<u>סיכום תכונות אלגוריתמים</u>
פתרון בעיות באמצעות חיפוש

Single Agent Path Finding Problems

	שלמות	אופטימליות	סיבוכיות זמן ריצה	סיבוכיות זכרון	תנאי עצירה	closed-list?	מבנה נתונים של ה-open list
BFS	V	V	$O(b^d)$	$O(b^d)$	**פיתוח** קודקוד ה-GOAL	רק עם loop avoidance	תור
DFS	X	X	$O(b^m)$	רקורסיבי: $O(m)$, מחסנית: $O(bm)$	**פיתוח** קודקוד ה-GOAL	X	מחסנית
Limited DFS	אם $d \leq l$	X	$O(b^l)$	$O(bl)$			
DFID	V	V	$O(b^m)$	$O(mb)$ $or\ O(m)$	כשהקודקוד שמקבלים ב-limited dfs הוא ה-goal (ילד שנשלח ברקורסיה) ואז מחזירים את path לפונקציה השנייה / כשמוחזר fail	X	hash table
Bi-directional Search	V	V	$O(b^d)$ ואם זמן ההשוואה קבוע אז $O(b^{d/2})$	$O(b^{d/2})$			

USC	רק אם מניחים חסם תחתון על משקל הצלעות	V	$O(b^{\lfloor c^*/e \rfloor+1})$	$O(b^{\lfloor c^*/e \rfloor+1})$	**הוצאת** קודקוד ה-GOAL מתור העדיפויות	V	תור עדיפויות
Greedy Search	X	X	$O(b^m)$	$O(b^m)$	כמו USC	V	תור עדיפויות
A*	רק אם הגרף סופי או שמתקיים תנאי 1 (למטה)	רק אם $h(n)$ היא consistent	אם $f = g$: $O(b^{\lfloor c^*/e \rfloor+1})$ אם $f = g + h^*$: $O(bd)$	כמו סיבוכיות הזמן (זה החיסרון העיקרי שלו)	כמו USC	V	תור עדיפויות
IDA*	V (ללא תלות ביוריסטיקה)	רק אם $h(n)$ היא admissible	במקרה גרוע יכול להיות בחזקת 2 מ-A*, אבל לפעמים מהיר יותר	$O(b\lfloor c^*/e \rfloor + 1$	**פיתוח** GOAL שלא עובר את ה-threshold, כלומר $f(g) \leq t$	X	מחסנית
DFBnB	V	V בהינתן lower-bound heuristic	בגרף עץ - כמו A*. בגרף כללי - A* יעיל יותר כי הוא גוזם צמתים כפולים (כאלו שב-open list)	$O(bd)$	המחסנית ריקה (עברנו על כל העץ), חוזר result שיכול להיות null אם מעולם לא מצאנו GOAL, או דרך כלשהי לGOAL הכי טוב שנמצא	X	מחסנית

פרמטרים:

b = branching factor	d = solution depth
m = the maximum depth of search tree	l = the depth limit (cutoff)
e = the minimum non-zero edge cost	c* = the cost of optimal solution
h* = the true cost to the goal	

תנאי 1:

In an infinite graph: if all edge costs are finite and have a minimum positive value, and all heuristic values are finite and non-negative. Under those conditions the cost of nodes will eventually increase without bound. Therefore, there could not be an infinite loop.

תכונות / משפחות של אלגוריתמים:

- גרף לא ממושקל בלבד - BFS, DFS, Limited DFS, DFID. בכל השאר אפשר גרף ממושקל.
- Uninformed-search - מההתחלה עד USC. כל השאר Informed-search.
- Iterative Deepening - גם DFID וגם IDA*.
- Optimally efficient - רק DFID, A*
- אלגוריתמי BFS:
- אלגוריתמי DFS:
- any-time algorithm - רק DFBnB. בכל שלב יכול לעצור ולהחזיר פתרון כלשהו (הטוב ביותר שהוא הצליח למצוא עד כה).
- אלגוריתמים המשתמשים ב-global cost bound - גם DFBnB וגם IDA*. רק שהראשון מחזיק חסם עליון ומקטין אותו תוך כדי, והשני חסם תחתון ומגדיל אותו.
- לכולם יש open-list, למי שיש closed-list זה רק:

השוואת IDA* לאחרים:

Q: What are the advantages of IDA* over:

- A*?
- DFS (no closed list)?
- Uniform-Cost (closed list)?

Alg. / Adv.	Endless branch	Informed pruning	Space	Optimality
A*			V	
DFS	V	V		V
UC		V	V	

DFBnB ו-IDA* מפתחים יותר קודקודים מאשר A*.

הסיבה: IDA* מרחיב רק קודקודים עם עלות נמוכה מ-c*, אבל יכול להרחיב את אותו אחד כמה פעמים.

DFBnB מרחיב כל קודקוד פעם אחת (אם יש אליו מסלול אחד), אבל יכול להרחיב קודקודים עם עלות גבוהה מ-c*.

Constraint-Satisfaction Problems

	מה פותר	סיבוכיות זמן ריצה	סיבוכיות זכרון
AC-3	Arc Consistency	$O(n^2 d^3)$	
backtracking search	מציאת goal (אחד) ב-CSP (לאחר שעשינו constraint prop)	כמו DFS. מספר העלים = d^n	פחות מ-$n! \cdot d$ זכרון קבוע. כי הוא יוצר פעם אחת מצב ורק מעדכן אותו, לא יוצר קודקודים חדשים כל פעם

פרמטרים:

b = branching factor
d = (max) domain size
n = number of variables

Local Search, Genetic

Adversarial Search

	סיבוכיות זמן ריצה	סיבוכיות זכרון
Minimax	$O(b^m)$	אם שומרים לכל פעולה שהיריב יעשה את התגובה האופטימלית, אז $O(b^m)$ ואם משתמשים ב-DFS עבור כל פעולה (מחדש) אז $O(bm)$
alpha-beta pruning	עם " "almost perfect ordering $O(b^{m/2})$ עומק החיפוש יכול להיות כפול	
Expectiminimax	$O(b^m b^n)$	

פרמטרים:

b = branching factor
m = Number of moves (depth)
n = the number of distinct rolls