

# Parallel Processing for Black Box Optimization in R

Meadow Monticello '26, Mathematics & Data Science  
Faculty Mentor: Dr. Rommel Regis, Mathematics



Name: Meadow Monticello

Year: Junior

Major: Mathematics & Data Science

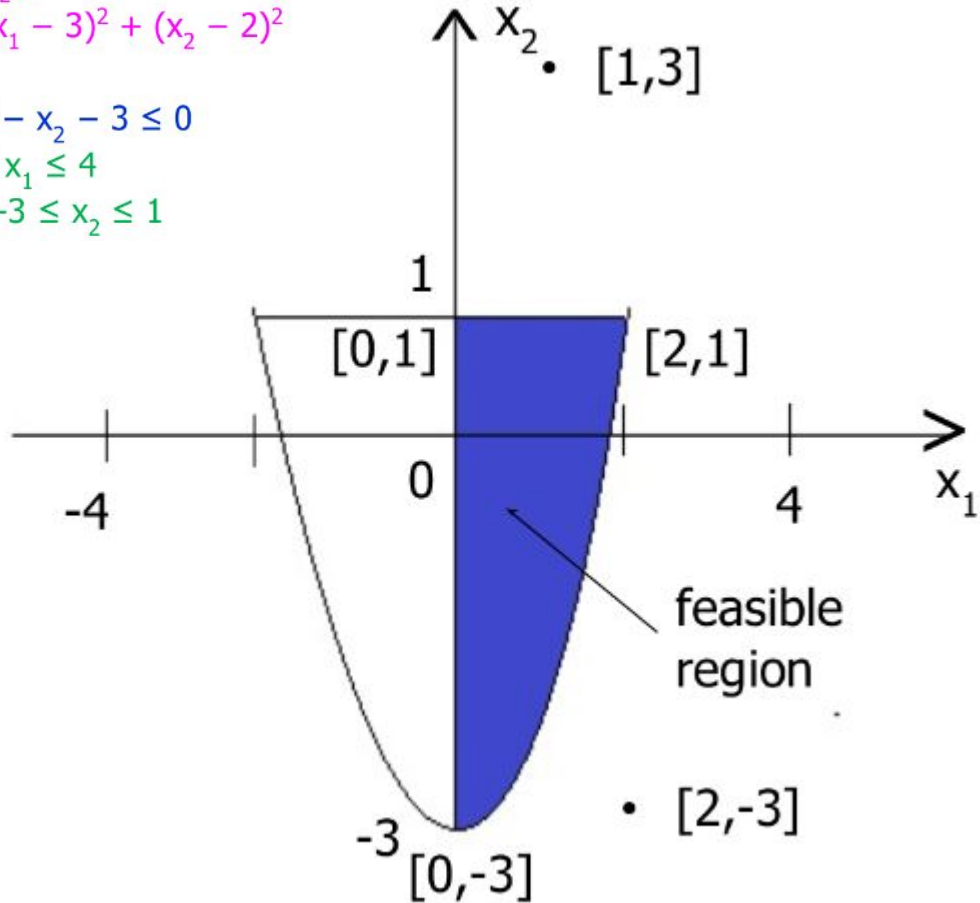
Hometown: Burlington, NJ

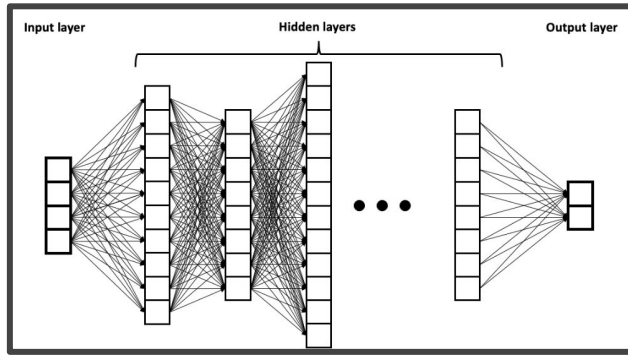
---

# Optimization

- The goal is to find values for the input variables, say  $x_1, x_2, \dots, x_n$ , that maximize or minimize an objective function  $f(x_1, x_2, \dots, x_n)$  possibly subject to constraints that the input variables need to satisfy.

$$\begin{aligned} \min & f([x_1, x_2]) \\ &= (x_1 - 3)^2 + (x_2 - 2)^2 \\ \text{s.t.} & \\ & x_1^2 - x_2 - 3 \leq 0 \\ & 0 \leq x_1 \leq 4 \\ & -3 \leq x_2 \leq 1 \end{aligned}$$





# Motivation



- Optimization is widely used in **engineering** (e.g. mechanical and aerospace)
  - Minimizing fuel consumption
  - Crash safety
- Improving the predictive ability of machine learning models



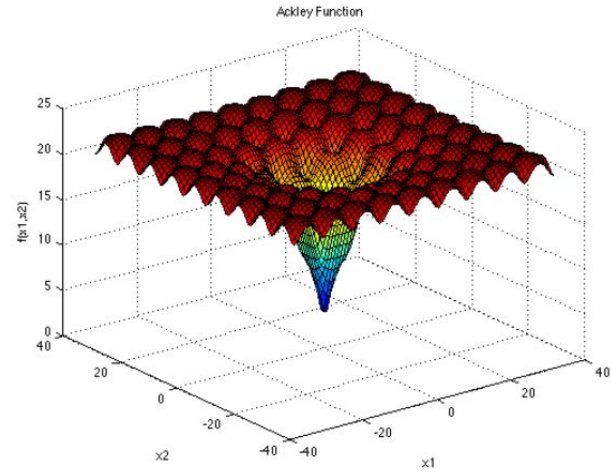
```

1 ackley <- function(xx)
2 {
3   a=20; b=0.2; c=2*pi
4
5   d <- length(xx)
6
7   sum1 <- sum(xx^2)
8   sum2 <- sum(cos(c*xx))
9
10  term1 <- -a * exp(-b*sqrt(sum1/d))
11  term2 <- -exp(sum2/d)
12
13  y <- term1 + term2 + a + exp(1)
14  return(y)
15 }

```

# Challenges

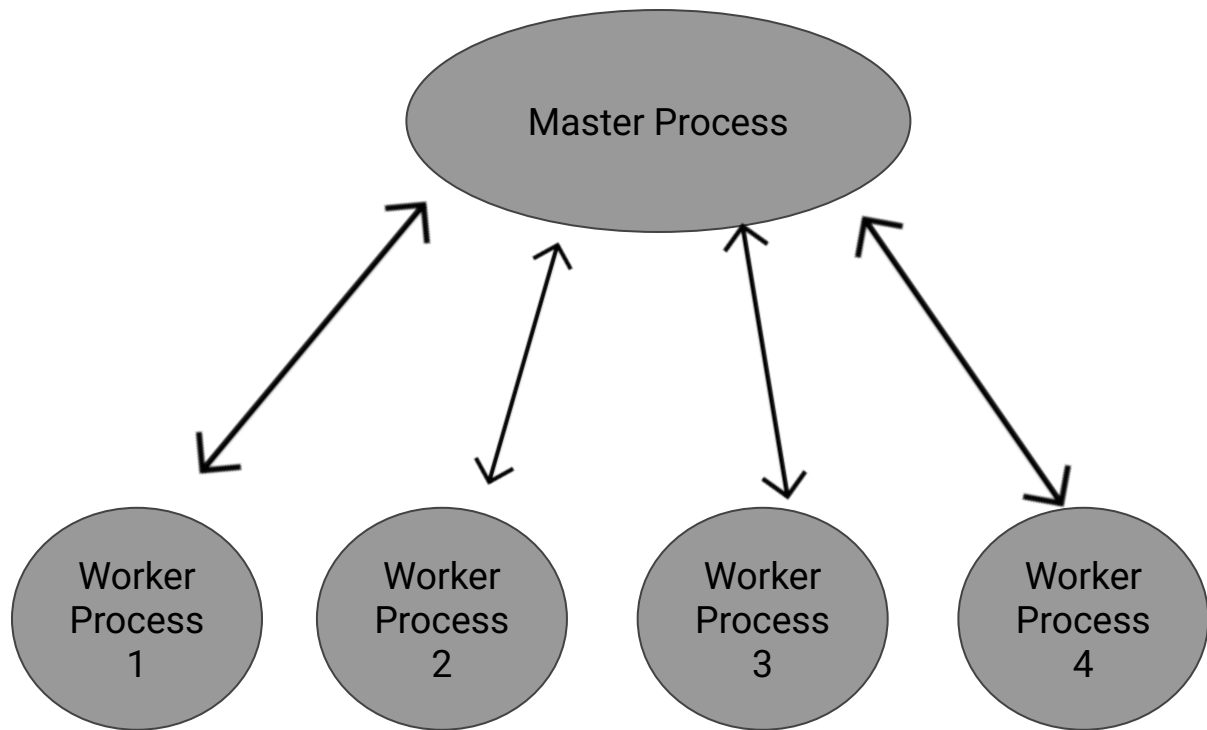
With Optimization



$$f(\mathbf{x}) = -a \exp \left( -b \sqrt{\frac{1}{d} \sum_{i=1}^d x_i^2} \right) - \exp \left( \frac{1}{d} \sum_{i=1}^d \cos(cx_i) \right) + a + \exp(1)$$

## Master-Worker

- Use different processors simultaneously
- Detect **cores**
  - 12
- Use to run more trials efficiently and quickly



```

34 # problem info including bounds on the variables (lb, ub)
35 funcinfo <- ObtainFunctionInfo(funclabel, dim)
36 lb <- funcinfo$lb
37 ub <- funcinfo$ub
38 n <- length(lb)
39
40 # number of local optimization runs for each process for the
41 # parallel version
42 num_runs_per_proc <- floor(num_runs/num_proc)
43
44 # generate initial solutions for all runs
45 set.seed(randseed)
46 init_sols_mat <- t(replicate(num_runs,
47                             lb + runif(n)*(ub - lb)))
48
49
50 # run optim sequentially on a given processor j
51 multistart_optim <- function(j) {
52
53   process_results <- list()
54
55   # loop through the local optimization runs for the given processor
56   for (i in 1:num_runs_per_proc) {
57
58     # obtain initial solution
59     init_sol <- init_sols_mat[(j-1)*num_runs_per_proc+i, ]
60
61     # perform local optimization run
62     process_results[[i]] <-
63       optim(par = init_sol, fn = funcname, method = "L-BFGS-B",
64           lower = lb, upper = ub)
65
66   }
67
68   return(process_results)
69
70 }

```

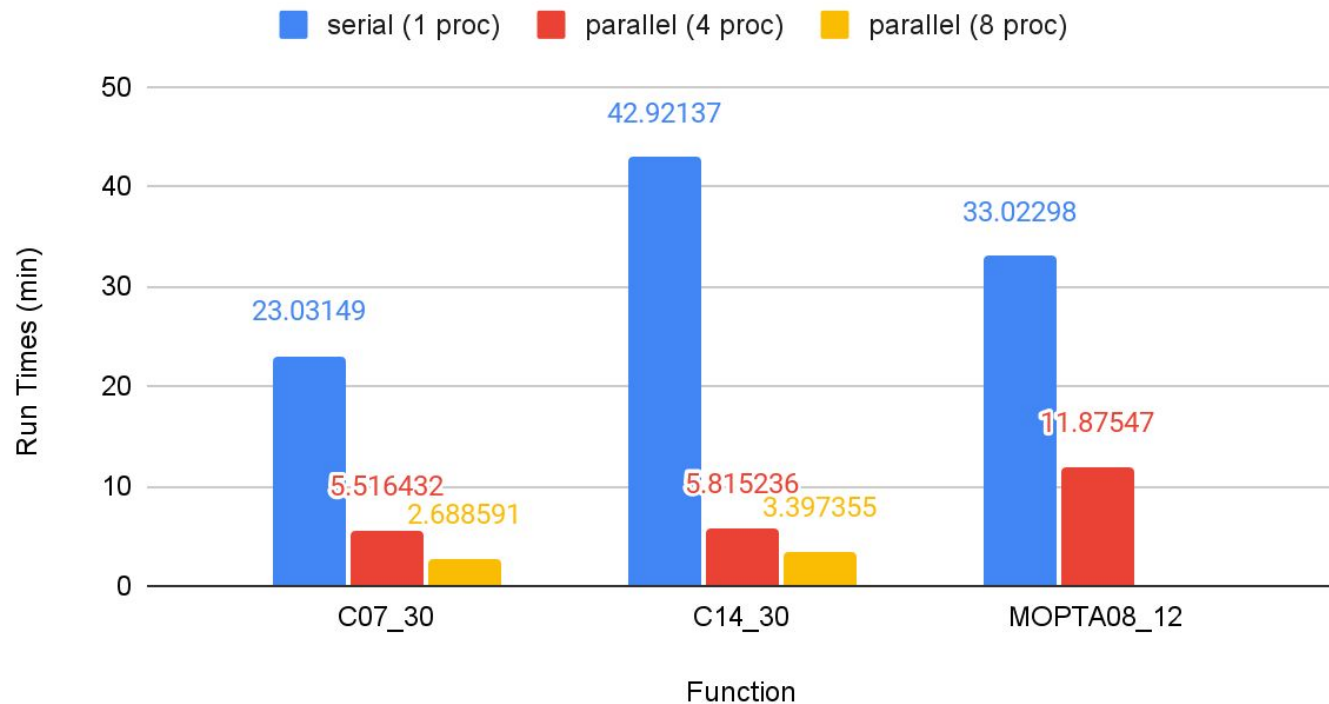
```

71
72 # initialize cluster
73 cl <- makeCluster(num_proc)
74 registerDoParallel(cl)
75 start_time <- Sys.time()
76
77 # run multistart optimization in parallel
78 results <- foreach(j = 1:num_proc) %dopar% {
79   multistart_optim(j)
80 }
81 # results is a list with num_proc elements each of which
82 # is a list with num_runs_per_proc elements
83
84 # stop the parallel backend
85 stopCluster(cl)
86
87 # reorganize results
88 all_results <- list()
89 for (j in 1:num_proc) {
90   all_results <- c(all_results, results[[j]])
91 }
92
93 # gather results from all processes and find the overall
94 # best solution
95 best_result <- all_results[[1]]
96 for (i in 2:num_runs) {
97   if (all_results[[i]]$value < best_result$value) {
98     best_result <- all_results[[i]]
99   }
100 }

```

## Parallel Code

## Run Times of a Multi-Start Constrained Optimization Algorithm



Charted Results

# Potential Future Work

- Work on
  - Other real world optimization on problems
  - Using different multistart optimization algorithm
  - Using other ways to parallelize optimization algorithms
- If access to better technology
  - Run codes with larger dimensions
  - Run more trials
  - Running codes longer

---

# Acknowledgements

- Dr. Rommel Regis
- SJU Summer Scholars Program
- John P. McNulty Program for Leadership in Science and Mathematics
  - Katie Harper
- SJU Mathematics Department

