Αλγόριθμοι υπολογισμού ελάχιστου μονοπατιού

Μιλτιάδης Αναγνώστου

25/11/2022

Το πρόβλημα της ελάχιστης διαδρομής

- Δεδομένου ενός προσανατολισμένου γράφου $\{\mathcal{N}, \mathcal{A}\}$ με κόμβους αριθμημένους $1, 2, \ldots, N$ και *μήκος* a_{ij} για κάθε ακμή (i, j), το *μήκος* της ορθής (forward) διαδρομής $(i_1, i_2, \ldots, i_k)$ είναι το μήκος των ακμών της

$$\sum_{n=1}^{k-1} a_{i_n i_{n+1}}.$$

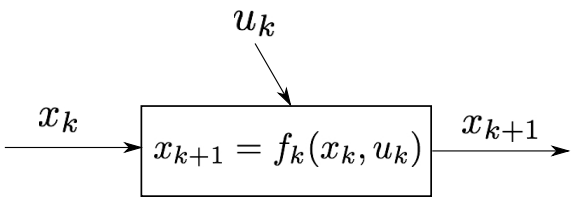
- Η διαδρομή λέγεται *ελάχιστη* αν έχει το μικρότερο μήκος από όλες τις ορθές διαδρομές, που αρχίζουν και τελειώνουν στους ίδιους κόμβους i_1, i_k.
- Στη συνέχεια δίνονται μερικά προβλήματα, που μπορούν να πάρουν τη μορφή προβλήματος ελάχιστης διαδρομής.[1]

[1] Βιβλιογραφία: D. Bertsekas, "Network Optimization", Athena Scientific, Boston, 1998.

Δρομολόγηση σε δίκτυα μεταγωγής πακέτου

- Η χρήση της ελάχιστης διαδρομής αποτελεί δημοφιλή μέθοδο δρομολόγησης σε δίκτυα μεταγωγής πακέτου.
- Το *μήκος* ενός καναλιού μπορεί απλά να τεθεί ίσο με 1, οπότε επιδιώκουμε να στέλνουμε τα πακέτα μέσω του μικρότερου δυνατού αριθμού κόμβων.
- Μπορεί επίσης να τεθεί ίσο με το χρόνο μετάδοσης ενός πακέτου, προκειμένου να ελαχιστοποιηθεί ο χρόνος μετάδοσης από άκρο σε άκρο.
- Μπορεί να χρησιμοποιηθεί και κάποιο μέτρο του αναμενόμενου φορτίου, προκειμένου να αποφευχθεί η συμφόρηση σε ορισμένες διαδρομές.
- Μπορεί επίσης να τεθεί ίσο με το συνολικό χρόνο αναμονής και μετάδοσης του πακέτου μέσα από ένα κανάλι. Έτσι δημιουργείται δυναμικός αλγόριθμος, που έχει ανάγκη να ενημερώνεται για τις καθυστερήσεις, που συμβαίνουν στην έξοδο κάθε καναλιού.

Δυναμικός προγραμματισμός



- Δίνεται δυναμικό σύστημα διακριτού χρόνου με φάσεις.
- Η κατάσταση του συστήματος στην k φάση παριστάνεται με x_k.
- Δίνεται η αρχική κατάσταση x_0. Κατά την k φάση η κατάσταση του συστήματος μεταβάλλεται με χρήση και μιας μεταβλητής ελέγχου u_k ως εξής

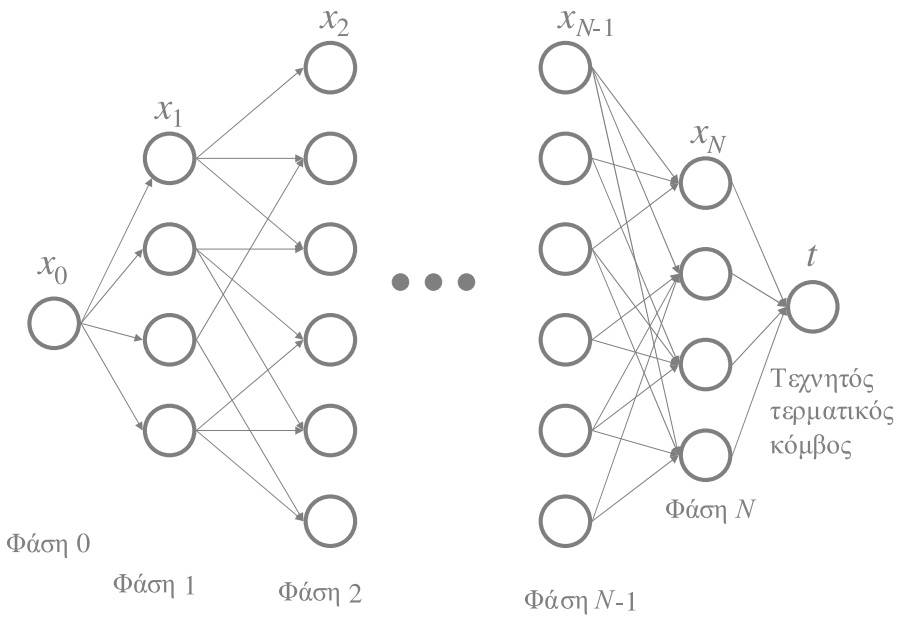
$$x_{k+1} = f_k(x_k, u_k)$$

- Η μετάβαση δημιουργεί κόστος $g_k(x_k, u_k)$ και το συνολικό προς ελαχιστοποίηση κόστος είναι

$$G(x_N) + \sum_{k=0}^{N-1} g(x_k, u_k)$$

όπου $G(x_N)$ είναι επιπρόσθετο κόστος τερματισμού.

Μετατροπή σε πρόβλημα ελάχιστης διαδρομής



- Στο σχήμα φαίνεται με ποιο τρόπο μετατρέπεται το πρόβλημα του δυναμικού προγραμματισμού σε πρόβλημα ελάχιστης διαδρομής μεταξύ του κόμβου x_0 και του τεχνητού κόμβου t.

Παράδειγμα δυναμικού προγραμματισμού:

- Σε μια αποθήκη το εμπόρευμα παραδίδεται κάθε μήνα και είναι σε ποσότητα x_k, που μεταβάλλεται με τον τύπο

$$x_{k+1} = x_k + u_k - v_k$$

όπου η ζήτηση v_k μέσα στο μήνα είναι άγνωστη. Αρνητικό x_k μπορεί να σημαίνει εμπόρευμα που παραγγέλλεται χωρίς να μπορεί να παραδοθεί αμέσως και έχει κάποιο πρόσθετο κόστος.

- Η συνάρτηση κόστους μπορεί να είναι της μορφής

$$g_k(x_k, u_k) = h(x_k) + c\,u_k$$

όπου c είναι το κόστος αποθήκευσης ανά μονάδα εμπορεύματος και $h(x_k)$ είναι το κόστος αποθήκευσης για υπερβολική ποσότητα εμπορεύματος αν $x_k > 0$ ή το κόστος καθυστερημένης παραγγελίας αν $x_k < 0$.

Διάρκεια έργου, I

- Ας υποτεθεί ότι πρόκειται να γίνει χρονοπρογραμματισμός της εκτέλεσης ενός έργου, το οποίο αποτελείται από φάσεις γνωστής διάρκειας. Επίσης γνωστή είναι μια μερική διάταξη των εργασιών με την έννοια ότι ορισμένες εργασίες προηγούνται άλλων.
- Παράδειγμα ας θεωρηθεί το χτίσιμο ενός σπιτιού. Θέλουμε να προσδιορίσουμε την ελάχιστη διάρκεια εκτέλεσης του έργου, καθώς και ποιες εργασίες είναι κρίσιμες για τον προσδιορισμό της.
- Το πρόβλημα μπορεί να λυθεί κατασκευάζοντας ένα γράφο, του οποίου οι κόμβοι παριστάνουν το πέρας φάσεων.
- Η ακμή (i, j) παριστάνει μια εργασία, που αρχίζει μετά το πέρας της φάσης i και έχει γνωστή διάρκεια t_{ij}. Μια φάση ολοκληρώνεται όταν όλες οι εργασίες, που παριστάνονται από εισερχόμενες ακμές, έχουν ολοκληρωθεί. Δυο ειδικοί κόμβοι, ας είναι οι 1 και N, παριστάνουν την αρχή και το τέλος του έργου.

Διάρκεια έργου, II

- Για τον προσδιορισμό του χρόνου ολοκλήρωσης της φάσης i απαιτείται ο προσδιορισμός της μέγιστης διαδρομής μεταξύ 1 και i.
- Το πρόβλημα μπορεί να μετατραπεί σε πρόβλημα ελάχιστης διαδρομής εφόσον τεθεί διάρκεια $-t_{ij}$ στην ακμή (i, j).

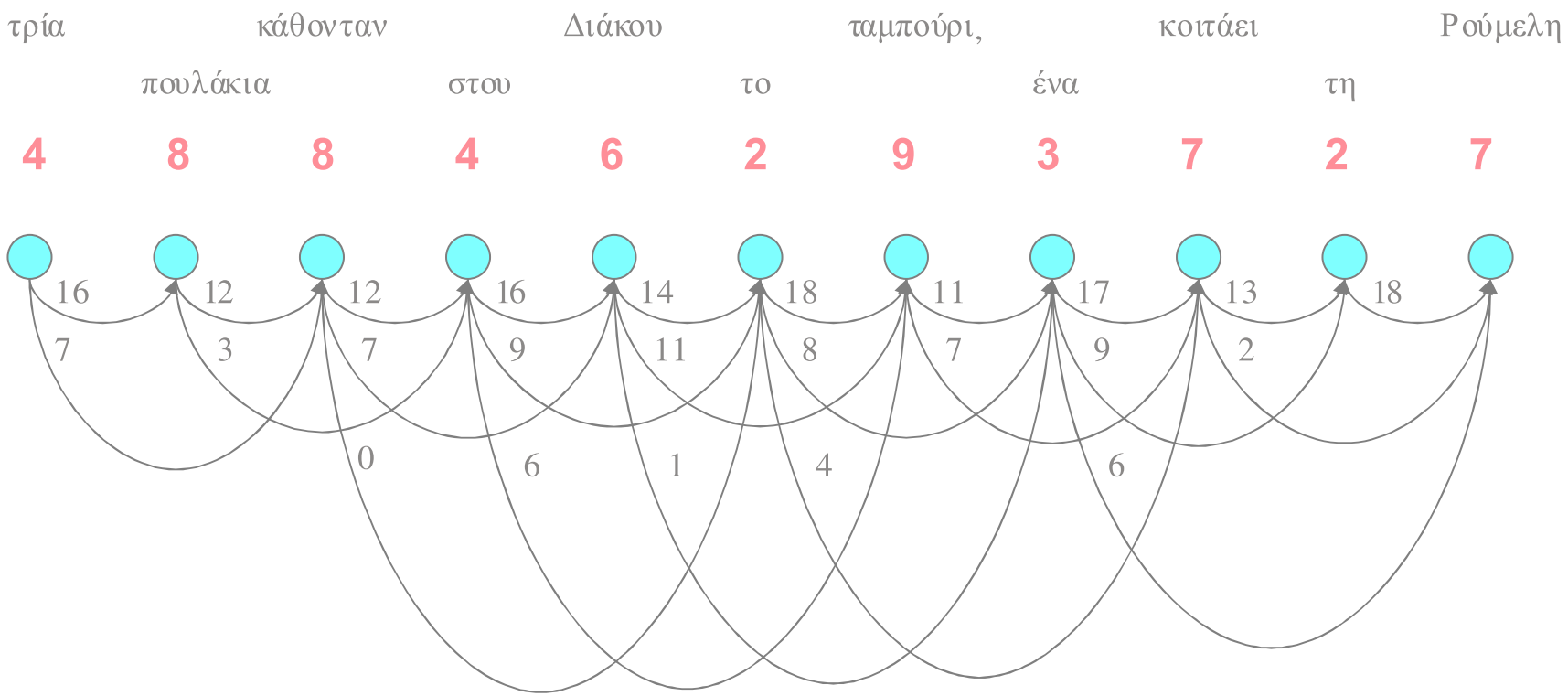
Το πρόβλημα του διαχωρισμού των γραμμών μιας παραγράφου

- Δίνεται μια παράγραφος, που αποτελείται από N λέξεις. Πρόκειται να χωριστούν σε ομάδες, που καθεμιά τους παριστάνει μια γραμμή. Υποτίθεται γνωστός ένας κανόνας, που σε κάθε ακολουθία λέξεων αποδίδει ένα συγκεκριμένο κόστος, εφόσον αυτές βρεθούν στην ίδια γραμμή. Ξεκινώντας από αυτόν τον κανόνα, μπορεί να κατασκευασθεί το κόστος c_{ij}, που παριστάνει το κόστος μιας γραμμής, που ξεκινάει από τη λέξη i και καταλήγει στη λέξη j.
- Το πρόβλημα κατόπιν λύνεται κατασκευάζοντας ένα γράφο με κόμβους τις λέξεις και κατασκευάζοντας ακμές (i, j) με αντίστοιχα κόστη c_{ij}. Οι ακμές κατά μήκος της ελάχιστης διαδρομής παριστάνουν τις γραμμές της κατά βέλτιστο τρόπο χωρισμένης παραγράφου.

Εφαρμογή της προηγ. μεθόδου

- Δίνεται το κείμενο *Τρία πουλάκια κάθονταν στου Διάκου το ταμπούρι, ένα κοιτάει τη Ρούμελη*.
  Το κείμενο αυτό πρέπει να χωρέσει σε σειρές των 20 χαρακτήρων, όπου ο κάθε χαρακτήρας θεωρείται ίδιου πλάτους.
- Θεωρούμε πως η συνάρτηση κόστους θα παριστάνει τον αριθμό χαρακτήρων, που θα μείνουν συνολικά κενοί σε όσες γραμμές χρησιμοποιηθούν.

Εφαρμογή της προηγ. μεθόδου, I



Εφαρμογή της προηγ. μεθόδου, II

- Αρχικά σχηματίζεται ο γράφος, όπου η ακμή (i, j) επιγράφεται με το c_{ij}, ήτοι τον αριθμό των κενών σε σειρά, που αρχίζει με τη λέξη i και τελειώνει με τη λέξη $j-1$. Δηλαδή υπολογίζεται το άθροισμα $s_{i,j-1}$ των γραμμάτων των λέξεων i ως και $j-1$ μαζί με τα μεταξύ τους κενά (καθώς και το ενδεχόμενο σημείο στίξης που ακολουθάει μια λέξη) και τίθεται $c_{ij} = 20 - s_{i,j-1}$. Οι αριθμοί πάνω από τους κόμβους είναι τα αντίστοιχα μήκη λέξεων (χωρίς τα κενά).
- Κατόπιν υπολογίζεται η ελάχιστη διαδρομή (π.χ. με Dijkstra), η οποία εδώ συμβαίνει να είναι αυτή που περνάει από τους κόμβους 1, 3, 6, 9, 11 με συνολικό μήκος $7 + 0 + 4 + 2 = 13$.
- Κατά συνέπεια το κείμενο διατάσσεται ως εξής: Τρία πουλάκια / κάθονταν στου Διάκου / το ταμπούρι, ένα / κοιτάει τη Ρούμελη

Αλγόριθμοι ελάχιστης διαδρομής

- Μια κατηγορία αλγορίθμων χρησιμοποιεί ένα διάνυσμα $(d_1, d_2, \ldots, d_N)$, του οποίου η συνιστώσα d_i αντιστοιχεί στον κόμβο i. Η χρήση αυτών των μεγεθών στηρίζεται στην επόμενη πρόταση.
- *Πρόταση:* Έστω ότι τα (μονόμετρα) μεγέθη $d_1, d_2, \ldots, d_N$ ικανοποιούν $\forall (i,j) \in \mathcal{A}$ την ανισότητα

$$d_j \leq d_i + a_{ij} \tag{1}$$

και P είναι μια διαδρομή που αρχίζει από τον κόμβο i_1 και τελειώνει στον κόμβο i_k. Αν για όλες τις ακμές (i,j) του P ισχύει

$$d_j = d_i + a_{ij}, \tag{2}$$

η διαδρομή P είναι ελάχιστη μεταξύ i_1 και i_k.
- Η συνθήκη 1 λέγεται *συνθήκη συμπληρωματικής χαλάρωσης για το πρόβλημα ελάχιστης διαδρομής*.

Διόρθωση των επιγραφών

- Μια δημοφιλής κατηγορία αλγορίθμων βασίζεται στην εκτέλεση ενός βήματος διόρθωσης επιγραφών ως εξής:
- Όπου παραβιάζεται η

$$d_j \leq d_i + a_{ij}$$

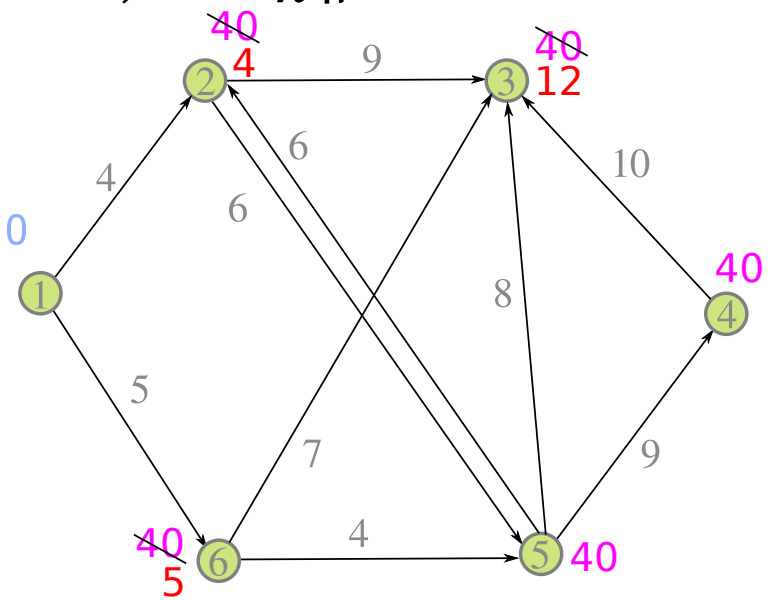
τίθεται

$$d_j = d_i + a_{ij}$$

- Η επανάληψη αυτού του βήματος ενώ διατηρεί την ιδιότητα $d_j \leq d_i + a_{ij}$ μεταξύ γειτονικών κόμβων, δημιουργεί τις προϋποθέσεις για την ισότητα σε μια διαδρομή.
- Η εκτέλεση ενός τέτοιου αλγορίθμου βασίζεται και στις κατάλληλες αρχικές τιμές για να επιτευχθεί διαδρομή με την ιδιότητα $d_j = d_i + a_{ij}$.

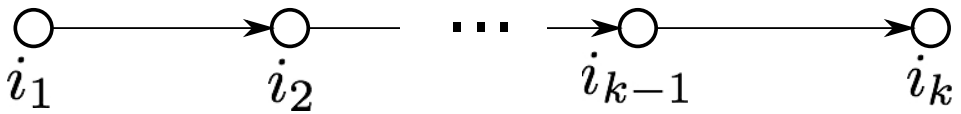
Διόρθωση των επιγραφών - παράδειγμα

Κάνοντας αυτό το βήμα στις ακμές $(1,2)$, $(1,6)$ και $(6,3)$ αλλάζουν οι επιγραφές όπως στο σχήμα:



- Υπάρχουν άλλες δυνατές διορθώσεις;
- Υπάρχει διαδρομή τέτοια, ώστε να ισχύει η συνθήκη $d_j = d_i + a_{ij}$;

Απόδειξη της προηγούμενης πρότασης:



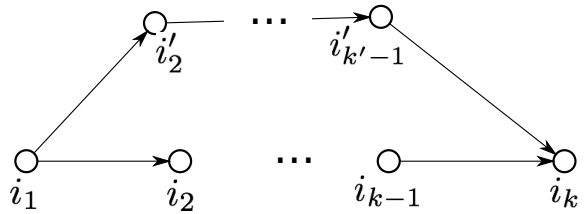
Έστω ότι η διαδρομή P περιλαμβάνει τους κόμβους $i_1, i_2, \ldots, i_k$.
Προσθέτοντας κατά μέλη τις

$$\begin{aligned} d_{i_2} &= d_{i_1} + a_{i_1,i_2} \\ d_{i_3} &= di_2 + a_{i_2,i_3} \\ &\ldots \\ d_{i_k} &= d_{i_{k-1}} + a_{i_{k-1},i_k} \end{aligned}$$

προκύπτει ότι

$$d_{i_k} - d_{i_1} = \sum_{n=1}^{k-1} a_{i_n,i_{n+1}}.$$

'ρα για το μήκος $d(P) \equiv \sum_{n=1}^{k-1} a_{i_n,i_{n+1}}$ της διαδρομής P ισχύει $d(P) = d_{i_k} - d_{i_1}$.



Έστω άλλη διαδρομή $P' = (i_1, i'_2, \ldots, i'_{k'-1}, i_k)$.
Προσθέτοντας κατά μέλη τις

$$\begin{aligned} d_{i'_2} &\leq d_{i_1} + a_{i_1,i'_2} \\ d_{i'_3} &\leq di'_2 + a_{i'_2,i'_3} \\ &\ldots \\ d_{i_k} &\leq d_{i'_{k'-1}} + a_{i'_{k'-1},i_k} \end{aligned}$$

προκύπτει ότι

$$d_{i_k} - d_{i_1} \leq a_{i_1,i'_2} + \sum_{n=2}^{k'-2} a_{i'_n,i'_{n+1}} + a_{i'_{k'-1},i_k}$$

Άρα για το μήκος της διαδρομής P' ισχύει $d(P) = d_{i_k} - d_{i_1} \leq d(P')$.

Ένας γενικός αλγόριθμος ελάχιστης διαδρομής

- Για τους κόμβους $1, 2, \ldots, N$ του γράφου συντηρούμε σε κάθε βήμα τις παραμέτρους $d_1, d_2, \ldots, d_N$.
- Σε κάθε βήμα διαλέγουμε μια ακμή (i, j), που δεν ικανοποιεί τη συνθήκη (1), δηλαδή διαλέγουμε μια ακμή για την οποία ισχύει $d_j > d_i + a_{ij}$. Κατόπιν θέτουμε
$$d_j := d_i + a_{ij}$$
- Η βασική ιδέα είναι ότι η παράμετρος d_i μπορεί να θεωρηθεί ότι παριστάνει την απόσταση από κάποιο κόμβο αναφοράς, έστω τον κόμβο 1 εφόσον θέσουμε $d_1 = 0$.

Παρατηρήσεις:

Η παραπάνω διαδικασία δεν είναι δυνατή αρχίζοντας από οποιεσδήποτε αρχικές συνθήκες. Π.χ. αν για όλους τους κόμβους i ισχύει $d_i = 0$, οι συνθήκες $d_j \leq d_i + a_{ij}$ δεν παραβιάζονται, αλλά παραβιάζεται η συνθήκη ισότητας (2), οπότε δεν ισχύουν οι προϋποθέσεις του θεωρήματος και δεν προσδιορίζεται μέσω αυτού η ελάχιστη διαδρομή. Συνάμα δεν είναι δυνατή η εκτέλεση του αλγορίθμου.

Η σειρά επιλογής των ακμών δεν έχει σημασία για την επιτυχία του αλγορίθμου. Εναλλακτικά μπορεί κανείς να επιλέγει ένα κόμβο κάθε φορά και να επεξεργάζεται όλες τις εξερχόμενες ακμές του. Αυτό μπορεί να γίνει διατηρώντας ένα σύνολο V, που περιλαμβάνει τους κόμβους προς επεξεργασία στο επόμενο βήμα. Αυτή η μέθοδος εκτίθεται στην επόμενη παράγραφο.

Γενικός αλγόριθμος

Αρχική συνθήκη:

$$V = \{1\}, d_1 = 0, \forall i \neq 1 : d_i = \infty$$

Βήμα:
Να αφαιρεθεί ένας κόμβος i από το σύνολο V. Για κάθε εξερχόμενη ακμή $(i,j) \in \mathcal{A}$ αν $d_j > d_i + a_{ij}$, να τεθεί

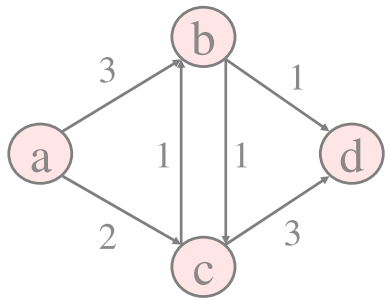
$$d_j := d_i + a_{ij}$$

και να προστεθεί το j sto σύνολο V αν δεν ανήκει ήδη σ' αυτό.
Παρατηρήσεις:

- Οι τιμές των d_i είναι μονότονα μη αύξουσες.
- $d_i < \infty$ αν και μόνον αν το i έχει περιληφθεί στο V τουλάχιστον μια φορά.

Παράδειγμα

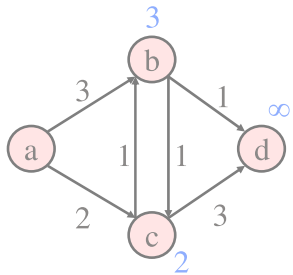
:



Αρχικά ισχύει

$$V = \{a\}, d_a = 0, d_b = d_c = d_d = \infty$$

Βήμα 1: Αρχίζουμε από τον κόμβο a που είναι και ο μοναδικός που για την ώρα ανήκει στο V και διορθώνουμε τις αποστάσεις προς τους κόμβους b, c, επειδή προς αυτούς εξέρχονται ακμές από τον a. Επομένως το νέο V είναι $V = \{b, c\}$. Η συνθήκη $d_j > d_i + a_{ij}$ ισχύει για αμφότερους τους b, c, οπότε $d_b := d_a + 3 = 3$, $d_c := d_a + 2 = 2$, ενώ παραμένει $d_d = \infty$.



Βήμα 2: Αρχίζουμε από τον κόμβο c με αυθαίρετη επιλογή ανάμεσα στα στοιχεία του $V = \{b, c\}$ και διορθώνουμε τις αποστάσεις προς τους κόμβους b, d, επειδή προς αυτούς εξέρχονται ακμές από τον c. Προς το παρόν ισχύει $(d_a, d_b, d_c, d_d) = (0, 3, 2, \infty)$. Η συνθήκη $d_j > d_i + a_{ij}$ ισχύει μόνο για τον d και $d_d := d_c + 3 = 5$. 'ρα $(d_a, d_b, d_c, d_d) = (0, 3, 2, 5)$ Το νέο V είναι $V = \{b, d\}$.
Βήμα 3: Εφόσον $V = \{b, d\}$ και από τον d δεν υπάρχει εξερχόμενη ακμή, ας επιλεγεί ο b με εξερχόμενες ακμές προς c, d. Η συνθήκη $d_j > d_i + a_{ij}$ ισχύει μόνο για τον d και γίνεται $d_d := d_b + 1 = 3 + 1 = 4$. 'ρα $(d_a, d_b, d_c, d_d) = (0, 3, 2, 4)$ Το νέο V είναι $V = \{d\}$.
Βήμα 4: Ο κόμβος d που έχει απομείνει στο V δεν έχει εξερχόμενες ακμές. Επομένως γίνεται το τετριμμένο βήμα εξαγωγής του d από το V, το οποίο πλέον μένει κενό και φέρνει τον τερματισμό του αλγορίθμου.

Πεπερασμένα d_j είναι μήκη διαδρομών

Πρόταση 1

Στο τέλος ενός βήματος αν $d_j < \infty$, το d_j είναι ίσο με το μήκος μιας διαδρομής, που έχει για αρχή τον κόμβο 1 και τελειώνει στον j.

Επαγωγική απόδειξη:

1ο βήμα Οι κόμβοι που περιλαμβάνονται στο V στο τέλος του πρώτου βήματος είναι ο 1 και όσοι κόμβοι i συνδέονται απ' ευθείας με τον 1 με ακμές $(1, i)$. Για τον 1 ισχύει $d_1 = 0$ και είναι το μήκος της τετριμμένης διαδρομής από τον 1 στον 1. Για τον ως άνω i ισχύει $d_i = a_{1i} + d_1 = a_{1i}$, που είναι το μήκος της διαδρομής 1, i.

n-οστό βήμα Έστω i ο αφαιρούμενος από το V κόμβος στο τρέχον βήμα, ενώ προστίθεται ο j. Από την υπόθεση της επαγωγής για τα προηγούμενα βήματα ισχύει πως ο κόμβος i είναι τελευταίος σε μια διαδρομή P_i από τον 1 στον i με μήκος d_i. Η αλλαγή του d_j sto βήμα κάνει τον j τελευταίο σε μια διαδρομή που αποτελείται από το P_i ακολουθούμενο από την ακμή (i, j) και μήκος $a_{ij} + d_i$.

Οι τιμές d_j κατά την εκτέλεση του αλγόριθμου

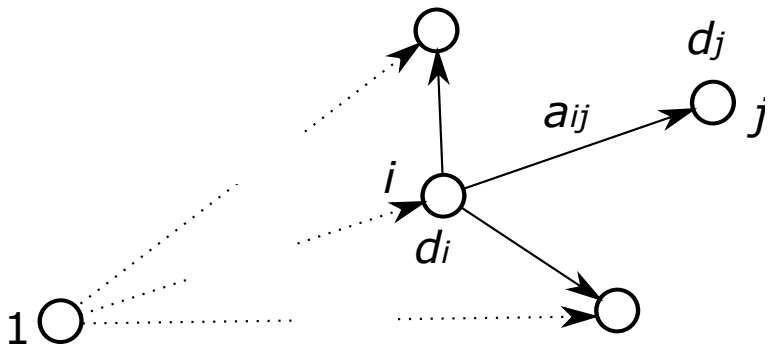
Πρόταση 2

Αν $i \notin V$, ή ισχύει $d_i = \infty$ ή $d_j \leq d_i + a_{ij}$ για κάθε j στο οποίο εισέρχεται ακμή (i, j) προερχόμενη από το i.

Απόδειξη:

- Αν $d_i = \infty$ η πρόταση ισχύει.
- Αν $d_i < \infty$, αυτό σημαίνει ότι ο κόμβος i έχει τουλάχιστον μια φορά συμπεριληφθεί στο σύνολο V κι έχει ξαναβγεί.
- Τη στιγμή της εξόδου του όλοι οι γειτονικοί του κόμβοι j (δηλ. οι συνδεόμενοι με εξερχόμενες ακμές) αποκτούν $d_j = d_i + a_{ij}$.
- Στη συνέχεια και μέχρις ο κόμβος i να ξαναμπεί στο V το d_i δεν μεταβάλλεται, ενώ οι γειτονικοί κόμβοι j έχουν d_j σταθερό είτε μειούμενο, οπότε ισχύει $d_j \leq d_i + a_{ij}$.

Οι τιμές d_j κατά τον τερματισμό



Πρόταση 3

Αν ο αλγόριθμος τερματίσει και για τον κόμβο $j \neq 1$ ισχύει $d_j < \infty$, η ελάχιστη απόσταση μεταξύ 1 και j είναι ίση με d_j και

$$d_j = \min_{(i,j)\in\mathcal{A}} \{d_i + a_{ij}\} \tag{3}$$

Αλλιώς, αν δηλαδή ο αλγόριθμος τερματίσει και για τον κόμβο j ισχύει $d_j = \infty$, δεν υπάρχει διαδρομή μεταξύ 1 και j και αντιστρόφως.

Απόδειξη της Πρότασης 3

Απόδειξη του β' μέρους, ότι δηλαδή «αν ο αλγόριθμος τερματίσει και για τον κόμβο j ισχύει $d_j = \infty$, δεν υπάρχει διαδρομή μεταξύ 1 και j και αντιστρόφως».

- Έστω I το σύνολο των κόμβων i με $d_i < \infty$ κατά τον τερματισμό. Για κάθε κόμβο j προς τον οποίο εξέρχεται ακμή από κόμβο $i \in I$ ισχύει $d_j \leq d_i + a_{ij}$ από την πρόταση 2, άρα $d_j < \infty$.
- Επομένως αν υπάρχει διαδρομή από τον 1 στον j, ισχύει επαγωγικά ότι $d_j < \infty$. Άρα αν $d_j = \infty$ δεν υπάρχει διαδρομή από τον 1 στον j (και γενικότερα δεν υπάρχει διαδρομή από κόμβο του I σε κόμβο του $N - I$).
- Αντίστροφα, αν δεν υπάρχει διαδρομή από τον 1 στον j, λόγω της πρότασης 1 δεν μπορεί να ισχύει $d_j < \infty$ κατά τον τερματισμό, οπότε $j \notin I$.

Απόδειξη της Πρότασης 3

Απόδειξη του α' μέρους, ότι δηλαδή «αν ο αλγόριθμος τερματίσει και για τον κόμβο $j \neq 1$ ισχύει $d_j < \infty$, η ελάχιστη απόσταση μεταξύ 1 και j είναι ίση με d_j και $d_j = \min_{(i,j)\in\mathcal{A}}\{d_i + a_{ij}\}$».

- Από τις προτάσεις 1 και 2 προκύπτει ότι κατά τον τερματισμό για κάθε $i \in I$ και τους κόμβους j προς τους οποίους εξέρχεται ακμή από τον i, ισχύει $d_j \leq d_i + a_{ij}$, όπου d_i είναι το μήκος μιας διαδρομής P_i από τον 1 στον i.
- Με επαναληπτική χρήση της $d_j \leq d_i + a_{ij}$ κατά μήκος μιας διαδρομής P από τον κόμβο 1 μέχρι κάποιο κόμβο m προκύπτει ότι το μήκος του P δεν μπορεί να είναι μικρότερο από $d_m - d_1 = d_m$ (αφού $d_1 = 0$ και κατόπιν δεν αυξάνεται). Βέβαια d_m είναι το μήκος μιας διαδρομής P_m από τον 1 στον m και επομένως αυτή αποτελεί ελάχιστη διαδρομή.
- Κατά μήκος της ελάχιστης διαδρομής ισχύει πλέον η ισότητα $d_j = d_i + a_{ij}$, γεγονός που υποδηλώνει ότι ισχύει $d_j = \min_{(i,j)\in\mathcal{A}}\{d_i + a_{ij}\}$ για όλα τα $j \in I$.

Ακόμη δυο προτάσεις

Πρόταση 5

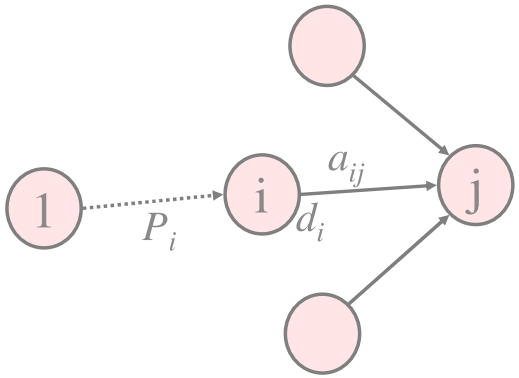
Αν ο αλγόριθμος δεν τερματίζει, υπάρχει κόμβος j και ακολουθία διαδρομών μεταξύ 1 και j, των οποίων το μήκος αποκλίνει προς το $-\infty$.

Πρόταση 6

Ο αλγόριθμος τερματίζει αν και μόνον αν δεν υπάρχει διαδρομή που να αρχίζει από τον κόμβο 1 και να περιέχει ένα κύκλο αρνητικού μήκους.

Για τις αποδείξεις αυτών των δύο προτάσεων, καθώς και των προτάσεων 1-3, δείτε την ενότητα 2.2, σελ. 61, στο βιβλίο: D. Bertsekas, “Network Optimization”, Athena Scientific, Boston, 1998.

Κατασκευή της ελάχιστης διαδρομής



- Η συνθήκη (3)

$$d_j = \min_{(i,j)\in\mathcal{A}} \{d_i + a_{ij}\}$$

λέγεται *συνθήκη του Bellman* και υποδηλώνει ότι η συντομότερη διαδρομή P_j μεταξύ 1 και j κατασκευάζεται διαλέγοντας για πρόγονο του j τον κόμβο i, που ελαχιστοποιεί το άθροισμα της διαδρομής από 1 μέχρι i και του μήκους της ακμής (i,j). Επίσης υποδηλώνει ότι το τμήμα 1 ως i του P_j δεν είναι παρά το P_i.

- Με την ιδιότητα αυτή μπορεί κανείς να ξεκινήσει από τον κόμβο j προς τα πίσω, δηλ. προς τον κόμβο 1, και να καθορίσει την ελάχιστη διαδρομή μεταξύ 1 και j.

Εναλλακτικές αρχικές συνθήκες

Αντί της αρχικής συνθήκης

$$V = \{1\}, d_1 = 0, \forall i \neq 1 : d_i = \infty$$

μπορούν να χρησιμοποιηθούν διαφορετικές αρχικές συνθήκες, αρκεί να ικανοποιούν τις συνθήκες των πρότασεων 1 και 2. Εφόσον αυτές ισχύουν, μπορεί να δει κανείς ότι και τα υπόλοιπα βήματα στις αποδείξεις είναι εφικτά και οι υπόλοιπες προτάσεις ισχύουν.
Η χρήση διαφορετικής αρχικής συνθήκης μπορεί να χρησιμεύσει στις εξής περιπτώσεις:

- Σε προβλήματα με μικρές διαφορές, οπότε η λύση του ενός μπορεί να αποτελέσει εκκίνηση για τη λύση του άλλου.
- Σε περίπτωση που υπολογίζεται μια αρχική λύση με ευριστική μέθοδο.

Εκδοχές του γενικού αλγορίθμου

Και οι δυο παρακάτω μέθοδοι κάνουν χρήση *επιγραφών*:

- *Μέθοδοι καθορισμού επιγραφών:* Από το V απομακρύνεται κάθε φορά ο κόμβος με τη μικρότερη τιμή επιγραφής. Το πλεονέκτημα είναι ότι κάθε κόμβος μπαίνει ακριβώς μια φορά στο σύνολο V. Ωστόσο σε κάθε βήμα πρέπει να υπολογισθεί ο κόμβος με την ελάχιστη τιμή. Οι μέθοδοι αυτές είναι γνωστές ως μέθοδοι Dijkstra από το 1959 με τη σχετική δημοσίευση του Dijkstra.
- *Μέθοδοι διόρθωσης επιγραφών:* Οι επιγραφές έχουν τιμές που μεταβάλλονται κατά τη διάρκεια της εκτέλεσης του αλγορίθμου. Κάθε κόμβος μπορεί να μπει στο V και να βγει από αυτό περισσότερες από μια φορές.

Μέθοδος Dijkstra

- Η αρχική συνθήκη είναι

$$V = \{1\}, d_1 = 0, \forall i \neq 1 : d_i = \infty$$

- Το βήμα είναι:
  - Να απομακρυνθεί από το V ο κόμβος i με την ιδιότητα

$$d_i = \min_{j \in V} d_j$$

  - Κατόπιν, στους κόμβους j, όπου καταλήγουν εξερχόμενες από τον i ακμές (i, j), αν $d_j > d_i + a_{ij}$ να τεθεί

$$d_j := d_i + a_{ij}$$

    και να προστεθεί ο j στο V εφόσον δεν ανήκει ήδη σ' αυτό.

Η επόμενη πρόταση διευκρινίζει τη μέθοδο Dijkstra. Σ' αυτήν το σύνολο W περιλαμβάνει όσους κόμβους έχουν μπει στο V και ήδη βγεί.

Πρόταση 6

Έστω ότι όλα τα μήκη ακμών είναι μη αρνητικά.
Σε κάθε βήμα για το σύνολο

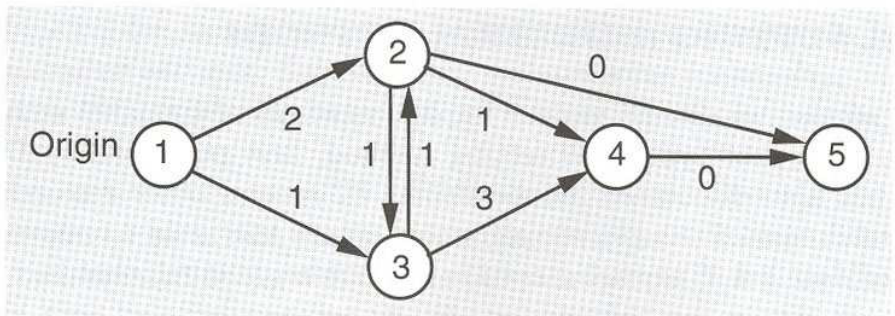
$$W = \{i | d_i < \infty, i \notin V\}$$

ισχύουν τα εξής:

- Κόμβοι που ανήκουν στο W πριν το βήμα δεν μπαίνουν στο V κατά την εκτέλεση του βήματος.
- Στο τέλος του βήματος ισχύει $d_i \leq d_j$ για όλα τα $i \in W$ και τα $j \notin W$.
- Εφόσον για ένα κόμβο j υπάρχει απλή διαδρομή που αρχίζει από τον 1 και τελειώνει στον j και έχει όλους τους άλλους κόμβους μέσα στο W, η επιγραφή d_j στο τέλος του βήματος είναι ίση με το μήκος της ελάχιστης διαδρομής, εκτός αν δεν υπάρχει τέτοια διαδρομή και $d_j = \infty$.

Ο αλγόριθμος τερματίζει και όλοι οι κόμβοι με πεπερασμένη τελική τιμή της επιγραφής απομακρύνονται από το V ακριβώς μια φορά και μάλιστα με σειρά αύξουσας ελάχιστης απόστασης από τον κόμβο 1.

Παράδ. εκτέλεσης του αλγόριθμου του Dijkstra



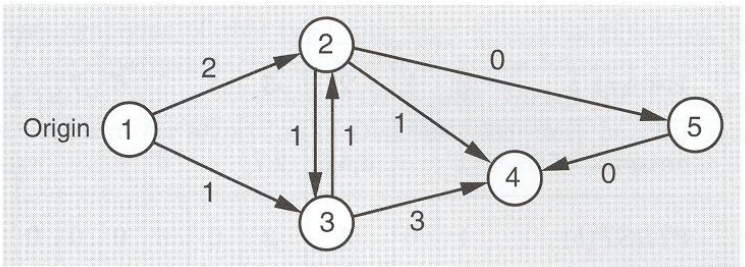
Iteration #	Candidate List V	Node Labels	Node out of V
1	$\{1\}$	$(0, \infty, \infty, \infty, \infty)$	1
2	$\{2, 3\}$	$(0, 2, 1, \infty, \infty)$	3
3	$\{2, 4\}$	$(0, 2, 1, 4, \infty)$	2
4	$\{4, 5\}$	$(0, 2, 1, 3, 2)$	5
5	$\{4\}$	$(0, 2, 1, 3, 2)$	4
	$\emptyset$	$(0, 2, 1, 3, 2)$	

Βλ. D. Bertsekas, “Network Optimization”, Athena Scientific, Boston, 1998.

Ο αλγόριθμος του Dial

- Προϋπόθεση για τον αλγόριθμο του Dial, που δημοσιεύθηκε το 1969, είναι να έχουν οι ακμές μη αρνητικές ακέραιες τιμές.
- Οι επιγραφές (αν δεν έχουν την τιμή ∞) είναι ίσες με το μήκος διαδρομής χωρίς κύκλους και δεν μπορούν να ξεπεράσουν την τιμή $(N-1)C$, όπου $C = \max_{(i,j)\in\mathcal{A}} a_{ij}$. Στην περίπτωση αυτή παίρνουν μια από $1+(N-1)C$ ακέραιες τιμές.
- Στη *θέση*-0 βάζουμε τον κόμβο 1. Στο πρώτο βήμα εξετάζουμε τους κόμβους j, όπου φθάνουν εξερχόμενες ακμές $(1,j)$. Τοποθετούμε στη *θέση*-a_{1j} τον κόμβο j και συγχρόνως τον περιλαμβάνουμε στο σύνολο V.
- Στο δεύτερο βήμα προχωρούμε στη *θέση*-1 και εξετάζουμε όσους κόμβους έχουν πέσει εκεί από το προηγούμενο βήμα (δηλ. η απόστασή τους έχει βρεθεί να είναι ίση με 1). Αυτούς τους αφαιρούμε από το V και υπολογίζουμε εκ νέου τις αποστάσεις των γειτόνων τους. Αυτούς τοποθετούμε στο V, εφόσον δεν ανήκουν ήδη, και επαναπροσδιορίζουμε τις αποστάσεις τους εφόσον ήδη υπήρξαν στο V, επανατοποθετώντας τους ενδεχομένως σε χαμηλότερες θέσεις.

Παράδ. για τον αλγόριθμο του Dial



Iter. #	Cand. List V	Node Labels	Buck. 0	Buck. 1	Buck. 2	Buck. 3	Buck. 4	Out of V
1	$\{1\}$	$(0,\infty,\infty,\infty,\infty)$	1	–	–	–	–	1
2	$\{2,3\}$	$(0,2,1,\infty,\infty)$	1	3	2	–	–	3
3	$\{2,4\}$	$(0,2,1,4,\infty)$	1	3	2	–	4	2
4	$\{4,5\}$	$(0,2,1,3,2)$	1	3	2,5	4	–	5
5	$\{4\}$	$(0,2,1,2,2)$	1	3	2,4,5	–	–	4
	$\emptyset$	$(0,2,1,2,2)$	1	3	2,4,5	–	–	

Βλ. D. Bertsekas, “Network Optimization”, Athena Scientific, Boston, 1998.

Αλγόριθμοι δημοπρασίας - γενικές ιδιότητες

- Ο αλγόριθμος, που δίνεται στη συνέχεια, σε κάθε του βήμα επιφέρει μια μόνο μεταβολή σε μια διαδρομή

$$P = ((s, i_1), (i_1, i_2), \ldots, (i_{k-1}, i_k),$$

την οποία διορθώνει ενόσω εκτελείται.

- Η διόρθωση μπορεί να είναι είτε *διαστολή* (επέκταση) με προσθήκη ενός ακόμη κόμβου, οπότε η διαδρομή P αντικαθίσταται με τη διαδρομή

$$((s, i_1), (i_1, i_2), \ldots, (i_{k-1}, i_k), (i_k, i_{k+1}))$$

είτε *συστολή* με αφαίρεση του τερματικού του (τελευταίου) κόμβου, οπότε η διαδρομή P αντικαθίσταται με τη διαδρομή

$$((s, i_1), (i_1, i_2), \ldots, (i_{k-2}, i_{k-1})).$$

- Ο αλγόριθμος τερματίζει όταν ο προορισμός γίνει τερματικός κόμβος.

Τιμές στον αλγόριθμο δημοπρασίας

- Εισάγεται η μεταβλητή p_i, που λέγεται *τιμή του κόμβου* i. Ο αλγόριθμος χρησιμοποιεί το διάνυσμα των τιμών $p = (p_1, p_2, \ldots, p_N)$.
- Το διάνυσμα αυτό μαζί με τη διαδρομή P ικανοποιούν τις εξής ιδιότητες:

$$\forall (i,j) \in \mathcal{A} : p_i \leq a_{ij} + p_j$$

$$\forall (i,j) \text{ της διαδρομής} P : p_i = a_{ij} + p_j$$

  Αν τεθεί $p_i = -d_i$ οι παραπάνω συνθήκες είναι ίδιες με τις γνωστές συνθήκες συμπληρωματικής χαλάρωσης (CS), οπότε κι αυτές οι συνθήκες είναι γνωστές ως CS.
- Μπορεί να δειχθεί ότι εφόσον ένα ζεύγος (P, p) ικανοποιεί τις CS, το μέρος του P μεταξύ του κόμβου s και οιουδήποτε κόμβου $i \in P$ είναι ελάχιστη διαδρομή, ενώ η αντίστοιχη ελάχιστη απόσταση είναι ίση με $p_s - p_i$.
- Η εκτέλεση του αλγορίθμου μπορεί να αρχίσει από το ζεύγος $(P, p) = ((s), (0, 0, \ldots, 0))$, που ικανοποιεί προφανώς τις CS.

Κύκλος του αλγορίθμου δημοπρασίας

- Έστω ότι προς το παρόν ο τερματικός κόμβος του P είναι ο i. Αν

$$p_i < \min_{\{j|(i,j)\in\mathcal{A}\}} \{a_{ij} + p_j\},$$

πήγαινε στο *Βήμα 1*, αλλιώς πήγαινε στο *Βήμα 2*.

- *Βήμα 1* (Συστολή διαδρομής): Να τεθεί

$$p_i := \min_{\{j|(i,j)\in\mathcal{A}\}} \{a_{ij} + p_j\}$$

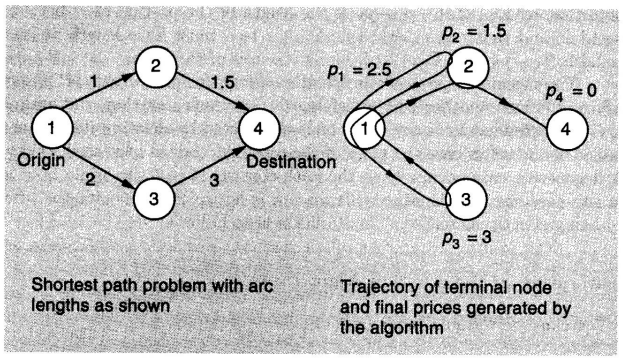
και αν $i \neq s$ να γίνει συστολή του P. Πήγαινε στον επόμενο κύκλο.

- *Βήμα 2* (Διαστολή διαδρομής): Να επεκταθεί η διαδρομή P με τον κόμβο j_i, όπου

$$j_i = \arg \min_{\{j|(i,j)\in\mathcal{A}\}} \{a_{ij} + p_j\}$$

(Αν περισσότεροι του ενός κόμβοι επιτυγχάνουν το ελάχιστο, επιλέγεται κάποιος αυθαίρετα). Αν $j_i = t$, stop και η P είναι η συντομότερη διαδρομή, αλλιώς πήγαινε στον επόμενο κύκλο.

Παράδ. για τον αλγόριθμο δημοπρασίας



Iteration #	Path P prior to iteration	Price vector p prior to iteration	Type of action during iteration
1	(1)	(0, 0, 0, 0)	contraction at 1
2	(1)	(1, 0, 0, 0)	extension to 2
3	(1, 2)	(1, 0, 0, 0)	contraction at 2
4	(1)	(1, 1.5, 0, 0)	contraction at 1
5	(1)	(2, 1.5, 0, 0)	extension to 3
6	(1, 3)	(2, 1.5, 0, 0)	contraction at 3
7	(1)	(2, 1.5, 3, 0)	contraction at 1
8	(1)	(2.5, 1.5, 3, 0)	extension to 2
9	(1, 2)	(2.5, 1.5, 3, 0)	extension to 4
10	(1, 2, 4)	(2.5, 1.5, 3, 0)	stop

Βλ. D. Bertsekas, “Network Optimization”, Athena Scientific, Boston, 1998.