Automatizované riešenie úloh o dláždení polyominami

Ročníkový projekt, letný semester 2020/2021

Dávid Mišiak
školiteľ doc. RNDr. Ján Mazák, PhD.

1 Predchádzajuci semester

Stav po zimnom semestri je popísaný v reporte zo zimného semestra.

2 Nová funkcionalita

2.1 Tvary zadané obvodom

Pridal som štvrtý vstupný formát – tvary zadané obvodom. Vstupný reťazec popisuje obvod útvaru, pričom písmená `U`, `D`, `R`, `L`, znamenajú postupne kroky hore, dole, doprava a doľava. Teda napríklad vstup `DDRRUULL` popisuje štvorec 2×2 políčka.

Obvod sa musí začínať a končiť na tom istom mieste a okrem tohto bodu sa nikdy nesmie dotknúť sám seba. Útvary s dierami sa v tomto formáte nedajú zadať.

2.2 Generovanie obrázkov

Ak dláždiaci problém má riešenie, po vyriešení si ho môžeme nechať výpísať na konzolu, nebude to však veľmi prehľadné. Program preto po novom vie vytvoriť aj obrázok s riešením vo formáte SVG (použil som drobnú knižnicu SVGWrite[1]).

2.3 Prevod na SAT

Do programu som integroval až dva špičkové SAT solvery – CaDiCaL[2] a CryptoMiniSat[3]. Vďaka tomu bude možné konforontovať výsledky jedného SAT solvera s výsledkami druhého (napríklad pri debugovaní).

SAT solvery sú do programu linkované staticky, vďaka čomu je skompilovaný program relatívne slušne portabilný.

Pre začiatok som zvolil pomerne jednoduchý algoritmus prevodu problému dláždenia na problém splniteľnosti. Pre každú možnú pozíciu a otočenie každého kusu každého dieliku vytvoríme jednu logickú premennú, ktorá bude pravdivá práve vtedy, ak sa daný kus daného dieliku vyskytuje na danej pozícii v danom otočení. Pre každý kus každého dieliku teda potrebujeme pridať sadu klauzul, ktoré zaručia, že najviac jedna z premenných daného kusu je pravdivá – teda sa daný kus nachádza na najviac jedne pozícii na doske.

Okrem toho potrebujeme zaručiť, že je celá doska pokrytá a že sa žiadne dva dieliky neprekrývajú. Stačí nám pre každé políčko dosky pridať sadu klauzul, ktoré zaručia, že dané políčko je pokryté práve jedným kusom jedného dielika, teda že je pravdivá práve jedna z premenných prislúchajúcich kusom dielikov, ktoré pokrývajú dané políčko.

Vzťah „aspoň jedna z daných n premenných je pravdivá“ vieme vyjadriť jednoduchou disjunkciou premenných. Stačí nám už iba vyjadriť „najviac jedna z daných n premenných je pravdivá“ – to sa dá mnohými spôsobmi, najjednoduchší z nich však vyžaduje až kvadraticky veľa klauzul. Namiesto neho som použil Commander-Variable techniku[4], ktorá pridá len $O(n)$ klauzul za cenu $O(n)$ nových premenných.

[1] `https://gitlab.com/dvd0101/svgwrite`
[2] `https://github.com/arminbiere/cadical`
[3] `https://github.com/msoos/cryptominisat`
[4] `https://www.cs.cmu.edu/~wklieber/papers/2007_efficient-cnf-encoding-for-selecting-1.pdf`

2.4 Meranie výkonu

Na meranie času riešenia problémov pomocou jednotlivých backendov som použil knižnicu Benchmark[5] od Googlu.

Nateraz som zostavil len pomerne malú sadu vstupov na benchmarking. Pri aktuálnej implementácii sa jednoduchý backend založený na backtrackingu (prekvapivo) javí byť takmer vždy značne rýchlejší než SAT solvery.

V tabuľke sú uvedené CPU časy vykonávania v milisekundách. Zadania problémov sa nachádzajú v repozitári, v adresári `benchmark/data`.

problém	backtracking	CaDiCaL	CryptoMiniSat
01	0.002	0.115	0.335
02	0.021	36.6	34.1
03	0.085	52.4	42.7
04	0.065	1716	371
05	2.62	17483	222
06	1.34	2737	690
07	2522	11948	38487
08	319	15822	111727
09	0.003	37.3	40.4
10	0.383	14118	104942

Závratný rozdiel napríklad pri probléme 10 vieme ľahko vysvetliť tým, že náš SAT problém obsahuje veľa symetrií – SAT solver napríklad nevie, že všetky kusy rovnakého domina sú naozaj rovnaké, a teda nemusí pre každé rozloženie „skúšať“ všetky permutácie týchto rovnakých kusov (backtrackingové riešenie týmto netrpí). Bolo by preto potrebné tieto symetrie rozbiť pridaním nových klauzul (či úplne iným prevodom na SAT problém).

Tieto symetrie však nevysvetľujú napríklad problém 06, kde sú všetky dieliky rôzne a z každého máme len jeden kus. Tu by som od SAT solverov očakával lepší výkon.

3 Ďalšie kroky

Ak bude tento projekt pokračovať ako bakalárska práca, plánujem sa venovať naslejúcim bodom:

- Zostaviť lepšiu sadu problémov na benchmarking (prípadne aj rozšíriť testy).
- Lepšie si naštudovať štandardné konštrukcie SAT problémov a pokúsiť sa vylepšiť výkon SAT backendov.
- Pridať nové backendy a spraviť poriadne porovnanie.

[5] `https://github.com/google/benchmark`