Problemi sovravincolati

- Alcuni problemi nella vita reale non hanno soluzione: sono sovravincolati
- CLP in questo caso dà come risposta semplicemente 'no'
- Questo in applicazioni reali non è sufficiente
- Come gestire questo problema?

Vincoli soft

- Uno dei metodi è quello di usare vincoli soft (che possono essere rilassati)
  - cerchiamo di massimizzare i vincoli soft che sono soddisfatti
- I vincoli reificati possono essere usati come vincoli soft: massimizzo il numero di vincoli soddisfatti
- Es. Orario delle lezioni:
  - cerco di soddisfare le richieste degli insegnanti

    `Mat #> 10, Fisica #< 14, …`
  - Con vincoli Soft:

```
Mat #> 10 #<=>B1, Fisica#< 14 #<=> B2,
sumlist([B1,B2,…],Sum),
F #= -Sum,
minimize(labeling(…),F).
```


Ancora sui vincoli reificati

- I vincoli reificati possono servire per collegare due o più vincoli:
  - `X#>Y #<=> B1, X#>Z #<=> B2, X+Y#>0`
    - Ci sono altri connettori `#\/, #/\, #=>`, …
    - Si può scrivere anche `X#>Y #\/ X#>Z`
- Poi posso rilassarli insieme:
  - `Elettr#<10 #\/ Elettr#>12 #<=> B1, sumlist([B1, …],Sum), F #= -Sum, minimize(labeling(…),Sum).`

CLP(R) in ECLiPSe

- ECLiPSe ha una libreria per l'uso di CLP(R), chiamata **`eplex`**
- Utilizzo di un risolutore esterno (Cplex, Xpress-MP oppure solver free), basato sull'algoritmo del simplesso
- Creazione di un'istanza del solver, in cui si stabilisce una funzione obiettivo

```
eplex:eplex_solver_setup(min(X))
eplex:eplex_solver_setup(max(X))
```

- Definizione di domini:

  `$::` *oppure* `eplex:(L :: Dom)`
- Vincoli lineari:

  `$>=, $=<, $=`
- risoluzione

  `eplex_solve(Cost)`

Esempio

```
:- lib(eplex).
lp_example(X,Y,Cost) :-
  eplex_solver_setup(min(X)),
  [X,Y] $:: 0..10,
     X+Y $>= 3,
     X-Y $= 0,
  eplex_solve(Cost).
```

- *Risultato:*

```
:- lp_example(X,Y,C)
X = X{0 .. 10 @ 1.5}
Y = Y{0 .. 10 @ 1.5}
C = 1.5
```

Soluzione ottima

Valore della soluzione ottima

Integrazione dei due solver

- *In molte applicazioni, fare cooperare i due solver porta a soluzioni più velocemente di ciascuno dei due*
- *Nell'esempio precedente, la propagazione Arc-Consistency non restringe i domini:*

```
[X,Y] #:: 0..10,X+Y #>= 3,X-Y #= 0.
X = X{[0..10]}
Y = X{[0..10]}
Delayed goals:  X{[0..10]} + X#>=3
```

- *eppure il valore ottimo del rilassamento lineare è X=1.5 (quindi non ci sono soluzioni con X=0, X=1)*
- *Posso calcolare il valore ottimo del rilassamento lineare e imporre X #>= C.*
- *E` anche possibile creare un vincolo che esegue il simplesso*

CONSTRAINT PROGRAMMING TOOLS

	Declarative	Object-Oriented
Open source	ECLiPSe, SWI Prolog, mozart, Gnu Prolog	Gecode, CHOCO, SCIP, JaCoP
Commercial	CHIP, BProlog, SICStus	ILOG An IBM Company, Comet, dynadec

… and many more

Scheduling Scientific Experiments on the Rosetta/Philae Mission

Gilles Simonin, Christian Artigues, Emmanuel Hebrard, and Pierre Lopez

CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
Univ de Toulouse, LAAS, F-31400 Toulouse, France
{gsimonin,artigues,hebrard,lopez}@laas.fr

Abstract. The Rosetta/Philae mission was launched in 2004 by the European Space Agency (ESA). It is scheduled to reach the comet 67P/Churyumov-Gerasimenko in 2014 after traveling more than six billion kilometers. The Philae module will then be separated from the orbiter (Rosetta) to attempt the first ever landing on the surface of a comet. If it succeeds, it will engage a sequence of scientific exploratory experiments on the comet.

In this paper we describe a constraint programming model for scheduling the different experiments of the mission. A feasible plan must satisfy a number of constraints induced by energetic resources, precedence relations on activities, or incompatibility between instruments. Moreover, a very important aspect is related to the transfer (to the orbiter then to the Earth) of all the data produced by the instruments. The capacity of inboard memories and the limitation of transfers within visibility windows between lander and orbiter, make the transfer policy implemented on the lander's CPU prone to data loss. We introduce a global constraint to handle data transfers. The goal of this constraint is to ensure that data-producing activities are scheduled in such a way that no data is lost.

Thanks to this constraint and to the filtering rules we propose, mission control is now able to compute feasible plans in a few seconds for scenarios where minutes were previously often required. Moreover, in many cases, data transfers are now much more accurately simulated, thus increasing the reliability of the plans.

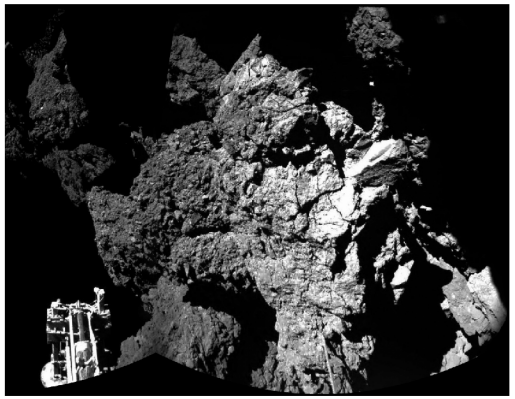
1 Introduction

The international Rosetta/Philae project is an European consortium mission approved in 1993, and is under the leadership of the German Aerospace Research Institute (DLR) and ESA[1]. The spacecraft was launched in 2004 by Ariane 5, and is set to travel more than six billion kilometers to finally reach and land on the comet 67P/Churyumov-Gerasimenko in 2014 in order to analyze the comet structure. It will follow a complex trajectory which includes four gravity assist maneuvers (3 x Earth, 1 x Mars) before finally reaching the comet and enter its orbit. Then, the lander Philae will be deployed and will land on the surface of the comet. Philae features ten instruments, each developed by a European laboratory, to accomplish a given scientific experiment when approaching, or once landed on the comet. For instance, *CIVA* and *ROLIS* are two imaging instruments, used to take panoramic pictures of the comet and microscopic images.

[1] European Space Agency.

M. Milano (Ed.): CP 2012, LNCS 7514, pp. 23–37, 2012.

Rosetta/Philae



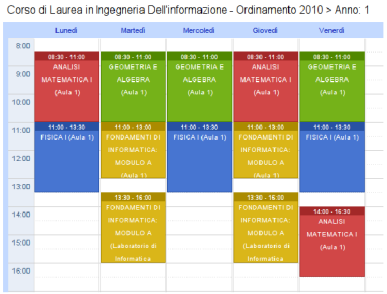
Credits: ESA (European Space Agency)

/home/marco/marco/dida/ia/AppliIA/Lucidi/2014-15/Philae's mission at comet 67P-YoqokoZkviM.mp4

TIMETABLING: problem definition

- Timetabling concerns the definitions of agenda (similar to scheduling)

- Constraints
  - temporal restrictions
    - ordering among activities
    - due dates - release dates
  - resource capacity
    - discrete resources
- Optimization Criteria
  - cost/preferences
  - resource balance



Timetabling

- *Si devono assegnare ad ogni corso un'aula ed un orario*
- *Ho aule di diversa capacità, con servizi (videoproiettore fisso, laboratori, …)*
- *Ogni corso ha un docente e viene seguito da diversi gruppi di studenti.*
- *Non si devono sovrapporre i corsi di uno stesso docente, seguiti dagli stessi studenti o che si trovano nella stessa aula*
- *Per ogni corso so quante ore fa alla settimana*
- *I docenti possono esprimere preferenze sugli orari (nel limite del possibile). Alta priorità ai docenti a contratto*

Specifiche

- *Corsi*
  - *corso(sistemioperativi,stefanelli,130,[[info,2],[info,4],[ele,3],[tlc,3],[auto,4],[ele,5]],7,_,3,2,3,"Sistemi Operativi","http://www.ing.unife.it/informatica/SO-2/").*
  - *corso(strument_misure_elettr,corticelli,180,[[info,2],[auto,2],[tlc,2],[ele,2]],3,_,1,2,3,"Strumentazione e misure elettroniche",_).*
  - *corso(strument_misure_elettr_lab,corticelli,120,[[info,2],[auto,2],[tlc,2],[ele,2]],4,ele,1,4,4,"Strumentazione e misure elettroniche",_).*
  - *corso(inglese_turno1,inlingua,50,[[info,1],[auto,1]],4,_,2,1,2,"Inglese",_).*
  - *corso(inglese_turno2,inlingua,50,[[tlc,1],[ele,1]], 4,_,2,1,2,"Inglese",_).*
- *Aule*
  - *aula(1,250,n,_,s,s,_,_).*
  - *aula(lab_info,64, s,info,n,s, "http://www.ing.unife.it/sidi/cs_lab/cs_lab.htm","Laboratorio di Informatica Grande").*

Domini delle variabili Start

- *Attività fittizie*
  - *pausa pranzo*
  - *fine giornata*

		Lun	Mar	Mer	Gio	Ven
08:30	09:30	1	13	25	37	49
09:30	10:30	2	14	26	38	50
10:30	11:30	3	15	27	39	51
11:30	12:30	4	16	28	40	52
12:30	13:30	5	17	29	41	53
13:30	14:00	6	18	30	42	54
14:00	15:00	7	19	31	43	55
15:00	16:00	8	20	32	44	56
16:00	17:00	9	21	33	45	57
17:00	18:00	10	22	34	46	58
18:00	19:00	11	23	35	47	59
		12	24	36	48	60

Specifiche

	Lunedì	Martedì	Mercoledì	Giovedì	Venerdì
8.30 – 9.30	A	B	A	A	D
9.30 – 10.30	A	B	A	A	D
10.30 – 11.30	A	B	C	B	G
11.30 – 12.30	B	C	C	B	G
12.30 – 13.30	B	C	C	D	F
14 – 15	C	F	G	D	F
15 – 16	C	F	G	G	E
16 - 17	D	F	G	G	E
17 - 18	D	E	E	F	E
18 - 19	D	E	E	F	

- *Normalmente, i corsi devono stare in un "turno"*
- *In questo modo o due corsi si sovrappongono totalmente o non si sovrappongono*
- *Alcuni corsi sono in comunanza con altri CdL (Informatica a scienze, meccanica)*

Implementazione vincolo turni

- *Propia:*

```
turni_def(Turno,Ora1,
   O2,O3) infers fd.

turni_def(a,25,37,1).
turni_def(b,4,39,13).
turni_def(c,7,16,27).
turni_def(d,41,49,9).
turni_def(e,22,34,56)
turni_def(f,46,53,19)
turni_def(g,44,51,31)
```

		Lun	Mar	Mer	Gio	Ven
08:30	09:30	1	13	25	37	49
09:30	10:30	2	14	26	38	50
10:30	11:30	3	15	27	39	51
11:30	12:30	4	16	28	40	52
12:30	13:30	5	17	29	41	53
13:30	14:00	6	18	30	42	54
14:00	15:00	7	19	31	43	55
15:00	16:00	8	20	32	44	56
16:00	17:00	9	21	33	45	57
17:00	18:00	10	22	34	46	58
18:00	19:00	11	23	35	47	59
		12	24	36	48	60

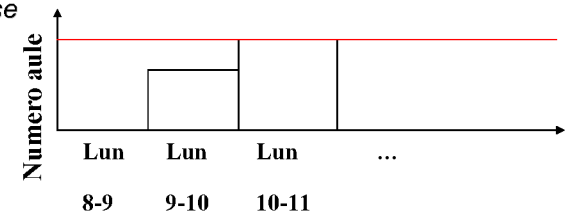
Timetabling

```
solve_ttturni(LSlot,N,LCosts):-
   crea_slot_turni(LSlot),
   imponi_vincoli(LSlot),
   break_symmetries(LSlot),
   obj_function(LSlot,N,LCosts),    N #< 10000,
   min_max(
    (    time_assignment(LSlot),
   aula_assignment(LSlot),
    save_solution(LSlot,N), print_solution(LSlot)
    )
   ,N).
```


Imposizione vincoli

```
imponi_vincoli(LSlot):-
   no_overlap_docente(LSlot),
   no_overlap_studente(LSlot),
   impose_cumulative(LSlot),
   other_constraints_courses(LSlot).
```

- *impose_cumulative impone il vincolo cumulativo in cui le aule sono risorse*



Vincoli sulle aule

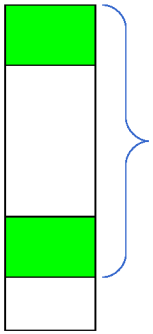
- Per tutti i possibili sottoinsiemi di aule
  - Seleziona i corsi che possono stare **solo** in quel sottoinsieme di aule
  - Ciascun corso consuma 1 aula (1 risorsa)
  - Imponi il vincolo cumulativo su quelle risorse
- Es
  - Info1 => 130 stud — Aula 1 => 250 posti
  - Analisi1 => 160 stud — Aula 5 => 157 posti
  - Fisica 2 => 100 stud — Aula 7 => 120 posti
  - Reti => 100 stud
- Imponiamo:
  - Aula 1 => `cumulative([Analisi1],…,1).`
  - Aula 5 => `cumulative([],…,1).`
  - Aula 7 => `cumulative([],…,1).`
  - Aule 1,5 => `cumulative([Info1,Analisi1],…,2).`
  - Aule 1,5,7 => `cumulative([Info1,Analisi1,Fisica2,Reti], …,3).`
  - Aule 1,7 => `[]` …

Euristiche

- Ho vincolo di non sovrapposizione sulle aule, sugli studenti, sui docenti. Se decido che un gruppo di studenti stanno tutti nella stessa aula, soddisfacendo il no_overlap sull'aula, soddisfo automaticamente il no_overlap sugli studenti (può essere visto come least constraining principle)
- Quindi seleziono il gruppo di studenti che hanno più corsi ed assegno loro un'aula. Tolgo dal dominio degli altri corsi l'aula in questione.
- Alcune aule sono uguali dal punto di vista del modello. Questo introduce simmetrie: il labeling evita di assegnare l'aula simmetrica.
- Problema: io voglio imporre il vincolo cumulativo sui corsi assegnati ad una stessa aula. Però posso imporlo solo dopo che ho assegnato i corsi alle aule!
- Esistono dimostrazioni formali che ripartire daccapo è utile se si usa una selezione casuale (min_max va d'accordo con indomain_random)

Funzione obiettivo

- Minimizzare il numero di "buchi" di orario per gli studenti
- Allo stesso tempo, bisogna evitare che gli studenti abbiano giornate troppo "piene"
- Un giorno libero deve essere considerato positivo!
- Contano di più i gruppi di studenti che sono più numerosi
- Gli studenti che hanno molte scelte non vengono considerati



Differenza fra la fine dell'ultima ora e l'inizio della prima

•Aggiungo un valore negativo per ogni giorno libero

•Minimizzo la somma pesata (considerando il numero di studenti)

Risultati

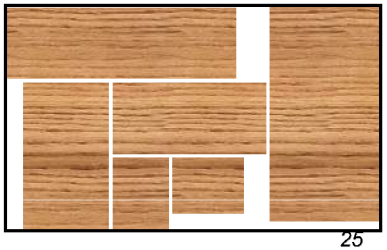
- Minimizzazione delle sovrapposizioni:
  - AA 2003/04: 9 sovrapposizioni fra corsi obbligatori e facoltativi (dato calcolato su 2 trimestri)
  - AA 2004/05: 0 sovrapposizioni (su 3 trim)
  - AA 2003/04: 20 sovrapposizioni per recupero (su 2 trim)
  - AA 2004/05: 0 sovrapposizioni per recupero (su 3 trim)

CUTTING & PACKING: problem definition

- Packing concerns the placement of a number of squares (of different sizes) in one or more larger boxes in such a way that squares do not overlap and minimizing the empty space

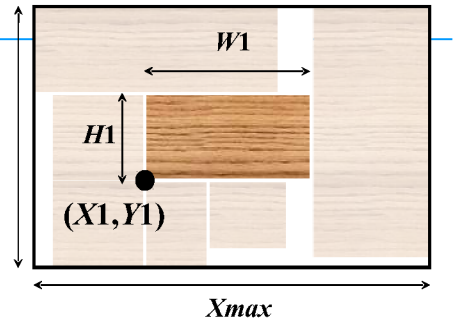
- Cutting is the problem of finding cuts of a master piece in order to obtain a given number of pieces with fixed dimensions, minimizing residues

- Many variants:
  - strip packing
  - guillottine cuts
  - rotations allowed
  - 1 dimension - 2 dimensions - 3 dimensions - 4 dimensions



Packing

- *For each rectangle, 2 variables:*
  - *X position*
  - *Y position*
- *Domains:*
  - *X variables: from 0 to Xmax-W*
  - *Y variables: from 0 to Ymax-H*
- *Constraints:*
  - *Given 2 rectangles (X1,Y2) (X2,Y2) then:*
    - *either Rectangle 1 is left of Rectangle 2: X1 + W1 ≤ X2*
    - *or Rectangle 1 is right of Rectangle 2: X2+W2 ≤ X1*
    - *or Rectangle 1 is under Rectangle 2: Y1+H1 ≤ Y2*
    - *or Rectangle 1 is over Rectangle 2: Y2+H2 ≤ Y1*



2-D PACKING: CP model

- Constraints:
  - non overlapping constraints: given two squares whose coordinates are (`X1`, `Y1`) and (`X2`, `Y2`) and dimensions `D1`, `H1` and `D2`, `H2` respectively

- `X1+W1 ≤ X2 OR Y1+H1 ≤ Y2 OR X2+W2 ≤ X1 OR Y2+H2 ≤ Y1`

- Very hard form of disjunction: no propagation even after instantiation

- Redundant constraints can help:
- `cumulative(Xcoordinates,XDimension,Ydimension,H)`
  `cumulative(Ycoordinates,YDimension,Xdimension,D)`

PACKING: code

```
packing(Data,Xcoords,Ycoords,D,H):-
        create_variables(Data,Xcoords,Ycoords,D,H),
        state_disjunctive(Data,Xcoords,Ycoords),
        state_cumulatives(Data,Xcoords,Ycoords,D,H),

        create_objective(Xcoords,Ycoords,D,H,Z),
        minimize(label_squares(Xcoords,Ycoords),Z).
```

- `create_objective`: creates a variable representing the spare space (or the number of bins if more than one is present)

- `label_squares` selects bigger squares first and assigns the coordinates in order to minimize spare space.

MODEL BASED VISION
VISUAL SEARCH: definition

- Visual Search in model based vision concerns the problem of recognizing an object in a scene given its model

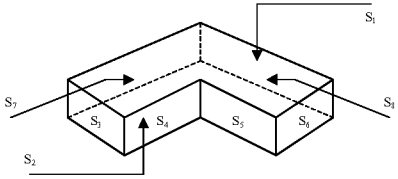
- How to describe the model
- How to perform the mapping

} *Both problems can be solved with constraints*

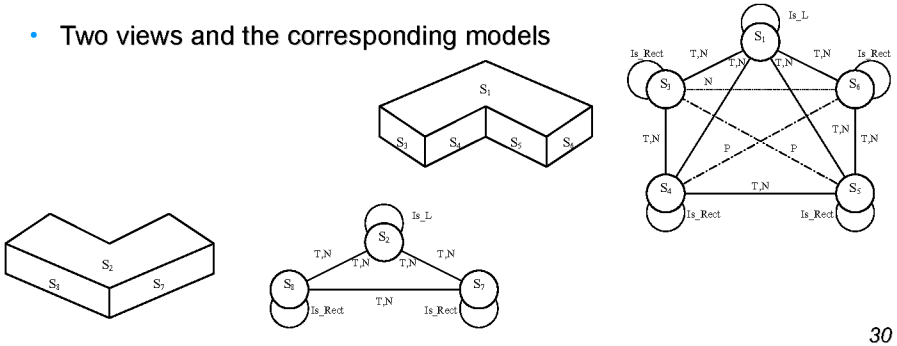
- MODEL: Constraint graph
  - Nodes: object parts
  - Arcs (constraints): their relations
- RECOGNITION: Constraint satisfaction

OBJECT RECOGNITION

- 3-D OBJECT



- Two views and the corresponding models



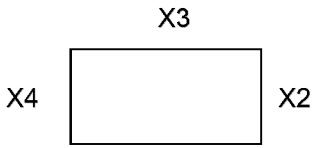
CONSTRAINT-BASED OBJECT RECOGNITION

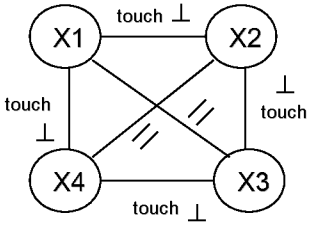
- In order to recognize an object, a low level vision system should extract from the image some visual features (surfaces/edges)
- Constraint satisfaction techniques can be applied in order to recognize the object in the scene.
- The object is recognized if the extracted features satisfy the constraints contained in the model.
- Constraints allow to reduce the search space to be explored.

EXAMPLE

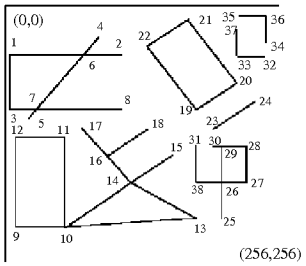
- Rectangle shape

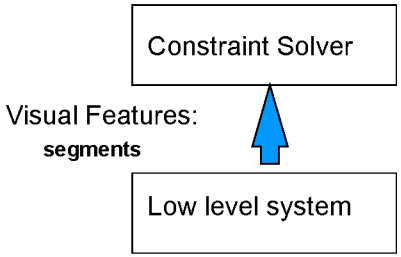
Four straight edges (variables), parallel two by two, which mutually touch themselves with a 90 degree angle ...





EXAMPLE (2)





rectangle CP model

```
touch(X1,X2), touch(X2,X3), touch(X3,X4), touch(X1,X4),
perpend(X1,X2), perpend(X2,X3), perpend(X3,X4), perpend(X1,X4),
same_len(X2,X4), same_len(X1,X3), parallel(X2,X4), parallel(X1,X3)
```


CP model: USER DEFINED CONSTRAINTS

rectangle CP model

```
recognize([X1,X2,X3,X4]):-
    X1,X2,X3,X4::[s1,s2,…,sn],
    touch(X1,X2), touch(X2,X3), touch(X3,X4), touch(X1,X4),
    perpend(X1,X2), perpend(X2,X3), perpend(X3,X4), perpend(X1,X4),
    same_len(X2,X4), same_len(X1,X3),
    parallel(X2,X4), parallel(X1,X3),
    labeling([X1,X2,X3,X4]).
```

- Give the declarative and operational semantics of the constraints: segments are described as facts: `segment(name,X1,Y1,X2,Y2)`
- In all CP languages there are tools that allow new constraints to be defined.
- An example in the CLP(FD) library of ECL^{i}PS^e

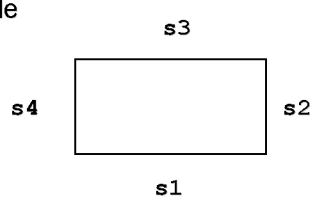
CP model: USER DEFINED CONSTRAINTS

```
touch(X1,X2) :-
      dvar_domain(X1,D1),
      dvar_domain(X2,D2),
      arc_cons_1(D1,D2,D1new),  % user defined propagation
      (dom_compare(>,D1,D1new) -> dvar_update(X1,D1new); true),
      arc_cons_2(D1new,D2,D2new), % user defined propagation
      (dom_compare(>,D2,D2new) -> dvar_update(X2,D2new); true),
      (var(X1),var(X2)
        ->    suspend(touch(X1,X2),3,[X1,X2]->fd:any)
        ;     true),
      wake.
```

- After the propagation, the constraint if not solved is suspended and awaked each time an event ***any of fd*** on one of the variables `(X1,X2)` happens.

CP model: SYMMETRIES

- Problem symmetries: arise when some permutations of the variables map a solution onto another solution.
- Four segments forming a rectangle
- One solution:
  - `X1 = s1`
  - `X2 = s2`
  - `X3 = s3`
  - `X4 = s4`



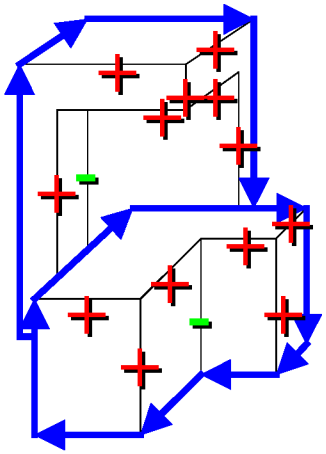
- Other identical solutions:
  - `X1 = s2` `X1 = s3` `X1 = s4`
  - `X2 = s3` `X2 = s4` `X2 = s1`
  - `X3 = s4` `X3 = s1` `X3 = s2`
  - `X4 = s1` `X4 = s2` `X4 = s3`

Time lost to look for already found solutions. Remove symmetries by imposing additional constraints

[Freuder AAAI91], [Puget ISMIS93], [Crawford et al. KR96], [Meseguer, Torras IJCAI99].

MODEL BASED VISION (2)
OBJECT RECOGNITION

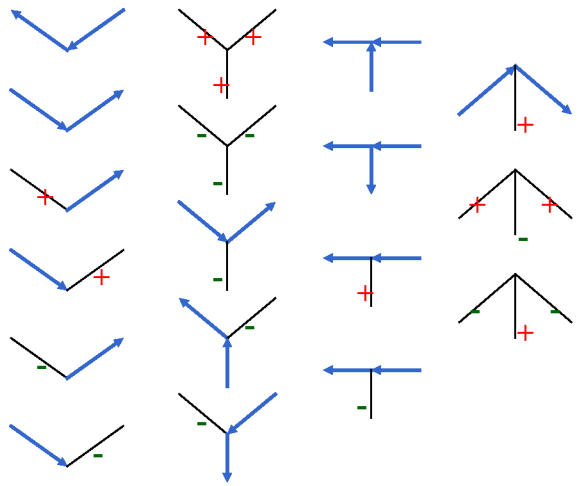
- *Viceversa, può essere richiesto di dare un significato ad un oggetto qualunque.*
- *Es, spiegare gli spigoli di una figura 3D*
  - *quali sono concavi - o convessi +*
  - *quali delimitano una figura →*



OBJECT RECOGNITION

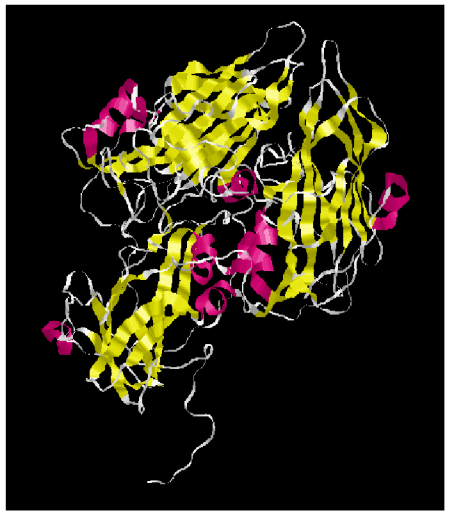
Waltz notò che le combinazioni sono in numero limitato

- *Variabili: segmenti nella figura*
- *Domini: {+,-,←,→}*
- *Vincoli: combinazioni possibili*



PROTEIN FOLDING

- *Una proteina può essere vista come una sequenza di amminoacidi*
- *Coppie di amminoacidi si attraggono o respingono*
- *La proteina si avvolge in una forma con energia minima*
- *Per trovare il modo in cui si avvolge una proteina, si devono svolgere analisi chimiche*



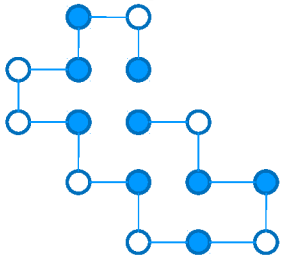
PROTEIN FOLDING

- *Le posizioni in cui un amminoacido si può trovare sono discrete*
- *Se due amminoacidi sono vicini, si attraggono o respingono a seconda dei valori di una tabella data*

	CYS	MET	PHE	ILE	LEU
CYS	-3.477	-2.240	-2.424	-2.410	-2.343
MET	-2.240	-1.901	-2.304	-2.286	-2.208
PHE	-2.424	-2.304	-2.467	-2.530	-2.491
ILE	-2.410	-2.286	-2.530	-2.691	-2.647
LEU	-2.343	-2.208	-2.491	-2.647	-2.501
…					

Simplified Example 2D

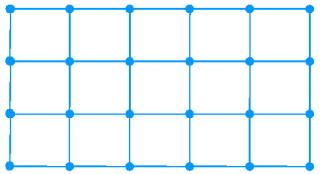
- Protein folds in the plane, possible positions for aminoacids are those with integer coordinates
- Aminoacids are of 2 types:
  - Type 0: does not attract, nor repulse,
  - Type 1: attracts aminoacids of type 1, but only if distance=1
- Example: protein
  [1,0,1,1,0,0,1,0,1,0,1,0,1,1,0,1]



CP Model

- Variables:

  - One variable for each aminoacid
- Domains:



  - Positions in the plane
- Constraints:
  - If two aminoacids are connected
    The second must be immediately up, down, left or right of the first
  - No two aminoacids can be in the same position
- Objective
  - Maximize number of neighboring aminoacids of type 1

Esempio in ECLiPSe

- Consideriamo il caso 2D: gli amminoacidi si possono disporre in una griglia
- 2 tipi di amminoacidi: 1 e 0. Gli 0 sono inerti, gli 1 si attirano, se sono in posizioni vicine
- Prendiamo in ingresso la lista che descrive la proteina (es. [1,0,0,1,0,1,1,0,1]).
- Forniamo in uscita una lista di uguale lunghezza che descrive la posizione di ogni amminoacido nello spazio
  - Ogni amminoacido ha una X ed una Y (intere)
  - Si può riassumere in un'unico valore K=Y*P+X (se P=Xmax+1)

jP	*jP*+1		(*j*+1)*P*-1
…	…	…	…
2*P*	2*P*+1	…	3*P*-1
P	*P*+1	…	2*P*-1
0	1	…	*P*-1

Esempio di codice

```
protein(Lin,Lout) :-
   length(Lin,N), P is N+1,
   length(Lout,N),
   connected(Lout,Steps,P),    /*Un elemento è connesso
   al successivo. Steps=lista di differenze fra le pos
   degli amminoacidi */
   alldifferent(Lout),   /*Una pos occupata da 1 solo
   amminoacido*/
   func(Lout,Lin,Lbool,P),     /*Creo una lista di var
   bool per ogni coppia di amminoacidi: se gli
   amminoacidi sono vicini, allora il bool=1*/
   sumlist(Lbool,F),     F1 #= -F,
   minimize(labeling(Steps),F1).
```

Altri predicati ...

```
% func(Lout,Lin,Lbool,P)
func([_,_],_,[0],_) :- !.
func([H,H1,H2|T],[1,Xin1,Xin2|Tin],Lbool,P) :-
    !,
% Non serve mettere nella funzione obiettivo
    due successivi. Anche il primo ed il terzo
    non possono essere vicini
    cal_fun(H,T,1,Tin,L1,P),
    func([H1,H2|T],[Xin1,Xin2|Tin],L2,P),
    append(L1,L2,Lbool).
func([_|T],[0|Tin],Lbool,P) :-
    func(T,Tin,Lbool,P).
cal_fun(_,[],_,[],[],_).
cal_fun(X,[H|T],1,[1|Tin],[Bool|R],P) :-!,
    X #= H+1#<=>B1, X #= H-1 #<=> B2,
    X #= H+P#<=>B3, X #= H-P #<=> B4,
    sumlist([B1,B2,B3,B4],Bool), Bool::0..1,
    cal_fun(X,T,1,Tin,R,P).
cal_fun(X,[_|T],1,[0|Tin],R,P) :-
    cal_fun(X,T,1,Tin,R,P).
```

```
connected([_],[],_):- !.
connected([A,B|C],[Step|Steps],P):-
    MP is -P,
    Step :: [MP,-1,1,P],
    A #= B+Step,
    connected([B|C],Steps,P).
```


OTHER APPLICATIONS of CP

- Constraint Databases
- Spreadsheet
- Robotics/Control
- Diagnosis
- Test data generation
- Circuit Verification
- Natural Language
- Graphical Interfaces
- Graphical Editors
- Biology (DNA sequencing)
- Qualitative reasoning
- Temporal reasoning
- SAT
- Other LSCO problems

ADVANTAGES OF CP

- Easy problem modelling
- Constraints provide a natural way of implementing propagation rules
- Flexible treatment of variants of original problems:
  - easy introduction of new constraints
  - transparent interaction with other constraints
- Easy control of the search

LIMITATIONS of PURE CP

- Optimization side not very effective
- Over-Constrained problems:
  - no effective way of relaxing constraints
  - hard/soft constraints
- Dynamic Changes:
  - addition/deletion of variables
  - addition/deletion of domain values
  - addition/deletion of constraints

CP EXTENSION FOR OPTIMIZATION

- Integration of OR techniques in CP tools:
  - MP based solvers: 2LP ***[McAloon, Tretkoof PPCP94]***, OPL ***[Van Hentenryck, 99]***, Planner ***[ILOG Planner Manual]***
    - Integration of CPLEX and XPRESS in FD solvers
  - Integration of specialized algorithms for:
    - computing bounds
    - using reduced costs

    } ***[Caseau, Laburthe ICLP97 and CP97], [Focacci, Lodi, Milano ICLP99 and CP99],***
  - Improvement of CP branch and bound
    - ***[Rodosek, Wallace, Hajian Annals OR 97], [Caseau, Laburthe ICLP94 and JICSLP96], [Beringer, DeBacker, LP Formal Method and Pract. Appl. 95]***
  - Integration of local search techniques
  - ***[DeBacker, Furnon,Shaw CPAIOR99], [Caseau, Laburthe CPAIOR99], [Gendreau, Pesant, Rousseau, Transp. Sci. 98]***
  - Integration of branch and cut in a logical setting
    - ***[Bockmayr ICLP95], [Kasper PhD, 99]***

CP EXTENSION FOR OVER-CONSTRAINED PROBLEMS

- HCSP:
  - Implementation of CP solvers exploiting Hierarchical CSP framework
  - Meta programming

[A. Borning OOPSLA87], [A. Borning et al. ICLP89], [M.Jamper PhD, 96]

CP EXTENSION FOR DYNAMIC CHANGES

- DCSP
- ATMS-based solvers

} Complex data-structures

- Interactive Constraint Satisfaction

[R.Dechter, A.Dechter, AAAI88], [Verfaillie, Schiex AAAI94], [Bessiere, AAAI91], [Lamma et al. IJCAI99]

TO KNOW MORE…..

- *Conferences:*
  - *International Conference on Principles and Practice of Constraint Programming (CP)*
  - *Logic programming conferences (ICLP - ILPS - JICSLP)*
  - *AI Conferences (ECAI - AAAI - IJCAI)*
  - *Operations research conferences (INFORMS - IFORS)*
  - *International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR)*
- *Books:*
  - *Krzysztof R. Apt and Mark Wallace,* Constraint Logic Programming using ECL^{i}PS^e, *Cambridge*
  - *K. Marriott and P. Stuckey,* Programming with constraints: An Introduction, *MIT Press*
  - *Rina Dechter,* Constraint Processing, *Morgan Kaufmann*

TO KNOW MORE…..

- *Journals:*
  - *Constraint - An International Journal*
  - *AI - LP - OR Journals*
- *Industrial Applications:*
  - *COSYTEC, ILOG, CrosscoreOptimization, SIEMENS, BULL*
- *News group:* `comp.constraints`
- *Mailing lists:* `CPWORLD@gmu.edu`
- *Constraint Archive:* `http://4c.ucc.ie/web/archive/`