P11

edgardjfc

May 2022

1 Introduction

The objective of this assignment is to optimize for two or more different variables. With the specific goal to graph the percentage of Pareto solutions as the function of number of functions $k \in [2, 3, 4, 5]$, with diagrams to verify the statistical variance.

2 Code

The base for this code was written based on the work of our teacher E. Schaeffer [3] and classmate J. "FeroxDeitas" [2]. With the entirety of this code being found in the github repository of E. "edgardjfc" [1].

The code in R starts declaring three different functions. The function "expressions" which is dedicated to produce the different polynomial expressions, the "evaluation" function, dedicated to evualuate certain functions, and the domin.by function, dedicated to check if there is an improvement between the functions generated.

```
expresions <- function(maxdeg, varcount, termcount) {
  f <- data.frame(variable=integer(), coef=integer(), degree=integer())
  for (t in 1:termcount) {
    var <- sample(1:varcount, 1)
    deg <- sample(0:maxdeg, 1)
    f <-  rbind(f, c(var, runif(1), deg))
  }
  names(f) <- c("variable", "coef", "degree")
  return(f)
}

evaluation <- function(pol, vars) {
  value <- 0.0
  terms = dim(pol)[1]
  for (t in 1:terms) {
```

```
    term <- pol[t,]
    value <-  value + term$coef * vars[term$variable]^term$degree
  }
  return(value)
}

domin.by <- function(target, challenger) {
  if (sum(challenger < target) > 0) {
    return(FALSE)
  }
  return(sum(challenger > target) > 0)
}
```

Next we can see the parameters given for the initial conditions of the experiment. Where vc is the amount of variables, md is the max degree of the polynomial, and tc is the amount of terms in the function.

```
df = data.frame()
vc <- 3
md <- 2
tc <- 4
funciones <- c(2, 3, 4, 5)
obj <- list()
k = 0
```

For the beginning of the loop, we have the following code that is in charge of describing the 20 experiments that will be run. Then this same part of the code evaluates 200 different solutions created randomly, for all objective functions. Afterwards, the function domin.by is used to determine which functions are dominant and which are not, results which are counted and then used to produce the violin graph.

```
for (j in funciones){
  k = j
  for (replica in 1:20){
    for (i in 1:k) {
      obj[[i]] <- expresions(md, vc, tc)
    }
    minim <- (runif(k) > 0.5)
    sign <- (1 + -2 * minim)
    n <- 200
    sol <- matrix(runif(vc * n), nrow=n, ncol=vc)
    val <- matrix(rep(NA, k * n), nrow=n, ncol=k)
    for (i in 1:n) {
      for (j in 1:k) {
        val[i, j] <- evaluation(obj[[j]], sol[i,])
```

```
      }
    }
    mejor1 <- which.max(sign[1] * val[,1])
    mejor2 <- which.max(sign[2] * val[,2])
    cual <- c("max", "min")
    no.dom <- logical()
    dominadores <- integer()
    for (i in 1:n) {
      d <- logical()
      for (j in 1:n) {
        d <- c(d, domin.by(sign * val[i,], sign * val[j,]))
      }
      cuantos <- sum(d)
      dominadores <- c(dominadores, cuantos)
      no.dom <- c(no.dom, sum(d) == 0)
    }
    frente <- subset(val, no.dom)
    porcentaje = (length(frente[,1])/n)*100
    resultado = c(k, replica, porcentaje)
    df = rbind(df, resultado)
    names(df) = c("k", "Replica", "Porcentaje")
  }
}
```

3 Results

In figure 1 we can see a diagram of 2 objective functions, where the green markers are functions that were not dominated by any other.

After producing all the different objective functions for $k \in [2, 3, 4, 5]$ a violin graph was produced with the respective information derived from each, shown in figure 2

4 Analysis

A variance analysis was done on the data obtained in the violin plot chart and after producing a result of p much smaller than 0.05, it was concluded that there was a significant variation given by the number of objective functions.

5 Conclusion

We can see how the number of non dominated functions is very low compared to higher objective functions, which means that it is harder to find optimal solutions for smaller amounts of objective functions. However, as the number

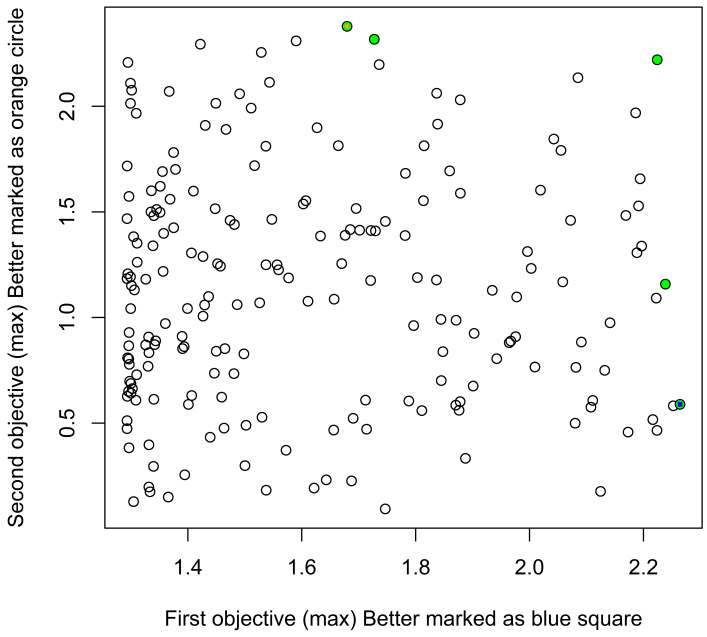

Figure 1: Two objective functions drawn

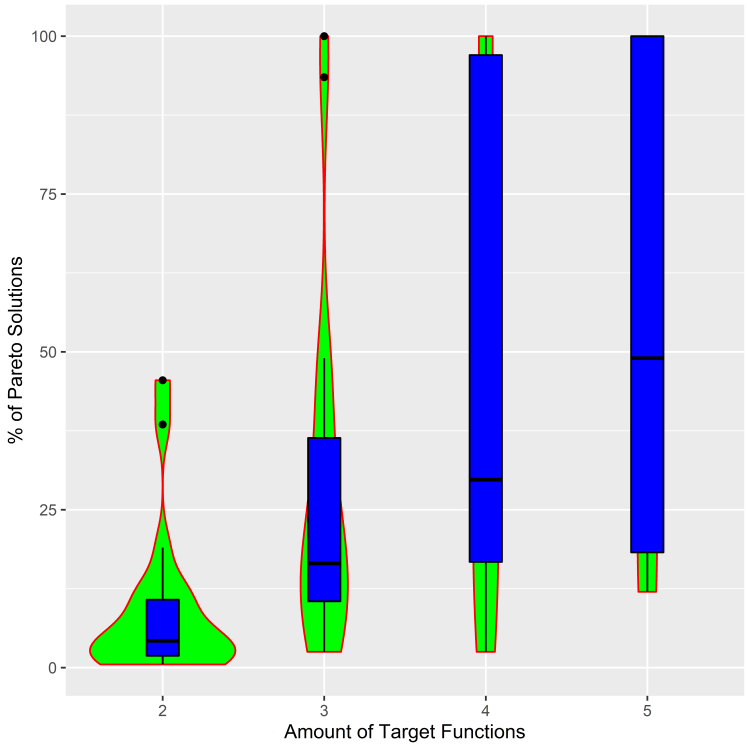

Figure 2: We can observe the changes in the process as the k increases

of objective functions increases we can see an increase on the number of non dominated functions.

6 Challenge

For this challenge the code was developed in a very similar way to the base code for this assignment, however there was a simplification in that the number of objective functions was 2 across all repetitions of the experiment. The difference now is that we now take a group of solutions which size depends on the percentage of the original front, by including the following code:

```
porcentaje=50
dispersos = kmeans(frente, round(dim(frente)[1]*porcentaje/100),
    iter.max = 1000, nstart = 50, algorithm = "Lloyd")
dispersos$cluster
dispersos$centers

mejor1 <- which.max((1 + (-2 * minim[1])) * val[,1])
mejor2 <- which.max((1 + (-2 * minim[2])) * val[,2])
```

This code produced figure 3 and figure 4, which represent the results given by having two objective functions.

Through this method we were able to find new solutions to the problem that were not present on the original one, with a really close approximation to the dominant solutions in the original problem.

References

[1] E. Edgardjfc. *GitHub,P11*. URL: `https://github.com/edgardjfc/Simulacion-Nano-2022/tree/main/P11`.

[2] J. FeroxDeitas. *GitHub,P11*. URL: `https://github.com/FeroxDeitas/Simulacion-Nano/blob/main/Tareas/P_11`.

[3] E. Schaeffer. *GitHub,parentoFronts*. URL: `https://github.com/satuelisa/Simulation/tree/master/ParetoFronts`.

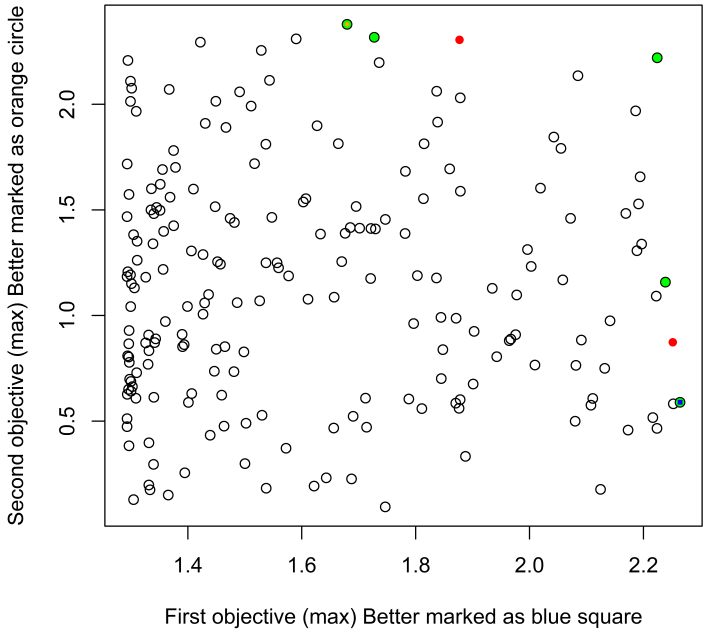

Figure 3: Green circles represent the dominant functions and the red circles represent the new solutions