RHCR2 Optimization Report

Agah Duzenli

February 27, 2026

1 Experimental Results: Full 32 Run Data

1.1 Table 1: Parameters $p = 120, z = 9$

Start Point	**Run 1 (Seed 42)**		**Run 2 (Seed 123)**	
	Calls (1,2,3,Sum)	**Results f(sol1, 2, 3)**	**Calls (1,2,3,Sum)**	**Results f(sol1, 2, 3)**
(-300, -400)	121, 121, 121, **363**	210.85, 198.58, 197.99	121, 121, 121, **363**	179.40, 165.12, 164.46
(0, 0)	121, 121, 121, **363**	-7.76, -8.39, -8.41	121, 121, 121, **363**	-7.78, -7.91, -7.91
(-222, -222)	121, 121, 121, **363**	-168.46, -203.98, -204.93	121, 121, 121, **363**	-161.92, -169.97, -184.15
(-510, 400)	121, 121, 121, **363**	-493.36, -493.58, -493.58	121, 121, 121, **363**	-492.70, -493.48, -493.50

1.2 Table 2: Parameters $p = 120, z = 50$

Start Point	**Run 1 (Seed 42)**		**Run 2 (Seed 123)**	
	Calls (1,2,3,Sum)	**Results f(sol1, 2, 3)**	**Calls (1,2,3,Sum)**	**Results f(sol1, 2, 3)**
(-300, -400)	121, 121, 121, **363**	-432.52, -435.03, -435.12	121, 121, 121, **363**	-434.48, -436.13, -436.16
(0, 0)	121, 121, 121, **363**	-45.82, -47.99, -48.11	121, 121, 121, **363**	-47.44, -50.74, -50.87
(-222, -222)	121, 121, 121, **363**	-250.36, -261.38, -261.45	121, 121, 121, **363**	-244.31, -257.85, -257.89
(-510, 400)	121, 121, 121, **363**	-493.91, -494.14, -494.16	121, 121, 121, **363**	-493.08, -494.12, -494.13

1.3 Table 3: Parameters $p = 400, z = 9$

Start Point	**Run 1 (Seed 42)**		**Run 2 (Seed 123)**	
	Calls (1,2,3,Sum)	**Results f(sol1, 2, 3)**	**Calls (1,2,3,Sum)**	**Results f(sol1, 2, 3)**
(-300, -400)	401, 401, 401, **1203**	188.99, 174.72, 174.05	401, 401, 401, **1203**	161.19, 147.16, 146.40
(0, 0)	401, 401, 401, **1203**	-7.97, -8.02, -8.02	401, 401, 401, **1203**	-8.15, -8.56, -8.58
(-222, -222)	401, 401, 401, **1203**	-202.52, -204.11, -206.66	401, 401, 401, **1203**	-186.81, -192.37, -196.04
(-510, 400)	401, 401, 401, **1203**	-493.65, -493.79, -493.80	401, 401, 401, **1203**	-493.68, -493.88, -493.89

1.4 Table 4: Parameters $p = 400, z = 50$

Start Point	**Run 1 (Seed 42)**		**Run 2 (Seed 123)**	
	Calls (1,2,3,Sum)	**Results f(sol1, 2, 3)**	**Calls (1,2,3,Sum)**	**Results f(sol1, 2, 3)**
(-300, -400)	401, 401, 401, **1203**	-432.52, -435.09, -435.18	401, 401, 401, **1203**	-434.48, -436.17, -436.22
(0, 0)	401, 401, 401, **1203**	-45.82, -48.03, -48.17	401, 401, 401, **1203**	-47.44, -50.69, -50.81
(-222, -222)	401, 401, 401, **1203**	-258.37, -264.01, -264.21	401, 401, 401, **1203**	-258.93, -263.83, -264.00
(-510, 400)	401, 401, 401, **1203**	-493.91, -494.15, -494.16	401, 401, 401, **1203**	-493.08, -494.15, -494.16

2 The 33rd Run

For the optimal search, parameters were set to $sp = (-510, 400), p = 200, z = 125, Seed = 123$.

- **Solutions Searched (1,2,3,Sum):** 201, 201, 201, **603**
- **Results f(sol1, sol2, sol3):** -510.84, -511.73, **-511.73**

3 Interpretation and Analysis

3.1 Solution Quality and Algorithm Speed

- **Solution Quality:** The near-global minimum of -511.73287 was achieved in the 33rd run by addressing the limitations of previous trials. Earlier runs with a small range ($z = 9$) lacked the step size to escape shallow local basins (values > 100), resulting in precision without global accuracy. By increasing the range to $z = 125$ and optimizing the starting point, the algorithm successfully bypassed these local traps. This success demonstrates the effectiveness of the RHCR2 multi-stage approach: Stage 1 scouts for a deep basin through broad exploration, while Stages 2 and 3 perform local exploitation to refine the solution.

- **Algorithm Speed and Efficiency:** The algorithm runs very quickly because it explores the search space using a fixed number of neighbors (p). Since p remains constant in each run, the execution time is consistent and almost instantaneous on modern computers. In every stage, the algorithm evaluates p neighbors plus the current solution, which results in a total of $3 \times (p + 1)$ evaluated points across the three stages. Therefore, the computational complexity is linear, $O(p)$, which makes the method efficient even when the search space is large.

- **Performance Summary:** The algorithm demonstrated high solution quality when sufficient exploration parameters were used, such as $z = 50$ or $z = 125$. In particular, the 33rd run successfully found a near-global minimum of -511.73287 by combining a wider search range with a strong starting point. When the neighborhood range was small ($z = 9$), the search often became trapped in poor local minima with values greater than 100. However, the multi-stage refinement process allowed the algorithm to correct these issues when z was large enough, balancing exploration and exploitation effectively.

3.2 Complexity of RHCR2 Runs

The computational complexity of the RHCR2 algorithm is directly proportional to the total number of objective function evaluations performed during a single execution.

- **Evaluation Breakdown:** Each of the three distinct RHC stages performs p evaluations for the randomly generated neighbors and 1 evaluation for the starting point of that specific stage.

- **Total Call Count:** This results in $(p + 1)$ calls per stage. For a full three-stage RHCR2 run, the total number of evaluations is:

$$\text{Total Evaluations} = 3 \times (p + 1)$$

- **Asymptotic Complexity:** Since the number of stages is constant (3), the growth of the algorithm's runtime is determined solely by the population size. This simplifies to a linear complexity of $O(p)$.

Note: Because p is a fixed parameter chosen by the user, the execution time remains consistent across different starting points and neighborhood ranges, allowing for highly predictable performance on modern hardware.

3.3 Impact of Parameters

- **Starting Point (sp):** The point $(-510, 400)$ is closer to the deep basin where the global minimum (approximately -511.73) is located. In optimization, a *basin of attraction* is a region where every downward step leads to the same local minimum. Since $(-510, 400)$ lies inside the correct basin, the algorithm is more likely to reach the global minimum.

  On the other hand, the point $(-300, -400)$ is located in a different region of the search space $[-512, 512]$. From this starting position, even with a large search radius z, the algorithm is more likely to move toward a shallow local minimum (such as the 197.99 result we observed). It may become stuck there because it cannot detect the much deeper valley located in another part of the search space.

- **Population Size (p):** With 400 neighbors, the algorithm samples about 233% more points than with 120 neighbors. This creates a much denser map of the local "frog" function landscape.

  In a randomized algorithm, a small value of p increases the risk of "bad luck," where none of the sampled points move in a downward direction. By increasing p, it becomes more likely that at least one of the 400 sampled points will find a steeper descent path. This makes the results of Run 1 and Run 2 more consistent.

  More samples also reduce the chance that the algorithm will "step over" a narrow but deep valley. Instead of settling for a weaker path, the hill-climbing process is more likely to find a better and more efficient direction downhill.

  However, increasing p also increases the computational cost in a linear way. The time complexity is $O(p)$. This means that while using 400 neighbors gives better search quality than 120 neighbors, it also requires about 3.3 times more function evaluations.

- **Neighborhood Range (z):** In optimization, the parameter z (neighborhood range) defines the maximum distance the algorithm can move from its current position in one step. This creates a balance between *exploration* and *exploitation*.

  Exploitation (Small $z = 9$): When z is small, the algorithm only checks points that are very close to its current location. This is useful for refining or "polishing" a solution and finding the exact bottom of the current valley. However, it also limits the search. The algorithm cannot see better valleys that are slightly farther away. If the starting point is in a bad region, a small z makes it likely that the algorithm will stay there.

  Exploration (Large $z = 50$ or 125): When z is large, the algorithm can sample points across a wider area of the "Frog" function landscape. This allows it to "jump over ridges" and escape shallow local minima. As a result, it has a better chance of finding a deeper basin of attraction.

  The RHCR2 algorithm is effective because it combines both strategies. In Stage 1, it uses a large z to explore the search space and locate a promising region. In Stages 2 and 3, it reduces z to focus on exploitation and precisely locate the minimum.

  In the experiments, runs with $z = 9$ often ended with values such as 197.99 or 164.46 because the step size was too small to escape the initial high region. However, when $z = 125$ was used, the algorithm was able to move across the landscape and reach the global minimum near -511.73.

3.4 Effectiveness of Resampling

- The fact that f decreases in all three stages shows that the algorithm successfully moves from global search to local refinement. In Stage 1, the algorithm performs a coarse search using the initial radius z. It acts like a scout and finds a promising region inside the large search space $[-512, 512]$. In Stages 2 and 3, the radius is reduced to $z/20$ and then $z/400$. This process, called "zooming," allows the algorithm to sample points much closer together. As a result, even if Stage 1 only reaches a steep slope, the next stages guide the solution down to the exact bottom of the basin with high precision.

  This strategy is effective because it avoids the weaknesses of using only one radius. A hill climber with a large z may overshoot a narrow minimum, while a small z may never find the correct valley at all. The strength of RHCR2 comes from combining both approaches: a large z to locate the basin and a small z to precisely reach the minimum.

3.5 Algorithm Assessment

I assess the performance of RHCR2 as **Good**. While it can be sensitive to the initial z value—potentially missing the global minimum if the search radius is too small—its multi stage refinement process makes it a very reliable local optimizer once a promising region is identified. It should be noted that as a heuristic approach, RHCR2 cannot mathematically guarantee the discovery of the global minimum in every scenario; however, it remains highly effective and computationally efficient when paired with adequate exploration parameters.