**Technical Report**

**Technical Report**

**Stegnography 2nd version**

|  |  |
| --- | --- |
| Prepared by: | Eng. Mohammad Sakka |
|  |  |
| Date: | 26/4/2021 |
|  |  |

**REPORT SENSITIVITY**

Does the report have any of the following sensitivities?

Intended for journal publication YES

Results are incomplete NO

Commercial/IP concerns NO

**pseudocodes**

1. Chromosome fixing pseudocode

There not still need to it since we have modified the crossover pseudocode to produce valid chromosomes

1. Enhanced crossover pseudocode:

|  |
| --- |
| 1. The structure of the chromosome 2. M1M2M3 ….MN 3. The problem we are trying to solve is to build a function named f that receives two chromosomes A, B and generate two chromosomes C, D where each of C, D contains information from A and B 4. [C,D]=F(A,B) 5. Assuming that A is combined of metameric Ai ,i=1…length(A) 6. And B is combined of metameric Bj , j=1 …length(B) 7. Assuming that we have a function called decompose 8. [Ai]=Decompose(A) 9. [Bj]=Decompose(B) 10. Let assume that we selected randomly two metameric one from Ai, a and one from Bj, b 11. There are the following cases to be discussed 12. (we define this as a and b do not share the same block) 13. Then a and b can be part of the new child 15. we avoid adding them to the same child 16. let us assume that have another function classify 17. [shared,non-shared]=classify(Ai,Bj) 18. Child =[] 19. While (not (the size of the message is completed)) 20. we selected randomly two metameric from non-shared 21. remove them from the non-shared 22. add them to the child 23. if (the non-shared is empty) 24. break 25. end 26. end 27. while(the message is not completed ) 28. number=checkforRemainingbits 29. select randomly metameric from shared 30. use its block to complete the message 31. change the code of the number of added bits in the block if needed 32. End |

1. Mutation formula:

mr = mr0\*exp(-genr/maxGen)

where:

mr: mutation ratio, mr0: initial ratio, genr: current generation, maxGen: number of generations

--- note: I have added mutation ratio parameter to ‘mutation’ pseudocode, this parameter takes values from 0 to 1 to give more number of metamerics the mutation chance and achive more exploration.

* By the enhanced pseudocode we aim to achive two goals:

1. The too big time spent by fixing code.
2. Avoid the infinite loop when executing fixing code
3. Make sure that there is no more than one metameric shares its block with an other block in the same chromosome, to negate possibility of message distoration.

* After applying the enhanced crossover code, the results were not satisfying and the benchmark algorithm was the supervisor in terms of MSE
* The possible reason for this worse MSE results in the proposed algorithm is that the proposed algorithm may require more number of image pixels since each chromosome consists of several metamerics and each metameric has different bit plane code.
* In terms of MSE measure, if we want to be fare, we should make sure that the pixels number is the same.
* We have circumvented the above problem by making the bit plane constant so that the pixels number used to hide the message bits are the same in the both two algorithms.

**Execution Parameters**

Metameric Version

1. popSize = 50
2. maxGen = 50
3. mr0 = 0.2// uniform mutation
4. mutation ratio = 0.5
5. mr = mr0\*exp(-genr/maxGen)
6. image dims scaling = [512,512]
7. images: first 10 images from brainTumorDataPublic\_1766 dataset
8. QL = 100
9. T\_nbc = 300
10. Selection mode = ‘roulette wheel’
11. Number of seed = 5
12. Fitness value = -mse //maximization
13. bitPlanes = 1 //means that only the lowest significant can be changed in the pixel.

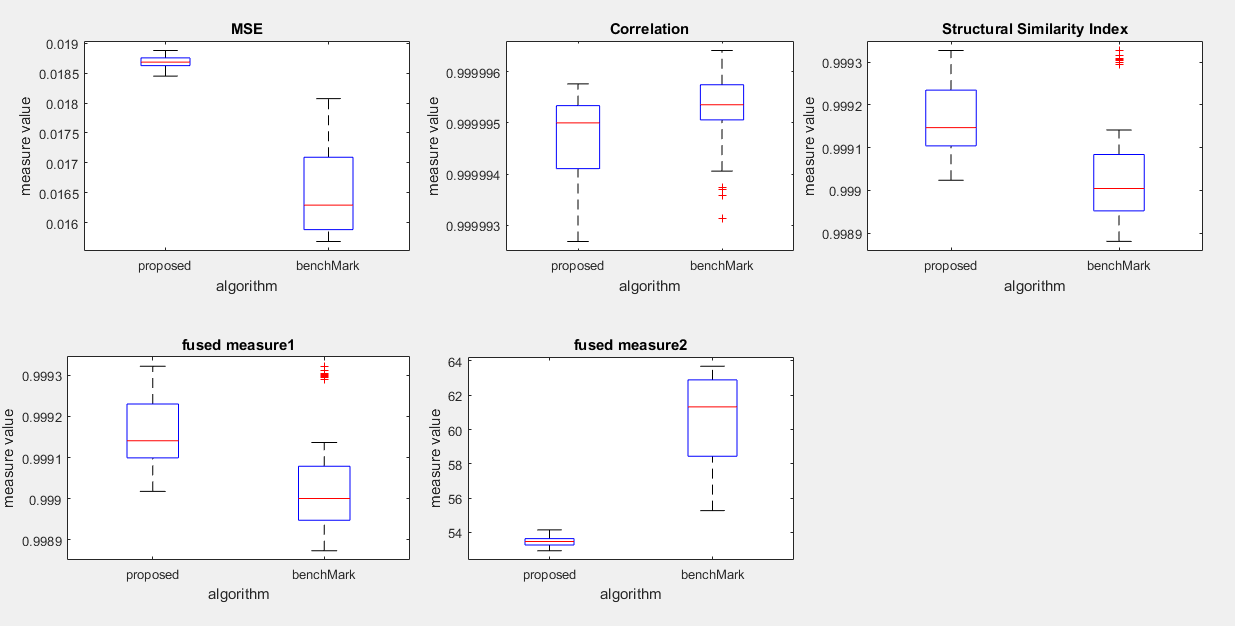
BenchMark Version

1. popSize = 50
2. maxGen = 50
3. gaussian mutation // shrink =1 , fraction = 1
4. image dims scaling = [512,512]
5. images: first 10 images from brainTumorDataPublic\_1766 dataset
6. Selection mode = ‘selectionstochunif’
7. Number of seed = 5
8. Fitness value = mse //minimization
9. bitPlanes = 1 //means that only the lowest significant can be changed in the pixel.

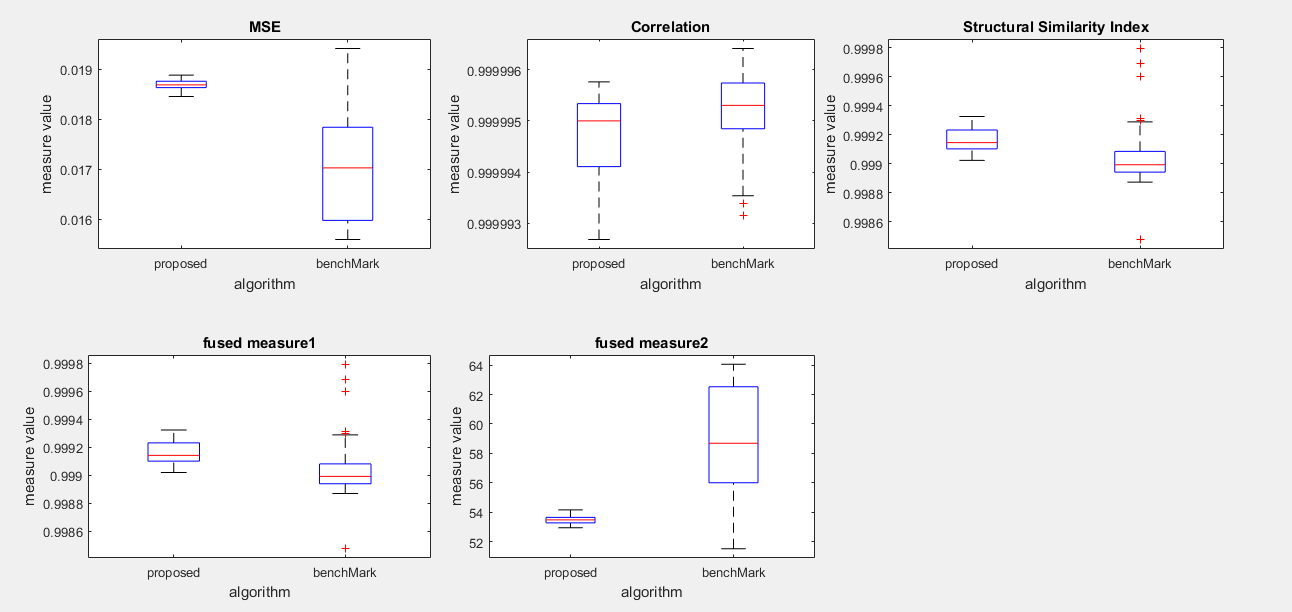
Secret Message:

'Steganography is the practice of concealing a message within another message or a physical object. In computing/electronic contexts, a computer file, message, image, or video is concealed within another file, message, image, or video. The word steganography comes from Greek steganographia, which combines the words steganَs (????????), meaning "covered or concealed", and -graphia (?????) meaning "writing".The first recorded use of the term was in 1499 by Johannes Trithemius in his Steganographia, a treatise on cryptography and steganography, disguised as a book on magic. Generally, the hidden messages appear to be (or to be part of) something else: images, articles, shopping lists, or some other cover text. For example, the hidden message may be in invisible ink between the visible lines of a private letter. Some implementations of steganography that lack a shared secret are forms of security through obscurity, and key-dependent steganographic schemes adhere to Kerckhoffss principle.The advantage of steganography over cryptography alone is that the intended secret message does not attract attention to itself as an object of scrutiny. Plainly visible encrypted messages, no matter how unbreakable they are, arouse interest and may in themselves be incriminating in countries in which encryption'

**Results**

The figure can be generated from visualization/MainFile

Results when the bitPlanes in the benchMark is varies between 1 and 15:



Chromosomes examples:

* proposed:

|  |
| --- |
| [block row,block col, bp\_dir, sb\_dir, sb\_pole, bp, pixel\_dir,row\_offset, col\_offset, number of pixels] |

{[4,2,1,1,1,1,12,29,67,614],

[6,2,1,1,0,1,9,3,83,604],

[6,6,1,0,1,1,3,3,4,144],

[5,6,1,1,0,1,2,65,4,658],

[6,1,0,1,0,1,10,5,89,634],

[1,5,1,1,0,1,8,70,53,639],

[5,1,0,0,0,1,8,40,32,651],

[1,3,0,1,0,1,3,16,12,632],

[3,5,1,0,0,1,9,21,73,628],

[2,3,1,0,1,1,5,26,57,298],

[6,4,1,1,0,1,5,5,12,638],

[3,6,1,0,0,1,4,96,0,658],

[3,1,1,0,0,1,5,76,52,627],

[2,1,1,1,0,1,1,21,51,394],

[5,3,0,0,0,1,5,13,54,632],

[1,6,1,1,0,1,14,61,4,659],

[1,1,0,1,0,1,7,92,1,622],

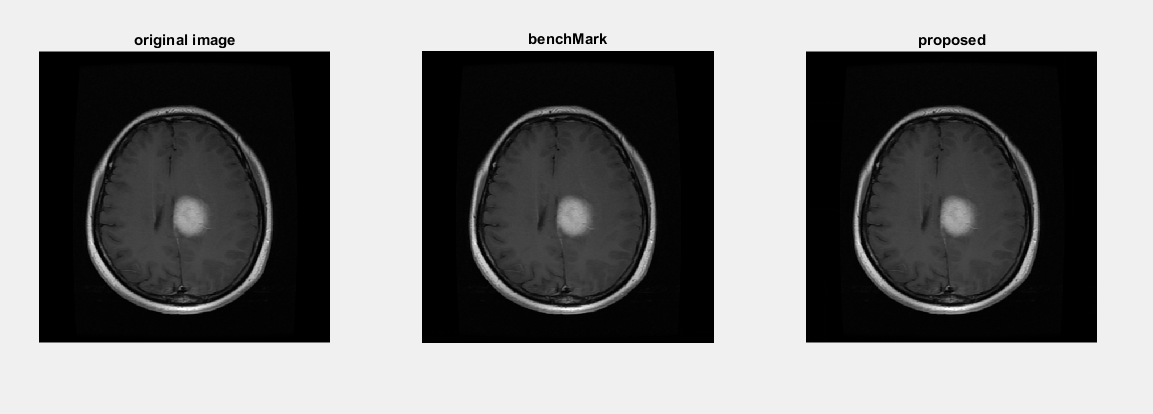
[3,4,1,0,1,1,12,60,65,260],

[5,4,1,0,0,1,5,69,0,488]}

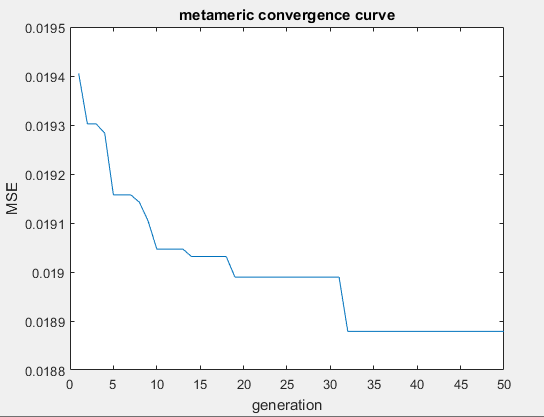
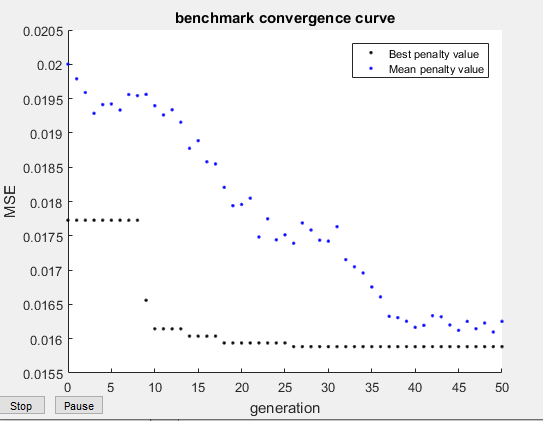
|  |
| --- |
| [ pixel\_dir , col\_offset , row\_offset , bp, sb\_pole, sb\_dir,bp\_dir ] |

* benchmark:

[0,208,488,1,0,1,0]

* Image example:

Comments:

* we notice that the metameric algorithm is better only in terms of SSIM, but generally, the results values are too similar, and the most possible cause is the bitPlane code that equals ‘1’, and this bitplanes has not significant effect on the pixel value.
* Now, let’s see the convergence curves:
* We notice that the proposed algorithm is not converged, and it needs much more iterations to converge, and the possible reason is the large number of metamerics in the chromosome, this means more decision variables need to be optimized versus too smaller number of decision variables to be optimized in the benchmark algorithm. This allows the benchmark algorithm to converge rapidely.
* The second possible reason is the bitplanes that equals one and this make the pixels values not affected significantly by changing their bits.

Recommendations:

* Change the bitplanes to make more significant difference in the pixels values, and allow to more effective comparison between the two algorithms.
* Executing the proposed algorithm with higher number of iterations.
* Note: I think that even if we make the bitplanes varies between 1 and 15 the proposed algorithm may result better performance if we allow it to converge