Εργασία Αλγόριθμοι και Πολυπλοκότητα

Ασημάκης Κύδρος
3881
asimakis@csd.auth.gr

January 19, 2024

1 Αποδείξτε ότι το πρόβλημα του Προσδιορισμού Νικητή σε Συνδυαστικούς Πλειστηριασμούς (WDCA) είναι $\mathcal{NP}$-complete.

1. $WDCA \in \mathcal{NP}$: Με δεδομένη συλλογή προσφορών, μπορούμε να υπολογίσουμε την τομή όλων των προσφορών ανά 2 σε γραμμικό χρόνο με χρήση hashtables. Το άθροισμα των x_i υπολογίζεται επίσης σε πολυωνυμικό χρόνο.

2. $ISP \leq_P WDCA$: Με δεδομένη την είσοδο του ISP, έστω $G = (V, E), \quad k$, θέλουμε να αποφανθούμε αν υπάρχει υποσύνολο ξένων κορυφών μεγέθους τουλάχιστον k.

   Θέτουμε:

   - $\{I_1, I_2, \cdots, I_n\} \leftarrow E$ $\quad \mathcal{O}(1)$
   - $S_i \leftarrow \{(V_i, V_j), \cdots \quad \forall \quad (V_i, V_j)_{i \neq j} \in E)\} \quad \forall \quad V_i \in V$ $\quad \mathcal{O}(|V||E|)$
   - $x_i \leftarrow 1 \quad \forall \quad V_i \in V$ $\quad \mathcal{O}(|V|)$
   - $B \leftarrow k$ $\quad \mathcal{O}(1)$

   Ισχυρισμός: το ISP έχει λύση αν και μόνο αν το $WDCA$ έχει λύση.

   $\Rightarrow$ Αν το ISP έχει λύση, τότε υπάρχουν τουλάχιστον k κόμβοι του G που είναι ανά 2 ξένοι, δηλαδή δεν έχουν ανά 2 καμία κοινή ακμή. Επομένως, εφόσον το σύνολο ακμών κάθε κόμβου αντιστοιχίζεται σε ένα σύνολο προσφοράς με αξία 1, και το κατώφλι του budget B είναι k, θα βρεθεί η συλλογή των k προσφορών που αντιστοιχούν σε αυτούς τους ξένους κόμβους από το black box του $WDCA$, άρα θα επιστρέψει ΄ναι΄.

   $\Leftarrow$ Αν το back box του $WDCA$ επιστρέψει ΄ναι΄, τότε βρέθηκε συλλογή ανά 2 ξένων συνόλων με κέρδος τουλάχιστον B. Εφόσον κάθε σύνολο προσφοράς περιλαμβάνει όλες τις ακμές ενός κόμβου του G, αυτό σημαίνει ότι βρέθηκε συλλογή κόμβων που δεν έχουν κοινή ακμή ανά 2, επομένως είναι ανά 2 ξένοι. Κάθε προσφορά έχει bid 1, επομένως αφού η συνολική αξία βρέθηκε να είναι τουλάχιστον B, το πλήθος των ξένων κόμβων είναι τουλάχιστον k. Άρα αποτελεί λύση του ISP.

   Από (1) και (2) συνεπάγεται πως το $WDCA$ είναι $\mathcal{NP}$-complete. □

2 Βασιζόμενοι στην απόδειξη $\mathcal{NP}$-completeness του TSP με αναγωγή από το πρόβλημα $Hamiltonian\ Cycle$, αποδείξτε ότι δεν υπάρχει ρ-προσεγγιστικός αλγόριθμος για το TSP για $\rho > 1$.

Έστω ότι υπάρχει ρ-προσεγγιστικός αλγόριθμος για το TSP, δηλαδή επιστρέφει περιοδεία μήκους το πολύ ρ φορές μεγαλύτερη από την βέλτιστη ή ΄όχι΄ σε πολυωνυμικό χρόνο.

Εφόσον $Hamiltonian\ Cycle \leq_P Decision{-}TSP$ τότε και $Hamiltonian\ Cycle \leq_P Optimization{-}TSP$. Λαμβάνουμε την είσοδο του $Hamiltonian\ Cycle$ και στέλνουμε στον προσεγγιστικό αλγόριθμο τους n κόμβους ως πόλεις και

$$dist(v, v') = \begin{cases} 1 & if (v, v') \in E \\ \infty & else \end{cases}$$

Ο προσεγγιστικός αλγόριθμος επιστρέφει ΄όχι΄ ή λύση το πολύ ρ φορές μακρύτερη από τη βέλτιστη. Ωστόσο, κάθε λύση σε αυτό το στιγμιότυπο μπορεί να έχει μόνο μήκος n. Άρα έχει αποφασιστεί αν υπάρχει περιοδεία μήκους n ή όχι σε πολυωνυμικό χρόνο, επομένως και αν υπάρχει κύκλος $Hamilton$ σε πολυωνυμικό χρόνο. Άρα $\mathcal{P} = \mathcal{NP}$. Εκτός απροόπτου αυτό είναι άτοπο. □

3 Διατυπώστε το πρόβλημα απόφασης της Μεγιστοποίησης Ευημερίας (DWM) και αποδείξτε πως είναι $\mathcal{NP}$-complete.

***Decision Welfare Maximization problem**: Δεδομένου m αντικειμένων, W κατώφλι ευημερίας και n συμμετεχόντων, όπου ο καθένας χρησιμοποιεί δεδομένη θετική και μονότονη συνάρτηση ικανοποίησης u_i, να αποφασισθεί αν υπάρχει ανάθεση $S = \{S_1, S_2, \cdots S_n\}$ των αντικειμένων στους συμμετεχόντες, τέτοια ώστε*

$$S_i \cap S_j \quad \forall i, j \ni i \neq j \quad \wedge \quad \prod_{i=1}^{n} u_i(S_i) \geq W$$

1. $DWM \in \mathcal{NP}$: Με δεδομένη ανάθεση $S = \{S_1, S_2, \cdots S_n\}$ μπορεί να ελεγχθεί αν λύνεται το πρόβλημα σε πολυωνυμικό χρόνο, καθώς μπορεί να δειχθεί ότι όλα τα S_i είναι ανά 2 ξένα σε γραμμικό χρόνο με χρήση πινάκων κατακερματισμού, και το γινόμενο μπορεί να υπολογισθεί σε $\mathcal{O}(n^d m^k)$ χρόνο, απαιτώντας όλες οι u_i να είναι $\mathcal{O}(m^k)$ σε χρόνο (d, k σταθερές).

2. $Partition \leq_P DWM$: Με δεδομένη την είσοδο $\{a_1, a_2, \cdots\ a_m\}$, ζητείται να αποφανθούμε αν υπάρχει διχοτόμηση, τέτοια ώστε τα υποσύνολα να έχουν ίσο άθροισμα.

   Θέτουμε:

   - $\{I_1, I_2, \cdots I_m\} \leftarrow \{(1, a_1), (2, a_2), \cdots (m, a_m)\}^{\dagger}$ $\quad \mathcal{O}(m)$
   - $n \leftarrow 2$ $\quad \mathcal{O}(1)$
   - $u_i \leftarrow f(S_i) = \begin{cases} \sum_{(j,a_j)\in S_i} a_j & if \quad S_i \not\equiv \varnothing \\ 0 & else \end{cases}^{\ddagger}$ $\quad \mathcal{O}(1)$
   - $W \leftarrow \frac{1}{4}\left(\sum_{i=1}^{m} a_i\right)^2$ $\quad \mathcal{O}(m)$

   Ισχυρισμός: Το $Partition$ έχει λύση αν και μόνο αν το DWM έχει λύση.

   $\Leftarrow$ Άν το DWM επιστρέψει ναι, αυτό σημαίνει πως υπάρχει ανάθεση τέτοια ώστε

   $$\left(\sum_{(i,ai)\in S_1} a_i\right)\left(\sum_{(j,aj)\in S_2} a_j\right) \geq \frac{1}{4}\left(\sum_{i=1}^{m} a_i\right)^2$$

   Αν θέσουμε ένα από τα 2 αθροίσματα στο αριστερό κομμάτι της ανισότητας ως x, και το $\sum_{i=1}^{m} a_i = A$, τότε η ανισότητα γράφεται

   $$x(A - x) \geq \frac{A^2}{4}$$

   Το γινόμενο $x(A-x)$ μεγιστοποιείται στο $x = \frac{A}{2}$ με τιμή $\frac{A^2}{4}$. Εφόσον η ανάθεση που βρέθηκε αντιστοιχεί σε διχοτόμηση του συνόλου των ακεραίων, και ικανοποιεί το κατώφλι, αυτό σημαίνει ότι βρέθηκε διχοτόμηση με ίσα αθροίσματα, εφόσον το κατώφλι ισούται με τη μέγιστη τιμή της παραπάνω παράστασης. Άρα αποτελεί λύση του $Partition$.

   $\Rightarrow$ Αν το $Partition$ έχει λύση, αυτό σημαίνει ότι υπάρχει διχοτόμηση του συνόλου των ακεραίων με ίσα αθροίσματα. Επομένως, θα βρεθεί η ανάθεση που αντιστοιχεί σε αυτή τη διχοτόμηση από το black box του DWM, καθώς η παραπάνω ανισότητα θα ικανοποιηθεί, εφόσον μεγιστοποιείται για $x = A - x = \frac{A}{2}$. Άρα το DWM θα επιστρέψει ναι.

   Από (1) και (2) συνεπάγεται πως το DWM είναι $\mathcal{NP}$-complete. $\square$

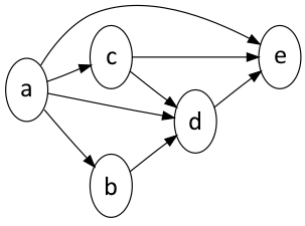
†: *Τα αντικείμενα ορίζονται έτσι ώστε κάθε ακέραιος στο σύνολο να θεωρείται ανεξάρτητος από τους άλλους και να μην απαλειφούν τυχόν διπλότυπα ακεραίων.*
‡: *Η u_i είναι προφανώς θετική, εφόσον κάθε $a_i \geq 0$, και επίσης είναι μονότονη, γιατί αν $S \subseteq T$, τότε το T περιέχει τους θετικούς ακέραιους του S και δυνιτικά κάποιους παραπάνω, επομένως το άθροισμά του θα είναι τουλάχιστον ίσο με αυτό του S.*

4 Δώστε αλγόριθμο πολυωνυμικού χρόνου για να αποφασίσετε αν ένας παίκτης έχει υποχρεωτική νίκη στη *Γεωγραφία σε ένα Γράφημα* όταν το G είναι DAG.

```
function score(v, M):
         v:  a node in the directed graph.
            Its children should be accessible.
         M: the hashmap of whether the starter from each position wins.

         if M[v] = null then
            if v has no children then
               M[v] ← false
            else
               M[v] ← ¬( ⋀_{u child of v} M[u] )
         return M[v]
end

function decide_winner(V, E, s):
         V: the set of verteces.
         E: the set of links.
         s:  the selected starting source point, s ∈ V.

         create hashmap M with keys the vertices
         and initialize all values to null.

         if score(s, M) then
            return "player 1 wins"
         else
            return "player 2 wins"
end
```

Ο παραπάνω αλγόριθμος είναι $\mathcal{O}(|V|)$ σε χώρο καθώς το hashmap έχει μήκος $|V|$, και $\mathcal{O}(|V|+|E|)$ σε χρόνο, διότι η δημιουργία του hashmap παίρνει $\mathcal{O}(|V|)$ χρόνο και η `score` καλείται ακριβώς μια φορά για κάθε ακμή, επομένως το `score(s, M)` παίρνει $\mathcal{O}(|E|)$.

Το τελευταίο φαίνεται από το παρακάτω παράδειγμα:



Το παράδειγμα αυτό είναι αντιπροσωπευτικό, διότι παρόλο που ένα DAG μπορεί να έχει πολλαπλές πηγές (sources) και πολλαπλά τέλη (shinks), το σημείο εκκίνησης εδώ είναι πάντα συγκεκριμένο, επομένως όλα τα υπόλοιπα sources αγνοούνται καθώς είναι απρόσιτα.

Με σημείο εκκίνησης το **a**, εφόσον έχει παιδιά, καλείται η `score` πρώτα για το **b**. Αυτό με τη σειρά του καλεί την `score` για το πρώτο και μόνο του παιδί **d**, και αυτό για το **e**. Το **e** δεν έχει παιδιά, επομένως βαθμολογείται στο false και η αναδρομική κλίση επιστρέφει στο **d**, το οποίο βαθμολογείται στο true και επιστρέφει στο **b**, το οποίο βαθμολογείται στο false και επιστρέφει στο **a**. Από το **a** καλείται ξανά η `score` για το επόμενο παιδί, **d**, το οποίο πλέον έχει βαθμολογηθεί ήδη και τερματίζει νωρίς επιστρέφοντας true. Ξανά από το **a** καλείται η `score` για το **c**, το οποίο κάνει 2 αναδρομικές κλίσεις της `score` για τα παιδιά του **d** και **e**, τα οποία θα τερματίσουν νωρίς επιστρέφοντας true και false αντίστοιχα, επομένως το **c** θα βαθμολογηθεί στο true και θα επιστρέψει στο **a**. Τέλος από το **a** καλείται ξανά η `score` για το τελευταίο παιδί **e**, το οποίο επιστρέφει false. Το **a** βαθμολογείται στο true και ο αλγόριθμος τερματίζει.

Οι κλίσεις της `score` ταυτίζονται με τις ακμές στο διάγραμμα, επομένως το `score(a, M)` είναι $\mathcal{O}(|E|)$. □

Ο αλγόριθμος αποφασίζει σωστά διότι κάθε κόμβος βαθμολογείται με το αν ο εισαγώμενος σε αυτόν τον κόμβο παίκτης έχει forced win με perfect play. Όταν ένας κόμβος δεν έχει παιδιά, όποιος παίκτης μπει σε αυτόν χάνει κατευθείαν γιατί δεν έχει κίνηση, άρα βαθμολογείται στο false. Όταν ένας κόμβος έχει τουλάχιστον ένα παιδί βαθμολογημένο στο false, τότε ο υποδεχόμενος παίκτης θα παίξει εκεί γιατί ξέρει ότι ο επόμενος παίκτης χάνει, και άρα ο κόμβος θα βαθμολογηθεί στο true. Έστω ότι αυτό το σκεπτικό ισχύει για όλα τα παιδιά της ρίζας. Αν υπάρχει έστω και ένα παιδί βαθμολογημένο στο false, είναι προς το συμφέρον του πρώτου παίκτη να παίξει εκεί γιατί ισχύει λόγω υπόθεσης ότι ο επόμενος παίκτης χάνει. Αλλιώς, ο πρώτος χάνει, διότι ισχύει λόγω υπόθεσης πώς σε κάθε κόμβο-παιδί ο επόμενος παίκτης έχει νίκη.

Άρα με επαγωγή βγαίνει πως αν ο πρώτος παίκτης έχει σίγουρη νίκη, η εκκίνηση θα βαθμολογηθεί στο true. □

5 **α. Δεδομένου του αλγορίθμου** `Explore`**, αποδείξτε ότι επιστρέφει 'ναι' αν και μόνο αν υπάρχει ικανοποιούσα ανάθεση τιμών Φ' σε απόσταση το πολύ** d **από την Φ και δώστε μια ανάλυση της χρονικής πολυπλοκότητας ως συνάρτηση των** $n,\ \ d$**.**
β. Χρησιμοποιείστε τον αλγόριθμο για να λύσετε το $3-SAT$ **σε χρόνο** $\mathcal{O}(p(n)\sqrt{3}^n)$**.**

α1.

$\Rightarrow$ Έστω ότι υπάρχει ικανοποιούσα Φ' σε απόσταση το πολύ d. Αν αυτή είναι η αρχική Φ, θα επιστραφεί 'ναι' στην πρώτη κλίση. Αλλιώς, θα ικανοποιηθεί μια πρόταση και θα ελεγθούν τρεις Φ' με απόσταση το πολύ $d-1$. Σε κάθε αναδρομική κλίση αυξάνονται οι ικανοποιημένες προτάσεις και μειώνεται το ανώφλι επιτρεπώμενης απόστασης, επομένως κάποια στιγμή θα βρεθεί η Φ'. Άρα η `Explore` θα επιστρέψει 'ναι'.

$\Leftarrow$ Έστω ότι η `Explore` επιστρέφει 'ναι'. Αυτό συμβαίνει όταν η αρχική Φ είναι κατευθείαν ικανοποιούσα, ή οταν μια παραγώμενη Φ' μπαίνει σε νέα αναδρομική κλίση με μικρότερο d και είναι κατευθείαν ικανοποιήσιμη. Εφόσον το d συνεχώς μειώνεται και δεν πέφτει ποτέ κάτω από 0, αυτό σημαίνει ότι έχει βρεθεί ικανοποιήσιμη ανάθεση με απόσταση το πολύ d από την Φ. □

α2.

Ο αλγόριθμος έχει bottleneck τον πρώτο έλεγχο, όπου ελέγχεται αν η Φ εισόδου είναι ικανοποιήσιμη ανάθεση. Αυτό παίρνει χρόνο $\mathcal{O}(|C_i|)$, καθώς στην χειρότερη περίπτωση θα ελεχθούν όλες οι προτάσεις. Οι προτάσεις, για n μεταβλητές, είναι το πολύ

$$\frac{8}{6}n(n-1)(n-2)$$

Αυτό ισχύει διότι κάθε πρόταση είναι τριπλέτα των n μεταβλητών(δεικτών). Κάθε μεταβλητή έχει μια 'ναι', x_i, και μια 'όχι', $\overline{x_i}$, έκδοση. Κάθε δείκτης εμφανίζεται το πολύ μια φορά σε κάθε τριπλέτα επομένως οι πιθανές τριπλέτες των δεικτών είναι

$$\binom{n}{3} = \frac{n!}{3!(n-3)!} = \frac{n(n-1)(n-2)}{6}$$

και εφόσον κάθε δείκτης έχει 2 εκδοχές και υπάρχουν 3 μεταβλητές αυστηρά ανά πρόταση, κάθε τέτοια τριπλέτα έχει $2^3 = 8$ εκδοχές.

Επομένως ο χρόνος του πρώτου ελέγχου φράσσεται από το $\mathcal{O}(n^3)$. Έτσι, όλος ο αλγόριθμος έχει την εξής μορφή:

$$T(n,k) \leq \begin{cases} 3T(n,k-1) + \mathcal{O}(n^3) & k \neq 0 \\ \mathcal{O}(1) & k = 0 \end{cases}$$

άρα

$$\begin{aligned} T(n,d) &= 3^d\mathcal{O}(1) + \sum_{i=0}^{d-1} 3^i\mathcal{O}(n^3) \\ \Rightarrow T(n,d) &= \mathcal{O}(3^d) + \frac{3^d-1}{2}\mathcal{O}(n^3) \\ \Rightarrow T(n,d) &= \mathcal{O}(3^d n^3) \end{aligned}$$

β.

Εφόσον `Explore(`Φ`,` n`)` είναι πανάκριβο, και `Explore(`Φ`,` d`)` $\mathcal{O}(3^d n^3)$, είναι αρκετά προφανές ότι ο αλγόριθμος μπορεί να ψάξει μέχρι απόσταση το πολύ $\frac{n}{2}$. Αυτό σημαίνει ότι το πολύ $\frac{n}{2}$ μεταβλητές μπορεί να είναι διαφορετικές μεταξύ της πιθανής ικανοποιούσας και της αρχικής, οι άλλες πρέπει να είναι ίδιες.

Τρέχουμε `Explore(`Φ_0`,` $\frac{n}{2}$`) or Explore(`Φ_1`,` $\frac{n}{2}$`)` για Φ_0 η ανάθεση όλων στο 0 και Φ_1 κατ΄ αντιστοιχία. Το πρώτο εξετάζει μέχρι την ανάθεση που οι μισές μεταβλητές ικανοποιούν και οι άλλες είναι στο 0, και το δεύτερο το ίδιο θέτοντας τις άλλες στο 1. Ξεκινώντας από τα αντίθετα άκρα, η καθεμιά εξετάζει τις μισές από τις πιθανές λύσεις. Επομένως, αν μια επιστρέψει ναι, τότε υπάρχει λύση για το $3-SAT$. Ο αλγόριθμος αυτός έχει προφανώς πολυπλοκότητα $\mathcal{O}(\sqrt{3}^n n^3)$.

6 Δώστε έναν 3-προσεγγιστικό αλγόριθμο για το πρόβλημα Εξισορρόπησης Φορτίου όταν υπάρχουν k γρήγορες μηχανές διπλάσιας ταχύτητας και m αργές μηχανές κανονικής ταχύτητας.

Τρέχουμε LPT. Όταν υπάρχουν m πανομοιότυπες μηχανές, ισχύουν τα εξής φράγματα για το L^*:

$$L^* \geq 2t_{m+1}, \quad L^* \geq \frac{1}{m}\sum_{i=1}^{n} t_i$$

Εδώ, καθώς οι k μηχανές διαιρούν τον χρόνο εκτέλεσης μερικών εργασιών, έχουμε:

$$L^* \geq 2\frac{t_{m+1}}{2} = t_{m+1} \geq t_j \quad \forall j \quad (1), \quad L^* \geq \frac{1}{k+m}\sum_{i=1}^{n} \frac{t_i}{2} \quad (2)$$

(worst-case μπορεί να υπάρχουν μόνο γρήγορες μηχανές).

Άρα, για το bottleneck L_i και την τελευταία εργασία j που δέχεται ισχύει:

$$\begin{aligned}
&L_i - t_j \leq L_d \quad \forall d = 1, 2, \cdots m+k \Rightarrow \\
&L_i - t_j \leq \frac{1}{k+m}\sum_{d=1}^{k+m} L_d \\
&\qquad \leq \frac{1}{k+m}\sum_{d=1}^{n} t_d \\
&\qquad \leq \frac{2}{k+m}\sum_{d=1}^{n} \frac{t_d}{2} \overset{(2)}{\Longrightarrow} \\
&L_i - t_j \leq 2L^* \quad (3)
\end{aligned}$$

Επομένως μέσω (1) και (3): $L = L_i = L_i - t_j + t_j \leq 2L^* + L^* \leq 3L^*$.

7 **Έστω ένα αγαθό σε δεδομένη ποσότητα $B \geq 0$. Καθένας από n αγοραστές προτίθεται να πληρώσει $b_j \geq 0$ ανά μονάδα αγαθού που του εκχωρείται. Ζητείται να υπολογισθεί η ποσότητα αγαθού που θα λάβει κάθε αγοραστής ώστε να μεγιστοποιηθεί η ελάχιστη πληρωμή μεταξύ των αγοραστών. Να γραφεί το αντίστοιχο γραμμικό πρόγραμμα και το δυϊκό του.**

Το primal πρόγραμμα μοιάζει κάπως έτσι:

$$\begin{aligned} &\text{maximize } P \\ &\text{subject to } \sum_{i=1}^{n} x_i = B \\ &\qquad x_i b_i \geq P \quad \forall i = 1, 2, \cdots n \\ &\qquad x_i \geq 0 \quad \forall i = 1, 2, \cdots n \end{aligned}$$

καθώς το άθροισμα των ποσοτήτων που λαμβάνουν οι αγοραστές δεν γίνεται να είναι περισσότερο από τη διαθέσιμη ποσότητα, αλλά ούτε λιγότερη, αφού δεν είναι λογικό να μείνει κάποια ποσότητα του αγαθού στο ράφι. Η ποσότητα που λαμβάνει ο καθένας πρέπει προφανώς να είναι μη αρνητική, και ο άλλος περιορισμός θέτει στο P τον ρόλο της ελάχιστης πληρωμής, όπως και θέλουμε.

Για να το μετατρέψουμε σε δυϊκό, πρέπει να το φέρουμε σε κανονική μορφή. Ισοδύναμο του παραπάνω είναι το εξής:

$$\begin{aligned} &\text{maximize } P \\ &\text{subject to } \sum_{i=1}^{n} x_i \leq B \\ &\qquad \sum_{i=1}^{n} -x_i \leq -B \\ &\qquad -x_i b_i \leq -P \quad \forall i = 1, 2, \cdots n \\ &\qquad x_i \geq 0 \quad \forall i = 1, 2, \cdots n \end{aligned}$$

άρα θέτουμε:

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \\ 1 \end{bmatrix}, \quad \vec{c} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ P \end{bmatrix}, \quad A = \begin{bmatrix} 1 & 1 & \cdots & 1 & 0 \\ -1 & -1 & \cdots & -1 & 0 \\ -b_1 & 0 & \cdots & 0 & 0 \\ 0 & -b_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & -b_n & 0 \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} B \\ -B \\ -P \\ -P \\ \vdots \\ -P \end{bmatrix}$$

Επομένως το δυϊκό βρίσκεται ξεδιπλώνοντας το

$$\begin{aligned} &\text{minimize } \vec{b}^T \vec{y} \\ &\text{subject to } A^T \vec{y} \geq \vec{c} \\ &\qquad\qquad \vec{y} \geq \vec{0} \end{aligned}$$

άρα έχουμε

$$\begin{aligned} &\text{minimize } B(y_1 - y_2) - P \sum_{i=3}^{n+2} y_i \\ &\text{subject to } P \leq 0 \\ &\qquad\qquad b_i y_{i+2} \leq y_1 - y_2 \quad \forall i = 1, 2, \cdots n \\ &\qquad\qquad y_i \geq 0 \quad \forall i = 1, 2, \cdots n \end{aligned}$$

Θέτωντας το $y_1 - y_2 = y_0$ και αλλάζοντας την αρίθμηση καταλήγουμε στο

$$\begin{aligned} &\text{minimize } By_0 - P \sum_{i=1}^{n} y_i \\ &\text{subject to } P \leq 0 \\ &\qquad\qquad b_i y_i \leq y_0 \quad \forall i = 1, 2, \cdots n \\ &\qquad\qquad y_0 \in \mathbb{R}, \quad y_i \geq 0 \quad \forall i = 1, 2, \cdots n \end{aligned}$$

8 Απαντήστε στα ερωτήματα των διαφανειών του Αλγορίθμου του Χριστοφίδη, δηλαδή

1. Γιατί υπάρχει πάντα περιοδεία Euler; και

2. Γιατί πρέπει $|W|$ άρτιο για να υπάρχει τέλειο ταίριασμα;

1. Περιοδεία Euler για να υπάρχει πρέπει το γράφημα να είναι συνεκτικό και κάθε κόμβος να έχει άρτιο βαθμό. Το πρώτο ισχύει πάντα διότι το πολυγράφημα αποτελεί επέκταση συνεκτικού δέντρου, επομένως είναι σίγουρα συνεκτικό. Το δεύτερο ισχύει διότι το πολυγράφημα Η είναι άθροισμα των T* και M, όπου M τέλειο ταίριασμα ελάχιστου κόστους **μόνο** στους κόμβους περιττού βαθμού του T*. Επομένως κάθε τέτοιος κόμβος θα λάβει ακριβώς μια ακόμα ακμή στο πολυγράφημα και πλέον θα έχει άρτιο βαθμό, όπως και οι υπόλοιποι. Άρα οι προϋποθέσεις καλύπτονται και θα υπάρχει πάντα περιοδεία Euler.

2. Τέλειο ταίριασμα σε γράφο $G = (V, E)$ είναι $M \subseteq E$ τέτοιο ώστε κάθε κόμβος $V_i \in V$ να έχει ακριβώς μια ακμή στο M. Επομένως οι ακμές στο M ενώνουν η καθεμία ένα μοναδικό ζεύγος κόμβων. Άρα οι περιττού βαθμού κόμβοι πρέπει να έρχονται σε ακέραιο πλήθος ζευγών και επομένως πρέπει να είναι άρτιοι στο πλήθος.