Approximation Algorithms for NP-Hard Problems

Approximation Algorithms (1/16)

Η προσέγγιση κάποιων προβλημάτων (π.χ. Traveling salesman) ως προβλήματα απόφασης τα κατατάσσει ως NP-complete προβλήματα ενώ η προσέγγισή τους ως προβλήματα βελτιστοποίησης τα κατατάσσει ως προβλήματα NP-hard

Τα NP-hard προβλήματα είναι δύσκολα τουλάχιστον όπως τα NP-complete προβλήματα

Έτσι, δεν υπάρχει γνωστός αλγόριθμος πολυωνυμικού χρόνου για την επίλυσή τους

Επίσης, μπορεί να υπάρχουν θεωρητικές ενδείξεις ότι τέτοιοι αλγόριθμοι δεν υπάρχουν

Approximation Algorithms (2/16)

Αν ένα στιγμιότυπο αυτών των προβλημάτων είναι αρκετά μικρό μπορούμε να εφαρμόσουμε exhaustive search

Κάποια μπορεί να λυθούν με δυναμικό προγραμματισμό

Η καλή επίδοση της branch-and-bound δεν είναι εγγυημένη

Μια λύση είναι η εφαρμογή προσεγγιστικών αλγορίθμων που είναι αρκετά γρήγοροι

Συνήθως, βασίζονται σε **ευρετικές λύσεις**

Ευρετική είναι η λύση που εξάγεται μέσω της εμπειρίας

Approximation Algorithms (3/16)

Αν υιοθετήσουμε ένα προσεγγιστικό αλγόριθμο το ερώτημα είναι το πόσο κοντά είναι το αποτέλεσμά του με την πραγματική βέλτιστη λύση

Η **ακρίβεια (accuracy)** είναι το βασικό μέγεθος σε αυτές τις περιπτώσεις

Μπορούμε να δούμε την ακρίβεια ως ένα πρόβλημα ελαχιστοποίησης του σφάλματος για μια προσεγγιστική λύση s_a με τη βοήθεια μιας συνάρτησης f (s* είναι οποιαδήποτε λύση του προβλήματος)

$$re(s_a) = \frac{f(s_a) - f(s^*)}{f(s^*)}$$

Approximation Algorithms (4/16)

Επίσης μπορούμε να υιοθετήσουμε το **λόγο ακρίβειας (accuracy ratio)**

$$r(s_a) = \frac{f(s_a)}{f(s^*)}$$

Αν πρόκειται για πρόβλημα μεγιστοποίησης τότε υιοθετούμε το λόγο

$$r(s_a) = \frac{f(s^*)}{f(s_a)}$$

Θέλουμε την ακρίβεια σφάλματος κοντά στο 0 ενώ το λόγο ακρίβειας κοντά στο 1

Approximation Algorithms (5/16)

Ορισμός

- **Ένας προσεγγιστικός αλγόριθμος πολυωνυμικού χρόνου λέμε ότι είναι ένας c-προσεγγιστικός αλγόριθμος (c-approximation algorithm), με c>1, αν ο λόγος ακρίβειας που παράγει δεν ξεπερνά το c για οποιοδήποτε στιγμιότυπο του προβλήματος**

$$r(s_a) \leq c$$

Η βέλτιστη τιμή του c καλείται **λόγος επίδοσης (performance ratio)**

Approximation Algorithms (6/16)

Approximations of the Traveling Salesman problem

- Οι απλούστερες προσεγγίσεις βασίζονται στην άπληστη μέθοδο
  - Nearest neighbor algorithm
  - Βήμα 1: επιλογή μιας πόλης ως αφετηρία
  - Βήμα 2: επαναληπτικά μέχρι όλες οι πόλεις να έχουν επισκεφθεί – πηγαίνουμε σε μια πόλη που δεν έχουμε επισκεφθεί που είναι πιο κοντά σε μια πόλη που έχουμε επισκεφθεί τελευταία
  - Βήμα 3: επιστρέφουμε στην αφετηρία

Approximation Algorithms (7/16)

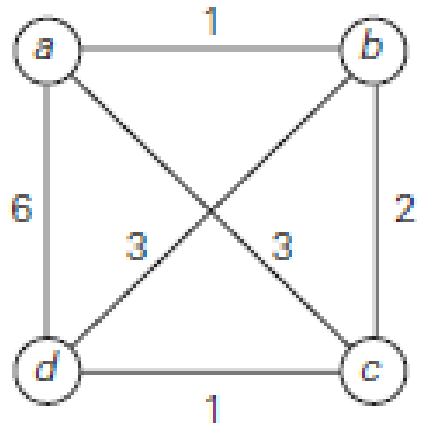
Approximations of the Traveling Salesman problem

- Οι απλούστερες προσεγγίσεις βασίζονται στην άπληστη μέθοδο
  - Nearest neighbor algorithm
  - Παράδειγμα

    s_a: $a - b - c - d - a$ of length 10.
  - Βέλτιστη λύση με exhaustive search

    s^*: $a - b - d - c - a$ of length 8

$$r(s_a) = \frac{f(s_a)}{f(s^*)} = \frac{10}{8} = 1.25$$



Approximation Algorithms (8/16)

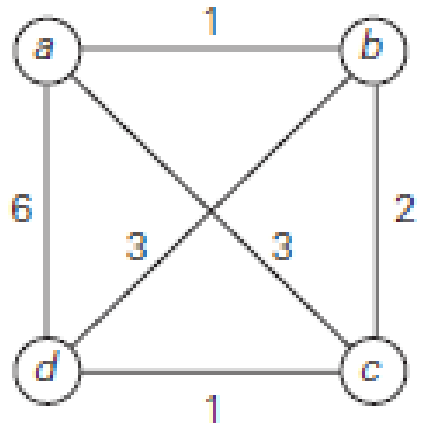
Approximations of the Traveling Salesman problem

- Οι απλούστερες προσεγγίσεις βασίζονται στην άπληστη μέθοδο
  - Multifragment-heuristic algorithm
  - Βήμα 1: ταξινόμηση των ακμών σε αύξουσα σειρά βαρών - Αρχικοποίηση του συνόλου των ακμών της διαδρομής ως το κενό σύνολο
  - Βήμα 2: επαναληπτικά n φορές – προσθήκη της επόμενης ακμής από την ταξινομημένη λίστα στο σύνολο των ακμών της διαδρομής δεδομένου ότι δεν δημιουργούνται κύκλοι; Διαφορετικά απορρίπτουμε την ακμή
  - Βήμα 3: επιστρέφουμε το σύνολο των ακμών

Approximation Algorithms (9/16)

Approximations of the Traveling Salesman problem

- Οι απλούστερες προσεγγίσεις βασίζονται στην άπληστη μέθοδο
  - Multifragment-heuristic algorithm
  - Παράδειγμα
  - Λύση: (a,b), (c,d), (b,c), (a,d)



Approximation Algorithms (10/16)

Approximations of the Traveling Salesman problem

- Minimum spanning tree algorithms
  - Αν διαγράψουμε μια ακμή από ένα Hamiltonian circuit προκύπτει ένα MST
  - Άρα, ένα MST αποτελεί μια καλή βάση για την εύρεση μιας προσέγγισης της τελικής διαδρομής
  - Twice-around-the-tree algorithm
    - Βήμα 1: δημιουργία ενός MST του γράφου
    - Βήμα 2: ξεκινώντας από μια ακμή, εκτελούμε μια διαδρομή στο MST καταγράφοντας τους κόμβους (π.χ. με DFS)
    - Βήμα 3: βλέποντας τους κόμβους της λίστας που προκύπτει από το προηγούμενο βήμα, εξαλείφουμε όλες τις επαναλαμβανόμενες εμφανίσεις κόμβων εκτός από τον πρώτο και τον τελευταίο

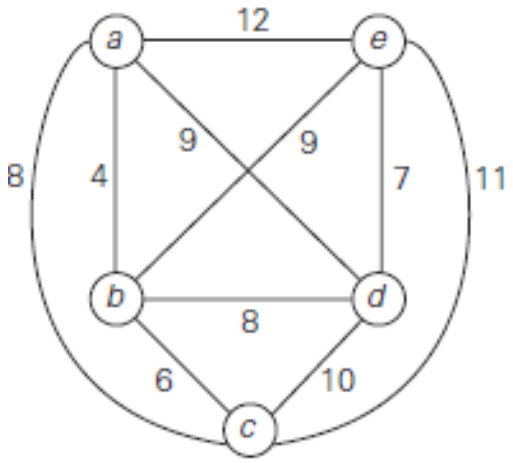
Approximation Algorithms (11/16)

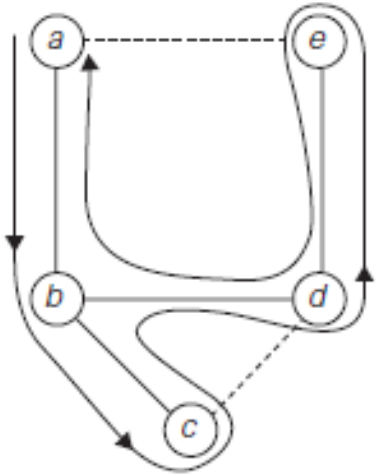
Approximations of the Traveling Salesman problem

- Minimum spanning tree algorithms
  - Twice-around-the-tree algorithm
    - Παράδειγμα
    - Αρχική λίστα κόμβων

      $a,\ b,\ c,\ b,\ d,\ e,\ d,\ b,\ a$
    - Τελική λίστα (μήκος 39)

      $a,\ b,\ c,\ d,\ e,\ a$





Approximation Algorithms (12/16)

Approximations of the Knapsack problem

- Άπληστη μέθοδος
  - Greedy algorithm for the discrete knapsack problem
    - Βήμα 1: υπολογίζουμε το λόγο αξία προς βάρος
    - Βήμα 2: ταξινομούμε τα αντικείμενα σε μη αύξουσα σειρά ως προς το λόγο που υπολογίσαμε
    - Βήμα 3: επαναληπτικά μέχρι να μην υπάρχουν στοιχεία μέσα στη λίστα – αν το επόμενο αντικείμενο χωράει στο σακίδιο, το τοποθετούμε και προχωράμε στο επόμενο; Διαφορετικά, απλά προχωράμε στο επόμενο αντικείμενο

Approximation Algorithms (13/16)

Approximations of the Knapsack problem

- Άπληστη μέθοδος
  - Greedy algorithm for the discrete knapsack problem
    - Παράδειγμα:
    - W=10
    - Επιλέγουμε τα αντικείμενα 1, 3

item	weight	value
1	7	$42
2	3	$12
3	4	$40
4	5	$25

item	weight	value	value/weight
1	4	$40	10
2	7	$42	6
3	5	$25	5
4	3	$12	4

Approximation Algorithms (14/16)

Approximations of the Knapsack problem

- Άπληστη μέθοδος
  - Greedy algorithm for the discrete knapsack problem
    - Παράδειγμα μη εύρεσης της βέλτιστης λύσης:

item	weight	value	value/weight
1	1	2	2
2	W	W	1

The knapsack capacity is $W > 2$.

Approximation Algorithms (15/16)

Approximations of the Knapsack problem

- Άπληστη μέθοδος
  - Greedy algorithm for the continuous knapsack problem
    - Βήμα 1: υπολογίζουμε το λόγο αξία προς βάρος
    - Βήμα 2: ταξινομούμε τα αντικείμενα σε μη αύξουσα σειρά ως προς το λόγο που υπολογίσαμε
    - Βήμα 3: επαναληπτικά μέχρι να μην υπάρχουν στοιχεία μέσα στη λίστα – αν το επόμενο αντικείμενο χωράει στο σακίδιο ολόκληρο, το τοποθετούμε και προχωράμε στο επόμενο; Διαφορετικά, παίρνουμε το ποσοστό του αντικειμένου που χωράει στο σακίδιο και σταματάμε

Approximation Algorithms (16/16)

Approximations of the Knapsack problem

- Άπληστη μέθοδος
  - Greedy algorithm for the discrete knapsack problem
    - Παράδειγμα:
    - W=10
    - Επιλέγουμε τα αντικείμενα 1, 2 (από το δεύτερο παίρνουμε τα 6/7)

item	weight	value
1	7	$42
2	3	$12
3	4	$40
4	5	$25

item	weight	value	value/weight
1	4	$40	10
2	7	$42	6
3	5	$25	5
4	3	$12	4