Práctica 11
Frentes de Pareto

Edson Edgardo Samaniego Pantoja **Materia:** Simulación computacional
Fecha 12 de mayo de 2021

1. Introducción

En la práctica se trabaja con optimización multicriterio, mejor explicado trata de que a un problema con múltiples objetivos, en donde estos objetivos pueden contraponerse, mientras uno mejora los otros empeoran y viceversa.

2. Objetivo

El objetivo es graficar el porcentaje de soluciones de Pareto como función del número de funciones objetivo para `k` $\in [2, 8]$ en pasos de dos y estos datos ser visualizados en diagramas de violín combinado con diagrama caja-bigote.

3. Simulación

El código base es dado en el repositorio de Schaeffer [3] que de igual manera se puede consultar en la pagina web [2] donde se explica cada código paso a paso. Primeramente la modificación realizada en el programa es hacer variante la variable `k` la cual manda la cantidad de número de funciones objetivo.

```
for k in range(2, 9, 2):
    n = 500 # cuantas soluciones aleatorias
    replicas=30
    porcentaje=[]
    for rep in range(0, replicas):
        obj = [poli(md, vc, tc) for i in range(k)]
        minim = np.random.rand(2) > 0.5
        sol = np.random.rand(n, vc)
        val = np.zeros((n, 2))
```

Como se puede observar se utiliza un ciclo `for` que toma valores de dos a nueve en saltos de dos y posteriormente entra a otro ciclo `for` el cual genera réplicas (treinta) del programa que realiza la obtención de los mejores resultados conocidos como frentes de Pareto.

Los mejores resultados de las treinta réplicas son convertidas a un porcentaje respectivo a las soluciones aleatorias dadas en la variable `n` de valor quinientos para poder almacenarlo en una lista llamada `porcentaje`.

```
        for i in range(n):
            for j in range(2):
```

```
            val[i, j] = evaluate(obj[j], sol[i])
        sign = [1 + -2 * m for m in minim]
        mejor1 = np.argmax(sign[0] * val[:, 0])
        mejor2 = np.argmax(sign[1] * val[:, 1])
        cual = {True: 'min', False: 'max'}

        dom = []
        for i in range(n):
            d = [domin_by(sign * val[i], sign * val[j]) for j in range(n)]
            dom.append(sum(d))
        frente = val[[d == 0 for d in dom], :]
        porc=(len(frente)*100)/n
        porcentaje.append(porc)
    pc_violin.append(porcentaje)
```

La lista indexada de treinta porcentajes es acumulada en otra lista `pcviolin` que acumula por cada función objetivo, de esta manera se tiene una lista de listas de porcentajes por función objetivo. El programa puede ser consultado en github [1].

4. Resultados

Los resultados a graficar mostrados en la figura 1 son tomados de la lista de listas donde se acumuló los porcentajes de cada función objetivo para de esa manera poder generar los diagramas de violín que muestran la diferencia entre cada una de las funciones.

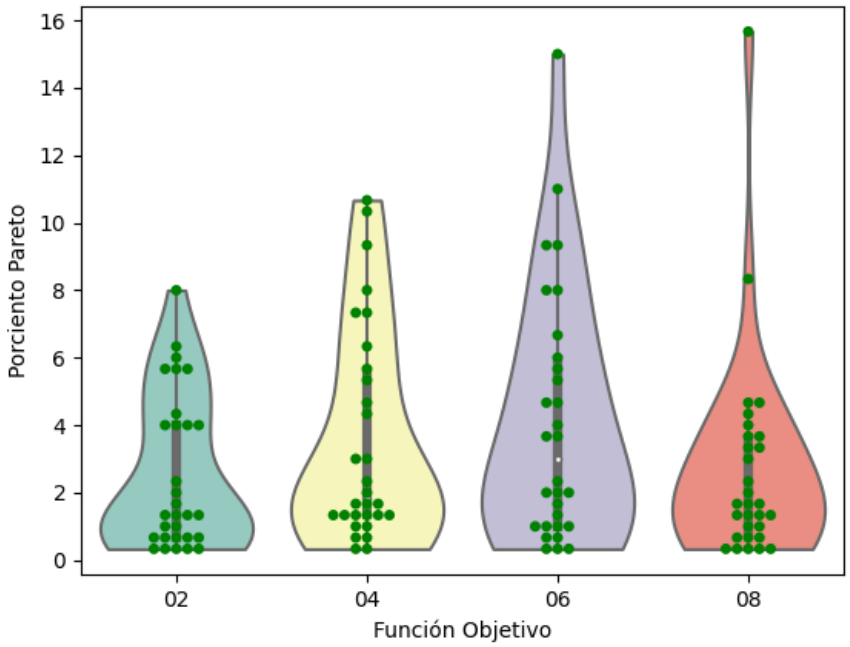

Figura 1: Diagrama violín por función de objetivo contra el porcentaje de resultados de Pareto

5. Conclusión

Como conclusión en base a la figura 1 se puede ver que los puntos comienzan distribuidos a lo largo del violín pero no llegan a más de ocho porciento y conforme aumenta la función objetivo la cantidad de porcentajes comienza a bajar para asentarse en la parte inferior pero llega a haber porcentajes que llegan más lejos (dieciséis porciento) aunque no sean muchos.

Referencias

[1] Samaniego E. Práctica 11, 2021. URL https://github.com/edson-samaniego/simulation-2021/tree/main/Pr%C3%A1ctica-11.

[2] Schaeffer E. Frentes de Pareto, 2021. URL https://github.com/satuelisa/Simulation/tree/master/ParetoFronts.

[3] Schaeffer E. *Práctica 11*, 2021. URL https://elisa.dyndns-web.com/teaching/comp/par/p11.html.