Parte 2

1.

Para que el programa permita el uso de n brazos robóticos debemos realizar las siguientes modificaciones:

1. Agregar un predicado que corresponda a los brazos robóticos y que este admita "n"brazos

```
brazorobot(1..n).
```

2. Aumentar la aridad del predicado ejecutar para que considere que brazo robot ejecuta el movimiento. Para esto modificamos las siguientes líneas

```
    0{ejecutar(X,Y,T,R) : movimientoposible(X,Y,T)}1 :-
            tiempo(T), T!=bound, brazorobot(R).
    origen_movimiento(X,T) :- ejecutar(X,Y,T,R).
    destino_movimiento(Y,T) :- ejecutar(X,Y,T,R).
    sobre(X,Y,T+1) :-
            bloque(X),lugar(Y),tiempo(T),
            ejecutar(X,Y,T,R).
    #show ejecutar/4.
```

3. Agregar una regla que establezca que dos robots no pueden mover el mismo bloque

```
    :- ejecutar(X,Y,T,R1), ejecutar(X,Z,T,R2), R1!=R2.
```

4. Agregar una regla que establezca que dos robots no pueden mover un bloque hacia un mismo bloque

```
    :- ejecutar(X,Y,T,R1), ejecutar(Z,Y,T,R2), R1!=R2, Y!=mesa.
```

5. Agregar una regla que establezca que un robot no puede mover un bloque a otro bloque que está moviendo otro robot

```
    :- ejecutar(X,Y,T,R1), ejecutar(Z,X,T,R2), R1!=R2.
```

2.

Para la siguiente evaluación experimental se diseñaron 20 problemas que se pueden observar con detalle en el siguiente link. A partir de los tiempos de solución y el instante de tiempo en el que se logra el objetivo se realizaron los siguientes gráficos (donde se incluye también los casos con 1,3 y 5 brazos robóticos).

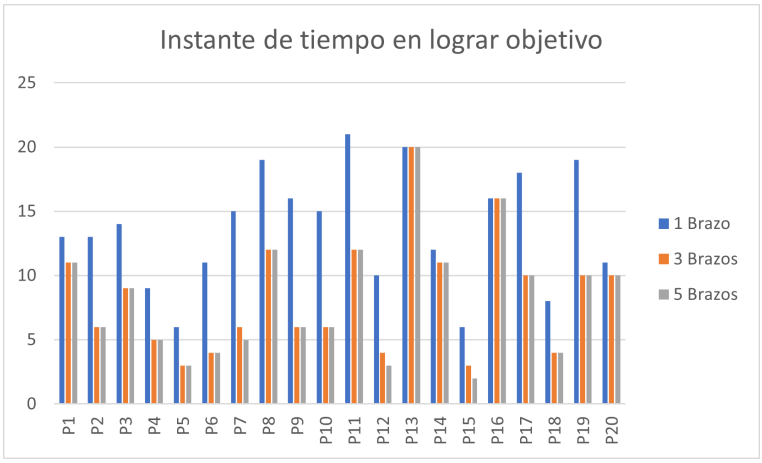

Figura 1: Tiempo en el que el problema logra el objetivo

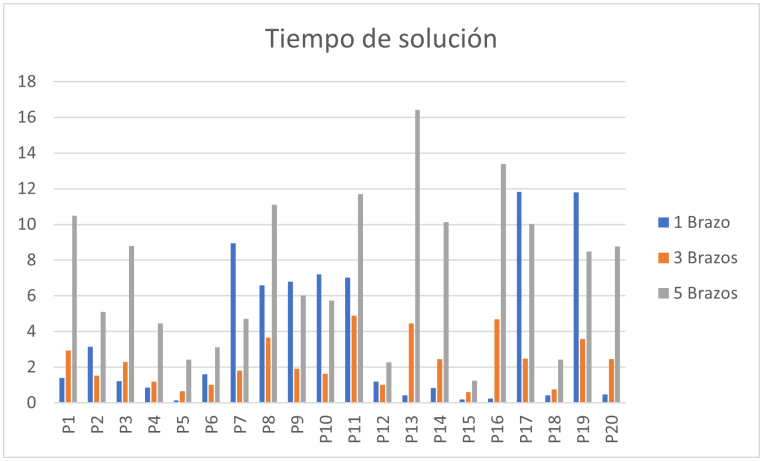

Figura 2: Tiempo de solución del programa

3.

A partir del gráfico de la *Figura 1* podemos ver que el las variantes con solo 1 brazo nunca demorarán menos que con 3 brazos y al mismo tiempo, las con 3 brazos nunca demorarán menos que con 5. Esto es simplemente debido a que claramente contar con más brazos robóticos permite mover más bloques al mismo tiempo por lo que entre más brazos el problema se resolverá en igual o menos tiempo. Por otro lado, aquellos problemas con mayor cantidad de torres se resuelven en un mayor instante de tiempo, al igual aquellos donde el objetivo requería más bloques en ciertos lugares específicos.
A partir del gráfico de la *Figura 2* podemos ver que generalmente la variante con 5 brazos es la que más tiempo tarda, esto puede deberse a que generalmente la variante con 5 tardaba lo mismo o un tiempo menos que la con 3 pero debía analizar muchos más casos para poder llegar al correcto, por lo que tardaría más tiempo en ejecutarse. Muchas veces fue la variante con 1 brazo la cual tardó bastante en encontrar una solución. Esto se debe simplemente al hecho que el instante en el que el problema lograba su objetivo era bastante alto por lo que el programa debía recorrer muchas más unidades de tiempo que para 3 y 5 brazos. Lo anterior podría verse en el *P19* donde para 1 brazo tomo aproximádamente 12 segundos (casi 4 segundos más que la variante con 5 brazos) pero el instante de tiempo es casi el doble. Con todo esto podemos

decir que grandes influyentes en los tiempos fueron el instante de solución del problema y la cantidad de combinaciones que debe recorrer el programa para llegar al resultado donde ambos causaron notorias diferencia entre casos.

4.

Para hacer que el programa se genere utilizando el menor numero de acciones debemos minimizar las apariciones de *ejecutar* moviendo la menor cantidad de veces los bloques. Es por esto que utilizamos la siguiente regla:

```
#minimize {1, X : ejecutar(X,Y,T,R)}.
```

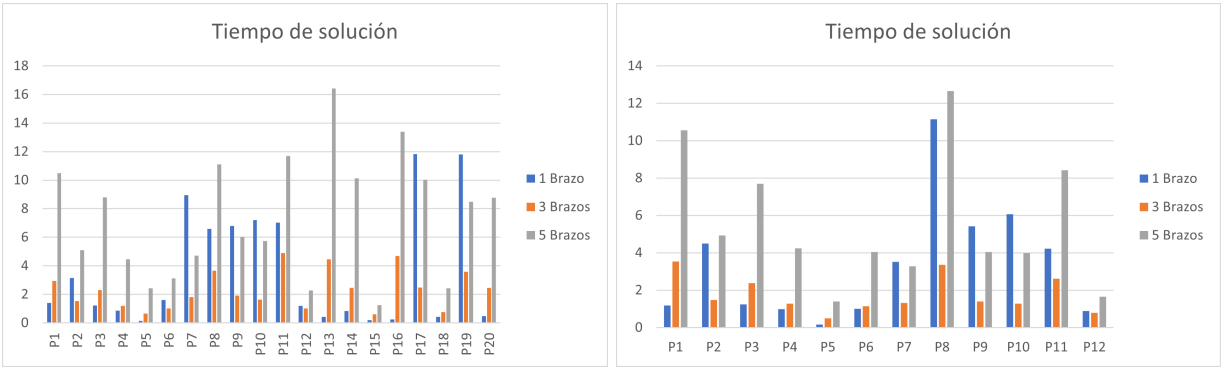

Figura 3: Comparación del tiempo de solución del programa con y sin minimize

Al comparar ambos tiempos sin y con minimize vemos que en general minimize tardó más tiempo (gráfico de la derecha). Esto puede ser debido al trabajo extra de encontrar una solución que minimice el número de acciones.