Name: Hyundoo Jeong
PSID: 2212332

Raunak Sarbajna, Md. Mahin and Christoph F. Eick

COSC 4368: Fundamentals of Artificial Intelligence Spring 2024
Problem Set1 (Individual Tasks[1] Centering on Search)

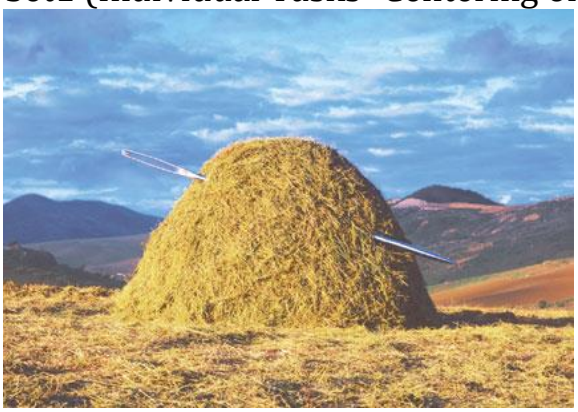

Fig. 1: Finding a Needle in a Large Haystack with Intelligent Search

Submission Deadlines: Task1: Sa., Feb. 10, 11:59p; Task2: Th., Feb. 22, 11:59p
Last Updated: February 1, 10a
Weight: Task weight: T1=35 points, T2=40 points.

1) On Probabilistic Search Algorithms: Implementing and Experimenting with Randomized Hill Climbing *with Resampling* Raunak
Third Draft

Implement a randomized hill climbing algorithm with resampling, called RHCR2 in the following, and conduct a set of experiments, minimizing the following function f:

$$f_{Frog}(x,y) = x*\cos(\sqrt{|x+y+1|})*\sin(\sqrt{|y-x+1|}) + (1+y)*\sin(\sqrt{|x+y+1|})*\cos(\sqrt{|y-x+1|})$$

with $x,y \in [-512,512]$

Your procedure should be called RHCR2 and have the following input parameters:

- *sp*: is the starting point[2] of the Randomized Hill Climbing Resampling2 (RHCR2) run
- *p:* the number of neighbors of the current solution that will be generated
- *z*: neighborhood size: for example, if *z* is set to *z*=10, *p* neighbors for the current solution *s* are generated by adding vectors $v = (z_1, z_2)$ with z_1 and z_2 being random numbers in [-10, +10] uniformly distributed

[1] Collaboration with other students is not allowed!

[2] A vector $(x_1,x_2) \in [-512,+512] \times [-512,+512]$; sample code to visualize f can be found in MS Teams!

- *seed*: which is an integer that will be used as the seed[3] for the random generator you employ in your implementation.

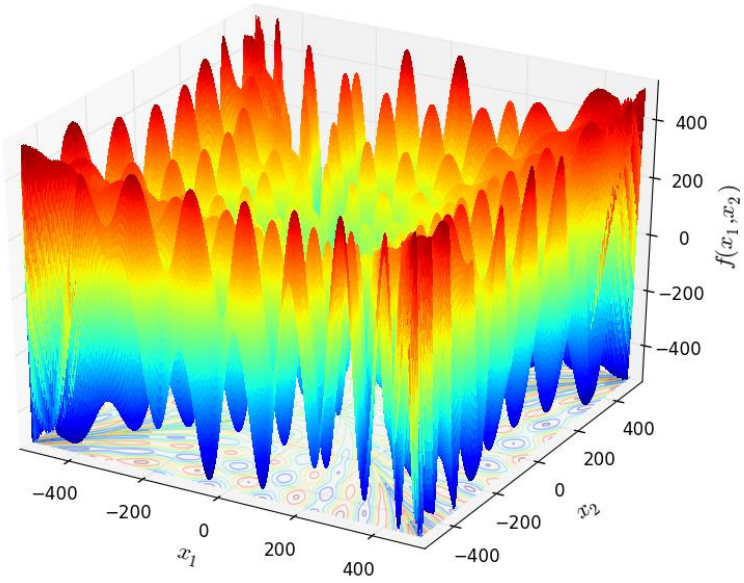

Figure 2: Visualization of f_{Frog}

RHCR2 runs Randomized Hill Climbing 3 times, using sp as the starting position for the first run, the result of the first run sol1 as the start position of the second run which uses a smaller neighborhood size of z/20, and uses the result of the second run sol2 as the starting position of the third run which uses a much smaller neighborhood size of z/400, returning sol3. RHCR2 returns sol1, sol2, and sol3 as well as f(sol1),f(sol2) and f(sol3) and reports how often function f was called in the three runs of RHC and the total number of calls.

RHCR2(sp,z,p,seed);
sol1=RHC(sp,z,p,seed);
sol2=RHC(sp,z/20,p,sol1,seed);
sol3=RHC(sp,z/400,p,sol2,seed);
Return {(sol1,f(sol1)),(sol2,f(sol2)),(sol3,f(sol3))};
Algorithm1: Pseudo code of RHCR@

[3] If you run RHCR2 with the same values for *sp, p, z* and *seed*, it will always return the same solution; if you run if with the same values for *sp, p, z* and a different *seed*, it likely will return a different solution and the number of solutions searched is almost always different.

Run your randomized hill climbing procedure RHCR2 twice[4] for the following parameters:
sp = (-300,-400), (0,0),(-222,-222), (-510,400)
p = 120 and 400; *z* = 9 and 50

REPORT

1. For each of the 32 runs report:
   a. the best solution (x, y) found and its value for *f*
   b. number of solutions generated during the run[5].

Answer: Experiment 1: sp=(-300, -400), z=9, p=120, seed=1
Best Solution: [-279.75098923157833, -438.213748782052], Best Value: -436.3544613457571
Total Solutions Generated: 1800

Experiment 2: sp=(-300, -400), z=9, p=120, seed=2
Best Solution: [-279.751113582985, -438.21543787184385], Best Value: -436.3544612650253
Total Solutions Generated: 1560

Experiment 3: sp=(-300, -400), z=9, p=400, seed=1
Best Solution: [-279.75117721325375, -438.2145424072645], Best Value: -436.3544616985554
Total Solutions Generated: 5200

Experiment 4: sp=(-300, -400), z=9, p=400, seed=2
Best Solution: [-279.75160698538934, -438.2153188020752], Best Value: -436.35446142248765
Total Solutions Generated: 4800

Experiment 5: sp=(-300, -400), z=50, p=120, seed=1
Best Solution: [-279.7419840553276, -438.21364340407655], Best Value: -436.3544501225777
Total Solutions Generated: 1200

Experiment 6: sp=(-300, -400), z=50, p=120, seed=2
Best Solution: [-279.7652310936201, -438.21897476535827], Best Value: -436.35443789851973
Total Solutions Generated: 1440

Experiment 7: sp=(-300, -400), z=50, p=400, seed=1
Best Solution: [-279.75948424904817, -438.2175026025657], Best Value: -436.35445295789157
Total Solutions Generated: 3600

Experiment 8: sp=(-300, -400), z=50, p=400, seed=2

[4] Make sure you use a different seed for your random generator to get a different sequence of random numbers for the 2 runs!
[5] Count the number of times function f is called during the search!

Best Solution: [-279.7499232151775, -438.21400579944975], Best Value: -436.3544618180854
Total Solutions Generated: 4000

Experiment 9: sp=(0, 0), z=9, p=120, seed=3
Best Solution: [-19.013053170958045, 6.127214662204109], Best Value: -17.53293306501865
Total Solutions Generated: 1440

Experiment 10: sp=(0, 0), z=9, p=120, seed=4
Best Solution: [12.174674784960711, 11.174682600292682], Best Value: -11.86804400428539
Total Solutions Generated: 1080

Experiment 11: sp=(0, 0), z=9, p=400, seed=3
Best Solution: [-19.01225865879261, 6.1273525779472955], Best Value: -17.532933237360467
Total Solutions Generated: 4400

Experiment 12: sp=(0, 0), z=9, p=400, seed=4
Best Solution: [12.174268965149006, 11.17423093044902], Best Value: -11.85862115775956
Total Solutions Generated: 4800

Experiment 13: sp=(0, 0), z=50, p=120, seed=3
Best Solution: [-197.75085749448584, 105.3934991048803], Best Value: -195.72877520738183
Total Solutions Generated: 1560

Experiment 14: sp=(0, 0), z=50, p=120, seed=4
Best Solution: [155.85080158421323, 93.44794725689076], Best Value: -155.0253808156428
Total Solutions Generated: 1320

Experiment 15: sp=(0, 0), z=50, p=400, seed=3
Best Solution: [63.701887084310734, -187.79837955609273], Best Value: -185.30947960914708
Total Solutions Generated: 4800

Experiment 16: sp=(0, 0), z=50, p=400, seed=4
Best Solution: [141.16438518752776, 17.764621184048035], Best Value: -140.28809217382138
Total Solutions Generated: 4000

Experiment 17: sp=(-222, -222), z=9, p=120, seed=5
Best Solution: [-219.6511805301565, -220.6512926635696], Best Value: -187.2832773599481
Total Solutions Generated: 960

Experiment 18: sp=(-222, -222), z=9, p=120, seed=6
Best Solution: [-214.9378918165105, -215.93811934838197], Best Value: -203.46675146343745
Total Solutions Generated: 720

Experiment 19: sp=(-222, -222), z=9, p=400, seed=5

Best Solution: [-208.91332608502793, -209.91342152115365], Best Value: -208.8179850842641
Total Solutions Generated: 7200

Experiment 20: sp=(-222, -222), z=9, p=400, seed=6
Best Solution: [-208.43529820383304, -209.43524918728642], Best Value: -208.43353394547216
Total Solutions Generated: 28800

Experiment 21: sp=(-222, -222), z=50, p=120, seed=5
Best Solution: [-350.1639088765779, -291.46107112979826], Best Value: -348.375615915522
Total Solutions Generated: 1440

Experiment 22: sp=(-222, -222), z=50, p=400, seed=5
Best Solution: [-350.16511984433714, -291.4542807599161], Best Value: -348.37564098279444
Total Solutions Generated: 5200

Experiment 23: sp=(-222, -222), z=50, p=120, seed=6
Best Solution: [-350.1687262630579, -291.4608107968532], Best Value: -348.3756404110453
Total Solutions Generated: 2640

Experiment 24: sp=(-222, -222), z=50, p=400, seed=6
Best Solution: [-350.15800872039534, -291.4436053838242], Best Value: -348.37563303063047
Total Solutions Generated: 3200

Experiment 25: sp=(-510, 400), z=9, p=120, seed=7
Best Solution: [-498.54615788805, 404.8300334399432], Best Value: -494.16429770754513
Total Solutions Generated: 1560

Experiment 26: sp=(-510, 400), z=9, p=120, seed=8
Best Solution: [-498.5517026262666, 404.83328426923504], Best Value: -494.1642962207291
Total Solutions Generated: 1200

Experiment 27: sp=(-510, 400), z=9, p=400, seed=7
Best Solution: [-498.5480801399798, 404.83094351340435], Best Value: -494.16429756612536
Total Solutions Generated: 3200

Experiment 28: sp=(-510, 400), z=9, p=400, seed=8
Best Solution: [-498.54762800399385, 404.8307225345975], Best Value: -494.1642976396114
Total Solutions Generated: 5200

Experiment 29: sp=(-510, 400), z=50, p=120, seed=7
Best Solution: [-498.55844913355077, 404.8382549937915], Best Value: -494.16428767730923
Total Solutions Generated: 1080

Experiment 30: sp=(-510, 400), z=50, p=120, seed=8

Best Solution: [-498.5538375012792, 404.8364896320651], Best Value: -494.16428950902764
Total Solutions Generated: 1440

Experiment 31: sp=(-510, 400), z=50, p=400, seed=7
Best Solution: [-498.5500405299811, 404.8341422401554], Best Value: -494.1642928846642
Total Solutions Generated: 3600

Experiment 32: sp=(-510, 400), z=50, p=400, seed=8
Best Solution: [-498.5498531364368, 404.83111039595383], Best Value: -494.1642960755009
Total Solutions Generated: 2800

2. Summarize your results in 4 tables using the format depicted in Fig. 2; one for each *p* and *z* combination (see example below).

Sp=(-300,-400)	Run1(seed = 1)			Run2(seed = 2)		
	Number of solutions sol1,sol2,sol3 3 sum of RHC	<SOLUTION 1, SOLUTION 2, SOLUTION 3>	Result f(sol1),f(sol2),f(sol3)	Number of solutions sol1,sol2,sol3 3 sums of RHC	<SOLUTION 1, SOLUTION 2, SOLUTION 3>	Result f(sol1),f(sol2
P=120&z=9	<1800, 1560, 1800> Total sum:5160	[-279.75098923157833, -438.213748782052] [279.75098923157833, -438.213748782052] [279.75098923157833, -438.213748782052]	[-436.3544613457571] [-436.3544613457571] [-436.3544613457571]	<1560, 1200, 1200> Total sum:3960	[279.751113582985,438.21543787184385] [279.751113582985,438.21543787184385] [279.751113582985,438.21543787184385]	-436.354461265 -436.354461265 -436.354461265
P=120&z=50	<1200, 1440, 1200> Total sum:3840	[-279.7419840553276, -438.2136434040765] [-279.7419840553276, -438.2136434040765] [-279.7419840553276, -438.2136434040765]	[-436.3544501225777] [-436.3544501225777] [-436.3544501225777]	<1440, 1440, 1200> Total sum:4080	[-279.7652310936201, 438.2189747653582] [-279.7652310936201, 438.2189747653587] [-279.7652310936201, 438.2189747653582]	[-436.35443789 [-436.35443789 [-436.35443789
P=400&z=9	<5200, 4800, 5200> Total sum:15200	[279.75117721325375, -438.2145424072645] [279.75117721325375, -438.2145424072645] [279.75117721325375, -438.2145424072645]	[-436.3544616985554] [-436.3544616985554] [-436.3544616985554]	<4800, 3200, 4800> Total sum:12800	[-279.75160698538934,438.2153188020752] [-279.75160698538934,438.2153188020752] [279.7516069853893 4,438.215318802075 2]	-436.354461422 -436.354461422 -436.354461422

P=400&z=50	<3600, 4000, 3600> Total Sum:11200	[-279.75948424904817,438.217502602565] [-279.75948424904817, -438.217502602565] [-279.75948424904817, -438.217502602565]	[-436.35445295789157] [-436.35445295789157] [-436.35445295789157]	<4000, 2800, 4000> Total sum:10800	[-279.74992321517775, -438.21400579944975] [-279.74992321517775, -438.21400579944975] [-279.74992321517775, -438.21400579944975]	[-436.3544618 [-[436.354461 [-[436.354461

Sp=(0,0)	Run1(seed = 3)			Run2(seed = 4)		
	Number of solutions sol1,sol2,sol3 3 sum of RHC	<SOLUTION 1, SOLUTION 2, SOLUTION 3>	Result f(sol1),f(sol2),f(sol3)	Number of solutions sol1,sol2,sol3 3 sum of RHC	<SOLUTION 1, SOLUTION 2, SOLUTION 3>	Result f(sol1),
P=120&z=9	<1440, 1440, 1440> Total sum:4320	[-19.013053170958045, 6.127214662204109] [-19.013053170958045, 6.127214662204109] [-19.013053170958045, 6.127214662204109]	-17.53293306501865, -17.53293306501865, -17.53293306501865	<1080, 1080, 1080> Total sum:3240	[12.174674784960711, 11.174682600292682] [12.174674784960711, 11.174682600292682] [12.174674784960711, 11.174682600292682]	-11.8680 -11.8680 -11.8680
P=120&z=50	<1560, 1560, 1560> Total sum:4680	[-197.75085749448584, 105.3934991048803] [-197.75085749448584, 105.3934991048803] [-197.75085749448584, 105.3934991048803]	-195.72877520738183, -195.72877520738183, -195.72877520738183	<1320, 1320, 1320> Total sum:3960	[155.85080158421323, 93.44794725689076] [155.85080158421323, 93.44794725689076] [155.85080158421323, 93.44794725689076]	-155.025 -155.025 -155.025
P=400&z=9	<4400, 4400, 4400> Total sum:13200	[-19.01225865879261, 6.1273525779472955] [-19.01225865879261, 6.1273525779472955] [-19.01225865879261, 6.1273525779472955]	-17.532933237360467, -17.532933237360467, -17.532933237360467	<4800, 4800, 4800> Total sum:14400	[12.174268965149006, 11.17423093044902] [12.174268965149006, 11.17423093044902] [12.174268965149006, 11.17423093044902]	-11.8586 -11.8586 -11.8586

P=400&z=50	<4800, 4800, 4800> Total Sum:14400	[63.701887084310734, -187.7983795560927] [63.701887084310734, -187.7983795560927] [63.701887084310734, -187.7983795560927]	-185.30947960914708, -185.30947960914708, -185.30947960914708	<4000, 4000, 4000> Total sum:12000	[141.16438518752776, 17.764621184048035] [141.16438518752776, 17.764621184048035] [141.16438518752776, 17.764621184048035]	-140.288092173 -140.288092173 -140.288092173

Sp=(-222,-222)	Run1(seed = 5)			Run2(seed = 6)		
	Number of solutions sol1,sol2,sol3 3 sum of RHC	<SOLUTION 1, SOLUTION 2, SOLUTION 3>	Result f(sol1),f(sol2),f(sol3)	Number of solutions sol1,sol2,sol3 3 sum of RHC	<SOLUTION 1, SOLUTION 2, SOLUTION 3>	Result f(sol1),f(
P=120&z=9	<960 960, 960> Total sum:2880	[219.6511805301565, 220.6512926635696]	-187.2832773599481	<720, 720, 720> Total sum:2160	[214.9378918165105, 215.9381193483819]	-203.4667
P=120&z=50	<1440 1440, 1440> Total sum:4320	[-350.1639088765779, 291.46107112979826]	-348.375615915522	<2640 2640, 2640> Total sum:7920	[350.1687262630579, 291.4608107968532]	-348.3756
P=400&z=9	<7200 7200, 7200> Total sum:21600	[208.91332608502793, 209.91342152115365]	-208.8179850842641	<28800 28800, 28800> Total sum: 86400	[208.43529820383304, 209.43524918728642]	-208.4335

P=400&z=50	<5200 5200, 5200> Total Sum:15600	[-350.16511984433714, -291.4542807599161]	-348.37564098279444	<3200 3200, 3200> Total sum:9600	[350.15800872039534, -291.4436053838242]	-348.37563

Sp=(-510,400)	Run1(seed = 7)			Run2(seed = 8)		
	Number of solutions sol1,sol2,sol3 3 sum of RHC	<SOLUTION 1, SOLUTION 2, SOLUTION 3>	Result f(sol1),f(sol2),f(sol3)	Number of solutions sol1,sol2,sol3 3 sum of RHC	<SOLUTION 1, SOLUTION 2, SOLUTION 3>	Result f(sol1),f(
P=120&z=9	<1560, 1200, 1440> Total sum:4200	[-498.54615788805, 404.8300334399432] [498.55298760552386, 404.8305108769123] [498.55132208441674, 404.8312766532516]	-494.16429770754513 -494.16429752282465 -494.1642974863766	<1200, 1080, 1800> Total sum:4080	[-498.5517026262666, 404.83328426923504] [498.55298760552386, 404.8332221972556] [-498.5542732039563, 404.833219530332]	-494.16429 -494.16429 -494.16429
P=120&z=50	<1080, 960, 960> Total sum:3000	[498.55844913355077, 404.8382549937915] [-498.5571635351184, 404.8382549937915] [-498.5571635351184, 404.8382549937915]	-494.16428767730923 -494.16428767851054 -494.16428767851054	<1440, 1560, 1320> Total sum:4320	[-498.5538375012792, 404.8364896320651] [-498.5538375012792, 404.8364896320651] [-498.5538375012792, 404.8364896320651]	-494.16428 -494.16428 -494.16428

P=400&z=9	<3200, 3600, 3200> Total sum:10000	[-498.5480801399798, 404.83094351340435] [-498.54936511883355, 404.83076506436327] [-498.54762800399385, 404.8307225345975]	-494.16429756612536 -494.1642976396114 -494.1642976396114	<5200, 5200, 4400> Total sum:14800	[498.54762800399385, 404.8307225345975] [498.54891298324765, 404.8305873567484] [498.54762800399385, 404.8307225345975]	-494.1642976 -494.1642976 -494.1642976
P=400&z=50	<3600, 3200, 3600> Total Sum:10400	[-498.55844913355077, 404.8382549937915] [-498.5571635351184, 404.8382549937915] [-498.5571635351184, 404.8382549937915]	-494.16428767730923 -494.16428767851054 -494.16428767851054	<2800 2400, 2000> Total sum:7200	[498.5538375012792,404.8364896320651] [498.5538375012792,404.8364896320651] [498.5538375012792,404.8364896320651]	-494.1642895090 -494.1642895090 -494.1642895090

3. Finally, run RHCR2 one more time with "your preferred choice" of values for *sp, p, z*, and report the result; students who find better solutions in this 33rd run will get more points for the 33rd run subtask.

   Answer:
   Experiment 33: sp=(-300, -400), z=9, p=120, seed=15439
   Best Solution: [-279.7493350141575, -438.21349666280014], Best Value: -436.3544617004057
   Total Solutions Generated: 1440

   I preferred the initial one when sp(-300,-400),z, = 9, p = 120 but I changed seed value is random(In above). The result is pretty similar value compared to my Experiment1~8 but It is slightly different, compared to speed it is better than initial Experiment because this result's Total Solutions is lower than Experiment 1~8 so I can assume Experiment 33 is better RHCR2 result.

4. Interpret[6] the obtained results evaluating solution quality, algorithm speed, impact of *sp, p, and z* on solution quality and algorithm speed. Summarize and interpret the complexity of the RHCR2 runs!

[6] At least 25% of the available points will be allocated to interpreting the results.

Answer: The solution quality is evaluated based on the best values found for each experiment. Lower function values indicate better solutions in the context of the optimization problem. Algorithm speed is related to the "Total Generated Number" if generated total number is low, which means it is faster convergence. Sp(starting point) impact on solution quality and algorithm speed can be analyzed. p(Number of Neighbor)affects the algorithm's exploration capability. Higher p values may lead to more extensive exploration but can increase computational cost. Z(size) determines the magnitude of changes in the solution space. Smaller z values may lead to more precise search but might increase the risk of getting stuck in local minima.

Time complexity of RHCR2 function: Time complexity is O(n * p). My generated function time complexity's O(n), and RHC's time complexity is iterating p times, and called generated function so O(p * N). Also, RHCR2 function 3 calls RHC function in terms of decreasing z value z, z/20, z/400 so O(p * N + p *N/20 + p * N/400) which is O(n * p).

5. Did the resampling option lead to better solutions? Do you believe with other values for *p* and *z* better results could be accomplished? Finally, assess if RHCR2 did a good, medium, or bad job in computing a (local) minimum for *f*.

Answer: Yes, I believe the resampling option contributes to achieving better solutions. For instance, in Experiment 33, by selecting a random seed, the algorithm generated solutions faster than in Experiments 1-8. Moreover, depending on the values of p and z, it influenced the total number of runs and the best solution obtained. I also believe that adjusting p and z can lead to improved solutions.

In evaluating my RHCR2 function, I would consider it to be of medium effectiveness. For instances such as sp(-300, -400) and (-510, 400), the best values remained relatively constant. However, when considering sp(0, 0), the majority of the best values showed irregular patterns, indicating a more dynamic behavior.

Pseudo code:

```
function frog(x):
  x1, x2 = x
  return (x1 * sin(sqrt(abs(x2 + 1 - x1))) * cos(sqrt(abs(x1 + x2 + 1))) +
          (x2 + 1) * cos(sqrt(abs(x2 + 1 - x1))) * sin(sqrt(abs(x1 + x2 + 1))))

function generate_neighbor(current_solution, z):
  neighbor = [
     current_solution[i] + random.uniform(-z, z)
     for i in range(length(current_solution))
  ]
  return neighbor

function RHC(sp, z, p, seed):
  seed_random_generator(seed)
  current_solution = array(sp)
  function_calls = 0

  while True:
    neighbors = [generate_neighbor(current_solution, z) for _ in range(p)]
    neighbor_values = [frog(neighbor) for neighbor in neighbors]
    best_neighbor_index = index_of(min(neighbor_values), neighbor_values)
    best_neighbor = neighbors[best_neighbor_index]
    function_calls += p

    if frog(best_neighbor) >= frog(current_solution):
      return current_solution, frog(current_solution), function_calls
    else:
      current_solution = best_neighbor

function RHCR2(sp, z, p, seed):
  sol1, val1, calls1 = RHC(sp, z, p, seed)
  sol2, val2, calls2 = RHC(sol1, z / 20, p, seed)
  sol3, val3, calls3 = RHC(sol2, z / 400, p, seed)

  solutions = [(sol1, val1), (sol2, val2), (sol3, val3)]
  total_calls = calls1 + calls2 + calls3

  return solutions, total_calls
```

p/z for sp = (-300,-400), p=120 & z=9	Run1			Run2		
	Number of solutions searched in the first, second and third run of RHC, and the sum of the 3 numbers	<SOLUTION 1, SOLUTION 2, SOLUTION 3>	Result f(sol1),f(sol2),f(sol3)	…	…	…
p=120 & z=50						
p=400 & z=9						
p=400 & z=50						

Fig. 3: Table Template to report the results of the 33 runs of RHCR2

You should summarize your results in 4/8 tables formatted as the above, for each of the 4 combinations of *p* & *z*. Don't forget to summarize the results of your 33rd run[7] and to provide the other information asked for in the task specification!

```
/* FROG FUNCTION DEFINITION in C++
AUTHOR: Raunak Sarbajna*/

#include <cmath>
#include <iostream>

double frog (const double x[])
{
  double x1 = x[0];
  double x2 = x[1];
```

[7] Also briefly explain why you chose the particular input parameters for *sp, p* and *z* for your 33rd run!

```
  double term1 =  x1 * sin (sqrt (fabs (x2 + 1 - x1))) * cos (sqrt (fabs
(x1 + x2 + 1)));
  double term2 =  (x2 + 1) * cos (sqrt (fabs (x2 + 1 - x1))) * sin (sqrt
(fabs (x1 + x2 + 1)));

  return term1 + term2;
}



int main ()
{
  double x[] = { -100, 100 };
  double result = frog (x);
  std::cout << "f4(x) = " << result << std::endl;
  return 0;
```

PYTHON:

```
import numpy as np

def frog(x):
  x1, x2 = x
  return (x1 * np.sin(np.sqrt(np.abs(x2 + 1 - x1))) *
np.cos(np.sqrt(np.abs(x1 + x2 + 1))) +
          (x2 + 1) * np.cos(np.sqrt(np.abs(x2 + 1 - x1))) *
np.sin(np.sqrt(np.abs(x1 + x2 + 1))))

# Example usage
x = np.array([-100,100])
result = frog(x)
print(result)
```

Fig. 4: C++ and Python Code for f_{frog}

Submission Guidelines:

The followings are expected for submission:

1. A clearly written report. The report should include the followings:
   - All 4 tables of obtained results.
   - Random seed used for your experiments.
   - Expected results interpretation and conclusions as described above.
   - Summary of your 33rd run
2. Source code (Implemented in any language of your choice with a README file of program instructions)
3. Submission will be on MS Teams (Submission link will be available once the assignment is created)

Failure to follow all instructions will lead to point deductions!

Useful Links:

Random Number Generator: How Do Computers Generate Random Numbers? (freecodecamp.org)