Programación con Restricciones

Carlos Castro
Departamento de Informática
UTFSM

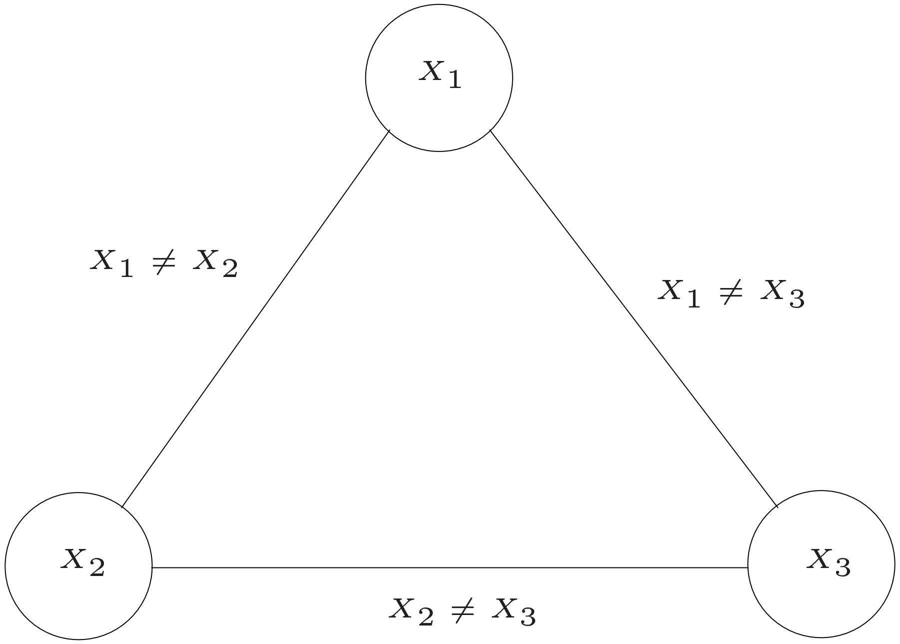
Plan

1. Problemas de satisfacción de restricciones
2. Técnicas de resolución de CSPs
3. Un enfoque deductivo de la resolución de CSPs
4. Modelamiento basado en restricciones

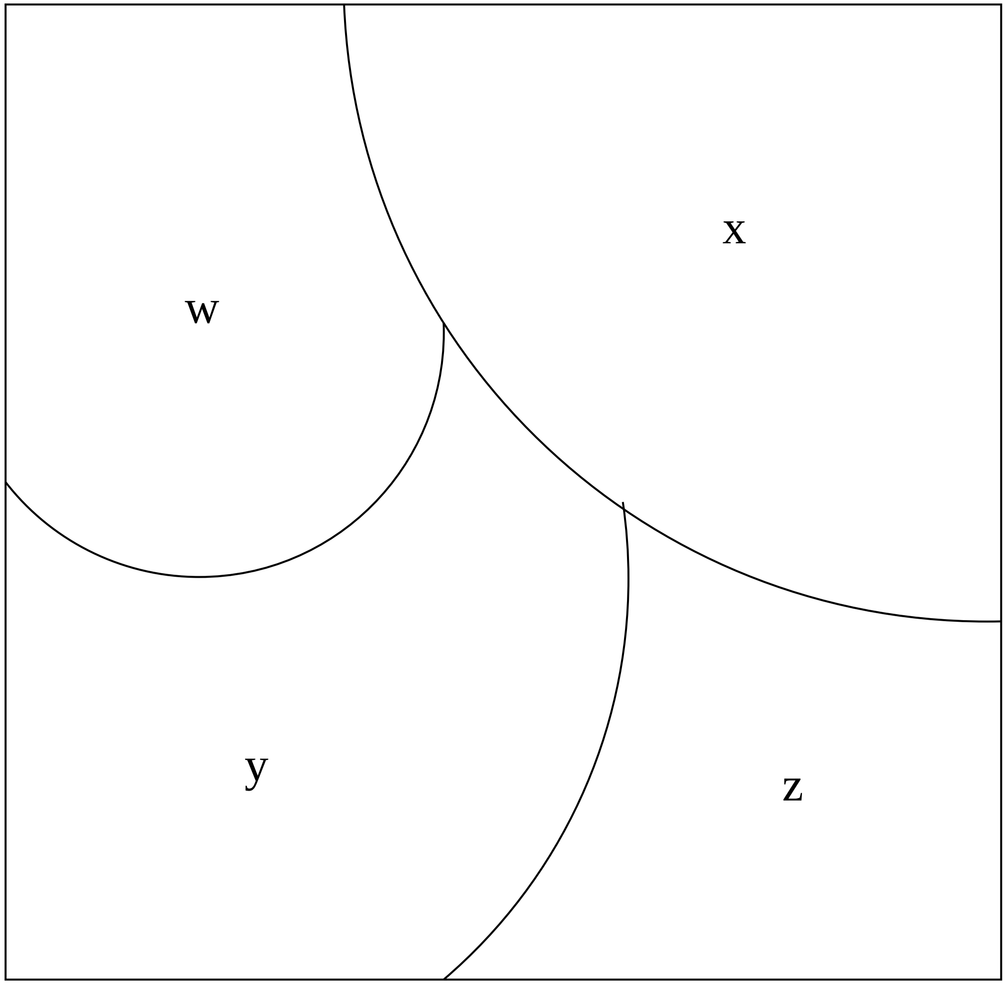
Problemas de satisfacción de restricciones (CSP)

$$CSP = \langle X, D, C \rangle$$

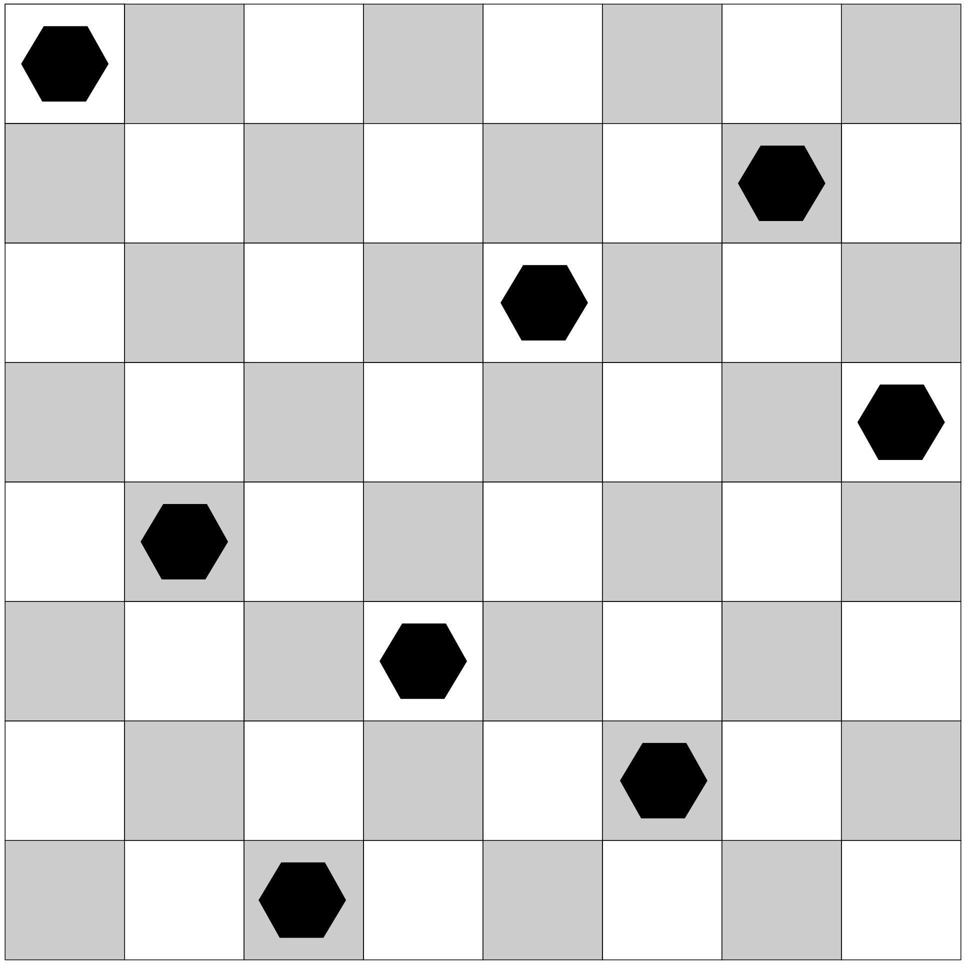
$X = \{X_1, X_2, X_3\}$, $\quad D_{X_1} = D_{X_2} = D_{X_3} = \{1, 2, 3\}$,
$C = \{X_1 \neq X_2, X_1 \neq X_3, X_2 \neq X_3\}$



CSP: Búsqueda de una solución



CSP: Búsqueda de todas las soluciones



Problemas de optimización con satisfacción de restricciones (CSOP)

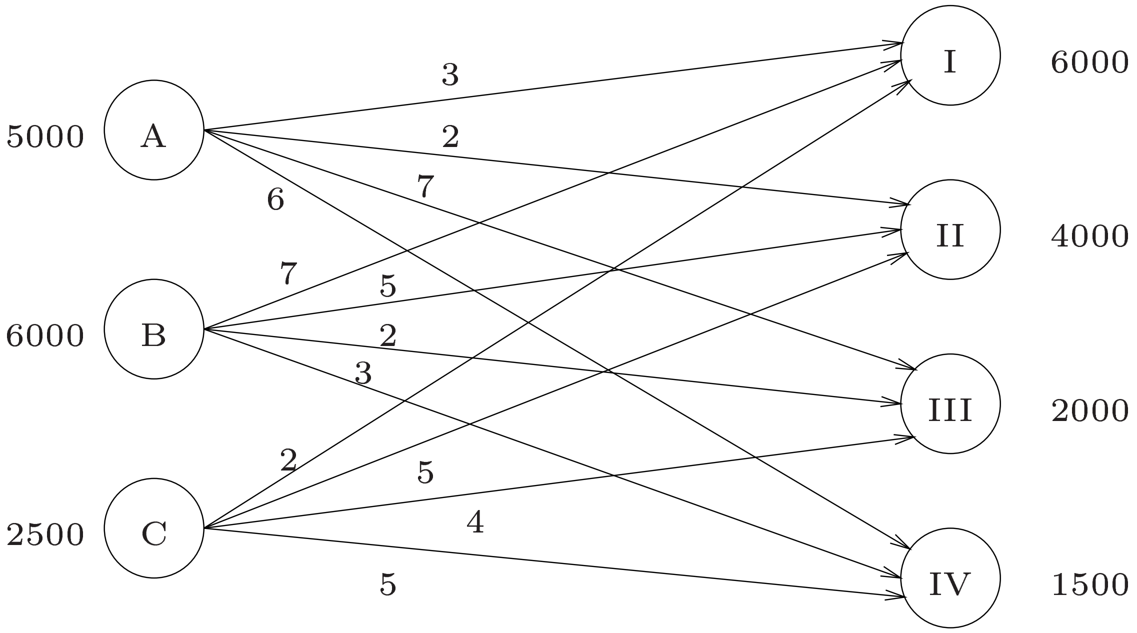
$$CSOP = \langle X, D, C, f \rangle = \langle CSP, f \rangle$$

Objetivo:

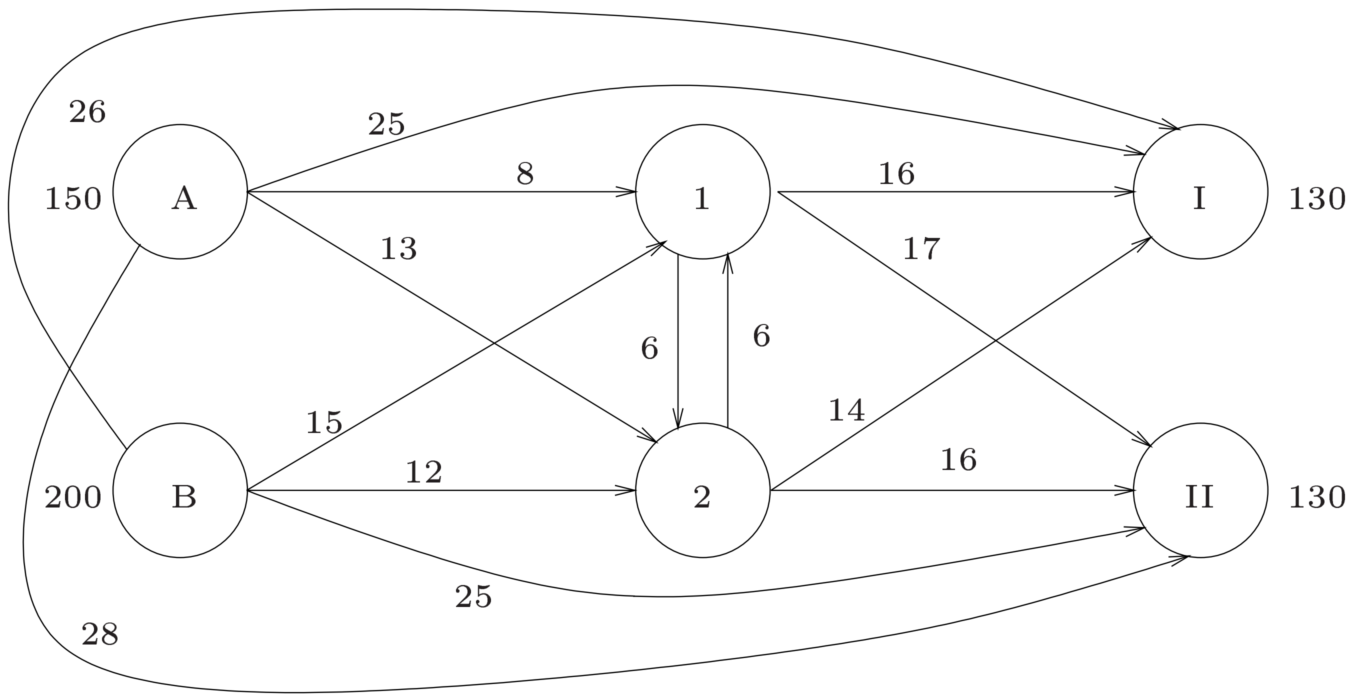
- La(s) mejor(es) solución(es) con respecto a f satisfaciendo C

Aplicaciones

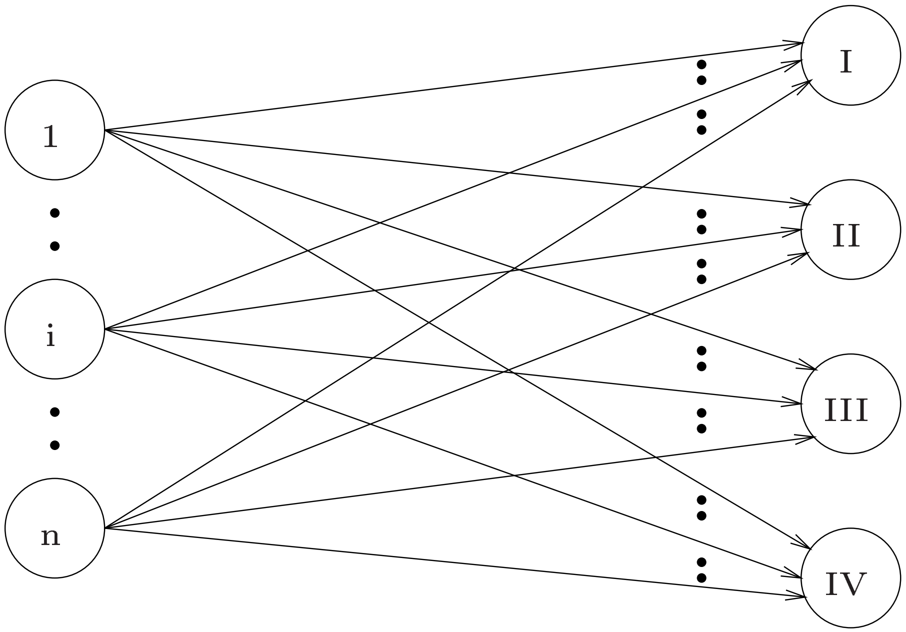
El problema de transporte



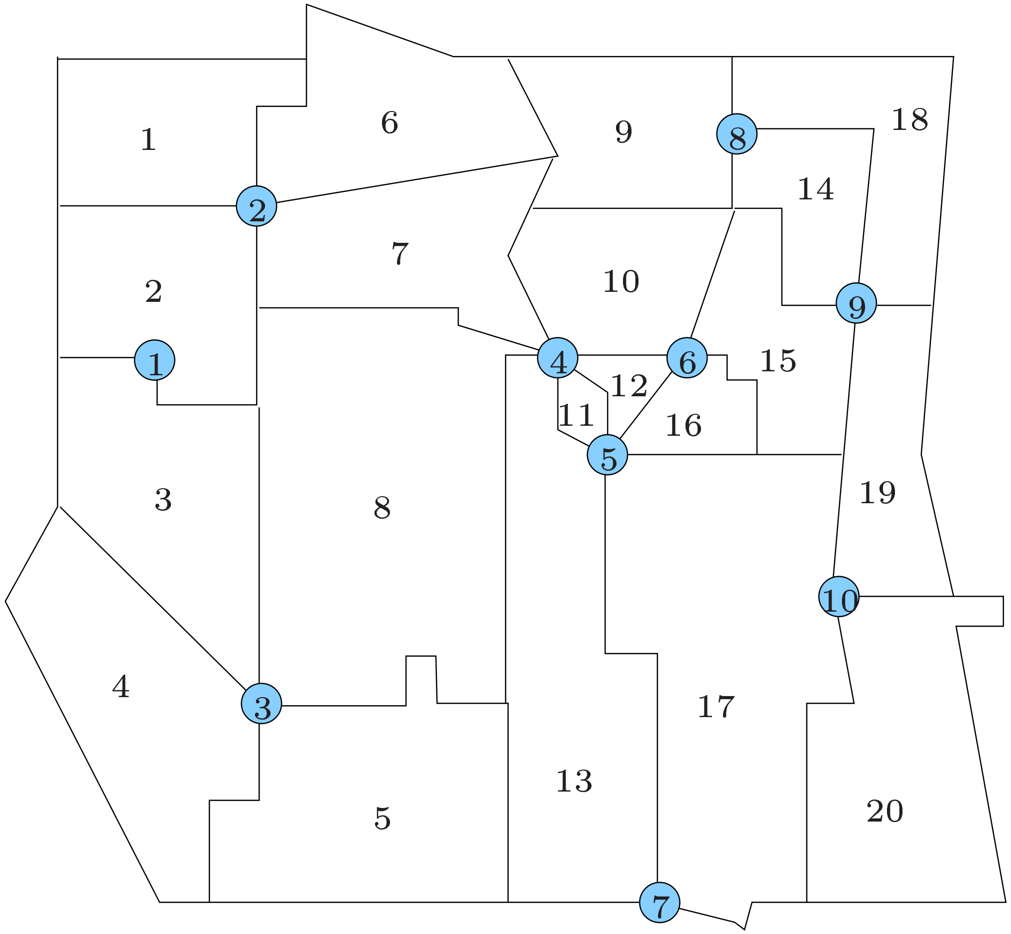
El problema de transbordo



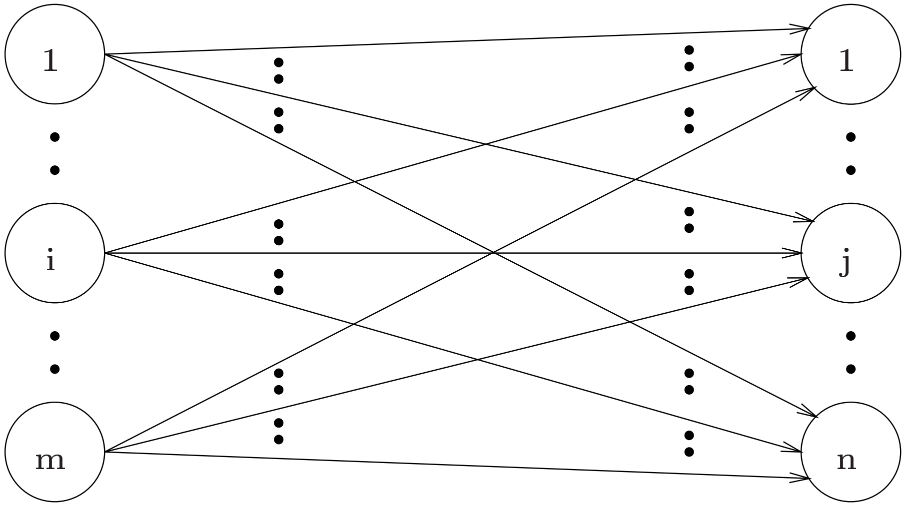
El problema de localización



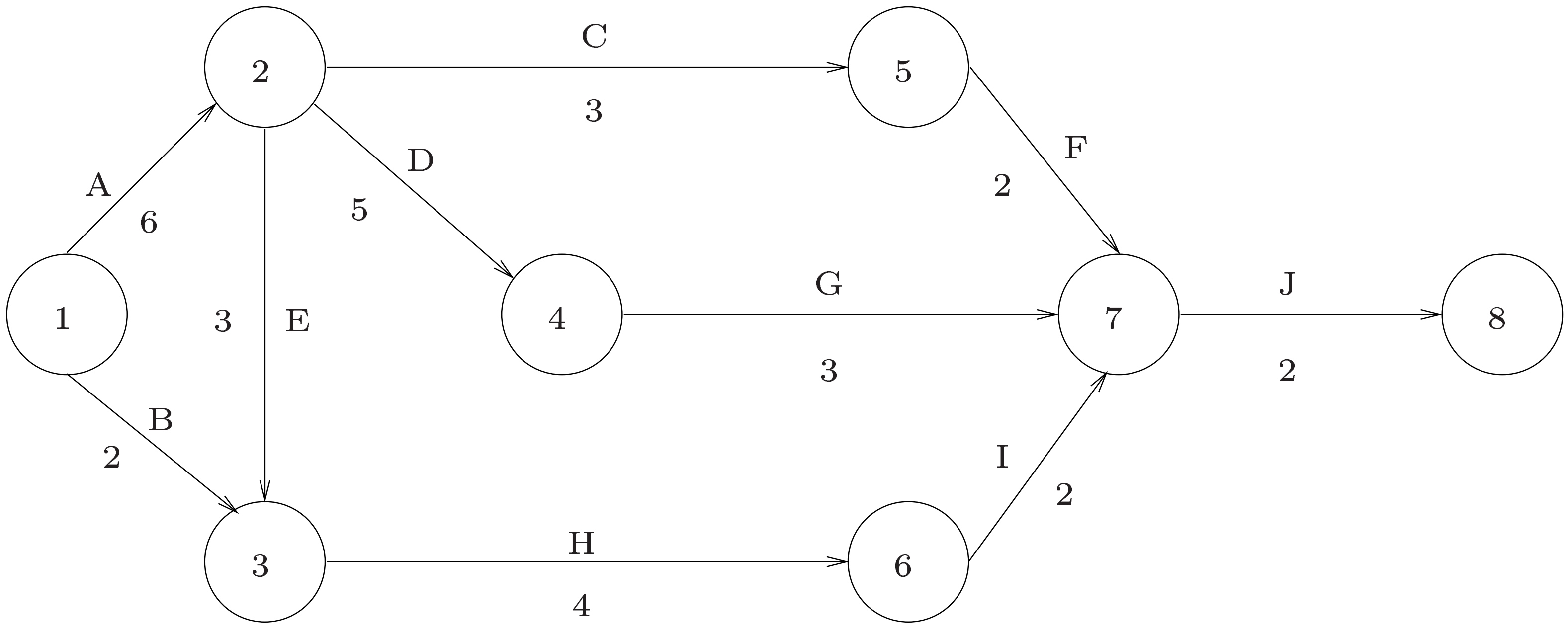
El problema de cobertura



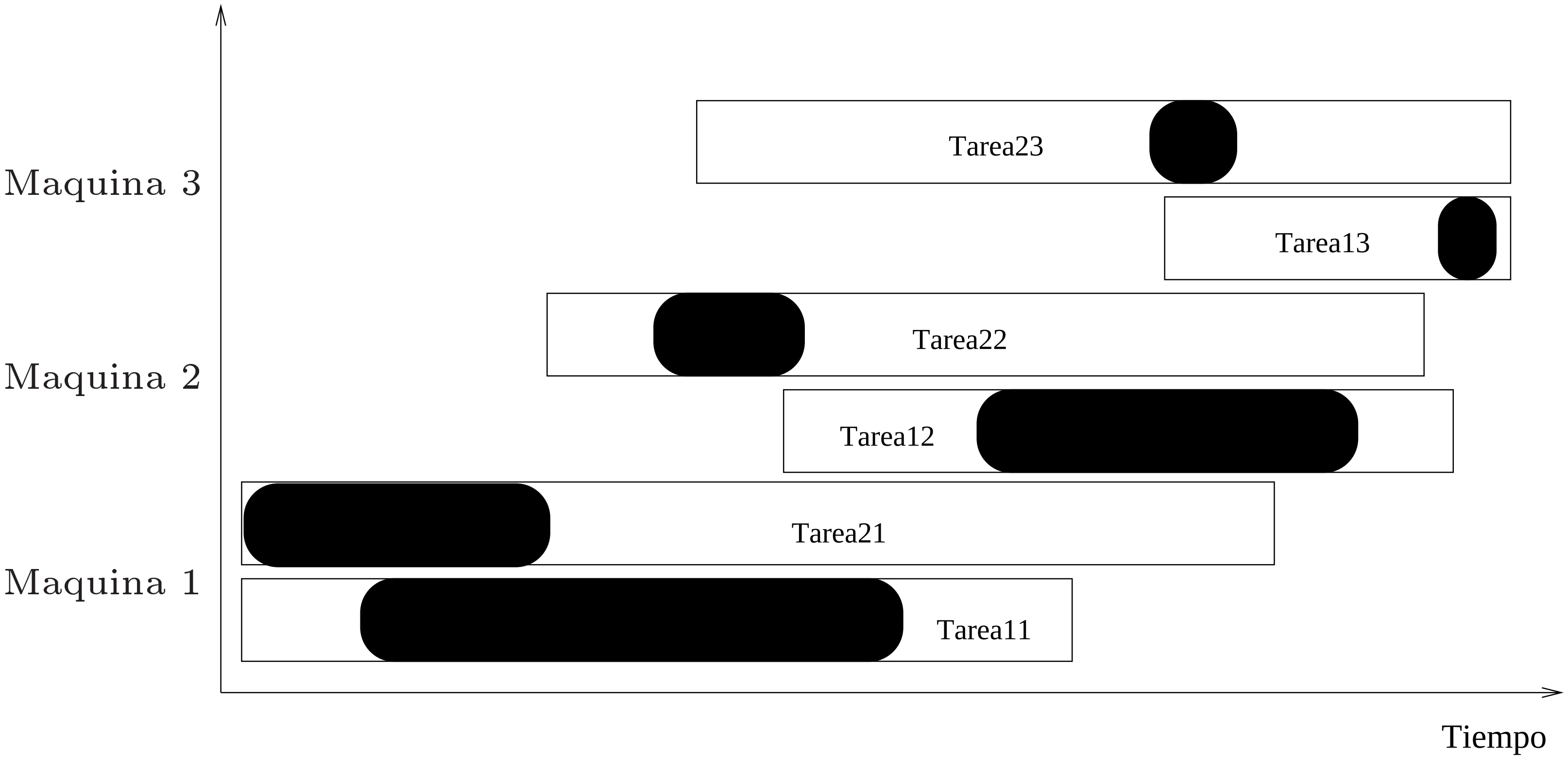
Diseño de redes



Programación de actividades



Job-shop

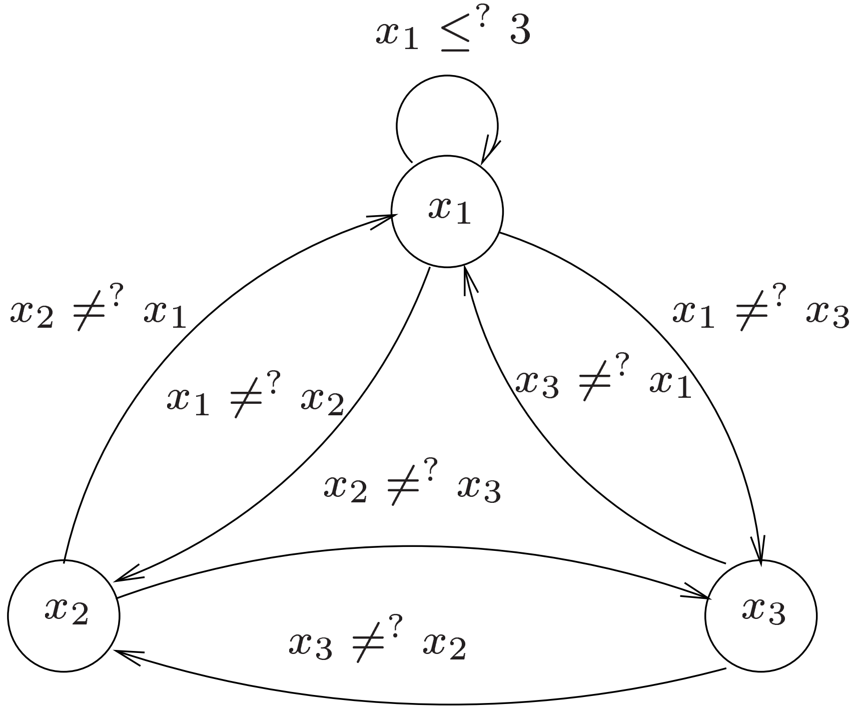

Tarea23
Maquina 3
Tarea13
Tarea22
Maquina 2
Tarea12
Tarea21
Maquina 1
Tarea11
Tiempo

Representación gráfica

CSPs binarios

$$\mathcal{X} = \{x_1, x_2, x_3\} \text{ con } \mathcal{D}_{x_1} = \mathcal{D}_{x_2} = \mathcal{D}_{x_3} = \{1, 2, 3, 4, 5\} \text{ y}$$

$$\mathcal{C} = \{x_1 \leq^? 3, x_1 \neq^? x_2, x_1 \neq^? x_3, x_2 \neq^? x_3\}$$

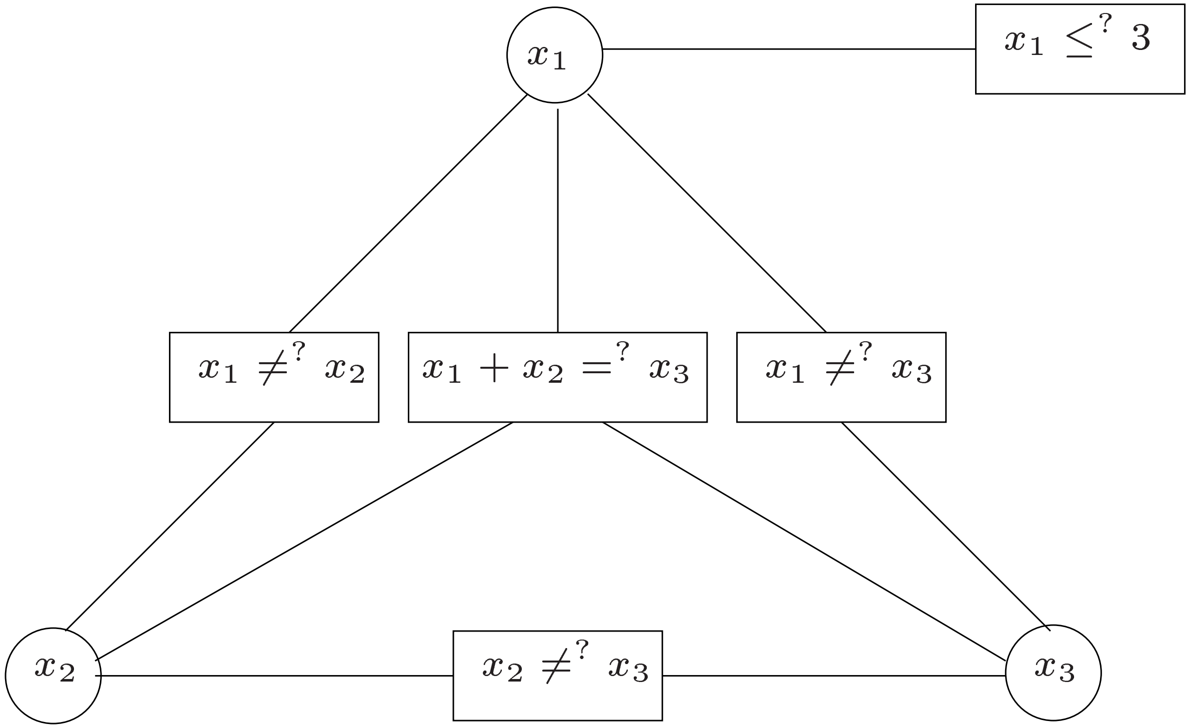


Representación gráfica

CSPs n-arios

$\mathcal{X} = \{x_1, x_2, x_3\}$ con $\quad \mathcal{D}_{x_1} = \mathcal{D}_{x_2} = \mathcal{D}_{x_3} = \{1, 2, 3, 4, 5\}$ y

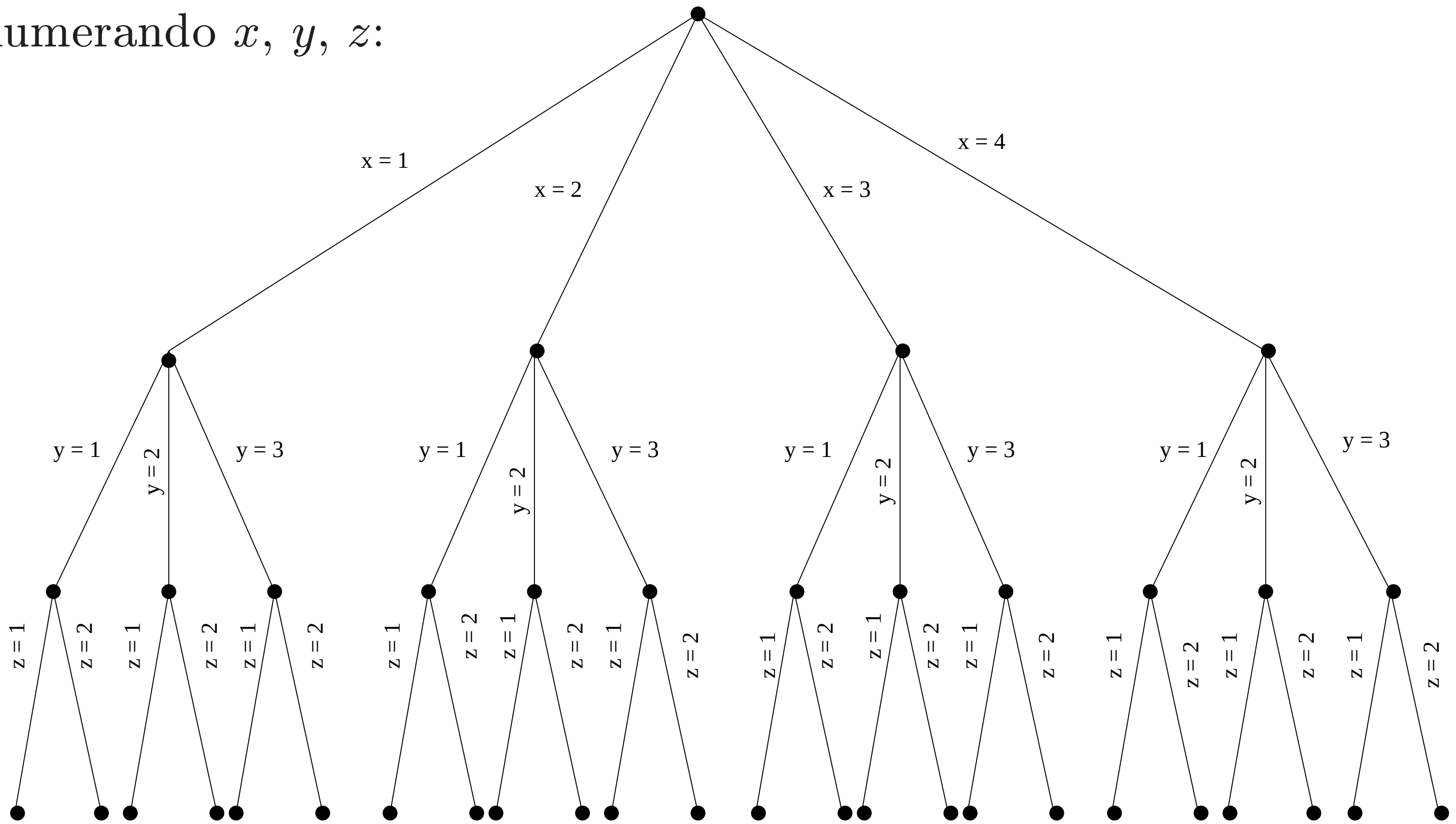
$C = \{x_1 \leq^? 3, x_1 \neq^? x_2, x_1 \neq^? x_3, x_2 \neq^? x_3, x_1 + x_2 =^? x_3\}$



Espacio de búsqueda

$$x \in \{1,2,3,4\}, y \in \{1,2,3\}, z \in \{1,2\}$$

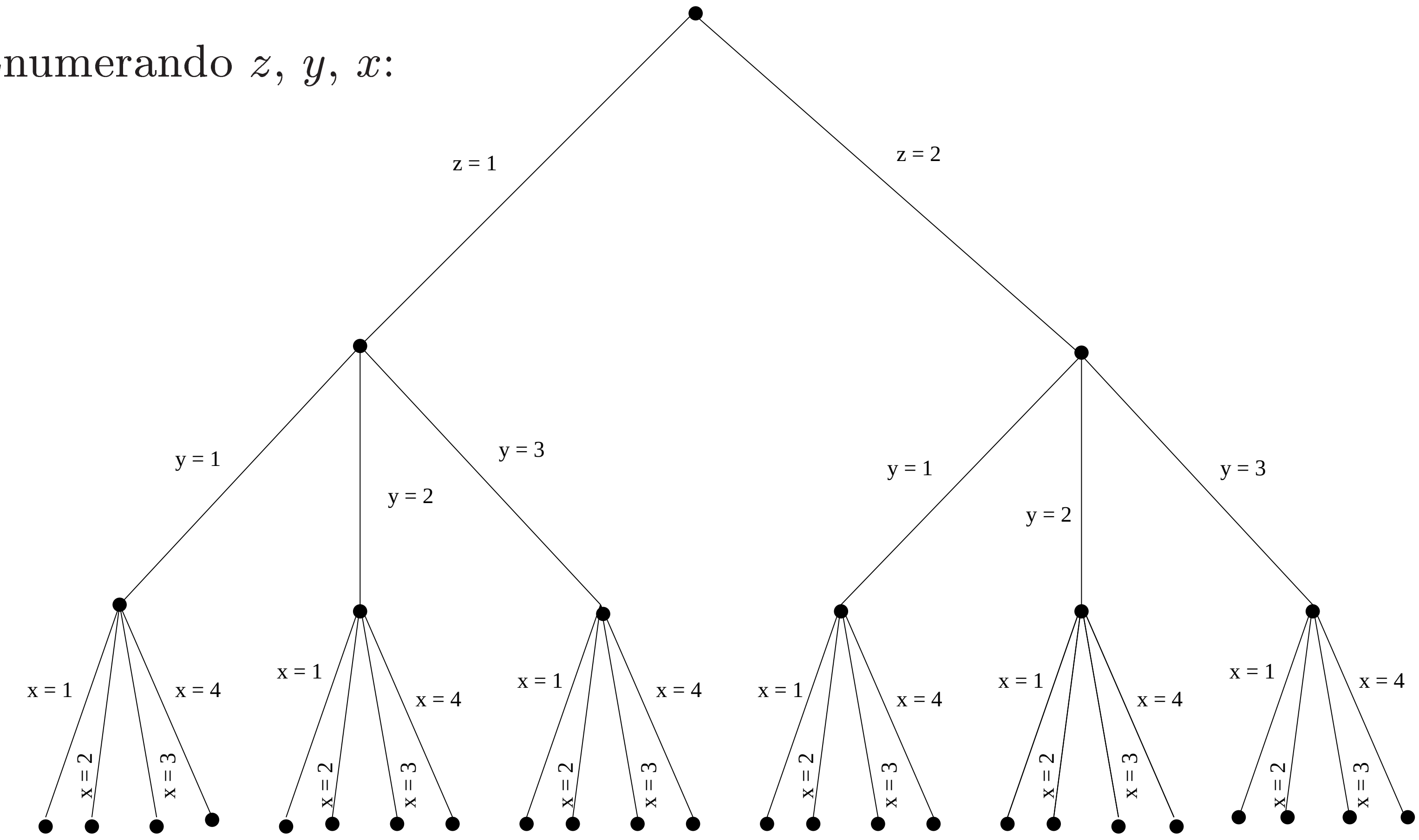
Enumerando x, y, z:



Espacio de búsqueda

$$x \in \{1,2,3,4\}, y \in \{1,2,3\}, z \in \{1,2\}$$

Enumerando z, y, x:



Espacio de búsqueda

- Profundidad del árbol = Número de variables
- Orden de enumeración fijo $\rightarrow$
  topología idéntica de sub-árboles de igual nivel en cada rama
- Cantidad de hojas del árbol:

$$| D_{x_1} | \times \ldots \times | D_{x_n} |$$

  independiente del orden de enumeración

Espacio de búsqueda

- Total de nodos enumerando $x_1, \ldots, x_n$:

$$1 + \sum_{i=1}^{n} (\Pi_{j=1}^{i} \mid D_{x_j} \mid)$$

depende del orden de enumeración

Por lo tanto, enumerar considerando tamaño de dominios en:

- orden decreciente $\rightarrow$ máximo número de nodos internos
- orden creciente $\rightarrow$ mínimo número de nodos internos

Resolución de CSPs

Enumeración de valores

$+$

Verificación de consistencia local

Resolución de CSPs

- Técnicas de enumeración
- Técnicas de verificación de consistencia local
- Técnicas híbridas

Técnicas de enumeración: *Generate & Test*

$$x \in^? [1,\ldots,100] \ \wedge \ y \in^? [1,\ldots,100] \ \wedge \ z \in^? [1,\ldots,100] \wedge$$

$$x \geq^? 5 \wedge \ x + y + z \leq^? 10$$

Primera asignación:

$$x =^? 1 \wedge y =^? 1 \wedge z =^? 1 \wedge x \geq^? 5 \ \wedge \ x + y + z \leq^? 10$$

$$\vdots$$

Primera solución:

$$x = 5 \ \wedge \ y = 1 \ \wedge \ z = 1$$

después de $4 \times 100 \times 100$ intentos.

Técnicas de enumeración: *Backtracking*

$$x \in^? [1,\ldots,100] \ \wedge \ y \in^? [1,\ldots,100] \ \wedge \ z \in^? [1,\ldots,100] \wedge$$

$$x \geq^? 5 \wedge \ x+y+z \leq^? 10$$

Subproblema 1: $x =^? 1 \ \wedge \ y \in^? [1,\ldots,100] \ \wedge \ z \in^? [1,\ldots,100] \wedge$

$$x \geq^? 5 \ \wedge \ x+y+z \leq^? 10$$

Subproblema 2: $x \in^? [2,\ldots,100] \ \wedge \ y \in^? [1,\ldots,100] \wedge$

$$z \in^? [1,\ldots,100] \ \wedge \ x \geq^? 5 \ \wedge \ x+y+z \leq^? 10$$

Subproblema 1: Solución imposible

Subproblema 2: Primera solución: $x = 5 \ \wedge \ y = 1 \ \wedge \ z = 1$ después de 4 intentos

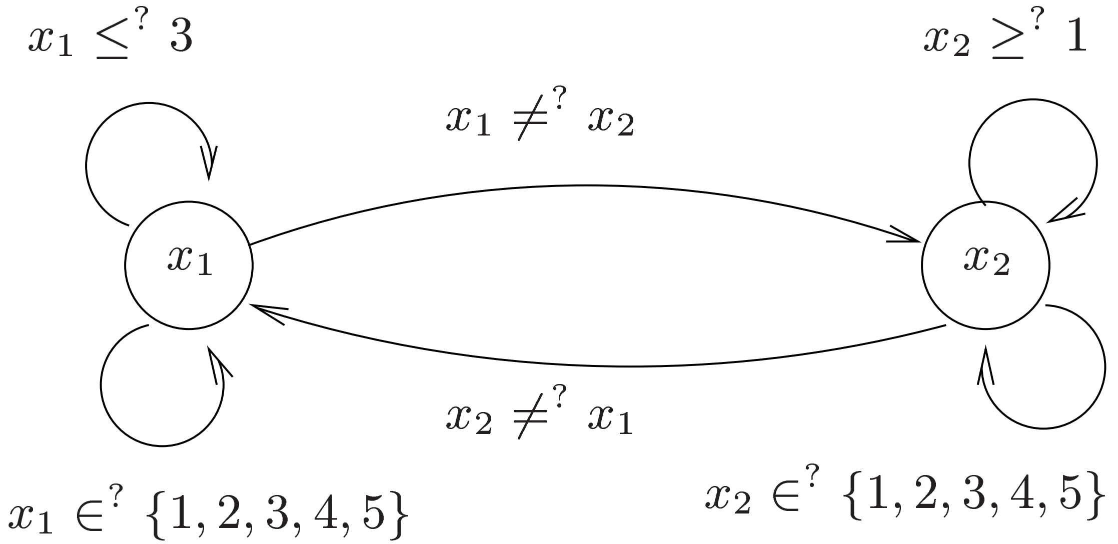
Técnicas de enumeración

- Completas: si existe una solución ésta será encontrada
- Utilización pasiva de las restricciones
- En general, impracticables en problemas grandes

Técnicas de verificación de consistencia local

Consistencia de nodo

$\mathcal{X} = \{x_1, x_2\}$ con $\mathcal{D} = \{1,2,3,4,5\}$ y $\mathcal{C} = \{x_1 \leq^? 3, x_2 \geq^? 1, x_1 \neq^? x_2\}$



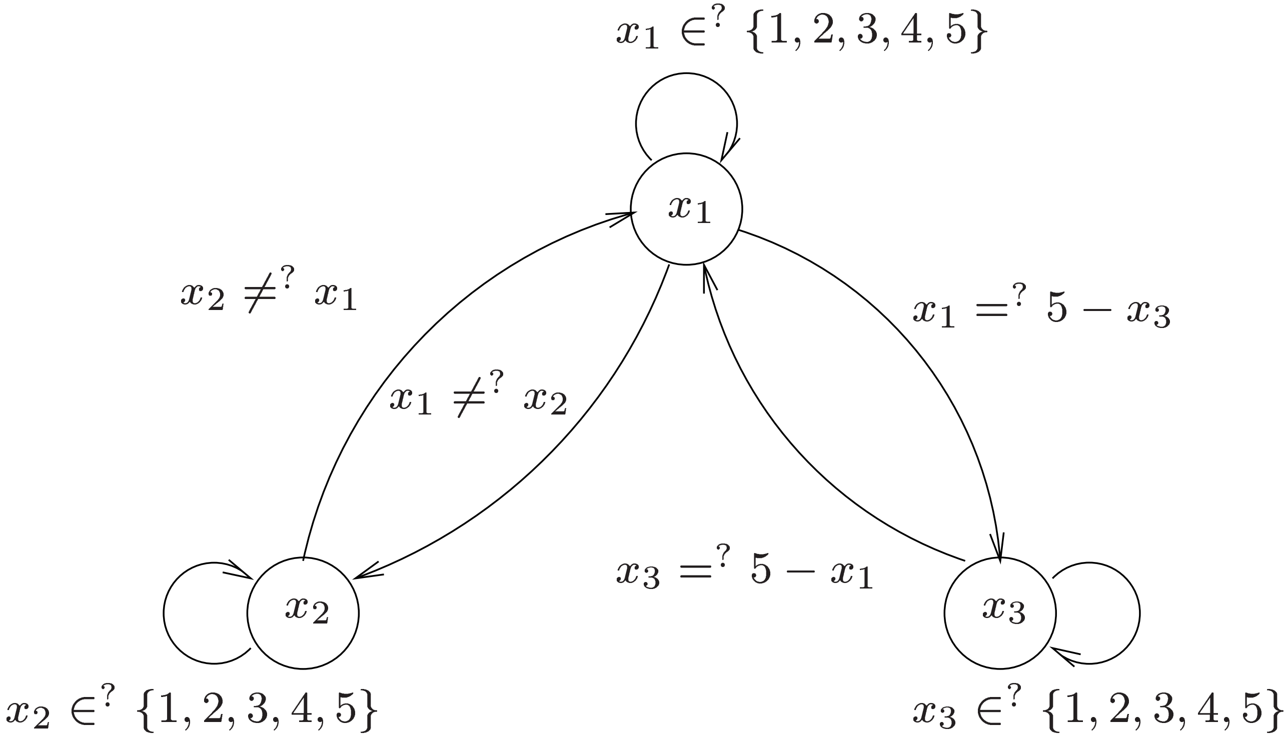
Nodo x_1: inconsistente; Nodo x_2: consistente

Técnicas de verificación de consistencia local

Consistencia de arco

$\mathcal{X} = \{x_1, x_2, x_3\}$ con $\mathcal{D}_{x_1} = \mathcal{D}_{x_2} = \mathcal{D}_{x_3} = \{1,2,3,4,5\}$ y

$\mathcal{C} = \{x_1 \neq^? x_2, x_1 + x_3 =^? 5\}$



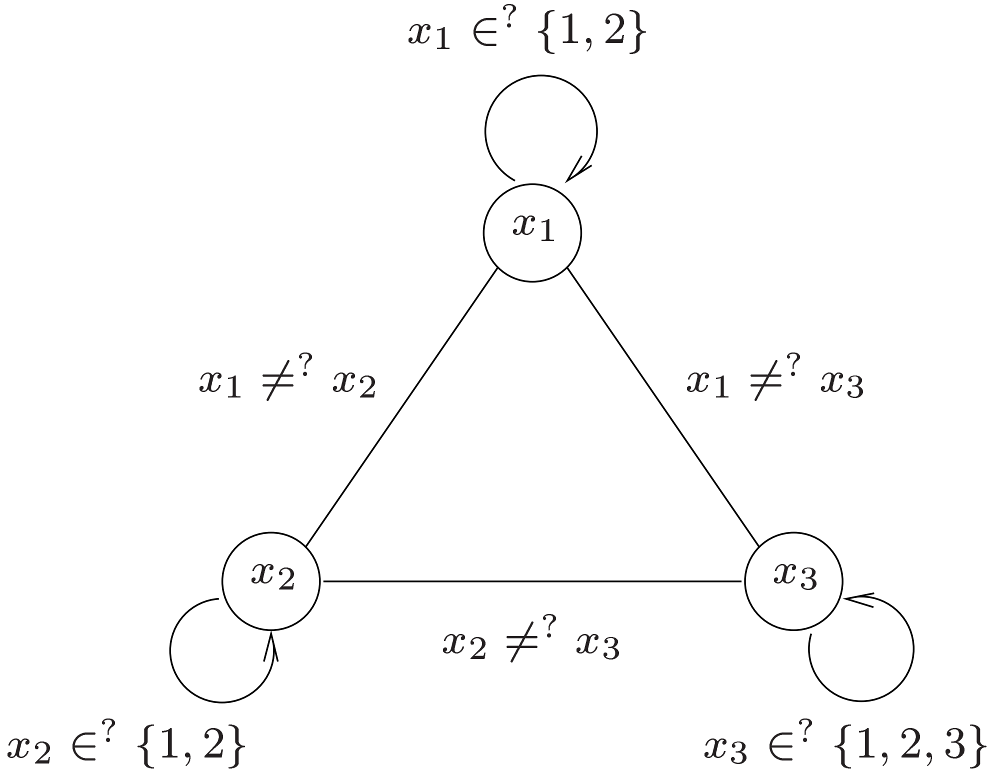
(x_1, x_2), (x_2, x_1): consistentes; (x_1, x_3), (x_3, x_1): inconsistentes

Técnicas de verificación de consistencia local

Consistencia de camino

$$\mathcal{X} = \{x_1, x_2, x_3\} \text{ con } \mathcal{D}_{x_1} = \mathcal{D}_{x_2} = \{1,2\}, \mathcal{D}_{x_3} = \{1,2,3\}$$

$$\text{y } \mathcal{C} = \{x_1 \neq^? x_2, x_1 \neq^? x_3, x_2 \neq^? x_3\}$$



Algoritmos de verificación de consistencia de arcos

Funcion REVISAR$((x_i, x_j))$: **boolean**

Inicio

  RETORNAR $\leftarrow$ **F** ;

  Para cada $\alpha \in Sol(x_i \in^? D_{x_i})$ **hacer**

    Si $Sol(x_j \in^? D_{x_j} \wedge c^?(\alpha(x_i), x_j)) = \emptyset$ **entonces**

      $D_{x_i} \leftarrow D_{x_i} \backslash \alpha(x_i)$;

      RETORNAR $\leftarrow$ **V**;

    fin_si

  fin_hacer

Fin

Basada en Fikes, 1970

Algoritmos de verificación de consistencia de arcos

Procedimiento AC-1;
Inicio
  $Q \leftarrow \{(x_i, x_j) \mid (x_i, x_j) \in arcs(G), x_i \neq x_j\}$;
  Repetir
    $cambio \leftarrow$ **F**;
    Para cada $(x_i, x_j) \in Q$ **hacer**
      $cambio \leftarrow cambio \ \vee$ REVISAR$((x_i, x_j))$;
    fin_hacer
  Hasta $\neg cambio$
Fin

Algoritmos de verificación de consistencia de arcos

- AC-1, $O(a^3ne)$, Mackworth, 1977
- AC-2, $O(a^3ne)$, Mackworth, 1977, basado en Waltz, 1975
- AC-3, $O(a^3n)$, Mackworth, 1977
- AC-4, $O(a^2n)$, Mohr y Henderson, 1986
- AC-5, Deville y Van Hentenryck, 1991; Perlin, 1992
- AC-6, Bessière, 1994
- AC-6+, Bessière, 1994
- AC-6++, Bessière y Régin, 1994
- AC-7, Freuder, 1994

Propagación de bordes

Considerando:

$$x_1 \in [1, \ldots, 100] \ \wedge \ x_2 \in [1, \ldots, 100] \ \wedge \ x_1 \geq x_2 + 5$$

Analizando los bordes:

- límite inferior $(x_1) \geq$ límite inferior $(x_2 + 5)$

  $\rightarrow \quad x_1 \in [6, \ldots, 100]$

- límite superior $(x_2 + 5) \leq$ límite superior (x_1)

  $\rightarrow \quad x_2 \in [1, \ldots, 95]$

Por lo tanto

$$x_1 \in [6, \ldots, 100] \ \wedge \ x_2 \in [1, \ldots, 95] \ \wedge \ x_1 \geq x_2 + 5$$

Disyunción constructiva

Considerando:

$$x_1 \in [1, \ldots, 10] \ \wedge \ x_2 \in [1, \ldots, 10] \ \wedge \ x_1 - x_2 \geq 8 \ \vee \ x_2 - x_1 \geq 8$$

Analizando cada componente de la disyunción:

- $x_1 - x_2 \geq 8$

  $\rightarrow \quad x_1 \in \{9, 10\} \wedge x_2 \in \{1, 2\}$

- $x_2 - x_1 \geq 8$

  $\rightarrow \quad x_1 \in \{1, 2\} \wedge x_2 \in \{9, 10\}$

Por lo tanto

$$x_1 \in \{1, 2, 9, 10\} \wedge x_2 \in \{1, 2, 9, 10\} \ \wedge \ x_1 - x_2 \geq 8 \ \vee \ x_2 - x_1 \geq 8$$

Disyunción constructiva

- Uso activo de componentes de la disyunción

- No hay elección a priori entre componentes

Técnicas de verificación de consistencia local

- Incompletas: eventualmente prueban no existencia de soluciones
- Uso activo de restricciones
- Aporte fundamental de la comunidad de IA
- Amplio uso de consistencia de arco
- Algoritmos específicos según restricciones y dominios

Técnicas híbridas: *Forward Checking*

Considerando:

$$x \in^? [1, \ldots, 100] \ \wedge \ y \in^? [1, \ldots, 100] \ \wedge \ z \in^? [1, \ldots, 100] \wedge$$

$$x \geq^? 5 \wedge \ x + y + z \leq^? 10$$

Analizando la consistencia local:

- $x \geq 5 \rightarrow$
  - $x \in [5, \ldots, 100]$
- $x + y + z + \leq 10 \rightarrow$
  - $x \in [5, \ldots, 8]$
  - $y \in [1, \ldots, 4]$
  - $z \in [1, \ldots, 4]$

$$x \in^? [5, \ldots, 8] \ \wedge \ y \in^? [1, \ldots, 4] \ \wedge \ z \in^? [1, \ldots, 4] \wedge$$

$$x \geq^? 5 \wedge \ x + y + z \leq^? 10$$

Técnicas híbridas: *Forward Checking*

Enumerando y verificando consistencia:

$$x = 5 \;\wedge\; y \in^? [1, \ldots, 4] \;\wedge\; z \in^? [1, \ldots, 4] \wedge$$

$$x \geq^? 5 \wedge \; x + y + z \leq^? 10$$

Enumerando y verificando consistencia:

$$x = 5 \;\wedge\; y = 1 \;\wedge\; z \in^? [1, \ldots, 4] \wedge$$

$$x \geq^? 5 \wedge \; x + y + z \leq^? 10$$

Enumerando y verificando consistencia:

$$x = 5 \;\wedge\; y = 1 \;\wedge\; z = 1 \wedge$$

$$x \geq^? 5 \wedge \; x + y + z \leq^? 10$$

Técnicas híbridas

- Completas
- Uso activo de restricciones
- Necesidad de balance enumeración/verificación de consistencia
- Complejas heurísticas incluyendo saltos en el árbol de búsqueda:
  - *Full Lookahead*
  - *Intelligent Backtracking*
  - ...

Criterios de selección de variables

- Mínimo dominio
- Variable más restringida

En general, basadas en principio *first fail*:

"intentar primero lo más probable de que falle"

Criterios de selección de valores

- División del dominio
- Borde inferior
- Borde superior

Resolución dinámica de CSPs

- Incorporación/eliminación de restricciones
- Especial para resolución incremental de restricciones
- Utilizada para resolver CSOPs

CSPs sobre restringidos

- CSP original no tiene solución
- Cantidad mínima restricciones violadas v/s calidad mínima de restricciones violadas

CSPs jerárquicos

- Jerarquía de restricciones:

  obligatorias $\rightarrow$ deseables

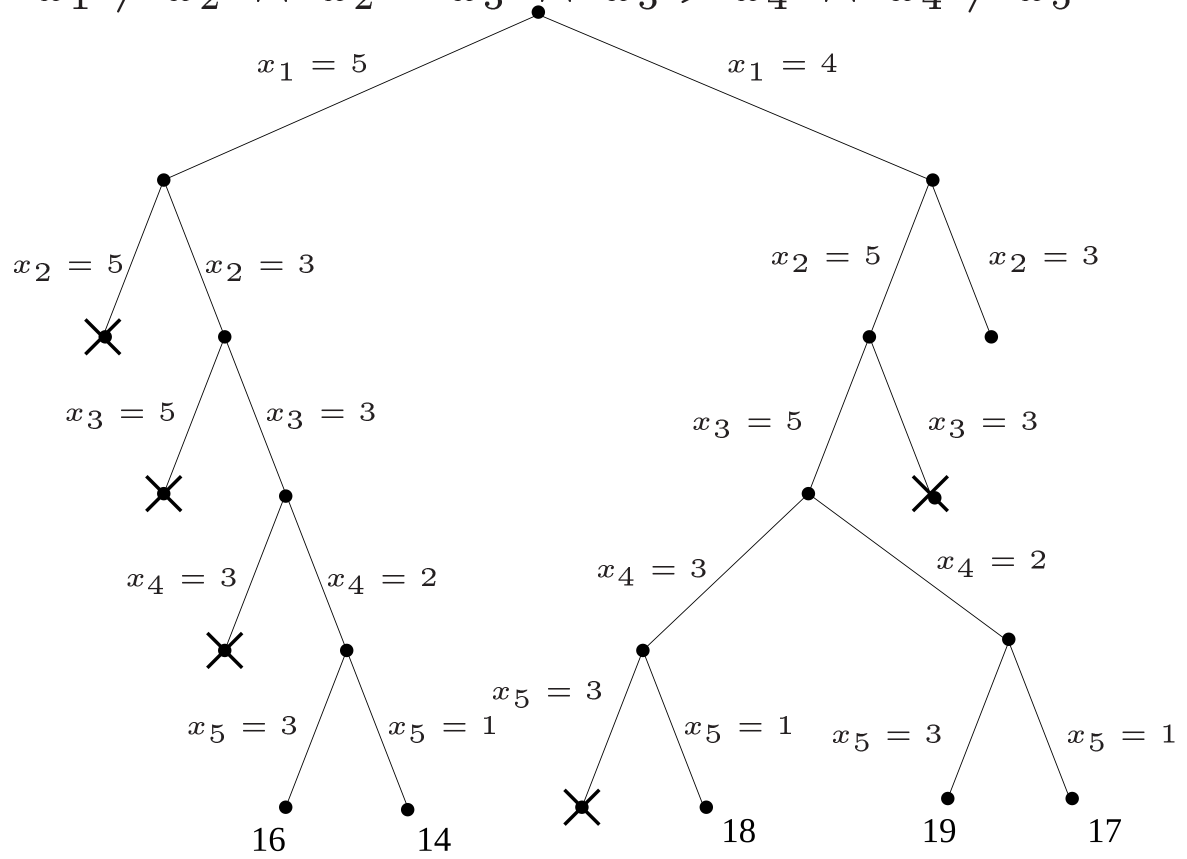
- Utilizadas para resolver problemas sobre restringidos

Resolución de CSOPs: Técnica basada en backtracking

Considerando: $Max \;\; f = x_1 + x_2 + x_3 + x_4 + x_5$

$$x_1 \in \{4,5\}, x_2 \in \{3,5\}, x_3 \in \{3,5\}, x_4 \in \{2,3\}, x_5 \in \{1,3\} \wedge$$

$$x_1 \neq x_2 \;\wedge\; x_2 = x_3 \;\wedge\; x_3 > x_4 \;\wedge\; x_4 \neq x_5$$



Resolución de CSOPs: Ramificación y acotamiento

Enfoque *Satisfiable to Unsatisfiable*

Aplicación repetida de la siguiente regla:

- $\langle P, f, lb, ub\rangle \quad \rightarrow \quad \langle P, f, lb, \alpha(f)\rangle$
  if $\alpha \in Sol(P \wedge f <^{?} ub)$

Resolución de CSOPs: Ramificación y acotamiento

Enfoque *Splitting*

Aplicación repetida de las siguientes reglas:

- $\langle P, f, lb, ub \rangle \;\rightarrow\; \langle P, f, lb, \alpha(f) \rangle$
  if $\alpha \in Sol(P \wedge f <^{?} \frac{lb+ub}{2})$

- $\langle P, f, lb, ub \rangle \;\rightarrow\; \langle P, f, \lceil \frac{lb+ub}{2} \rceil, ub \rangle$
  if $Sol(P \wedge f <^{?} \frac{lb+ub}{2}) = \emptyset$ **and** $lb \neq ub$

Un enfoque deductivo de la resolución de CSPs

Resolvedor de restricciones

=

Reglas de reescritura + Estrategias

Basado en noción de Sistema Computacional

Sistemas computacionales

Reglas de reescritura extendidas en ELAN:

$$[\ell] \quad l \Rightarrow r \qquad [\ \textbf{if} \ \ cond \quad | \quad \textbf{where} \ \ y{:=}(S)u\ ]^*$$

Sistemas computacionales

Estrategias en ELAN:

dk $(\ell_1, \ldots, \ell_n)$	**dk one** $(\ell_1, \ldots, \ell_n)$
dc $(\ell_1, \ldots, \ell_n)$	**dc one** $(\ell_1, \ldots, \ell_n)$
first $(\ell_1, \ldots, \ell_n)$	**first one** $(\ell_1, \ldots, \ell_n)$
";"	**repeat**$*$ (s)

Reglas de simplificación para CSPs

[**ArcConsistency**] $x \in^? D_x \ \wedge \ c^? \ \wedge \ C$

$\Rightarrow$

$x \in^? RD(x \in^? D_x, c^?) \ \wedge \ c^? \ \wedge \ C$

if $RD(x \in^? D_x, c^?) \neq D_x$

[**Falsity**] $x \in^? \emptyset \ \wedge \ C$

$\Rightarrow$

$\mathbf{F}$

Reglas de propagación para CSPs

[Instantiation]

$$x \in^? \{v\} \quad \wedge \quad C$$

$$\Rightarrow$$

$$x =^? v \quad \wedge \quad C$$

[Elimination]

$$x =^? v \quad \wedge \quad C$$

$$\Rightarrow$$

$$x =^? v \quad \wedge \quad C\{x \mapsto v\}$$

$$\textbf{if} \quad x \in \mathcal{V}ar(C)$$

Regla de enumeración para CSPs

[SplitDomain] $x \in^? D_x \quad \wedge \quad C$

$\Rightarrow$

$x \in^? D'_x \quad \wedge \quad C$ **or** $x \in^? D''_x \wedge \quad C$

if $D'_x \ \cup \ D''_x \ = \ D_x$ **and**

$D'_x \ \cap \ D''_x \ = \ \emptyset$ **and**

$D'_x \neq \emptyset$ **and** $D''_x \neq \emptyset$

Estrategias de búsqueda para CSPs

```
[.] GenerateTest =>
    repeat* (dk (SplitDomain)) ;
    repeat* (Elimination)
end}

[.] Backtracking =>
    repeat* (dk (SplitDomain) ;
             first one (Elimination , id))
end
```

Estrategias de verificación de consistencia para CSPs

```
[.] AC-1 =>
    repeat* (ArcConsistency ;
              first one (Falsity , id))
end

[.] LocalConsistency =>
    repeat* (ArcConsistency ;
             first one (Instantiation ;
                         first one (Elimination , id) ,
                         Falsity ,
                         id))
end
```

Estrategia híbrida para CSPs

```
[.] FullLookahead =>
    LocalConsistency ;
    repeat* (dk (SplitDomain) ;
             first one (Elimination , id) ;
             LocalConsistency)
end
```

Enfoque reglas + estrategias

- Pruebas de corrección, completitud y término
- Fácil prototipar heurísticas basadas en *backtracking*
- Difícil simular heurísticas basadas en *backjumping*

Cooperación de *solvers*

- Mejoramiento de eficiencia
- Resolución de problemas no resueltos por un *solver* elemental
- Gran interés actual por lenguajes de coordinación

Estrategias concurrentes en ELAN

dk $(S_1 \mid\mid \ldots \mid\mid S_n)$ **dk one** $(S_1 \mid\mid \ldots \mid\mid S_n)$

dc $(S_1 \mid\mid \ldots \mid\mid S_n)$ **dc one** $(S_1 \mid\mid \ldots \mid\mid S_n)$

Control de procesos externos en ELAN

dkcall (P) **dccall** (P)

Enfoque *Satisfiable to Unsatisfiable* para CSOPs

```
[.] MinSatToUnsat =>
      PostLTEUpperBound ;
      LocalConsistencyForEC ;
      GetNewUpperBound ;
      repeat* (first one (PostLTUpperBound ;
                          LocalConsistencyForEC ;
                          GetNewUpperBound
                          ,
                          SetLowerBoundToUpperBound)
              )
end
```

Enfoque *Splitting* para CSOPs

```
[.] MinSplitting =>
      PostLTEUpperBound ;
      LocalConsistencyForEC ;
      GetNewUpperBound ;
      UnderValueLowerBound ;
      repeat* (first one (SetUpperBoundToMiddle ;
                          PostLTUpperBound ;
                          LocalConsistencyForEC ;
                          GetNewUpperBound ;
                          UnderValueLowerBound
                          ,
                          SetLowerBoundToMiddle)
              )
end
```

Cooperación de *solvers* de restricciones

Enfoque *Satisfiable to Unsatisfiable*

```
[.] MinSatToUnsat =>
      SetUpperAndLowerBounds ; repeat* (strat1)
end

[.] strat1 =>
      first one (PostLTUpperBound ;
                 LocalConsistencyForEC ;
                 GetNewUpperBound
                 ,
                 SetLowerBoundToUpperBound)
end
```

Cooperación de *solvers* de restricciones

Enfoque *Splitting*

```
[.] MinSplitting =>
      SetUpperAndLowerBounds ; repeat* (strat2)
end

[.] strat2 =>
      first one (SetUpperBoundToMiddle ;
                 PostLTUpperBound ;
                 LocalConsistencyForEC ;
                 GetNewUpperBound ;
                 UnderValueLowerBound
                 ,
                 SetLowerBoundToMiddle)
end
```

Cooperación de *solvers* de restricciones

Solvers secuenciales

```
[.] CooperationI =>
SetUpperAndLowerBounds ;
repeat* (strat1 ;
         first one (strat2 , id)
        )
end

[.] CooperationII =>
SetUpperAndLowerBounds ;
repeat* (dccall (solver1CSOP) ;
         first one (dccall (solver2CSOP) , id)
        )
end
```

Resultados: S1 ; S2 ; S1 ; S2 ; S1 ; S2 ; ...

Cooperación de *solvers* de restricciones

Solvers concurrentes

```
[.] CooperationIII =>
SetUpperAndLowerBounds ;
repeat* (dc one (strat1 || strat2))
end

[.] CooperationIV =>
SetUpperAndLowerBounds ;
repeat* (dc one (dccall (solver1CSOP)
                 ||
                 dccall (solver2CSOP))
        )
end
```

Resultados: S2 ; S2 ; S1 ; S2 ; S1 ; S1 ; ...

Enfoque reglas + estrategias

- Flexibilidad para prototipar cooperación de *solvers* de manera abstracta
- Posibilidad de evitar reglas/estrategias que no terminan mediante el uso de estrategias concurrentes
- Extensión del enfoque para realizar colaboración de *solvers*

Programación con Restricciones

Aporta soluciones en problemas de búsqueda restringida:

- Administración de proyectos
- Gestión de la producción
- Distribución y logística
- Análisis financiero

En general, problemas de asignación de recursos

Problemas de búsqueda restringida

- No existen métodos generales (NP-completos)
  $\rightarrow$ probar diferentes estrategias y heurísticas

- Necesidades cambian frecuentemente
  $\rightarrow$ programas flexibles para adaptarse a los cambios

Problemas de búsqueda restringida

En particular,

- Restricciones son de tipos diferentes:
  - Simbólicas/numéricas
  - Temporal/espacial/recursos
- Contexto dinámico
  - Configuración cambia
  - Requerimientos cambian

Problemas de búsqueda restringida

- Obtención de soluciones requiere:
  - Desarrollo incremental
  - Ajuste de heurísticas
  - Uso interactivo
  - Respuestas rápidas
  - Integrable a sistemas de información globales

Lenguajes convencionales

Problemas:

- Programación requiere mucho tiempo
- Programas caros de mantener
- Programas difíciles de modificar y extender

Necesidades:

- Programación rápida
- Programas fáciles de modificar
- Programas fáciles de adaptar rápidamente a las necesidades

Todo esto sin sacrificar la eficiencia

Administración de la producción

- Conceptos principales:
  - Tareas
  - Recursos
  - Tiempo
  - Costo
- Problemas principales:
  - Planificación
  - Programación
  - Asignación

Administración de la producción

- Planificación

  ¿Qué producir y en qué cantidad?

- Programación

  ¿Cuándo producir?

- Asignación

  ¿Cómo producir?

El problema de transporte

	Destino			
Origen	1	2	3	Oferta
1	5	1	1	4
2	2	6	9	6
Demanda	3	5	2	

```
top:-
X11 + X21 >= 3,
X12 + X22 >= 5,
X13 + X23 >= 2,
X11 + X12 + X13 <= 4,
X21 + X22 + X23 <= 6,
costo = 5*X11 + X12 + X13 + 2*X21 + 6*X22 + 9*X23,
rmin(costo).
```

Restricciones globales

- Introducidas en CHIP V4 (1992/1993)
- Mayor expresividad
  - Generalización de restricciones existentes
  - Más cercanas a los problemas
  - Amplio uso
- Mayor eficiencia
  - Consideran interacción entre restricciones elementales
  - Mejor disminución del espacio de búsqueda
  - Menor utilización de memoria

Restricciones globales

- Alldifferent
  - Problemas de localización multidimensional
- Atmost
  - Problemas de conteo y secuencia de valores
- Cumulative
  - Problemas de scheduling con capacidad finita
- Cycle
  - Problemas de circuito en grafos

$$\boxed{SEND + MORE = MONEY}$$

- Resolver la suma
- Diferentes letras representan diferentes valores $[0, \ldots, 9]$

SEND + MORE = MONEY: Modelo 1

$$\begin{array}{cccccc} & & S & E & N & D \\ + & & M & O & R & E \\ \hline & M & O & N & E & Y \end{array}$$

```
sendmore([S,E,N,D],[M,O,R,E],[M,O,N,E,Y]):-
[S,E,N,D,M,O,R,Y] :: 0..9,
alldifferent([S,E,N,D,M,O,R,Y]),
S <> 0, M <> 0,
Send = 1000*S + 100*E + 10*N + D,
More = 1000*M + 100*O + 10*R + E,
Money = 10000*M + 1000*O + 100*N + 10*E + Y,
Send + More #= Money,
labeling([S,E,N,D,M,O,R,Y]).
```

SEND + MORE = MONEY: Modelo 2

$$
\begin{array}{cccccc}
 & R_4 & R_3 & R_2 & R_1 & \\
 & & S & E & N & D \\
+ & & M & O & R & E \\
\hline
 & M & O & N & E & Y
\end{array}
$$

```
sendmore([S,E,N,D],[M,O,R,E],[M,O,N,E,Y]):-
[S,E,N,D,M,O,R,Y] :: 0..9, [R1,R2,R3,R4] :: 0..1,
alldifferent([S,E,N,D,M,O,R,Y]),
S <> 0, M <> 0, R1 = M,
R2 + S + M = O + 10*R1,
R3 + E + O = N + 10*R2,
R4 + N + R = E + 10*R3,
D + E = Y + 10*R4,
labeling([S,E,N,D,M,O,R,Y]).
```

Propagación

```
1.- S <> 0 : S en [1..9]
2.- M <> 0 : M en [1..9]
3.- R1 = M : [0..1] & [1..9] -> R1 = 1, M = 1
           : S en [2..9], E en [0,2..9], N en [0,2..9],
             D en [0,2..9], O en [0,2..9],
             R en [0,2..9], Y en [0,2..9]
4.- R2 + S + M = O + 10*R1 -> R2 + S = O + 9,
    R2 + S <= 10 -> O + 9 <= 10 -> O = 0
    R2 + S = 9 -> 8 <= S <= 9
    [E,N,D,O,R,Y] # 0
5.- R3 + E + O = N + 10*R2 -> R3 + E = N + 10*R2
    R3 + E <= 10 -> N + 10*R2 <= 10
    2 <= N -> R2 = 0 -> S = 9
```

Coloreo de grafos: Programación de un congreso

- 11 sesiones de medio día, en 3 salas y en 2 días.
- Sesión E antes de J, sesiones D y F antes de K.
- Conjuntos de sesiones en paralelo:

  AJ, JI, IE, CF, FG, DH, BD, KE,

  BIHG, AGE, BHK, ABCH, DFJ.

Coloreo de grafos: Programación de un congreso

```
top:-
domain([A,B,C,D,E,F,G,H,I,J,K],1..4),
alldifferent([A,J]), alldifferent([J,I]),
alldifferent([I,E]), alldifferent([E,C]),
alldifferent([C,F]), alldifferent([F,G]),
alldifferent([D,H]), alldifferent([B,D]),
alldifferent([K,E]), alldifferent([B,I,H,G]),
alldifferent([A,G,E]), alldifferent([B,H,K]),
alldifferent([A,B,C,H]), alldifferent([D,F,J]),
J > E, K > D, K > F,
atmost(3,[A,B,C,D,E,F,G,H,I,J,K],1),
atmost(3,[A,B,C,D,E,F,G,H,I,J,K],2),
atmost(3,[A,B,C,D,E,F,G,H,I,J,K],3),
atmost(3,[A,B,C,D,E,F,G,H,I,J,K],4),
labeling(a[A,B,C,D,E,F,G,H,I,J,K]).
```

Job Shop

```
bridge(L,end) :-
define_domain(L,end),
stating_prec_const(L),
stating_dist_const(L),
stating_disj_const(L).

prec([X,Dx],[Y,Dy]):-
Y >= X + Dx.

disj([X,Dx],[Y,Dy]) :-
Y >= X + Dx.
disj([X,Dx],[Y,Dy]) :-
X >= Y + Dy.

?- minimize(bridge(L,end),end).
```

Herramientas industriales

Nombre	Compañía	Tecnología	Lenguaje
Prolog IV	PrologIA	R, B	Prolog
Charme	Bull	FD	C
Decision Power	ICL	FD, R, B	Prolog
SNI-Prolog	Siemens-Nixdorf	FD, R, B	Prolog
CHIP V4	Cosytec	FD, R, B	Prolog
Ilog Solver	Ilog	FD	C++
Chleo	Axia	FD	C

Modelamiento

- Etapa crucial para:
  - Descripción concisa (# variables y # restricciones)
  - Eficiencia de la resolución
- Elecciones diferentes:
  - Dominio de cálculo (discreto/continuo, problemas 0-1)
  - Variables y valores (frecuentemente intercambiables)
  - Restricciones (eficiencia del tratamiento)
  - Estrategias de búsqueda (enumeración de valores/alternativas)

Modelamiento

- Exige un buen conocimiento de:
  - El problema
  - Métodos de resolución
  - *Solvers* disponibles

Programación con Restricciones

- No es tan fácil:

  Gran diferencia entre un prototipo y un problema real

- Necesidad de personal calificado:

  Modelamiento del problema es fundamental

- Necesidad de soluciones integradas:

  El problema específico es importante pero sólo una pequeña parte del software a desarrollar