Software Analysis & Verification **Due: Mar 17, 2020**

PA1 Report

Instructor: Fei He
TA: Jianhui Chen, Fengmin Zhu

沈冠霖 (2017013569)

实现思路 带backjump的DPLL整体上是在DPLL基础上实现的。
相比用backtrack的DPLL，带backjump的DPLL只是添加了冲突图的维护，以及将backtrack替换成backjump。
backjump算法实现的基本思路和课件一样：维护一个冲突图，在冲突的时候对冲突图求割，然后进行对应的回退和添加对应的子句。
为了简化求割的步骤，减少开销，我将每次选的reason侧定义为这个连通子图的所有决策节点，选取的K节点集合也就是这个连通子图的所有决策节点。这样可以保证保证所有决策节点在reason侧，至少一个冲突文字在另一侧，而且选取的节点都有边连接另一侧。同时，这种选取方法能够大大简化求割的步骤。
这样实现的话，我的冲突图其实并不需要建成一个图，我只需要记录每个点的文字和它的根本祖先（产生它的决策节点集合）即可。遇到冲突的时候，把冲突的两个文字的根本祖先集合求并，就是选取的节点集合了。

测试环境 CPU:Inter Core i5-6300HQ,2.3GHZ
内存：12G
环境：ubuntu18.04, release模式

测试结果 ：虽然对于很多没有多少回溯的数据，backjump是更慢的，因为它有维护冲突图的开销，在backjump步骤中进行集合求并的开销也不小。但是对于回溯频繁的数据，backjump更快。
我选取了测试集合的三组数据，分别是test9.dimacs(a graph of 6 vertices with a 3-clique and a 4-coloring),test11.dimacs(Pigeonhole principle formula for 6 pigeons and 7 holes),test12.dimacs(5 vertices with a 3-clique and a 2-coloring)。对于这三组数据，我的backjump远远快于backtrack。

Table 1: backtrack与backjump在三组数据的结果比较(时间单位为ms)

数据	算法	1	2	3	4	5	平均
test9	backtrack	865.871	855.808	857.898	864.609	853.403	859.52
	backjump	15.5159	14.7249	13.9436	14.8857	13.7598	14.57
test11	backtrack	21.4267	18.2216	16.9051	19.796	17.3519	18.74
	backjump	7.19779	6.2324	6.7309	6.7755	8.06141	7.00
test12	backtrack	636.489	630.947	630.145	629.309	628.896	631.76
	backjump	81.2241	78.3284	78.7136	79.3428	78.1387	79.15

三组数据，backjump的时间分别是backtrack的$\frac{1}{60}, \frac{3}{8}, \frac{1}{8}$，可以充分说明，对于这三个问题，backjump比backtrack快很多。