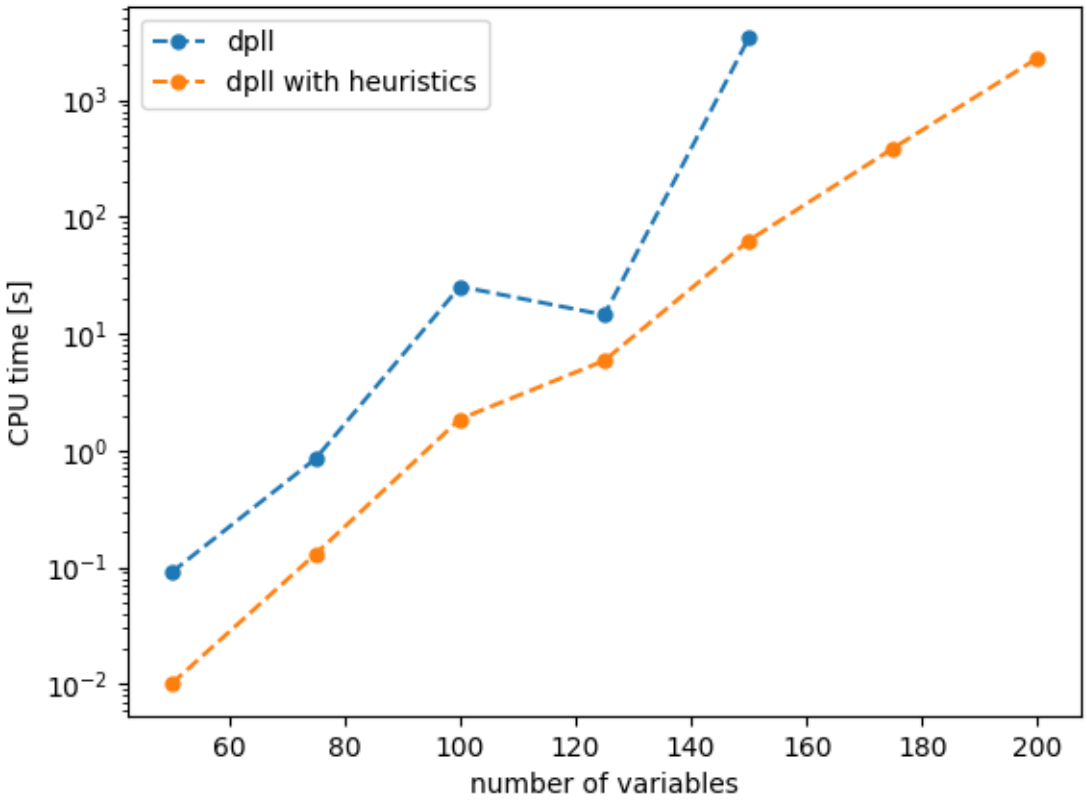
Report - DPLL solvery

David Surma

Má implementace obsahuje celkem 3 druhy DPLL solverů, první používá jen adjacency lists, druhý navíc používá jednoduchou heuristiku pro vybírání dalšího literálu a třetí používá watched literals bez rozhodovací heuristiky. Pro vyhodnocení jejich výsledků jsem používal instance 'uuf{x}-01.cnf' s počtem proměnných od 50 do 150, případně až 200.

Následující graf zachycuje závislost mezi velikostí instance SAT problému a dobou běhu solveru. Zároveň porovnává výsledky solveru s a bez jednoduché heuristiky. Svislá osa má logaritmické měřítko.

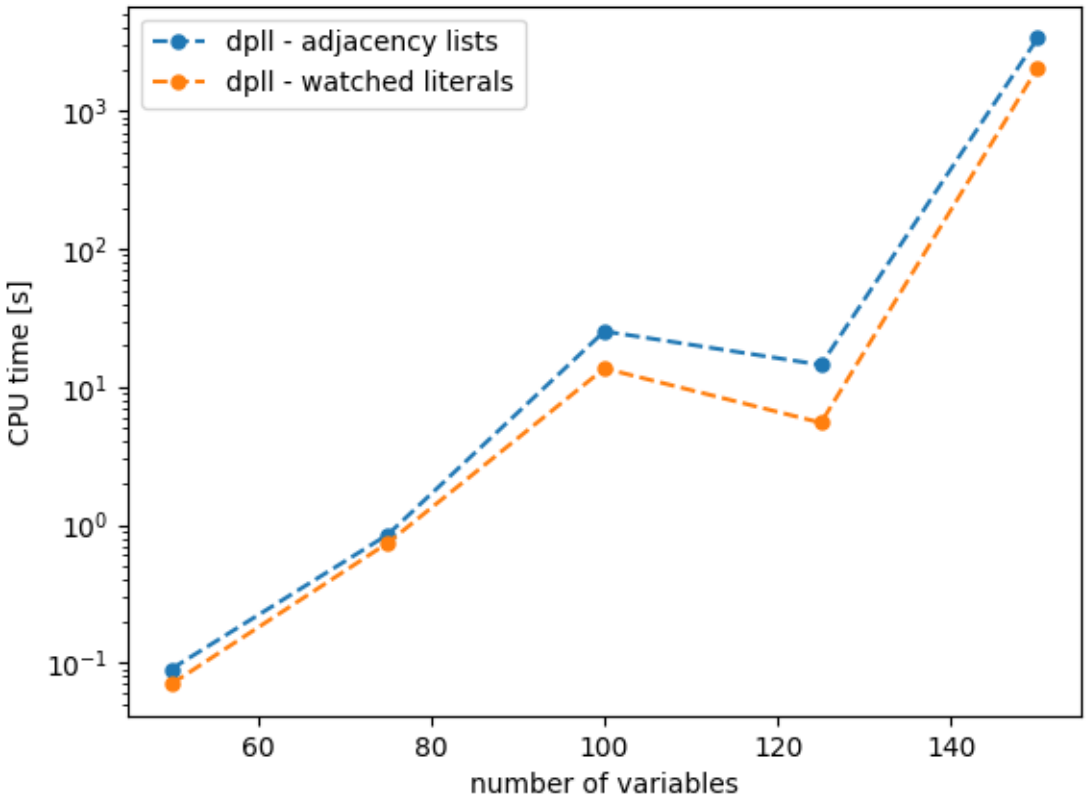


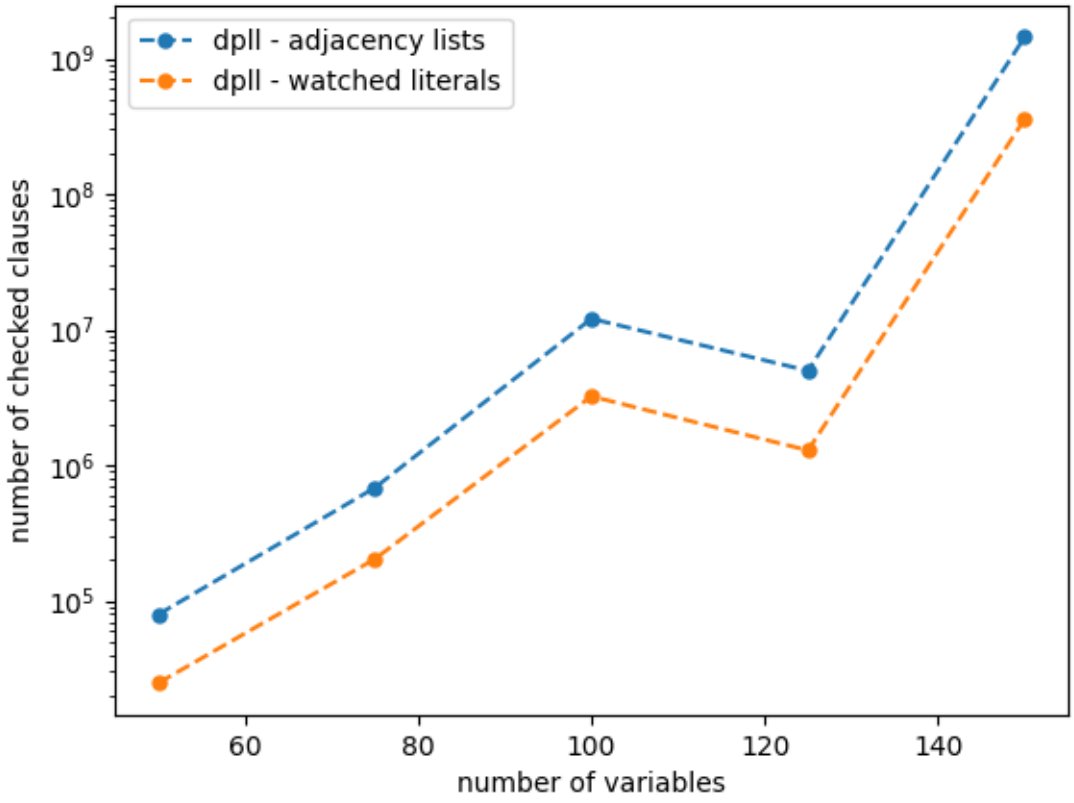
V grafu vidíme, že se zvětšující se velikostí instance problému roste doba běhu exponenciálně. Jediná zvláštní věc nastala u instance obsahující 125

proměnných, jejíž nesplnitelnost byla ověřena u obyčejného DPLL solveru rychleji, než menší instance se 100 proměnnými. Konkrétní použitá instance se 125 proměnnými musela být i přes větší velikost jednodušší na vyřešení. Průkaznějších výsledků bychom mohli dosáhnout použitím více instancí stejné velikosti a zprůměrováním jejich výsledků.

Každopádně jde vidět, že použitím jednoduché heuristiky, která při rozhodování vybírá literály pouze z nejkratších nesplněných klauzulí, se doba běhu řádově zmenší.

Následující grafy zachycují závislost mezi velikostí instance SAT problému a dobou běhu solveru, respektive počtem kontrolovaných klauzulí, pro solvery používající adjacency list a watched literals. Svislé osy mají logaritmické měřítko.





Vidíme, že použití watched literals je obecně lepší než adjacency list. Zatímco počet kontrolovaných klauzulí je pro watched literals menší asi 5x oproti adjacency lists, doba běhu je jen asi 2x menší. To je způsobeno overheadem pro kontrolování jednotkových a nesplněných klauzulí pomocí watched literals.