Ricardo Simas, 72769 Nuno Mendes, 73716 Prof. Luís Custódio

A versão de Python usada foi a 3.4. O código encontra-se distribuído por vários ficheiros. Para executar o código é necessário correr o ficheiro main.py da seguinte forma:

$ python main.py argumentos

Os argumentos existentes são: -f *ficheiro* em que *ficheiro* corresponde ao ficheiro que se pretende testar (caso se pretendam testar múltiplos ficheiros numa directoria, é possível colocar em *ficheiro* a directoria adicionando -b como argumento); -a *algoritmo* em que *algoritmo* corresponde ao algoritmo que se pretende testar (*gsat*, *walk* ou *dpll* para Greedy SAT, WalkSAT e DPLL respectivamente); se o algoritmo for gsat então é necessário colocar dois valores seguidos a gsat que indicam os parâmetros iniciais *max_restarts* e *max_climbs*; se o algoritmo for *walk* então é necessário colocar dois valores seguidos a *walk* que indicam os parâmetros iniciais *p* e *max_flips*; -o permite escrever o resultado em ficheiros de saída com o mesmo nome dos ficheiros de entrada e adição '_sol' ao nome; -t permite visualizar no terminal o tempo que o algoritmo demorou a correr; -c *clauses* em que *clauses* é o número de cláusulas do ficheiro que se pretende analisar (sem –c *clauses* todas as cláusulas são analisadas); -d permite visualizar o resultado no terminal, a eficácia do algoritmo (número de ficheiros com solução face ao número de ficheiro total), se for em modo –b, ou se a sentence é satisfazivel por defeito. A ordem dos argumentos é irrelevante.

(i) Um algoritmo *sound* garante que, se for retornada uma solução para um problema, esta solução será verdadeira. Um algoritmo *complete* retorna sempre uma solução, seja esta válida ou não. Um algoritmo não necessita de retornar sempre uma solução mas se a retornar esta deve ser verdadeira para o problema em questão. Por este motivo um algoritmo não tem que ser necessariamente *complete* mas deve ser obrigatoriamente *sound*.

O algoritmo GSAT apresenta aleatoriedade que o torna imprevisível. Por este motivo pode ser definido como um algoritmo estocástico. Isto acontece porque os valores iniciais dos símbolos existentes na *sentence* são atribuídos de forma aleatória. De seguida, uma função de avaliação testa o número de cláusulas satisfeitas e varia os valores dos símbolos, individualmente e de forma a maximizar o número de cláusulas satisfeitas, até se atingir uma solução (todas as cláusulas serem satisfeitas). Caso não seja atingida uma solução, são atribuídos novos valores aos símbolos de forma aleatória repetindo-se o processo. Este tipo de procura é incompleto pois nem todas as possibilidades são pesquisadas. A solução retornada pelo GSAT é válida mas como não são testadas todas as possibilidades, é possível que não seja retornada uma solução. Desta forma é possível concluir que o GSAT é *sound* mas não é *complete*.

O algoritmo WalkSAT também é estocástico. A imprevisibilidade deve-se à atribuição aleatória inicial de valores aos símbolos. É também seleccionada uma cláusula, que não seja satisfeita pelos valores presentes de símbolos, de forma aleatória. Possivelmente, é ainda escolhido um símbolo, de forma aleatória, dessa cláusula para que o seu valor seja alterado. Através desta descrição verifica-se que o WalkSAT apresenta vários elementos de aleatoriedade, pelo que, à semelhança do GSAT, pode não chegar à solução do problema. Assim verifica-se que o algoritmo é *sound*, pois as soluções que retorna são correctas, mas não é *complete*, pois nem sempre retorna uma solução.

O algoritmo DPLL atribui valores a símbolos de acordo com a existência de símbolos puros e cláusulas unitárias. Caso estes não existam é atribuído um valor inicial a um símbolo. Desta forma os valores de símbolos são atribuídos apenas com base em restrições existentes e hipóteses, não com base em aleatoriedade. Por isso o algoritmo é determinístico. Caso exista uma clausula que seja falsa pelo modelo, realiza-se *backtrack*. O algoritmo corre até ser encontrada uma solução, se esta existir. Por este motivo é possível concluir que o algoritmo é *sound* e *complete*.

O algoritmo DPLL é o único destes 3 que permite avaliar se uma sentence é satisfazível ou não.

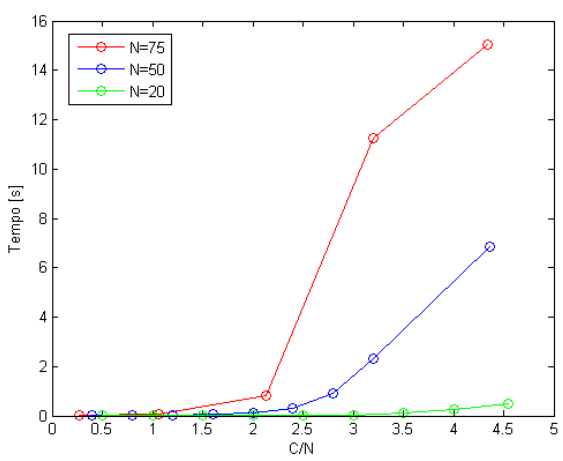
(ii) A implementação dos algoritmos encontra-se no ficheiro solver.py.

(iii)/(iv) Os três algoritmos foram testados com múltiplos problemas 3SAT. Os ficheiros de entrada 3SAT testados encontram-se na pasta **test_files**. Os ficheiros testados para cada algoritmo estão identificados nesta secção. É de notar que devido ao elevado número de ficheiros de teste (ficheiros esses testados com os três algoritmos, com diferentes parâmetros iniciais e com variações no número de cláusulas), decidiu-se não enviar todos os ficheiros de saída obtidos. Os ficheiros de saída enviados contêm um nome idêntico aos respectivos ficheiros de entrada mas com a terminação '_sol'.

Para testar os possíveis efeitos do número de cláusulas e do número de símbolos foram definidos os seguintes parâmetros da experiência: agrupou-se (por pastas) todos os ficheiros com igual número de símbolos; para cada grupo de ficheiros testou-se cada um dos três algoritmos (os parâmetros iniciais do WalkSAT e do GSAT mantiveram-se constantes); os testes em cada grupo de ficheiros foram realizados variando o número de cláusulas (argumento **–c clauses**); para cada teste registou-se o tempo médio para cada ficheiro; para cada teste registou-se a eficácia (número de ficheiros com solução certa face ao número total de ficheiros na pasta) encontrando-se o seu valor entre 0 e 1.

Para os três algoritmos foram testados os ficheiros com 20, 50, 75 e 100 símbolos (pastas identificadas com 20vars, 50vars, 75vars e 100vars em **test_files**). O número de cláusulas nos ficheiros foi variado de forma a ser possível relacionar o rácio C/N (nº cláusulas/nº símbolos) com o tempo de execução de cada algoritmo.

Os resultados experimentais para o algoritmo GSAT encontram-se nas figuras 1a e 1b.



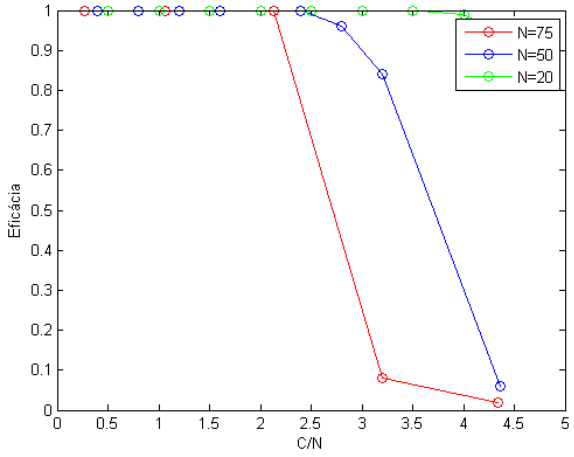
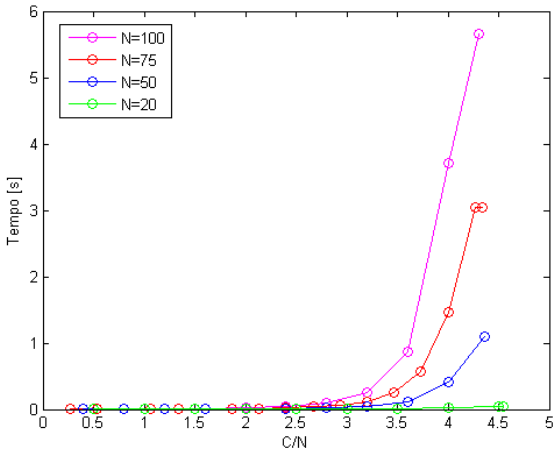


Figura 1a – Efeito do rácio C/N sobre o tempo de execução do GSAT. Figura 2b – Eficácia do GSAT face a C/N.

Os resultados experimentais para o algoritmo WalkSAT encontram-se nas figuras 2a e 2b.



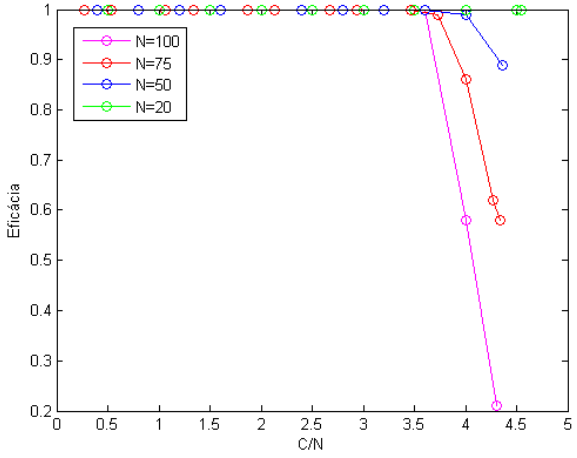


Figura 2a – Efeito do rácio C/N sobre o tempo de execução do WalkSAT. Figura 2b – Eficácia do WalkSAT face a C/N.

Os resultados experimentais do algoritmo DPLL encontram-se na figura 3.

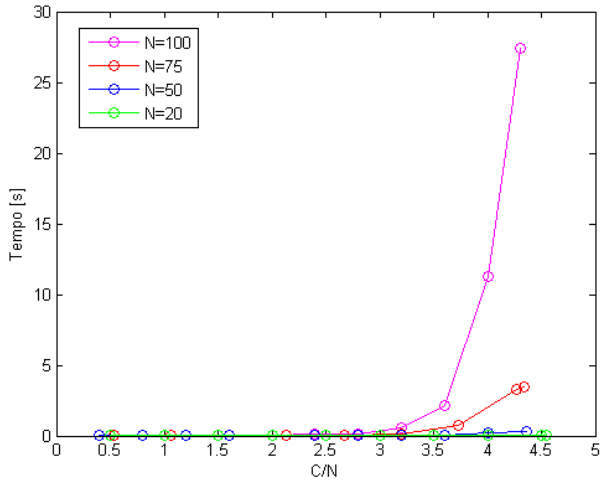


Figura 3 – Efeito do rácio C/N sobre o tempo de execução do algoritmo DPLL.

Os resultados apresentados nas figuras 1, 2 e 3 são correspondentes a problemas satisfazíveis. A partir dos testes efectuados para diversos problemas 3SAT, verifica-se que, para os três algoritmos, quanto maior for o rácio C/N, maior é o tempo de execução do algoritmo. Tal deve-se ao facto de um aumento do número de cláusulas face ao número de símbolos se traduzir num aumento de restrições para cada símbolo. Quanto mais restrições existirem, maior a dificuldade em encontrar uma hipótese viável que solucione o problema, portanto maior o número de testes até uma solução ser encontrada e por isso o tempo de execução dos algoritmos é superior. Note-se ainda que um maior número de símbolos e de cláusulas obriga os algoritmos a processar mais dados em cada ciclo. Pelas figuras 1a, 2a e 3 verifica-se que o aumento de tempo com o aumento de C/N é exponencial. Isto significa que o aumento de C/N aumenta a complexidade do problema cada vez mais à medida que o rácio entre cláusulas e símbolos é superior.

Não são demonstrados resultados numéricos dos problemas sem solução para o algoritmo *dpll*, apesar destes problemas terem sido testados. Tal deve-se ao facto de não ser possível alterar o número de cláusulas em ficheiros sem solução uma vez que, ao fazê-lo estes ficheiros passam a ter uma solução. Por este motivo apenas foi possível testar *unsatisfiable sentences* dos problemas *benchmark* que se encontram no website indicado no guia do laboratório. Estes problemas apresentam rácios C/N semelhantes pelo que não é possível apresentar uma relação entre o tempo de execução e o rácio C/N numa imagem.

Perante os resultados obtidos é possível concluir que o algoritmo GSAT apresenta severas dificuldades em solucionar problemas 3SAT. Mesmo para valores não muito elevados de N e de C, o algoritmo demora muito tempo a gerar uma solução e pode nem a conseguir encontrar tendo uma baixa eficácia como evidenciado na figura 1b. O algoritmo tem uma grande dificuldade em encontrar a solução para problemas satisfaziveis e é necessário efectuar várias tentativas para encontrar uma solução, ou seja, é necessário usar um número elevado de parâmetros iniciais *max restarts* e *max climbs* para achar uma solução.

Quanto ao algoritmo WalkSAT é possível concluir que o algoritmo tem uma elevada eficácia mas se o número de cláusulas e símbolos for muito elevado, este apresenta dificuldades em encontrar uma solução, apesar de demorar menos tempo que o dpll em média.

O DPLL, embora eficaz, consome bastante tempo para um elevado número de cláusulas, símbolos e do rácio entre ambos. Este algoritmo é o que apresenta uma maior relação exponencial entre o tempo de execução e o rácio C/N.

Para os três algoritmos verifica-se que, embora o rácio C/N tenha influência no tempo de execução, o principal factor no tempo de execução e maior dificuldade na procura de uma solução, é o número de clausulas e de símbolos e não o seu rácio.

Entre o GSAT e o WalkSAT, o WalkSAT é mais eficiente em termos de tempo. O GSAT testa valores para símbolos que são gerados aleatoriamente e sem ter em consideração as cláusulas inicialmente. O WalkSAT por outro lado

tem em consideração a existência de cláusulas falsas inicialmente. Ambos os algoritmos alteram valores de símbolos de forma a maximizar o número de cláusulas verdadeiras mas o comportamento do WalkSAT é mais completo que o do GSAT pois tem em consideração mais factores que o aproximam da solução cada vez que recomeça a atribuição aleatória de valores a símbolos. Desta forma conclui-se que o WalkSAT é o melhor algoritmo em termos de eficiência temporal para valores elevados do número de símbolos e de cláusulas enquanto que o DPLL é o melhor algoritmo para menores valores do número de cláusulas e de símbolos.

(v) O algoritmo DPLL testa no pior caso, todas as possibilidades para os valores de símbolos até encontrar uma solução, caso esta exista. Os algoritmos GSAT e WalkSAT testam um número limitado de possibilidades, número esse que depende dos parâmetros iniciais dos algoritmos.

A relação entre a eficácia do algoritmo GSAT e o rácio C/N encontra-se na figura 1b, enquanto que a relação entre a eficácia do algoritmo WalkSAT e o rácio C/N encontra-se na figura 2b.

O algoritmo GSAT tem uma baixa eficácia para valores elevados de cláusulas e de símbolos. O algoritmo tem uma grande dificuldade em encontrar a solução para problemas satisfazíveis e é necessário efectuar várias tentativas para encontrar uma solução, ou seja, é necessário usar um número elevado de parâmetros iniciais *max restarts* e *max climbs* para achar uma solução.

O algoritmo WalkSAT apresenta uma maior eficácia que o GSAT porque embora também seja estocástico, a atribuição de valores a símbolos tem em consideração a existência de cláusulas não satisfeitas, algo que não acontece no GSAT, pelo que o WalkSAT é mais completo e eficaz.

A eficácia do algoritmo DPLL é sempre 1, isto é, o algoritmo encontra sempre uma solução para qualquer problema. Porém tal nem sempre é benéfico. Caso o número de cláusulas e símbolos seja muito elevado e sobretudo se também não existir solução para o problema, testar bastantes possibilidades pode levar a que o algoritmo corra ao longo de um tempo demasiado elevado (tempo exponencial). Por este motivo o algoritmo DPLL não é sempre computacionalmente viável.

Nem todas as hipóteses são testadas nos algoritmos GSAT e WalkSAT, uma vez que o número de testes é limitado pelos parâmetros iniciais e pelo facto dos testes efectuados se poderem repetir por os algoritmos serem estocásticos (atribuição de valores a símbolos é parcialmente aleatória), ambos os algoritmos são menos eficazes que o DPLL. Uma melhor eficácia é garantida aumentando o número de testes efectuados, o que implica um maior consumo de tempo e de recursos computacionais.