Αλγόριθμοι online

Βελτιστοποίηση δικτύων

27 Ιανουαρίου 2023

Το πρόβλημα του χιονοδρόμου (the skiing problem)

- Ένας αρχάριος χιονοδρόμος γνωρίζει ότι το κόστος αγοράς του εξοπλισμού είναι €300, ενώ το ημερήσιο κόστος ενοικίασης €30. Δεδομένου ότι δεν γνωρίζει πόσες φορές θα χρησιμοποιήσει συνολικά τον εξοπλισμό, τι πρέπει να κάνει; [Karlin, 1988]
- Πιθανές απαντήσεις:
  1. Να αγοράσει αμέσως τον εξοπλισμό πληρώνοντας συνολικά €300.
  2. Να ενοικιάζει πάντοτε, με €30 κάθε φορά.
  3. Να αγοράσει αφού ενοικιάσει πρώτα b φορές. Πώς θα προσδιορίσει το b;
- Παρατήρηση: Αν π.χ. προτιμήσει τη λύση (1) και κάνει σκι μόνο μια φορά, θα έχει πληρώσει τα δεκαπλάσια από το ελάχιστο δυνατό, που θα είχε επιτύχει με ενοικίαση (λύσεις 2 ή 3).

Online vs offline problem

- Το πρόβλημα αυτό στο οποίο δεν είναι γνωστές όλες οι παράμετροι εξ αρχής λέγεται online.
- Το αντίστοιχο κλασσικό πρόβλημα βελτιστοποίησης που διαθέτει τη γνώση όλων των παραμέτρων λέγεται offline.
- Στο συγκεκριμένο παράδειγμα η offline εκδοχή του προβλήματος είναι αυτή όπου θα ήταν γνωστός ο συνολικός αριθμός t χρήσεων του εξοπλισμού.
- Αν ο χιονοδρόμος γνώριζε ότι θα χρησιμοποιήσει τον εξοπλισμό ακριβώς t φορές, θα έκανε αμέσως αγορά $t \geq 10$, ενώ θα προτιμούσε την ενοικίαση αν $t \leq 10$.
- Επομένως το ελάχιστο κόστος θα ήταν

$$F_{\text{off}}(t) = \min\{30t, 300\}$$

Λόγος online / offline

- Ο χιονοδρόμος χρειάζεται μια μέθοδο λήψης απόφασης που θα δώσει κόστος όσο γίνεται πιο κοντινό στο βέλτιστο $F_{\text{off}}(t)$.
- Έστω ότι χρησιμοποιεί τον αλγόριθμο A_b «αγοράζω μετά από b μέρες ενοικίασης».
- Το συνολικό κόστος που προκύπτει από τον A_b εξαρτάται από το συνολικό αριθμό ημερών t.
- Αν σταματήσει στις t μέρες, με χρήση του A_b θα πληρώσει

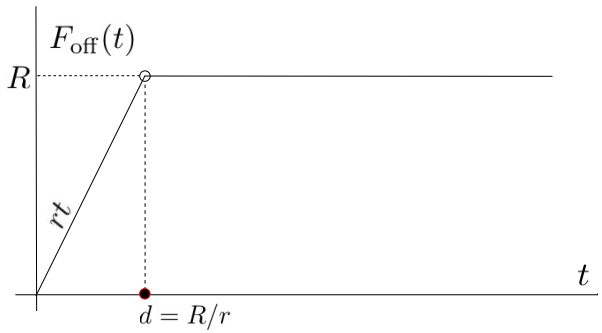
$$F_{A_b}(t) = 30t \times \chi\{t \leq b\} + (30b + 300) \times \chi\{t > b\}$$

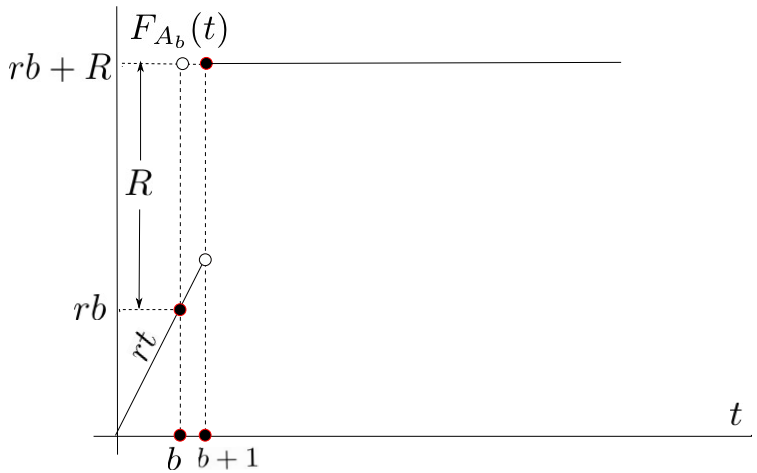
όπου $\chi\{E\} = 1$ αν το E είναι αληθές, αλλιώς $\chi\{E\} = 0$.
- Θέλουμε να βρούμε την τιμή του b που ελαχιστοποιεί το

$$\rho_b = \max_t \frac{F_{A_b}(t)}{F_{\text{off}}(t)}$$

Οι δύο αλγόριθμοι, βέλτιστος και online

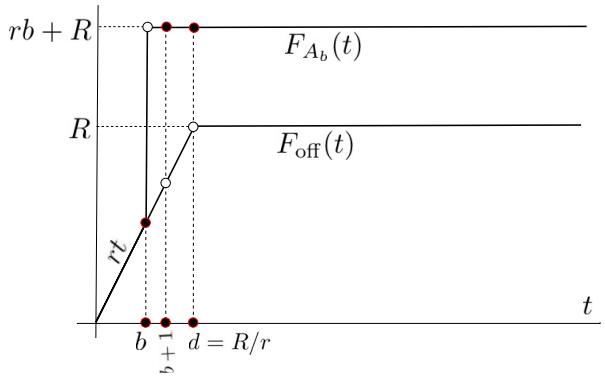
Επομένως έχουμε να συγκρίνουμε δύο αλγόριθμους, των οποίων οι επιδόσεις φαίνονται στα παρακάτω δύο σχήματα.





Περίπτωση $b < d$

- Έστω
  - r τιμή ενοικίασης,
  - R η τιμή αγοράς,
  - $d = R/r$ (με ακέραιο d) και
  - $b < d$.



- Ο λόγος του online προς τον βέλτιστο είναι

$$\rho_b(t) = \begin{cases} 1 & 1 \le t \le b \\ \frac{r\,b+R}{r\,t} & b+1 \le t \le d \\ \frac{r\,b+R}{R} & d \le t \end{cases}$$

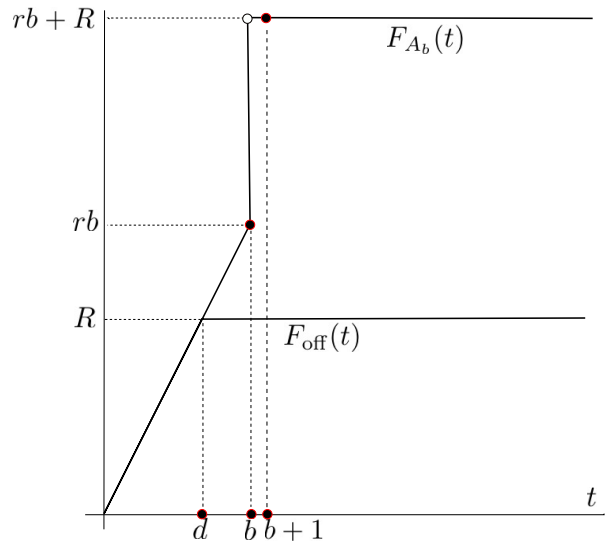
- Η μέγιστη τιμή συμβαίνει για $t = b + 1$, όπου

$$\rho_b(b+1) = \frac{rb+R}{r(b+1)} = \frac{rb+rd}{rb+r} = \frac{b+d}{b+1}$$

Παράδειγμα: $b > d$

- Ο λόγος του online προς τον βέλτιστο είναι

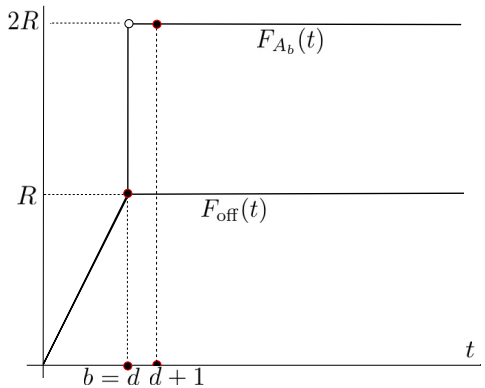
$$\rho_b(t) = \begin{cases} 1 & 1 \le t \le d \\ \frac{r\,t}{R} & d+1 \le t \le b \\ \frac{r\,b+R}{R} & b+1 \le t \end{cases}$$



- Η μέγιστη τιμή συμβαίνει για $t \ge b+1$, όπου

$$\rho_b(b+1) = \frac{rb+R}{R} = \frac{rb+rd}{rd} = 1 + \frac{b}{d} > 2$$

Παράδειγμα: $b = d$



- Ο λόγος του online προς τον βέλτιστο είναι

$$\rho_b(t) = \begin{cases} 1 & 1 \leq t \leq d = b \\ 2 & d + 1 \leq t \end{cases}$$

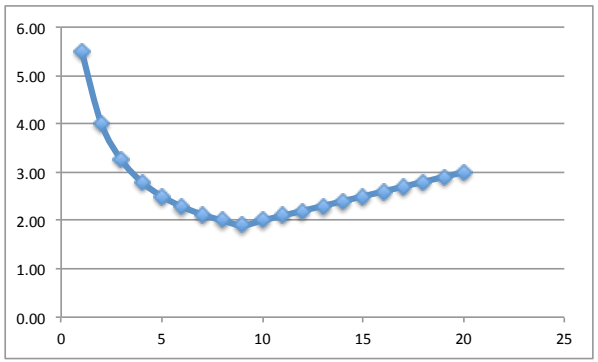
- Η μέγιστη τιμή συμβαίνει για οποιοδήποτε $t' \geq b + 1$, όπου

$$\rho_b(t') = 2$$

Ελαχιστοποίηση ως προς b

- Συνοψίζοντας

$$\rho_b = \begin{cases} \frac{b+d}{b+1} & b < d \\ 1 + \frac{b}{d} & b \geq d \end{cases}$$



- Το ελάχιστο ρ_b επιτυγχάνεται για $b = d - 1$, δηλαδή ο χιονοδρόμος πρέπει να αγοράσει την d-οστή μέρα, όπου $d = R/r$, και είναι ίσο με

$$\rho_b^* = \frac{d - 1 + d}{d - 1 + 1} = 2 - \frac{1}{d}$$

Η μέθοδος του αντίπαλου

- Η απόδοση ενός αλγορίθμου μπορεί να εξετασθεί με τη μέθοδο του αντίπαλου δαίμονα, ο οποίος κανονίζει πότε θα λήξει η δραστηριότητα του X (π.χ. σταματάει το σκι γιατί έσπασε το πόδι του).
- Η βέλτιστη τακτική του δαίμονα στην περίπτωση του χιονοδρόμου είναι να προκαλέσει τη λήξη της απόσβεσης αμέσως μετά την αγορά.

Η χιονοδρομία με τη μέθοδο αντιπάλου

- Έστω ότι η τακτική του χιονοδρόμου είναι να αγοράσει μετά από b φορές, για κάποιο b που έχει επιλέξει (δηλαδή εκτελεί τον A_b).
- Ο αντίπαλος επιλέγει το t, τον συνολικό αριθμό χιονοδρομιών, και θέτει $t = b + 1$ προκειμένου να αναγκάσει το χιονοδρόμο να πληρώσει το μεγάλο ποσό αγοράς, αλλά να κάνει στη συνέχεια μηδενική απόσβεση.
- Άρα ο χιονοδρόμος πληρώνει $rb + R$.
- Ωστόσο, αν εγνώριζε την τιμή του $t = b + 1$, θα μπορούσε να είχε πληρώσει $\min\{r(b+1), R\}$.
- Ο λόγος των παραπάνω

$$\frac{r\,b + R}{\min\{r(b+1), R\}}$$

ελαχιστοποιείται για $b = \frac{R}{r} - 1$ και γίνεται τότε ίσος με $2 - 1/d$.

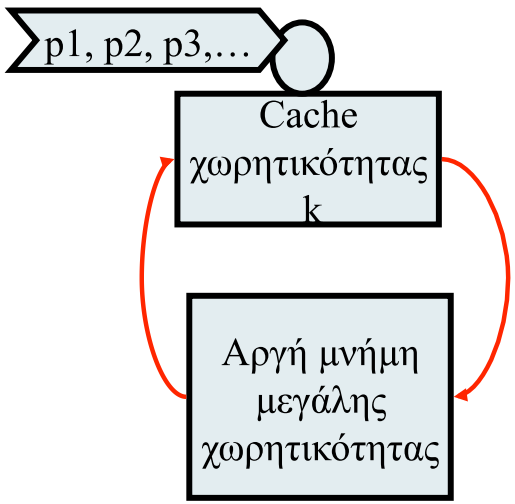
Η «αρχή της χιονοδρομίας»

Ένα γενικό συμπέρασμα που μπορεί να εξαχθεί από τα προηγούμενα είναι το εξής:

- Είναι καλύτερο σε ορισμένες περιπτώσεις να προτιμήσει κανείς ένα βήμα αυξημένου κόστους αντί βήματος μικρού κόστους και μάλιστα όταν συσσωρεύονται πολλά βήματα μικρού κόστους.

Cache memory problem

- Μια δεδομένη ακολουθία σελίδων αναζητείται από τον επεξεργαστή στη γρήγορη μνήμη.
- Αν η σελίδα είναι ήδη στη γρήγορη μνήμη, το κόστος ανάγνωσης είναι 0.
- Αν είναι στην αργή μνήμη, το κόστος είναι 1 για να «ανέβει» στη γρήγορη μνήμη.
- Επειδή είναι προφανώς βέλτιστο να κρατάμε το cache γεμάτο, το πρόβλημα είναι ποια σελίδα πρέπει να «κατέβει στην αργή».
- Ζητείται μια τακτική επιλογής της σελίδας που θα «κατεβαίνει» σε κάθε βήμα.



Ο αλγόριθμος επιλογής σελίδων

- Βέλτιστος (offline) είναι αυτός που απομακρύνει από τη γρήγορη μνήμη τη σελίδα, της οποίας η ζήτηση είναι η πιο απομακρυσμένη στο μέλλον.
- Ένας δημοφιλής αλγόριθμος online είναι αυτός που απομακρύνει τη σελίδα με την πιο παλιά ζήτηση.
- Ένας άλλος δημοφιλής αλγόριθμος online είναι αυτός που απομακρύνει τη σελίδα με την πιο πρόσφατη ζήτηση.
- Ποιος από τους παραπάνω είναι πιθανό να είναι πιο αποτελεσματικός;

Ανάλυση αντιπάλου για τον αλγόριθμο επιλογής σελίδων

- Όλοι οι αλγόριθμοι επιλογής σελίδων είναι από την άποψη του αντιπάλου το ίδιο κακοί, δεδομένου ότι όποιος κι αν είναι ο αλγόριθμος απομάκρυνσης, ο αντίπαλος μπορεί πάντοτε να ζητάει μια σελίδα που δεν είναι στη γρήγορη μνήμη.
- Έχουν προταθεί μελέτες του μέσου κόστους, αλλά εξαρτώνται από την επιλογή της κατανομής εισόδου (για την επιλογή των ζητουμένων σελίδων).
- Οι Sleator και Tarjan πρότειναν την ανταγωνιστική ανάλυση (competitive analysis)

Ανταγωνιστική ανάλυση (competitive analysis)

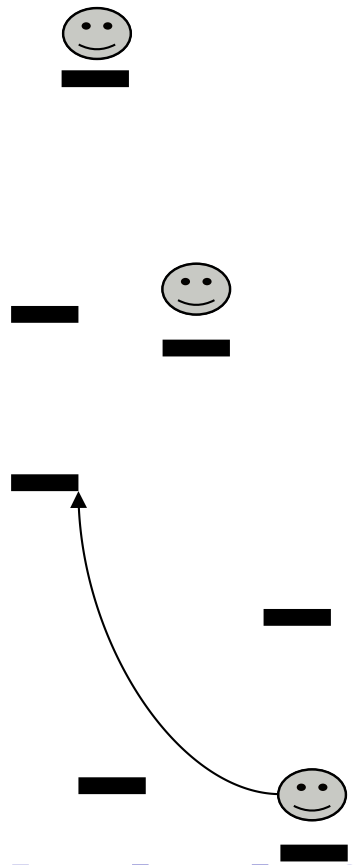
- Για την είσοδο σ υπολογίζεται η επίδοση (κόστος $\text{cost}(\sigma)$) του βέλτιστου (offline) αλγορίθμου (έστω OPT) και του αλγορίθμου online (έστω A).
- Λέγεται ο A ανταγωνιστικός κατά παράγοντα c (c competitive), όταν για κάθε ακολουθία εισόδου σ

$$\text{cost}_A(\sigma) \leq c \times \text{cost}_{\text{OPT}}(\sigma) + b$$

- Ανταγωνιστικός λόγος (competitive ratio) c_A για τον αλγόριθμο A λέγεται το infimum των c, που είναι τέτοια ώστε ο A να είναι ανταγωνιστικός κατά παράγοντα c.
- Ο αλγόριθμος A λέγεται ισχυρά ανταγωνιστικός εφόσον επιτυγχάνει τον καλύτερο δυνατό ανταγωνιστικό λόγο.
- Τα παραπάνω είναι ανεξάρτητα από την υπολογιστική πολυπλοκότητα του βέλτιστου αλγόριθμου επίλυσης του προβλήματος. Η εξάρτηση του ανταγωνιστικού λόγου από την τάξη πολυπλοκότητας είναι ανοιχτή περιοχή έρευνας.

Το πρόβλημα των k υπηρετών (k server problem)

- Δίνεται μετρικός χώρος M και k υπηρέτες σε δεδομένα σημεία του M.
- Η ακολουθία εισόδου σ αποτελείται από σημεία του M, όπου πρέπει να μετακινηθεί κάθε φορά ένας οιοσδήποτε υπηρέτης (πριν την εμφάνιση του επόμενου σημείου εισόδου).
- Το κόστος μετακίνησης ενός υπηρέτη εξαρτάται από την απόσταση του σημείου από την τρέχουσα θέση του.
- Στόχος είναι να ελαχιστοποιηθεί το συνολικό κόστος των μετακινήσεων.
- Το πρόβλημα επιλογής σελίδων είναι ειδική περίπτωση του προβλήματος των k υπηρετών.



Η τεχνική του σκληρού αντιπάλου

- Χρησιμοποιείται για την απόδειξη κάτω φραγμάτων του ανταγωνιστικού λόγου.
- Ο αντίπαλος κάνει αμφότερα τα εξής:
  - Γεννάει την ακολουθία εισόδου και
  - εκτελεί την ακολουθία των δράσεων.
- Σκοπός του είναι να βρει μια ακολουθία, με την οποία ο αλγόριθμος online επιφέρει υψηλό κόστος, όταν υπάρχει offline χαμηλό.

Για το cache memory problem και οποιονδήποτε αλγόριθμο A ισχύει $c_A \geq k$, μέρος I

- Ας υποτεθεί ότι οι A και OPT αρχίζουν με τις ίδιες σελίδες στη γρήγορη μνήμη (cache).
- Ο αντίπαλος χρησιμοποιεί στην ακολουθία σ μόνο $k+1$ σελίδες, όπου k είναι το μέγεθος της γρήγορης μνήμης, και επιλέγει πάντοτε τη σελίδα που είναι εκτός μνήμης, οπότε γίνεται αποτυχία σε κάθε βήμα και

$$\text{cost}_A(\sigma) = |\sigma|$$

Για το cache memory problem και οποιονδήποτε αλγόριθμο A ισχύει $c_A \geq k$, μέρος II

- Σύμφωνα με τον OPT απομακρύνεται η σελίδα, που η πρώτη της ζήτηση είναι η πιο μακρινή στο μέλλον. Δηλαδή με τον OPT γίνεται το πολύ ένα λάθος κάθε k γεγονότα, οπότε
$$\text{cost}_{\text{OPT}}(\sigma) \leq \left\lceil \frac{|\sigma|}{k} \right\rceil$$
- Μπορείτε τώρα να παρατηρήσετε ότι ο αντίπαλος δουλεύει με $k+1$ μόνο σελίδες ακριβώς για να επιτύχει ένα λάθος κάθε k, δηλαδή για να κάνει ελάχιστο το κόστος του OPT.

Μια παρατήρηση:

- Σε πολλά προβλήματα η γνώση του βέλτιστου αλγόριθμου δεν είναι διαθέσιμη, π.χ. όταν το πρόβλημα είναι NP-πλήρες.
- Ωστόσο ένα άνω φράγμα για το κόστος του αντιπάλου δίνει ένα κάτω φράγμα για τον ανταγωνιστικό λόγο.

Η τεχνική της συλλογής αλγορίθμων

Δυστυχώς το κόστος του βέλτιστου αλγόριθμου είναι συχνά δύσκολο να υπολογισθεί. Η παρακάτω τεχνική βρίσκει ένα κάτω φράγμα για το ανταγωνιστικό κόστος.

- Επινοείται μια συλλογή αλγορίθμων $A_1, A_2, \ldots, A_m$ που είναι τέτοιοι, ώστε όταν ο αντίπαλος κατασκευάσει την ακολουθία εισόδου σ, για το συνολικό κόστος εκτέλεσης όλων μαζί των αλγορίθμων να ισχύει το εξής:

$$\sum_{i=1}^{m} \text{cost}_{A_i}(\sigma) \leq c\, \text{cost}_A(\sigma)$$

- Υπάρχει αλγόριθμος A_{i_0} της συλλογής με κόστος το πολύ ίσο με το μέσο, άρα

$$\begin{aligned} \text{cost}_{\text{OPT}}(\sigma) &\leq \text{cost}_{A_{i_0}}(\sigma) \leq \frac{1}{m}\sum_{i=1}^{m}\text{cost}_{A_i}(\sigma) \leq \frac{c}{m}\,\text{cost}_A(\sigma) \\ &\Rightarrow \quad \text{cost}_A(\sigma) \geq \frac{m}{c}\,\text{cost}_{\text{OPT}}(\sigma) \end{aligned}$$

άρα το ανταγωνιστικό κόστος είναι το λιγότερο ίσο με m/c.

Εφαρμογή στο k-server problem

- Ο αντίπαλος επιλέγει μόνο $k+1$ σημεία $\{p_1, \ldots, p_{k+1}\}$, όπου αρχικά μόνο το τελευταίο είναι κενό. Κάθε φορά επιλέγει να κάνει αίτηση για το σημείο που είναι κενό.
- Κατασκευάζουμε k αλγόριθμους $A_1, \ldots, A_k$ που λειτουργούν ως εξής: Αρχικά έχουν υπηρέτες στα σημεία $p_1, \ldots, p_k$. Στην πρώτη αίτηση, ο A_i μετακινεί τον υπηρέτη του σημείου p_i στο p_{k+1}.
- Κατόπιν σε κάθε επόμενη αίτηση συνεχίζουν να έχουν ένα υπηρέτη στο σημείο της τελευταίας αίτησης και να μην έχουν κοινό κενό σημείο.

A

p1	p2	p3	p4	move to
1	2	3		p4
	2	3	1	p1
2		3	1	p2
2	3		1	p3
2	3	1		p4
	3	1	2	p1
3		1	2	p2
3	1		2	p3

Βήμα 1
1α . Να μετακινηθεί κάποιος στο σημείο p4.
1β. Ας μετακινηθεί ο 1

A1

p1	p2	p3	p4	mv
1	2	3		p4
	2	3	1	p1
1	2	3		p2
1	2	3		p3
1	2	3		p4
1	2		3	p1
1	2		3	p2
1	2		3	p3

A2

p1	p2	p3	p4	mv
1	2	3		p4
1		3	2	p1
1		3	2	p2
	1	3	2	p3
	1	3	2	p4
	1	3	2	p1
2	1	3		p2
2	1	3		p3

A3

p1	p2	p3	p4	mv
1	2	3		p4
1	2		3	p1
1	2		3	p2
1	2		3	p3
1		2	3	p4
1		2	3	p1
1		2	3	p2
	1	2	3	p3

Βιβλιογραφία

 Dorit Hochbaum, "Deterministic Algorithms for NP-hard problems", PWS Publishing Company, Boston, 1997.

 Karlin, Anna R., "On the performance of competitive algorithms in practice." *Online Algorithms*, pp. 373-384, Springer, 1998.

Τελευταία μεταβολή στις 27 Ιανουαρίου 2023, 09:19.