4 ΠΛΗΡΟΦΟΡΗΜΕΝΗ ΑΝΑΖΗΤΗΣΗ ΚΑΙ ΕΞΕΡΕΥΝΗΣΗ

Όπου θα δούμε πώς η πληροφόρηση για το χώρο καταστάσεων μπορεί να απαλλάξει τους αλγόριθμους από το παραπάτημα στο σκοτάδι.

Στο Κεφάλαιο 3 είδαμε ότι οι στρατηγικές απληροφόρητης αναζήτησης μπορούν να βρίσκουν λύσεις σε προβλήματα με τη συστηματική παραγωγή νέων καταστάσεων και με την εξέτασή τους σε σχέση με τον στόχο. Δυστυχώς, οι στρατηγικές αυτές είναι απίστευτα ανεπαρκείς στις περισσότερες περιπτώσεις. Σε αυτό το κεφάλαιο θα δούμε πώς μια στρατηγική πληροφορημένης αναζήτησης — που χρησιμοποιεί ειδική γνώση του προβλήματος — μπορεί να βρίσκει λύσεις πιο αποδοτικά. Η Ενότητα 4.1 περιγράφει εκδόσεις των αλγορίθμων του Κεφάλαιο 3 με πληροφόρηση και η Ενότητα 4.2 εξηγεί πώς μπορεί να αποκτηθεί η απαιτούμενη ειδική πληροφόρηση για το πρόβλημα. Οι Ενότητες 4.3 και 4.4 καλύπτουν αλγόριθμους που πραγματοποιούν καθαρά **τοπική αναζήτηση** (local search) στο χώρο των καταστάσεων, αξιολογώντας και τροποποιώντας μία ή περισσότερες τρέχουσες καταστάσεις αντί να εξερευνούν συστηματικά διαδρομές από μια αρχική κατάσταση. Οι αλγόριθμοι αυτοί είναι κατάλληλοι για προβλήματα στα οποία το κόστος διαδρομής είναι αδιάφορο και το μόνο που έχει σημασία είναι η ίδια η κατάσταση λύσης. Η οικογένεια των αλγορίθμων τοπικής αναζήτησης περιλαμβάνει μεθόδους εμπνευσμένες από τη στατιστική φυσική (**προσομοιωμένη ανόπτηση** — simulated annealing) και την εξελικτική βιολογία (**γενετικοί αλγόριθμοι**). Τέλος, η Ενότητα 4.5 εξερευνά την **online αναζήτηση** (online search), όπου ο πράκτορας βρίσκεται αντιμέτωπος με ένα χώρο καταστάσεων που του είναι εντελώς άγνωστος.

4.1 ΣΤΡΑΤΗΓΙΚΕΣ ΠΛΗΡΟΦΟΡΗΜΕΝΗΣ (ΕΥΡΕΤΙΚΗΣ) ΑΝΑΖΗΤΗΣΗΣ

ΠΛΗΡΟΦΟΡΗΜΕΝΗ ΑΝΑΖΗΤΗΣΗ

Σε αυτή την ενότητα θα δούμε πώς μια στρατηγική **πληροφορημένης αναζήτησης** (informed search) — η οποία χρησιμοποιεί ειδική γνώση του προβλήματος, πέρα από τον ίδιο τον ορισμό του προβλήματος — μπορεί να βρίσκει λύσεις πιο αποδοτικά από μια στρατηγική χωρίς πληροφόρηση.

ΑΝΑΖΗΤΗΣΗ ΠΡΩΤΑ ΣΤΟ ΚΑΛΥΤΕΡΟ

Η γενική προσέγγιση που θα εξετάσουμε ονομάζεται **αναζήτηση πρώτα στο καλύτερο** (best-first search). Η αναζήτηση πρώτα στο καλύτερο είναι μια περίπτωση των γενικών αλγορίθμων TREE-SEARCH ή GRAPH-SEARCH στην οποίο ένας κόμβος επιλέγεται για να επεκταθεί με βάση μια **συνάρτηση αξιολόγησης** (evaluation function), $f(n)$. Κατά παράδοση, επιλέγεται για επέκταση ο κόμβος με τη *μικρότερη* τιμή αξιολόγησης, επειδή η αξιολόγηση μετρά την απόσταση από τον στόχο. Στο γενικό μας πλαίσιο για την αναζήτηση, η αναζήτηση πρώτα στο καλύτερο

ΣΥΝΑΡΤΗΣΗ ΑΞΙΟΛΟΓΗΣΗΣ

μπορεί να υλοποιηθεί με μια ουρά προτεραιότητας, μια δομή δεδομένων που θα διατηρεί πάντα το σύνορο σε αύξουσα διάταξη των τιμών *f*.

Η ονομασία της αναζήτησης πρώτα στο καλύτερο είναι παραδοσιακή αλλά ανακριβής. Αν το καλοσκεφτείτε, αν μπορούσαμε *πραγματικά* να επεκτείνουμε πρώτο τον καλύτερο κόμβο αυτό βέβαια δε θα ήταν αναζήτηση· θα ήταν προέλαση προς τον στόχο. Το μόνο που μπορούμε να κάνουμε είναι να επιλέξουμε τον κόμβο που *φαίνεται* να είναι ο καλύτερος, σύμφωνα με τη συνάρτηση αξιολόγησης. Αν η συνάρτηση αξιολόγησης είναι εντελώς ακριβής, τότε αυτός ο κόμβος θα είναι πραγματικά ο καλύτερος· στην πραγματικότητα, η συνάρτηση αξιολόγησης μερικές φορές είναι άστοχη και μπορεί να κάνει μια αναζήτηση να παραστρατήσει. Ωστόσο, θα μείνουμε στην ονομασία "πρώτα στο καλύτερο" επειδή ο όρος "πρώτα στο φαινομενικά καλύτερο" είναι κάπως αδέξιος.

ΕΥΡΕΤΙΚΗ ΣΥΝΑΡΤΗΣΗ

Υπάρχει μια ολόκληρη οικογένεια αλγορίθμων BEST-FIRST-SEARCH με διαφορετικές συναρτήσεις αξιολόγησης.[1] Καθοριστικό στοιχείο αυτών των αλγορίθμων είναι μια **ευρετική συνάρτηση** (heuristic function),[2] η οποία συμβολίζεται ως $h(n)$:

$h(n)$ = εκτιμώμενο κόστος φθηνότερης διαδρομής από κόμβο n σε έναν κόμβο στόχου.

Στο παράδειγμα της Ρουμανίας, θα μπορούσε κανείς να εκτιμήσει το κόστος της φθηνότερης διαδρομής από το Arad στο Βουκουρέστι με βάση την ευθύγραμμη απόσταση των δύο πόλεων.

Οι ευρετικές συναρτήσεις είναι η πιο συνηθισμένη μορφή με την οποία μεταδίδεται πρόσθετη γνώση του προβλήματος στον αλγόριθμο αναζήτησης. Θα μελετήσουμε τους ευρετικούς μηχανισμούς σε μεγαλύτερο βάθος στην Ενότητα 4.2. Για την ώρα, θα τους θεωρούμε ως συνηθισμένες συναρτήσεις ειδικές για το συγκεκριμένο πρόβλημα, με έναν περιορισμό: Αν ο n είναι κόμβος στόχου, τότε $h(n) = 0$. Το υπόλοιπο αυτής της ενότητας καλύπτει δύο τρόπους χρήσης ευρετικής πληροφόρησης για την καθοδήγηση της αναζήτησης.

Άπληστη αναζήτηση πρώτα στο καλύτερο

ΑΠΛΗΣΤΗ ΑΝΑΖΗΤΗΣΗ ΠΡΩΤΑ ΣΤΟ ΚΑΛΥΤΕΡΟ

Η **άπληστη αναζήτηση πρώτα στο καλύτερο** (greedy best-first search)[3] προσπαθεί να επεκτείνει τον κόμβο που είναι ο πιο κοντινός στον στόχο, με το σκεπτικό ότι αυτό είναι πιο πιθανό να οδηγήσει σε γρήγορη λύση. Συνεπώς, αξιολογεί τους κόμβους χρησιμοποιώντας απλώς την ευρετική συνάρτηση: $f(n) = h(n)$.

ΕΥΘΥΓΡΑΜΜΗ ΑΠΟΣΤΑΣΗ

Ας δούμε πώς λειτουργεί αυτή η μέθοδος στα προβλήματα εύρεσης δρομολογίων στη Ρουμανία χρησιμοποιώντας τον ευρετικό μηχανισμό της **ευθύγραμμης απόστασης** (straight-line distance), συντομογραφικά h_{SLD}. Αν ο στόχος είναι το Βουκουρέστι, θα χρειαστεί να γνωρίζουμε τις ευθύγραμμες αποστάσεις για το Βουκουρέστι, οι οποίες δίνονται στον πίνακα της Εικόνας 4.1. Για παράδειγμα, $h_{SLD}(Εντός(Arad)) = 366$. Προσέξτε ότι οι τιμές της h_{SLD} δεν μπορούν να υπολογιστούν από την ίδια την περιγραφή του προβλήματος. Επίσης, χρειάζεται κάποια πείρα για να γνωρίζει κανείς ότι η h_{SLD} σχετίζεται με τις πραγματικές οδικές αποστάσεις, και επομένως είναι ένας χρήσιμος ευρετικός μηχανισμός.

---

[1] Η Άσκηση 4.3 σας ζητά να δείξετε ότι η οικογένεια αυτή περιλαμβάνει πολλούς γνωστούς μας αλγόριθμους χωρίς πληροφόρηση.

[2] Μια ευρετική συνάρτηση $h(n)$ δέχεται ως είσοδο έναν *κόμβο*, αλλά εξαρτάται μόνο από την *κατάσταση* σε αυτόν τον κόμβο.

[3] Στην πρώτη μας έκδοση την ονομάζαμε απλώς **άπληστη αναζήτηση**· άλλοι συγγραφείς την ονομάζουν απλώς **αναζήτηση πρώτα στο καλύτερο**. Ο γενικότερος τρόπος που χρησιμοποιούμε το δεύτερο όρο οφείλεται στον Pearl (1984).

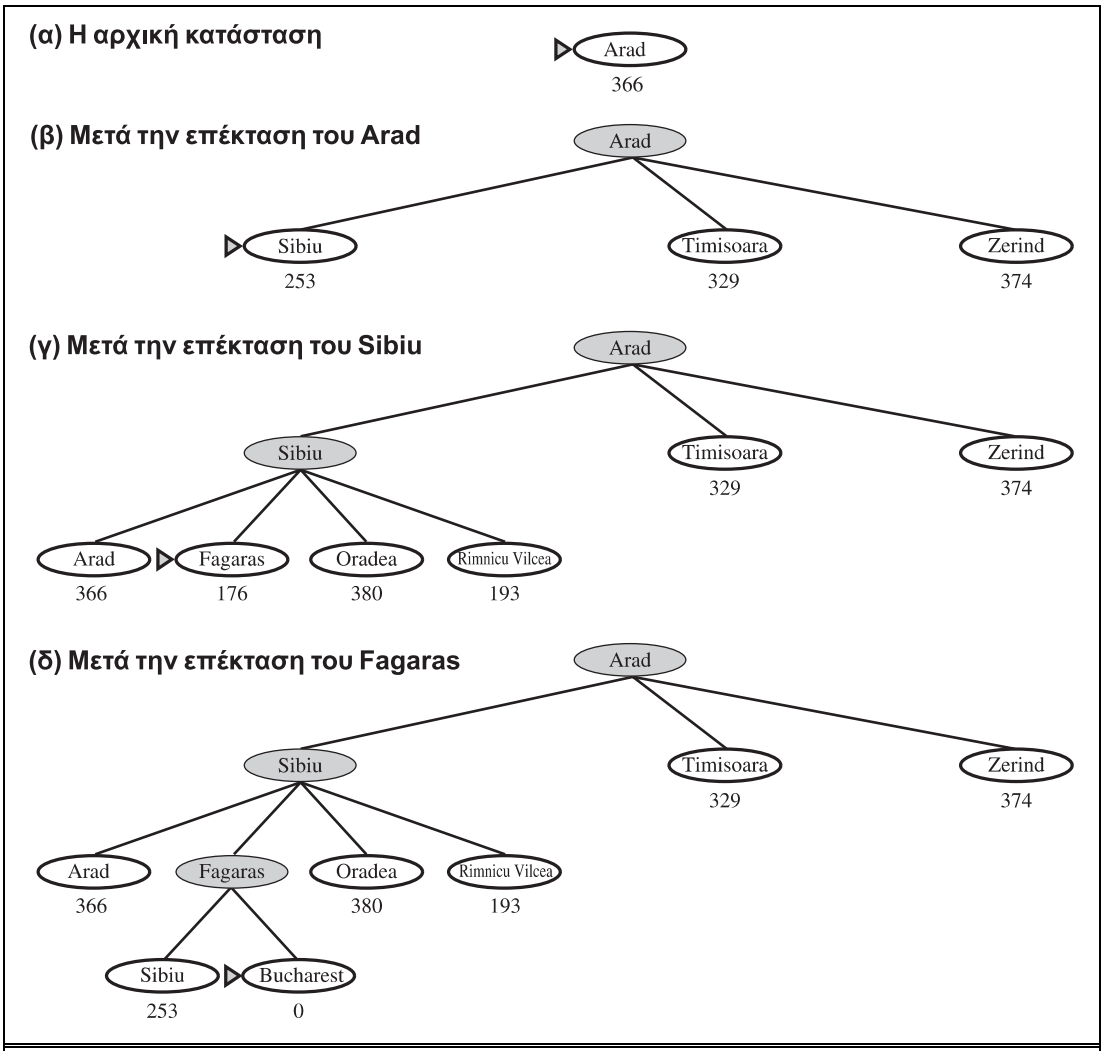
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Dobreta	242	**Pitesti**	100
Eforie	161	**Rimniscu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

Εικόνα 4.1 Τιμές h_{SLD} — ευθύγραμμες αποστάσεις για Βουκουρέστι.

Η Εικόνα 4.2 δείχνει την πρόοδο μιας άπληστης αναζήτηση πρώτα στο καλύτερο που χρησιμοποιεί την h_{SLD} για να βρει μια διαδρομή από το Arad στο Βουκουρέστι. Ο πρώτος κόμβος που θα επεκταθεί από το Arad είναι το Sibiu, επειδή είναι πιο κοντά στο Βουκουρέστι τόσο από το Zerind όσο και από την Timisoara. Ο επόμενος κόμβος που θα επεκταθεί είναι το Fagaras, επειδή είναι ο πιο κοντινός στον στόχο. Το Fagaras, με τη σειρά του, μας δίνει το Βουκουρέστι, το οποίο είναι ο στόχος. Για το συγκεκριμένο πρόβλημα, η άπληστη αναζήτηση πρώτα στο καλύτερο που χρησιμοποιεί την h_{SLD} βρίσκει λύση χωρίς να επεκτείνει ούτε έναν κόμβο που δε βρίσκεται πάνω στη διαδρομή της λύσης· συνεπώς, το κόστος αναζήτησής της είναι ελάχιστο. Όμως, δεν είναι βέλτιστη· η διαδρομή για το Βουκουρέστι μέσω Sibiu και Fagaras είναι 32 χιλιόμετρα μεγαλύτερη από τη διαδρομή μέσω Rimnicu Vilcea και Pitesti. Αυτός είναι ο λόγος που ο αλγόριθμος λέγεται "άπληστος" — σε κάθε βήμα, προσπαθεί να βρεθεί όσο το δυνατόν πιο κοντά στον στόχο.

Η ελαχιστοποίηση της τιμής $h(n)$ είναι επιρρεπής σε λανθασμένες εκκινήσεις. Ας εξετάσουμε το πρόβλημα της μετάβασης από το Iasi στο Fagaras. Ο ευρετικός μηχανισμός υποδεικνύει να επεκταθεί πρώτα ο κόμβος του Neamt επειδή είναι πιο κοντά στο Fagaras, αλλά έτσι φτάνει σε αδιέξοδο. Η λύση είναι να πάμε πρώτα στο Vaslui — ένα βήμα που μας απομακρύνει από τον στόχο σύμφωνα με τον ευρετικό μηχανισμό — και από εκεί να συνεχίσουμε για το Urziceni, για το Βουκουρέστι, και για το Fagaras. Σε αυτή την περίπτωση, λοιπόν, ο ευρετικός μηχανισμός προκαλεί επεκτάσεις κόμβων που δε χρειάζονται. Επίσης, αν δεν είμαστε προσεκτικοί στην ανίχνευση των επαναλαμβανόμενων καταστάσεων η λύση δε θα βρεθεί ποτέ — η αναζήτηση θα πηγαινοέρχεται μεταξύ του Neamt και του Iasi.

Η άπληστη αναζήτηση πρώτα στο καλύτερο μοιάζει με την αναζήτηση πρώτα κατά βάθος κατά το ότι προτιμά να ακολουθεί μία και μόνο διαδρομή σε όλη την πορεία για τον στόχο, αλλά οπισθοχωρεί όταν φτάνει σε αδιέξοδο. Έχει τα ίδια ελαττώματα με την αναζήτηση πρώτα κατά βάθος — δεν είναι ούτε βέλτιστη ούτε πλήρης (επειδή μπορεί να αρχίσει να κατεβαίνει σε μια άπειρη διαδρομή και να μην επιστρέψει ποτέ για να δοκιμάσει άλλες δυνατότητες). Η χρονική και η χωρική πολυπλοκότητα της χειρότερης περίπτωσης είναι $O(b^m)$, όπου m το μέγιστο βάθος του χώρου αναζήτησης. Με μια καλή ευρετική συνάρτηση, όμως, η πολυπλοκότητα μπορεί να μειωθεί σημαντικά. Το πόσο θα μειωθεί εξαρτάται από το συγκεκριμένο πρόβλημα και από την ποιότητα του ευρετικού μηχανισμού.



Εικόνα 4.2 Στάδια μιας άπληστης αναζήτησης πρώτα στο καλύτερο για το Βουκουρέστι, που χρησιμοποιεί τον ευρετικό μηχανισμό της ευθύγραμμης απόστασης, h_{SLD}. Οι τιμές h είναι σημειωμένες κάτω από τους αντίστοιχους κόμβους.

Αναζήτηση A*: Ελαχιστοποίηση του ολικού εκτιμώμενου κόστους λύσης

ΑΝΑΖΗΤΗΣΗ A*

Η ευρύτερα γνωστή μορφή αναζήτησης πρώτα στο καλύτερο ονομάζεται **αναζήτηση A*** (προφέρεται "έι σταρ"). Η αναζήτηση αυτή αξιολογεί τους κόμβους συνδυάζοντας το $g(n)$, το κόστος της μετάβασης στον κόμβο n, και το $h(n)$, το κόστος της μετάβασης από τον κόμβο n στον στόχο:

$$f(n) = g(n) + h(n)$$

Αφού η $g(n)$ μας δίνει το κόστος διαδρομής από τον αρχικό κόμβο στον κόμβο n, και $h(n)$ είναι το εκτιμώμενο κόστος της φθηνότερης διαδρομής από τον κόμβο n στον στόχο, έχουμε:

$f(n)$ = εκτιμώμενο κόστος της φθηνότερης λύσης μέσω του κόμβου n

Επομένως, αν επιδιώκουμε να βρούμε τη φθηνότερη λύση, είναι λογικό να δοκιμάσουμε πρώτα τον κόμβο με τη μικρότερη τιμή $g(n) + h(n)$. Όπως αποδεικνύεται, η στρατηγική αυτή είναι κάτι

παραπάνω από λογική· εφόσον η ευρετική συνάρτηση $h(n)$ ικανοποιεί ορισμένες συνθήκες, η αναζήτηση A* είναι και πλήρης και βέλτιστη.

ΠΑΡΑΔΕΚΤΟΣ ΕΥΡΕΤΙΚΟΣ ΜΗΧΑΝΙΣΜΟΣ

Η βέλτιστη συμπεριφορά της A* είναι απλό να αναλυθεί όταν χρησιμοποιείται σε αναζήτηση σε δένδρο (TREE-SEARCH). Στην περίπτωση αυτή, η A* είναι βέλτιστη αν η συνάρτηση $h(n)$ είναι **παραδεκτός ευρετικός μηχανισμός** (admissible heuristic) — δηλαδή, με την προϋπόθεση ότι η $h(n)$ *ποτέ δεν υπερεκτιμά* το κόστος επίτευξης του στόχου. Οι παραδεκτοί ευρετικοί μηχανισμοί είναι από τη φύση τους αισιόδοξοι, επειδή θεωρούν το κόστος της επίλυσης του προβλήματος μικρότερο από ότι είναι πραγματικά. Αφού $g(n)$ είναι το ακριβές κόστος της μετάβασης στον κόμβο n, μια άμεση συνέπεια είναι ότι η $f(n)$ ποτέ δεν υπερεκτιμά το πραγματικό κόστος μιας λύσης μέσω του n.

Ένα προφανές παράδειγμα παραδεκτού ευρετικού μηχανισμού είναι η ευθύγραμμη απόσταση h_{SLD} που χρησιμοποιήσαμε για τη μετάβαση στο Βουκουρέστι. Η ευθύγραμμη απόσταση είναι παραδεκτή επειδή η συντομότερη διαδρομή μεταξύ δύο οποιωνδήποτε σημείων είναι η ευθεία γραμμή, και η ευθεία γραμμή δεν μπορεί να είναι υπερεκτίμηση. Στην Εικόνα 4.3 παρουσιάζεται η πρόοδος μιας δενδρικής αναζήτησης A* για το Βουκουρέστι. Οι τιμές της g υπολογίζονται από τα κόστη βημάτων της Εικόνας 3.2, και οι τιμές της h_{SLD} δίνονται από τον πίνακα της Εικόνας 4.1. Ειδικότερα, παρατηρήστε ότι το Βουκουρέστι πρωτοεμφανίζεται στο σύνορο στο βήμα (ε), αλλά δεν επιλέγεται για να επεκταθεί επειδή το κόστος f του Βουκουρεστίου (450) είναι μεγαλύτερο από εκείνο του Pitesti (417). Ένας άλλος τρόπος για να πούμε το ίδιο πράγμα είναι ότι *θα μπορούσε* να υπάρχει λύση μέσω του Pitesti με κόστος μόνο 417, γι' αυτό ο αλγόριθμος δε θα αρκεστεί σε μια λύση που κοστίζει 450. Από αυτό το παράδειγμα μπορούμε να αποδείξουμε γενικά ότι *η αναζήτηση A* που χρησιμοποιεί τον αλγόριθμο* TREE-SEARCH *είναι βέλτιστη αν η* $h(n)$ *είναι παραδεκτή*. Έστω ότι ένας μη βέλτιστος κόμβος στόχου G_2 εμφανίζεται στο σύνορο, και έστω C^* το κόστος της βέλτιστης λύσης. Επειδή τότε ο G_2 είναι μη βέλτιστος, και επειδή $h(G_2) = 0$ (ισχύει για οποιονδήποτε κόμβο στόχου), γνωρίζουμε ότι:

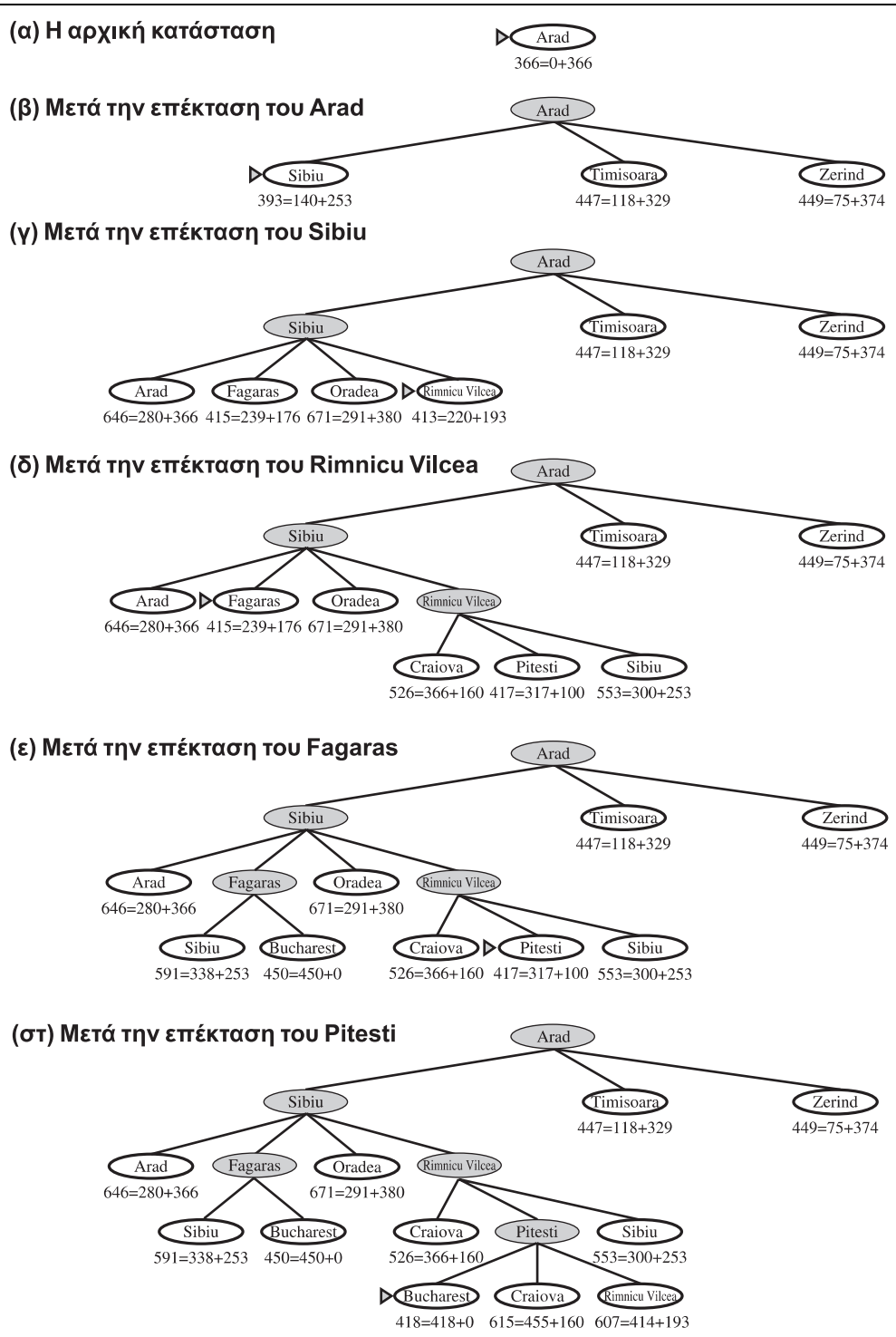
$$f(G_2) = g(G_2) + h(G_2) = g(G_2) > C^*$$

Έστω τώρα ένας κόμβος συνόρου n που βρίσκεται πάνω σε μια βέλτιστη διαδρομή λύσης — το Pitesti στο παράδειγμα της προηγούμενης παραγράφου. (Πρέπει πάντα να υπάρχει ένας τέτοιος κόμβος αν υπάρχει λύση.) Αν η $h(n)$ δεν υπερεκτιμά το κόστος ολόκληρης της διαδρομής λύσης, τότε γνωρίζουμε ότι:

$$f(n) = g(n) + h(n) \leq C^*$$

Αποδείξαμε ότι $f(n) \leq C^* < f(G_2)$ και επομένως ο κόμβος G_2 δεν επεκτείνεται και η αναζήτηση A* πρέπει να επιστρέφει μια βέλτιστη λύση.

Αν χρησιμοποιήσουμε τον αλγόριθμο αναζήτησης σε γράφημα (GRAPH-SEARCH) της Εικόνας 3.19 αντί του TREE-SEARCH, τότε η απόδειξη αυτή παύει να ισχύει. Μπορούν να επιστρέφονται μη βέλτιστες λύσεις, επειδή ο αλγόριθμος GRAPH-SEARCH μπορεί να απορρίπτει τη βέλτιστη διαδρομή προς μια επαναλαμβανόμενη κατάσταση αν δεν είναι η πρώτη που παράχθηκε (δείτε την Άσκηση 4.4). Υπάρχουν δύο τρόποι για να αντιμετωπιστεί αυτό το πρόβλημα. Η πρώτη λύση είναι να επεκταθεί ο αλγόριθμος GRAPH-SEARCH έτσι ώστε να απορρίπτει την πιο δαπανηρή από δύο οποιεσδήποτε διαδρομές βρίσκει προς τον ίδιο κόμβο (δείτε σχετικά στην Ενότητα 3.5). Η επιπλέον λογιστική εργασία είναι μπελάς, αλλά εγγυάται τη βέλτιστη συμπεριφορά. Η



Εικόνα 4.3 Στάδια μιας αναζήτησης A* για το Βουκουρέστι. Κάτω από κάθε κόμβο σημειώνονται οι τιμές $f = g + h$. Οι τιμές h είναι οι ευθύγραμμες αποστάσεις για το Βουκουρέστι και προέρχονται από τον πίνακα της Εικόνας 4.1.

δεύτερη λύση είναι να εξασφαλιστεί ότι η βέλτιστη διαδρομή προς οποιαδήποτε επαναλαμβανόμενη κατάσταση θα είναι πάντα η πρώτη που επιλέγεται — όπως συμβαίνει στην αναζήτηση ομοιόμορφου κόστους. Η ιδιότητα αυτή ισχύει αν επιβάλουμε μια πρόσθετη απαίτηση στη συνάρτηση $h(n)$, και συγκεκριμένα την απαίτηση της **συνέπειας** (consistency) ή **μονοτονικότητα** (monotonicity). Ένας ευρετικός μηχανισμός $h(n)$ είναι συνεπής αν, για κάθε κόμβο n και για κάθε διάδοχο n' του n που παράγεται από οποιαδήποτε ενέργεια a, το εκτιμώμενο κόστος της επίτευξης από τον n του στόχου δεν είναι μεγαλύτερο από το κόστος βήματος της μετάβασης στον n' συν το εκτιμώμενο κόστος της επίτευξης από τον n' του στόχου:

ΣΥΝΕΠΕΙΑ
ΜΟΝΟΤΟΝΙΚΟΤΗΤΑ

$$h(n) \leq c(n, a, n') + h(n')$$

ΤΡΙΓΩΝΙΚΗ ΑΝΙΣΟΤΗΤΑ

Αυτό είναι μια μορφή της γενικής **τριγωνικής ανισότητας**, η οποία ορίζει ότι η κάθε πλευρά ενός τριγώνου δεν μπορεί να είναι μεγαλύτερη από το άθροισμα των δύο άλλων πλευρών. Εδώ, το τρίγωνο σχηματίζεται από τους κόμβους n, n', και τον κόμβο στόχου που είναι πιο κοντά στον n. Είναι αρκετά εύκολο να αποδειχτεί (Άσκηση 4.7) ότι κάθε συνεπής (consistent) ευρετικός μηχανισμός είναι και παραδεκτός (admissible). Η σημαντικότερη επίπτωση αυτής της συνέπειας είναι η εξής: *Η αναζήτηση A* που χρησιμοποιεί τον αλγόριθμο* GRAPH-SEARCH *είναι βέλτιστη αν η $h(n)$ είναι συνεπής.*

Αν και η συνέπεια είναι αυστηρότερη απαίτηση από την παραδεκτότητα, θα πρέπει κανείς να προσπαθήσει πολύ για να επινοήσει ευρετικούς μηχανισμούς που είναι παραδεκτοί αλλά όχι συνεπείς. Όλοι οι παραδεκτοί ευρετικοί μηχανισμοί που εξετάζουμε σε αυτό το κεφάλαιο είναι και συνεπείς. Κοιτάξτε, για παράδειγμα, την h_{SLD}. Γνωρίζουμε ότι όταν κάθε πλευρά μετράται με την ευθύγραμμη απόσταση η γενική τριγωνική ανισότητα ικανοποιείται, και ότι η ευθύγραμμη απόσταση μεταξύ του n και του n' δεν είναι μεγαλύτερη από την $c(n, a, n')$. Επομένως, η h_{SLD} είναι συνεπής ευρετικός μηχανισμός.

Μια άλλη σημαντική επίπτωση της συνέπειας είναι η εξής: *Αν η $h(n)$ είναι συνεπής, τότε οι τιμές της $f(n)$ πάνω σε οποιαδήποτε διαδρομή είναι μη φθίνουσες.* Η απόδειξη προκύπτει άμεσα από τον ορισμό της συνέπειας. Έστω n' ένας διάδοχος κόμβος του n· τότε $g(n') = g(n) + c(n, a, n')$ για κάποια ενέργεια a, οπότε έχουμε:

$$f(n') = g(n') + h(n') = g(n) + c(n, a, n') + h(n') \geq g(n) + h(n) = f(n)$$

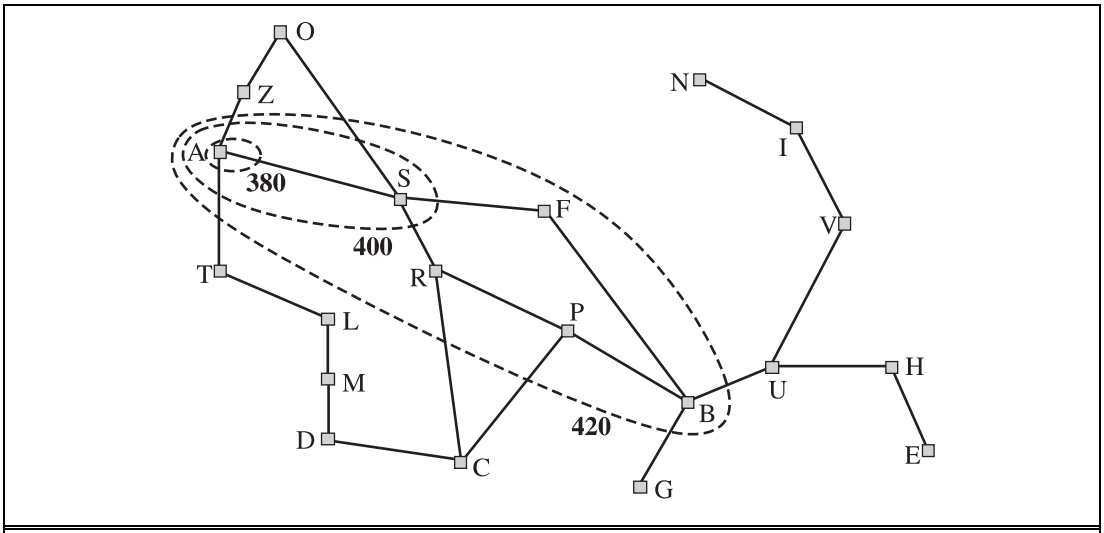
Προκύπτει ότι η ακολουθία των κόμβων που επεκτείνονται από την αναζήτηση A* που χρησιμοποιεί αλγόριθμο GRAPH-SEARCH είναι σε μη φθίνουσα διάταξη ως προς την $f(n)$. Επομένως, ο πρώτος κόμβος στόχου που επιλέγεται να επεκταθεί πρέπει να είναι βέλτιστη λύση, αφού όλοι οι επόμενοι κόμβοι θα είναι τουλάχιστον το ίδιο δαπανηροί.

ΙΣΟΫΨΕΙΣ ΚΑΜΠΥΛΕΣ

Το γεγονός ότι τα κόστη f είναι μη φθίνοντα κατά μήκος οποιασδήποτε διαδρομής σημαίνει επίσης ότι μπορούμε να σχεδιάζουμε μέσα στο χώρο καταστάσεων **ισοϋψείς καμπύλες** (contours), όπως ακριβώς οι ισοϋψείς καμπύλες σε έναν τοπογραφικό χάρτη. Ένα παράδειγμα παρουσιάζεται στην Εικόνα 4.4. Στο εσωτερικό της ισοϋψούς καμπύλης με την ετικέτα 400 όλοι οι κόμβοι έχουν τιμή $f(n)$ μικρότερη από ή ίση με 400 κ.ο.κ. Επειδή λοιπόν ο αλγόριθμος A* επεκτείνει τον κόμβο συνόρου με το μικρότερο κόστος f, μπορούμε να δούμε ότι μια αναζήτηση A* εξαπλώνεται από τον αρχικό κόμβο, προσθέτοντας κόμβους σε ομόκεντρες ζώνες με αυξανόμενο κόστος f.

Στην αναζήτηση ομοιόμορφου κόστους (αναζήτηση A* με $h(n) = 0$), οι ζώνες θα είναι κυκλικές γύρω από την αρχική κατάσταση. Με πιο ακριβείς ευρετικούς μηχανισμούς οι ζώνες θα "τεντώνονται" προς την κατάσταση στόχου και θα εστιάζονται πιο στενά γύρω από τη βέλτιστη διαδρομή. Αν C^* είναι το κόστος της βέλτιστης διαδρομή λύσης, τότε διαπιστώνουμε τα εξής:

- Η αναζήτηση A* επεκτείνει όλους τους κόμβους με $f(n) < C^*$.



Εικόνα 4.4 Χάρτης της Ρουμανίας που απεικονίζει ισοϋψείς καμπύλες με $f = 380$, $f = 400$ και $f = 420$, με αρχική κατάσταση το Arad. Οι κόμβοι στο εσωτερικό μιας δεδομένης ισοϋψούς καμπύλης έχουν κόστη f μικρότερα από ή ίσα με την τιμή της ισοϋψούς.

- Η αναζήτηση A* μπορεί έπειτα να επεκτείνει μερικούς από τους κόμβους που βρίσκονται ακριβώς πάνω στην "ισοϋψή στόχου" (όπου $f(n) = C^*$) προτού επιλέξει έναν κόμβο στόχου.

Διαισθητικά, είναι φανερό ότι η πρώτη λύση που βρίσκεται πρέπει να είναι βέλτιστη, επειδή οι κόμβοι στόχου σε όλες τις επόμενες ισοϋψείς καμπύλες θα έχουν μεγαλύτερο κόστος f, και επομένως μεγαλύτερο κόστος g (επειδή όλοι οι κόμβοι στόχου έχουν $h(n) = 0$). Εποπτικά, είναι επίσης φανερό ότι η αναζήτηση A* είναι πλήρης. Καθώς προσθέτουμε ζώνες με αυξανόμενη τιμή f, πρέπει τελικά να φτάσουμε σε μια ζώνη όπου η τιμή f είναι ίση με τα κόστος της διαδρομής προς μια κατάσταση στόχου.[4]

Προσέξτε ότι η αναζήτηση A* δεν επεκτείνει κανέναν κόμβο με $f(n) > C^*$ — για παράδειγμα, ο κόμβος της Timisoara στην Εικόνα 4.3 δεν επεκτείνεται, αν και είναι θυγατρικός της ρίζας. Στην περίπτωση αυτή, λέμε ότι το υποδένδρο κάτω από την Timisoara **κλαδεύεται** (pruned)· επειδή η h_{SLD} είναι παραδεκτή, ο αλγόριθμος μπορεί με ασφάλεια να αγνοεί αυτό το υποδένδρο ενώ εξακολουθεί να εγγυάται τη βέλτιστη συμπεριφορά. Η έννοια του κλαδέματος — της απαλοιφής περιπτώσεων χωρίς να χρειάζεται να εξεταστούν — είναι σημαντική σε πολλές περιοχές της ΤΝ.

ΚΛΑΔΕΜΑ

Μια τελευταία παρατήρηση είναι ότι μεταξύ των βέλτιστων αλγορίθμων αυτού του τύπου — των αλγορίθμων που επεκτείνουν διαδρομές αναζήτησης ξεκινώντας από τη ρίζα — ο A* είναι **βέλτιστα αποδοτικός** (optimally efficient) για οποιαδήποτε δεδομένη ευρετική συνάρτηση. Δηλαδή, κανένας άλλος βέλτιστος αλγόριθμος δεν επεκτείνει εγγυημένα λιγότερους κόμβους από τον A* (εκτός ίσως με την άρση των ισοδυναμιών μεταξύ κόμβων με $f(n) = C^*$). Αυτό συμβαίνει επειδή οποιοσδήποτε αλγόριθμος που *δεν* επεκτείνει όλους τους κόμβους με $f(n) < C^*$ κινδυνεύει να χάσει τη βέλτιστη λύση.

ΒΕΛΤΙΣΤΑ ΑΠΟΔΟΤΙΚΟΣ

[4] Η πληρότητα απαιτεί να υπάρχει πεπερασμένο πλήθος κόμβων με κόστος μικρότερο από ή ίσο με C^*, μια συνθήκη που αληθεύει αν όλα τα κόστη βημάτων υπερβαίνουν κάποιο πεπερασμένο ϵ και αν το b είναι πεπερασμένο.

Το ότι η αναζήτηση A* είναι πλήρης, βέλτιστη και βέλτιστα αποδοτική μεταξύ όλων των αλγορίθμων αυτού του είδους είναι μάλλον ευχάριστο. Δυστυχώς, αυτό δε σημαίνει ότι η A* είναι η απάντηση για όλες μας τις ανάγκες αναζήτησης. Το πρόβλημα είναι ότι, στα περισσότερα προβλήματα, ο αριθμός των κόμβων μέσα στο χώρο αναζήτησης της ισοϋψούς καμπύλης στόχου εξακολουθεί να αυξάνεται εκθετικά με το μήκος της λύσης. Αν και η απόδειξη ξεφεύγει από τα όρια αυτού του βιβλίου, έχει αποδειχτεί ότι η αύξηση θα είναι εκθετική εκτός αν το σφάλμα της ευρετικής συνάρτησης δεν αυξάνεται γρηγορότερα από το λογάριθμο του πραγματικού κόστους διαδρομής. Σε μαθηματική σημειογραφία, η συνθήκη για αύξηση χαμηλότερη της εκθετικής είναι

$$| h(n) - h^*(n) | \leq \mathrm{O}(\log h^*(n))$$

όπου $h^*(n)$ είναι το *πραγματικό* κόστος της μετάβασης από τον κόμβο n στον στόχο. Για όλους σχεδόν τους ευρετικούς μηχανισμούς που χρησιμοποιούνται στην πράξη, το σφάλμα είναι τουλάχιστον ανάλογο με το κόστος διαδρομής και η εκθετική αύξηση που προκύπτει τελικά ξεπερνά τις δυνατότητες οποιουδήποτε υπολογιστή. Γι' αυτόν το λόγο, συχνά είναι άσκοπο να επιμένουμε να βρούμε μια βέλτιστη λύση. Μπορεί κανείς να χρησιμοποιεί παραλλαγές του A* που βρίσκουν γρήγορα μη βέλτιστες λύσεις, ή μπορεί κανείς μερικές φορές να σχεδιάζει ευρετικούς μηχανισμούς που είναι πιο ακριβείς αλλά όχι αυστηρά παραδεκτοί. Σε οποιαδήποτε περίπτωση, η χρήση ενός καλού ευρετικού μηχανισμού εξακολουθεί να παρέχει τεράστια οικονομία σε σύγκριση με τη χρήση απληροφόρητης αναζήτησης. Θα εξετάσουμε το ζήτημα της σχεδίαση καλών ευρετικών μηχανισμών στην Ενότητα 4.2.

Ο χρόνος υπολογισμού δεν είναι όμως το κύριο μειονέκτημα της αναζήτησης A*. Επειδή διατηρεί στη μνήμη όλους τους κόμβους που παράγονται (όπως κάνουν όλοι οι αλγόριθμοι GRAPH-SEARCH), ο αλγόριθμος A* συνήθως εξαντλεί το χώρο πολύ πριν εξαντλήσει το χρόνο. Γι' αυτόν το λόγο, ο A* δεν είναι πρακτικά χρήσιμος για πολλά προβλήματα μεγάλης κλίμακας. Πρόσφατα έχουν επινοηθεί αλγόριθμοι που ξεπερνούν το πρόβλημα του χώρου χωρίς να θυσιάζουν τη βέλτιστη συμπεριφορά ή την πληρότητα, με μικρό κόστος σε χρόνο εκτέλεσης. Αυτοί εξετάζονται στη συνέχεια.

Ευρετική αναζήτηση περιορισμένης μνήμης

Ο απλούστερος τρόπος για να μειωθούν οι απαιτήσεις μνήμης της αναζήτησης A* είναι να υιοθετηθεί η ιδέα της επαναληπτικής εκβάθυνσης στα πλαίσια της ευρετικής αναζήτησης, ώστε να προκύψει ένας αλγόριθμος A* με επαναληπτική εκβάθυνση (iterative-deepening A*, IDA*). Η κύρια διαφορά μεταξύ της αναζήτησης IDA* και της συνηθισμένης επαναληπτικής εκβάθυνσης είναι ότι το όριο αποκοπής που χρησιμοποιείται είναι το κόστος $f = (g + h)$ και όχι το βάθος· σε κάθε επανάληψη, η τιμή αποκοπής είναι το μικρότερο κόστος f οποιουδήποτε κόμβου ξεπέρασε το όριο αποκοπής στην προηγούμενη επανάληψη. Η αναζήτηση IDA* είναι πρακτικά χρήσιμη για πολλά προβλήματα με μοναδιαία κόστη βημάτων και αποφεύγει τη σημαντική επιβάρυνση που σχετίζεται με την τήρηση μιας ταξινομημένης ουράς κόμβων. Δυστυχώς, αντιμετωπίζει τις ίδιες δυσκολίες με τις πραγματικές τιμές κόστους όπως η επαναληπτική έκδοση της αναζήτησης ομοιόμορφου κόστους που περιγράψαμε στην Άσκηση 3.11. Σε αυτή την ενότητα εξετάζουμε συνοπτικά δύο πιο πρόσφατους αλγόριθμους περιορισμένης μνήμης (bounded memory), που ονομάζονται RBFS και MA*.

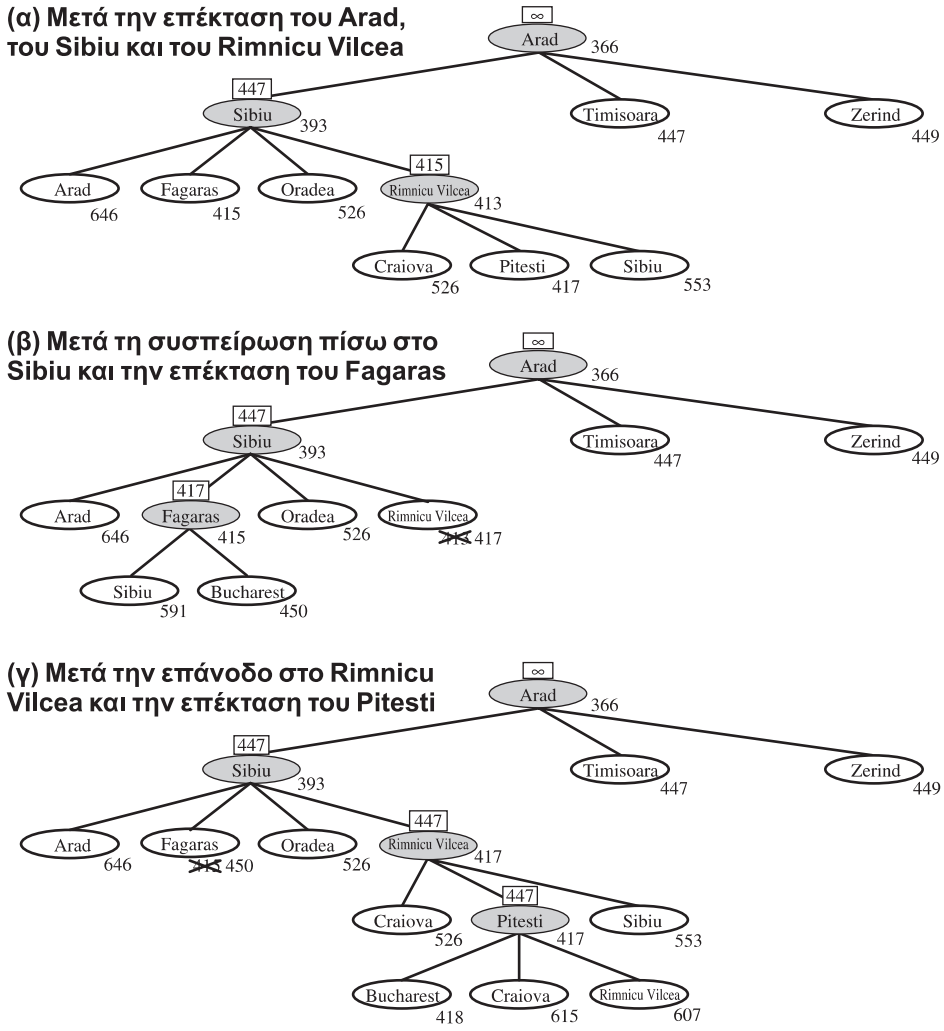
```
function RECURSIVE-BEST-FIRST-SEARCH( πρόβλημα ) returns μια λύση ή αποτυχία
    RBFS( πρόβλημα, MAKE-NODE( INITIAL-STATE[πρόβλημα]), ∞ )

function RBFS( πρόβλημα, κόμβος, όριο_f ) returns λύση, ή αποτυχία και νέο όριο κόστους f
    if GOAL-TEST[πρόβλημα]( STATE[κόμβος]) then return κόμβος
    διάδοχοι ← EXPAND( κόμβος, πρόβλημα )
    if διάδοχοι είναι κενό then return αποτυχία, ∞
    for each s in διάδοχοι do
        f[s] ← max( g(s) + h(s), f[κόμβος])
    repeat
        καλύτερος ← κόμβος με τη μικρότερη τιμή f στο διάδοχοι
        if f[καλύτερος] > όριο_f then return αποτυχία, f[καλύτερος]
        εναλλακτικός ← κόμβος με τη δεύτερη μικρότερη τιμή f στο διάδοχοι
        αποτέλεσμα, f[καλύτερος] ← RBFS( πρόβλημα, καλύτερος, min( όριο_f, εναλλακτικός ))
        if αποτέλεσμα ≠ αποτυχία then return αποτέλεσμα
```

Εικόνα 4.5 Ο αλγόριθμος της αναδρομικής αναζήτησης πρώτα στο καλύτερο.

ΑΝΑΔΡΟΜΙΚΗ ΑΝΑΖΗΤΗΣΗ ΠΡΩΤΑ ΣΤΟ ΚΑΛΥΤΕΡΟ

Η **αναδρομική αναζήτηση πρώτα στο καλύτερο** (recursive best-first search, RBFS) είναι ένας απλός αναδρομικός αλγόριθμος που προσπαθεί να μιμηθεί τον τρόπο λειτουργίας της συνηθισμένης αναζήτησης πρώτα στο καλύτερο, χρησιμοποιώντας όμως μόνο γραμμικό χώρο. Ο αλγόριθμος παρουσιάζεται στην Εικόνα 4.5. Η δομή του είναι παρόμοια με της αναδρομικής αναζήτησης πρώτα κατά βάθος, αλλά αντί να συνεχίζει επ' άπειρον την κάθοδο στην τρέχουσα διαδρομή παρακολουθεί την τιμή *f* της καλύτερης εναλλακτικής διαδρομής που είναι διαθέσιμη από οποιονδήποτε πρόγονο του τρέχοντος κόμβου. Αν ο τρέχων κόμβος ξεπεράσει το όριο, η αναδρομή συσπειρώνεται πίσω στην εναλλακτική διαδρομή. Καθώς συσπειρώνεται η αναδρομή, ο αλγόριθμος RBFS αντικαθιστά την τιμή *f* κάθε κόμβου της διαδρομής με την καλύτερη τιμή *f* των θυγατρικών του. Με αυτόν τον τρόπο, η αναζήτηση RBFS θυμάται την τιμή *f* του καλύτερου φύλλου του ξεχασμένου υποδένδρου και έτσι μπορεί να αποφασίζει αν αξίζει να αναπτύξει πάλι το υποδένδρο σε κάποια επόμενη φάση. Η Εικόνα 4.6 δείχνει πώς η αναζήτηση RBFS φτάνει στο Βουκουρέστι.

Η αναδρομική αναζήτηση πρώτα στο καλύτερο (RBFS) είναι κάπως πιο αποδοτική από την αναζήτηση A* με επαναληπτική εκβάθυνση (IDA*), αλλά και αυτή πάσχει από υπερβολική επαναπαραγωγή κόμβων. Στο παράδειγμα της Εικόνας 4.6, ο αλγόριθμος RBFS ακολουθεί πρώτα τη διαδρομή μέσω Rimnicu Vilcea, έπειτα "αλλάζει γνώμη" και δοκιμάζει το Fagaras, και έπειτα επανέρχεται πάλι στην αρχική διαδρομή. Αυτές οι αλλαγές γνώμης συμβαίνουν επειδή κάθε φορά που επεκτείνεται η τρέχουσα καλύτερη διαδρομή υπάρχει σημαντική πιθανότητα να αυξηθεί η τιμή *f* — η *h* είναι συνήθως λιγότερο αισιόδοξη για τους κόμβους που βρίσκονται πιο κοντά στον στόχο. Όταν συμβαίνει αυτό, ιδιαίτερα σε μεγάλους χώρους αναζήτησης, η δεύτερη καλύτερη διαδρομή μπορεί να γίνει η καλύτερη, οπότε η αναζήτηση πρέπει να υπαναχωρήσει για να την ακολουθήσει. Κάθε αλλαγή γνώμης αντιστοιχεί σε μια επανάληψη της αναζήτησης IDA*, και μπορεί να χρειαστούν πολλές επανεπεκτάσεις ξεχασμένων κόμβων για να ξαναδημιουργηθεί η καλύτερη διαδρομή και να επεκταθεί κατά έναν κόμβο ακόμα.



Εικόνα 4.6 Στάδια μιας αναζήτησης RBFS για το συντομότερο δρομολόγιο προς το Βουκουρέστι. Η τιμή του ορίου *f* για κάθε αναδρομική κλήση εμφανίζεται πάνω από κάθε τρέχοντα κόμβο. (α) Ακολουθείται η διαδρομή μέσω του Rimnicu Vilcea μέχρι το σημείο που το τρέχον καλύτερο φύλλο (το Pitesti) έχει χειρότερη τιμή από την καλύτερη εναλλακτική διαδρομή (το Fagaras). (β) Η αναδρομή συσπειρώνεται και η καλύτερη τιμή φύλλου του ξεχασμένου υποδένδρου (417) αντιγράφεται στο Rimnicu Vilcea· έπειτα επεκτείνεται το Fagaras, αποκαλύπτοντας μια καλύτερη τιμή φύλλου 450. (γ) Η αναδρομή συσπειρώνεται και η καλύτερη τιμή φύλλου του ξεχασμένου υποδένδρου (450) αντιγράφεται στο Fagaras· έπειτα επεκτείνεται το Rimnicu Vilcea. Αυτή τη φορά, επειδή η καλύτερη εναλλακτική διαδρομή (μέσω της Timisoara) κοστίζει τουλάχιστον 447, συνεχίζεται η επέκταση μέχρι το Βουκουρέστι.

Όπως και ο αλγόριθμος A*, ο RBFS είναι βέλτιστος αν η ευρετική συνάρτηση $h(n)$ είναι παραδεκτή. Η χωρική του πολυπλοκότητα είναι $O(bd)$, αλλά η χρονική του πολυπλοκότητα είναι μάλλον δύσκολο να προσδιοριστεί· εξαρτάται και από την ακρίβεια της ευρετικής συνάρτησης και από το πόσο συχνά αλλάζει η καλύτερη διαδρομή καθώς επεκτείνονται οι κόμβοι. Τόσο η αναζήτηση IDA* όσο και η RBFS υπόκεινται σε ενδεχόμενη εκθετική αύξηση της πολυπλοκότητας σε σχέση με την αναζήτηση σε γραφήματα (δείτε στην Ενότητα 3.5), επειδή δεν μπορούν να

ελέγχουν για επαναλαμβανόμενες καταστάσεις εκτός από εκείνες που βρίσκονται πάνω στην τρέχουσα διαδρομή. Έτσι, μπορεί να εξερευνούν την ίδια κατάσταση πολλές φορές.

Οι αλγόριθμοι IDA* και RBFS πάσχουν από τη χρήση *πολύ λίγης* μνήμης. Μεταξύ των επαναλήψεων, ο αλγόριθμος IDA* διατηρεί μόνο έναν αριθμό· το τρέχον όριο του κόστους *f*. Ο RBFS διατηρεί περισσότερες πληροφορίες στη μνήμη, αλλά η μνήμη που χρησιμοποιεί είναι μόνο $O(bd)$· ακόμα και αν υπάρχει περισσότερη διαθέσιμη μνήμη, ο αλγόριθμος RBFS δεν έχει τρόπο να την αξιοποιήσει.

MA*
SMA*

Φαίνεται λοιπόν λογικό να χρησιμοποιείται όλη η διαθέσιμη μνήμη. Δύο αλγόριθμοι που το κάνουν αυτό είναι ο **MA*** (memory-bounded A* — A* περιορισμένης μνήμης) και ο **SMA*** (simplified MA* — απλουστευμένος MA*). Θα περιγράψουμε τον αλγόριθμο SMA* επειδή είναι απλούστερος. Ο SMA* προχωρεί ακριβώς όπως ο A*, επεκτείνοντας το καλύτερο φύλλο μέχρι να γεμίσει η μνήμη. Στο σημείο αυτό, δεν μπορεί να προσθέσει ένα νέο κόμβο στο δένδρο αναζήτησης χωρίς να καταργήσει έναν παλιό. Ο SMA* καταργεί πάντα το *χειρότερο* κόμβο-φύλλο — εκείνον με τη μεγαλύτερη τιμή *f*. Όπως και ο RBFS, ο αλγόριθμος SMA* αντιγράφει την τιμή του ξεχασμένου κόμβου στο μητρικό του κόμβο. Με αυτόν τον τρόπο, ο πρόγονος ενός ξεχασμένου υποδένδρου γνωρίζει την ποιότητα της καλύτερης διαδρομής αυτού του υποδένδρου. Με αυτή την πληροφορία, ο SMA* επαναπαράγει το υποδένδρο μόνο όταν διαπιστωθεί ότι *όλες οι άλλες διαδρομές* φαίνονται χειρότερες από την ξεχασμένη διαδρομή. Ένας άλλος τρόπος για να πούμε το ίδιο πράγμα είναι ότι, αν όλοι οι απόγονοι ενός κόμβου *n* έχουν ξεχαστεί, τότε δε θα γνωρίζουμε προς τα πού πρέπει να πάμε από τον *n*, αλλά θα εξακολουθούμε να έχουμε μια ιδέα για το κατά πόσον αξίζει να πάμε οπουδήποτε από τον *n*.

Ο πλήρης αλγόριθμος είναι πολύ περίπλοκος για να τον παρουσιάσουμε εδώ,[5] αλλά υπάρχει ένα λεπτό σημείο που αξίζει να αναφέρουμε. Είπαμε ότι ο SMA* επεκτείνει το καλύτερο φύλλο και διαγράφει το χειρότερο. Τι γίνεται αν *όλοι* οι κόμβοι-φύλλα έχουν την ίδια τιμή *f*; Τότε ο αλγόριθμος θα μπορούσε να επιλέξει τον ίδιο κόμβο για διαγραφή και για επέκταση. Ο SMA* λύνει αυτό το πρόβλημα επεκτείνοντας τον *νεότερο* καλύτερο φύλλο και διαγράφοντας το *παλαιότερο* χειρότερο φύλλο. Αυτά τα δύο μπορεί να είναι ο ίδιος κόμβος μόνο αν υπάρχει μόνο ένα φύλλο· στην περίπτωση αυτή, το τρέχον δένδρο αναζήτησης θα πρέπει να είναι μια μεμονωμένη διαδρομή από τη ρίζα μέχρι το φύλλο η οποία καταλαμβάνει όλη τη μνήμη. Αν το φύλλο δεν είναι κόμβος στόχου τότε, *ακόμα και αν βρίσκεται πάνω σε μια βέλτιστη διαδρομή λύσης*, η λύση αυτή δεν είναι προσπελάσιμη με τη διαθέσιμη μνήμη. Γι' αυτό, ο κόμβος μπορεί να απορριφθεί ακριβώς σαν να μην είχε διαδόχους.

Η αναζήτηση SMA* είναι πλήρης αν υπάρχει οποιαδήποτε προσπελάσιμη λύση — δηλαδή, αν το *d*, το βάθος του ρηχότερου κόμβου στόχου, είναι μικρότερο από τη διαθέσιμη μνήμη (εκφρασμένη σε κόμβους). Είναι βέλτιστη αν οποιαδήποτε βέλτιστη λύση είναι προσπελάσιμη· αλλιώς, επιστρέφει την καλύτερη προσπελάσιμη λύση. Πρακτικά, η αναζήτηση SMA* μπορεί να θεωρηθεί ο καλύτερος αλγόριθμος γενικής χρήσης για την εύρεση βέλτιστων λύσεων, ιδιαίτερα όταν ο χώρος καταστάσεων είναι γράφημα, τα κόστη βημάτων δεν είναι ομοιόμορφα, και η παραγωγή κόμβων είναι δαπανηρή σε σύγκριση με την πρόσθετη επιβάρυνση της συντήρησης των ανοιχτών και κλειστών λιστών.

Στα πολύ δύσκολα προβλήματα όμως, θα παρουσιάζεται συχνά η περίπτωση να αναγκάζεται η αναζήτηση SMA* να πηγαινοέρχεται συνεχώς μεταξύ ενός συνόλου διαδρομών υποψηφίων λύσεων, από τις οποίες μόνο ένα μικρό υποσύνολο χωρά στη μνήμη. (Αυτό μοιάζει με το πρόβλημα του **"αλωνίσματος"** — **thrashing** — στα συστήματα σελιδοποίησης των δίσκων.) Στην περίπτωση αυτή, ο επιπλέον χρόνος που απαιτείται για τη συνεχή επαναπαραγωγή των ίδιων κόμβων σημαίνει ότι κάποια προβλήματα που θα ήταν πρακτικά επιλύσιμα με την αναζήτηση A*, με δεδομένη απεριόριστη μνήμη, γίνονται δυσεπίλυτα (intractable) για την αναζήτηση

ΑΛΩΝΙΣΜΑ

[5] Μια χονδρική σκιαγράφηση του SMA* υπήρχε στην πρώτη έκδοση του βιβλίου.

SMA*. Με άλλα λόγια, *οι περιορισμοί μνήμης μπορούν να κάνουν ένα πρόβλημα δυσεπίλυτο από την άποψη του χρόνου υπολογισμού*. Αν και δεν υπάρχει θεωρία που να εξηγεί τη σχέση εξισορρόπησης μεταξύ χρόνου και μνήμης, φαίνεται ότι το πρόβλημα είναι αναπόφευκτο. Η μόνη διέξοδος είναι να παραιτηθούμε από την απαίτηση της βέλτιστης συμπεριφοράς.

Εκμάθηση καλύτερων τρόπων αναζήτησης

Παρουσιάσαμε πολλές σταθερές στρατηγικές αναζήτησης — πρώτα κατά πλάτος, άπληστη πρώτα στο καλύτερο κ.ο.κ. — οι οποίες έχουν σχεδιαστεί από επιστήμονες των υπολογιστών. Θα μπορούσε ένας πράκτορας να *μαθαίνει* πώς να κάνει καλύτερες αναζητήσεις; Η απάντηση είναι ναι, και η μέθοδος βασίζεται σε μια σημαντική έννοια που ονομάζεται **χώρος καταστάσεων του μεταεπιπέδου** (metalevel state space). Κάθε κατάσταση σε ένα χώρο καταστάσεων του μεταεπιπέδου συλλαμβάνει την εσωτερική (υπολογιστική) κατάσταση ενός προγράμματος το οποίο κάνει μια αναζήτηση σε ένα **χώρο καταστάσεων του επιπέδου αντικειμένων** (object-level state space) όπως η Ρουμανία. Για παράδειγμα, η εσωτερική κατάσταση του αλγόριθμου A* αποτελείται από το τρέχον δένδρο αναζήτησης. Κάθε ενέργεια στο χώρο καταστάσεων του μεταεπιπέδου είναι ένα υπολογιστικό βήμα που μεταβάλλει την εσωτερική κατάσταση· για παράδειγμα, κάθε υπολογιστικό βήμα της αναζήτησης A* επεκτείνει έναν κόμβο-φύλλο και προσθέτει τους διαδόχους του στο δένδρο. Έτσι, η Εικόνα 4.3 που παρουσιάζει μια ακολουθία όλο και μεγαλύτερων δένδρων αναζήτησης μπορεί να θεωρηθεί ότι απεικονίζει μια διαδρομή στο χώρο καταστάσεων του μεταεπιπέδου, όπου κάθε κατάσταση πάνω στη διαδρομή είναι ένα δένδρο αναζήτησης του επιπέδου αντικειμένων.

ΧΩΡΟΣ ΚΑΤΑΣΤΑΣΕΩΝ ΤΟΥ ΜΕΤΑΕΠΙΠΕΔΟΥ

ΧΩΡΟΣ ΚΑΤΑΣΤΑΣΕΩΝ ΤΟΥ ΕΠΙΠΕΔΟΥ ΑΝΤΙΚΕΙΜΕΝΩΝ

Τώρα, η διαδρομή της Εικόνας 4.3 έχει πέντε βήματα, μεταξύ των οποίων ένα βήμα — την επέκταση του κόμβου του Fagaras — που δε βοηθά ιδιαίτερα. Σε δυσκολότερα προβλήματα θα υπάρχουν πολλά τέτοια παραπατήματα, και ένας αλγόριθμος **μάθησης στο μεταεπίπεδο** (metalevel learning) μπορεί να μαθαίνει από αυτές τις εμπειρίες να μην εξερευνά τα μη ευνοϊκά υποδένδρα. Οι τεχνικές που χρησιμοποιούνται σε αυτό το είδος μάθησης περιγράφονται στο Κεφάλαιο 21. Στόχος της μάθησης είναι η ελαχιστοποίηση του ολικού κόστους της επίλυσης προβλημάτων, με ανταλλαγή μεταξύ υπολογιστικής δαπάνης και κόστους διαδρομής.

ΜΑΘΗΣΗ ΣΤΟ ΜΕΤΑΕΠΙΠΕΔΟ

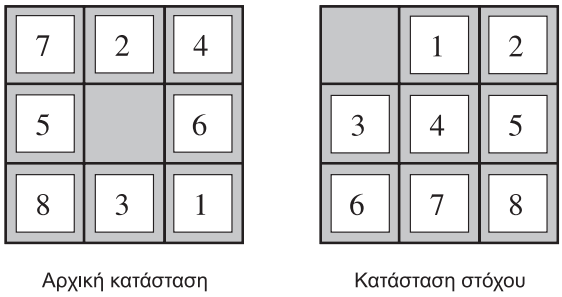
4.2 ΕΥΡΕΤΙΚΕΣ ΣΥΝΑΡΤΗΣΕΙΣ

Σε αυτή την ενότητα θα εξετάσουμε ευρετικούς μηχανισμούς για το παζλ των 8 πλακιδίων, με σκοπό να ρίξουμε περισσότερο φως στη φύση των ευρετικών μηχανισμών γενικά.

Το παζλ των 8 πλακιδίων ήταν ένα από τα πρώτα προβλήματα ευρετικής αναζήτησης. Όπως είδαμε στην Ενότητα 3.2, το αντικείμενο του παζλ είναι να σύρονται τα πλακίδια οριζόντια ή κάθετα στην κενή θέση μέχρι να συμπέσει η διάταξή τους με την επιθυμητή διάταξη στόχου (Εικόνα 4.7).

Το μέσο κόστος λύσης για ένα στιγμιότυπο του παζλ των 8 πλακιδίων που έχει παραχθεί τυχαία είναι περίπου 22 βήματα. Ο παράγοντας διακλάδωσης είναι περίπου 3. (Όταν η κενή θέση πλακιδίου είναι στο κέντρο υπάρχουν τέσσερις δυνατές κινήσεις· όταν είναι σε γωνία υπάρχουν δύο· όταν είναι πάνω σε μια πλευρά υπάρχουν τρεις.) Αυτό σημαίνει ότι μια εξαντλητική αναζήτηση σε βάθος 22 θα εξέταζε περίπου $3^{22} \approx 3{,}1 \times 10^{10}$ καταστάσεις. Παρακολουθώντας τις επαναλαμβανόμενες καταστάσεις, θα μπορούσαμε να μειώσουμε αυτόν τον αριθμό κατά έναν παράγοντα περίπου 170.000, επειδή υπάρχουν μόνο 9! / 2 = 181.440 διακριτές καταστάσεις που είναι προσπελάσιμες (δείτε την Άσκηση 3.4). Αυτός ο αριθμός είναι μέσα στις δυνατότητες, αλλά ο αντίστοιχος αριθμός για το παζλ των 15 πλακιδίων είναι περίπου 10^{13}, γι' αυτό η επόμενή μας δουλειά είναι να βρούμε μια καλή ευρετική συνάρτηση. Αν θέλουμε να βρούμε τις συντομότερες λύσεις χρησιμοποιώντας αναζήτηση A*, χρειαζόμαστε μια ευρετική συνάρτηση που να μην υπε-

ρεκτιμά ποτέ τον αριθμό των βημάτων για τον στόχο. Υπάρχει μακρόχρονη ιστορία τέτοιων ευρετικών μηχανισμών για το παζλ των 15 πλακιδίων· ας δούμε δύο υποψήφιους μηχανισμούς που χρησιμοποιούνται συχνά:



Εικόνα 4.7 Αντιπροσωπευτική περίπτωση του παζλ των 8 πλακιδίων. Η λύση έχει μήκος 26 βήματα.

- h_1 = ο αριθμός των πλακιδίων που δεν είναι στη θέση τους. Στην Εικόνα 4.7, και τα οκτώ πλακίδια είναι σε λάθος θέση, και επομένως η αρχική κατάσταση θα είχε $h_1 = 8$. Η h_1 είναι παραδεκτός ευρετικός μηχανισμός, επειδή είναι σαφές ότι οποιοδήποτε πλακίδιο που δεν είναι στη θέση του θα πρέπει να μετακινηθεί τουλάχιστον μία φορά.
- h_2 = το άθροισμα των αποστάσεων των πλακιδίων από τις θέσεις προορισμού τους. Επειδή τα πλακίδια δεν μπορούν να μετακινούνται διαγώνια, η απόσταση που θα μετράμε είναι το άθροισμα των οριζόντιων και κάθετων αποστάσεων. Αυτό λέγεται μερικές φορές **απόσταση οικοδομικών τετραγώνων** (city block distance) ή **απόσταση Manhattan**. Η h_2 είναι επίσης παραδεκτή, επειδή το καλύτερο που μπορεί να κάνει οποιαδήποτε κίνηση είναι να μετακινήσει ένα πλακίδιο κατά ένα βήμα πιο κοντά στον στόχο. Η απόσταση Manhattan των πλακιδίων 1 έως 8 στην αρχική κατάσταση είναι:

ΑΠΟΣΤΑΣΗ ΜΑΝΗΑΤΤΑΝ

$$h_2 = 3 + 1 + 2 + 2 + 2 + 3 + 3 + 2 = 18$$

Όπως ελπίζαμε, κανένας από τους δύο ευρετικούς μηχανισμούς δεν υπερεκτιμά το πραγματικό κόστος λύσης, το οποίο είναι 26.

Επίδραση της ευρετικής ακρίβειας στην απόδοση

ΔΡΑΣΤΙΚΟΣ ΠΑΡΑΓΟΝΤΑΣ ΔΙΑΚΛΑΔΩΣΗΣ

Ένας τρόπος προσδιορισμού της ποιότητας ενός ευρετικού μηχανισμού είναι ο **δραστικός παράγοντας διακλάδωσης** b^*. Αν ο συνολικός αριθμός των κόμβων που παράγονται από την αναζήτηση A* για ένα συγκεκριμένο πρόβλημα είναι N και το βάθος της λύσης είναι d, τότε b^* είναι ο παράγοντας διακλάδωσης που θα έπρεπε να έχει ένα ομοιόμορφο δένδρο με βάθος d για να περιέχει $N + 1$ κόμβους. Συγκεκριμένα:

$$N + 1 = 1 + b^* + (b^*)^2 + \cdots + (b^*)^d$$

Για παράδειγμα, αν η αναζήτηση A* βρει μια λύση σε βάθος 5 χρησιμοποιώντας 52 κόμβους, τότε ο δραστικός παράγοντας διακλάδωσης είναι 1,92. Ο δραστικός παράγοντας διακλάδωσης μπορεί να ποικίλλει στα διάφορα στιγμιότυπα ενός προβλήματος, αλλά συνήθως είναι σχετικά σταθερός για τα αρκετά δύσκολα προβλήματα. Έτσι, πειραματικές μετρήσεις του b^* σε ένα μικρό σύνολο προβλημάτων μπορούν να παρέχουν έναν καλό οδηγό για τη συνολική χρησιμότητα του ευρετικού μηχανισμού. Ένας καλά σχεδιασμένος ευρετικός μηχανισμός θα είχε τιμή του b^* κοντά στο 1, επιτρέποντας την επίλυση αρκετά μεγάλων προβλημάτων.

Για να δοκιμάσουμε τις ευρετικές συναρτήσεις h_1 και h_2, δημιουργήσαμε 1200 τυχαία προβλήματα με μήκος λύσης από 2 μέχρι 24 (100 προβλήματα για κάθε ζυγό αριθμό) και τα λύσαμε με αναζήτηση επαναληπτικής εκβάθυνσης και με αναζήτηση A* σε δένδρο χρησιμοποιώντας και την h_1 και την h_2. Ο πίνακας της Εικόνας 4.8 δίνει το μέσο αριθμό κόμβων που επεκτάθηκαν με την κάθε στρατηγική και το δραστικό παράγοντα διακλάδωσης. Τα αποτελέσματα δείχνουν ότι η h_2 είναι καλύτερη από την h_1, και είναι πολύ καλύτερη από τη χρήση αναζήτησης επαναληπτικής εκβάθυνσης. Στις λύσεις μας με μήκος 14, η αναζήτηση A* με την h_2 είναι 30.000 φορές αποδοτικότερη από την απληροφόρητη αναζήτηση επαναληπτικής εκβάθυνσης.

	Κόστος αναζήτησης			Δραστικός παράγοντας διακλάδωσης		
d	IDS	A*(h_1)	A*(h_2)	IDS	A*(h_1)	A*(h_2)
2	10	6	6	2,45	1,79	1,79
4	112	13	12	2,87	1,48	1,45
6	680	20	18	2,73	1,34	1,30
8	6384	39	25	2,80	1,33	1,24
10	47127	93	39	2,79	1,38	1,22
12	3644035	227	73	2,78	1,42	1,24
14	—	539	113	—	1,44	1,23
16	—	1301	211	—	1,45	1,25
18	—	3056	363	—	1,46	1,26
20	—	7276	676	—	1,47	1,27
22	—	18094	1219	—	1,48	1,28
24	—	39135	1641	—	1,48	1,26

Εικόνα 4.8 Σύγκριση των κοστών αναζήτησης και των δραστικών παραγόντων διακλάδωσης για τους αλγόριθμους αναζήτησης επαναληπτικής εκβάθυνσης και A* με τους ευρετικούς μηχανισμούς h_1 και h_2. Τα δεδομένα είναι μέσοι όροι για 100 στιγμιότυπα του παζλ των 8 πλακιδίων, για διαφορετικά μήκη λύσης.

Θα μπορούσε να ρωτήσει κανείς αν η h_2 είναι *πάντα* καλύτερη από την h_1. Η απάντηση είναι ναι. Από τον ορισμό των δύο ευρετικών μηχανισμών μπορούμε εύκολα να δούμε ότι, για οποιονδήποτε κόμβο n, είναι $h_2(n) \geq h_1(n)$. Λέμε λοιπόν ότι η h_2 **κυριαρχεί** (dominates) έναντι της h_1. Η κυριαρχία μεταφράζεται άμεσα σε αποδοτικότητα: Η αναζήτηση A* που χρησιμοποιεί την h_2 ποτέ δε θα επεκτείνει περισσότερους κόμβους από την A* που χρησιμοποιεί την h_1 (εκτός ίσως μερικούς κόμβους με $f(n) = C^*$). Ο λόγος είναι απλός. Θυμηθείτε την παρατήρηση στην Ενότητα 4.1, στην υποενότητα για την αναζήτηση A*, ότι κάθε κόμβος με $f(n) < C^*$ θα επεκταθεί σίγουρα. Είναι σαν να λέμε ότι κάθε κόμβος με $h(n) < C^* - g(n)$ θα επεκταθεί σίγουρα. Επειδή όμως η τιμή h_2 είναι τουλάχιστον όσο μεγάλη είναι και η h_1 για όλους τους κόμβους, κάθε κόμβος που επεκτείνεται σίγουρα από την αναζήτηση A* με την h_2 θα επεκταθεί σίγουρα και με την h_1, και η h_1 θα μπορούσε επίσης να προκαλέσει την επέκταση και άλλων κόμβων. Επομένως, είναι πάντα καλύτερο να χρησιμοποιούμε μια ευρετική συνάρτηση με μεγαλύτερες τιμές, με την προϋπόθεση ότι δεν υπερεκτιμά και ότι ο χρόνος υπολογισμού για το συγκεκριμένο ευρετικό μηχανισμό δεν είναι υπερβολικά μεγάλος.

ΚΥΡΙΑΡΧΙΑ

Επινόηση παραδεκτών ευρετικών συναρτήσεων

Είδαμε ότι και οι δύο συναρτήσεις, h_1 (πλακίδια που δεν είναι στη θέση τους) και h_2 (απόσταση Manhattan), είναι αρκετά καλοί ευρετικοί μηχανισμοί για το παζλ των 8 πλακιδίων, και ότι η h_2 είναι καλύτερη. Πώς μπορεί κανείς να επινοήσει την h_2; Είναι δυνατό να επινοήσει ένας υπολογιστής έναν τέτοιο ευρετικό μηχανισμό μηχανικά;

Οι h_1 και h_2 είναι εκτιμήσεις του μήκους διαδρομής που απομένει για το παζλ των 8 πλακιδίων, αλλά είναι επίσης εντελώς ακριβή μήκη διαδρομής για *απλουστευμένες* εκδόσεις του παζλ. Αν οι κανόνες του παζλ άλλαζαν έτσι ώστε ένα πλακίδιο να μπορεί να μετακινείται οπου-

δήποτε, και όχι μόνο σε ένα γειτονικό κενό τετράγωνο, τότε η h_1 θα έδινε τον ακριβή αριθμό βημάτων της συντομότερης λύσης. Παρόμοια, αν ένα πλακίδιο μπορούσε να μετακινείται κατά ένα τετράγωνο προς οποιαδήποτε κατεύθυνση, ακόμα και αν είναι κατειλημμένο, τότε η h_2 θα έδινε τον ακριβή αριθμό βημάτων της συντομότερης λύσης. Ένα πρόβλημα με λιγότερους περιορισμούς για τις ενέργειες ονομάζεται **χαλαρό πρόβλημα** (relaxed problem). *Το κόστος μιας βέλτιστης λύσης για ένα χαλαρό πρόβλημα είναι ένας παραδεκτός ευρετικός μηχανισμός για το αρχικό πρόβλημα.* Ο ευρετικός μηχανισμός είναι παραδεκτός (admissible) επειδή η βέλτιστη λύση του αρχικού προβλήματος είναι εξ ορισμού και λύση του χαλαρού προβλήματος και επομένως πρέπει να είναι τουλάχιστον τόσο δαπανηρή όσο η βέλτιστη λύση του χαλαρού προβλήματος. Επειδή ο ευρετικός μηχανισμός που προκύπτει είναι ένα ακριβές κόστος για το χαλαρό πρόβλημα, πρέπει να υπακούει στην τριγωνική ανισότητα και επομένως είναι **συνεπής** (consistent — δείτε στην Ενότητα 4.1, στην υποενότητα για την αναζήτηση A*).

ΧΑΛΑΡΟ ΠΡΟΒΛΗΜΑ

Αν ένας ορισμός προβλήματος διατυπωθεί σε μια τυπική γλώσσα, είναι δυνατό να κατασκευαστούν χαλαρά προβλήματα αυτόματα.[6] Για παράδειγμα, αν οι ενέργειες του παζλ των 8 πλακιδίων περιγραφούν έτσι

> Ένα πλακίδιο μπορεί να μετακινηθεί από το τετράγωνο Α στο τετράγωνο Β αν
> το Α συνορεύει οριζόντια ή κάθετα με το Β **και** το Β είναι κενό

τότε μπορούμε να παραγάγουμε τρία χαλαρά προβλήματα αφαιρώντας τη μία ή και τις δύο συνθήκες:

(α) Ένα πλακίδιο μπορεί να μετακινηθεί από το τετράγωνο Α στο τετράγωνο Β αν το Α συνορεύει με το Β.

(β) Ένα πλακίδιο μπορεί να μετακινηθεί από το τετράγωνο Α στο τετράγωνο Β αν το Β είναι κενό.

(γ) Ένα πλακίδιο μπορεί να μετακινηθεί από το τετράγωνο Α στο τετράγωνο Β.

Από το (α) μπορεί να προκύψει η h_2 (απόσταση Manhattan). Το σκεπτικό είναι ότι η h_2 θα ήταν η κατάλληλη βαθμολογία αν μετακινούσαμε κάθε πλακίδιο με τη σειρά στον προορισμό του. Ο ευρετικός μηχανισμός που προκύπτει από το (β) εξετάζεται στην Άσκηση 4.9. Από το (γ) μπορεί να προκύψει η h_1 (πλακίδια που δεν είναι στη θέση τους), επειδή αυτή θα ήταν η κατάλληλη βαθμολογία αν τα πλακίδια μπορούσαν να μετακινούνται στον προορισμό τους με ένα μόνο βήμα. Προσέξτε ότι έχει καθοριστική σημασία τα χαλαρά προβλήματα που παράγονται με αυτή την τεχνική να μπορούν να λύνονται ουσιαστικά *χωρίς αναζήτηση*, επειδή οι χαλαροί κανόνες επιτρέπουν να αναλυθεί το πρόβλημα σε οκτώ ανεξάρτητα υποπροβλήματα. Αν το χαλαρό πρόβλημα είναι δύσκολο να λυθεί, τότε οι τιμές του αντίστοιχου ευρετικού μηχανισμού θα είναι δαπανηρό να προκύψουν.[7]

Ένα πρόγραμμα που ονομάζεται ABSOLVER μπορεί να παράγει ευρετικούς μηχανισμούς αυτόματα από ορισμούς προβλημάτων, χρησιμοποιώντας τη μέθοδο των "χαλαρών προβλημάτων" και διάφορες άλλες τεχνικές (Prieditis, 1993). Το ABSOLVER επινόησε έναν ευρετικό μηχανισμό για το παζλ των 8 πλακιδίων καλύτερο από κάθε προϋπάρχοντα και βρήκε τον πρώτο χρήσιμο ευρετικό μηχανισμό για το διάσημο γρίφο του Κύβου του Rubik.

Ένα πρόβλημα με την παραγωγή νέων ευρετικών συναρτήσεων είναι ότι συχνά αποτυγχάνει κανείς να βρει μία και μόνο "σαφώς καλύτερη". Αν είναι διαθέσιμη μια συλλογή παραδεκτών

---

[6] Στα Κεφάλαια 8 και 11, θα περιγράψουμε τυπικές γλώσσες κατάλληλες γι' αυτή τη δουλειά· με τυπικές περιγραφές τις οποίες μπορούμε να χειριζόμαστε η κατασκευή χαλαρών προβλημάτων μπορεί να αυτοματοποιηθεί. Προς το παρόν, θα χρησιμοποιήσουμε φυσική γλώσσα.

[7] Σημειώστε ότι μπορεί να προκύψει ένας τέλειος ευρετικός μηχανισμός αν επιτρέψουμε απλώς στην h να εκτελέσει στα κρυφά μια πλήρη αναζήτηση πρώτα κατά πλάτος. Υπάρχει λοιπόν για τις ευρετικές συναρτήσεις μια σχέση εξισορρόπησης μεταξύ ακρίβειας και χρόνου υπολογισμού.

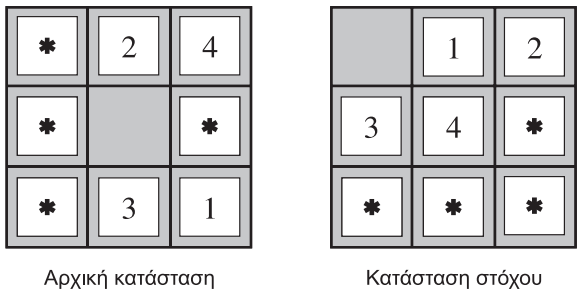
ευρετικών μηχανισμών $h_1 \ldots h_m$ για ένα πρόβλημα και κανένας από αυτούς δεν κυριαρχεί έναντι οποιουδήποτε από τους άλλους, ποιον θα πρέπει να διαλέξουμε; Όπως αποδεικνύεται, δε χρειάζεται να κάνουμε καμία εκλογή. Μπορούμε να πάρουμε ό,τι καλύτερο διαθέτουν όλοι, ορίζοντας τη συνάρτηση:

$$h(n) = \max\{ h_1(n), \ldots, h_m(n) \}$$

Αυτός ο σύνθετος ευρετικός μηχανισμός χρησιμοποιεί όποια συνάρτηση είναι ακριβέστερη για το συγκεκριμένο κόμβο. Επειδή οι ευρετικοί μηχανισμοί που τη συνθέτουν είναι παραδεκτοί, η h είναι παραδεκτή· είναι επίσης εύκολο να αποδείξουμε ότι η h είναι συνεπής. Επίσης, η h κυριαρχεί έναντι όλων των ευρετικών μηχανισμών που τη συνθέτουν.

ΥΠΟΠΡΟΒΛΗΜΑ

Παραδεκτοί ευρετικοί μηχανισμοί μπορούν επίσης να προκύψουν από το κόστος λύσης ενός **υποπροβλήματος** ενός δεδομένου προβλήματος. Για παράδειγμα, η Εικόνα 4.9 παρουσιάζει ένα υποπρόβλημα του στιγμιοτύπου του παζλ των 8 πλακιδίων που είδαμε στην Εικόνα 4.7. Το υποπρόβλημα συνίσταται στο να πάμε τα πλακίδια 1, 2, 3, 4 στις σωστές τους θέσεις. Είναι φανερό ότι το κόστος της βέλτιστης λύσης αυτού του υποπροβλήματος είναι ένα κάτω φράγμα για το κόστος του πλήρους προβλήματος. Προκύπτει ότι αυτός ο ευρετικός μηχανισμός είναι πολύ ακριβέστερος από την απόσταση Manhattan σε μερικές περιπτώσεις.



Εικόνα 4.9 Υποπρόβλημα του στιγμιοτύπου του παζλ των 8 πλακιδίων που είδαμε στην Εικόνα 4.7. Το ζητούμενο είναι να πάνε τα πλακίδια 1, 2, 3 και 4 στις σωστές τους θέσεις, χωρίς να μας ενδιαφέρει τι θα γίνει με τα άλλα πλακίδια.

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ ΠΡΟΤΥΠΩΝ

Η ιδέα στην οποία στηρίζονται οι **βάσεις δεδομένων προτύπων** (pattern databases) είναι να αποθηκεύονται αυτά τα ακριβή κόστη λύσης για κάθε δυνατό στιγμιότυπο υποπροβλήματος — στο παράδειγμά μας, για κάθε δυνατή διάταξη των τεσσάρων πλακιδίων και του κενού. (Προσέξτε ότι οι θέσεις των άλλων τεσσάρων πλακιδίων είναι αδιάφορες για την επίλυση του υποπροβλήματος, αλλά οι μετακινήσεις αυτών των πλακιδίων υπολογίζονται στο κόστος.) Έπειτα βρίσκουμε έναν παραδεκτό ευρετικό μηχανισμό h_{DB} για την κάθε πλήρη κατάσταση που συναντάμε σε μια αναζήτηση ψάχνοντας απλώς για την αντίστοιχη διάταξη υποπροβλήματος στη βάση δεδομένων. Η ίδια η βάση δεδομένων κατασκευάζεται με αναζήτηση προς τα πίσω από την κατάσταση στόχου και με καταγραφή του κόστους κάθε νέου προτύπου που εμφανίζεται· το κόστος αυτής της αναζήτησης ξεπληρώνεται τμηματικά σε πολλά επόμενα στιγμιότυπα του προβλήματος.

Η εκλογή των πλακιδίων 1-2-3-4 είναι μάλλον αυθαίρετη· θα μπορούσαμε να κατασκευάσουμε επίσης βάσεις δεδομένων για τα πλακίδια 5-6-7-8, για τα 2-4-6-8 κ.ο.κ. Κάθε βάση δεδομένων δίνει έναν παραδεκτό ευρετικό μηχανισμό, και αυτοί οι ευρετικοί μηχανισμοί μπορούν να συνδυάζονται, όπως εξηγήσαμε παραπάνω, με χρήση της μέγιστης τιμής. Ένας τέτοιος συνδυασμένος ευρετικός μηχανισμός είναι πολύ πιο ακριβής από την απόσταση Manhattan· ο αριθμός

των κόμβων που παράγονται όταν επιλύεται ένα τυχαίο παζλ των 15 πλακιδίων μπορεί να μειωθεί κατά έναν παράγοντα 1000.

Θα μπορούσε να αναρωτηθεί κανείς αν οι ευρετικοί μηχανισμοί που προκύπτουν από τις βάσεις δεδομένων 1-2-3-4 και 5-6-7-8 μπορούν να *προστεθούν*, αφού τα δύο υποπροβλήματα φαίνεται να μην επικαλύπτονται. Θα μας έδινε αυτό και πάλι έναν παραδεκτό ευρετικό μηχανισμό; Η απάντηση είναι όχι, επειδή οι λύσεις του υποπροβλήματος 1-2-3-4 και του υποπροβλήματος 5-6-7-8 για μια δεδομένη κατάσταση είναι σχεδόν βέβαιο ότι θα έχουν κοινές μερικές κινήσεις — είναι απίθανο τα πλακίδια 1-2-3-4 να μπορέσουν να μετακινηθούν στη θέση τους χωρίς να αγγίξουν τα 5-6-7-8, και *αντίστροφα*. Τι γίνεται όμως αν δε μετράμε αυτές τις κινήσεις; Μπορούμε δηλαδή να μην καταγράφουμε το ολικό κόστος της επίλυσης του υποπροβλήματος 1-2-3-4, αλλά μόνο τον αριθμό των κινήσεων που αφορούν τα πλακίδια 1-2-3-4. Τότε μπορούμε εύκολα να δούμε ότι το άθροισμα των δύο κοστών είναι και πάλι κάτω φράγμα του κόστους επίλυσης ολόκληρου του προβλήματος. Σε αυτή την ιδέα στηρίζονται οι **βάσεις δεδομένων ξένων προτύπων** (disjoint pattern databases). Με τη χρήση τέτοιων βάσεων δεδομένων, μπορούν να επιλύονται τυχαία παζλ των 15 πλακιδίων σε μερικά millisecond — ο αριθμός των κόμβων που παράγονται μειώνεται κατά έναν παράγοντα 10.000 σε σύγκριση με τη χρήση της απόστασης Manhattan. Για τα παζλ των 24 πλακιδίων, μπορεί να επιτευχθεί αύξηση της ταχύτητας κατά ένα εκατομμύριο περίπου.

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ ΞΕΝΩΝ ΠΡΟΤΥΠΩΝ

Οι βάσεις δεδομένων ξένων προτύπων είναι κατάλληλες για τα παζλ συρόμενων αντικειμένων επειδή το πρόβλημα μπορεί να υποδιαιρεθεί με τέτοιον τρόπο ώστε η κάθε μετακίνηση να επηρεάζει μόνο ένα υποπρόβλημα — επειδή μετακινείται μόνο ένα πλακίδιο τη φορά. Σε ένα πρόβλημα όπως ο Κύβος του Rubik, δεν μπορεί να γίνει τέτοια υποδιαίρεση επειδή κάθε μετακίνηση επηρεάζει 8 ή 9 από τους 26 μικρούς κύβους. Μέχρι σήμερα δεν έχει ξεκαθαριστεί πώς μπορούν να οριστούν βάσεις δεδομένων ξένων προτύπων για τέτοια προβλήματα.

Εκμάθηση ευρετικών μηχανισμών από την εμπειρία

Μια ευρετική συνάρτηση $h(n)$ αναμένεται να εκτιμά το κόστος μιας λύσης ξεκινώντας από την κατάσταση στον κόμβο n. Πώς θα μπορούσε ένας πράκτορας να κατασκευάσει μια τέτοια συνάρτηση; Μία απάντηση δόθηκε στην προηγούμενη ενότητα — συγκεκριμένα, να επινοεί ο πράκτορας χαλαρά προβλήματα για τα οποία μπορεί εύκολα να βρεθεί βέλτιστη λύση. Ένας άλλος τρόπος είναι να μαθαίνει ο πράκτορας από την εμπειρία. "Εμπειρία" εδώ σημαίνει, για παράδειγμα, να επιλύσει ο πράκτορας πολλά παζλ των 8 πλακιδίων. Κάθε βέλτιστη λύση για ένα πρόβλημα παζλ των 8 πλακιδίων παρέχει παραδείγματα από τα οποία ο πράκτορας μπορεί να μαθαίνει την $h(n)$. Κάθε παράδειγμα αποτελείται από μια κατάσταση από τη διαδρομή λύσης, και από το πραγματικό κόστος από εκείνο το σημείο. Από αυτά τα παραδείγματα, μπορεί να χρησιμοποιηθεί ένας αλγόριθμος **επαγωγικής μάθησης** (inductive learning) για να κατασκευαστεί μια συνάρτηση $h(n)$ που να μπορεί (με λίγη τύχη) να προβλέπει τα κόστη λύσης για άλλες καταστάσεις που προκύπτουν κατά την αναζήτηση. Τεχνικές γι' αυτόν ακριβώς το σκοπό, οι οποίες χρησιμοποιούν νευρωνικά δίκτυα, δένδρα αποφάσεων και άλλες μεθόδους, παρουσιάζονται στο Κεφάλαιο 18. (Οι μέθοδοι ενισχυτικής μάθησης που περιγράφονται στο Κεφάλαιο 21 έχουν επίσης εφαρμογή.)

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

Οι μέθοδοι επαγωγικής μάθησης λειτουργούν καλύτερα όταν τους παρέχονται κάποια **χαρακτηριστικά** (features) μιας κατάστασης τα οποία έχουν σχέση με την αξιολόγησή της, και όχι μόνο η απλή περιγραφή της κατάστασης. Για παράδειγμα, το χαρακτηριστικό "αριθμός πλακιδίων που δεν είναι στη θέση τους" θα μπορούσε να βοηθήσει στην πρόγνωση της πραγματικής απόστασης μιας κατάστασης από τον στόχο. Ας ονομάσουμε αυτό το χαρακτηριστικό $x_1(n)$. Θα μπορούσαμε να πάρουμε 100 διατάξεις παζλ των 8 πλακιδίων που έχουν παραχθεί τυχαία και να συλλέξουμε στατιστικά στοιχεία για τα πραγματικά τους κόστη λύσης. Μπορεί

να βρίσκαμε ότι όταν η τιμή του $x_1(n)$ είναι 5 το μέσο κόστος λύσης είναι περίπου 14 κ.ο.κ. Με δεδομένα αυτά τα στοιχεία, η τιμή του x_1 μπορεί να χρησιμοποιηθεί για την πρόγνωση της $h(n)$. Φυσικά, μπορούμε να χρησιμοποιήσουμε πολλά χαρακτηριστικά. Ένα δεύτερο χαρακτηριστικό $x_2(n)$ θα μπορούσε να είναι "ο αριθμός των ζευγών γειτονικών πλακιδίων που είναι επίσης γειτονικά και στην κατάσταση στόχου". Πώς θα μπορούσαν να συνδυαστούν τα $x_1(n)$ και $x_2(n)$ για την πρόγνωση της $h(n)$; Μια συνηθισμένη προσέγγιση είναι να χρησιμοποιείται ένας γραμμικός συνδυασμός:

$$h(n) = c_1\, x_1(n) + c_2\, x_2(n)$$

Οι σταθερές c_1 και c_2 προσαρμόζονται έτσι ώστε να μας δώσουν την καλύτερη ταύτιση με τα πραγματικά δεδομένα για τα κόστη λύσης. Ίσως η c_1 να πρέπει να είναι θετική και η c_2 να πρέπει να είναι αρνητική.

4.3 Αλγοριθμοι Τοπικης Αναζητησης και Προβληματα Βελτιστοποιησης

Οι αλγόριθμοι αναζήτησης που είδαμε μέχρι εδώ είναι σχεδιασμένοι να εξερευνούν τους χώρους αναζήτησης συστηματικά. Αυτή τη συστηματικότητα την επιτυγχάνουν διατηρώντας μία ή περισσότερες διαδρομές στη μνήμη και καταγράφοντας ποιες εναλλακτικές περιπτώσεις έχουν εξερευνηθεί σε κάθε σημείο της διαδρομής και ποιες όχι. Όταν βρεθεί μια κατάσταση στόχου, η *διαδρομή* προς αυτή την κατάσταση στόχου είναι και *λύση* του προβλήματος.

Σε πολλά προβλήματα, όμως, η διαδρομή προς τον στόχο δεν έχει σημασία. Για παράδειγμα, στο πρόβλημα των 8 βασιλισσών (δείτε στην Ενότητα 3.2, στην υποενότητα για τα "προβλήματα-παιχνίδια"), εκείνο που έχει σημασία είναι η τελική διάταξη των βασιλισσών και όχι η σειρά με την οποία προστίθενται. Αυτή η κλάση προβλημάτων περιλαμβάνει πολλές σημαντικές εφαρμογές, όπως η σχεδίαση ολοκληρωμένων κυκλωμάτων, η διάταξη εργοστασιακού χώρου παραγωγής (factory-floor layout), ο χρονοπρογραμματισμός εργασιών παραγωγής (job-shop scheduling), ο αυτόματος προγραμματισμός, η βελτιστοποίηση των τηλεπικοινωνιακών δικτύων, η δρομολόγηση της οδικής κυκλοφορίας και η διαχείριση χαρτοφυλακίων.

Αν η διαδρομή προς τον στόχο δεν έχει σημασία, θα μπορούσαμε να εξετάσουμε μια διαφορετική κλάση αλγορίθμων που δεν ενδιαφέρονται καθόλου για τις διαδρομές. Οι αλγόριθμοι **τοπικής αναζήτησης** (local search) λειτουργούν χρησιμοποιώντας μόνο μια **τρέχουσα κατάσταση** (αντί για πολλές διαδρομές) και γενικά μετακινούνται μόνο σε γειτονικές της καταστάσεις. Κανονικά, οι διαδρομές που ακολουθούνται από την αναζήτηση δε διατηρούνται στη μνήμη. Αν και οι αλγόριθμοι τοπικής αναζήτησης δεν είναι συστηματικοί, έχουν δύο σημαντικά πλεονεκτήματα: (1) Χρησιμοποιούν πολύ λίγη μνήμη — συνήθως μια σταθερή ποσότητα — και (2) μπορούν συχνά να βρίσκουν λογικές λύσεις σε μεγάλους ή και άπειρους (συνεχείς) χώρους καταστάσεων για τους οποίους οι συστηματικοί αλγόριθμοι είναι ακατάλληλοι.

ΤΟΠΙΚΗ ΑΝΑΖΗΤΗΣΗ
ΤΡΕΧΟΥΣΑ ΚΑΤΑΣΤΑΣΗ

Εκτός από την επίτευξη των στόχων, οι αλγόριθμοι τοπικής αναζήτησης είναι επίσης χρήσιμοι για την επίλυση αμιγών **προβλήματα βελτιστοποίησης** (optimization problems), στα οποία ο σκοπός είναι να βρεθεί η καλύτερη κατάσταση σύμφωνα με μια **αντικειμενική συνάρτηση** (objective function). Πολλά προβλήματα βελτιστοποίησης δεν ταιριάζουν στο "κανονικό" μοντέλο αναζήτησης που παρουσιάσαμε στο Κεφάλαιο 3. Για παράδειγμα, η φύση παρέχει μια αντικειμενική συνάρτηση — την αναπαραγωγική καταλληλότητα (fitness) — την οποία η δαρβίνεια εξέλιξη θα μπορούσε να θεωρηθεί ότι προσπαθεί να βελτιστοποιήσει, αλλά δεν υπάρχει ούτε "έλεγχος στόχου" ούτε "κόστος διαδρομής" σε αυτό το πρόβλημα.

ΠΡΟΒΛΗΜΑΤΑ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ
ΑΝΤΙΚΕΙΜΕΝΙΚΗ ΣΥΝΑΡΤΗΣΗ

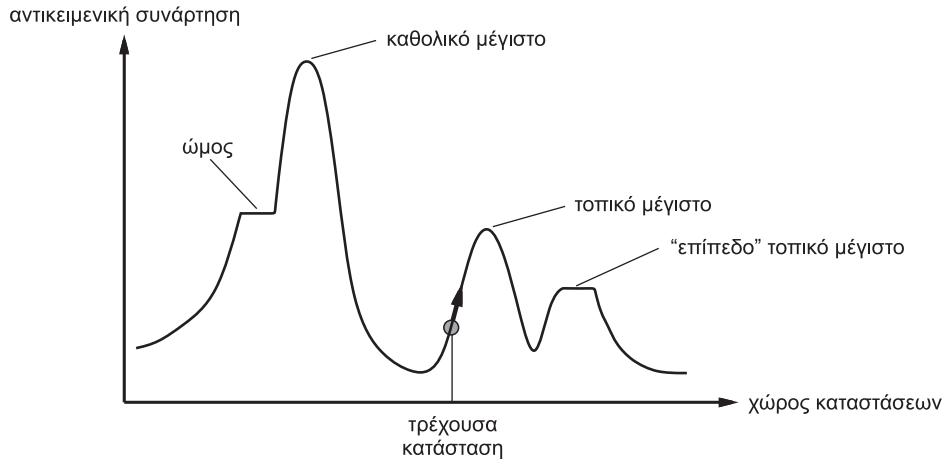
Για να γίνει κατανοητή η τοπική αναζήτηση, είναι πολύ χρήσιμο να εξετάσουμε το **τοπίο του χώρου καταστάσεων** (state space landscape — όπως αυτό της Εικόνας 4.10). Ένα τοπίο έχει "τοποθεσία" (που ορίζεται από την κατάσταση) και "υψόμετρο" (που ορίζεται από την τιμή της ευρετικής συνάρτησης κόστους ή της αντικειμενικής συνάρτησης). Αν το υψόμετρο αντιστοιχεί

ΤΟΠΙΟ ΧΩΡΟΥ ΚΑΤΑΣΤΑΣΕΩΝ

ΚΑΘΟΛΙΚΟ ΕΛΑΧΙΣΤΟ

ΚΑΘΟΛΙΚΟ ΜΕΓΙΣΤΟ

στο κόστος, τότε ο σκοπός είναι να βρεθεί η χαμηλότερη κοιλάδα — ένα **καθολικό ελάχιστο**· αν το υψόμετρο αντιστοιχεί σε μια αντικειμενική συνάρτηση, τότε ο σκοπός είναι να βρεθεί η υψηλότερη κορυφή — ένα **καθολικό μέγιστο**. (Μπορείτε να μετατρέψετε το ένα στο άλλο με απλή εισαγωγή ενός αρνητικού προσήμου.) Οι αλγόριθμοι τοπικής αναζήτησης εξερευνούν αυτό το τοπίο. Ένας **πλήρης** (complete) αλγόριθμος τοπικής αναζήτησης βρίσκει πάντα μια κατάσταση στόχου, αν υπάρχει· ένας **βέλτιστος** (optimal) αλγόριθμος βρίσκει πάντα ένα καθολικό ελάχιστο ή μέγιστο.



Εικόνα 4.10 Μονοδιάστατο τοπίο χώρου καταστάσεων όπου το υψόμετρο αντιστοιχεί στην αντικειμενική συνάρτηση. Ο σκοπός είναι να βρεθεί το καθολικό μέγιστο. Η αναζήτηση με αναρρίχηση λόφων (hill-climbing) τροποποιεί την τρέχουσα κατάσταση επιδιώκοντας να τη βελτιώσει, όπως δείχνει το βέλος. Τα διάφορα τοπογραφικά χαρακτηριστικά ορίζονται στο κείμενο.

Αναζήτηση με αναρρίχηση λόφων

ΑΝΑΡΡΙΧΗΣΗ ΛΟΦΩΝ

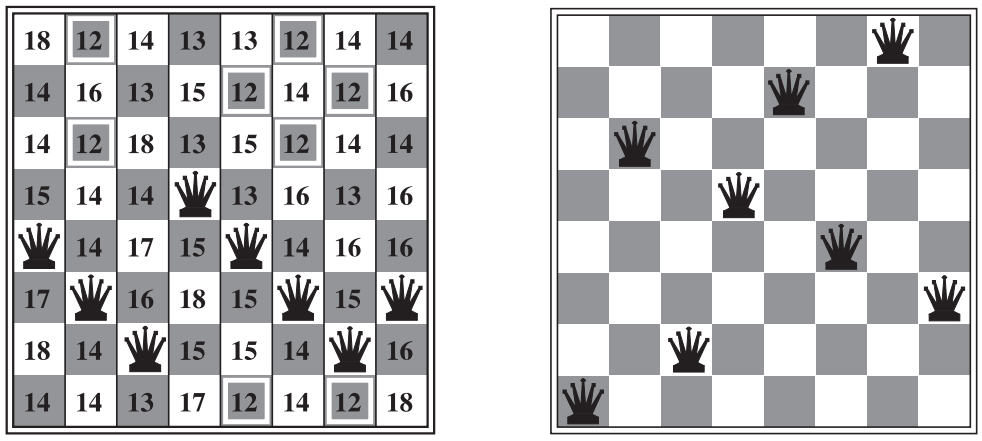
Στην Εικόνα 4.11 παρουσιάζεται ο αλγόριθμος αναζήτησης με **αναρρίχηση λόφων** (hill-climbing). Είναι απλώς ένας βρόχος που μετακινείται συνεχώς στην κατεύθυνση της αυξανόμενης τιμής — δηλαδή, προς τα επάνω. Τερματίζει όταν φτάσει σε μια "κορυφή" όπου κανένας γειτονικός κόμβος δεν έχει υψηλότερη τιμή. Ο αλγόριθμος δε συντηρεί δένδρο αναζήτησης, γι' αυτό στη δομή δεδομένων του τρέχοντος κόμβου χρειάζεται να καταγράφεται μόνο η κατάσταση και η τιμή της αντικειμενικής της συνάρτησης. Η αναρρίχηση λόφων δεν κοιτάζει πιο πέρα από τους άμεσους γείτονες της τρέχουσας κατάστασης. Αυτό μοιάζει σαν να προσπαθεί κανείς να βρει την κορυφή του Έβερεστ μέσα σε πυκνή ομίχλη ενώ πάσχει από αμνησία.

Για να επιδείξουμε την αναρρίχηση λόφων, θα χρησιμοποιήσουμε το **πρόβλημα των 8 βασιλισσών** που παρουσιάσαμε στην Ενότητα 3.2, στην υποενότητα για τα "προβλήματα-παιχνίδια". Οι αλγόριθμοι τοπικής αναζήτησης κανονικά χρησιμοποιούν μια **διατύπωση με πλήρεις καταστάσεις** (complete-state formulation), όπου κάθε κατάσταση έχει 8 βασίλισσες στη σκακιέρα, μία σε κάθε στήλη. Η συνάρτηση διαδόχων επιστρέφει όλες τις δυνατές καταστάσεις που παράγονται από τη μετακίνηση μίας μόνο βασίλισσας σε άλλο τετράγωνο της ίδιας στήλης (επομένως, κάθε κατάσταση έχει $8 \times 7 = 56$ διαδόχους). Η ευρετική συνάρτηση κόστους h είναι ο αριθμός των ζευγών βασιλισσών που αλληλοαπειλούνται, είτε άμεσα είτε έμμεσα. Το καθολικό ελάχιστο της συνάρτησης είναι μηδέν, το οποίο εμφανίζεται μόνο στις τέλειες λύσεις. Η Εικόνα 4.12(α) δείχνει μια κατάσταση με $h = 17$. Η εικόνα δείχνει επίσης τις τιμές όλων των διαδόχων

της, όπου οι καλύτεροι διάδοχοι έχουν $h = 12$. Οι αλγόριθμοι αναρρίχησης λόφων κανονικά διαλέγουν τυχαία από το σύνολο των καλύτερων διαδόχων, αν υπάρχουν περισσότεροι από ένας.

```
function HILL-CLIMBING( πρόβλημα ) returns μια κατάσταση που είναι τοπικό μέγιστο
    inputs: πρόβλημα, ένα πρόβλημα
    local variables:  τρέχων, ένας κόμβος
                      γειτονικός, ένας κόμβος

    τρέχων ← MAKE-NODE( INITIAL-STATE[πρόβλημα])
    loop do
        γειτονικός ← διάδοχος με μεγαλύτερη τιμή από τον κόμβο τρέχων
        if VALUE[γειτονικός] ≤ VALUE[τρέχων] then return STATE[τρέχων]
        τρέχων ← γειτονικός
```

Εικόνα 4.11 Ο αλγόριθμος αναζήτησης με αναρρίχηση λόφων (έκδοση της **πλέον απότομης ανάβασης** — steepest ascend), ο οποίος είναι η πιο βασική τεχνική τοπικής αναζήτησης. Σε κάθε βήμα, ο τρέχων κόμβος αντικαθίσταται από τον καλύτερο γειτονικό· στη συγκεκριμένη έκδοση αυτό σημαίνει το γειτονικό κόμβο με τη μεγαλύτερη τιμή (VALUE), αλλά αν χρησιμοποιούσαμε μια ευρετική εκτίμηση κόστους h θα βρίσκαμε το γειτονικό κόμβο με το μικρότερο h.



Εικόνα 4.12 (α) Κατάσταση 8 βασιλισσών με ευρετική εκτίμηση κόστους $h = 17$, που δείχνει την τιμή του h για κάθε δυνατό διάδοχο που προέκυψε από τη μετακίνηση μιας βασίλισσας πάνω στη στήλη της. Οι καλύτερες κινήσεις είναι σημειωμένες. (β) Τοπικό ελάχιστο στο χώρο καταστάσεων των 8 βασιλισσών· η κατάσταση αυτή έχει $h = 1$ αλλά όλοι οι διάδοχοι έχουν υψηλότερο κόστος.

ΑΠΛΗΣΤΗ ΤΟΠΙΚΗ ΑΝΑΖΗΤΗΣΗ

Η αναρρίχηση λόφων λέγεται μερικές φορές **άπληστη τοπική αναζήτηση** (greedy local search), επειδή αρπάζει μια καλή γειτονική κατάσταση χωρίς να σκεφτεί παραπέρα πού θα πάει στη συνέχεια. Αν και η απληστία θεωρείται ένα από τα επτά θανάσιμα αμαρτήματα, αποδεικνύεται ότι οι άπληστοι αλγόριθμοι συχνά αποδίδουν αρκετά καλά. Η αναρρίχηση λόφων συχνά κάνει πολύ γρήγορη πρόοδο προς μια λύση, επειδή συνήθως είναι αρκετά εύκολο να βελτιώσει μια κακή κατάσταση. Για παράδειγμα, από την κατάσταση της Εικόνας 4.12(α) χρειάζεται μόλις πέντε βήματα για να φτάσει στην κατάσταση της Εικόνας 4.12(β), η οποία έχει $h = 1$ και είναι πολύ

κοντά σε μια λύση. Δυστυχώς, η αναρρίχηση λόφων συχνά παγιδεύεται, για τους παρακάτω λόγους:

- ◊ **Τοπικά μέγιστα:** Τοπικό μέγιστο είναι μια κορυφή υψηλότερη από κάθε γειτονική της κατάσταση αλλά χαμηλότερη από το καθολικό μέγιστο. Οι αλγόριθμοι αναρρίχησης λόφων που φτάνουν κοντά σε ένα τοπικό μέγιστο ανεβαίνουν προς την κορυφή αλλά μετά παγιδεύονται μη έχοντας πού αλλού να πάνε. Αυτό το πρόβλημα απεικονίζεται σχηματικά στην Εικόνα 4.10. Πιο συγκεκριμένα, η κατάσταση της Εικόνας 4.12(β) είναι στην πραγματικότητα ένα τοπικό μέγιστο (τοπικό ελάχιστο για το κόστος h)· κάθε κίνηση μιας μεμονωμένης βασίλισσας χειροτερεύει τα πράγματα.
- ◊ **Κορυφογραμμές:** Μια κορυφογραμμή παρουσιάζεται στην Εικόνα 4.13. Οι κορυφογραμμές έχουν ως αποτέλεσμα μια ακολουθία τοπικών μεγίστων που κάνουν πολύ δύσκολη την πλοήγηση για τους άπληστους αλγόριθμους.
- ◊ **Οροπέδια:** Οροπέδιο είναι μια περιοχή του τοπίου του χώρου καταστάσεων όπου η συνάρτηση αξιολόγησης είναι επίπεδη. Μπορεί να είναι ένα επίπεδο τοπικό μέγιστο από το οποίο δεν υπάρχει διέξοδος προς τα επάνω ή ένας **ώμος**, από τον οποίο μπορεί να γίνει πρόοδος (δείτε στην Εικόνα 4.10). Μια αναζήτηση με αναρρίχηση λόφων είναι δυνατό να μη μπορεί να βρει διέξοδο από το οροπέδιο.

ΩΜΟΣ

Σε όλες αυτές τις περιπτώσεις, ο αλγόριθμος φτάνει σε ένα σημείο από το οποίο δεν μπορεί να κάνει καμία πρόοδο. Ξεκινώντας από μια κατάσταση 8 βασιλισσών που έχει παραχθεί τυχαία, η αναρρίχηση λόφων με την πλέον απότομη ανάβαση παγιδεύεται στο 86% των περιπτώσεων, ενώ επιλύει μόνο το 14% των στιγμιοτύπων του προβλήματος. Δουλεύει γρήγορα, και χρειάζεται μόνο 4 βήματα κατά μέσον όρο όταν επιτυγχάνει και 3 βήματα όταν παγιδεύεται — καθόλου άσχημα για ένα χώρο καταστάσεων με $8^8 \approx 17$ εκατομμύρια καταστάσεις.

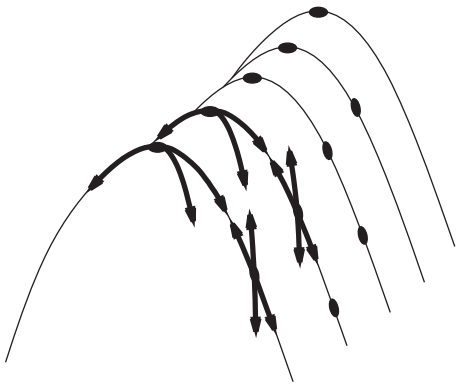
Ο αλγόριθμος της Εικόνας 4.11 σταματά αν φτάσει σε ένα οροπέδιο όπου η καλύτερη διάδοχη κατάσταση έχει την ίδια τιμή με την τρέχουσα κατάσταση. Μήπως θα ήταν καλή ιδέα να συνεχίζει να προχωρά — να επιτρέπεται μια **πλάγια κίνηση** (sideways move) με την ελπίδα ότι το οροπέδιο είναι στην πραγματικότητα ένας ώμος σαν αυτόν που είδαμε στην Εικόνα 4.10; Η απάντηση είναι συνήθως ναι, αλλά πρέπει να προσέξουμε. Αν επιτρέπονται πάντα οι πλάγιες κινήσεις όταν δεν υπάρχουν κινήσεις προς τα επάνω, θα παρουσιάζεται ένας ατέρμονας βρόχος όποτε ο αλγόριθμος φτάνει σε ένα επίπεδο τοπικό μέγιστο που δεν είναι ώμος. Μια συνηθισμένη λύση είναι να τίθεται ένα όριο στον αριθμό των διαδοχικών πλαγίων κινήσεων που επιτρέπονται. Για παράδειγμα, θα μπορούσαμε να επιτρέπουμε μέχρι 100 διαδοχικές πλάγιες κινήσεις για το πρόβλημα των 8 βασιλισσών. Αυτό ανεβάζει το ποσοστό των στιγμιοτύπων του προβλήματος που λύνονται με την αναρρίχηση λόφων από 14% σε 94%. Όμως η επιτυχία έχει το κόστος της· ο αλγόριθμος χρειάζεται κατά μέσον όρο 21 βήματα για κάθε περίπτωση επιτυχίας και 64 για κάθε αποτυχία.

ΠΛΑΓΙΑ ΚΙΝΗΣΗ

Έχουν επινοηθεί πολλές παραλλαγές της αναρρίχησης λόφων. Η **στοχαστική αναρρίχηση λόφων** (stochastic hill climbing) διαλέγει τυχαία από τις διαθέσιμες κινήσεις προς τα επάνω· η πιθανότητα της επιλογής μπορεί να διαφέρει, ανάλογα με το πόσο απότομη κλίση έχει κάθε τέτοια κίνηση. Η παραλλαγή αυτή συνήθως συγκλίνει πιο αργά από τη μέθοδο της πλέον απότομης ανάβασης, αλλά σε μερικά τοπία καταστάσεων βρίσκει καλύτερες λύσεις. Η **αναρρίχηση λόφων με την πρώτη επιλογή** (first-choice hill climbing) υλοποιεί τη στοχαστική αναρρίχηση λόφων παράγοντας διάδοχες καταστάσεις τυχαία, μέχρι να παραχθεί κάποια που είναι καλύτερη από την τρέχουσα κατάσταση. Η στρατηγική αυτή είναι καλή όταν μια κατάσταση έχει πολλές (π.χ. χιλιάδες) διάδοχες καταστάσεις. Η Άσκηση 4.16 σας ζητά να τη διερευνήσετε.

ΣΤΟΧΑΣΤΙΚΗ ΑΝΑΡΡΙΧΗΣΗ ΛΟΦΩΝ

ΑΝΑΡΡΙΧΗΣΗ ΛΟΦΩΝ ΜΕ ΤΗΝ ΠΡΩΤΗ ΕΠΙΛΟΓΗ



Εικόνα 4.13 Η εικόνα δείχνει γιατί οι κορυφογραμμές προκαλούν δυσκολίες στην αναρρίχηση λόφων. Το πλέγμα των καταστάσεων (μαύροι κύκλοι) υπερτίθεται σε μια κορυφογραμμή που ανέρχεται από τα αριστερά προς τα δεξιά, δημιουργώντας μια ακολουθία τοπικών μεγίστων που δε συνδέονται άμεσα μεταξύ τους. Από κάθε τοπικό μέγιστο, όλες οι διαθέσιμες ενέργειες έχουν κατεύθυνση προς τα κάτω.

ΑΝΑΡΡΙΧΗΣΗ ΛΟΦΩΝ ΜΕ ΤΥΧΑΙΕΣ ΕΠΑΝΕΚΚΙΝΗΣΕΙΣ

Οι αλγόριθμοι αναρρίχησης λόφων που περιγράψαμε μέχρι εδώ είναι μη πλήρεις — συχνά δεν καταφέρνουν να βρουν μια κατάσταση στόχου ενώ υπάρχει, επειδή μπορεί να παγιδευτούν σε τοπικά μέγιστα. Η **αναρρίχηση λόφων με τυχαίες επανεκκινήσεις** (random-restart hill climbing) υιοθετεί το γνωστό ρητό "Αν δεν τα καταφέρεις με την πρώτη, δοκίμασε ξανά και ξανά". Πραγματοποιεί μια σειρά αναζητήσεων αναρρίχησης λόφων από τυχαία παραγόμενες αρχικές καταστάσεις,[8] σταματώντας όταν βρεθεί μια κατάσταση στόχου. Είναι πλήρης με πιθανότητα που πλησιάζει το 1, για τον απλό λόγο ότι τελικά θα παραγάγει ως αρχική κατάσταση μια κατάσταση στόχου. Αν κάθε αναζήτηση με αναρρίχηση λόφων έχει πιθανότητα επιτυχίας p, ο αναμενόμενος αριθμός επανεκκινήσεων που απαιτούνται είναι $1/p$. Στα στιγμιότυπα του προβλήματος των 8 βασιλισσών όπου δεν επιτρέπονται πλάγιες κινήσεις είναι $p \approx 0{,}14$ και επομένως χρειάζονται περίπου 7 επαναλήψεις για να βρεθεί μια κατάσταση στόχου (6 αποτυχίες και 1 επιτυχία). Ο αναμενόμενος αριθμός βημάτων είναι ίσος με το κόστος μίας επιτυχημένης επανάληψης συν $(1 - p) / p$ επί το κόστος αποτυχίας, δηλαδή περίπου 22 βήματα. Όταν επιτρέπονται οι πλάγιες κινήσεις, χρειάζονται κατά μέσον όρο $1 / 0{,}94 \approx 1{,}06$ επαναλήψεις και $(1 \times 21) + (0{,}06 / 0{,}94) \times 64 \approx 25$ βήματα. Επομένως, η αναρρίχηση λόφων με τυχαίες επανεκκινήσεις είναι πραγματικά πολύ αποτελεσματική για το πρόβλημα των 8 βασιλισσών. Ακόμα και με τρία εκατομμύρια βασίλισσες, η προσέγγιση αυτή μπορεί να βρει λύσεις σε λιγότερο από ένα λεπτό.[9]

Η επιτυχία της αναρρίχησης λόφων εξαρτάται σε μεγάλο βαθμό από το σχήμα του τοπίου του χώρου καταστάσεων. Αν υπάρχουν λίγα τοπικά μέγιστα και οροπέδια, η αναρρίχηση λόφων με τυχαίες επανεκκινήσεις θα βρει μια καλή λύση πολύ γρήγορα. Από την άλλη, πολλά πραγματικά προβλήματα έχουν ένα τοπίο που μοιάζει περισσότερο με μια οικογένεια σκαντζόχοιρων σε επίπεδο έδαφος, με μικροσκοπικούς σκαντζόχοιρους που ζουν πάνω στην μύτη κάθε βελόνας σκαντζόχοιρου, και αυτό επαναλαμβάνεται επ' άπειρον. Τα NP-δύσκολα προβλήματα συνήθως έχουν εκθετικά μεγάλο αριθμό τοπικών μεγίστων στα οποία μπορεί να παγιδευτούν. Ωστόσο, ένα

[8] Η παραγωγή μιας *τυχαίας* κατάστασης από έναν έμμεσα καθορισμένο χώρο καταστάσεων μπορεί να είναι και η ίδια δύσκολο πρόβλημα.

[9] Οι Luby *κ.α.* (1993) απέδειξαν ότι είναι καλύτερο σε μερικές περιπτώσεις να επανεκκινεί ένας τυχαιοποιημένος αλγόριθμος αναζήτησης μετά από ένα συγκεκριμένο σταθερό χρονικό διάστημα και ότι αυτό μπορεί να είναι *πολύ* πιο αποδοτικό από το να αφήνουμε κάθε αναζήτηση να συνεχίζει επ' αόριστον. Η απαγόρευση ή ο περιορισμός των πλαγίων κινήσεων είναι ένα τέτοιο παράδειγμα.

αρκετά καλό τοπικό μέγιστο συχνά μπορεί να βρεθεί μετά από ένα μικρό αριθμό επανεκκινήσεων.

Αναζήτηση με προσομοιωμένη ανόπτηση

Ένας αλγόριθμος αναρρίχησης λόφων που δεν κάνει *ποτέ* "κατηφορικές" κινήσεις προς καταστάσεις μικρότερης αξίας (ή μεγαλύτερου κόστους) είναι σίγουρα μη πλήρης, επειδή μπορεί να παγιδευτεί σε ένα τοπικό μέγιστο. Αντίθετα, ένας καθαρά τυχαίος περίπατος — δηλαδή, μετακίνηση σε μια διάδοχη κατάσταση που διαλέγεται ομοιόμορφα στην τύχη από το σύνολο των διάδοχων καταστάσεων — είναι πλήρης αλλά έχει εξαιρετικά χαμηλή απόδοση. Φαίνεται λοιπόν λογικό να προσπαθήσουμε να συνδυάσουμε την αναρρίχηση λόφων με έναν τυχαίο περίπατο με κάποιον τρόπο που να προσφέρει και αποδοτικότητα και πληρότητα. Ένας τέτοιος αλγόριθμος είναι η **προσομοιωμένη ανόπτηση** (simulated annealing). Στη μεταλλουργία, **ανόπτηση** είναι η διαδικασία που χρησιμοποιείται για τη σκλήρυνση των μετάλλων και του γυαλιού, με θέρμανσή τους σε υψηλή θερμοκρασία και μετά με βαθμιαία ψύξη τους, με αποτέλεσμα να καταλήγει το υλικό σε μια κρυσταλλική κατάσταση χαμηλής ενέργειας. Για να κάνουμε κατανοητή την προσομοιωμένη ανόπτηση, ας αλλάξουμε την άποψή μας από αναρρίχηση λόφων σε **κατάβαση πλαγιάς** (gradient descent — δηλαδή, ελαχιστοποίηση του κόστους) και ας φανταστούμε ότι η δουλειά που έχουμε να κάνουμε είναι να πάμε ένα μπαλάκι του πινγκ-πονγκ στη βαθύτερη ρωγμή μιας ανώμαλης επιφάνειας. Αν αφήσουμε απλώς το μπαλάκι να κυλήσει, θα πάει και θα σταματήσει σε ένα τοπικό ελάχιστο. Αν τραντάξουμε την επιφάνεια, μπορούμε να κάνουμε το μπαλάκι να αναπηδήσει έξω από το τοπικό ελάχιστο. Το κόλπο είναι να τραντάζουμε την επιφάνεια όσο δυνατά χρειάζεται για να αναπηδά το μπαλάκι έξω από τα τοπικά ελάχιστα αλλά όχι τόσο δυνατά ώστε να το κάνουμε να βγει και από το καθολικό ελάχιστο. Η λύση της προσομοιωμένης ανόπτησης είναι να αρχίζουμε με δυνατό τράνταγμα (δηλαδή σε υψηλή θερμοκρασία) και μετά να μειώνουμε βαθμιαία την ένταση του τραντάγματος (δηλαδή να χαμηλώνουμε τη θερμοκρασία).

ΠΡΟΣΟΜΟΙΩΜΕΝΗ ΑΝΟΠΤΗΣΗ

ΚΑΤΑΒΑΣΗ ΠΛΑΓΙΑΣ

Ο εσωτερικός βρόχος του αλγόριθμου της προσομοιωμένης ανόπτησης (Εικόνα 4.14) μοιάζει πολύ με την αναρρίχηση λόφων. Αντί όμως να διαλέγει την *καλύτερη* κίνηση, διαλέγει μια *τυχαία* κίνηση. Αν η κίνηση βελτιώνει τα πράγματα γίνεται πάντα αποδεκτή. Αλλιώς, ο αλγόριθμος αποδέχεται την κίνηση με κάποια πιθανότητα μικρότερη του 1. Η πιθανότητα μειώνεται εκθετικά με το "πόσο κακή" είναι η κίνηση — την ποσότητα ΔE κατά την οποία χειροτερεύει η αξιολόγηση. Η πιθανότητα μειώνεται επίσης καθώς κατεβαίνει η "θερμοκρασία" T· οι "κακές" κινήσεις είναι πιο πιθανό να γίνουν αποδεκτές στην αρχή, όταν η θερμοκρασία είναι υψηλή, ενώ γίνονται λιγότερο πιθανές καθώς μειώνεται το T. Μπορεί να αποδειχτεί ότι, αν το *χρονοδιάγραμμα* χαμηλώνει το T αρκετά αργά, ο αλγόριθμος θα βρει ένα καθολικό βέλτιστο σημείο με πιθανότητα που πλησιάζει το 1.

Η προσομοιωμένη ανόπτηση χρησιμοποιήθηκε για πρώτη φορά εκτεταμένα για την επίλυση προβλημάτων διάταξης κυκλωμάτων VLSI στις αρχές της δεκαετίας του 1980. Έχει εφαρμοστεί ευρύτατα στο χρονοπρογραμματισμό εργοστασίου και σε άλλες εργασίες βελτιστοποίησης μεγάλης κλίμακας. Στην Άσκηση 4.16, σας ζητείται να συγκρίνετε την απόδοσή της με εκείνη της αναρρίχησης λόφων με τυχαίες επανεκκινήσεις για το πρόβλημα των n βασιλισσών.

```
function SIMULATED-ANNEALING( πρόβλημα, χρονοδιάγραμμα) returns μια κατάσταση λύσης
    inputs:          πρόβλημα, ένα πρόβλημα
                     χρονοδιάγραμμα, μια αντιστοιχία χρόνων σε "θερμοκρασίες"
    local variables: τρέχων, ένας κόμβος
                     επόμενος, ένας κόμβος
                     T, μια "θερμοκρασία" που καθορίζει την πιθανότητα βημάτων προς τα κάτω

    τρέχων ← MAKE-NODE( INITIAL-STATE[πρόβλημα])
    for t ← 1 to ∞ do
        T ← χρονοδιάγραμμα[t]
        if T = 0 then return τρέχων
        επόμενος ← τυχαία επιλεγμένος διάδοχος του κόμβου τρέχων
        ΔE ← VALUE[επόμενος] – VALUE[τρέχων]
        if ΔE > 0 then τρέχων ← επόμενος
        else τρέχων ← επόμενος μόνο με πιθανότητα e^(ΔE / T)
```

Εικόνα 4.14 Ο αλγόριθμος αναζήτησης με προσομοιωμένη ανόπτηση, μια έκδοση της στοχαστικής αναρρίχησης λόφων όπου μερικές κινήσεις προς τα κάτω επιτρέπονται. Οι κατωφερείς κινήσεις γίνονται εύκολα αποδεκτές νωρίς κατά το χρονοπρογραμματισμό της ανόπτησης και λιγότερο συχνά με το πέρασμα του χρόνου. Η είσοδος *χρονοδιάγραμμα* προσδιορίζει την τιμή του *T* σε συνάρτηση με το χρόνο.

Τοπική ακτινική αναζήτηση

ΤΟΠΙΚΗ ΑΚΤΙΝΙΚΗ ΑΝΑΖΗΤΗΣΗ

Η διατήρηση στη μνήμη μόνο ενός κόμβου φαίνεται ίσως ακραία αντίδραση στο πρόβλημα των περιορισμών μνήμης. Ο αλγόριθμος **τοπικής ακτινικής αναζήτησης** (local beam search)[10] παρακολουθεί *k* καταστάσεις αντί μόνο μία. Ξεκινά με *k* τυχαία παραγόμενες καταστάσεις. Σε κάθε βήμα, παράγονται όλοι οι διάδοχοι και των *k* καταστάσεων. Αν οποιαδήποτε από αυτές ικανοποιεί το στόχο, ο αλγόριθμος σταματά. Αλλιώς, επιλέγει τους *k* καλύτερους διαδόχους από ολόκληρη τη λίστα και επαναλαμβάνει.

Με την πρώτη ματιά, μια τοπική ακτινική αναζήτηση με *k* καταστάσεις ίσως φαίνεται να μην είναι τίποτα περισσότερο από την εκτέλεση *k* τυχαίων επανεκκινήσεων παράλληλα, αντί με τη σειρά. Στην πραγματικότητα, οι δύο αλγόριθμοι είναι εντελώς διαφορετικοί. Στην αναζήτηση με τυχαίες επανεκκινήσεις, η κάθε διαδικασία αναζήτησης εκτελείται ανεξάρτητα από τις άλλες. *Σε μια τοπική ακτινική αναζήτηση, μεταβιβάζονται χρήσιμες πληροφορίες μεταξύ των k παράλληλων νημάτων αναζήτησης*. Για παράδειγμα, αν μία κατάσταση παραγάγει πολλούς καλούς διαδόχους και οι άλλες $k - 1$ καταστάσεις παραγάγουν όλες κακούς διαδόχους, το αποτέλεσμα είναι ότι η πρώτη κατάσταση λέει στις άλλες: "Ελάτε εδωπέρα! Το χορτάρι είναι πιο πράσινο!" Ο αλγόριθμος εγκαταλείπει γρήγορα τις άγονες αναζητήσεις και μεταφέρει τους πόρους του εκεί που γίνεται η περισσότερη πρόοδος.

Στην απλούστερη μορφή της, η τοπική ακτινική αναζήτηση μπορεί να πάσχει από έλλειψη ποικιλίας μεταξύ των *k* καταστάσεων — οι αναζητήσεις μπορεί γρήγορα να συγκεντρωθούν σε μια μικρή περιοχή του χώρου καταστάσεων, κάνοντας την αναζήτηση ελάχιστα καλύτερη από μια δαπανηρή έκδοση της αναρρίχησης λόφων. Μια παραλλαγή που ονομάζεται **στοχαστική ακτινική αναζήτηση** (stochastic beam search), ανάλογη της στοχαστικής αναρρίχησης λόφων, βοηθά να αμβλυνθεί αυτό το πρόβλημα. Αντί να διαλέγει τους *k* καλύτερους από τη δεξαμενή των υποψηφίων διαδόχων, η στοχαστική ακτινική αναζήτηση διαλέγει *k* διαδόχους στην τύχη,

ΣΤΟΧΑΣΤΙΚΗ ΑΚΤΙΝΙΚΗ ΑΝΑΖΗΤΗΣΗ

[10] Η τοπική ακτινική αναζήτηση είναι μια προσαρμογή της **ακτινικής αναζήτησης** (**beam search**), η οποία είναι ένας αλγόριθμος που βασίζεται στη διαδρομή.

όπου η πιθανότητα εκλογής ενός δεδομένου διαδόχου είναι αύξουσα συνάρτηση της αξίας του. Η στοχαστική ακτινική αναζήτηση έχει κάποια ομοιότητα με τη διαδικασία της φυσικής επιλογής, όπου οι "διάδοχοι" (οι απόγονοι) μιας "κατάστασης" (ενός οργανισμού) αποτελούν τον πληθυσμό της επόμενης γενιάς σύμφωνα με την "αξία" τους (εξελικτική καταλληλότητα).

Γενετικοί αλγόριθμοι

ΓΕΝΕΤΙΚΟΣ ΑΛΓΟΡΙΘΜΟΣ

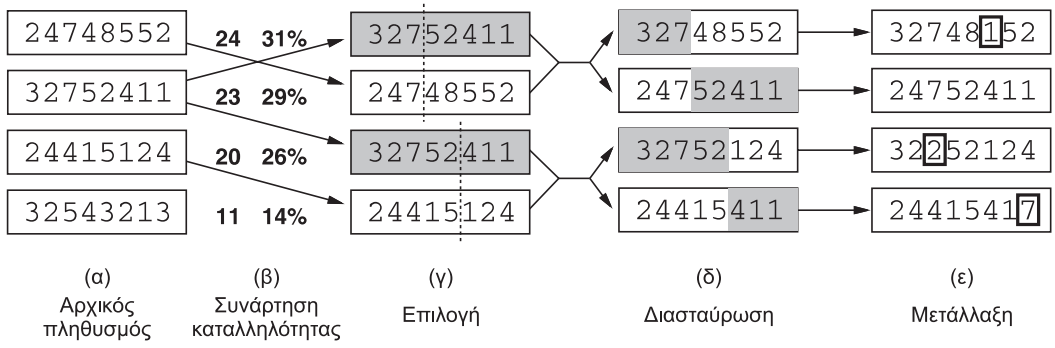
Γενετικός αλγόριθμος (genetic algorithm) είναι μια παραλλαγή της στοχαστικής ακτινικής αναζήτησης όπου οι διάδοχες καταστάσεις παράγονται με το συνδυασμό *δύο* γονικών καταστάσεων, και όχι με την τροποποίηση μιας μεμονωμένης κατάστασης. Η αναλογία με τη φυσική επιλογή είναι η ίδια όπως της στοχαστικής ακτινικής αναζήτησης, με τη διαφορά ότι τώρα έχουμε να κάνουμε με γενετήσια και όχι άφυλη αναπαραγωγή.

ΠΛΗΘΥΣΜΟΣ
ΑΤΟΜΟ

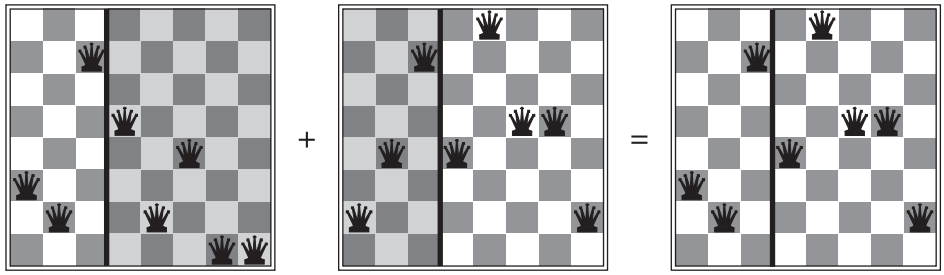
Όπως και η ακτινική αναζήτηση, οι γενετικοί αλγόριθμοι ξεκινούν με ένα σύνολο k τυχαία παραγόμενων καταστάσεων που ονομάζονται **πληθυσμός**. Κάθε κατάσταση, ή **άτομο** (individual), αναπαρίσταται με μια συμβολοσειρά από ένα πεπερασμένο αλφάβητο — συνήθως μια ακολουθία από 0 και 1. Για παράδειγμα, μια κατάσταση 8 βασιλισσών πρέπει να καθορίζει τις θέσεις 8 βασιλισσών, κάθε μία τοποθετημένη σε μια στήλη 8 τετραγώνων, και επομένως απαιτεί $8 \times \log_2 8 = 24$ bit. Εναλλακτικά, η κατάσταση θα μπορούσε να αναπαρασταθεί με 8 ψηφία από την περιοχή 1 έως 8. (Όπως θα δούμε αργότερα, οι δύο κωδικοποιήσεις συμπεριφέρονται διαφορετικά.) Η Εικόνα 4.15(α) δείχνει έναν πληθυσμό τεσσάρων οκταψήφιων συμβολοσειρών που αντιπροσωπεύουν καταστάσεις των 8 βασιλισσών.

ΣΥΝΑΡΤΗΣΗ ΚΑΤΑΛΛΗΛΟΤΗΤΑΣ

Η παραγωγή της επόμενης γενιάς καταστάσεων παρουσιάζεται στις Εικόνες 4.15(β)–(ε). Στη (β), κάθε κατάσταση βαθμολογείται με μια συνάρτηση αξιολόγησης ή (στην ορολογία των γενετικών αλγορίθμων) τη **συνάρτηση καταλληλότητας** (fitness function). Μια συνάρτηση καταλληλότητας θα πρέπει να επιστρέφει υψηλότερες τιμές για τις καλύτερες καταστάσεις, γι' αυτό στο πρόβλημα των 8 βασιλισσών χρησιμοποιούμε τον αριθμό των ζευγών βασιλισσών που *δεν απειλούνται*, ο οποίος για μια λύση είναι 28. Οι τιμές των τεσσάρων καταστάσεων είναι 24, 23, 20 και 11. Σε αυτή τη συγκεκριμένη παραλλαγή του γενετικού αλγόριθμου, η πιθανότητα εκλογής για αναπαραγωγή είναι ευθέως ανάλογη με τη βαθμολογία καταλληλότητας, και τα ποσοστά είναι σημειωμένα δίπλα στις αριθμητικές βαθμολογίες.



Εικόνα 4.15 Ο γενετικός αλγόριθμος. Ο αρχικός πληθυσμός στο (α) βαθμολογείται από τη συνάρτηση καταλληλότητας στο (β), με αποτέλεσμα τα ζευγαρώματα στο (γ). Τα ζεύγη παράγουν απογόνους στο (δ), οι οποίοι υπόκεινται σε μετάλλαξη στο (ε).



Εικόνα 4.16 Οι καταστάσεις των 8 βασιλισσών που αντιστοιχούν στους δύο πρώτους γονείς της Εικόνας 4.15(γ) και στον πρώτο απόγονο της Εικόνας 4.15(δ). Οι σκιασμένες στήλες χάνονται κατά το βήμα της διασταύρωσης ενώ οι μη σκιασμένες στήλες διατηρούνται.

Στο (γ), επιλέγεται για αναπαραγωγή μια τυχαία εκλογή δύο ζευγών, σύμφωνα με τις πιθανότητες της (β). Προσέξτε ότι ένα άτομο επιλέγεται δύο φορές και ένα δεν επιλέγεται καθόλου.[11] Για κάθε ζεύγος που θα ζευγαρωθεί, επιλέγεται τυχαία ένα σημείο **διασταύρωσης** (crossover) από τις θέσεις μέσα στη συμβολοσειρά. Στην Εικόνα 4.15, το σημείο διασταύρωσης του πρώτου ζεύγους είναι μετά το τρίτο ψηφίο και του δεύτερου ζεύγους μετά το πέμπτο ψηφίο.[12]

ΔΙΑΣΤΑΥΡΩΣΗ

Στο (δ), δημιουργούνται οι ίδιοι οι απόγονοι με διασταύρωση των γονικών συμβολοσειρών στο σημείο διασταύρωσης. Για παράδειγμα, το πρώτο παιδί του πρώτου ζεύγους παίρνει τα πρώτα τρία ψηφία από τον πρώτο γονέα και τα υπόλοιπα ψηφία από το δεύτερο, ενώ το δεύτερο παιδί παίρνει τα τρία πρώτα ψηφία από το δεύτερο γονέα και τα υπόλοιπα από τον πρώτο. Οι καταστάσεις των 8 βασιλισσών που αντιστοιχούν σε αυτό το βήμα της αναπαραγωγής παρουσιάζονται στην Εικόνα 4.16. Το παράδειγμα επιδεικνύει το γεγονός ότι, όταν δύο γονικές καταστάσεις είναι πολύ διαφορετικές, η πράξη της διασταύρωσης μπορεί να παραγάγει μια κατάσταση που απέχει πολύ και από τις δύο γονικές καταστάσεις. Συμβαίνει συχνά ο πληθυσμός να είναι αρκετά ποικιλόμορφος νωρίς στη διαδικασία, και έτσι η διασταύρωση (όπως και η προσομοιωμένη ανόπτηση) συχνά κάνει μεγάλα βήματα στο χώρο καταστάσεων νωρίς στη διαδικασία αναζήτησης και μικρότερα βήματα αργότερα, όταν τα περισσότερα άτομα είναι αρκετά όμοια.

ΜΕΤΑΛΛΑΞΗ

Τέλος, στο (ε) κάθε θέση υπόκειται σε τυχαία **μετάλλαξη** (mutation) με μια μικρή ανεξάρτητη πιθανότητα. Έχει μεταλλαχτεί ένα ψηφίο στον πρώτο, τον τρίτο και τον τέταρτο απόγονο. Στο πρόβλημα των 8 βασιλισσών, αυτό αντιστοιχεί με την επιλογή μιας βασίλισσας στην τύχη και τη μετακίνησή της σε ένα τυχαίο τετράγωνο της στήλης της. Στην Εικόνα 4.17 περιγράφεται ένας αλγόριθμος που υλοποιεί όλα αυτά τα βήματα.

Όπως και η στοχαστική ακτινική αναζήτηση, οι γενετικοί αλγόριθμοι συνδυάζουν μια τάση αναζήτησης καλύτερων λύσεων με την τυχαία εξερεύνηση και ανταλλαγή πληροφοριών μεταξύ των παράλληλων νημάτων αναζήτησης. Το κύριο πλεονέκτημα των γενετικών αλγορίθμων, αν υπάρχει, οφείλεται στην πράξη της διασταύρωσης. Μπορεί ωστόσο να αποδειχτεί μαθηματικά ότι αν οι θέσεις του γενετικού κώδικα μετατεθούν αρχικά σε μια τυχαία διάταξη η διασταύρωση δεν προσφέρει κανένα πλεονέκτημα. Διαισθητικά μπορούμε να πούμε ότι το πλεονέκτημα οφείλεται στην ικανότητα της διασταύρωσης να συνδυάζει μεγάλες ομάδες γραμμάτων που εξελίχθηκαν ανεξάρτητα για να πραγματοποιούν χρήσιμες λειτουργίες, ανεβάζοντας έτσι το επίπεδο

[11] Υπάρχουν πολλές παραλλαγές αυτού του κανόνα επιλογής. Η μέθοδος του **ξεδιαλέγματος** (**culling**), όπου όλα τα άτομα κάτω από μια δεδομένη στάθμη απορρίπτονται, μπορεί να αποδειχτεί ότι συγκλίνει γρηγορότερα από την τυχαία επιλογή (Baum *κ.α.*, 1995).

[12] Αυτό είναι το σημείο όπου η κωδικοποίηση έχει σημασία. Αν χρησιμοποιηθεί κωδικοποίηση των 24 bit αντί των 8 ψηφίων, τότε το σημείο διασταύρωσης έχει πιθανότητα 2/3 να βρίσκεται στο εσωτερικό ενός ψηφίου, με αποτέλεσμα μια ουσιαστικά αυθαίρετη μετάλλαξη αυτού του ψηφίου.

της αναλυτικότητας στο οποίο λειτουργεί η αναζήτηση. Για παράδειγμα, μπορεί η τοποθέτηση των τριών πρώτων βασιλισσών στις θέσεις 2, 4 και 6 (όπου δεν αλληλοαπειλούνται) να αποτελεί μια χρήσιμη ομάδα που μπορεί να συνδυαστεί με άλλες ομάδες για να κατασκευαστεί μια λύση.

```
function GENETIC-ALGORITHM( πληθυσμός, FITNESS-FN ) returns ένα άτομο
    inputs:   πληθυσμός, ένα σύνολο ατόμων
              FITNESS-FN, συνάρτηση που μετρά την καταλληλότητα ενός ατόμου

    repeat
        νέος_πληθυσμός ← κενό σύνολο
        loop for i from 1 to SIZE( πληθυσμός ) do
            x ← RANDOM-SELECTION( πληθυσμός, FITNESS-FN )
            y ← RANDOM-SELECTION( πληθυσμός, FITNESS-FN )
            παιδί ← REPRODUCE( x, y )
            if (μικρή τυχαία πιθανότητα) then παιδί ← MUTATE( παιδί )
            πρόσθεσε παιδί σε νέος_πληθυσμός
        πληθυσμός ← νέος_πληθυσμός
    until κάποιο άτομο είναι αρκετά κατάλληλο ή έχει περάσει αρκετός χρόνος
    return το καλύτερο άτομο από τον πληθυσμό, σύμφωνα με τη FITNESS-FN


function REPRODUCE( x, y ) returns ένα άτομο
    inputs: x, y, γονικά άτομα

    n ← LENGTH( x )
    c ← τυχαίος αριθμός από 1 μέχρι n
    return APPEND( SUBSTRING( x, 1, c ), SUBSTRING( y, c + 1, n ))
```

Εικόνα 4.17 Γενετικός αλγόριθμος. Ο αλγόριθμος είναι ο ίδιος με εκείνον που σκιαγραφήσαμε στην Εικόνα 4.15, με μία διαφορά: Σε αυτή την πιο δημοφιλή έκδοση, κάθε ζευγάρωμα δύο γονέων παράγει μόνο ένα απόγονο, και όχι δύο.

ΣΧΗΜΑ

Η θεωρία των γενετικών αλγορίθμων εξηγεί πώς γίνεται αυτό χρησιμοποιώντας την ιδέα του **σχήματος** (schema), το οποίο είναι ένα τμήμα συμβολοσειράς όπου μερικές από τις θέσεις μπορούν να αφεθούν απροσδιόριστες. Για παράδειγμα, το σχήμα 246***** περιγράφει όλες τις καταστάσεις των 8 βασιλισσών στις οποίες οι τρεις πρώτες βασίλισσες βρίσκονται στις θέσεις 2, 4 και 6, αντίστοιχα. Οι συμβολοσειρές που ταιριάζουν στο σχήμα (όπως η 24613578) ονομάζονται **στιγμιότυπα** (instances) του σχήματος. Μπορεί να αποδεχτεί ότι, αν η μέση καταλληλότητα των στιγμιοτύπων ενός σχήματος είναι πάνω από τη μέση τιμή, τότε ο αριθμός των στιγμιοτύπων του σχήματος μέσα στον πληθυσμό θα αυξάνεται με το χρόνο. Βέβαια, αυτό το φαινόμενο είναι απίθανο να είναι σημαντικό αν τα γειτονικά bit είναι εντελώς άσχετα μεταξύ τους, επειδή τότε θα υπάρχουν λίγες συνεχόμενες ομάδες που να παρέχουν ένα συνεπές πλεονέκτημα. Οι γενετικοί αλγόριθμοι δουλεύουν καλύτερα όταν τα σχήματα αντιστοιχούν σε συνιστώσες μιας λύσης που έχουν νόημα. Για παράδειγμα, αν η συμβολοσειρά είναι αναπαράσταση μιας κεραίας, τότε τα σχήματα μπορεί να αντιπροσωπεύουν συνιστώσες της κεραίας, όπως ανακλαστήρες και εκτροπείς. Μια καλή συνιστώσα είναι δυνατό να είναι καλή για μια ποικιλία διαφορετικών σχεδιάσεων. Αυτό σημαίνει ότι η επιτυχημένη χρήση των γενετικών αλγορίθμων απαιτεί προσεκτική σχεδίαση της αναπαράστασης. Στην πράξη, οι γενετικοί αλγόριθμοι είχαν ευρεία επίδραση στα προβλήματα βελτιστοποίησης, όπως η διάταξη ηλεκτρονικών κυκλωμάτων και ο χρονοπρογραμμα-

ΕΞΕΛΙΞΗ ΚΑΙ ΑΝΑΖΗΤΗΣΗ

Η θεωρία της **εξέλιξης** (evolution) αναπτύχθηκε στο σύγγραμμα του Charles Darwin *On the Origin of Species by Means of Natural Selection* (*Σχετικά με την προέλευση των ειδών μέσω της φυσικής επιλογής* — 1859). Η κεντρική ιδέα είναι απλή: Παραλλαγές (γνωστές ως **μεταλλάξεις** — mutations) εμφανίζονται κατά την αναπαραγωγή και διατηρούνται στις επόμενες γενιές περίπου ανάλογα με τις επιπτώσεις τους στην αναπαραγωγική καταλληλότητα.

Η θεωρία του Darwin αναπτύχθηκε χωρίς καμία γνώση του πώς μπορούν να κληρονομούνται και να τροποποιούνται τα γνωρίσματα των οργανισμών. Οι πιθανοτικοί νόμοι που διέπουν αυτές τις διαδικασίες προσδιορίστηκαν για πρώτη φορά από τον Gregor Mendel (1866), ένα μοναχό που πειραματίστηκε με μπιζέλια χρησιμοποιώντας μια μέθοδο που ονόμαζε τεχνητή γονιμοποίηση. Πολύ αργότερα, οι Watson και Crick (1953) προσδιόρισαν τη δομή του μορίου DNA και το αλφάβητό του, AGTC (αδενίνη, γουανίνη, θυμίνη, κυτοζίνη). Σύμφωνα με το καθιερωμένο μοντέλο, η διαφοροποίηση γίνεται τόσο με σημειακές μεταλλάξεις στην ακολουθία των γραμμάτων όσο και με "διασταύρωση" (όπου το DNA ενός απογόνου παράγεται με το συνδυασμό μεγάλων ενοτήτων DNA από τον κάθε γονέα).

Έχουμε ήδη περιγράψει την αναλογία με τους αλγόριθμους τοπικής αναζήτησης· η κύρια διαφορά μεταξύ της στοχαστικής ακτινικής αναζήτησης και της εξέλιξης είναι η χρήση *γενετήσιας* (*sexual*) αναπαραγωγής, κατά την οποία οι διάδοχοι παράγονται από *πολλούς* οργανισμούς, και όχι μόνο από έναν. Οι πραγματικοί μηχανισμοί της εξέλιξης είναι όμως πολύ πλουσιότεροι από ό,τι επιτρέπουν οι περισσότεροι γενετικοί αλγόριθμοι. Για παράδειγμα, οι μεταλλάξεις μπορεί να περιλαμβάνουν αναστροφές, αντιγραφές και μετακινήσεις μεγάλων τμημάτων του DNA· μερικοί ιοί δανείζονται DNA από έναν οργανισμό και το εισάγουν σε έναν άλλο· υπάρχουν επίσης μεταθέσιμα γονίδια που δεν κάνουν τίποτα άλλο από το να αντιγράφονται πολλές χιλιάδες φορές μέσα στο γονιδίωμα. Υπάρχουν ακόμα γονίδια που δηλητηριάζουν τα κύτταρα των υποψηφίων για ζευγάρωμα που δεν έχουν το ίδιο γονίδιο, αυξάνοντας έτσι τις πιθανότητές τους για αντιγραφή. Κάτι πολύ σημαντικό είναι ότι *τα ίδια τα γονίδια περιέχουν κωδικοποιημένους τους μηχανισμούς* με τους οποίους το γονιδίωμα αναπαράγεται και μεταφράζεται σε έναν οργανισμό. Στους γενετικούς αλγόριθμους, οι μηχανισμοί αυτοί είναι ένα ξεχωριστό πρόγραμμα που δεν υπάρχει αναπαράστασή του στις συμβολοσειρές στις οποίες γίνονται οι χειρισμοί.

Η δαρβίνεια εξέλιξη θα μπορούσε να θεωρηθεί μη αποδοτικός μηχανισμός, αφού έχει παραγάγει τυφλά κάπου 10^{45} οργανισμούς χωρίς να βελτιώσει τους ευρετικούς μηχανισμούς αναζήτησής του ούτε στο ελάχιστο. Πενήντα χρόνια πριν από τον Darwin, όμως, ο κατά τα άλλα μεγάλος Γάλλος φυσιοδίφης Jean Lamarck (1809) πρότεινε μια θεωρία της εξέλιξης σύμφωνα με την οποία τα γνωρίσματα *που αποκτώνται με προσαρμογή κατά το χρόνο ζωής ενός οργανισμού* μεταβιβάζονται στους απογόνους του. Μια τέτοια διαδικασία θα ήταν αποδοτική, αλλά δε φαίνεται να συμβαίνει στη φύση. Πολύ αργότερα, ο James Baldwin (1896) πρότεινε μια φαινομενικά παρόμοια θεωρία σύμφωνα με την οποία η συμπεριφορά που μαθαίνεται κατά το χρόνο ζωής ενός οργανισμού μπορεί να επιταχύνει το ρυθμό της εξέλιξης. Σε αντίθεση με τη θεωρία του Lamarck, η θεωρία του Baldwin είναι εντελώς συνεπής με τη δαρβίνεια εξέλιξη, επειδή βασίζεται σε τάσεις επιλογής που επενεργούν σε άτομα τα οποία έχουν βρει τοπικά βέλτιστα μεταξύ του συνόλου των δυνατών συμπεριφορών που επιτρέπονται από τη γενετική τους συγκρότηση. Σύγχρονες προσομοιώσεις σε υπολογιστή επιβεβαιώνουν ότι το "φαινόμενο Baldwin" είναι πραγματικό, με την προϋπόθεση ότι η "κανονική" εξέλιξη μπορεί να δημιουργεί οργανισμούς που το εσωτερικό τους μέτρο απόδοσης σχετίζεται κατά κάποιον τρόπο με την πραγματική καταλληλότητα.

τισμός εργασιών παραγωγής. Σήμερα, δεν έχει γίνει ακόμα σαφές αν η γοητεία των γενετικών αλγορίθμων οφείλεται στην απόδοσή τους ή σε αισθητικούς λόγους εξαιτίας της καταγωγής τους από τη θεωρία της εξέλιξης. Πολλή δουλειά απομένει να γίνει ακόμα για να προσδιοριστούν οι συνθήκες υπό τις οποίες οι γενετικοί αλγόριθμοι αποδίδουν καλά.

4.4 ΤΟΠΙΚΗ ΑΝΑΖΗΤΗΣΗ ΣΕ ΣΥΝΕΧΕΙΣ ΧΩΡΟΥΣ

Στο Κεφάλαιο 2 εξηγήσαμε τη διαφορά μεταξύ διακριτών και συνεχών περιβαλλόντων, επισημαίνοντας ότι τα περισσότερα περιβάλλοντα του πραγματικού κόσμου είναι συνεχή. Ωστόσο, κανένας από τους αλγόριθμους που περιγράψαμε δεν μπορεί να χειριστεί συνεχείς χώρους καταστάσεων — η συνάρτηση διαδόχων στις περισσότερες περιπτώσεις θα επέστρεφε άπειρο αριθμό καταστάσεων! Αυτή η ενότητα κάνει μια *πολύ συνοπτική* παρουσίαση μερικών τεχνικών τοπικής αναζήτησης για την εύρεση βέλτιστων λύσεων σε συνεχείς χώρους. Η σχετική βιβλιογραφία είναι τεράστια· πολλές από τις βασικές τεχνικές έχουν τις ρίζες τους στο 17ο αιώνα, μετά την επινόηση του μαθηματικού λογισμού από τον Newton και τον Leibniz.[13] Θα βρούμε χρήσεις γι' αυτές τις τεχνικές σε πολλά σημεία του βιβλίου, μεταξύ των οποίων τα κεφάλαια για τη μάθηση, την όραση και τη ρομποτική· με μία φράση, σε οτιδήποτε έχει να κάνει με τον πραγματικό κόσμο.

Ας ξεκινήσουμε με ένα παράδειγμα. Έστω ότι θέλουμε να τοποθετήσουμε τρία νέα αεροδρόμια κάπου στη Ρουμανία, έτσι ώστε το άθροισμα των τετραγώνων των αποστάσεων από κάθε πόλη του χάρτη (Εικόνα 3.2) μέχρι το κοντινότερό της αεροδρόμιο να ελαχιστοποιηθεί. Τότε ο χώρος καταστάσεων ορίζεται από τις συντεταγμένες των αεροδρομίων: (x_1, y_1), (x_2, y_2) και (x_3, y_3). Είναι ένας χώρος *έξι διαστάσεων*· λέμε επίσης ότι οι καταστάσεις ορίζονται από έξι **μεταβλητές**. (Γενικά, οι καταστάσεις ορίζονται από ένα διάνυσμα n διαστάσεων, έστω $\mathbf{x}$.) Η μετακίνηση μέσα στο χώρο αυτό αντιστοιχεί στη μετακίνηση ενός ή περισσότερων αεροδρομίων πάνω στο χάρτη. Η αντικειμενική συνάρτηση $f(x_1, y_1, x_2, y_2, x_3, y_3)$ είναι σχετικά εύκολο να υπολογιστεί για οποιαδήποτε συγκεκριμένη κατάσταση αφού υπολογιστούν οι κοντινότερες πόλεις, αλλά είναι κάπως δύσκολο να διατυπωθεί γενικά.

Ένας τρόπος για να αποφεύγουμε τα προβλήματα της συνέχειας είναι απλώς να **διακριτοποιούμε** (discretize) τη γειτονιά της κάθε κατάστασης. Για παράδειγμα, μπορούμε να μετακινούμε μόνο ένα αεροδρόμιο τη φορά είτε στη διεύθυνση x είτε στην y κατά μια σταθερή ποσότητα $\pm\delta$. Με 6 μεταβλητές, αυτό μας δίνει 12 διαδόχους για κάθε κατάσταση. Έπειτα μπορούμε να εφαρμόσουμε οποιονδήποτε από τους αλγόριθμους τοπικής αναζήτησης που περιγράψαμε προηγουμένως. Μπορεί επίσης να χρησιμοποιήσει κανείς απευθείας τη στοχαστική αναρρίχηση λόφων και την προσομοιωμένη ανόπτηση, χωρίς να διακριτοποιήσει το χώρο. Αυτοί οι αλγόριθμοι επιλέγουν διαδόχους τυχαία, και αυτό μπορεί να γίνει με παραγωγή τυχαίων διανυσμάτων με μήκος δ.

ΚΛΙΣΗ

Υπάρχουν πολλές μέθοδοι που προσπαθούν να χρησιμοποιήσουν την **κλίση** (gradient) του τοπίου για να βρουν ένα μέγιστο. Κλίση της αντικειμενικής συνάρτησης είναι ένα διάνυσμα ∇f που μας δίνει το μέτρο και τη διεύθυνση του πιο απότομου επικλινούς. Στο πρόβλημά μας, έχουμε:

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial y_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial y_2}, \frac{\partial f}{\partial x_3}, \frac{\partial f}{\partial y_3} \right)$$

[13] Μια βασική γνώση του λογισμού πολλών μεταβλητών και της αριθμητικής διανυσμάτων είναι χρήσιμη για να διαβάσει κανείς αυτή την ενότητα.

Σε μερικές περιπτώσεις, μπορούμε να βρούμε ένα μέγιστο επιλύοντας την εξίσωση $\nabla f = 0$. (Αυτό θα μπορούσε να γίνει, για παράδειγμα, αν είχαμε να τοποθετήσουμε μόνο ένα αεροδρόμιο· η λύση είναι ο αριθμητικός μέσος όλων των συντεταγμένων των πόλεων.) Σε πολλές περιπτώσεις όμως, η εξίσωση αυτή δεν μπορεί να επιλυθεί σε κλειστή μορφή. Για παράδειγμα, με τρία αεροδρόμια η παράσταση της κλίσης εξαρτάται από το ποιες πόλεις είναι οι πιο κοντινές στο κάθε αεροδρόμιο στην τρέχουσα κατάσταση. Αυτό σημαίνει ότι μπορούμε να υπολογίζουμε την κλίση *τοπικά* αλλά όχι *καθολικά*. Ακόμα και έτσι, μπορούμε να χρησιμοποιήσουμε αναρρίχηση λόφων με την πλέον απότομη ανάβαση (steepest-ascent hill climbing), ενημερώνοντας την τρέχουσα κατάσταση με τον τύπο

$$\mathbf{x} \leftarrow \mathbf{x} + \alpha \nabla \mathrm{f}(\mathbf{x})$$

όπου α είναι μια μικρή σταθερά. Σε άλλες περιπτώσεις, η αντικειμενική συνάρτηση μπορεί να μην είναι καν διαθέσιμη σε διαφορίσιμη μορφή — για παράδειγμα, η τιμή ενός συγκεκριμένου συνόλου θέσεων αεροδρομίων μπορεί να προσδιορίζεται με την εκτέλεση κάποιου πακέτου οικονομικής προσομοίωσης μεγάλης κλίμακας. Σε τέτοιες περιπτώσεις, μπορεί να προσδιοριστεί η λεγόμενο **εμπειρική κλίση** (empirical gradient) από την αξιολόγηση της απόκρισης σε μικρές αυξήσεις και μειώσεις της κάθε συντεταγμένης. Η αναζήτηση με εμπειρική κλίση είναι ίδια με την αναρρίχηση λόφων με την πλέον απότομη ανάβαση σε μια διακριτοποιημένη έκδοση του χώρου καταστάσεων.

ΕΜΠΕΙΡΙΚΗ ΚΛΙΣΗ

Πίσω από τη φράση "το α είναι μια μικρή σταθερά" κρύβεται μια τεράστια ποικιλία μεθόδων προσαρμογής του α. Το βασικό πρόβλημα είναι ότι αν το α είναι πολύ μικρό χρειάζονται πάρα πολλά βήματα· αν το α είναι πολύ μεγάλο η αναζήτηση θα μπορούσε να προσπεράσει το μέγιστο. Η τεχνική της **ευθύγραμμης αναζήτησης** (line search) προσπαθεί να ξεπεράσει αυτό το δίλημμα επεκτείνοντας την τρέχουσα διεύθυνση κλίσης — συνήθως με επανειλημμένο διπλασιασμό του α — μέχρι η f να αρχίσει να μειώνεται ξανά. Το σημείο όπου συμβαίνει αυτό γίνεται η νέα τρέχουσα κατάσταση. Υπάρχουν πολλές σχολές σκέψης για το πώς θα πρέπει να επιλεχθεί η νέα διεύθυνση στο σημείο αυτό.

ΕΥΘΥΓΡΑΜΜΗ ΑΝΑΖΗΤΗΣΗ

Για πολλά προβλήματα, ο πιο αποτελεσματικός αλγόριθμος είναι η αξιοσέβαστη μέθοδος **Newton–Raphson** (Newton, 1664· Raphson, 1690). Πρόκειται για μια γενική τεχνική για την εύρεση ριζών συναρτήσεων — δηλαδή, για την επίλυση εξισώσεων της μορφής $g(x) = 0$. Ο αλγόριθμος υπολογίζει μια νέα εκτιμώμενη τιμή για τη ρίζα x σύμφωνα με τον τύπο του Newton:

NEWTON–RAPHSON

$$x \leftarrow x - g(x) / g'(x)$$

Για να βρούμε ένα μέγιστο ή ελάχιστο της f, χρειάζεται να βρούμε ένα $\mathbf{x}$ τέτοιο ώστε η *κλίση* να είναι μηδέν (δηλαδή, $\nabla f(\mathbf{x}) = \mathbf{0}$). Έτσι, η $g(x)$ στον τύπο του Newton γίνεται $\nabla f(\mathbf{x})$, και η εξίσωση ενημέρωσης μπορεί να γραφεί σε μορφή μητρώων — διανυσμάτων ως

$$\mathbf{x} \leftarrow \mathbf{x} - \mathbf{H}_f^{-1}(\mathbf{x})\, \nabla f(\mathbf{x})$$

όπου $\mathbf{H}_f(\mathbf{x})$ είναι το **εσσιανό** (Hessian) μητρώο των δεύτερων παραγώγων, που τα στοιχεία του, H_{ij}, προκύπτουν από την $\partial^2 f / \partial x_i \partial x_j$. Επειδή το εσσιανό μητρώο έχει n^2 καταχωρήσεις, ο αλγόριθμος Newton–Raphson γίνεται πολύ δαπανηρός σε χώρους με πολλές διαστάσεις, γι' αυτό έχουν επινοηθεί πολλές προσεγγιστικές μέθοδοι.

ΕΣΣΙΑΝΟ ΜΗΤΡΩΟ

Οι τοπικές μέθοδοι αναζήτησης πάσχουν από τα προβλήματα των τοπικών μεγίστων, των κορυφογραμμών και των οροπεδίων στους συνεχείς χώρους καταστάσεων όπως ακριβώς και στους διακριτούς χώρους. Οι τυχαίες επανεκκινήσεις και η προσομοιωμένη ανόπτηση μπορούν να χρησιμοποιηθούν και εδώ, και συχνά βοηθούν. Όμως, οι συνεχείς χώροι πολλών διαστάσεων είναι πολύ μεγάλοι και μπορεί εύκολα να χαθεί κανείς.

ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΜΕ ΠΕΡΙΟΡΙΣΜΟΥΣ

Ένα τελευταίο θέμα με το οποίο είναι χρήσιμη μια σύντομη γνωριμία είναι η **βελτιστοποίηση με περιορισμούς** (constrained optimization). Πρόβλημα βελτιστοποίησης με περιορισμούς είναι ένα πρόβλημα που οι λύσεις του πρέπει να ικανοποιούν κάποιους αυστηρούς περιορισμούς για τις τιμές της κάθε μεταβλητής. Για παράδειγμα, στο πρόβλημα της θέσης του αεροδρομίου, θα μπορούσαμε να περιορίσουμε τις τοποθεσίες στο εσωτερικό της Ρουμανίας και σε στερεό έδαφος (όχι μέσα σε λίμνες). Η δυσκολία των προβλημάτων βελτιστοποίησης με περιορισμούς εξαρτάται από τη φύση των περιορισμών και από την αντικειμενική συνάρτηση.

ΓΡΑΜΜΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Η πιο γνωστή κατηγορία είναι τα προβλήματα **γραμμικού προγραμματισμού** (linear programming), όπου οι περιορισμοί πρέπει να είναι γραμμικές ανισότητες που σχηματίζουν μια *κυρτή* περιοχή και η αντικειμενική συνάρτηση να είναι επίσης γραμμική. Τα γραμμικά προβλήματα μπορούν να επιλύονται σε χρόνο πολυωνυμικό ως προς τον αριθμό των μεταβλητών. Έχουν επίσης μελετηθεί προβλήματα με διαφορετικούς τύπους περιορισμών και αντικειμενικών συναρτήσεων — ο τετραγωνικός προγραμματισμός (quadratic programming), ο κωνικός προγραμματισμός δεύτερης τάξης κ.ο.κ.

4.5 Πρακτορες Online Αναζητησης και Αγνωστα Περιβαλλοντα

OFFLINE ΑΝΑΖΗΤΗΣΗ

Μέχρι εδώ, εστιάσαμε την προσοχή μας σε πράκτορες που χρησιμοποιούν αλγόριθμους **offline αναζήτησης** (offline search). Οι πράκτορες αυτοί υπολογίζουν πλήρως μια λύση πριν πατήσουν το πόδι τους στον πραγματικό κόσμο (δείτε στην Εικόνα 3.1) και μετά εκτελούν τη λύση χωρίς να προσφεύγουν στις αντιλήψεις τους. Αντίθετα, ένας πράκτορας **online αναζήτησης** (online search)[14] λειτουργεί με **διαπλοκή** (interleaving) υπολογισμών και ενεργειών· κάνει πρώτα μια ενέργεια και έπειτα παρατηρεί το περιβάλλον και υπολογίζει την επόμενη ενέργεια. Η online αναζήτηση είναι καλή ιδέα για τα δυναμικά ή ημιδυναμικά πεδία εφαρμογών — τα πεδία όπου υπάρχει ποινή αν ο πράκτορας κάθεται και κάνει υπολογισμούς για πολύ χρόνο. Η online αναζήτηση είναι ακόμα καλύτερη ιδέα για τα στοχαστικά πεδία εφαρμογών. Γενικά, μια offline αναζήτηση θα πρέπει να καταστρώνει ένα εκθετικά μεγάλο πλάνο ενδεχομένων που να παίρνει υπόψη όλα όσα είναι δυνατόν να συμβούν, ενώ μια online αναζήτηση χρειάζεται να παίρνει υπόψη μόνο όσα συμβαίνουν πραγματικά. Για παράδειγμα, ένας πράκτορας που παίζει σκάκι θα έκανε καλά να παίξει την πρώτη του κίνηση πολύ πριν προβλέψει όλη την πορεία του παιχνιδιού.

ONLINE ΑΝΑΖΗΤΗΣΗ

ΠΡΟΒΛΗΜΑ ΕΞΕΡΕΥΝΗΣΗΣ

Η online αναζήτηση είναι *απαραίτητη* για τα **προβλήματα εξερεύνησης** (exploration problems), όπου οι καταστάσεις και οι ενέργειες είναι άγνωστες στον πράκτορα. Ένας πράκτορας σε τέτοια κατάσταση άγνοιας πρέπει να χρησιμοποιεί τις ενέργειές του ως πειράματα για να προσδιορίζει τι θα κάνει στη συνέχεια, και επομένως πρέπει να εναλλάσσει υπολογισμούς και ενέργειες.

Το παραδοσιακό παράδειγμα της online αναζήτησης είναι ένα ρομπότ που τοποθετείται σε ένα νέο κτίριο και πρέπει να το εξερευνήσει για να δημιουργήσει ένα χάρτη που να μπορεί να τον χρησιμοποιήσει για να πάει από το σημείο A στο B. Οι μέθοδοι διαφυγής από λαβύρινθους — απαραίτητη γνώση για τους επίδοξους ήρωες της αρχαιότητας — είναι επίσης παραδείγματα αλγορίθμων online αναζήτησης. Η χωρική εξερεύνηση δεν είναι όμως η μόνη μορφή εξερεύνησης. Σκεφτείτε ένα νεογέννητο μωρό· έχει στη διάθεσή του πολλές δυνατές ενέργειες αλλά δε γνωρίζει τα αποτελέσματα καμίας, και οι εμπειρίες του περιορίζονται σε μερικές καταστάσεις στις οποίες μπορεί να φτάσει. Η βαθμιαία ανακάλυψη από το μωρό τού πώς λειτουργεί ο κόσμος είναι, κατά ένα μέρος, μια διαδικασία online αναζήτησης.

[14] Ο όρος "online" χρησιμοποιείται συνήθως στην επιστήμη των υπολογιστών για να αναφερόμαστε σε αλγόριθμους οι οποίοι πρέπει να επεξεργάζονται τα δεδομένα εισόδου καθώς λαμβάνονται, και όχι να περιμένουν να γίνει διαθέσιμο ολόκληρο το σύνολο των δεδομένων εισόδου.

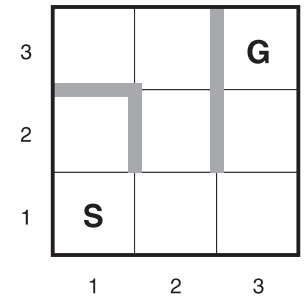
Προβλήματα online αναζήτησης

Ένα πρόβλημα online αναζήτησης μπορεί να επιλυθεί μόνο από έναν πράκτορα που εκτελεί ενέργειες και όχι από μια καθαρά υπολογιστική διαδικασία. Θα θεωρήσουμε ότι ο πράκτορας γνωρίζει τα παρακάτω:

- Τη συνάρτηση ACTIONS(s), η οποία επιστρέφει μια λίστα ενεργειών που είναι επιτρεπτές στην κατάσταση s
- Τη συνάρτηση κόστους βήματος $c(s, a, s')$ — σημειώστε ότι αυτή δεν μπορεί να χρησιμοποιηθεί μέχρι ο πράκτορας να γνωρίζει ότι το αποτέλεσμα είναι το s'
- Τη συνάρτηση ελέγχου στόχου GOAL-TEST(s)

Ιδιαίτερα, σημειώστε ότι ο πράκτορας *δεν μπορεί* να προσπελάσει τους διαδόχους μιας κατάστασης παρά μόνο επιχειρώντας πραγματικά όλες τις ενέργειες από την κατάσταση αυτή. Για παράδειγμα, στο πρόβλημα του λαβυρίνθου που παρουσιάζεται στην Εικόνα 4.18, ο πράκτορας δε γνωρίζει ότι η μετάβαση *Επάνω* από το (1, 1) οδηγεί στο (1, 2)· ούτε γνωρίζει ότι, αφού κάνει αυτή την ενέργεια, η μετάβαση *Κάτω* θα τον οδηγήσει πίσω στο (1, 1). Αυτός ο βαθμός άγνοιας μπορεί σε μερικές εφαρμογές να είναι μειωμένος — για παράδειγμα, ένα ρομπότ-εξερευνητής μπορεί να γνωρίζει πώς λειτουργούν οι ενέργειες μετακίνησής του και να αγνοεί μόνο τις θέσεις των εμποδίων.

Θα θεωρήσουμε ότι ο πράκτορας μπορεί πάντα να αναγνωρίζει μια κατάσταση που έχει επισκεφτεί προηγουμένως, και θα θεωρήσουμε επίσης ότι οι ενέργειες είναι αιτιοκρατικές. (Αυτές οι δύο τελευταίες παραδοχές χαλαρώνουν στο Κεφάλαιο 17.) Τέλος, ο πράκτορας θα μπορούσε να έχει πρόσβαση σε μια παραδεκτή ευρετική συνάρτηση $h(s)$ που εκτιμά την απόσταση της τρέχουσας κατάστασης από μια κατάσταση στόχου. Για παράδειγμα, στην Εικόνα 4.18, ο πράκτορας θα μπορούσε να γνωρίζει τη θέση του στόχου και να είναι σε θέση να χρησιμοποιεί τον ευρετικό μηχανισμό της απόστασης Manhattan.



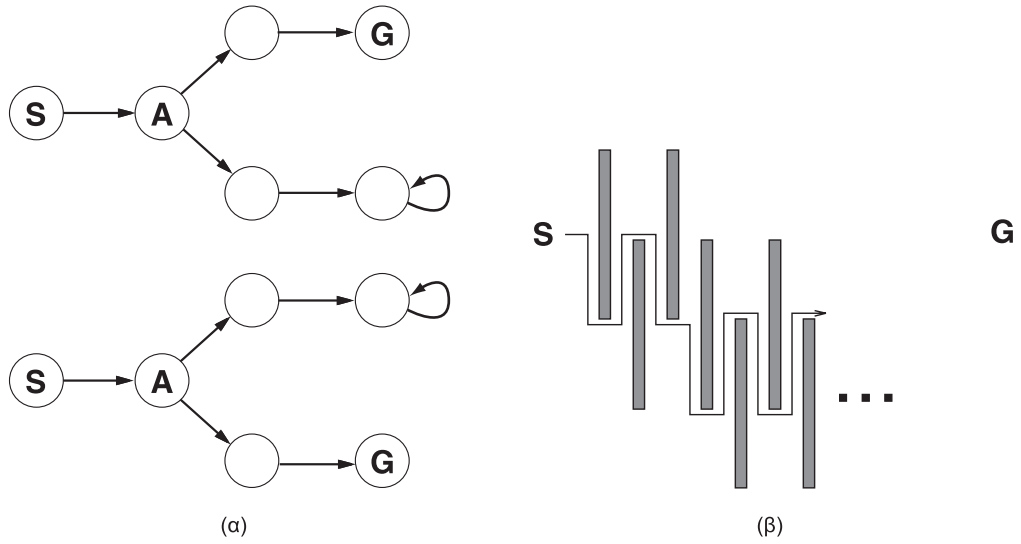
Εικόνα 4.18 Απλό πρόβλημα λαβυρίνθου. Ο πράκτορας ξεκινά από το S και πρέπει να φτάσει στο G, αλλά δε γνωρίζει τίποτα για το περιβάλλον.

Στην συνήθη περίπτωση, ο σκοπός του πράκτορα είναι να φτάσει σε μια κατάσταση στόχου ενώ ελαχιστοποιεί το κόστος. (Ένας άλλος ενδεχόμενος σκοπός είναι απλώς να εξερευνήσει πλήρως το περιβάλλον.) Το κόστος είναι το ολικό κόστος της διαδρομής που διανύει πραγματικά ο πράκτορας. Συνηθίζεται να συγκρίνεται αυτό το κόστος με το κόστος της διαδρομής που θα ακολουθούσε ο πράκτορας *αν γνώριζε το χώρο αναζήτησης από πριν* — δηλαδή, την πραγματική συντομότερη διαδρομή (τη συντομότερη πλήρη εξερεύνηση). Στη γλώσσα των online αλγορίθμων, αυτό ονομάζεται **ανταγωνιστικός λόγος** (competitive ratio)· θα θέλαμε αυτός ο λόγος να είναι όσο το δυνατόν μικρότερος.

ΑΝΤΑΓΩΝΙΣΤΙΚΟΣ ΛΟΓΟΣ

Αν και ο μικρός ανταγωνιστικός λόγος φαίνεται λογικό αίτημα, είναι εύκολο να δούμε ότι σε μερικές περιπτώσεις ο καλύτερος εφικτός ανταγωνιστικός λόγος είναι άπειρος. Για παράδειγμα, αν μερικές ενέργειες είναι μη αναστρέψιμες, η online αναζήτηση μπορεί κατά τύχη να φτάσει σε μια αδιέξοδη κατάσταση από την οποία δεν είναι προσπελάσιμη καμία κατάσταση στόχου. Ίσως ο όρος "κατά τύχη" να μη σας φαίνεται πειστικός — στο κάτω-κάτω, μπορεί να υπάρχει ένας αλγόριθμος που συμβαίνει να μην παίρνει την αδιέξοδη διαδρομή καθώς εξερευνά. Για να είμαστε πιο ακριβείς, ο ισχυρισμός μας είναι ότι *κανένας αλγόριθμος δεν μπορεί να αποφεύγει τα αδιέξοδα σε όλους τους χώρους καταστάσεων*. Ας εξετάσουμε τους δύο αδιέξοδους χώρους καταστάσεων της Εικόνας 4.19(α). Για έναν αλγόριθμο online αναζήτησης που έχει επισκεφτεί τις καταστάσεις *S* και *A*, οι δύο χώροι καταστάσεων φαίνονται *ίδιοι*, και επομένως θα πρέπει να πάρει την ίδια απόφαση και στους δύο. Έτσι, θα αποτύχει στον ένα από αυτούς. Αυτό είναι ένα παράδειγμα του **επιχειρήματος του αντιπάλου** (adversary argument) — μπορούμε να φανταστούμε έναν αντίπαλο που κατασκευάζει το χώρο καταστάσεων ενώ ο πράκτορας τον εξερευνά, και μπορεί να τοποθετεί τους στόχους και τα αδιέξοδα όπου θέλει.

ΕΠΙΧΕΙΡΗΜΑ ΤΟΥ ΑΝΤΙΠΑΛΟΥ



Εικόνα 4.19 (α) Δύο χώροι καταστάσεων που θα μπορούσαν να οδηγήσουν έναν πράκτορα online αναζήτησης σε αδιέξοδο. Οποιοσδήποτε δεδομένος πράκτορας θα αποτύχει τουλάχιστον σε έναν από αυτούς τους χώρους. (β) Ένα διδιάστατο περιβάλλον που μπορεί να κάνει έναν πράκτορα online αναζήτησης να ακολουθήσει ένα οσοδήποτε μη αποδοτικό δρομολόγιο προς τον στόχο. Όποια επιλογή και να κάνει ο πράκτορας, ο αντίπαλος μπλοκάρει αυτό το δρομολόγιο με άλλον ένα μακρύ και λεπτό τοίχο, ώστε η διαδρομή που ακολουθείται να είναι πολύ μεγαλύτερη από την καλύτερη δυνατή διαδρομή.

Τα αδιέξοδα είναι μια πραγματική δυσκολία για τη ρομποτική εξερεύνηση — οι σκάλες, οι κεκλιμένοι διάδρομοι, οι γκρεμοί και η κάθε είδους φυσική διαμόρφωση του εδάφους παρέχουν ευκαιρίες για μη αναστρέψιμες ενέργειες. Για να μπορέσουμε να προχωρήσουμε, θα θεωρήσουμε απλώς ότι ο χώρος καταστάσεων είναι **ασφαλώς εξερευνήσιμος** (safely explorable) — δηλαδή, ότι από κάθε προσπελάσιμη κατάσταση είναι προσπελάσιμη κάποια κατάσταση στόχου. Οι χώροι καταστάσεων με αναστρέψιμες ενέργειες, όπως οι λαβύρινθοι και τα παζλ των 8 πλακιδίων, μπορούν να θεωρηθούν μη προσανατολισμένα γραφήματα και είναι οπωσδήποτε ασφαλώς εξερευνήσιμοι.

ΑΣΦΑΛΩΣ ΕΞΕΡΕΥΝΗΣΙΜΟΣ

Ακόμα και σε ασφαλώς εξερευνήσιμα περιβάλλοντα, τίποτα δε μας εγγυάται ότι ο ανταγωνιστικός λόγος είναι φραγμένος αν υπάρχουν διαδρομές με μη φραγμένο κόστος. Αυτό είναι εύκολο να αποδειχτεί για τα περιβάλλοντα με μη αναστρέψιμες ενέργειες, αλλά παραμένει αληθές και στην περίπτωση των αναστρέψιμων, όπως δείχνει η Εικόνα 4.19(β). Γι' αυτόν το λόγο, η απόδοση των αλγορίθμων online αναζήτησης συνηθίζεται να περιγράφεται με βάση το μέγεθος ολόκληρου του χώρου καταστάσεων, και όχι απλώς με βάση το βάθος του ρηχότερου στόχου.

Πράκτορες online αναζήτησης

Έπειτα από κάθε ενέργεια, ένας πράκτορας online προσλαμβάνει μια αντίληψη που του λέει σε ποια κατάσταση έχει φτάσει· από αυτή την πληροφορία, μπορεί να βελτιώσει το χάρτη του περιβάλλοντος που έχει. Ο τρέχων χάρτης χρησιμοποιείται για να αποφασίσει ο πράκτορας πού θα πάει στη συνέχεια. Αυτή η διαπλοκή σχεδιασμού και δράσης σημαίνει ότι οι αλγόριθμοι online αναζήτησης είναι πολύ διαφορετικοί από τους αλγόριθμους offline αναζήτησης που είδαμε προηγουμένως. Για παράδειγμα, οι offline αλγόριθμοι όπως ο A* έχουν την ικανότητα να επεκτείνουν έναν κόμβο σε ένα μέρος του χώρου και αμέσως μετά να επεκτείνουν έναν κόμβο σε ένα άλλο μέρος του χώρου, επειδή η επέκταση των κόμβων αποτελείται από προσομοιωμένες και όχι πραγματικές ενέργειες. Ένας online αλγόριθμος, από την άλλη, μπορεί να επεκτείνει μόνο έναν κόμβο στον οποίο είναι φυσικά παρών. Για να αποφύγει ο πράκτορας να διασχίζει ολόκληρο το δένδρο για να επεκτείνει τον επόμενο κόμβο, φαίνεται καλύτερο να επεκτείνει τους κόμβους με *τοπική* προτεραιότητα. Η αναζήτηση πρώτα κατά βάθος έχει ακριβώς αυτή την ιδιότητα, επειδή (με εξαίρεση την υπαναχώρηση) ο επόμενος κόμβος που επεκτείνεται είναι θυγατρικός του προηγούμενου κόμβου που επεκτάθηκε.

```
function ONLINE-DFS-AGENT( s′ ) returns μια ενέργεια
    inputs: s′, μια αντίληψη που προσδιορίζει την τρέχουσα κατάσταση
    static: αποτέλεσμα, πίνακας ευρετηριασμένος κατά ενέργεια και κατάσταση, αρχικά κενός
            ανεξερεύνητα, πίνακας που για κάθε εξετασθείσα κατάσταση περιέχει τις ενέργειες
                          που δεν έχουν δοκιμαστεί
            υπαναχωρήσεις, πίνακας που για κάθε εξετασθείσα κατάσταση περιέχει τις
                          υπαναχωρήσεις που δεν έχουν δοκιμαστεί
            s, a, η προηγούμενη κατάσταση και ενέργεια, αρχικά κενά

    if GOAL-TEST( s′ ) then return τέλος
    if s′ είναι νέα κατάσταση then ανεξερεύνητα[s′] ← ACTIONS( s′ )
    if s δεν είναι κενό then do
        αποτέλεσμα[a, s] ← s′
        πρόσθεσε το s στην αρχή του υπαναχωρήσεις[s′]
    if ανεξερεύνητα[s′] είναι κενό then
        if υπαναχωρήσεις[s′] είναι κενό then return τέλος
        else a ← μια ενέργεια b τέτοια ώστε αποτέλεσμα[b, s′] = POP( υπαναχωρήσεις[s′] )
    else a ← POP( ανεξερεύνητα[s′] )
    s ← s′
    return a
```

Εικόνα 4.20 Πράκτορας online αναζήτησης που χρησιμοποιεί εξερεύνηση πρώτα κατά βάθος. Ο πράκτορας αυτός είναι κατάλληλος μόνο για χώρους αμφίδρομης αναζήτησης.

Ένας πράκτορας online αναζήτησης πρώτα κατά βάθος παρουσιάζεται στην Εικόνα 4.20. Ο πράκτορας αυτός αποθηκεύει το χάρτη του σε έναν πίνακα της μορφής *αποτέλεσμα*[*a*, *s*], στον οποίο καταγράφεται η κατάσταση που προκύπτει από την εκτέλεση της ενέργειας *a* στην κατά-

σταση *s*. Οποτεδήποτε μια ενέργεια από την τρέχουσα κατάσταση δεν έχει εξερευνηθεί, ο πράκτορας την επιχειρεί. Η δυσκολία παρουσιάζεται όταν ο πράκτορας έχει επιχειρήσει όλες τις ενέργειες σε μια κατάσταση. Στην offline αναζήτηση πρώτα κατά βάθος, η κατάσταση αυτή απλώς αφαιρείται από την ουρά· σε μια online αναζήτηση ο πράκτορας πρέπει φυσικά να υπαναχωρήσει. Στην αναζήτηση πρώτα κατά βάθος, αυτό σημαίνει επιστροφή στην κατάσταση από την οποία ο πράκτορας εισήλθε πιο πρόσφατα στην τρέχουσα κατάσταση. Αυτό επιτυγχάνεται με την τήρηση ενός πίνακα που περιέχει, για κάθε κατάσταση, τις προηγούμενες καταστάσεις στις οποίες ο πράκτορας δεν έχει ακόμα υπαναχωρήσει. Αν οι καταστάσεις στις οποίες μπορεί να υπαναχωρήσει ο πράκτορας έχουν εξαντληθεί, τότε η αναζήτηση έχει τελειώσει.

Συνιστούμε να παρακολουθήσετε την πορεία του αλγόριθμου ONLINE-DFS-AGENT όπως εφαρμόζεται στο λαβύρινθο της Εικόνας 4.18. Είναι αρκετά εύκολο να δείτε ότι ο πράκτορας, στη χειρότερη περίπτωση, θα καταλήξει να διατρέξει όλους τους συνδέσμους του χώρου καταστάσεων ακριβώς δύο φορές. Για την εξερεύνηση, αυτό είναι βέλτιστο· για την επίτευξη του στόχου, από την άλλη, ο ανταγωνιστικός λόγος του πράκτορα θα μπορούσε να είναι απεριόριστα κακός αν ξεκινήσει για ένα μακρινό ταξίδι ενώ υπάρχει μια κατάσταση στόχου ακριβώς δίπλα στην αρχική κατάσταση. Μια online παραλλαγή της επαναληπτικής εκβάθυνσης επιλύει αυτό το πρόβλημα· για ένα περιβάλλον που είναι ομοιόμορφο δένδρο, ο ανταγωνιστικός λόγος ενός τέτοιου πράκτορα είναι μια μικρή σταθερά.

Λόγω της μεθόδου υπαναχώρησης που χρησιμοποιεί, ο αλγόριθμος ONLINE-DFS-AGENT είναι κατάλληλος μόνο για χώρους καταστάσεων όπου οι ενέργειες είναι αναστρέψιμες. Υπάρχουν κάπως πιο πολύπλοκοι αλγόριθμοι που είναι κατάλληλοι για γενικούς χώρους καταστάσεων, αλλά κανένας τέτοιος αλγόριθμος δεν έχει φραγμένο ανταγωνιστικό λόγο.

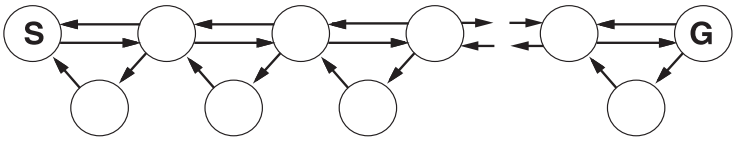
Τοπική online αναζήτηση

Όπως και η αναζήτηση πρώτα κατά βάθος, η **αναζήτηση με αναρρίχηση λόφων** (hill-climbing search) έχει την ιδιότητα της τοπικότητας στις επεκτάσεις των κόμβων της. Επειδή μάλιστα διατηρεί στη μνήμη μόνο μία τρέχουσα κατάσταση, η αναζήτηση με αναρρίχηση λόφων είναι *ήδη* ένας αλγόριθμος online αναζήτησης! Δυστυχώς, δεν είναι πολύ χρήσιμη στην αρχική της μορφή επειδή αφήνει τον πράκτορα να κάθεται στα τοπικά μέγιστα χωρίς να έχει πού να πάει. Επίσης, δεν μπορούν να χρησιμοποιούνται τυχαίες επανεκκινήσεις, επειδή ο πράκτορας δεν μπορεί να μεταφέρεται σε μια νέα κατάσταση.

ΤΥΧΑΙΟΣ ΠΕΡΙΠΑΤΟΣ

Αντί των τυχαίων επανεκκινήσεων, θα μπορούσε κανείς να σκεφτεί να χρησιμοποιήσει έναν **τυχαίο περίπατο** (random walk) για την εξερεύνηση του περιβάλλοντος. Ένας τυχαίος περίπατος απλώς επιλέγει τυχαία μία από τις ενέργειες που είναι διαθέσιμες από την τρέχουσα κατάσταση· μπορεί να δίνεται προτίμηση σε ενέργειες που δεν έχουν δοκιμαστεί ακόμα. Είναι εύκολο να αποδειχτεί ότι ένας τυχαίος περίπατος θα βρει *τελικά* μια κατάσταση στόχου ή θα ολοκληρώσει την εξερεύνηση, με την προϋπόθεση ότι ο χώρος είναι πεπερασμένος.[15] Από την άλλη, η διαδικασία μπορεί να είναι πολύ αργή. Η Εικόνα 4.21 δείχνει ένα περιβάλλον στο οποίο ο αριθμός των βημάτων που θα χρειαστεί ένας τυχαίος περίπατος για να βρει την κατάσταση στόχου αυξάνεται εκθετικά επειδή σε κάθε βήμα η πορεία προς τα πίσω είναι δύο φορές πιο πιθανή από την πορεία προς τα εμπρός. Το παράδειγμα είναι βέβαια τεχνητό, αλλά υπάρχουν πολλοί χώροι καταστάσεων του πραγματικού κόσμου που η τοπολογία τους προκαλεί τέτοιου είδους "παγίδες" για τους τυχαίους περιπάτους.

[15] Η περίπτωση του άπειρου χώρου καταστάσεων είναι πιο δύσκολη. Οι τυχαίοι περίπατοι είναι πλήρεις σε άπειρα μονοδιάστατα και διδιάστατα πλέγματα, όχι όμως σε τρισδιάστατα πλέγματα! Στην τελευταία περίπτωση, η πιθανότητα να επιστρέψει ποτέ ο περίπατος στο σημείο εκκίνησης είναι μόνο 0,3405 περίπου (δείτε Hughes, 1995, για μια γενική εισαγωγή).



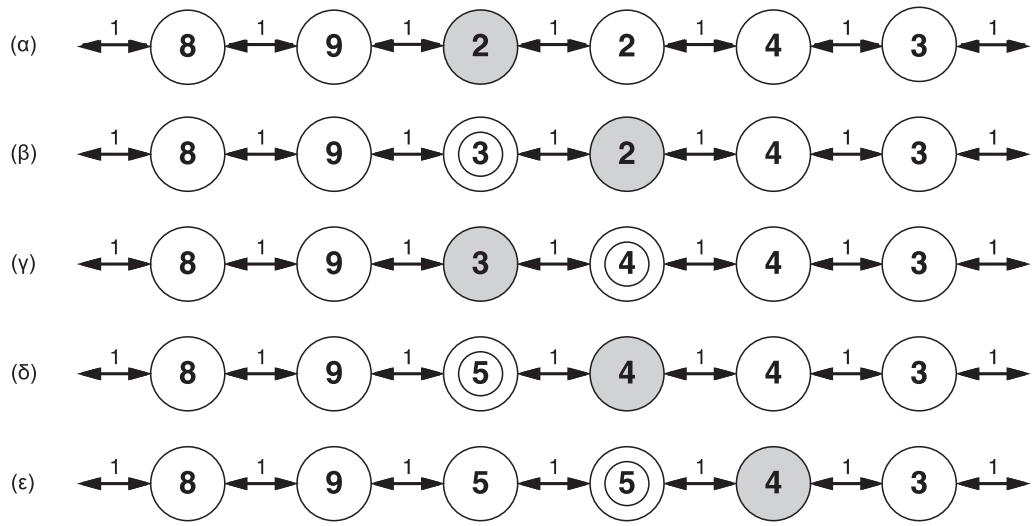
Εικόνα 4.21 Περιβάλλον στο οποίο ένας τυχαίος περίπατος θα χρειαστεί εκθετικά αυξανόμενο αριθμό βημάτων για να βρει την κατάσταση στόχου.

Η ενίσχυση της αναρρίχησης λόφων με *μνήμη* αντί με τυχαιότητα αποδεικνύεται ότι είναι πιο αποτελεσματική προσέγγιση. Η βασική ιδέα είναι να αποθηκεύεται μια "τρέχουσα καλύτερη εκτίμηση" $H(s)$ του κόστους μετάβασης στην κατάσταση στόχου από κάθε κατάσταση που έχουμε επισκεφτεί. Η $H(s)$ ξεκινά απλώς από την τιμή της ευρετικής εκτίμησης $h(s)$ και ενημερώνεται καθώς ο πράκτορας αποκτά πείρα στο χώρο καταστάσεων. Η Εικόνα 4.22 παρουσιάζει ένα απλό παράδειγμα σε ένα μονοδιάστατο χώρο καταστάσεων. Στο (α), ο πράκτορας φαίνεται να έχει παγιδευτεί σε ένα επίπεδο τοπικό ελάχιστο στη σκιασμένη κατάσταση. Αντί να μείνει εκεί που είναι, ο πράκτορας θα πρέπει να ακολουθήσει τη διαδρομή που φαίνεται καλύτερη για να φτάσει στην κατάσταση στόχου, με βάση τις τρέχουσες εκτιμήσεις κόστους για τις γειτονικές του καταστάσεις. Το εκτιμώμενο κόστος για να φτάσει στην κατάσταση στόχου μέσω μιας γειτονικής κατάστασης s' είναι το κόστος για να φτάσει στην s' συν το εκτιμώμενο κόστος για να φτάσει από εκεί στην κατάσταση στόχου — δηλαδή, $c(s, a, s') + H(s')$. Στο παράδειγμα υπάρχουν δύο ενέργειες, με εκτιμώμενα κόστη 1+9 και 1+2, και επομένως φαίνεται ότι το καλύτερο είναι να πάει δεξιά. Τώρα, είναι φανερό ότι η εκτίμηση κόστους 2 για τη σκιασμένη κατάσταση ήταν υπερβολικά αισιόδοξη. Αφού η καλύτερη κίνηση κόστισε 1 και οδήγησε σε μια κατάσταση που βρίσκεται τουλάχιστον 2 βήματα από μια κατάσταση στόχου, η σκιασμένη κατάσταση θα πρέπει να απέχει τουλάχιστον 3 βήματα από μια κατάσταση στόχου, και επομένως η εκτίμηση H της θα πρέπει να ενημερωθεί κατάλληλα, όπως φαίνεται στην Εικόνα 4.21(β). Συνεχίζοντας αυτή τη διαδικασία, ο πράκτορας θα μετακινηθεί εμπρός-πίσω δύο φορές ακόμα ενημερώνοντας την H κάθε φορά και "ισοπεδώνοντας" το τοπικό ελάχιστο, μέχρι να ξεφύγει προς τα δεξιά.

LRTA*

Ένας πράκτορας που υλοποιεί αυτόν το μηχανισμό, ο οποίος ονομάζεται **A* πραγματικού χρόνου που μαθαίνει** (learning real-time A* — LRTA*), παρουσιάζεται στην Εικόνα 4.23. Όπως και ο αλγόριθμος ONLINE-DFS-AGENT, κατασκευάζει ένα χάρτη του περιβάλλοντος χρησιμοποιώντας τον πίνακα *αποτέλεσμα*. Ενημερώνει την εκτίμηση κόστους για την κατάσταση από την οποία έχει μόλις φύγει και μετά διαλέγει τη "φαινομενικά καλύτερη" κίνηση σύμφωνα με τις τρέχουσες εκτιμήσεις κόστους του. Μια σημαντική λεπτομέρεια είναι ότι οι ενέργειες που δεν έχουν δοκιμαστεί ακόμα σε μια κατάσταση s θεωρείται πάντα ότι οδηγούν αμέσως στην κατάσταση στόχου με το ελάχιστο δυνατό κόστος, συγκεκριμένα $h(s)$. Αυτή η **αισιοδοξία υπό αβεβαιότητα** (optimism under uncertainty) ενθαρρύνει τον πράκτορα να εξερευνεί νέες και ίσως ελπιδοφόρες διαδρομές.

ΑΙΣΙΟΔΟΞΙΑ ΥΠΟ ΑΒΕΒΑΙΟΤΗΤΑ



Εικόνα 4.22 Πέντε επαναλήψεις του αλγόριθμου LRTA* σε ένα μονοδιάστατο χώρο καταστάσεων. Σε κάθε κατάσταση είναι σημειωμένη η τιμή $H(s)$, η τρέχουσα εκτίμηση κόστους για να φτάσουμε σε μια κατάσταση στόχου, και σε κάθε τόξο είναι σημειωμένο το αντίστοιχο κόστος βήματος. Η σκιασμένη κατάσταση δείχνει τη θέση του πράκτορα και οι ενημερωμένες τιμές σε κάθε επανάληψη είναι σε κύκλο.

```
function LRTA*-AGENT( s' ) returns μια ενέργεια
    inputs: s', μια αντίληψη που προσδιορίζει την τρέχουσα κατάσταση
    static: αποτέλεσμα, πίνακας ευρετηριασμένος κατά ενέργεια και κατάσταση, αρχικά κενός
            H, πίνακας εκτιμήσεων κόστους ευρετηριασμένος κατά κατάσταση, αρχικά κενός
            s, a, η προηγούμενη κατάσταση και ενέργεια, αρχικά κενές

    if GOAL-TEST( s' ) then return τέλος
    if s' είναι νέα κατάσταση (δεν υπάρχει στον H) then H[s'] ← h( s' )
    unless s είναι κενό
        αποτέλεσμα[a, s] ← s'
        H[s] ← min_{b ∈ ACTIONS(s)} LRTA*-COST( s, b, αποτέλεσμα[b, s], H )
    a ← μια ενέργεια b από τις ACTIONS( s' ) που ελαχιστοποιεί την LRTA*-COST( s', b,
                                                                  αποτέλεσμα[b, s'], H )
    s ← s'
    return a

function LRTA*-COST( s, a, s', H ) returns μια εκτίμηση κόστους
    if s' δεν έχει οριστεί then return h(s)
    else return c( s, a, s' ) + H[s']
```

Εικόνα 4.23 Ο αλγόριθμος LRTA*-AGENT επιλέγει μια ενέργεια σύμφωνα με τις τιμές των γειτονικών καταστάσεων, οι οποίες ενημερώνονται καθώς ο πράκτορας κινείται μέσα στο χώρο καταστάσεων.

Ένας πράκτορας LRTA* βρίσκει εγγυημένα μια κατάσταση στόχου σε οποιοδήποτε πεπερασμένο και ασφαλώς εξερευνήσιμο περιβάλλον. Αντίθετα από τον A*, όμως, δεν είναι πλήρης

για τους άπειρους χώρους καταστάσεων — υπάρχουν περιπτώσεις όπου μπορεί να παραπλανηθεί εντελώς. Μπορεί να εξερευνά ένα περιβάλλον n καταστάσεων σε $O(n^2)$ βήματα στη χειρότερη περίπτωση, αλλά συχνά τα καταφέρνει πολύ καλύτερα. Ο πράκτορας LRTA* είναι μόνο ένας από μια μεγάλη οικογένεια πρακτόρων online που μπορούν να οριστούν με καθορισμό του κανόνα επιλογής ενέργειας και του κανόνα ενημέρωσης με διαφορετικούς τρόπους. Θα εξετάσουμε αυτή την οικογένεια πρακτόρων, οι οποίοι σχεδιάστηκαν αρχικά για στοχαστικά περιβάλλοντα, στο Κεφάλαιο 21.

Μάθηση στην online αναζήτηση

Η αρχική άγνοια των πρακτόρων online αναζήτησης παρέχει πολλές ευκαιρίες για μάθηση. Πρώτον, οι πράκτορες μαθαίνουν ένα "χάρτη" του περιβάλλοντος — για την ακρίβεια, το αποτέλεσμα κάθε ενέργειας σε κάθε κατάσταση — καταγράφοντας απλώς κάθε εμπειρία τους. (Προσέξτε ότι η παραδοχή του αιτιοκρατικού περιβάλλοντος σημαίνει ότι μία εμπειρία είναι αρκετή για την κάθε ενέργεια.) Δεύτερον, οι πράκτορες τοπικής αναζήτησης αποκτούν πιο ακριβείς εκτιμήσεις για την αξία της κάθε κατάστασης χρησιμοποιώντας τοπικούς κανόνες ενημέρωσης, όπως στον αλγόριθμο LRTA*. Στο Κεφάλαιο 21 θα δούμε ότι αυτές οι ενημερώσεις τελικά συγκλίνουν σε *ακριβείς* τιμές για κάθε κατάσταση, με την προϋπόθεση ότι ο πράκτορας εξερευνεί το χώρο καταστάσεων σωστά. Από τη στιγμή που είναι γνωστές οι ακριβείς τιμές, μπορούν να παίρνονται βέλτιστες αποφάσεις με απλή μετακίνηση στο διάδοχο με τη μεγαλύτερη τιμή — δηλαδή, η αμιγής αναρρίχηση λόφων είναι τότε μια βέλτιστη στρατηγική.

Αν ακολουθήσατε την υπόδειξή μας να παρακολουθήσετε τη συμπεριφορά τού ONLINE-DFS-AGENT στο περιβάλλον της Εικόνας 4.18, θα παρατηρήσατε ότι ο πράκτορας δεν είναι και τόσο έξυπνος. Για παράδειγμα, αφού έχει δει ότι η ενέργεια *Επάνω* πηγαίνει από το (1, 1) στο (1, 2), ο πράκτορας εξακολουθεί να μην έχει ιδέα ότι η ενέργεια *Κάτω* πηγαίνει πίσω στο (1, 1), ή ότι η ενέργεια *Επάνω* πηγαίνει επίσης από το (2, 1) στο (2, 2), από το (2, 2) στο (2, 3) κ.ο.κ. Γενικά, θα θέλαμε να μάθει ο πράκτορας ότι η ενέργεια *Επάνω* αυξάνει τη συντεταγμένη y εκτός αν υπάρχει τοίχος μπροστά του, ότι η ενέργεια *Κάτω* τη μειώνει κ.ο.κ. Για να γίνει αυτό, χρειαζόμαστε δύο πράγματα. Πρώτον, χρειαζόμαστε μια τυπική αναπαράσταση γι' αυτό το είδος γενικών κανόνων που να επιδέχεται ρητό χειρισμό· μέχρι εδώ, κρατούσαμε τις πληροφορίες κρυμμένες μέσα στο μαύρο κουτί που ονομάσαμε συνάρτηση διαδόχων. Το Μέρος III του βιβλίου είναι αφιερωμένο σε αυτό το ζήτημα. Δεύτερον, χρειαζόμαστε αλγόριθμους που να μπορούν να κατασκευάζουν κατάλληλους γενικούς κανόνες από τις συγκεκριμένες παρατηρήσεις που κάνει ο πράκτορας. Αυτοί καλύπτονται στο Κεφάλαιο 18.

4.6 ΣΥΝΟΨΗ

Σε αυτό το κεφάλαιο εξετάσαμε την εφαρμογή **ευρετικών μηχανισμών** (heuristics) με σκοπό τη μείωση του κόστους αναζήτησης. Εξετάσαμε μερικούς αλγόριθμους που χρησιμοποιούν ευρετικούς μηχανισμούς και βρήκαμε ότι η βέλτιστη συμπεριφορά έχει ακριβό τίμημα σε κόστος αναζήτησης, ακόμα και με καλούς ευρετικούς μηχανισμούς.

- Η **αναζήτηση πρώτα στο καλύτερο** (best-first search) είναι απλώς μια αναζήτηση σε γράφημα (GRAPH-SEARCH) όπου επιλέγονται για επέκταση οι κόμβοι που δεν έχουν επεκταθεί οι οποίοι έχουν το ελάχιστο κόστος (σύμφωνα με κάποιο μέτρο). Οι αλγόριθμοι αναζήτησης πρώτα στο καλύτερο κανονικά χρησιμοποιούν μια **ευρετική συνάρτηση** $h(n)$ που εκτιμά το κόστος μιας λύσης από τον κόμβο n.
- Η **άπληστη αναζήτηση πρώτα στο καλύτερο** (greedy best-first search) επεκτείνει τους κόμβους με τη μικρότερη τιμή $h(n)$. Δεν είναι βέλτιστη, αλλά συχνά είναι αποδοτική.

- Η **αναζήτηση A*** (A* search) επεκτείνει τους κόμβους με την ελάχιστη τιμή $f(n) = g(n) + h(n)$. Η A* είναι πλήρης και βέλτιστη, με την προϋπόθεση ότι μπορούμε να εγγυηθούμε ότι η $h(n)$ είναι παραδεκτή για αναζήτηση σε δένδρο (TREE-SEARCH) ή συνεπής για αναζήτηση σε γράφημα (GRAPH-SEARCH). Η χωρική πολυπλοκότητα της αναζήτησης A* είναι πάντως απαγορευτική.
- Η απόδοση των αλγορίθμων ευρετικής αναζήτησης εξαρτάται από την ποιότητα της ευρετικής συνάρτησης. Καλοί ευρετικοί μηχανισμοί μπορούν μερικές φορές να κατασκευαστούν με χαλάρωση του ορισμού του προβλήματος ή με προηγούμενο υπολογισμό των κοστών λύσης για υποπροβλήματα σε μια βάση δεδομένων προτύπων (pattern database), ή με μάθηση από την εμπειρία στη συγκεκριμένη κλάση προβλημάτων.
- Οι αλγόριθμοι **RBFS** (αναδρομική αναζήτηση πρώτα στο καλύτερο) και **SMA*** (απλουστευμένη αναζήτηση A* περιορισμένης μνήμης) είναι εύρωστοι και βέλτιστοι αλγόριθμοι αναζήτησης που χρησιμοποιούν περιορισμένη ποσότητα μνήμης· αν τους δοθεί αρκετός χρόνος, μπορούν να επιλύουν προβλήματα που ο αλγόριθμος A* δεν μπορεί να επιλύσει επειδή εξαντλεί τη μνήμη.
- Οι μέθοδοι *τοπικής αναζήτησης*, όπως η **αναρρίχηση λόφων** (hill climbing), εφαρμόζονται σε διατυπώσεις με πλήρεις καταστάσεις, διατηρώντας στη μνήμη μόνο ένα μικρό αριθμό κόμβων. Έχουν επινοηθεί πολλοί στοχαστικοί αλγόριθμοι, μεταξύ των οποίων η **προσομοιωμένη ανόπτηση** (simulated annealing), η οποία επιστρέφει βέλτιστες λύσεις όταν της δοθεί ένας κατάλληλος χρονοπρογραμματισμός "ψύξης". Πολλές μέθοδοι τοπικής αναζήτησης μπορούν επίσης να χρησιμοποιούνται για την επίλυση προβλημάτων σε συνεχείς χώρους.
- **Γενετικός αλγόριθμος** (genetic algorithm) είναι μια στοχαστική αναζήτηση με αναρρίχηση λόφων στην οποία συντηρείται ένας μεγάλος πληθυσμός καταστάσεων. Νέες καταστάσεις παράγονται με **μετάλλαξη** (mutation) και με **διασταύρωση** (crossover), η οποία συνδυάζει ζεύγη καταστάσεων από τον πληθυσμό.
- Τα **προβλήματα εξερεύνησης** (exploration problems) προκύπτουν όταν ο πράκτορας δεν έχει ιδέα για τις καταστάσεις και τις ενέργειες του περιβάλλοντός του. Για τα ασφαλώς εξερευνήσιμα περιβάλλοντα, οι πράκτορες **online αναζήτησης** (online search) μπορούν να κατασκευάζουν έναν χάρτη και να βρίσκουν μια κατάσταση στόχου, αν υπάρχει. Η ενημέρωση των ευρετικών εκτιμήσεων με βάση την εμπειρία αποτελεί μια αποτελεσματική μέθοδο για να ξεφεύγει από τοπικά ελάχιστα.

ΒΙΒΛΙΟΓΡΑΦΙΚΕΣ ΚΑΙ ΙΣΤΟΡΙΚΕΣ ΣΗΜΕΙΩΣΕΙΣ

Η χρήση ευρετικής πληροφόρησης στην επίλυση προβλημάτων εμφανίστηκε σε μια πρώιμη δημοσίευση των Simon και Newell (1958), αλλά η φράση "ευρετική αναζήτηση" και η χρήση ευρετικών συναρτήσεων που εκτιμούν την απόσταση για την κατάσταση στόχου εμφανίστηκε κάπως αργότερα (Newell και Ernst, 1965· Lin, 1965). Οι Doran και Michie (1966) έκαναν εκτεταμένες πειραματικές μελέτες για την ευρετική αναζήτηση όπως εφαρμόζεται σε έναν αριθμό προβλημάτων, και ιδιαίτερα στα παζλ των 8 πλακιδίων και των 15 πλακιδίων. Αν και οι Doran και Michie έκαναν θεωρητικές αναλύσεις του μήκους διαδρομής και της "διεισδυτικότητας" (penetrance — ο λόγος του μήκους διαδρομής προς το συνολικό αριθμό κόμβων που έχουν εξεταστεί μέχρι στιγμής) στην ευρετική αναζήτηση, φαίνεται ότι αγνόησαν τις πληροφορίες που παρέχονται από το τρέχον μήκος διαδρομής (current path length). Ο αλγόριθμος A*, που ενσωμάτωσε το τρέχον μήκος διαδρομής στην ευρετική αναζήτηση, επινοήθηκε από τους Hart, Nilsson και Raphael (1968), με κάποιες μετέπειτα διορθώσεις (Hart *κ.α.*, 1972). Οι Dechter και Pearl (1985) έδειξαν ότι ο A* είναι βέλτιστα αποδοτικός.

Στην αρχική δημοσίευση για τον A* πρωτοεμφανίστηκε η συνθήκη της συνέπειας (consistency) για τις ευρετικές συναρτήσεις. Η συνθήκη της μονοτονίας προτάθηκε από τον Pohl (1977)

ως απλούστερο υποκατάστατο, αλλά ο Pearl (1984) απέδειξε ότι οι δύο συνθήκες είναι ισοδύναμες. Μερικοί αλγόριθμοι προγενέστεροι του A* χρησιμοποιούσαν το ισοδύναμο των ανοιχτών και κλειστών λιστών· σε αυτούς του αλγόριθμους περιλαμβάνονταν οι αναζητήσεις πρώτα κατά πλάτος, πρώτα κατά βάθος και ομοιόμορφου κόστους (Bellman, 1957· Dijkstra, 1959). Ιδιαίτερα η εργασία του Bellman έδειξε τη σπουδαιότητα των προσθετικών κοστών διαδρομής για την απλοποίηση των αλγορίθμων βελτιστοποίησης.

Ο Pohl (1970, 1977) ήταν πρωτοπόρος στη μελέτη της σχέσης μεταξύ του σφάλματος των ευρετικών συναρτήσεων και της χρονικής πολυπλοκότητας του αλγόριθμου A*. Η απόδειξη του ότι ο A* εκτελείται σε γραμμικό χρόνο αν το σφάλμα της ευρετικής συνάρτησης είναι φραγμένο από μια σταθερά μπορεί να βρεθεί στον Pohl (1977) και στον Gaschnig (1979). Ο Pearl (1984) ενίσχυσε αυτό το συμπέρασμα για την περίπτωση της λογαριθμικής αύξησης του σφάλματος. Ο "δραστικός παράγοντας διακλάδωσης" (effective branching factor) ως μέτρο της αποδοτικότητας της ευρετικής αναζήτησης προτάθηκε από τον Nilsson (1971).

Υπάρχουν πολλές παραλλαγές του αλγόριθμου A*. Ο Pohl (1973) πρότεινε τη χρήση *δυναμικής στάθμισης* (dynamic weighting), η οποία χρησιμοποιεί ένα σταθμισμένο άθροισμα $f_w(n) = w_g g(n) + w_h h(n)$ του τρέχοντος μήκους διαδρομής και την ευρετική συνάρτηση ως συνάρτηση αξιολόγησης, αντί για το απλό άθροισμα $f(n) = g(n) + h(n)$ που χρησιμοποιείται στον αλγόριθμο A*. Τα βάρη w_g και w_h προσαρμόζονται δυναμικά καθώς προχωρά η αναζήτηση. Μπορεί να αποδειχτεί ότι ο αλγόριθμος του Pohl είναι ϵ-παραδεκτός (ϵ-admissible) — δηλαδή βρίσκει εγγυημένα λύσεις μέσα στα όρια ενός παράγοντα $1+\epsilon$ από τη βέλτιστη λύση — όπου ϵ είναι μια παράμετρος που δίνεται στον αλγόριθμο. Την ίδια ιδιότητα έχει και ο αλγόριθμος A^*_ϵ (Pearl, 1984), ο οποίος μπορεί να επιλέγει οποιονδήποτε κόμβο από το σύνορο με την προϋπόθεση ότι το κόστος του f είναι μέσα στα όρια ενός παράγοντα $1+\epsilon$ από τον κόμβο συνόρου με το χαμηλότερο κόστος f. Η επιλογή μπορεί να γίνεται έτσι ώστε να ελαχιστοποιείται το κόστος αναζήτησης.

Ο A* και οι άλλοι αλγόριθμοι αναζήτησης στο χώρο καταστάσεων σχετίζονται στενά με τις τεχνικές *διακλάδωσης και οριοθέτησης* (*branch-and-bound*) που χρησιμοποιούνται ευρύτατα στην επιχειρησιακή έρευνα (Lawler και Wood, 1966). Οι σχέσεις μεταξύ της αναζήτησης στο χώρο καταστάσεων και της διακλάδωσης και οριοθέτησης έχουν ερευνηθεί σε βάθος (Kumar και Kanal, 1983· Nau *κ.α.*, 1984· Kumar *κ.α.*, 1988). Οι Martelli και Montanari (1978) έδειξαν ότι υπάρχει σύνδεση μεταξύ του δυναμικού προγραμματισμού (δείτε στο Κεφάλαιο 17) και ορισμένων τύπων αναζήτησης στο χώρο καταστάσεων. Οι Kumar και Kanal (1988) επιχείρησαν μια "μεγάλη ενοποίηση" της ευρετικής αναζήτησης, του δυναμικού προγραμματισμού και των τεχνικών διακλάδωσης και οριοθέτησης με την ονομασία CDP (composite decision process — σύνθετη διαδικασία απόφασης).

Επειδή οι υπολογιστές στα τέλη της δεκαετίας του 1950 και στις αρχές της δεκαετίας του 1960 είχαν το πολύ μερικές χιλιάδες λέξεις κύριας μνήμης, η ευρετική αναζήτηση περιορισμένης μνήμης ήταν από τα πρώτα αντικείμενα έρευνας. Το Graph Traverser (Doran και Michie, 1966), ένα από τα πρώτα προγράμματα αναζήτησης, απευθύνεται στο χειριστή αφού κάνει αναζήτηση πρώτα στο καλύτερο μέχρι το όριο μνήμης. Ο αλγόριθμος IDA* (A* με επαναληπτική εκβάθυνση — Korf, 1985a· Korf, 1985b) ήταν ο πρώτος δημοφιλής βέλτιστος αλγόριθμος ευρετικής αναζήτησης περιορισμένης μνήμης, και επινοήθηκαν πολλές παραλλαγές του. Μια ανάλυση της αποδοτικότητας του IDA* και των δυσκολιών του με τους ευρετικούς μηχανισμούς πραγματικών τιμών έχει γίνει από τους Patrick *κ.α.* (1992).

ΕΠΑΝΑΛΗΠΤΙΚΗ ΕΠΕΚΤΑΣΗ

Ο αλγόριθμος RBFS (αναδρομική αναζήτηση πρώτα στο καλύτερο — Korf, 1991· Korf, 1993) είναι στην πραγματικότητα κάπως πιο περίπλοκος από εκείνον που παρουσιάζεται στην Εικόνα 4.5, ο οποίος πλησιάζει περισσότερο τον αλγόριθμο **επαναληπτικής επέκτασης** (iterative expansion) ή IE (Russell, 1992) που αναπτύχθηκε ανεξάρτητα. Ο RBFS χρησιμοποιεί επίσης ένα κάτω φράγμα, εκτός από το άνω φράγμα· οι δύο αλγόριθμοι συμπεριφέρονται πανομοιότυπα αν χρησιμοποιηθούν παραδεκτοί (admissible) ευρετικοί μηχανισμοί, όμως ο RBFS επεκτείνει

τους κόμβους με προτεραιότητα πρώτα στο καλύτερο ακόμα και με ένα μη παραδεκτό ευρετικό μηχανισμό. Η ιδέα να παρακολουθείται η καλύτερη εναλλακτική διαδρομή εμφανίστηκε νωρίτερα στην όμορφη υλοποίηση του A* σε Prolog από τον Bratko (1986) και στον αλγόριθμο DTA* (Russell και Wefald, 1991). Αυτή η τελευταία εργασία εξετάζει επίσης τους χώρους καταστάσεων του μεταεπιπέδου και τη μάθηση στο μεταεπίπεδο.

Ο αλγόριθμος MA* (A* περιορισμένης μνήμης) παρουσιάστηκε από τους Chakrabarti *κ.α.* (1989). Ο SMA*, ή απλουστευμένος MA*, προέκυψε από μια απόπειρα να υλοποιηθεί ο MA* ως αλγόριθμος σύγκρισης για τον IE (Russell, 1992). Οι Kaindl και Khorsand (1994) χρησιμοποίησαν τον SMA* για να δημιουργήσουν έναν αλγόριθμο αμφίδρομης αναζήτησης που είναι σημαντικά ταχύτερος από προηγούμενους αλγόριθμους. Οι Korf και Zhang (2000) περιγράφουν μια προσέγγιση "διαίρει και βασίλευε" (divide-and-conquer), και οι Zhou και Hansen (2002) παρουσιάζουν μια αναζήτηση A* περιορισμένης μνήμης σε γράφημα. Ο Korf (1995) κάνει μια επισκόπηση των τεχνικών αναζήτησης περιορισμένης μνήμης.

Η ιδέα ότι παραδεκτοί ευρετικοί μηχανισμοί μπορούν να προκύψουν από τη χαλάρωση του προβλήματος εμφανίστηκε σε μια βαρυσήμαντη δημοσίευση των Held και Karp (1970), οι οποίοι χρησιμοποίησαν τον ευρετικό μηχανισμό του ελάχιστου απλωμένου δένδρου για να επιλύσουν το πρόβλημα του πλανόδιου πωλητή (δείτε την Άσκηση 4.8).

Η αυτοματοποίηση της διαδικασίας χαλάρωσης υλοποιήθηκε με επιτυχία από τον Prieditis (1993), ο οποίος οικοδόμησε πάνω στην προηγούμενη εργασία του με τον Mostow (Mostow και Prieditis, 1989). Η χρήση βάσεων δεδομένων προτύπων για την παραγωγή παραδεκτών ευρετικών μηχανισμών οφείλεται στον Gasser (1995) και τους Culberson και Schaeffer (1998)· οι βάσεις δεδομένων ξένων προτύπων περιγράφονται από τους Korf και Felner (2002). Η πιθανοτική ερμηνεία των ευρετικών μηχανισμών διερευνήθηκε σε βάθος από τον Pearl (1984) και τους Hansson και Mayer (1989).

Η πιο περιεκτική πηγή για τους ευρετικούς μηχανισμούς και τους αλγόριθμους ευρετικής αναζήτησης είναι σαφώς το *Heuristics* του Pearl (1984). Το βιβλίο αυτό καλύπτει εξαιρετικά τη μεγάλη ποικιλία των παραφυάδων και των παραλλαγών του αλγόριθμου A*, παρέχοντας αυστηρές αποδείξεις των τυπικών ιδιοτήτων τους. Οι Kanal και Kumar (1988) παρουσίασαν μια ανθολογία σημαντικών άρθρων για την ευρετική αναζήτηση. Νέα ευρήματα για τους αλγόριθμους αναζήτησης εμφανίζονται τακτικά στο περιοδικό *Artificial Intelligence*.

Οι τεχνικές τοπικής αναζήτησης έχουν μακρόχρονη ιστορία στα μαθηματικά και στην επιστήμη των υπολογιστών. Η μέθοδος Newton–Raphson (Newton, 1664· Raphson, 1690) μπορεί μάλιστα να θεωρηθεί μια πολύ αποδοτική μέθοδος τοπικής αναζήτησης για συνεχείς χώρους στους οποίους είναι διαθέσιμες πληροφορίες για τις κλίσεις. Το βιβλίο του Brent (1973) είναι μια κλασική πηγή αναφοράς για τους αλγόριθμους βελτιστοποίησης που δεν απαιτούν τέτοιες πληροφορίες. Η ακτινική αναζήτηση (beam search), την οποία παρουσιάσαμε ως αλγόριθμο τοπικής αναζήτησης, ξεκίνησε ως παραλλαγή φραγμένου πλάτους του δυναμικού προγραμματισμού για την αναγνώριση ομιλίας στο σύστημα HARPY (Lowerre, 1976). Ένας σχετικός αλγόριθμος αναλύεται σε βάθος από τον Pearl (1984, Κεφ. 5).

Το θέμα της τοπικής αναζήτησης αναζωπυρώθηκε τα τελευταία χρόνια από τα εκπληκτικά καλά αποτελέσματα σε μεγάλα προβλήματα ικανοποίησης περιορισμών όπως το πρόβλημα των *n* βασιλισσών (Minton *κ.α.*, 1992) και η λογική συλλογιστική (Selman *κ.α.*, 1992), και από την ενσωμάτωση τυχαιότητας, πολλών ταυτόχρονων αναζητήσεων και άλλων βελτιώσεων. Αυτή η αναγέννηση των αλγορίθμων τους οποίους ο Χρήστος Παπαδημητρίου ονόμασε αλγόριθμους "New Age" ήταν επίσης ένα έναυσμα για το αυξημένο ενδιαφέρων από την πλευρά των ειδικών της θεωρητικής επιστήμης των υπολογιστών (Koutsoupias και Papadimitriou, 1992· Aldous και Vazirani, 1994). Στο πεδίο της επιχειρησιακής έρευνας έχει γίνει δημοφιλής μια παραλλαγή της αναρρίχησης λόφων που ονομάζεται **αναζήτηση ταμπού** (tabu search — Glover, 1989· Glover και Laguna, 1997). Ο αλγόριθμος αυτός, ο οποίος βασίζεται σε μοντέλα της περιορισμένης βρα-

ΑΝΑΖΗΤΗΣΗ ΤΑΜΠΟΥ

χυπρόθεσμης μνήμης του ανθρώπου, συντηρεί μια λίστα ταμπού των k προηγούμενων καταστάσεων που επισκέφτηκε, τις οποίες δεν μπορεί να επισκεφτεί ξανά· εκτός από τη βελτίωση της αποδοτικότητας όταν γίνεται αναζήτηση σε γραφήματα, αυτό επιτρέπει στον αλγόριθμο να ξεφεύγει από μερικά τοπικά ελάχιστα. Μια άλλη χρήσιμη βελτίωση της αναρρίχησης λόφων είναι ο αλγόριθμος STAGE (Boyan και Moore, 1998). Η ιδέα είναι να χρησιμοποιούνται τα τοπικά μέγιστα που βρίσκονται από την αναρρίχηση λόφων με τυχαίες επανεκκινήσεις για να πάρει ο αλγόριθμος μια ιδέα για τη συνολική μορφή του τοπίου. Ο αλγόριθμος προσαρμόζει μια ομαλή επιφάνεια στο σύνολο των τοπικών μεγίστων και μετά υπολογίζει το καθολικό μέγιστο αυτής της επιφάνειας αναλυτικά. Αυτό γίνεται το νέο σημείο εκκίνησης. Ο αλγόριθμος αυτός έχει αποδειχτεί πρακτικά αποτελεσματικός για δύσκολα προβλήματα. Οι Gomes *κ.α.* (1998) έδειξαν ότι οι κατανομές χρόνου εκτέλεσης των συστηματικών αλγορίθμων υπαναχώρησης συχνά έχουν **κατανομή με βαριά ουρά** (heavy-tailed distribution), που σημαίνει ότι η πιθανότητα πολύ μεγάλου χρόνου εκτέλεσης είναι μεγαλύτερη από εκείνη που θα προβλεπόταν αν οι χρόνοι εκτέλεσης είχαν κανονική κατανομή. Αυτό παρέχει θεωρητική υποστήριξη στις τυχαίες επανεκκινήσεις.

ΚΑΤΑΝΟΜΗ ΜΕ ΒΑΡΕΙΑ ΟΥΡΑ

Η προσομοιωμένη ανόπτηση (simulated annealing) πρωτοπαρουσιάστηκε από τους Kirkpatrick *κ.α.* (1983), που τη δανείστηκαν απευθείας από τον **αλγόριθμο του Metropolis** (ο οποίος χρησιμοποιείται για την προσομοίωση πολύπλοκων συστημάτων στη Φυσική (Metropolis *κ.α.*, 1953) και λέγεται ότι εφευρέθηκε σε ένα εορταστικό δείπνο στο Los Alamos). Η προσομοιωμένη ανόπτηση είναι σήμερα ένα πεδίο από μόνη της, με εκατοντάδες δημοσιεύσεις κάθε χρόνο.

Η εύρεση βέλτιστων λύσεων σε συνεχείς χώρους είναι το θέμα πολλών πεδίων, μεταξύ των οποίων η **θεωρία της βελτιστοποίησης** (optimization theory), η **θεωρία του βέλτιστου ελέγχου** (optimal control theory) και ο **λογισμός των μεταβολών** (calculus of variations). Κατάλληλα (και πρακτικά) εισαγωγικά εγχειρίδια έχουν γραφεί από τους Press *κ.α.* (2002) και τον Bishop (1995). Ο **γραμμικός προγραμματισμός** (linear programming) ήταν μία από τις πρώτες εφαρμογές των υπολογιστών· ο **αλγόριθμος simplex** (Wood και Dantzig, 1949· Dantzig, 1949) εξακολουθεί να χρησιμοποιείται παρά την εκθετική του πολυπλοκότητα στη χειρότερη περίπτωση. Ο Karmarkar (1984) ανέπτυξε έναν πρακτικό αλγόριθμο πολυωνυμικού χρόνου για το γραμμικό προγραμματισμό.

Η εργασία του Sewall Wright (1931) πάνω στην έννοια του **τοπίου καταλληλότητας** (fitness landscape) ήταν ένας σημαντικός πρόδρομος για την ανάπτυξη των γενετικών αλγορίθμων. Στη δεκαετία του 1950, πολλοί στατιστικολόγοι, μεταξύ των οποίων ο Box (1957) και ο Friedman (1959), χρησιμοποίησαν εξελικτικές τεχνικές σε προβλήματα βελτιστοποίησης, όμως η προσέγγιση αυτή έγινε δημοφιλής μόνο αφού ο Rechenberg (1965, 1973) παρουσίασε **στρατηγικές εξέλιξης** (evolution strategies) για την επίλυση προβλημάτων βελτιστοποίησης για αεροδυναμικές επιφάνειες. Στις δεκαετίες του 1960 και του 1970, ο John Holland (1975) υποστήριξε τους γενετικούς αλγόριθμους, τόσο ως χρήσιμο εργαλείο όσο και ως μέθοδο για τη διεύρυνση της κατανόησής μας για την προσαρμογή, βιολογική ή όχι (Holland, 1995). Το κίνημα της **τεχνητής ζωής** (artificial life — Langton, 1995) ώθησε αυτή την ιδέα ένα βήμα παραπέρα, θεωρώντας τα προϊόντα των γενετικών αλγορίθμων ως *οργανισμούς* και όχι ως λύσεις προβλημάτων. Η δουλειά σε αυτό το πεδίο από του Hinton και Nowlan (1987) και από τους Ackley και Littman (1991) συνέβαλε πολύ στο να ξεκαθαριστούν οι συνέπειες του φαινομένου Baldwin. Για γενικές γνώσεις υποβάθρου πάνω στην εξέλιξη, συνιστούμε θερμά το βιβλίο των Smith και Szathmary (1999).

ΣΤΡΑΤΗΓΙΚΕΣ ΕΞΕΛΙΞΗΣ

ΤΕΧΝΗΤΗ ΖΩΗ

Οι περισσότερες συγκρίσεις γενετικών αλγορίθμων με άλλες προσεγγίσεις (ιδιαίτερα με τη στοχαστική αναρρίχηση λόφων) έχουν βρει ότι οι γενετικοί αλγόριθμοι συγκλίνουν πιο αργά (O'Reilly και Oppacher, 1994· Mitchell *κ.α.*, 1994· Juels και Wattenberg, 1996· Baluja, 1997). Τα ευρήματα αυτά δεν είναι γενικά δημοφιλή στην κοινότητα των γενετικών αλγορίθμων, αλλά πρόσφατες προσπάθειες μέσα στην ίδια την κοινότητα για να γίνει κατανοητή η αναζήτηση με βάση

βάση τον πληθυσμό ως προσεγγιστική μορφή μάθησης Bayes (δείτε στο Κεφάλαιο 20) μπορεί να βοηθήσουν να κλείσει το χάσμα μεταξύ του πεδίου αυτού και των επικριτών του (Pelikan *κ.α.*, 1999). Η θεωρία των **τετραγωνικών δυναμικών συστημάτων** (quadratic dynamical systems) μπορεί επίσης να εξηγήσει την απόδοση των γενετικών αλγορίθμων (Rabani *κ.α.*, 1998). Δείτε Lohn *κ.α.* (2001) για ένα παράδειγμα εφαρμογής γενετικών αλγορίθμων στη σχεδίαση κεραιών, και στο Larranaga *κ.α.* (1999) για μια εφαρμογή στο πρόβλημα του πλανόδιου πωλητή.

ΓΕΝΕΤΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Το πεδίο του **γενετικού προγραμματισμού** (genetic programming) έχει στενή σχέση με τους γενετικούς αλγόριθμους. Η κύρια διαφορά είναι ότι οι αναπαραστάσεις που μεταλλάσσονται και συνδυάζονται είναι προγράμματα και όχι ψηφιοσειρές. Τα προγράμματα αναπαρίστανται με τη μορφή δένδρων παραστάσεων· οι παραστάσεις μπορεί να είναι σε μια καθιερωμένη γλώσσα όπως η Lisp, ή μπορεί να είναι ειδικά σχεδιασμένες για να αντιπροσωπεύουν κυκλώματα, ελεγκτές ρομπότ κ.λπ. Η διασταύρωση συνίσταται στη συνένωση υποδένδρων αντί για τμήματα συμβολοσειρών. Αυτή η μορφή μετάλλαξης εγγυάται ότι οι απόγονοι είναι καλά διατυπωμένες παραστάσεις, το οποίο δε θα συνέβαινε αν τα προγράμματα αντιμετωπίζονταν ως συμβολοσειρές.

Το πρόσφατο ενδιαφέρον για το γενετικό προγραμματισμό υποκινήθηκε από τη δουλειά του John Koza (Koza, 1992· Koza, 1994), αλλά χρονολογείται τουλάχιστον από τα πρώιμα πειράματα του Friedberg (1958) με κώδικα μηχανής και από τα αυτόματα πεπερασμένων καταστάσεων των Fogel *κ.α.* (1966). Όπως συμβαίνει και με τους γενετικούς αλγόριθμους, υπάρχει αμφισβήτηση για την αποτελεσματικότητα αυτής της τεχνικής. Οι Koza *κ.α.* (1999) περιγράφουν μια ποικιλία πειραμάτων πάνω στην αυτοματοποιημένη σχεδίαση συσκευών ηλεκτρονικών κυκλωμάτων με τη χρήση γενετικού προγραμματισμού.

Τα περιοδικά *Evolutionary Computation* και *IEEE Transactions on Evolutionary Computation* καλύπτουν τους γενετικούς αλγόριθμους και το γενετικό προγραμματισμό· άρθρα δημοσιεύονται επίσης στο *Complex Systems*, στο *Adaptive Behavior* και στο *Artificial Life*. Τα κυριότερα επιστημονικά συνέδρια είναι το *International Conference on Genetic Algorithms* και το *Conference on Genetic Programming*, τα οποία πρόσφατα συγχωνεύτηκαν στο *Genetic and Evolutionary Computation Conference*. Τα βιβλία της Melanie Mitchell (1996) και του David Fogel (2000) αποτελούν καλές επισκοπήσεις του πεδίου.

Οι αλγόριθμοι για την εξερεύνηση άγνωστων χώρων καταστάσεων έχουν αποτελέσει αντικείμενο ενδιαφέροντος εδώ και πολλούς αιώνες. Η αναζήτηση πρώτα κατά βάθος σε ένα λαβύρινθο μπορεί να υλοποιηθεί με το κράτημα του αριστερού μας χεριού πάνω στον τοίχο· οι βρόχοι μπορούν να αποφευχθούν με την σήμανση κάθε διασταύρωσης. Η αναζήτηση πρώτα κατά βάθος αποτυγχάνει με τις μη αναστρέψιμες ενέργειες· το γενικότερο πρόβλημα της εξερεύνησης των **γραφημάτων Euler** (Eulerian graphs — γραφήματα στα οποία κάθε κόμβος έχει ίσο αριθμό εισερχόμενων και εξερχόμενων ακμών) επιλύθηκε με έναν αλγόριθμο που οφείλεται στον Hierholzer (1873). Η πρώτη ολοκληρωμένη αλγοριθμική μελέτη του προβλήματος της εξερεύνησης για τυχαία γραφήματα πραγματοποιήθηκε από τους Deng και Papadimitriou (1990), οι οποίοι ανέπτυξαν έναν εντελώς γενικό αλγόριθμο, αλλά έδειξαν ότι δεν είναι εφικτός κανένας φραγμένος ανταγωνιστικός λόγος στην εξερεύνηση ενός γενικού γραφήματος. Οι Papadimitriou και Yannakakis (1991) εξέτασαν το ζήτημα της εύρεσης διαδρομών προς μια κατάσταση στόχου σε γεωμετρικά περιβάλλοντα σχεδιασμού διαδρομών (όπου όλες οι ενέργειες είναι αναστρέψιμες). Έδειξαν ότι ένας μικρός ανταγωνιστικός λόγος είναι εφικτός για τετράγωνα εμπόδια, αλλά κανένας φραγμένος ανταγωνιστικός λόγος δεν μπορεί να επιτευχθεί για γενικά ορθογώνια εμπόδια (δείτε στην Εικόνα 4.19).

ΓΡΑΦΗΜΑΤΑ EULER

ΑΝΑΖΗΤΗΣΗ ΣΕ ΠΡΑΓΜΑΤΙΚΟ ΧΡΟΝΟ

Ο αλγόριθμος LRTA* (A* πραγματικού χρόνου που μαθαίνει) επινοήθηκε από τον Korf (1990) στα πλαίσια μιας διερεύνησης της **αναζήτησης σε πραγματικό χρόνο** (real-time search) για περιβάλλοντα στα οποία ο πράκτορας πρέπει να ενεργεί αφού κάνει αναζήτηση μόνο για ένα καθορισμένο χρονικό διάστημα (μια πολύ συνηθισμένη περίπτωση για τα παιχνίδια δύο παι-

κτών). Ο LRTA* είναι στην πραγματικότητα ειδική περίπτωση των αλγορίθμων ενισχυτικής μάθησης για στοχαστικά περιβάλλοντα (Barto *κ.α.*, 1995). Η πολιτική της αισιοδοξίας υπό αβεβαιότητα που ακολουθεί — κατευθύνεται πάντα στην πιο κοντινή κατάσταση που δεν έχει επισκεφτεί — μπορεί να οδηγήσει σε ένα πρότυπο εξερεύνησης λιγότερο αποδοτικό από την απλή αναζήτηση πρώτα κατά βάθος στην περίπτωση χωρίς πληροφόρηση (Koenig, 2000). Οι Dasgupta *κ.α.* (1994) έδειξαν ότι η online αναζήτηση επαναληπτικής εκβάθυνσης είναι βέλτιστα αποδοτική για την εύρεση μιας κατάστασης στόχου σε ένα ομοιόμορφο δένδρο χωρίς ευρετική πληροφόρηση. Έχουν αναπτυχθεί πολλές παραλλαγές στο θέμα του LRTA* με πληροφόρηση, με διαφορετικές μεθόδους αναζήτησης και ενημέρωσης μέσα στα πλαίσια του γνωστού μέρους του γραφήματος (Pemberton και Korf, 1992). Μέχρι σήμερα, δεν έχει γίνει καλά κατανοητό το πώς πρέπει να βρίσκονται με βέλτιστη αποδοτικότητα οι καταστάσεις στόχου όταν χρησιμοποιείται ευρετική πληροφόρηση.

ΠΑΡΑΛΛΗΛΗ ΑΝΑΖΗΤΗΣΗ

Το θέμα των αλγορίθμων **παράλληλης αναζήτησης** (parallel search) δεν καλύφθηκε σε αυτό το κεφάλαιο επειδή, μεταξύ άλλων, απαιτεί εκτεταμένη συζήτηση των παράλληλων αρχιτεκτονικών υπολογιστών. Η παράλληλη αναζήτηση αναδεικνύεται σε σημαντικό θέμα τόσο στην ΤΝ όσο και στη θεωρητική επιστήμη των υπολογιστών. Μια σύντομη εισαγωγή στη σχετική βιβλιογραφία της ΤΝ μπορείτε να βρείτε στη δημοσίευση των Mahanti και Daniels (1993).

ΑΣΚΗΣΕΙΣ

4.1 Παρακολουθήστε τη λειτουργία μιας αναζήτησης Α* που εφαρμόζεται στο πρόβλημα της μετάβασης από το Lugoj στο Βουκουρέστι, χρησιμοποιώντας τον ευρετικό μηχανισμό της ευθύγραμμης απόστασης. Δηλαδή, δείξτε την ακολουθία κόμβων που θα εξετάσει ο αλγόριθμος και τη βαθμολογία f, g και h του κάθε κόμβου.

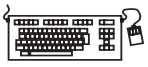
4.2 Ο **ευρετικός αλγόριθμος διαδρομής** (heuristic path algorithm) είναι μια αναζήτηση πρώτα στο καλύτερο όπου η αντικειμενική συνάρτηση είναι $f(n) = (2 - w)g(n) + wh(n)$. Για ποιες τιμές του w είναι ο αλγόριθμος εγγυημένα βέλτιστος; Μπορείτε να θεωρήσετε ότι η h είναι αποδεκτή. Τι είδους αναζήτηση κάνει όταν είναι $w = 0$; Όταν είναι $w = 1$; Όταν είναι $w = 2$;

4.3 Αποδείξτε τις παρακάτω προτάσεις:

α. Η αναζήτηση πρώτα κατά πλάτος είναι ειδική περίπτωση της αναζήτησης ομοιόμορφου κόστους.

β. Η αναζήτηση πρώτα κατά πλάτος, η αναζήτηση πρώτα κατά βάθος και η αναζήτηση ομοιόμορφου κόστους είναι ειδικές περιπτώσεις της αναζήτησης πρώτα στο καλύτερο.

γ. Η αναζήτηση ομοιόμορφου κόστους είναι ειδική περίπτωση της αναζήτησης Α*.



4.4 Επινοήστε ένα χώρο καταστάσεων στον οποίο ο αλγόριθμος Α* που χρησιμοποιεί αναζήτηση σε γράφημα (GRAPH-SEARCH) επιστρέφει μια μη βέλτιστη λύση με συνάρτηση $h(n)$ που είναι παραδεκτή (admissible) αλλά όχι συνεπής (inconsistent).

4.5 Στην Ενότητα 4.1, στην υποενότητα για την άπληστη αναζήτηση πρώτα στο καλύτερο, είδαμε ότι ο ευρετικός μηχανισμός της ευθύγραμμης απόστασης παραπλανά την άπληστη αναζήτηση πρώτα στο καλύτερο στο πρόβλημα της μετάβασης από το Iasi στο Fagaras. Όμως, η ευρετική είναι άψογη στο αντίστροφο πρόβλημα, της μετάβαση από το Fagaras στο Iasi. Υπάρχουν προβλήματα στα οποία αυτός ο ευρετικός μηχανισμός παραπλανά και προς τις δύο κατευθύνσεις;

4.6 Επινοήστε μια ευρετική συνάρτηση για το παζλ των 8 πλακιδίων η οποία μερικές φορές να υπερεκτιμά, και δείξτε πώς αυτή μπορεί να οδηγήσει σε μη βέλτιστη λύση σε ένα συγκεκριμένο πρόβλημα. (Μπορείτε να χρησιμοποιήσετε υπολογιστή για βοήθεια, αν θέλετε.) Αποδείξτε ότι, αν η h δεν υπερεκτιμά ποτέ κατά περισσότερο από c, τότε ο αλγόριθμος Α* που χρησιμο-

ποιεί την h επιστρέφει μια λύση που το κόστος της δεν ξεπερνά εκείνο της βέλτιστης λύσης κατά περισσότερο από c.

4.7 Αποδείξτε ότι αν ένας ευρετικός μηχανισμός είναι συνεπής, είναι οπωσδήποτε παραδεκτός. Κατασκευάστε έναν παραδεκτό ευρετικό μηχανισμό που να μην είναι συνεπής.

4.8 Το πρόβλημα του πλανόδιου πωλητή μπορεί να επιλυθεί με τον ευρετικό μηχανισμό του ελάχιστου απλωμένου δένδρου (minimum spanning tree, MST), ο οποίος χρησιμοποιείται για την εκτίμηση του κόστους μιας πλήρους περιοδείας, με δεδομένο ότι έχει ήδη κατασκευαστεί μια μερική περιοδεία. Το κόστος MST ενός συνόλου πόλεων είναι το μικρότερο άθροισμα των κοστών των συνδέσμων οποιουδήποτε δένδρου που συνδέει όλες τις πόλεις.

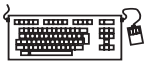
α. Δείξτε πώς μπορεί αυτός ο ευρετικός μηχανισμός να προκύψει από μια χαλαρή έκδοση του προβλήματος του πλανόδιου πωλητή.

β. Δείξτε ότι ο ευρετικός μηχανισμός MST κυριαρχεί έναντι της ευθύγραμμης απόστασης.

γ. Γράψτε μια γεννήτρια προβλημάτων για στιγμιότυπα του προβλήματος του πλανόδιου πωλητή όπου οι πόλεις αναπαρίστανται με τυχαία σημεία στο μοναδιαίο τετράγωνο.

δ. Βρείτε στη βιβλιογραφία έναν αποδοτικό αλγόριθμο για την κατασκευή του ευρετικού μηχανισμού MST, και χρησιμοποιήστε τον με έναν παραδεκτό αλγόριθμο αναζήτησης για να επιλύσετε στιγμιότυπα του προβλήματος του πλανόδιου πωλητή.

4.9 Στην Ενότητα 4.2, στην υποενότητα για την επινόηση παραδεκτών ευρετικών συναρτήσεων, ορίσαμε μια χαλάρωση του παζλ των 8 πλακιδίων όπου ένα πλακίδιο μπορεί να μετακινείται από ένα τετράγωνο A σε οποιοδήποτε τετράγωνο B αν το B είναι κενό. Η ακριβής λύση αυτού του προβλήματος ορίζει τον **ευρετικό μηχανισμό του Gaschnig** (Gaschnig, 1979). Εξηγήστε γιατί ο ευρετικός μηχανισμός του Gaschnig είναι τουλάχιστον το ίδιο ακριβής με τον h_1 (πλακίδια εκτός θέσης), και βρείτε περιπτώσεις όπου είναι πιο ακριβής και από τον h_1 και από τον h_2 (απόσταση Manhattan). Μπορείτε να προτείνετε έναν αποδοτικό τρόπο υπολογισμού για τον ευρετικό μηχανισμό του Gaschnig;



4.10 Παρουσιάσαμε δύο απλούς ευρετικούς μηχανισμούς για το παζλ των 8 πλακιδίων· την απόσταση Manhattan και τα πλακίδια εκτός θέσης. Πολλοί ευρετικοί μηχανισμοί στη σχετική βιβλιογραφία προτείνονται ως βελτιώσεις — δείτε, για παράδειγμα, Nilsson (1971), Mostow και Prieditis (1989), και Hansson *κ.α.* (1992). Ελέγξτε αυτούς τους ισχυρισμούς υλοποιώντας τους ευρετικούς μηχανισμούς και συγκρίνοντας τις αποδόσεις των αλγορίθμων που προκύπτουν.

4.11 Δώστε το όνομα του αλγόριθμου που προκύπτει από κάθε μία από τις παρακάτω ειδικές περιπτώσεις:

α. Τοπική ακτινική αναζήτηση με $k = 1$.

β. Τοπική ακτινική αναζήτηση με μια αρχική κατάσταση και χωρίς όριο στο πλήθος των καταστάσεων που διατηρούνται.

γ. Προσομοιωμένη ανόπτηση με $T = 0$ πάντοτε, και παραλείποντας τον έλεγχο τερματισμού.

δ. Γενετικός αλγόριθμος με μέγεθος πληθυσμού $N = 1$.

4.12 Μερικές φορές δεν υπάρχει καλή συνάρτηση αξιολόγησης για ένα πρόβλημα, αλλά υπάρχει μια καλή μέθοδος σύγκρισης· ένας τρόπος για να διακρίνουμε αν ένας κόμβος είναι καλύτερος από έναν άλλο χωρίς να αποδίδουμε αριθμητικές τιμές σε κανέναν από τους δύο. Δείξτε ότι αυτό αρκεί για να κάνουμε μια αναζήτηση πρώτα στο καλύτερο. Υπάρχει κάτι ανάλογο για την αναζήτηση A*;

4.13 Συσχετίστε τη χρονική πολυπλοκότητα του αλγόριθμου LRTA* (A* πραγματικού χρόνου που μαθαίνει) με τη χωρική του πολυπλοκότητα.

4.14 Ας υποθέσουνε ότι ένας πράκτορας βρίσκεται σε ένα περιβάλλον λαβύρινθου 3×3 σαν αυτό της Εικόνας 4.18. Ο πράκτορας γνωρίζει ότι η αρχική του θέση είναι στο (1, 1), ότι η κατά-

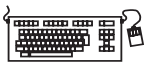
σταση στόχου είναι στο (3, 3), και ότι οι τέσσερις ενέργειες *Επάνω*, *Κάτω*, *Αριστερά*, *Δεξιά* έχουν το συνηθισμένο τους αποτέλεσμα εκτός αν εμποδίζονται από τοίχο. Ο πράκτορας *δεν* γνωρίζει πού βρίσκονται οι εσωτερικοί τοίχοι. Σε οποιαδήποτε δεδομένη κατάσταση, ο πράκτορας αντιλαμβάνεται το σύνολο των επιτρεπτών ενεργειών· μπορεί επίσης να διακρίνει αν η κατάσταση είναι μία από αυτές που έχει επισκεφτεί προηγουμένως ή νέα κατάσταση.

α. Εξηγήστε πώς αυτό το πρόβλημα online αναζήτησης μπορεί να θεωρηθεί ως offline αναζήτηση στο χώρο των καταστάσεων πεποίθησης, όπου η αρχική κατάσταση πεποίθησης περιλαμβάνει όλες τις δυνατές διευθετήσεις του περιβάλλοντος. Τι μέγεθος έχει η αρχική κατάσταση πεποίθησης; Τι μέγεθος έχει ο χώρος των καταστάσεων πεποίθησης;

β. Πόσες διαφορετικές αντιλήψεις είναι δυνατές στην αρχική κατάσταση;

γ. Περιγράψτε μερικές από τις πρώτες διακλαδώσεις ενός πλάνου ενδεχομένων (contingency plan) γι' αυτό το πρόβλημα. Τι μέγεθος (περίπου) έχει το πλήρες πλάνο;

Προσέξτε ότι αυτό το πλάνο ενδεχομένων είναι μια λύση για *όλα τα δυνατά περιβάλλοντα* που ταιριάζουν στη δεδομένη περιγραφή. Γι' αυτό, η διαπλοκή αναζήτησης και εκτέλεσης δεν είναι εντελώς απαραίτητη, ακόμα και σε άγνωστα περιβάλλοντα.

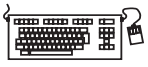


4.15 Σε αυτή την άσκηση θα εξερευνήσουμε τη χρήση μεθόδων τοπικής αναζήτησης για την επίλυση προβλημάτων πλανόδιου πωλητή του τύπου που ορίσαμε στην Άσκηση 4.8.

α. Επινοήστε μια προσέγγιση με αναρρίχηση λόφων για την επίλυση προβλημάτων πλανόδιου πωλητή. Συγκρίνετε τα αποτελέσματα με τις βέλτιστες λύσεις που προκύπτουν από τον αλγόριθμο A* με τον ευρετικό μηχανισμό του ελάχιστου απλωμένου δένδρου (MST — Άσκηση 4.8).

β. Επινοήστε μια προσέγγιση με γενετικό αλγόριθμο για το πρόβλημα του πλανόδιου πωλητή. Συγκρίνετε τα αποτελέσματα με τις άλλες προσεγγίσεις. Καλό θα ήταν να συμβουλευτείτε τη δημοσίευση των Larranaga *κ.α.* (1999) για να πάρετε ιδέες για τις αναπαραστάσεις.

4.16 Παράγετε ένα μεγάλο αριθμό στιγμιοτύπων των προβλημάτων των 8 πλακιδίων και των 8 βασιλισσών και επιλύστε τα (όπου είναι δυνατόν) με αναρρίχηση λόφων (παραλλαγές της πλέον απότομης ανάβασης και της πρώτης επιλογής), με αναρρίχηση λόφων με τυχαίες επανεκκινήσεις, και με προσομοιωμένη ανόπτηση. Μετρήστε το κόστος αναζήτησης και το ποσοστό των λυμένων προβλημάτων και σχεδιάστε μια γραφική παράσταση αυτών των τιμών ως προς το βέλτιστο κόστος λύσης. Σχολιάστε τα αποτελέσματά σας.



4.17 Σε αυτή την άσκηση θα εξετάσουμε την αναρρίχηση λόφων στα πλαίσια της πλοήγησης ρομπότ, χρησιμοποιώντας το περιβάλλον της Εικόνας 3.22 ως παράδειγμα.

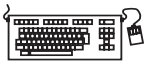
α. Επαναλάβετε την Άσκηση 3.16 χρησιμοποιώντας αναρρίχηση λόφων. Παγιδεύεται ο πράκτοράς σας ποτέ σε τοπικό ελάχιστο; Είναι *δυνατό* να παγιδευτεί ο πράκτορας με κυρτά εμπόδια;

β. Κατασκευάστε ένα μη κυρτό πολυγωνικό περιβάλλον στο οποίο ο πράκτορας να παγιδεύεται.

γ. Τροποποιήστε τον αλγόριθμο αναρρίχηση λόφων έτσι ώστε, αντί να κάνει αναζήτηση βάθους 1 για να αποφασίσει πού θα πάει στη συνέχεια, να κάνει αναζήτηση βάθους k. Θα πρέπει να βρίσκει την καλύτερη διαδρομή k βημάτων και να κάνει ένα βήμα πάνω σε αυτή, και μετά να επαναλαμβάνει τη διαδικασία.

δ. Υπάρχει κάποιο k για το οποίο ο νέος αλγόριθμος να ξεφεύγει εγγυημένα από τα τοπικά ελάχιστα;

ε. Εξηγήστε πώς ο αλγόριθμος LRTA* (A* πραγματικού χρόνου που μαθαίνει) επιτρέπει στον πράκτορα να ξεφεύγει από τα τοπικά ελάχιστα σε αυτή την περίπτωση.



4.18 Συγκρίνετε την απόδοση των αλγορίθμων A* και RBFS (αναδρομικής αναζήτησης πρώτα στο καλύτερο) για ένα σύνολο προβλημάτων που έχουν παραχθεί τυχαία, στα πεδία του παζλ των 8 πλακιδίων (με απόσταση Manhattan) και του προβλήματος του πλανόδιου πωλητή (με MST — δείτε της Άσκηση 4.8). Συζητήστε τα αποτελέσματά σας. Τι συμβαίνει με την απόδοση του RBFS όταν προστίθεται ένας μικρός τυχαίος αριθμός στις ευρετικές τιμές, στο πεδίο του παζλ των 8 πλακιδίων;