National Technical University of Athens

School of Electrical
and Computer Engineering

Algorithms & Complexity

Semester 7 - Flow L

Analytic Assignment 4
Graphs - NP Completeness

Dimitris Dimos - 031 17 165

Athens
January, 2021

Περιεχόμενα

1 Άσκηση 1: Ταξιδεύοντας με Ηλεκτρικό Αυτοκίνητο

1.1 Ερώτημα (α)

Το ζητούμενό μας είναι να βρούμε τη συντομότερη διαδρομή από την κορυφή s στην κορυφή t, η οποία διέρχεται από τουλάχιστον μία κορυφή c του συνόλου C.

Με χρήση του αλγορίθμου $Dijkstra$ υπολογίζουμε σε χρόνο $\Theta(|E|+|V|log|V|)$ τα συντομότερα μονοπάτια από την κορυφή s μέχρι την κορυφή c_i, $\forall c_i \in C$. Έτσι, μετά από αυτή την εκτέλεση διαθέτουμε σε έναν πίνακα όλες τις (συντομότερες) αποστάσεις $dist_s[c_i]$.

Στη συνέχεια, εφαρμόζουμε ξανά $Dijkstra$ με αφετηρία την κορυφή t, μόνο που τώρα ο αλγόριθμος "κινείται" ανάποδα, δηλαδή ενώ κανονικά "τρέχει" μέσω ακμών συμφωνα με την κατεύθυνσή τους, τώρα "τρέχει" μόνο μέσω ακμών με ανάποδη κατεύθυνση. Ισοδυνάμως, θα μπορούσαμε να αντιστρέψουμε όλες τις ακμές του γράφου (κόστος $O(|E|)$ με αναπαράσταση λιστών γειτνίασης) και ύστερα να εφαρμόσουμε $Dijkstra$ από την t. Έτσι, σε επιπλέον χρόνο $\Theta(|E|+|V|log|V|)$ έχουμε έναν πίνακα με τις αποστάσεις $dist_t[c_i]$.

Το ζητούμενό μας είναι η ελαχιστοποίηση της απόστασης $s \longrightarrow c_i \longrightarrow t$, δηλαδή η διαδρομή με το:

$$min\{dist_s[c_i] + dist_t[c_i]\}$$

Για κάθε μία από τις αποστάσεις $dist_s[c_i]$ υπάρχει μία $dist_t[c_i]$ (που μπορεί αν είναι και άπειρη αν δεν υπάρχει μονοπάτι). Επομένως, σε χρόνο $O(|C|) = O(|V|)$ βρίσκουμε το ελάχιστο άθροισμα αποστάσεων.

Επειδή, το ζητούμενο είναι το συντομότερο μονοπάτι (όχι το ελάχιστο κόστος) θα πρέπει κατά την εκτέλεση των δύο $Dijkstra$ να αποθηκεύουμε τον πατέρα κάθε κορυφής που επισκεπτόμαστε. Αυτό δεν αυξάνει την πολυπλοκότητα του αλγορίθμου. Πρακτικά, οι πατέρες στο κομμάτι $s \longrightarrow c_i$ είναι όντως πρόγονοι των κορυφών του μονοπατιού, ενώ οι πατέρες στο μονοπάτι $c_i \longrightarrow t$ είναι απόγονοι, ώστε η ροή της διαδρομής να είναι σωστή.

Ορθότητα

Η ορθότητα του αλγορίθμου μας βασίζεται στην ορθότητα του $Dijkstra$. Αφού τα μονοπάτια $s \longrightarrow c_i$ και $c_i \longrightarrow t$ είναι ελάχιστα, τότε το ελάχιστο άθροισμα $dist_s[c_i] + dist_t[c_i]$ θα πρέπει να αντιστοιχεί στο συντομότερο μονοπάτι που διέρχεται από μία τουλάχιστον c_i. Αν υπάρχει άλλο μονοπάτι που διέρχεται από μια άλλη c_j με τις παραπάνω προδιαγραφές, τότε θα πρέπει:

$$dist_s[c_j] + dist_t[c_j] < dist_s[c_i] + dist_t[c_i],$$

που είναι άτοπο, αφού το $dist_s[c_i] + dist_t[c_i]$ είναι ελάχιστο.

Πολυπλοκότητα

Για να υπολογίσουμε το ζητούμενο, εκτελούμε 2 φορές $Dijkstra$ (η αντιστροφή των ακμών δεν μας επιβαρύνει ασυμπτωτικά) και μια γραμμική αναζήτηση στη σύνολο C. Άρα, η πολυπλοκότητα είναι:

$$O(|E|+|V|log|V|+|C|) = O(|E|+|V|log|V|+|V|) =$$

$$\Theta(|E|+|V|log|V|)$$

1.2 Ερώτημα (β)

Το ζητούμενό μας είναι η εύρεση του συντομότερου μονοπατιού μεταξύ s και t κατά τη διάσχιση του οποίου δεν ξεμένουμε από καύσιμα. Αυτό συνεπάγεται ότι κάθε ζεύγος διαδοχικών σταθμών φόρτισης του μονοπατιού έχει απόσταση μικρότερη ή ίση με την αυτονομία α του αυτοκινήτου μας.

Σκεπτόμαστε, λοιπόν, ως εξής:

Αν η αυτονομία του αυτοκινήτου μας ήταν αρκετή δεν θα χρειαζόταν να κάνουμε έλεγχο εγκυρότητας μονοπατιού και θα λύναμε το πρόβλημα απλώς με εφαρμογή $Dijkstra$. Τώρα, όμως, κάποια μονοπάτια είναι πιο μεγάλα από την αυτονομία. Οπότε θα τα "κλαδέψουμε".

Αλγόριθμος

Κατά την εκτέλεση των παρακάτω $Dijkstra$ αποθηκεύουμε τις κορυφές που συναποτελούν τα συντομότερα μονοπάτια.

1. Εφαρμόζουμε $Dijkstra$ από την s.
2. Για κάθε μία εκ των $c_i \in C$ εκτελούμε $Dijkstra$.
3. Σχεδιάζουμε εκ νέου γράφο με:
   - κορυφές τις s, t και c_i, $\forall c_i \in C$ και
   - ακμές $s \longrightarrow c_i$ με βάρος ίσο με το κόστος του συντομότερου μονοπατιού που ενώνει κάθε ζεύγος (s, c_i), αν αυτό είναι μικρότερο ή ίσο της αυτονομίας α
   - ακμές $c_i \longrightarrow c_j$ με βάρος ίσο με το κόστος του συντομότερου μονοπατιού που ενώνει κάθε ζεύγος (c_i, c_j), αν αυτό είναι μικρότερο ή ίσο της αυτονομίας α
   - ακμές $c_i \longrightarrow t$ με βάρος ίσο με το κόστος του συντομότερου μονοπατιού που ενώνει κάθε ζεύγος (c_i, t), αν αυτό είναι μικρότερο ή ίσο της αυτονομίας α
   - ακμή $s \longrightarrow t$ με βάρος ίσο με το κόστος του συντομότερου μονοπατιού μεταξύ s και t, αν αυτό είναι μικρότερο ή ίσο της αυτονομίας α
4. Εφαρμόζουμε $Dijkstra$ στον νέο γράφο από την s.
5. Οι κορυφές που αντιστοιχούν στις ακμές του συντομότερου μονοπατιού $s \longrightarrow t$ (τις οποίες ουσιαστικά συγχωνεύσαμε πριν την εκτέλεση του τελευταίου $Dijkstra$) αποτελούν το ζητούμενο συντομότερο μονοπάτι.

Ορθότητα

Με τη διαδικασία συγχώνευσης των συντομότερων εφικτών μονοπατιών μεταξύ κορυφών - σταθμών εξασφαλίζουμε (χάρη στην ορθότητα του $Dijkstra$) ότι αν πρόκειται, κατά το μονοπάτι που τελικώς ζητάμε, να περάσουμε από έναν σταθμό σε έναν άλλο, τότε θα ακολουθήσουμε νομοτελειακά την ακμή που απομένει στο τελικό μας γράφημα. Οι ακμές μας έχουν βάρος μικρότερο ή ίσο με την αυτονομία, οπότε σε κάθε μετάβαση από σταθμό σε σταθμό δεν ξεμένουμε ποτέ. Οπότε το μονοπάτι που επιστρέφει ο τελικός $Dijkstra$ είναι σίγουρα έγκυρο από την άποψη του βάρους ακμών, αλλά και βέλτιστο αφού αυτό εξασφαλίζεται από την ορθότητα του ίδιου του $Dijkstra$.

Εμένοντας, ακόμη περισσότερο, στην τελευταία πρόταση, επισημαίνουμε ότι το τελικό μονοπάτι $best\_path$ που επιστρέφεται αποτελείται από επιμέρους βέλτιστα μονοπάτια. Δηλαδή δεν υπάρχει

"υπομονοπάτι" που να συνδέει δύο οποιεσδήποτε κορυφές του $best\_path$ με μικρότερο βάρος από το "υπομονοπάτι" που τις συνδέει στο $best\_path$.

Άρα, αφού στον νέο γράφο όλες οι ακμές είναι οι βέλτιστες δυνατές μεταξύ των κορυφών που ενώνουν, τότε και το συνολικό συντομότερο μονοπάτι του τελικού $Dijkstra$ θα είναι βέλτιστο.

Πολυπλοκότητα

Εκτελούμε $1 + |C| + 1$ φορές $Dijkstra$, ενώ χτίζουμε κι έναν γράφο εκ νέου. Πολυπλοκότητα:

$$O(|E| + |V|log|V|) + |C|O(|E| + |V|log|V|) + O(|E| + |V|log|V|) + O(|V| + |E|) =$$

$$O\Big(|C|\big(|E| + |V|log|V|\big)\Big)$$

2 Άσκηση 2: Στρίβοντας μόνο δεξιά!

2.1 Ερώτημα (α)

Το ζητούμενο του ερωτήματος είναι ο υπολογισμός του μονοπατιού από ένα σημείο εκκίνησης, έστω s, μέχρι ένα τελικό σημείο, έστω t, το οποίο θα έχει ελάχιστο πλήθος δεξιών στροφών. Σε περίπτωση ισοβαθμίας, επιλέγουμε το μονοπάτι που ελαχιστοποιεί τη συνολική απόσταση.

Διατηρούμε:

- Έναν *bool* πίνακα $marked$ που είναι $4 \times (n \times m)$
- Μια *priority queue* που διατηρεί τα στοιχεία της ταξινομημένα κατά αύξουσα σειρά σύμφωνα με το πρώτο στοιχείο (πλήθος δεξιών στροφών) μιας τριπλέτας φυσικών και, σε περίπτωση ισοβαθμίας, σύμφωνα με το δεύτερο (συνολικό μήκος).

Η θέση $marked[d][x][y]$ είναι 1 αν έχουμε επισκεφτεί την κορυφή (x, y) του πλέγματος με κατεύθυνση d, όπου:

- d = 1: κατεύθυνση ↑
- d = 2: κατεύθυνση →
- d = 3: κατεύθυνση ↓
- d = 4: κατεύθυνση ←

Στα παρακάτω επισημαίνουμε ότι σε κάθε εισαγωγή στην ουρά, αν το στοιχείο που εισάγεται υπάρχει ήδη, τότε το "χειρότερο" εκ των δύο αφαιρείται (όπως ακριβώς συμβαίνει στον $Dijkstra$).

Αλγόριθμος

1. Τοποθετούμε στην ουρά την κορυφή εκκίνησης (a,b) με ζεύγος (δεξιές στροφές $= 0$, συνολικό μήκος $= 0$, d $= 1$) και σημειώνουμε 1 στη θέση $marked[1][a][b]$.
2. Επαναλαμβάνουμε μέχρι να αδειάσει η ουρά ή να φτάσουμε στο τελικό σημείο t:
   - Βγάζουμε το πρώτο στοιχείο, έστω (x, y), της ουράς που έχει τριπλέτα, έστω (r, s, d), και το μαρκάρουμε στην $marked[d][x][y]$.
   - Αν $d = 1$ ($\uparrow$): ελέγχουμε αν έχουμε μαρκάρει προηγουμένως τους γείτονες $(x+1, y)$ και $(x, y+1)$ με κατεύθυνση $\rightarrow$ και $\uparrow$, αντίστοιχα, (σύμφωνα με τον πίνακα $marked$) και, όσους δεν έχουμε, τους τοποθετούμε στην ουρά:
     - τον $(x, y+1)$ με τριπλέτα $(r, s+1, 1)$
     - τον $(x+1, y)$ με τριπλέτα $(r+1, s+1, 2)$
   - Αν $d = 2$ ($\rightarrow$): ελέγχουμε αν έχουμε μαρκάρει προηγουμένως τους γείτονες $(x, y-1)$ και $(x+1, y)$ με κατεύθυνση $\downarrow$ και $\rightarrow$, αντίστοιχα, (σύμφωνα με τον πίνακα $marked$) και, όσους δεν έχουμε, τους τοποθετούμε στην ουρά:
     - τον $(x+1, y)$ με τριπλέτα $(r, s+1, 1)$
     - τον $(x, y-1)$ με τριπλέτα $(r+1, s+1, 2)$
   - Αν $d = 3$ ($\downarrow$): ελέγχουμε αν έχουμε μαρκάρει προηγουμένως τους γείτονες $(x-1, y)$ και $(x, y-1)$ με κατεύθυνση $\leftarrow$ και $\downarrow$, αντίστοιχα, (σύμφωνα με τον πίνακα $marked$) και, όσους δεν έχουμε, τους τοποθετούμε στην ουρά:
     - τον $(x, y-1)$ με τριπλέτα $(r, s+1, 1)$
     - τον $(x-1, y)$ με τριπλέτα $(r+1, s+1, 2)$
   - Αν $d = 4$ ($\leftarrow$): ελέγχουμε αν έχουμε μαρκάρει προηγουμένως τους γείτονες $(x, y+1)$ και $(x-1, y)$ με κατεύθυνση $\uparrow$ και $\leftarrow$, αντίστοιχα, (σύμφωνα με τον πίνακα $marked$) και, όσους δεν έχουμε, τους τοποθετούμε στην ουρά:
     - τον $(x-1, y)$ με τριπλέτα $(r, s+1, 1)$
     - τον $(x, y+1)$ με τριπλέτα $(r+1, s+1, 2)$
3. Αν η ουρά αδειάσει χωρίς να φτάσουμε στο t, τότε επιστρέφουμε αποτυχία, διαφορετικά επιστρέφουμε (μέσω *backtracking* που θα εξηγηθεί λίαν συντόμως) το ζητούμενο μονοπάτι.

Για να είναι εφικτό το *backtracking* και να μπορούμε να παράγουμε το επιθυμητό μονοπάτι $s \longrightarrow t$ θα πρέπει κάθε φορά που εισάγουμε μια κορυφή στην ουρά με κατεύθυνση d, να αποθηκεύουμε τον πατέρα με την κατεύθυνσή του.

Πολυπλοκότητα

Ο αλγόριθμος είναι, κατά βάση, τροποποίηση του $Dijkstra$. Σε κάθε κορυφή μπορούμε να πάμε με 4 τρόπους (κατευθύνσεις), αλλά ποτέ με τον ίδιο. Οπότε, ασυμπτωτικά η πολυπλοκότητα είναι ίδια με του $Dijkstra$ αφού η διαδικασία διάσχισης του γράφου είναι η ίδια, απλώς κάθε κόμβος μπορεί αν εξερευνηθεί το πολύ 4 φορές. Επιπλέον, οι ακμές του γράφου είναι της τάξης του $O(4 \cdot |V|) = O(nm)$ Άρα:

$$4 \cdot O(|E| + |V| log|V|) = O(nm + nmlog(nm)) =$$

$$O\big(nm \cdot log(nm)\big)$$

Ορθότητα

Η ορθότητα του αλγορίθμου μας βασίζεται στην ορθότητα του ίδιου του $Dijkstra$. Ναι μεν τον τροποποιούμε, έτσι ώστε κάποιες κορυφές να ξαναμπαίνουν στην ουρά (με διφορετική κάθε φορά κατεύθυνση), ωστόσο με την *priority queue* εξασφαλίζουμε ότι κάθε επίσκεψή μας (εξαγωγή από την ουρά) σε κάποια κορυφή είναι και η συντομότερη δυνατή (λιγότερες δεξιές στροφές), αφού πάντοτε προηγούνται τα αναπτυσσόμενα μονοπάτια με τον ελάχιστο αριθμό δεξιών στροφών. Επομένως, όταν φτάσουμε στο τελικό σημείο, τότε το μονοπάτι μας θα έχει τις λιγότερες δυνατές δεξιές στροφές, διαφορετικά, κάποιο με λιγότερες θα προηγούνταν στην *priority queue* και θα αναπτυσσόταν πρώτο.

2.2 Ερώτημα (β)

Το ερώτημα μπορεί να απαντηθεί με μια τροποποίηση του αλγορίθμου του ερωτήματος (α). Πλέον δεν θέλουμε να επιτρέπονται δεξιές στροφές που δεν ακολουθούνται αμέσως από μια ακόμα δεξιά στροφή. Για αυτό τον λόγο, πριν εκτελέσουμε κάποιον αλγόριθμο διάσχισης του πλέγματος, τροποποιούμε τον γράφο ως εξής:

Για κάθε κορυφή u η οποία μπορεί να φτάσει σε μια άλλη κορυφή v με δύο διαδοχικές δεξιές στροφές, προσθέτουμε μια κατευθυνόμενη ακμή $u \longrightarrow v$.

Αυτή η προσθήκη θα μας κοστίσει χρόνο $O(|V|) = O(nm)$, αφού θα χρειαστεί να διασχίσουμε τις κορυφές του πλέγματος ακριβώς μία φορά την κάθε μία για να προσθέσουμε τις νέες ακμές.

Τώρα, στον νέο γράφο εκτελούμε τον αλγόριθμο του ερωτήματος (α), με την εξής τροποποίηση:

Δεν θα εξετάζουμε ποτέ, από μια κορυφή που μόλις βγήκε από την ουρά, τον γείτονα στον οποίο φτάνουμε με μια δεξιά στροφή. Αντ'αυτού, ελέγχουμε τον γείτονα (που τώρα, λόγω των νέων ακμών, έγινε γείτονας) στον οποίο φτάνουμε μέσω της νέας ακμής. Φυσικά, ο γείτονας αυτός εισέρχεται στην ουρά και η τριπλέτα του είναι κατάλληλα διαμορφωμένη (οι δεξιές στροφές είναι αυξημένες κατά 2 και ο προσανατολισμός κατάλληλος, πχ από ↑ γίνεται ↓).

Πολυπλοκότητα

Η πολυπλοκότητα παραμένει αμετάβλητη, αφού ο αλγόριθμος ακολουθεί ίσα σε χρόνο βήματα με πριν, ενώ η τροποποίησή του στην αρχή δεν μας επιβαρύνει με χρόνο μεγαλύτερο του $O\big(nmlog(nm)\big)$.

$$O\big(nmlog(nm)\big)$$

Ορθότητα

Η εξήγηση για την ορθότητα είναι εντελώς όμοια με την εξήγηση ορθότητας του ερωτήματος (α).

3 Άσκηση 3: Σχεδιασμός Videogame

Πρωτού προσωρήσουμε στα ερωτήματα, τροποποιούμε κατάλληλα τον γράφο, δίνοντας μια πιο βολική, αλλά ισοδύναμη μορφή.

Για κάθε κορυφή v με βάρος w, αναθέτουμε σε όλες τις ακμές $u \longrightarrow v$ βάρος w. Τώρα, ο γράφος μας έχει στις ακμές του μόνο βάρη, γεγονός που καθιστά πιο εύκολη την ερφαρμογή γνωστός αλγορίθμων.

3.1 Ερώτημα α: Όχι κύκλοι θετικού μήκους

Το ζητούμενο είναι να βρεθεί μονοπάτι από την αρχή μέχρι το τέλος, χωρίς να τελειώσουν ποτέ οι πόντοι ζωής κατά τη διάρκεια. Ισοδύναμα, αυτό σημαίνει να βρεθεί μονοπάτι του οποίου όλα τα υπομονοπάτια με αρχική κορυφή την s δεν έχουν αρνητικό μήκος.

Θα τροποποιήσουμε τον $Bellman-Ford$ προκειμένου να βρίσκει τα μακρύτερα δυνατά μονοπάτια από την κορυφή s προς κάθε άλλη κορυφή v (που ισοδυναμεί με τους περισσότερους δυνατούς πόντους ζωής που μπορούμε να έχουμε μέχρι την v), υπό τον περιορισμό ότι ποτέ δεν θα υπάρξει υπομονοπάτι από την s προς την v με αρνητικό μήκος (δηλαδή ότι ποτέ δεν θα ξεμείνουμε από πόντους ζωής στη διάρκεια).

Αν το μακρύτερο μονοπάτι προς την t κατά τον αλγόριθμο πάρει θετική τιμή, σημαίνει ότι οι περισσότεροι δυνατοί πόντοι ζωής που μπορούμε να έχουμε κατά την άφηξή μας στην t είναι θετικοί και ο γράφος είναι r-ασφαλής.

Επομένως, ο αλγόριθμος που ακολουθούμε είναι ο $Bellman-Ford$ τροποποιημένος ως εξής:

1. Αρχικοποιούμε τον πίνακα $long$ με τα μακρύτερα μονοπάτια. Θέτουμε $long[0] = r$ και $long[u] = -\infty, \forall u \neq 0$.
2. Εκτελούμε τον αλγόριθμο, αλλά κατά τις επαναλήψεις, ανανεώνουμε τα $long[u]$ μόνο όταν βρίσκουμε
$$long[v] + w(v \longrightarrow u) > long[u],$$
όπου η v είναι γειτονική κορυφή της u με κατεύθυνση $v \longrightarrow u$ και, **ταυτόχρονα**,
$$long[v] + w(v \longrightarrow u) > 0,$$
έτσι ώστε αν το $long[v] + w(v \longrightarrow u)$ δεν είναι $-\infty$, αλλά είναι ακόμα αρνητικό, να μην γίνεται ανανέωση, αφού αυτή θα σημαίνει ότι φτάνουμε με αρνητικούς πόντους στην u.

Με δεδομένο, λοιπόν, ότι δεν υπάρχουν κύκλοι θετικού μήκους, οι οποίοι θα προκαλούσαν ατέρμονες ανανεώσεις, ύστερα από |V| επαναλήψεις του τροποποιημένου $Bellman-Ford$ κάθε $long[i]$ περιέχει τους μέγιστους πόντους ζωής που μπορούμε να έχουμε όταν φτάνουμε στην κορυφή i. Επομένως, ελέγχουμε τους πόντους $long[t]$ και αν αυτοί είναι θετικοί, τότε ο γράφος είναι $r-$ασφαλής, διαφορετικά όχι.

Πολυπλοκότητα

Το χρονικό κόστος είναι οι επαναλήψεις, οι οποίες είναι, σύμφωνα με τον $Bellman-Ford$, |V| - 1 σε πλήθος, και σε κάθε μία από αυτές "περνάμε" μία φορά από κάθε ακμή. Επομένως, η πολυπλοκότητα είναι:

$$O(|V||E|)$$

Ορθότητα

Η ορθότητα του αλγορίθμου απορρέει από την ορθότητα του ίδιου του $Bellman - Ford$.

3.2 Ερώτημα β: Υπάρχουν κύκλοι θετικού μήκους

Ακολουθώντας τον αλγόριθμο του προηγούενου ερωτήματος, φτάνουμε σε έναν πίνακα $long$ με τους περισσότερους δυνατούς πόντους για κάθε κορυφή. Αν εκτελέσουμε μια ακόμη επανάληψη και υπάρξει έστω μία ακόμη ανανέωση, αυτό σημαίνει ότι υπάρχει κύκλος με θετικό μήκος, ο οποίος και ευθύνεται για αυτή την ανανέωση. Αυτό, όμως, σημαίνει, ταυτόχρονα, ότι η κορυφή στην οποία έγινε η ανανέωση, είναι μέρος του κύκλου και μπορούμε να την "φτάσουμε". Έχοντας, λοιπόν, ρυθμίσει τον αλγόριθμο ώστε να αποθηκεύει τους πατέρες που προκαλούν ανανέωση των $long$ των κορυφών, γνωρίζουμε ποιές είναι οι κορυφές του κύκλου, κι έτσι μπορούμε να φτάσουμε σε μια εκ των κορυφών του, να κάνουμε όσους κυκλικούς περιπάτους χρειαζόμαστε για να αυξήσουμε αρκετά τους πόντους ζωής μας και να συνεχίσουμε μέχρι την t. Έτσι ο γράφος είναι $r-$ασφαλής.

Στη συνέχεια, παρά την ύπαρξη κύκλων, ενδέχεται να μην μπορούμε να φτάσουμε σε καμία εκ των κορυφών τους με θετικούς πόντους. Τότε ο αλγόριθμος δεν θα προκαλέσει καμία ανανέωση στην |V|-οστή επανάληψη και το αν ο γράφος είναι $r-$ασφαλής εξαρτάται μόνο από την τιμή του $long[t]$.

Πολυπλοκότητα

Ασυμπτωτικά, αμετάβλητη σε σχέση με πριν

$$O(|V||E|)$$

Ορθότητα

Η ορθότητα του αλγορίθμου απορρέει από την ορθότητα του ίδιου του $Bellman - Ford$.

3.3 Ερώτημα γ: Το ελάχιστο r

Πλέον, μετα τα δύο προηγούμενα ερωτήματα διαθέτουμε αλγορίθμους που βρίσκουν αν ο γράφος είναι $r-$ασφαλής με ή χωρίς την ύπαρξη κύκλων θετικού μήκους. Τώρα το ζητούμενό μας είναι να βρούμε πιο είναι το μικρότερο r για το οποίο ο γράφος είναι ασφαλής.

Μια πρώτη (ίσως, αφελής) προσέγγιση θα ήταν να δοκιμάσουμε τον τροποποιημένο $Bellman - Ford$ για όλες τις τιμές του r γραμμικά από το 1 και ανεβαίνοντας, μέχρι να βρούμε το πρώτο r για το οποίο ο γράφος μας είναι ασφαλής. Αυτό θα μας κόστιζε χρόνο ανάλογο του r που αναζητούμε, δηλαδή $O(r \cdot |V||E|)$.

Μπορούμε, ωστόσο, να βελτιώσουμε αυτόν τον χρόνο με την εξής τεχνική:

Δοκιμάζουμε τον τροποποιημένο $Bellman - Ford$ με $r =$ δύναμη του 2. Σε κάθε μας δοκιμή, θέτουμε την τιμή του r στην αμέσως μεγαλύτερη δύναμη του 2. Στο $i - οστ$ βήμα το δομιμαζόμενο r θα δώσει $r-$ασφαλή γράφο. Τότε, κάνουμε $binary\ search$ στο διάστημα $[2^{i-1}, 2^i]$, για κάθε τιμή του r στο διάστημα, αν δίνει $r-$ασφαλή γράφο, επαναλαμβάνουμε με μικρότερο r, διαφορετικά με μεγαλύτερο, μέχρι να βρούμε το μικρότερο δυνατό r.

Πολυπλοκότητα

Μέχρι να εντοπίσουμε το κατάλληλο 2^i θα κάνουμε επαναλήψεις της τάξης του $log(r)$, ενώ η δυαδική αναζήτηση θα μας κοστίσει ακόμη $log(r)$. Επομένως, είναι:

$$\big(O\big(log(r)\big) + O\big(log(r)\big) \cdot O(|V| \cdot |E|) = O\big(|V||E|log(r)\big)$$

Ορθότητα

Η ορθότητα του τροποποιημένου $Bellman - Ford$ είναι ζήτημα των παραπάνω ερωτημάτων, ενώ η αναζήτηση του r είναι ορθή, λόγω της μονοτονίας που παρουσιάζει ως προς την ασφάλεια του γράφου. Αυτό σημαίνει ότι αν για κάποιο r ο γράφος είναι ασφαλής, τότε θα είναι ασφαλής και για κάθε r' που είναι μεγαλύτερο του r, ενώ αν για κάποιο r ο γράφος δεν είναι ασφαλής τότε δεν θα είναι για κανένα r' μικρότερο του r. Άρα, όταν εξετάσουμε το 2^{i-1} και αυτό δεν δίνει $r-$ ασφαλή γράφο (ενώ το 2^i δίνει), τότε κανένα $r' \leq 2^{i-1}$ δεν δίνει $r-$ασφαλή γράφο. Οπότε με δυαδική αναζήτηση στο $[2^{i-1}, 2^i]$ βρίσκουμε το ζητούμενο.

4 Άσκηση 4: Επαναφορά της ομαλότητας

4.1 Μοντελοποίηση Προβλήματος

Εύκολα γίνεται φανερό ότι για κάθε σημείο Εξτρεμιστών ή βάση, υπάρχει ένα και μόνο σημείο δυνάμεων στρατού που θα χρησιμοποιήσουμε για να επιτεθούμε: αυτό που απέχει τη μικρότερη απόσταση.

Επομένως, οι "σχέσεις επίθεσης" στρατού-εξτρεμιστών και στρατού-βάσεων μπορούν να αναπαρασταθούν από γράφο, του οποίου οι κορυφές είναι:

- στρατιωτικές βάσεις
- βάσεις εξτρεμιστών
- εξτρεμιστές

και οι ακμές (κατευθυνόμενες):

- βάση στρατού $\longrightarrow$ εξτρεμιστές, βάρος ίσο με το κόστος επίθεσης
- βάση στρατού $\longrightarrow$ βάση εξτρεμιστών, βάρος ίσο με το κόστος επίθεσης
- βάση εξτρεμιστών $\longrightarrow$ εξτρεμιστές, βάρος 0

Ο γράφος θα είναι, εμφανώς, DAG και κάθε source είναι βάση στρατού και κάθε sink είναι εξτρεμιστές. Επομένως, μπορούμε να συγχωνεύσουμε όλα τα sources σε μία κορυφή, διατηρώντας την ακεραιότητα της μοντελοποίησης. Ακολουθεί ένα παράδειγμα για οπτικοποίηση:

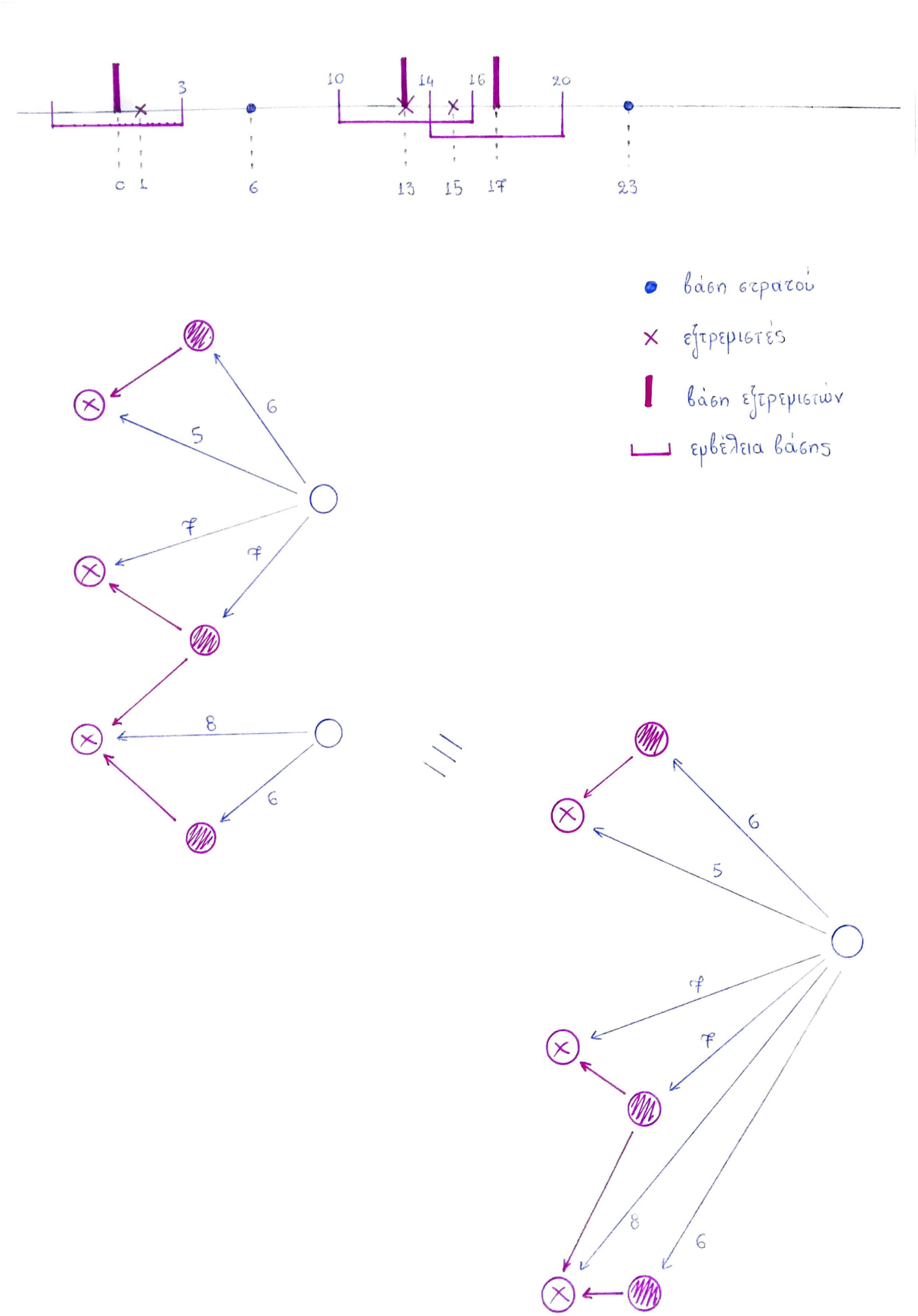

Figure 1: Παράδειγμα

Το ζητούμενο μας, είναι να ελαχιστοποιήσουμε το κόστος της συνολικής επίθεσης. Προτού προχωρήσουμε είναι σημαντικό να επισημάνουμε ορισμένα γεγονότα που διευκολύνουν την προσπάθειά μας.

- Για την αιχμαλώτιση ενός σημείου εξτρεμιστών μπορούμε είτε να τους αιχμαλωτίσουμε άμεσα είτε να καταρίψουμε όλες τις βάσεις που το υποστηρίζουν
- Για την αιχμαλώτιση ενός σημείου εξτρεμιστών θα χρησιμοποιήσουμε κόστος *το πολύ* όσο αυτό της άμεσης αιχμαλώτισης, εκτός κι αν με την κατάριψη βάσης μειώνουμε το συνολικό κόστος αιχμαλώτισης του συγκεκριμένου σημείου και τουλάχιστον ενός άλλου.
- Τα δύο παραπάνω σημαίνουν άμεσα ότι σε κάθε περίπτωση το συνολικό κόστος για την ολική αιχμαλώτιση θα είναι το πολύ ίσο με το να επιτεθούμε άμεσα σε κάθε σημείο εξτρεμιστών. Δηλαδή, αν το μοιράσουμε σε κάθε σημείο εξτρεμιστών, δεν υπάρχει τρόπος να κατανέμουμε *σε όλα* τα σημεία περισσότερο κόστος από αυτό της άμεσης αιχμαλώτισης.

Το τελικό συμπέρασμα των παραπάνω μας οδηγεί στην εξής σκέψη: το κόστος για την ολική αιχμαλώτιση είναι μια "ροή" μέσα στον γράφο μας, και οι "σωλήνες διοχέτευσης" προς τα σημεία εξτρεμιστών έχουν χωρητικότητα τόση όση η άμεση αιχμαλώτιση. Έτσι, ο γράφος μπορεί να πάρει μια πιο βολική, αλλά ισοδύναμη μορφή:

- Θεωρούμε όλα τα βάρη ακμών σαν χωρητικότητες σωλήνων
- για κάθε ακμή άμεσης αιχμαλώτισης, την αφαιρούμε και προσθέτουμε μια ακμή από το αντίστοιχο σημείο εξτρεμιστών σε έναν νέο κόμβο με χωρητικότητα ακμής ίση με το βάρος της αφαιρούμενης ακμής.
- μεταβάλλουμε όλες τις χωρητικότητες ακμών που εξέρχονται από βάση εξτρεμιστών και τις θετουμε ίσε με την χωρητικότητα της ακμής που εισέρχεται στην αντίστοιχη βάση εξτρεμιστών.
- επειδή ο τελικός γράφος έχει πολλά sinks, αλλά καθένα έχει έναν μοναδικό πατέρα, ενώνουμε όλα τα sinks σε ένα, χωρίς βλάβη της μοντελοποίησης.

Έτσι, το προηγούμενο παράδειγμα παίρνει την ακόλουθη μορφή:

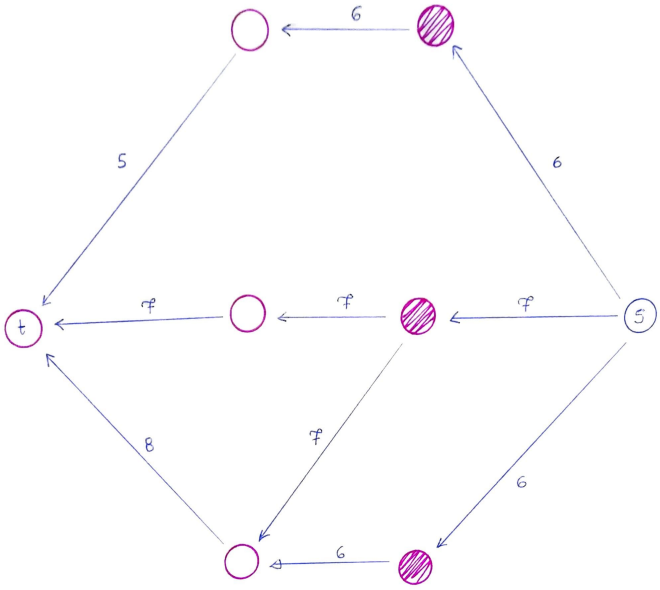

Figure 2: Μοντελοποιημένο παράδειγμα

4.2 Αναγωγή του προβλήματος

Θεωρώντας ότι οι ακμές του γράφου μας είναι χωρητικότητες των ακμών, μπορούμε να δούμε ότι το πρόβλημά μας μοιάζει πάρα πολύ, πλέον, με το πρόβλημα της μέγιστης ροής.

Αναγωγή

1. Από τα δεδομένα εισόδου κατασκευάζουμε τον γράφο της τελικής μορφής που περιγράφηκε
2. Λύνουμε το πρόβλημα max flow στον νέο γράφο και η απάντηση θα είναι ταυτόσημη με το ελάχιστο κόστος που ζητάμε

Μένει να αποδείξουμε ότι η αναγωγή είναι ορθή και να υπολογίσουμε το χρονικό κόστος της. Απο εκεί και πέρα, η πολυπλοκότητα εξαρτάται από τον χρόνο του αλγορίθμου που θα χρησιμοποιήσουμε για να επιλύσουμε το max flow (πχ Ford - Fulkerson με κάποια Edmond - Karp βελτίωση).

4.3 Ορθότητα Αναγωγής

Στο μοντελοποιημένο πρόβλημα έχουμε έναν γράφο με 3 κατηγορίες ακμών:

- αμκές $s \longrightarrow$ (κορυφή-βάση)$=x$
- ακμές $x \longrightarrow$ (κορυφή σημείο εξτρεμιστών)$=y$
- $y \longrightarrow t$

Παρατηρούμε ότι για να έχουμε εξόντωση ενός σημείου εξτρεμιστών θα πρέπει να κάνουμε απευθείας επίθεση στο σημείο ή να ρίξουμε όλες τις βάσεις που το υποστηρίζουν. Αυτό ισοδυναμεί με το εξής "κάθε κορυφή σημείο εξτρεμιστών πρέπει στον flow graph να έχει κορεσμένο ή την ακμή προς την t ή όλες τις ακμές που έρχονται από κορυφές x (όπως ορίστηκαν παραπάνω)". Άρα, πρακτικά

ψάχνουμε έναν συνδιασμό ακμών $y \longrightarrow t$ και $s \longrightarrow x$, ώστε να "καλύψουμε" όλες τις ακμές $x \longrightarrow y$. Επιπλέον, θέλουμε τις πιο φθηνές ακμές. Το ίδιο ακριβώς πρόβλημα λύνει το min-cut στον γράφο μας. Άρα αφού min-cut και max-flow έχουν την ίδια λύση, το πρόβλημά μας ανάγεται ορθά στο max-flow.

4.4 Πολυπλοκότητα Αναγωγής

Η μοντελοποίηση του προβλήματος με τον γράφο που περιγράφηκε παραπάνω μπορεί να γίνει σε πεπερασμένο και σταθερό πέρασμα των δεδομένων, δηλαδή γίνεται σε γραμμικό χρόνο. Επομένως, η πολυπλοκότητα του συνολικού προβλήματος ταυτίζεται με την πολυπλοκότητα του αλγορίθμου που επιλέγουμε για την επίλυση του max-flow.

5 Άσκηση 5: Λογιστικές Αλχημείες

Μπορούμε να κινηθούμε ως ακολούθως:

Λύνουμε το πρόβλημα για $T \leq T_0$, και αναζητούμε το T_0 με δυαδική αναζήτηση. Η δυαδική αναζήτηση θα έχει ως άνω όριο το b_{max}.

Για να λύνουμε το πρόβλημα για το εκάστοτε T, κατασκευάζουμε τον παρακάτω γράφο:

- προσθέτουμε κορυφές s και t
- για κάθε $b_i > 0$, προσθέτουμε ακμή $i \longrightarrow t$ χωρητικότητας b_i
- για κάθε $b_i < 0$, προσθέτουμε ακμή $s \longrightarrow i$ χωρητικότητας $-b_i$
- συνδέουμε όλες τις κορυφές με όλες τις άλλες, εξαιρώντας τις s και t, μεταξύ τους με ακμή χωρητικότητας T_0

5.1 Ορθότητα

Για να διαπιστώσουμε αν το εκάστοτε T_0 είναι αρκετό, λύνουμε το max flow πρόβλημα στον παραπάνω γράφο κι ελέγχουμε αν το αποτέλεσμα είναι ίσο με το άθροισμα όλων των θετικών εταιριών ή το απόλυτο του αθροίσματος των αρνητικών. Αν ισχύει αυτό, τότε το χρέος των αρνητικών προς τις θετικές έχει καλυφθεί με συναλλαγές που έχουν αξία το πολύ T_0. Δηλαδή, στις αποδεκτές περιπτώσεις, για κάθε εταιρία θα ισχύει για τις ενδιάμεσες ροές $b_i = f_{i_{in}} - f_{i_{out}}$, αφού οι ακμές από το s και προς το t θα είναι κορεσμένες.

5.2 Πολυπλοκότητα

Η πολυπλοκότητα θα είναι όση της λύσης του max-flow πολλαπλασιασμένη επί την πολυπλοκότητα της δυαδικής αναζήτησης για το T_0, δηλαδή:

$$O\big(|V|^3 log(b_{max})\big)$$

6 Άσκηση 6: Αναγωγές και NP-Πληρότητα

Για τις παρακάτω αποδείξεις θα πρέπει να αποδείξουμε δύο ισχυρισμούς:

1. το εκάστοτε πρόβλημα ανήκει στην κλάσση NP
2. και, ταυτόχρονα, ένα γνωστό NP-complete πρόβλημα ανάγεται στο δικό μας

6.1 Άθροισμα Υποσυνόλου κατά Προσέγγιση

Μπορούμε σε πολυωνυμικό χρόνο να ελέγξουμε αν μια λύση είναι πράγματι λύση στο πρόβλημά μας, οπότε αυτό ανήκει στην κλάση NP.

Μπορούμε να κάνουμε αναγωγή του Subset Sum που, ως γνωστόν, είναι NP-complete στο δικό μας "κατά προσέγγιση Subset Sum" ως εξής.

Έστω το instance του Subset Sum: $A = \{w_1, \ldots, x_n\}$ και B η παράμετρός του.

Πολλαπλασιάζουμε όλα τα στοιχεία του A με 2 και προκύπτει το σύνολο $A' = \{2w_1, \ldots, 2x_n\}$. Αν έχουμε αλγόριθμο που λύνει το Approximate Subset Sum με είσοδο το A' και παράμετρο το 2B και x = 1, τότε μπορούμε να αποφανθούμε αν υπάρχει υποσύνολο του A' με άθροισμα 2B ή 2B-1. Προφανώς, το 2B-1 δεν θα είναι ποτέ πιθανό άθροισμα, αφού όλα τα δυνατά αθροίσματα στοιχείων του A' θα είναι ζυγός αριθμός. Άρα αν ο αλγόριθμος επιστρέψει "NAI" τότε υπάρχει υποσύνολο του A' με άθροισμα 2B και, κατά συνέπεια, το ίδιο υποσύνολο με κάθε στοιχείο του διαιρεμένο με το 2, δίνει άθροισμα B $\Longrightarrow$ άρα και το Subset Sum έχει ως απάντηση "NAI" με είσοδο το A και παράμετρο B. Εντελώς όμοια συμπεράσματα έχουμε για το αν επιστραφεί "OXI" ως απάντηση.

Άρα το Subset Sum, που είναι NP-Complete, ανάγεται στο Approximate Subset Sum, ενώ το Approximate Subset Sum είναι στην NP. Άρα, το approximate subset sum είναι NP-Complete.

6.2 Κύκλος Hamilton κατά Προσέγγιση

Με δεδομένη μια λύση του προβλήματος μπορούμε σε γραμμικό χρόνο να διαπιστώσουμε την ορθότητα μιας λύσης, αφού μπορούμε να ελέγχουμε αν η δεδομένη ακολουθία κόμβων είναι κύκλος με επίσκεψη κόμβων από 1 έως 2 φορές.

Για να δείξουμε ότι το πρόβλημα είναι NP-Hard, αρκεί να δείξουμε ότι ένα NP-Hard πρόβλημα ανάγεται στο Approximate Hamilton Cycle. Θα κάνουμε, λοιπόν, αναγωγή του Hamilton Cycle στο Approximate Hamilton Cycle. Έχοντας αλγόριθμο που λύνει το Approximate Hamilton Cycle, τροποποιούμε ένα instance του Hamilton Cycle ως εξής συνδέουμε κάθε κορυφή u μια με νέα κορυφή v της οποίας μόνος γείτονας είναι η u. Τότε, η απάντηση του αλγορίθμου που αποφαίνεται για το αν ο νέος γράφος έχει Approximate Hamilton Cycle ταυτίζεται με το αν έχει Hamilton Cycle ο αρχικός γράφος. Κι αυτό, γιατί αν υπάρχει Approximate Hamilton Cycle στον νέο γράφο, τότε αυτός περνά από κάθε κορυφή που υπάρχει και στον παλιό γράφο ακριβώς 2 φορές (μία οταν φτάνουμε σε αυτή πρώτη φορά και μια δεύτερη όταν ξαναφτάνουμε σε αυτή επιστρέφοντας από την αντίστοιχη v κορυφή), αναγκαστικά. Άρα, αφαιρώντας όλες τις νέες κορυφές, και μειώνοντας κατά συνέπεια τις επισκέψεις σε κάθε κορυφή κατά ένα, προκύπτει κύκλος με μοναδική επίσκεψη σε κάθε κορυφή, δηλαδή κύκλος Hamilton.

6.3 Ικανοποιησιμότητα με Περιορισμούς

Εύκολα μπορούμε να επαληθεύοσυμε την εγκυρότητα ανάθεσης σε γραμμικό χρόνο, επομένως το πρόβλημα ανήκει στην κλάση NP.

Θα δείξουμε ότι το 3-CNF που είναι NP-Hard ανάγεται στο περιορισμένο 4-CNF. Θεωρούμε instance του 3-CNF. Προσθέτουμε σε κάθε clause ϕ του instance 3-CNF ένα literal ονόματι x, που δεν ανήκει στο σύνολο των αρχικών literals και είναι κοινό για κάθε clause. Ισχυριζόμαστε ότι αν για την καινούργια πρόταση ϕ' έχουμε απάντηση από το περιορισμένο 4-CNF, τότε έχουμε ταυτόσημη απάντηση για την αρχική πρόταση ϕ.

Αν η ϕ' ικανοποιείται υπό τους περιορισμούς, τότε έχει ένα (τουλάχιστον) literal ίσο με 1. Αν x = 0, τότε αφαιρώντας το x από κάθε πρόταση επιστρέφουμε στην ϕ, χωρίς βλάβη στην ικανοποιησιμότητά της. Αν x = 1, λόγω του περιορισμού κάποιο άλλο literal για κάθε clause θα είναι 0, οπότε αν αντιστρέψουμε όλες τις αληθοτιμές για κάθε clause, αναγκαστικά προκύπτει 0 στο x και κάποιος

άσσος σε κάποιο άλλο literal. Αφαιρούμε το x από κάθε clause και έχουμε μια ανάθεση που ικανοποιεί την πρόταση με 3 literals ανά clause.

Αν η ϕ ικανοποιείται τότε έχουμε έναν άσσο για να ικανοποιείται και η ϕ' για την ίδια ανάθεση μεταβλητών και x = 0 ικανοποιείται υπό τον περιορισμό ότι υπάρχει και 0 στα literals της ανάθεσης που την κάνει αληθή.

6.4 Επιλογή Ανεξάρτητων Υποσυνόλων

Έυκολα μπορούμε διατρέχοντας μια φορά το κάθε σύνολο να εξακριβώσουμε αν συναντάμε ποτέ ίδιο στοιχείο με κάποιο προηγούμενο. Οπότε εξακριβώνουμε την ορθότητα της λύσης σε γραμμικό χρόνο και το πρόβλημα ανήκει στην κλάση NP.

Στη συνέχεια, θα κάνουμε αναγωγή του Max Independent Set στο πρόβλημά μας.

Θεωρούμε ένα instance του max independent set. Ύστερα, για κάθε κορυφή του instance φτιάχνουμε ένα σύνολο στο S που περιέχει όλες τις προσπίπτουσες ακμές στην αντίστοιχη κορυφή στο G. Αντιστοιχίζουμε τη σταθερά k του προβλήματός μας σε αυτή του MIS και θεωρούμε το U ως το σύνολο ακμών E.

Εστω ότι έχουμε ανεξάρτητο σύνολο κορυφών με πληθάριθμο τουλάχιστον k. Κανένα ζεύγος κορυφών αυτού του συνόλου δε θα έχει κοινή ακμή κι, επομένως, καμία δυάδα των αντίστοιχων υποσυνόλων του προβλήματος δε θα έχει κοινό στοιχείο. Δηλαδή, όλα τα, τουλάχιστον k το πλήθος, υποσύνολα που δημιουργούμε είναι ξένα μεταξύ τους.

Αντιστρόφως, έστω ότι έχουμε τουλάχιστον k ξένα υποσύνολα του U. Εφόσον το καθένα αντιστοιχεί σε κορυφή του αρχικού γραφήματος, θα έχουμε ίδιου πλήθους σύνολο κορυφών χωρίς κοινές ακμές στο G, δηλαδή ένα ανεξάρτητο σύνολο με πληθάριθμο k.