Magister en Inteligencia Artificial
Facultad de Ingeniería y Ciencias

Trabajo Sesión Nº 1
Técnica de Búsqueda y Heurísticas

Nombre Eduardo Carrasco
Profesor Jordi Pereira

2021

1. Preguntas Cortas:

a. En clase se vio una posible formulación del problema SEND+MORE=MONEY usando MiniZinc. Podría plantear otra? Discuta las ventajas e inconvenientes de la formulación alternativa.

El planteamiento formulado del problema SEND+MORE=MONEY usando MiniZinc consistió en definir las variables: S, E, N, D, M, O, R, Y que pertenecen a un conjunto de valores enteros [0,9].

a.1. Original:

Variables:

```
var 1..9: s;    var 0..9: e;    var 0..9: n;    var 0..9: d;
var 1..9: m;    var 0..9: o;    var 0..9: r;    var 0..9: y;
```

Restricciones:

- Lo primero fue definir el espacio entre millares, decenas, centenas y unidades que permitieran identificar las variables en una suma por orden de precedencia.

```
constraint
        1000*S + 100*E + 10*N + D
      + 1000*M + 100*O + 10*R + E
= 10000*M + 1000*O + 100*N + 10*E + Y
```

- Lo segundo fue definir que todas las variables fueran distintas, para ello en MiniZinc se debe especificar lo siguiente:
  `constraint alldifferent ([s,e,n,d,m,o,r,y]);(include "globals.mzn";` cargar esta librería nos permite usar esa herramienta.

a.2. Alternativo:

Si queremos plantear una formulación distinta pero que cumpla con las mismas restricciones, podríamos agrupar el conjunto de variables de manera de obtener el siguiente planteamiento:

Variables:

```
set of int: DIGIT = 0..9
set of int: DIGIT1 = 1..9
var DIGIT1: S;  var DIGIT: E;   var DIGIT: N;   var DIGIT: D;
var DIGIT1: M;  var DIGIT: O;   var DIGIT: R;   var DIGIT: Y;
```

Restricciones:

```
Constraint
```

$$10^3(s+m) + 10^2(e+o) + 10(n+r) + d + e = 10^4 m + 10^3 o + 10^2 n + 10e + y$$

- De igual manera, en minizinc, podemos generar que cada uno sea diferente del otro con la siguiente herramienta:
  Constraint s!=e; Constraint s!=n; Constraint s!=d;
  Constraint s!=m; Constraint s!=o; Constraint s!=r;
  Constraint s!=y; (así con cada uno de las variables de manera que no sean iguales).

  La solución alternativa planteada no es la más eficiente puesto que no utiliza las librerías disponibles del sistema al diferenciar cada variable distinta de otra; esto afectará directamente en el tiempo en el que demora efectuar el planteamiento de una tarea y por otra parte, en el tiempo de cálculo y procesamiento que requiere el computador; que si bien, no es significativo en este tipo de problemas, puede resultar relevante al analizar grandes cantidades de datos con diversas restricciones.

b. En clase se ha mostrado métodos de búsqueda en que el conjunto de opciones que pueden tomarse es finito (por ejemplo, sólo se puede ir arriba, abajo, izquierda y derecha) ¿Qué opciones cree que podría aplicar para resolver problemas en que el conjunto de opciones no es finito? (por ejemplo, buscar un camino en el desierto).

La búsqueda es un proceso que a través de pasos, permite encontrar un estado deseado (objetivo). Estos pasos son definidos por el conjunto de opciones que permiten expandir el estado actual, aplicando una función sucesora, y generando con ello un nuevo conjunto de estados.
Para el ejemplo propuesto, si tenemos un conjunto de opciones que no es finito, tendremos un número infinito de caminos, por lo tanto, al ser ejemplificados en un árbol de búsqueda, existirán un conjunto infinito de nodos.
Para este conjunto infinito de opciones es posible efectuar técnicas como lenguaje restrictivo, además de restricciones lineales y no lineales; que permita acotar el problema a un dominio finito.
Por lo anterior, se produce un cambio en el procedimiento de búsqueda que permita acotar el número de opciones, evitando lo que no lleva a nada.
En el ejemplo del problema (desierto), nuestros pasos tienen un costo, por lo cual, no podemos probar infinitas opciones; si tenemos información de cómo medir (distancia) y hacia donde orientarnos directamente (búsqueda informada); aplicaríamos una función heurística como camino más corto y llegaríamos a la ciudad más cercana sin las restricciones de tener que tomar una izquierda, derecha, arriba o abajo.
Por otro lado, si nuestro caso en el desierto corresponde a una búsqueda no informada (conociendo sólo el estado objetivo, ciudad);

debemos aplicar la mayor cantidad de restricciones posibles que nos permitan disminuir la cantidad de nodos en nuestro árbol; una buena estrategia de búsqueda sería aplicar de manera inicial inferencias para posteriormente aplicar técnicas de búsqueda, que permitiera a disminuir la cantidad de opciones posibles (pasando este problema de infinito a finito), por ejemplo, detectar la presencia de agua o nodos de interés, como estado deseado intermedio.

c. En clase se ha demostrado la diferencia entre la búsqueda informada y desinformada. ¿En qué caso cree que es conveniente el uso de búsqueda desinformada por encima de la búsqueda informada?

La búsqueda no informada sólo es conveniente cuando no se tiene la información acerca del resto de los estados, es adecuada usarla en un laberinto no explorado, en un problema en el que se ve enfrentada una máquina por primera vez y del que no se tiene ningún conocimiento más que el estado deseado (objetivo).
Lo anterior, debe ir concatenado con un algoritmo capaz de reconocer (recordar) cada estado que ha visitado y aquellos que han sido expandidos, evitando bucles.
De acuerdo a lo planteado, en ningún caso sería correcto no utilizar la información disponible.

d. El problema de las n-reinas consiste en poner n-reinas en un tablero de ajedrez con nxn casillas (una grilla cuadrada) de tal manera que ninguna reina pueda "matar" a otra en un movimiento (una reina puede matar a otra si se encuentra en una misma fila, columna o diagonal). Describa el dominio de las variables y exprese de alguna manera las condiciones.

Si debemos posicionar n reinas en un tablero o grilla cuadrada de nxn, de forma que no se ataquen (misma fila, columna o diagonal), entonces debemos considerar lo siguiente:
Este problema es de dominio discreto (variables acotadas).
Las variables son las reinas Qn "cantidad de queen" [Q1, Q2, Q3,…, Qn] que pertenezcan a un conjunto de [1, n].
Para este ejemplo, se usará un n=4, por lo tanto las variables son Q1, Q2, Q3, Q4 pertenecen a un conjunto [1,4]; en otras palabras, la variable Q1 puede encontrarse en un dominio (posición en la primera columna del tablero) del 1 al 4.
Para ejemplificar lo anterior, Q1=3; Q2=1.

	Q2		
Q1			

Las restricciones, por lo tanto, equivalentes al problema de ejemplo serían: i que pertenece [1,4] y j que pertenece a [i+ 1,4], se tiene que:
Qi ≠ Qj (reinas en filas no pueden ser iguales a reina en columna).
Qi + i ≠ Qj + j (reinas no pueden estar en una misma diagonal directa).
Qi - i ≠ Qj - j (reinas no pueden estar en una misma diagonal inversa).

2. Preguntas Cortas:

a. Proponga dos modelos de MiniZinc para el problema de las n-reinas. Compárelo con los términos de eficiencia (en otras palabras, rapidez en la solución).

Objeto comparar dos modelos de MiniZinc para el problema de las n-reinas, se presentan dos cuadros con el código utilizado:

```
% Use this editor as a MiniZinc scratch book
include "globals.mzn"; % usamos esta librería que incluye alldifferent

% Parámetro
int: n;

% Definición de variables
array [1..n] of var 1..n: rows;

%Definición de condiciones:
 constraint alldifferent (rows);
 constraint forall (i, j in 1..n where i < j)(
% rows[i]-rows[j]     /\ %reemplaza la función alldifferent
 rows[i] -rows[j]!= i - j /\
 rows[i] -rows[j]!= j - i );


% búsqueda y solución
 solve satisfy;


% salida
 output [
 show(rows)++"\n"
 ];
 n = 8;
```

```
% Use this editor as a MiniZinc scratch book
include "alldifferent.mzn";% usamos esta librería que sólo incluye alldifferent


% Parámetro
int: n;

% Definición de variables
array [1..n] of var 1..n: q; % reina en la columna i está en la fila q[i]

%% Definición de condiciones:
constraint alldifferent(q);                    % distintas filas
constraint alldifferent([ q[i] + i | i in 1..n]); % disntintas diagonales
constraint alldifferent([ q[i] - i | i in 1..n]); % hacia arriba y abajo

% búsqueda y solución
solve :: int_search(q, first_fail, indomain_min)
      satisfy;


% salida
output [ if fix(q[j]) == i then "Q" else "." endif ++
    if j == n then "\n" else "" endif | i,j in 1..n]
```

En ambos modelos, se utilizó n=8, obteniendo los siguientes resultados:

```
Running modelo N-reinas2.mzn
[4, 2, 7, 3, 6, 8, 5, 1]
----------
Finished in 339msec
```

```
Running nqueens.mzn, additional arguments -D n=8;
Q.......
......Q.
....Q...
.......Q
.Q......
...Q....
.....Q..
..Q.....
----------
Finished in 95msec
```

En ambos modelos se obtuvo un resultado que cumple las reglas propuestas, pero los tiempos necesarios para generar solución son muy diferentes.
Efectuado un análisis, la cantidad de restricciones y entendimiento de las restricciones que se le ingresan al algoritmo, influyen en su eficiencia. A mayor cantidad de procesos por compilar, mayor tiempo de procesamiento.