(p) και r(p) ισχύει:

1. $\forall p\le c\cdot\underline{p}: q(p)=0, r(p)=0$
2. $\forall p>c\cdot\underline{p}: r(p)+\underline{p}\cdot(1-q(p))=p/c$
3. $q(\overline{p})=1$

όπου το c ορίζεται από την εξίσωση 3.3.1.

Λήμμα 3.3.12 (El-Yaniv,Fiat,Karp και Turpin)
Για κάθε σταθερά $\tilde{c}<c$, δεν υπάρχει συνάρτηση $\tilde{q}(p)$ για την οποία να ισχύει:

$$\forall p\in[\underline{p},\overline{p}],\ \tilde{r}(p)+\underline{p}\cdot(1-\tilde{q}(p))\ge p/\tilde{c}$$

όπου $\tilde{r}(p)=\int_0^{\tilde{q}(p)}\tilde{p}(x)dx$ και $\tilde{p}(q)=\tilde{q}^{-1}(p)$ είναι η αντίστροφη συνάρτηση της

$\tilde{q}(p)$.

Με βάση τα παραπάνω μπορούν να αποδειχθούν τα ακόλουθα λήμματα.

Λήμμα 3.3.13 Για κάθε ακολουθία αποτιμήσεων σ έχουμε $R_{cola}(\sigma) \geq R_{vic}(\sigma)/c$, όπου «cola» σημαίνει Competitive On-line Auction (Ανταγωνιστικός On-line Πλειστηριασμός).

Λήμμα 3.3.14 Για κάθε ακολουθία αποτιμήσεων σ έχουμε $E_{cola}(\sigma) \geq E_{opt}(\sigma)/c$, όπου $E_{opt}(\sigma)$ είναι η βέλτιστη κοινωνική αποδοτικότητα για την σ.

Από τα παραπάνω λήμματα προκύπτει το συμπέρασμα ότι:

Θεώρημα 3.3.15 Ο Ανταγωνιστικός On-line Πλειστηριασμός είναι c-ανταγωνιστικός όσον αφορά το έσοδο και την κοινωνική αποδοτικότητα.

Επίσης αποδεικνύονται ότι ο λόγος ανταγωνισμού του Ανταγωνιστικού On-line Πλειστηριασμού είναι ο καλύτερος που μπορούμε να επιτύχουμε.

Θεώρημα 3.3.16 Κάθε φιλαλήθης on-line πλειστηριασμός έχει λόγο ανταγωνισμού τουλάχιστον c όσον αφορά είτε το έσοδο είτε την κοινωνική αποδοτικότητα. Όπου το c είναι η λύση της εξίσωσης 3.3.1.

Αν περιορίσουμε τώρα το on-line μοντέλο θεωρώντας ότι είναι γνωστός εκ' των προτέρων ο συνολικός αριθμός n των παιχτών, τότε αποδεικνύεται ότι ο λόγος ανταγωνισμού οποιουδήποτε on-line αλγορίθμου έχει κάτω φράγμα τον όρο c_n όπου $c_n = \dfrac{c}{\left(\sqrt[n]{\overline{p}/\underline{p}}\right)^2}$ και το c είναι η λύση της εξίσωσης 3.3.1.

Λήμμα 3.3.17 Κανένας on-line πλειστηριασμός με n παίχτες δεν είναι καλύτερος από c_n-ανταγωνιστικός όσον αφορά το έσοδο του Vickrey πλειστηριασμού.

3.3.5 Αδιαίρετα Αντικείμενα

Στην συνέχεια θα εξετάσουμε την περίπτωση όπου τα αντικείμενα προς πώληση είναι διακριτά.

Ι) Πιθανοτικός πλειστηριασμός για ένα αδιαίρετο αντικείμενο
Χρησιμοποιώντας πιθανότητες μπορούμε όπως θα δείξουμε να έχουμε αναμενόμενο έσοδο και κοινωνική αποδοτικότητα που να είναι c-ανταγωνιστικά. Η συνάρτηση $q(p) = p^{-1}(q)$ είναι η αντίστροφη συνάρτηση της Ανταγωνιστικής Εφαρμοζόμενης Καμπύλης, μπορούμε να την σκεφτόμαστε ως αθροιστική συνάρτηση κατανομής στο διάστημα $[\underline{p}, \overline{p}]$.

Αν θεωρήσουμε ότι έχουμε ένα μόνο αδιαίρετο αντικείμενο, τότε η ολική εφαρμοζόμενη καμπύλη γι' αυτή την περίπτωση είναι απλά μία συγκεκριμένη τιμή γι'αυτό το αντικείμενο.

Ορισμός 3.3.18 (Πιθανοτικός On-Line Πλειστηριασμός για ένα αντικείμενο)
Πριν ληφθεί οποιαδήποτε προσφορά, ο πλειστηριαστής επιλέγει πιθανοτικά κάποια συγκεκριμένη τιμή p* , χρησιμοποιώντας την αθροιστική συνάρτηση κατανομής q(p). Μετά ο πλειστηριαστής πουλά το αντικείμενο στον πρώτο παίχτη με αποτίμηση τουλάχιστον p*, σε τιμή p*.

Μπορούμε να επαληθεύσουμε ότι ο παραπάνω πλειστηριασμός είναι φιλαλήθης με την ακόλουθη έννοια του όρου: Για κάθε πιθανοτική επιλογή ο παίχτης θα μεγιστοποιήσει την utility του αν αποκαλύψει την αληθή αποτίμηση. Μπορούμε να θεωρήσουμε μία ασθενέστερη συνθήκη στην οποία ο παίχτης μεγιστοποιεί την *αναμενόμενη* utility του (με βάση την κατανομή των πιθανοτικών επιλογών) αν δηλώσει την αληθή αποτίμηση. Σε αυτή την περίπτωση πρέπει να θεωρήσουμε ότι ο παίχτης είναι oblivious δηλαδή αγνοεί τα αποτελέσματα της πιθανοτικής συνάρτησης.

Ο πλειστηριασμός αυτός αποδεικνύεται ότι είναι c-ανταγωνιστικός σχετικά με το *αναμενόμενο* έσοδο και την κοινωνική αποδοτικότητα. Έτσι σε κάποιες περιπτώσεις μπορεί οι τιμές του on-line εσόδου και της κοινωνικής αποδοτικότητας να είναι μικρότερες του $1/c$ των αντίστοιχων τιμών του Vickrey πλειστηριασμού. Στις περισσότερες όμως περιπτώσεις και για οποιαδήποτε ακολουθία αποτιμήσεων, οι αναμενόμενες on-line τιμές εσόδου και κοινωνικής αποδοτικότητας θα είναι τουλάχιστον το $1/c$ των αντίστοιχων τιμών του Vickrey πλειστηριασμού.

ΙΙ) Ντετερμινιστικός πλειστηριασμός για k αδιαίρετα αντικείμενα
Τώρα θα εξετάσουμε την ντετερμινιστική περίπτωση. Αρχικά θεωρούμε την περίπτωση για $k=1$. Από το θεώρημα 3.3.2 προκύπτει ότι ο on-line πλειστηριασμός θα πρέπει να καθορίσει μία τιμή κράτησης p_i για τον i-οστό παίχτη. Έτσι το αντικείμενο θα πωληθεί στον i-οστό παίχτη σε τιμή p_i αν $\upsilon_i(1) > p_i$. Για την γενική περίπτωση όπου $k \geq 1$ παραθέτουμε συνοπτικά τα ακόλουθα αποτελέσματα.

Ορισμός 3.3.19 (Ο Διακριτός On-Line Πλειστηριασμός)
Ο Διακριτός On-Line Πλειστηριασμός βασίζεται στην ακόλουθη ολική εφαρμοζόμενη καμπύλη:

$$p(j) = \underline{p} \cdot \varphi^{\frac{j}{k+1}} \ , \ \text{για} \ \ j = 1, \ldots, k \ , \ \text{όπου} \ \varphi = \overline{p}/\underline{p} \ .$$

Τα ακόλουθα θεωρήματα καθορίζουν την ανταγωνιστικότητα του παραπάνω πλειστηριασμού και το κάτω φράγμα της.

Θεώρημα 3.3.20 Ο Διακριτός On-Line Πλειστηριασμός είναι $k \cdot \varphi^{\frac{1}{k+1}}$ - ανταγωνιστικός σε σχέση με το έσοδο και την κοινωνική αποδοτικότητα. Όταν

$k \geq 2 \cdot \ln \varphi$ τότε ο Διακριτός On-Line Πλειστηριασμός είναι $2 \cdot e \cdot (\ln(\varphi) + 1)$-ανταγωνιστικός όσον αφορά το έσοδο και την κοινωνική αποδοτικότητα.

Θεώρημα 3.3.21 Κάθε φιλαλήθης on-line πλειστηριασμός k αντικειμένων έχει λόγο ανταγωνισμού τουλάχιστον $m = \max\left\{\varphi^{\frac{1}{k+1}}, c\right\}$ σχετικά με το έσοδο και την κοινωνική αποδοτικότητα, όπου το c δίνεται από την εξίσωση 3.3.1.

3.4 Συμπεράσματα, μελλοντικές κατευθύνσεις

Στην ενότητα 3.1 Εξετάσαμε τους συνδυαστικούς πλειστηριασμούς και τα προβλήματα ελαχιστοποίησης κόστους. Τα προβλήματα αυτά είναι NP-Complete δηλαδή ο υπολογισμός της βέλτιστης λύσης είναι δύσκολος. Κατά συνέπεια οποιοσδήποτε μηχανισμός τα υλοποιεί χρησιμοποιώντας κάποιο βέλτιστο αλγόριθμο εξόδου θα είναι υπολογιστικά ανέφικτος.

Με σκοπό να κατασκευάσουμε υπολογιστικά εφικτούς μηχανισμούς, ορίσαμε τους *VCG-based μηχανισμούς*, που προκύπτουν όταν αντί για βέλτιστο αλγόριθμο εξόδου χρησιμοποιήσουμε κάποιο προσεγγιστικό αλγόριθμο. Το πρόβλημα που δημιουργείται από αυτή την αντικατάσταση είναι ότι οι μηχανισμοί που προκύπτουν δεν πάντα φιλαλήθεις και γενικά παρουσιάζουν ανώμαλη συμπεριφορά. Ορίζοντας την έννοια του *λογικού μηχανισμού* για συνδυαστικούς πλειστηριασμούς οδηγηθήκαμε στο συμπέρασμα ότι κάθε φιλαλήθης αλλά όχι βέλτιστος VCG-based μηχανισμός για συνδυαστικούς πλειστηριασμούς δεν είναι λογικός. Ανάλογη ανώμαλη συμπεριφορά δείξαμε (μέσω της έννοιας του *εκφυλισμένου αλγορίθμου*) ότι έχουν και οι φιλαλήθεις VCG-based μηχανισμοί για προβλήματα ελαχιστοποίησης κόστους.

Στη συνέχεια με σκοπό να ξεπεράσουμε τα προβλήματα αυτά ορίσαμε την έννοια της *εφικτής κυρίαρχης ενέργειας*, της *στρατηγικής γνώσης* καθώς και τον μηχανισμό *δεύτερης ευκαιρίας* που χρησιμοποιεί τις *συναρτήσεις προσφυγής*. Ο μηχανισμός αυτός είναι μία τροποποίηση του VCG-based μηχανισμού όπου οι παίχτες εκτός από τις δηλώσεις των τύπων τους μπορούν να υποβάλουν και συναρτήσεις προσφυγής. Έτσι δείξαμε ότι υπό κάποιες προϋποθέσεις για τους παίχτες, αυτή η προσθήκη των συναρτήσεων προσφυγής είναι αρκετή για να δώσει εφικτή φιλαλήθεια. Αποδείξαμε ότι εάν όλες οι συναρτήσεις στρατηγικής γνώσης των παιχτών είναι υπολογιστικά περιορισμένες και αν ο αλγόριθμος εξόδου είναι επίσης υπολογιστικά περιορισμένος, τότε ο μηχανισμός δεύτερης ευκαιρίας είναι εφικτά φιλαλήθης.

Στην ενότητα 3.2 μελετήσαμε το πρόβλημα της κατανομής του κόστους για multicast μεταδόσεις. Υποθέσαμε ότι ο strategyproof cost-sharing μηχανισμός θα πρέπει να ικανοποιεί τις συνθήκες NPT, CS και VP. Καθώς δεν υπάρχει strategyproof cost-sharing μηχανισμός που να είναι ταυτόχρονα budget-balanced και αποδοτικός (efficient), μελετήσαμε δύο cost-sharing μηχανισμούς τον Marginal Cost και τον Shapley Value. Σκοπός μας ήταν το κατά πόσο μπορούν οι μηχανισμοί αυτοί να υλοποιηθούν αποτελεσματικά σε ένα δίκτυο. Έτσι εξετάσαμε την υπολογιστική τους πολυπλοκότητα και βρήκαμε ότι ο MC είναι

πολύ πιο αποδοτικός από τον SH.

Ο MC υλοποιείται από ένα αλγόριθμο που απαιτεί την μετάδοση ενός μόνο μηνύματος για κάθε κατεύθυνση καθενός link στο δέντρο T(P). Θα καλούμε ένα μηχανισμό *αδιαχώριστο (nonseparable)* εάν τα μερίδια κόστους του παίχτη εξαρτώνται από εκείνα των άλλων παιχτών που μοιράζονται το link. Έτσι εάν για κάθε i, j με $T(i) \bigcap T(j) \neq \varnothing$, υπάρχουν διανύσματα utility u και υ με $u_k = \upsilon_k$ για όλα τα $k \neq j$, έτσι ώστε $x_i(u) \neq x_i(\upsilon)$ ή $\sigma_i(u) \neq \sigma_i(\upsilon)$. Η υλοποίηση ενός αδιαχώριστου μηχανισμού απαιτεί τουλάχιστον ένα μήνυμα να διασχίζει κάθε link στο T(P). Το αποτέλεσμα που δείξαμε είναι ότι ο MC είναι εύκολο να υλοποιηθεί όπως κάθε άλλος αδιαχώριστος μηχανισμός. Θα χρησιμοποιούμε τον όρο «minimal» για να περιγράφουμε μηχανισμούς που η υλοποίηση τους απαιτεί O(1) μηνύματα να διασχίζουν κάθε link. Σε αντίθεση με τον MC υπάρχει μία άπειρη οικογένεια περιπτώσεων για τις οποίες ο SH απαιτεί γραμμικό αριθμό μηνυμάτων σε γραμμικού αριθμού links. Θα χαρακτηρίζουμε ως «maximal» μηχανισμούς που απαιτούν τέτοιο αριθμό μηνυμάτων.

Από την ανάλυση μας προκύπτει μια σειρά ερωτημάτων. Κατ' αρχάς γνωρίζουμε πολύ λίγα για τους minimal και maximal μηχανισμούς. Μερικά ερωτήματα είναι τα εξής: Είναι όλοι οι αδιαχώριστοι strategyproof budget-balanced μηχανισμοί (που ικανοποιούν τις CS, VP και NPT) maximal; Δηλαδή το ερώτημα είναι αν η budget-balanced απαίτηση οδηγεί από τη φύση της σε μηχανισμούς που δεν είναι εφικτοί. Ένα άλλο ερώτημα είναι το αν υπάρχει κάποιος άλλος μηχανισμός εκτός από τον MC που να είναι αποδοτικός και strategyproof. Υπενθυμίζουμε ότι ο MC είναι ο μοναδικός αποδοτικός και strategyproof μηχανισμός που ικανοποιεί τις VP και NPT . Μήπως «χαλαρώνοντας» την απαίτηση μας για την VP ή NPT συνθήκη οδηγούμαστε σε μηχανισμούς που υλοποιούνται πιο δύσκολα από τον MC;

Στην ενότητα 3.3 μελετήσαμε τους on-line πλειστηριασμούς. Στους πλειστηριασμούς αυτούς οι πλειοδότες φθάνουν σε διαφορετικές χρονικές στιγμές και ο μηχανισμός του πλειστηριασμού θα πρέπει να παίρνει τις αποφάσεις την στιγμή που γίνεται μία προσφορά Ο μηχανισμός μόλις πάρει την προσφορά πρέπει να αποφασίσει άμεσα (χωρίς να ξέρει τις μελλοντικές προσφορές) πόσα αντικείμενα θα δώσει στον πλειοδότη και σε ποια τιμή. Το μοντέλο βασίστηκε στους *on-line πλειστηριασμούς που βασίζονται σε εφαρμογή καμπυλών*. Καθώς η φιλαλήθεια των παιχτών δεν μπορεί να θεωρηθεί δεδομένη μας ενδιέφερε κυρίως πότε και με ποιο τρόπο ένας πλειστηριασμός είναι strategyproof. Έτσι αρχικά αποδείξαμε ότι ένας on-line πλειστηριασμός είναι strategyproof αν και μόνο αν βασίζεται σε εφαρμογή καμπυλών.

Επειδή το πρόβλημα είναι on-line χρησιμοποιήσαμε competitive ανάλυση. Ορίσαμε τις έννοιες του *εσόδου* και της *κοινωνικής αποδοτικότητας* με σκοπό να μπορούμε να συγκρίνουμε οποιοδήποτε on-line πλειστηριασμό με τον off-line Vickrey πλειστηριασμό. Έτσι ορίσαμε την *c- ανταγωνιστικότητα* πλειστηριασμού Α ως προς το έσοδο και την κοινωνική αποδοτικότητα.

Για την περίπτωση πλειστηριασμού ενός διαιρετού αντικειμένου ορίσαμε τον Ανταγωνιστικό On-Line Πλειστηριασμό. Το βασικό αποτέλεσμα στο οποίο καταλήξαμε είναι ότι ο Ανταγωνιστικός On-Line Πλειστηριασμός είναι c-ανταγωνιστικός ως προς το έσοδο και την κοινωνική αποδοτικότητα του off-line

Vickrey πλειστηριασμού. Κανένας άλλος on-line πλειστηριασμός δεν έχει καλύτερο λόγο ανταγωνισμού είτε ως προς το έσοδο είτε ως προς την κοινωνική αποδοτικότητα.

Τέλος εξετάσαμε την περίπτωση πλειστηριασμού k αδιαίρετων αντικειμένων και ορίσαμε τον Πιθανοτικός On-Line Πλειστηριασμός για ένα αντικείμενο καθώς και τον Διακριτό On-Line Πλειστηριασμό για την ντετερμινιστική περίπτωση δίνοντας τον λόγο ανταγωνισμού του.

Βιβλιογραφία

[FPS00] Joan Feigenbaum, Christos Papadimitriou, and Scott Shenker. *Sharing the Cost of Multicast Transmissions*. In Thirty-Second Annual ACM Symposium on Theory of Computing (STOC00), May 2000.

[HS01] John Hershberger and Subhash Suri. *Vickrey Pricing in network routing: Fast payment computation.* In Proc. of the 42nd IEEE Symposium on Foundations of Computer Science, 2001.

[LN00] Ron Lavi and Noam Nisan. *Competitive Analysis of Incentive Compatible On line Auctions*. In the proceedings of "The 2nd ACM Conference on Electronic Commerce (EC-00)", October 2000.

[N99] Noam Nisan. *Algorithms for selfish agents*. In To appear in Proceeding of the 16th Symposium on Theoretical Aspects of Computer Science, Trier, Germany, March 1999.

[NR99] Noam Nisan and Amir Ronen. *Algorithmic Mechanism Design*. Extended Abstract appeared at the Thirty First Annual ACM Symbosium on Theory of Computing (STOC99), pages 129-140, May 1999. Games and Economic Behaviour 35, 2001: 166-196.

[NR00] Noam Nisan and Amir Ronen. *Computationally Feasible VCG Mechanisms*. In Proceedings of the Second ACM Conference on Electronic Commerce (EC00), 2000.

[OR94] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT press 1994.

[R00] Amir Ronen. *Algorithms for rational agents*. In Proceedings of 27th Annual Conference on Current Trends in Practice of Informatics, 2000.