Σχολή
Ηλεκτρολόγων Μηχανικών
και
Μηχανικών Υπολογιστών

Αλγόριθμοι και Πολυπλοκότητα

4η Σειρά Ασκήσεων

Μιχαλάκη Ελένη-Αικατερίνη Α.Μ. 03110126

Ακαδημαϊκό έτος 2014-2015

Άσκηση 1: Επιβεβαίωση και Αναπροσαρμογή Συντομότερων Μονοπατιών

(α)

Θα χρησιμοποιήσουμε μια παραλλαγή του αλγορίθμου BFS. Αρχικοποιούμε τον πίνακα προγόνων parent ώστε να έχει NULL σε όλες τις θέσεις. Έστω ότι βρισκόμαστε στην κορυφή v_i και έστω v_j γειτονική κορυφή. Τότε έχουμε τις εξής περιπτώσεις:

- $\delta_i + w_{ij} < \delta_j$. Στην περίπτωση αυτή οι δοσμένες αποστάσεις είναι λάθος, αφού έχουμε βρει μικρότερη απόσταση από την υποτιθέμενη.
- $\delta_i + w_{ij} = \delta_j$. Στην περίπτωση αυτή υποθέτουμε ότι οι αποστάσεις είναι σωστές. Βάζουμε την v_j στις εξερευνημένες κορυφές και θέτουμε parent[j] = i. Συνεχίζουμε κανονικά την εκτέλεση του αλγορίθμου.
- $\delta_i + w_{ij} > \delta_j$. Στην περίπτωση αυτή δεν προσθέτουμε την v_j στη λίστα με τις εξερευνημένες κορυφές, δεν ανανεώνουμε τον πίνακα προγόνων και συνεχίζουμε κανονικά την εκτέλεση του αλγορίθμου. (Θα περιμένουμε να ανακαλύψουμε την v_j από κάποια άλλη κορυφή για την οποία θέλουμε να ισχύει $\delta_i + w_{ij} = \delta_j$ προκειμένου οι αποστάσεις να είναι σωστές.)

Αφού ολοκληρωθεί ο BFS ελέγχουμε τον πίνακα προγόνων parent. Εάν σε κάποια θέση του (εκτός της πρώτης) υπάρχει NULL σημαίνει ότι για την κορυφή αυτή δεν υπάρχει μονοπάτι με την υποτιθέμενη απόσταση, επομένως οι αποστάσεις είναι λάθος. Εάν σε καμία θέση (εκτός της πρώτης) δεν υπάρχει NULL σημαίνει ότι οι δοσμένες αποστάσεις είναι σωστές και επομένως επιστρέφουμε τον πίνακα προγόνων ο οποίος αποτελεί ένα δέντρο συντομότερων μονοπατιών με ρίζα τη v_1.

(β)

Έαν μειώσουμε το μήκος της ακμής $e = (x, y)$ σε $w'(x, y) < w(x, y)$ τότε θα μειωθούν και οι αποστάσεις $d(v_i, v_j)$ για τις οποίες το συντομότερο μονοπάτι περιέχει την ακμή e. Για να βρούμε εάν μια απόσταση χρειάζεται να μειωθεί αρκεί να τη συγκρίνουμε με το νέο μήκος του μονοπατιού που περιλαμβάνει την e, το οποίο θα είναι ίσο με $d(v_i, x)+w'(x, y)+d(y, v_j)$. Εάν δηλαδή ισχύει $d(v_i, v_j) > d(v_i, x)+w'(x, y)+d(y, v_j)$ τότε θέτουμε $d(v_i, v_j) = d(v_i, x) + w'(x, y) + d(y, v_j)$, αλλιώς η απόσταση δεν χρειάζεται αναπροσαρμογή. Έχουμε n^2 ζεύγη κορυφών και για το καθένα ο παραπάνω έλεγχος γίνεται σε $O(1)$, επομένως η πολυπλοκότητα του αλγορίθμου είναι $O(n^2)$.

(γ)

Εάν αυξήσουμε το μήκος μιας ακμής e δεν μπορούμε να εργαστούμε όπως στο προηγούμενο ερώτημα, επειδή υπάρχει η πιθανότητα το συντομότερο μονοπάτι μεταξύ δύο κορυφών το οποίο αρχικά περιλάμβανε την ακμή e να μην την περιλαμβάνει μετά την αύξηση του βάρους της και να πρέπει να αναζητηθεί ένα καινούριο μονοπάτι. Συγκεκριμένα, εάν ισχύει $d(v_i, v_j) = d(v_i, x)+w'(x, y)+d(y, v_j)$ δεν χρειάζεται κάποια αλλαγή. Εάν όμως ισχύει $d(v_i, v_j) < d(v_i, x) + w'(x, y) + d(y, v_j)$ τότε χρειάζεται να συγκρίνουμε το $d(v_i, v_j)$ με τα μήκη όλων των μονοπατιών που συνδέουν τις δύο κορυφές και να το θέσουμε ίσο με το μικρότερο από αυτά.

Άσκηση 2: Σύστημα Ανισοτήτων και Μετατροπή Σταθερών Όρων

(α)

Δημιουργούμε ένα (κατευθυνόμενο) γράφημα G(V,E) με n κορυφές καθεμια από τις οποίες αντιστοιχεί σε μία μεταβλητή x_i. Επίσης για κάθε ανισότητα $x_i - x_j \leq b_{ij}$ ενώνουμε τις κορυφές v_j και v_i με μια ακμή βάρους b_{ij}. Τότε το σύστημα S θα είναι ικανοποιήσιμο εάν το G δεν περιέχει αρνητικούς κύκλους.

Απόδειξη:
(Ευθύ) Έστω ότι το γράφημα G περιέχει αρνητικό κύκλο στον οποίο ανήκουν οι κορυφές $v_1, v_2, ..., v_k$. Τότε εάν προσθέσουμε κατά μέλη όλες τις ανισότητες στις οποίες αντιστοιχούν οι ακμες αυτού του κύκλου θα έχουμε:
$x_1 - x_2 + x_2 - x_3 + ... + x_{k-1} + x_k \leq b_{12} + b_{23} + ... + b_{k-1k}$, δηλαδή
$0 \leq b_{12} + b_{23} + ... + b_{k-1k} < 0$, που είναι προφανώς άτοπο.

(Αντίστροφο) Έστω τώρα ότι το γράφημα δεν περιέχει αρνητικό κύκλο. Προσθέτουμε στο G μια κορυφή s την οποία συνδέουμε με όλες τις υπόλοιπες κορυφές με ακμές μηδενικού βάρους. Από την κορυφή s τρέχουμε τον αλγόριθμο Bellman-Ford και βρίσκουμε τις συντομότερες αποστάσεις προς κάθε μία από τις υπόλοιπες κορυφές. Θέτουμε κάθε μεταβλητή x_i ίση με την απόσταση $d(s, v_i)$ που βρήκαμε. Τότε όμως λόγω τριγωνικής ανισότητας θα ισχύει $\forall (v_j, v_i) \in E : d(s, v_i) - d(s, v_j) \leq w(v_j, v_i) \Rightarrow x_i - x_j \leq b_{ij}$, δηλαδή το σύστημα είναι ικανοποιήσιμο.

Επομένως για να διαπιστώσουμε εάν ενα σύνολο S είναι ικανοποιήσιμο ή όχι αρκεί να χρησιμοποιήσουμε τον αλγόριθμο Bellman-Ford για το παραπάνω γράφημα. Εάν ο αλγόριθμος βρει αρνητικό κύκλο, το σύνολο S δεν είναι ικανοποιήσιμο. Αλλιώς ο αλγόριθμος επιστρέφει τα συντομότερα μονοπάτια από την s προς κάθε κορυφή, οι τιμές των οποίων είναι μια λύση του S. Το γράφημα που χρησιμοποιήσαμε έχει $n + 1$ κορυφές (όσες είναι οι άγνωστες μεταβλητές +1 που προσθέσαμε) και $n + m$ ακμές (m λόγω των ανισοτήτων και n αυτές που προσθέσαμε από την επιπλέον κορυφή s), επομένως χρειαζόμαστε χρόνο $O(2n + m)$ για την κατασκευή του. Ο αλγόριθμος Bellman-Ford τρέχει σε χρόνο $O((n)(n + m)) = O(n^2 + nm)$. Επειδή όμως οι n ακμές που προσθέσαμε έχουν όλες μηδενικά βάρη μπορούμε να θέσουμε αρχική απόσταση από την s προς κάθε κορυφή το 0 (αφού οι n ακμές συνδέουν την s με κάθε άλλη κορυφή) και στη συνέχεια να τρέξουμε τον Bellman-Ford αλλά ελέγχοντας μόνο τις m ακμές σε κάθε επανάληψη. Με αυτό τον τρόπο η πολυπλοκότητα του αλγορίθμου γίνεται $O(nm)$.

(β)

Έχουμε δύο περιπτώσεις για το σύστημα S:

- Το S δεν είναι ικανοποιήσιμο. Τότε σύμφωνα με τα παραπάνω το γράφημα που θα αντιστοιχεί στο S θα περιέχει κάποιον αρνητικό κύκλο. Επομένως δεν γίνεται να μετατρέψουμε το S σε ένα ισοδύναμο S' με θετικά βάρη, αφού το γράφημα που θα αντιστοιχεί στο S' προφανώς δεν θα περιέχει κανέναν αρνητικό κύκλο και επομένως αυτό θα είναι ικανοποιήσιμο οπότε και δεν θα είναι ισοδύναμο με το S.

- Το S είναι ικανοποιήσιμο. Τότε θέλουμε να μετατρέψουμε το γράφημα G που αντιστοιχεί στο S σε ένα νέο γράφημα G' με τέτοιο τρόπο ώστε τα βάρη όλων των ακμών να είναι θετικά και τα συντομότερα μονοπάτια του να παραμείνουν ίδια, ώστε το S' να είναι ικανοποιήσιμο. Αυτό μπορεί να γίνει εφαρμόζοντας ένα μετασχηματισμό παρόμοιο με αυτόν που εφαρμόζει ο αλγόριθμος Johnson. Δηλαδή το βάρος κάθε ακμής (v_j, v_i) θα γίνει $w'(v_j, v_i) = w(v_j, v_i) + d(s, v_j) - d(s, v_i)$, άρα οι σταθεροί

όροι του S' θα είναι: $b'_{ij} = b_{ij} + x_j - x_i$. Αυτό μπορεί να γίνει εάν για κάθε κορυφή v_i καλέσουμε την Change(S,i,k) με k την απόσταση της κορυφής v_i από την αρχική κορυφή s. Επομένως για να μετατρέψουμε το S σε ένα ισοδύναμο σύστημα S' με θετικούς σταθερούς όρους αρκεί να βρούμε μια λύση του S και στη συνέχεια να καλέσουμε την Change n φορές.

Επομένως ένας αλγοριθμος που θα αποφασίζει εάν ένα σύστημα S μπορεί να μετατραπεί σε ένα ισοδύναμο S' με θετικούς σταθερούς όρους αρκεί να ελέγχει εάν το S είναι ικανοποιήσιμο. Για το σκοπό αυτό μπορούμε να εφαρμόσουμε αλγόριθμο Bellman-Ford όπως είπαμε στο (α) με πολυπλοκότητα $O(nm) = O(n^2 log^2 n)$.

Άσκηση 3: Συντομότερα Μονοπάτια με Δύο Κριτήρια

(α)

Δημιουργούμε δύο πανομοιότυπα αντίγραφα του αρχικού γραφήματος G, έστω G0 και G1. Εάν v μια κορυφή στο αρχικό γράφημα έστω v0 η αντίστοιχη κορυφή στο G0 και v1 στο G1. Εργαζόμαστε ως εξής:

1. Αντικαθιστούμε κάθε κόκκινη ακμή (v0, u0) στο G0 με μια κόκκινη ακμή (v0, u1) ίδιου μήκους.

2. Διαγράφουμε όλες τις κόκκινες ακμές στο G1. Δηλαδή τώρα τα G0, G1 περιέχουν (ως "εσωτερικές" ακμές) μόνο τις πράσινες ακμές του αρχικού γραφήματος και οι κόκκινες ακμές χρησιμοποιούνται μόνο για να πάμε από το G0 στο G1. Συνολικά έχουμε κατασκευάσει ένα νέο γράφημα που αποτελείται από τα G0, G1.

3. Εφαρμόζουμε αλγόριθμο Dijkstra ξεκινώντας από την κορυφή s0 (στο υπογράφημα G0).

Για κάθε μονοπάτι που βρίσκει ο αλγόριθμος υπάρχουν δύο επιλογές: είτε να είναι μόνο στο G0, οπότε και θα αποτελείται μόνο από πράσινες ακμές, είτε να περάσει κάποια στιγμή στο G1 οπότε και θα περάσει από μία κόκκινη ακμή και μετά θα συνεχίσει πάλι σε πράσινες αφού έχουμε σβήσει τις κόκκινες από το G1 και δεν υπάρχει τρόπος να ξαναπάει στο G0 και να ξαναανέβει (οπότε και θα ξαναπερνούσε από κόκκινη). Επομένως τα συντομότερα μονοπάτια που βρίσκει ο Dijksta περιέχουν το πολύ μία κόκκινη ακμή.

Μπορούμε για G0 να χρησιμοποιήσουμε το αρχικό γράφημα G οπότε αρκεί να κατασκευάσουμε ένα αντίγραφό του σε χρόνο $O(n + m)$. Στη συνέχεια εφαρμόζουμε Dijkstra στο γράφημα G0+G1 το οποίο συνολικά έχει 2n κορυφές και 2m ακμές, επομένως χρειάζεται χρόνο $O(2m + 2nlog(2n))$, που είναι και η πολυπλοκότητα του αλγορίθμου μας.

(β)

Για τη γενική περίπτωση να θέλουμε τα συντομότερα μονοπάτια να περιέχουν το πολύ k κόκκινες ακμές μπορούμε να γενικεύσουμε τον παραπάνω αλγόριθμο ως εξής: Αντί για ένα αντίγραφο του αρχικού γραφήματος δημιουργούμε k αντίγραφα, επομένως έχουμε συνολικά τα k+1 σε πλήθος υπογραφήματα $G_0, G_1, ..., G_k$. Όπως και πριν αντικαθιστούμε τις κόκκινες ακμές του υπογραφήματος G_i, (v_i, u_i) με αντίστοιχες (v_i, u_{i+1}) (από το επίπεδο i στο επόμενο). Στο τελευταίο επίπεδο G_k διαγράφουμε τις κόκκινες ακμές. Στη συνέχεια όπως και στο προηγούμενο ερώτημα εφαρμόζουμε αλγόριθμο Dijksta ξεκινώντας από την αρχική κορυφή στο G_0. Αντίστοιχα με πριν, τα μονοπάτια θα περιέχουν μέχρι k

κόκκινες ακμές (αφού άμα φτάσουν στο τελευταίο επίπεδο έχοντας περάσει από k κόκκινες ακμές δεν μπορούν να επιλέξουν κάποια κόκκινη αφού τις έχουμε διαγράψει).

Χρειαζόμαστε χρόνο $O(k(n+m))$ για να κατασκεύασουμε το γράφημα στο οποίο θα εφαρμόσουμε τον Dijsktra. Το γράφημα αυτό έχει συνολικά $(k+1)*n$ κορυφές και $(k+1)*m$ ακμές, επομένως ο Dijsktra χρειάζεται χρόνο $O((k+1)m+(k+1)nlog((k+1)n))$ = $O(km+nklog(kn))$. Εάν το k πάρει μεγαλύτερη τιμή από n σημαίνει ότι κάπου θα μπορούσαμε να έχουμε κάποιο κύκλο, που προφανώς δεν έχει νόημα αφού ψάχνουμε για ελάχιστα μονοπάτια. Εάν δηλαδή k>n μπορούμε απλά να εφαρμόσουμε τον αλγόριθμο για k = n (αφού έτσι κι άλλιως ζητάμε *το πολύ* k κόκκινες ακμές). Επομένως η μέγιστη τιμή που μπορεί να πάρει το k είναι k=n (όσο και το μήκος του μεγαλύτερου δυνατού μονοπατιού), επομένως θα έχουμε πολυπλοκότητα $O(nm+n^2log(n^2))$ = $O(nm+n^2log(n))$. Στη χειρότερη περίπτωση που θα ισχύει $m=n(n-1)$ η πολυπλοκότητα του αλγορίθμου γίνεται $O(n^3+n^2log(n))=O(n^3)$, δηλαδή ο αλγόριθμος είναι πολυωνυμικού χρόνου.

Άσκηση 4: Επιβεβαίωση και Αναπροσαρμογή Μέγιστης Ροής

(α)

Με δεδομένη τη ροή f κατασκευάζουμε το υπολειμματικό γράφημα σε χρόνο $O(|V|+|E|)$. Αρκεί τώρα να ελέγξουμε εάν στο υπολειμματικό γράφημα υπάρχει επαυξητικό μονοπάτι. Εάν υπάρχει τότε η ροή f δεν είναι μέγιστη, αλλιώς είναι. Ο έλεγχος για επαυξητικό μονοπάτι μπορεί να γίνει με BFS ή DFS σε χρόνο $O(|V|+|E|)$, επομένως ο αλγόριθμός μας είναι γραμμικού χρόνου.

(β)

Εάν η νέα χωρητικότητα c'_e είναι μεγαλύτερη ή ίση από τη μέγιστη ροή, δηλαδή εάν $c'_e \geq f$ τότε δεν χρειάζεται να γίνει κάποια τροποποίηση και η μέγιστη ροή παραμένει ίδια. Στην περίπτωση όμως που $c'_e < f$ θα χρειαστεί αναπροσαρμογή της μέγιστης ροής f σε μια νέα μέγιστη ροή f'. Αυτό γίνεται ως εξής:

1. Θα βρούμε αρχικά μία εφικτή ροή στο G'. Έστω e=(x,y) η ακμή της οποίας μειώσαμε τη χωρητικότητα. Θέτουμε $f' = c'_e$. Για να βρούμε τώρα μια εφικτή ροή στο G' πρέπει να μειώσουμε κατά $c'_e - c_e = k$ τη ροή κατά μήκος των διαδρομών x-s και y-t.

2. Για το σκοπό αυτό θα τρέξουμε δύο BFS: ένα από το x στο s και ένα από το y στο t. Κάθε φορά που ο BFS ανακαλύπτει μια κορυφή v_i στην οποία φτάνει ροή f_i θέλουμε να μειώνουμε τη (συνολική) ροή στις ακμές που περιέχουν τη v_i κατά k. Ξεκινάμε μειώνοντας τη ροή της ακμής με τη μέγιστη ροή μέχρι αυτή είτε να μηδενιστεί οπότε και ξεκινάμε να μειώνουμε τη ροή της επόμενης ακμής, είτε μέχρι να μειώσουμε συνολικά k μονάδες ροής.

3. Αφου τελειώσουν οι BFS θα έχουμε δημιουργήσει μια εφικτή ροή. Θέλουμε τώρα να τη μετατρέψουμε σε μέγιστη. Για το σκοπό αυτό θα χρησιμοποιήσουμε τον αλγόριθμο Ford-Fulkerson.

Ο αλγόριθμος Ford-Fulkerson θα τερματίσει μετά από το πολύ k επαναλήψεις (αφού η ροή αυξάνεται κατά τουλάχιστον ένα σε κάθε επανάληψη και δεν μπορεί να ξεπεράσει την αρχική τιμή της πριν τη μείωση της χωρητικότητας της e), επομένως ο παραπάνω αλγόριθμος έχει πολυπλοκότητα $O(k(|V|+|E|))$.

Άσκηση 5: Αναγωγές και NP-Πληρότητα

Υποσύνολα Διαφορετικών Συνόλων

Εάν μας δοθούν δύο υποσύνολα μπορούμε να ελέγξουμε σε πολυωνυμικό χρόνο εάν η διαφορά των αθροισμάτων τους ισούται με Β. Επομένως το πρόβλημα ανήκει στο NP.

Παρατηρούμε ότι εάν τα στοιχεία του Υ είναι όλα μηδενικά το πρόβλημά μας ανάγεται στο Subset Sum, το οποίο είναι NP-hard. Επομένως θα είναι NP-hard και το δικό μας πρόβλημα ως γενίκευση του Subset Sum. Άρα το πρόβλημά μας είναι NP-complete.

Άθροισμα Υποσυνόλου κατά Προσέγγιση

Δεδομένου ενός υποσυνόλου S' μπορούμε να διαπιστώσουμε σε πολυωνυμικό χρόνο εάν το άθροισμα των στοιχείων του βρίσκεται μεταξύ των δύο επιθυμητών τιμών. Επομένως το πρόβλημα ανήκει στο NP.

Για να αποδείξουμε ότι το πρόβλημα είναι NP-hard θα κάνουμε αναγωγή από το Subset Sum. Έστω σύνολο $S = \{s_1, ..., s_n\}$. Τότε μπορούμε να κατασκεύασουμε το σύνολο $S' = \{2s_1, ...2s_n\}$ το οποίο περιέχει το διπλάσιο κάθε στοιχείου του S. Προφανώς το να περιέχει το S ένα υποσύνολο με άθροισμα W είναι ισοδύναμο με το να περιέχει το S' ένα υποσύνολο με άθροισμα 2W. Εάν τώρα εφαρμόσουμε το πρόβλημά μας στο σύνολο S' με B=2W και x=1 ψάχνουμε εάν υπάρχει $A \subseteq S'$ τέτοιο ώστε $2W - 1 \leq w(A) \leq 2W$. Όμως τα στοιχεία του S' είναι όλα άρτια, επομένως υπάρχει τέτοιο υποσύνολο αν και μόνο αν ισχύει $w(A) = 2W$, δηλαδή αν και μόνο αν το σύνολο αυτό είναι και λύση του Subset sum για το σύνολο S', ή αντίστοιχα εάν υπάρχει λύση για το Subset sum για το αρχικό σύνολο S. Άρα το πρόβλημα είναι NP-hard, επομένως είναι και NP-complete.

Συνδετικό Δέντρο Περιορισμένο ως προς το Μέγιστο Βαθμό

Εάν μας δοθεί ένα συνδετικό δέντρο μπορούμε εύκολα να διαπιστώσουμε σε πολυωνυμικό χρόνο εάν η κάθε κορυφή του έχει βαθμό μικρότερο ή ίσο του k. Επομένως το πρόβλημα ανήκει στο NP.

Παρατηρούμε ότι για k=2 το πρόβλημά μας είναι ουσιαστικά το πρόβλημα του Hamilton path. Επομένως το πρόβλημα είναι NP-hard, αφού αποτελεί γενίκευση του Hamilton path το οποίο όπως αποδεικνύεται στις διαφάνειες με τις λυμένες ασκήσεις είναι NP-hard. Άρα το πρόβλημα είναι NP-complete.

Διαχωρισμός σε Κλάσεις (Clustering)

Δεδομένης μιας διαμέρισης του P σε k κλάσεις μπορούμε να ελέγξουμε σε πολυωνυμικό χρόνο εάν όλα τα σημεία της κάθε κλάσης απέχουν το πολύ Β συγκρίνοντας όλες τις αποστάσεις με το Β. Επομένως το πρόβλημα ανήκει στο NP.

Θα αποδείξουμε ότι το πρόβλημα είναι NP-hard με αναγωγή από το πρόβλημα του k-χρωματισμού. Έστω ένα γράφημα G(V,E). Κατασκευάζουμε ένα σύνολο $P = \{p_1, ..., p_n\}$, $n = |V|$ και ορίζουμε τις αποστάσεις $d(p_i, p_j) = d(p_j, p_i)$ να είναι ίσες με B+1 εάν υπάρχει ακμή (v_i, v_j) στο γράφημα, αλλιώς να είναι Β. Τότε το G διαθέτει έναν k-χρωματισμό αν και μόνο αν υπάρει μια διαμέριση του P σε k-κλάσεις τα στοιχεία των οποίων θα απέχουν το πολύ Β. Πράγματι, εάν βρούμε μια διαμέριση του P σε k κλάσεις θα πάρουμε k υποσύνολα του V τα οποία είναι ανεξάρτητα μεταξύ τους, αφού εάν δύο κορυφές συνδέονται με ακμή, τα σημεία που αντιστοιχούν σε αυτές στο P θα βρίσκονται σε διαφορετικές κλάσεις. Αντίστροφα, εάν βρούμε έναν k-χρωματισμό του G σημαίνει ότι έχουμε βρει k ανεξάρτητα υποσύνολα τα στοιχεία του καθενός απέχουν μεταξύ τους απόσταση το πολύ Β.

Το πρόβλημα ανήκει στο NP και είναι NP-hard, άρα είναι NP-complete.

Συντομότερο Μονοπάτι με Περιορισμούς (Constraint Shortest Path)

Εάν μας δοθεί ένα υποψήφιο μονοπάτι μπορούμε να ελέγξουμε σε πολυωνυμικό χρόνο εάν αυτό ικανοποιεί τους περιορισμούς, αθροίζοντας τα μήκη και τα κόστη και συγκρίνοντάς τα με τους περιορισμούς. Επομένως το πρόβλημα ανήκει στο NP.

Για να αποδείξουμε ότι είναι και NP-hard θα κάνουμε αναγωγή από το Knapsack problem. Έστω B το μέγεθος του σακιδίου, P η ελάχιστη συνολική αξία που θέλουμε να έχουν τα αντικείμενα που θα επιλέξουμε και n αντικείμενα που το καθένα έχει μέγεθος s_i και αξία p_i. Έστω P_{sum} το άθροισμα όλων των p_i. Δημιουργούμε ένα κατευθυνόμενο γράφημα με n+1 κορυφές όπου καθεμία από τις κορυφές $v_1, ..., v_n$ αντιστοιχεί σε ένα από τα n αντικείμενα και v_0 μια επιπλέον αρχική κορυφή. Κάθε κορυφή v_{i-1} συνδέεται με την κορυφή v_i με δύο ακμές (κατευθυνόμενες από την v_{i-1} στη v_i). Η μία ακμή έχει μήκος ίοο με s_i και κόστος $P_{sum} - p_i$ και αντιστοιχεί στην επιλογή του αντικειμένου i. Η άλλη ακμή έχει μήκος ίσο με 0 και κόστος P_{sum} και αντιστοιχεί στη μη επιλογή του i.

Εάν θέσουμε $s = x_0$, $t = x_n$, $W = B$ και $C = nP_{sum} - P$ παρατηρούμε ότι η εύρεση συντομότερου μονοπατιού s-t με τους δεδομένους περιορισμούς είναι ισοδύναμη με την επιλογή των κατάλληλων αντικειμένων για το σακίδιο, αφού θέλουμε $\sum w_i \leq W$ δηλαδή $\sum s_i \leq B$ και $\sum c_i \leq C$ δηλαδή $nP_{sum} - \sum p_i \leq nP_{sum} - P \Rightarrow \sum p_i \geq P$.

Αποδείξαμε ότι το πρόβλημα ανήκει στο NP και είναι NP-hard, επομένως είναι NP-complete.