Optimization of the Rosenbrock Function Using the Jaya Algorithm

Priyanshu Tiwari

March 14, 2025

Introduction

This notebook demonstrates the application of the Jaya algorithm to minimize the Rosenbrock function. The Jaya algorithm, introduced by Rao (2016), is a parameter-free optimization technique known for its simplicity and effectiveness.

Mathematical Background

The Rosenbrock function is a classic non-convex benchmark function in optimization, defined as:

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ represents the decision variables.

This function is characterized by: - A global minimum at $\mathbf{x} = (1, 1, \ldots, 1)$ with $f(\mathbf{x}) = 0$ - A narrow, curved valley where the minimum lies - Gradient-based methods often struggle with this function due to its shape

Implementation

First, let's load the required package:

```r
library(Jaya)
```

```
## Warning: package 'Jaya' was built under R version 4.4.3
```

Now, we define the Rosenbrock function:

```r
rosenbrock <- function(x) {
  sum(100 * (x[-length(x)]^2 - x[-1])^2 + (x[-length(x)] - 1)^2)
}
```

Let's run the Jaya optimization algorithm:

```r
set.seed(123) # For reproducibility
result <- jaya(
  fun = rosenbrock,
  lower = rep(-5, 3),  # Lower bounds for 3 variables
  upper = rep(5, 3),   # Upper bounds for 3 variables
  popSize = 30,        # Population size
  maxiter = 100,       # Maximum iterations
  n_var = 3,           # Number of variables
  opt = "minimize"     # Minimize the function
)
```

Results Analysis

Let's display the summary of our optimization results:

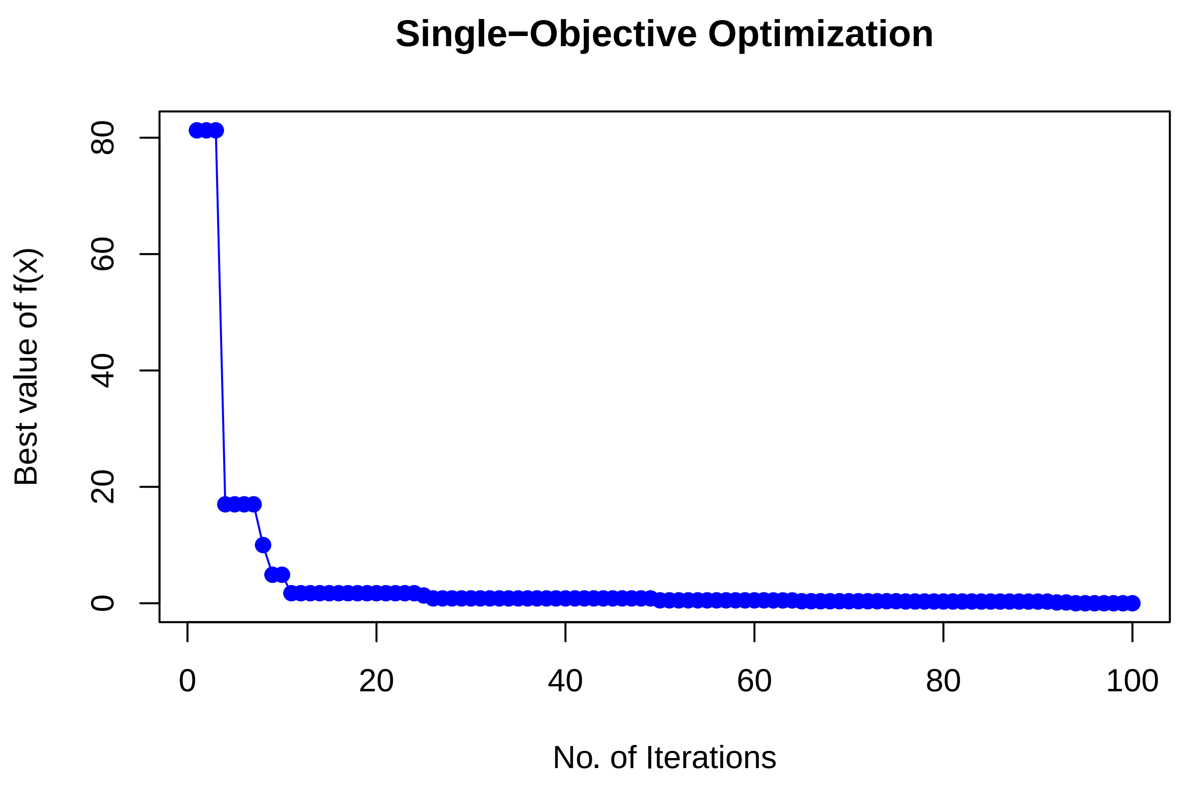
```r
summary(result)
```

```
## Jaya Algorithm
## Population Size      = 30
## Number of iterations = 100
## Number of variables  = 3
##
## Objective: minimize
## Objective Function:
## [[1]]
## function (x)
## {
##     sum(100 * (x[-length(x)]^2 - x[-1])^2 + (x[-length(x)] -
##         1)^2)
## }
## <bytecode: 0x000001b5b0d4cd20>
##
##
## Limits:
## x1 = [-5, 5]
## x2 = [-5, 5]
## x3 = [-5, 5]
##
## Best Result:
##       Best.x1  Best.x2  Best.x3    Best.f.x.
## Best 1.001879 1.002691 1.007286 0.0004854349
```

The summary provides information about: - The optimal solution found (variable values) - The objective function value at this solution - The optimization parameters used

Next, let's visualize the convergence of the algorithm:

```r
plot(result)
```



The plot shows how the best objective function value improved with each iteration.

Let's extract the best solution and verify how close it is to the known global minimum:

```r
# Extract the best solution
# Print the structure to see what we're working with
str(result$Best)
```

```
## 'data.frame':    1 obs. of  4 variables:
##  $ x1  : num 1
##  $ x2  : num 1
##  $ x3  : num 1.01
##  $ f(x): num 0.000485
```

```r
# Access the solution components more safely
best_x1 <- result$Best[1, 1]
best_x2 <- result$Best[1, 2]
best_x3 <- result$Best[1, 3]
best_value <- result$Best[1, 4]

# Print results
cat("Best solution found: x1 =", best_x1, ", x2 =", best_x2, ", x3 =", best_x3, "\n")
```

```
## Best solution found: x1 = 1.001879 , x2 = 1.002691 , x3 = 1.007286
```

```r
cat("Global minimum:      x1 = 1, x2 = 1, x3 = 1\n")
```

```
## Global minimum:      x1 = 1, x2 = 1, x3 = 1
```

```r
cat("Function value at best solution:", best_value, "\n")
```

```
## Function value at best solution: 0.0004854349
```

```r
cat("Function value at global minimum: 0\n")
```

```
## Function value at global minimum: 0
```

Interpretation

The Rosenbrock function presents a challenging optimization landscape due to its narrow curved valley. The Jaya algorithm navigates this landscape by:

1. Maintaining a population of candidate solutions
2. Moving each solution toward the best solution while avoiding the worst solution
3. Accepting improvements without algorithm-specific parameters

This approach is especially valuable for problems where: - The objective function has multiple local minima - Gradient information is unavailable or unreliable - Simplicity and robustness are preferred over specialized tuning

Conclusion

The Jaya algorithm has successfully optimized the Rosenbrock function, demonstrating its capability to handle challenging optimization problems without parameter tuning. This makes it particularly suitable for real-world optimization scenarios where problem characteristics might be unknown or variable.

References

Rao, R. (2016). Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. International Journal of Industrial Engineering Computations, 7(1), 19-34.