Traveling PokeManGo Problem[1]

Nuestro grupo de juego de Pokèmon Go quiere conquistar los 5 gimnasios más importantes de la Ciudad. Para eso necesitamos planificar los viajes entre los gimnasios de manera que podamos hacerlo en un tiempo razonable y, además, pasando por todas las pokeparadas que podamos.

Los caminos están modelados de esta forma:

```
% camino(GymA, GymB, DuracionViajeMin, CantidadPokeparadas).
camino(plaza_de_mayo, congreso, 9, 15).
camino(plaza_de_mayo, teatro_colon, 11, 15).
camino(plaza_de_mayo, abasto_shopping, 19, 28).
camino(plaza_de_mayo, cementerio_recoleta, 26, 36).
camino(congreso, teatro_colon, 10, 11).
camino(congreso, cementerio_recoleta, 15, 16).
camino(teatro_colon, abasto_shopping, 13, 20).
camino(teatro_colon, cementerio_recoleta, 17, 24).
camino(abasto_shopping, cementerio_recoleta, 27, 32).
```

Nuestro primer objetivo es hacer **`mejorTour/2`**, que relacione un límite de minutos de duración y un tour detallado (siendo un tour, una lista de etapas) que nos diga de qué gimnasio a qué gimnasio debemos ir para poder **pasar por todos, pero sólo una vez por cada uno**, maximizando la cantidad de pokeparadas y dentro de nuestro límite de duración. Para esto, pensar en:

1. ¿Qué representa “una etapa” o elemento en el tour? Hay dos enfoques posibles principales:
    a. Cada etapa es un nuevo gimnasio.
    b. Cada etapa es un trayecto y se corresponde con un camino que conecta un gimnasio con otro. Recomendamos que sea alguno de estos, pero si piensan en otro modelo también puede ser válido. Consultar.
2. Tener en cuenta que, en un camino, no es necesario ir de un gimnasio A hacia otro B: bien se podría recorrer el trayecto en el sentido inverso, de B hacia A. Definir una abstracción que refleje lo anterior.
3. Generar una secuencia de etapas que pasa por todos los gimnasios, sin repetir los mismos y sin exceder el límite de tiempo.
    a. Tip: Puede ser más sencillo primero armar la secuencia y luego controlar el total, pero también puede hacerse todo junto.
4. Pueden ser útiles los predicados:
    a. **`permutation/2`**, el cual relaciona 2 listas cuando la segunda es una permutación de la primera[2]. Este predicado es inversible para la segunda lista.

---

[1] Ver “Traveling Salesman Problem”.
[2] Es decir, tiene los mismos elementos, pero pueden estar cambiados de posición.

   b. **list_to_set/2**, que relaciona una lista con otra, si la segunda contiene los mismos elementos que la primera pero sin repeticiones de los mismos. Es inversible para la segunda lista y respeta el orden de aparición de los elementos (primera vez de c/u).
5. En base a esas consideraciones, implementar **mejorTour/2**, que debe cumplirse para aquella secuencia de pasos que maximice la cantidad de paradas en su trayecto, siempre dentro del límite de tiempo establecido. No necesita ser inversible para el límite de tiempo.

Ahora agregamos información: Un gimnasio está, en un determinado momento, ocupado por un equipo de un color (rojo, azul, amarillo).

6. Agregar está información a la base de conocimiento para todos los gimnasios anteriores, sin modificar lo realizado hasta ahora. El shopping tendrá un color, el Congreso otro, y los demás un tercero.
7. Implementar **estaSitiado/2**. Un gimnasio está sitiado cuando todos sus vecinos están ocupados por equipos de un mismo color, que no es el mismo del equipo gimnasio. Un gimnasio "vecino" es aquel conectado con un camino directamente, es decir, sin pasar por otros gimnasios en medio.

Validar el modelo con los tests dados a continuación. En el caso del tour, ajustar el formato de las etapas del mismo de acuerdo a la representación elegida por ustedes (en este caso, es de gimnasios, la opción A entre las planteadas).

```
:- begin_tests(mejorTour). % Hay 2 mejores tours ya que ir o volver por cada camino es
igual

test("El mejor camino es shopping, teatro, plaza, congreso, cementerio", nondet):-
  mejorTour(50, [abasto_shopping, teatro_colon, plaza_de_mayo, congreso,
cementerio_recoleta]).

test("El mejor camino es cementerio, congreso, plaza, teatro, shopping", nondet):-
  mejorTour(50, [cementerio_recoleta, congreso, plaza_de_mayo, teatro_colon,
abasto_shopping]).

:- end_tests(mejorTour).

:- begin_tests(estaSitiado).

/* Congreso es de un color y sus vecionos son de otro e iguales entre sí, ya que no está
conectado por un camino con el Shopping. Está sitiado. */
test("El Gym del Congreso está sitiado", nondet):-
    estaSitiado(congreso).

/* Plaza de Mayo tiene vecinos del mismo color, no está sitiado. */
test("El Gym de Plaza de Mayo no está sitiado", fail):-
    estaSitiado(plaza_de_mayo).

:- end_tests(estaSitiado).
```