Assignment 2

COMP 9058 - Metaheuristic Optimisation

December 1, 2021

Due date:
Assignment should be submitted to Canvas in a single pdf by **Sunday, Dec 19th, 23:59**. 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. However, I will waive the first week penalty so if submitted before Dec 26th there will be no penalty. The 20% penalty will apply for the week thereafter.
The pdf MUST be named *Lastname_Firstname_StudentNumber_MH2.pdf*. Similarly the folder containing the code to be zipped MUST be named *Lastname_Firstname_StudentNumber_MH2_code*.
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. As always, the random number generator **must be seeded to your student number** for your experiments.

1 Assignment Description

1.1 SAT - [75 Marks]

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. $F = (x_1 \vee x_3 \vee \neg x_4) \wedge (x_4) \wedge (x_2 \vee \neg x_3)$
where: $\vee$ represents the logical *OR* operator, $\wedge$ represents logical *AND*, and $\neg x_i$ 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 $\neg x_3$ satisfies the last clause.

Local Search for SAT

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.

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'
```

Figure 1: Outline of Local Search SAT approach

As one may expect, the 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 variants to implement are GSAT, GWSAT, GSAT+Tabu, and a WalkSAT style GSAT (GWalkSAT). GSAT selects the variable with maximum net gain, GWSAT adds a random walk step (with probability w_p select at random a variable involved in at least one currently unsatisfied clause, otherwise select the variable with maximum net gain), and GSAT/Tabu combines the original GSAT with tabu search (select the best variable that isn't currently tabu).
The GWalkSAT algorithm starts by selecting, uniformly at random, an unsatisfied clause C and then picks a variable from C. The variable to choose is either chosen at random with probability w_p or the one with maximum net gain (the same heuristic as for GSAT).

The main file must follow the same naming convention (Lastname_Firstname_StudentNumber_MH2.py) and be created such that it can be run from the command line with parameters passed in for each parameter as follows:

`python Grimes_Diarmuid_R1234_MH2.py inst_file alg nRuns nBadIters nRestarts` w_p `tl`

where

- inst_file: the full file path for the instance

- alg: one of { gsat, gwsat, gtabu, gwalksat}
- nRuns: the number of runs of the algorithm to perform
- nBaditers: is the number of successive iterations without an improving move before restarting
- nRestarts: is the number of restarts
- w_p: is the random walk probability
- tl: the length of the tabu list

The default settings are given as follows (using the command line arguments listed above in that order):

```
python Grimes_Diarmuid_R1234_MH2.py myfolder/uf20-01.cnf gsat 10 20 50 0.1 5
```

Note, here gsat does not use w_p nor tabu search so it will just ignore those parameter values in the code. If your code doesn't run in this exact format, marks will be lost.

You must evaluate your implementation of the four methods (GSAT, GWSAT, GSAT/Tabu, GWalkSAT) on three SAT instances from the uf_20 lib and discuss the results. Further tests involving the parameter settings of nRestarts, nBadIters, w_p and tl should be performed. There should be at least 10 runs per experimental setup.

Instances

You will test your algorithms on three instances of the *uf20-91* set from SATLIB (note this contains 1000 instances, all satisfiable). The id of the three instances you choose must be generated randomly, choosing a number between 1 and 1000, three times having set your seed to your student number. (Note this should **not** be part of your program, just run offline to generate the three ids).
random.seed(12345)
for i in range(3): print(random.choice(range(1,1001)))

1.2 Runtime Distribution - [25 Marks]

You should compare the runtime distributions of two of the SAT algorithms of your choice, with 100 runs each on at least one instance, generating the runtime distribution plots (to be discussed in week 11 lecture).

2 Submission, marking and academic integrity

2.1 Submission:

This assignment is due on Friday, Dec 18th, 2020. You must submit the pdf separately via the Turnitin assignment submission: You must submit the following files (in a single zip file) 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.

Your pdf **MUST** be named Lastname_Firstname_StudentNumber_MH2.pdf (e.g. Grimes_Diarmuid_R001234567_MH2.pdf). Similarly the folder containing the code to be zipped MUST be named *Lastname_Firstname_StudentNumber_MH2_code*.

2.2 Rubics:

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.
	(31-40 Marks)	(21-30 Marks)	(11-20 Marks)	(0-10 Marks)

Evaluation

	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.
TSP	(28-35 Marks)	(18-27 Marks)	(8-17 Marks)	(0-8 Marks)
RTD	(19-25 Marks)	(13-18 Marks)	(6-12 Marks)	(0-5 Marks)

2.3 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. Any online source that is used must be cited, and a

full citation given, e.g. do NOT give "stackoverflow", but the full citation, e.g. https://meta.stackoverflow.com/questions/339152/plagiarism-and-using-copying-code-from-stack-overflow-and-submitting-it-in-an-as. If you are unsure on whether something should be cited, general rule of thumb is to err on the side of caution and include the citation. You can also ask me via email

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 Institute.

Your report and code 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.