Sokoban

Jan Kuchař
ČVUT–FIT
kuchaj19@fit.cvut.cz

26. května 2024

1 Úvod

Tato semestrální práce se zabývá logickou retro hrou s názvem Sokoban. Hlavním záměrem bylo nalezení algoritmu, který by našel efektivní řešení, případně zda řešení vůbec existuje.

2 O problému podrobněji

Sokoban je starší hra, kde hráč posouvá krabice na určené pozice, přitom může posouvat nanejvýš jednu krabici najednou a nemůže ji posunout do zdi, sám nemůže vstoupit do zdi ani do krabice.

Sokoban je NP-těžký problém, [1] což mi velmi stížilo práci. Ve své podstatě neexistuje efektivní algoritmus, který by určil, zda je úroveň řešitelná a jak nejefektivněji, či nikoliv. Musel jsem tedy zvolit přístup brute-force.

3 Použitý algoritmus

Pro nalezení řešení načtené úrovně jsem použil algoritmus BFS, kde vrcholy představuje dvojice obsahující dvojici, představující pozici hráči, a n-tici dvojic, představující pozici n krabic. Orientované hrany mezi těmito vrcholy jsou různé akce hráče, a to krok nahoru, dolu, doprava a doleva.

Tento algoritmus v případě, že level má řešení v n krocích, potřebuje projít v nejhorším případě 4^n vrcholů. V opačném případě potřebuje projít úplně všechny dosažitelné vrcholy, teda teoreticky všechny možné kombinace pozice hráče a krabic. Vzorcem, pokud má úroveň velikost $n \times m$ a má k krabic, tak potřebuje projít $\binom{n \times m}{k+1}$ vrcholů.

Naštěstí jsou tyto odhady sníženy pomocí nevalidních a nedosažitelných pozic a nevalidních pohybů. Také pro některé pohyby nemá smysl se vracet ihned zpátky, což je v BFS zohledněno (je to již navštívený vrchol).

4 Výsledky

S velmi primitivními levely si algoritmus poradí výborně. Testovací úrovně a úrovně 1 a 2 jsou vyřešeny dost rychle. Začne to ale kulhat s úrovní 3, kdy její vyřešení potřebuje program asi tak 15 vteřin. Úroveň z původního Sokobanu, která má optimálně 230 kroků, řešil tento algoritmus tak dlouho, že jsem vzdal měření.

5 Závěr

Zkusil jsem si poprvé něco naprogramovat v pythonu a naučil jsem se používání knihoven jako je numpy a tkinter. I když je použitý algoritmus pomalý, jsem s implementací celého programu spokojený.

Je zde obrovský prostor pro vylepšení algoritmu. Lze přidat algoritmus pro vypočítání vah hran, a těchto vah pak využít. Také lze implementovat jednoduchý dead-lock detection, kdy je z nějaké pozice krabic nemožné je posunout do cílů (např. 4 krabice ve čtverci nebo krabice v rohu).

Reference

[1] Michael Fryers; Michael Greene. Sokoban. online, 1995. (str. 25–32) [cit. 2024–05–26] `https://www.archim.org.uk/eureka/archive/Eureka-54.pdf#page=28`.