CORK INSTITUTE OF TECHNOLOGY

INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Metaheuristic Optimization

Assignment 2

Due date:

Assignment should be submitted to Canvas before 10PM **Tuesday Dec 17th**

As per CIT regulations, submitting within 7 days of the deadline will result in a 10% penalty, between 7 and 14 days late will result in a 20% penalty, and later than 14 days after the due date will result in a 100% penalty applied.

SAT

The Propositional Satisfiability Problem (SAT) can be represented by a pair F = (V, C) where, V indicates a set of Boolean variables and C a set of clauses in conjunctive normal form (CNF). The following formula shows a SAT example described by means of the CNF.

$F = (x_1 \vee x_3 \vee -x_4) \wedge (x_4) \wedge (x_2 \vee -x_3)$

Where $\vee$ represents the ***or*** Boolean connective, $\wedge$ represents **and,** and $-x_i$ is the negation of x_i. Given a set of clauses, $C_1, C_2, \ldots, C_m$ on the variables $x_1, x_2, \ldots, x_n$, the satisfiability problem is to determine if the formula

$C_1 \wedge C_2 \wedge \ldots \wedge C_m$

is satisfiable. That is, is there an assignment of values to the variables so that the above

formula evaluates to *true*?.

For instance, a potential solution for F would be

x_1=True, x_2=False, x_3=False, x_4=True.

x_1 satisfies the first clause, x_4 satisfies the second clause, and $-x_3$ satisfies the last clause.

Local Search

Algorithm 1 describes a traditional local search algorithm for SAT solving. It starts with a random truth assignment for the variables in F, and the the local search algorithm is depicted in lines (3-9) here the algorithm flips the most appropriate variable candidate until a solution is found or a given number of flips is reached (MaxFlips), after this process the algorithm restarts itself with a new (fresh) random assignment for the variables.

As one may expect, a critical part of the algorithm is the variable selection function (select-variable) which indicates the next variable to be flipped in the current iteration of the algorithm. The WalkSAT algorithm introduces a diversification strategy in order to avoid local minimums. The algorithm starts by selecting, uniformly at random, an unsatisfied clause c and then picks a variable from c. The variable that is generally picked will result in the fewest previously satisfied clauses become unsatisfied, with some probability of picking a random variable from c.

Algorithm 1 Local Search For SAT (CNF formula F, Max-Flips, Max-Tries)

```
 1: for try := 1 to Max-Tries do
 2:    A := initial-configuration(F).
 3:    for flip := 1 to Max-Flips do
 4:       if  A satisfies F then
 5:          return A
 6:       end if
 7:       x := select-variable(A)
 8:       A := A with x flipped
 9:    end for
10: end for
11: return 'No solution found'
```

I/O Specification

In this assignment you will use the same I/O format as defined in the lab work of Week 8.

GWSAT (30 Marks)

In this part of the assignment you will extend the popular GSAT algorithm with random walk. The basic GSAT algorithm will always flip the best neighbour (with ties broken randomly) in each iteration, **even if this results in a worse state than current**. The best neighbour is defined as the one that has largest **net gain**, i.e. largest value for $B_0 – B_1$ where B_0 is the number of clauses that are currently unsat, and B_1 is the number of clauses that would be unsat after flipping the variable.

The random walk component is that for each step of the algorithm, the variable to be flipped is selected randomly (from all variables involved in at least one unsatisfied clause) with probability *wp*, and otherwise will be the one whose flip which has largest net gain as described above.

Use the following baseline parameters for evaluating on the three instances given (uf20-1,uf20-2, uf50-1):

1000 iterations per restart
10 restarts
wp=0.4
30 executions (how many runs with different seeds, similar to doing 5 runs in the previous assignment).

Additional evaluation should vary these parameters and discuss/explain the impact on results.

Your code must be have a main python file named MySurname_MyID_GWSAT.py which can be run from the command line and takes the following arguments:

- ***an instance***
- ***#Executions***
- ***#Restarts***
- ***#Iterations***
- ***wp***

e.g. for above settings

```
python Grimes_1234_GWSAT.py uf20-01.cnf 100 1000 10 0.4
```

WalkSAT\SKC with Tabu Search(45 Marks)

Extend WalkSAT with Tabu search (without aspiration criteria). WalkSAT variable selection step is as follows:

1. Select an unsatisfied clause BC (uniformly at random)
2. If at least one variable in BC has negative gain of 0 (i.e. flipping the variable does not make any clause that is currently satisfied go unsat), randomly select one of these variables.
3. Otherwise, with probability p, select random variable from BC to flip, and with probability (1-p), select variable in BC with minimal negative gain (break ties randomly)

In this context, after a variable *x* has been flipped, it cannot be flipped back within the next *tl* steps (the tabu tenure tl is a parameter of your algorithm). For each step of the algorithm, the variable to be flipped is selected like in the basic WalkSAT\SKC, being the one whose flip minimizes the number of unsatisfied clauses, except that the choice is restricted to variables that are currently not Tabu.

Use the following baseline parameters for evaluating on the three instances given (uf20-1,uf20-2, uf50-1):

tl = 5 steps
p=0.4
1000 iterations limit
10 restarts

Additional evaluation should vary these parameters and discuss/explain the impact on results.

Your code must have a main python file named MySurname_MyID_WalkSAT.py which can be run from the command line and takes the same arguments as GWSAT above with the addition of final argument of the tabu length, e.g. for above settings:

```
python Grimes_1234_WalkSAT.py uf20-01.cnf 100 1000 10 0.4 5
```

Run-Time Distribution Evaluation (25 Marks)

Analyse and compare the RTDs (with at least 100 executions) of GWSAT and WalkSAT+Tabu for uf20-020.cnf and uf20-021.cnf.

Rubric and Assignment of Marks:

Coding:

	The algorithm is logically well designed and efficient without inappropriate design choices (e.g., unnecessary loops). Code is well commented.	The algorithm always works properly and meets the specification. Code is clean, understandable and well organized.	The algorithm works properly in limited cases	The algorithm is incorrectly implemented
GWSAT	(16 - 20 Marks)	(11 - 15 Marks)	(6-12 Marks)	(0-5 Marks)
WalkSAT + Tabu	(27 - 35 Marks)	(15 - 26 Marks)	(6-14 Marks)	(0-5 Marks)

Evaluation:

You need to write a report with a comprehensive description of the experiments and an extensive evaluation (and analysis) of the results. Use tables and figures where appropriate.

Rubric:

	Excellent presentation, depth and insight analysis of the empirical results	Good presentation of the results (e.g., describing the results with well structured tables)	Incomplete and/or unclear presentation of the results	The results are inconsistent with the logic of the configuration/ operators
GWSAT	(7-10 Marks)	(5 6 Marks)	(2-4 Marks)	(0-2 Marks)
WalkSAT + Tabu	(7-10 Marks)	(5-6 Marks)	(2-4 Marks)	(0-2 Marks)
RTDs	(20-25 Marks)	(13-19 Marks)	(6-12 Marks)	(0-5 Marks)

Submission:

This assignment is due on Tuesday, Dec 17th, 2019. You must **submit the pdf via the Turnitin assignment** submission. It must be PDF format.

You must submit the following files (in a single zip file) **SEPARATELY** via **the code submission**:

- All source code.
- A Readme file, which briefly describes all submitted files. In the Readme file, you should also provide information about compiling environment, compiling steps, execution instructions, etc.

Academic Integrity:

This is an **individual** assignment. The work you submit must be your own. In no way, shape or form should you submit work as if it were your own when some or all of it is not.

Collusion: Given how much freedom there is in the assignment, everybody's work will be different. It will be obvious if there is collusion. All parties to collusion will be penalized.

Deliberate plagiarism: You must not plagiarise the programs, results, writings or other efforts of another student or any other third-party. Plagiarism will meet with severe penalties, which can include exclusion from the University.

Inadvertent plagiarism: In reporting your exploration of the research literature be careful to avoid inadvertent plagiarism (e.g where paraphrases of the source material are too close to the original).

Falsification and fabrication: The experimental results reported must come from the experiments that you have run. Do not falsify or fabricate results.

Your report will be checked for signs of collusion, plagiarism, falsification and fabrication. You may be called to discuss your submission and implementation with me and this will inform the grading, any penalties and any disciplinary actions.