NatAlgReport

Name: Aleksandar Yordanov User-name: jvgw34

Two-letter code for your chosen NatAlgReal algorithm: FF

Definitions: ρ – Scaling constant for λ, Δn – Dimension n range, kN – Comparison neighbourhood size.

Research, insights, and experimentation: To reduce time complexity, I have implemented the comparison neighbourhood strategy detailed in [Wang, Wang, Zhou. Et. Al.]. This allowed me to achieve 350 cycles in 3.3 seconds vs 86.7 seconds with a naïve algorithm with minima of -62.14 and -62.05 respectively. I have decided to not use the memetic α strategy proposed and have opted for scaling on Δn ($\alpha \cdot \Delta n$) as α itself converges to zero far before any useful exploration can be done. Furthermore, the paper they cite for this strategy explicitly mentions that attempted α adaptation did not lead to improvements. Another improvement I have made is scaling λ on $\lambda/\Delta n^{\rho}$ as I determined that the relationship was not linear. For ranges [0,10] and [-500,500]. The λ values I found where fireflies converge appropriately were 0.5 and $5x10^{-5}$, solving for λ and ρ in the equation gives values of $\rho = 2$ and $\lambda = 50$ to get 0.5 and $5x10^{-5}$. To extrapolate, I tested these values at deltas [0.1, 100000], dim [2,4], and the formula provides good scaling to avoid early convergence within those ranges.

For the following functions: Given function, Schwefel (2D, 3D, 4D) [-500, 500]. I have found minima of -62.14, 0.01, 0.09, 0.41 respectively. These were all obtained in a 10-second run, with: N = 200, $\alpha = 0.04$, $\lambda = 50$, $\rho = 2$, kN = 3. I found that an α of 0.04 was the minimum value for 10 runs where global minima were found each run; any lower and local minima started to appear (Schwefel, given function). I chose N = 200 as that provided enough fireflies and enough time to explore and exploit minima, Values higher (e.g., 1000) do not allow enough time to exploit minima found, Values lower (e.g., 50) often do not find global minima. For kN. I used the value recommended in the paper. Testing on extrema, a value of 1 lead to little convergence. A value of 11 lead to early convergence.

Two-letter code for your chosen NatAlgDiscrete algorithm: FF

Foreword: For every test mentioned below, the minimum number of conflicts were taken from a 5 run trial.

Discretisation Methodology and experimentation:
I implemented and tested two strategies for discretisation. The first being: Encoding each firefly as an improper colouring, and moving a firefly on the notion of distance being the number of vertices with conflicting colours. The second being: Encoding each firefly as a real valued vector where each colour represents a range of [c, c+1] and imposing an artificial distance metric with Euclidean distance. I decided to go with the second strategy as the first strategy provided results that were not much better than random (best of 50 on A) while the second strategy with an adjusted move rule provided far better results (best of 3 on A)

Border Strategy:
To improve colour representation, I clipped my vectors using the modulus of the number of colours, so vertices with the maximum colour "wrap around" to the minimum colour. Compared to the original algorithm where I clipped each vector to min_range and max_range, this alternative strategy provides improvements on large graphs with fewer colours. On graph B, a clipping strategy got a best of 26, while the alternative strategy got a best of 16. On graph C, a clipping strategy got a best of 304, while the alternative strategy got a best of 259. On graph A, the alternative strategy showed minimal variation, both strategies got results within 1 conflict of each other. Tested on $\alpha = 0.5$, $\lambda = 0.005$, $\rho = 1.5$, N = 40, time = 60s.

(continued over)

Movement rule:
In the original firefly algorithm, the move rule is applied to every element in a vector. Moving a point I towards a point J in every dimension. Applying this to an improper colouring results in vertices that are not conflicting being potentially recoloured. To solve this, I only applied movement on vertices that share the same colour in adjacent edges. This way we focus on only resolving conflicts rather than reconfiguring the entire colouring. This modification reduced the conflicts from 30 to 3 on graph A, 243 conflicts to 16 conflicts on B, and 771 conflicts to 259 conflicts on C.
Tested on $\alpha = 0.5$, $\lambda = 0.005$, $\rho = 1.5$, $N = 40$, time = 60s.

Firefly re-randomisation:
To promote exploration, I re-randomise the bottom 80% of my fireflies every 100 iterations. Without this, my fireflies converge, getting stuck exploring a similar colouring. Re-randomisation solves this by forcing diversity and exploration of other, more optimal solutions. For example, on graph A, without re-randomisation, my fireflies converge in around 100 iterations with no further improvements, leading to a best result of 11 conflicts. With re-randomisation, the best result I achieved was 3 conflicts.

I found that the absolute number of fireflies is less critical as long as at least one firefly remains ($N = 5$), though too few fireflies limit comprehensive searching. The more crucial factor is the cycle count for re-randomisation. After analysing the minimum number of conflicts every 5 cycles for graphs A, B, and C, I determined 100 iterations as the optimal reset point for a λ of 0.005 and ρ of 1.5. 100 is the upper limit for where the algorithm converges at these values. Testing different reset intervals revealed that re-randomising every 25 iterations resulted in 6 conflicts, while re-randomising every 100 iterations achieved the best result of 3 conflicts.

This is better shown empirically with the real version of firefly however as minima are known and easily defined. When applying this technique to the Schwefel function [-500,500], dim = 3, for an α of 0.04, in 20 runs, a minima of < 1 was found 20 times, whilst without re-randomisation, the algorithm hit a minima of 118 once in 20 runs.

λ and ρ experimentation:
In NatAlgReal, I defined λ based on Δn to scale movement with the range of the search space, that way movement is consistent between ranges. A major issue with this approach is that I could not find a good way to test the formula at higher dimensions, as the algorithm ran too slowly to get results that I could use to find any corelation between dimensionality and convergence. Therefore, I cannot verify if this is a good approach for kGCP as I use the number of vertices as the “dimension” of the algorithm which is much higher than the number of dimensions tested in NatAlgReal. Considering that, I still managed to find good values for α and λ through trial and error. Though I cannot guarantee good performance for other graphs.

α experimentation:
For α, I attempted using a static value for α and a scaled value based on k. When scaling with k, the chance of an unrepresented colour being used is much more likely, however the results I got were far worse compared to $\alpha = 0.5$. $\alpha = 0.5$ led to 3 conflicts for graph A, while α * colours led to 42 conflicts for graph A. I reached a value of 0.5 through trial and error. Tests on graph A for $\alpha = [0.1, 0.5, 1]$ led to conflicts of 6, 3 and 4 respectively. Tests on B led to 29, 16 and 18, while tests on C led to 271, 259 and 332. The notably high number of conflicts for $\alpha = 1$ on graph C warrants further investigation. Tested on $\lambda = 0.005$, $\rho = 1.5$, $N = 40$, time = 60s.

Summary and other insights:
To summarise, with parameters: $\alpha = 0.5$, $\lambda = 0.005$, $\rho = 1.5$, $N = 40$, time = 60s. I got, for graphs A, B and C, results of 3, 16 and 262 respectively. It is also important to note that I am running this on an Apple M1 chip. When downclocked to low power mode (approx. 40% reduced clock speed), my results were 7, 23 and 324, with the number of cycles being cut in half (e.g. on graph A, 2406 -> 1305 cycles), so results will vary on CPU speed. If needed, adjust max time for verification.