Hybridization of complete and incomplete methods to solve CSP

Tony Lambert

LERIA, Angers, France

LINA, Nantes, France

October 27th, 2006

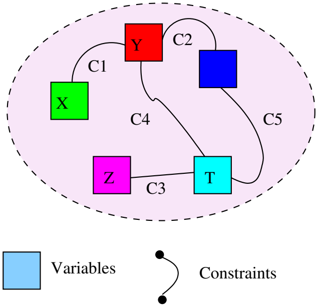
Rapporteurs :	François FAGES	Directeur de Recherche, INRIA Rocquencourt
	Bertrand NEVEU,	Ingénieur en Chef des Ponts et Chaussées, HDR
Examinateurs :	Steven PRESTWICH,	Associate Professor, University College Cork
	Jin-Kao HAO,	Professeur à l'Université d'Angers
Directeurs de thèse :	Éric MONFROY,	Professeur à l'Université de Nantes
	Frédéric SAUBION,	Professeur à l'Université d'Angers

Constraint programming process

Formulate the problem with constraints as a CSP

- constraint : a relation on variables and their domains
- Constraint Satisfaction Problem (CSP) : a set of constraints together with a set of variable domains

CSP model



- $\mathcal{X} = \{x_1, \ldots, x_n\}$ set of n variables,
- $\mathcal{D} = \{D_{x_1}, \ldots, D_{x_n}\}$ set of n domains,
- $\mathcal{C} = \{c_1, \ldots, c_m\}$ set of m constraints.

Problems are modeled as CSPs (X, D, C)

Some variables to represent objects

$(X = \{X_1, \ldots, X_n\})$

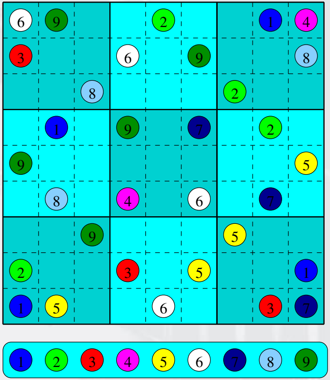
Domains over which variables can range

$(D = D_1 \times \ldots \times D_n)$

Some constraints to set relation between objects

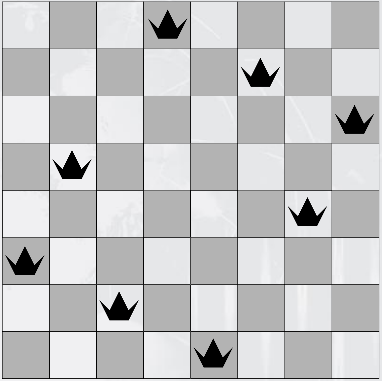
$$C_1 : X \leq Y * 3$$
$$C_2 : Z \neq X - Y$$
$$\ldots$$

Example : Sudoku problem



- $\mathcal{X}$ = { all the cells in the grid},
- $\mathcal{D}$ = { to each cell a value in [1..9]},
- $\mathcal{C}$ = { set of alldiff constraints : all digits appears only once in each row ; once in each column and once in each 3 $\times$ 3 square the grid has been subdivided}

Example : 8 Queens



- $\mathcal{X} = \{x_1, ...x_8\}$ Columns,
- $\mathcal{D} = \{D_{x_1}, ..., D_{x_8}\}$ for each columns, the row the queen is placed $D_{x_i} = [1..8]$,
- $\mathcal{C} = \{$ not 2 queens in the same row or same diagonal$\}$

CSP solving

A solution

- Given a search space $\mathcal{S} = D_{x_1} \times ... \times D_{x_n}$
- an assignment s is a solution if :
  - $s \in \mathcal{S}$ and
  - $\forall c \in \mathcal{C}, s \in c$

Solving a CSP can be :

- compute whether the CSP has a solution (satisfiability)
- find A solution
- find ALL solutions
- find optimal solutions (global optimum)
- find A good solution (local optimum)

Outline

complete/incomplete methods

- **complete methods** (such as propagation + split)
  - complete exploration of the search space
  - detect if no solution
  - global optimum
  - generally slow for hard combinatorial problems
- **incomplete methods** (such as local search)
  - focus on some “promising” parts of the search space
  - do not detect if no solution
  - no global optimum
  - “fast” to find a “good” solution

Solving CSP

Methods

- **Complete methods**
- Incomplete methods

Complete methods

General methods that can be adapted to several types of constraints and several types of variables

- search methods : to explore the search space
- constraint propagation algorithms : to repeatedly remove inconsistent values from domains of variables

First approach

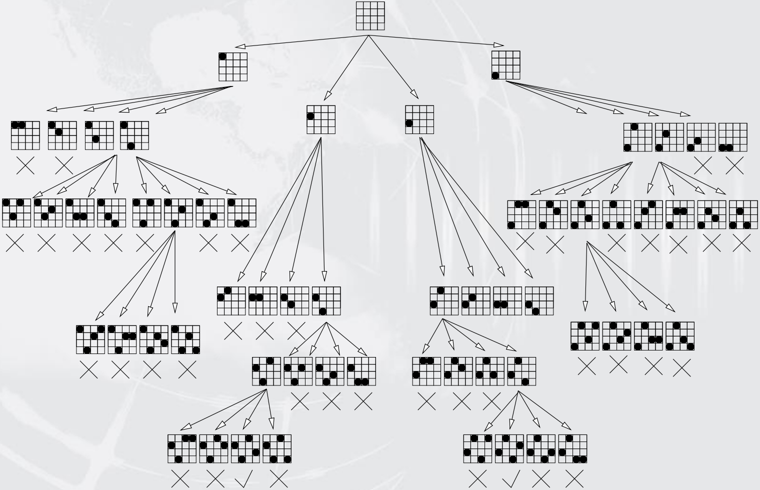
Complete Methods

- Search space : $D_{x_1} \times \cdots \times D_{x_n}$
- Enumerate all assignments

Backtracking

- search (backtrack)
- Select variables
- Split / enumeration

Backtracking for 4 - Queens



Constraint propagation : reducing domains

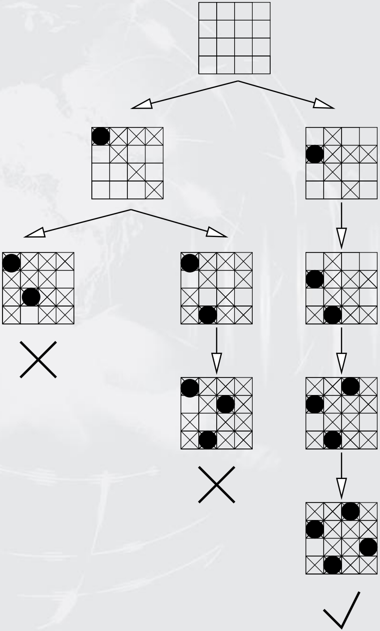
Generally :

- reduce domains using constraint and domains
- $\rightarrow$ reduce the search space

Generic domain reduction :

- given a constraint C over $x_1, \ldots, x_n$ with domains $D_1, \ldots, D_n$
- select a variable x_i reduce its domain
- delete from D_i all values for x_i that do not participate in a solution of C

Forward Checking



Constraint propagation

- constraint propagation mechanism : repeatedly reduce domains
- replace a CSP by a CSP which is :
  - equivalent (same set of solutions)
  - “smaller” (domains are reduced)

Constraint propagation framework

Can be seen as a fixed point of application of reduction functions

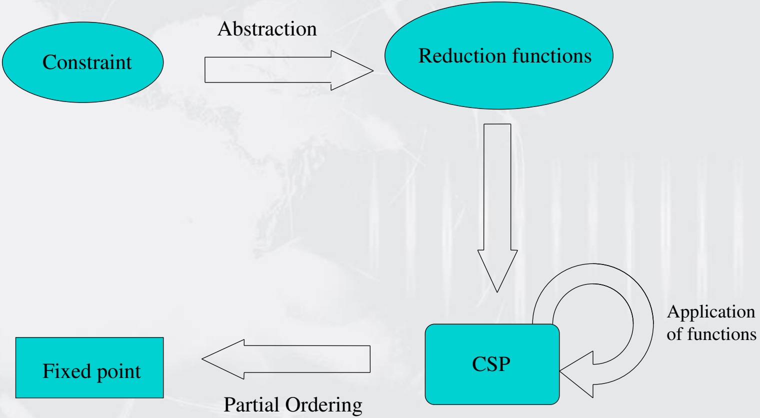
- reduction function to reduce domains or constraints
- can be seen as an abstraction of the constraints by reduction functions

Chaotic iteration

- Compute a limit of a set of functions [Cousot and Cousot 77]
- monotonic and inflationary functions in a generic algorithm to achieve consistency [Apt 97]

Abstract model K.R. Apt [CP99]

For propagation (based on chaotic iterations)



Partial ordering and functions

Partial Ordering

Given a CSP $(\mathcal{X}, \mathcal{D}, \mathcal{C})$

- $\mathcal{P}(\mathcal{D})$: all possible subset from $\mathcal{D}$
- $\sqsubseteq$: subset relation $\supseteq$

$\Longrightarrow (\mathcal{P}(\mathcal{D}), \sqsubseteq)$ is a partial ordering

Functions

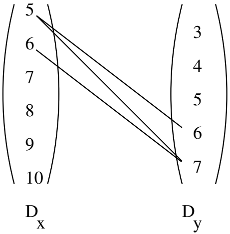
Given a set F of functions on $\mathcal{D}$, every $f \in F$ is :

- inflationary : $x \sqsubseteq f(x)$
- monotonic : $x \sqsubseteq y$ implies $f(x) \sqsubseteq f(y)$
- idempotent : $f(f(x)) = f(x)$

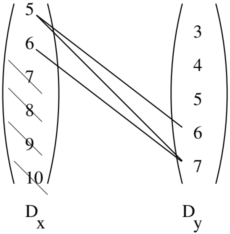
$\Longrightarrow$ Every sequence of elements $d_0 \sqsubseteq d_1 \sqsubseteq \ldots$ with $d_j = f_{i_j}(d_{j-1})$ stabilizes to a fix point.

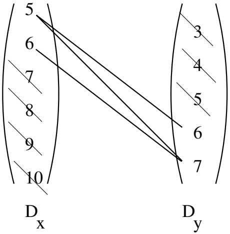
Example of reduction functions (1)

Constraint $x < y$



Arc_consistence on D_x and Arc_consistence on D_y





Example of reduction functions (2)

Arc consistency

Consider a constraint $x < y$ with $D_x = [5..10]$ and $D_y = [3..7]$ then 2 functions $\mathcal{D} \rightarrow \mathcal{D}$:

- Arc_consistence 1 :

$$(D_x, D_y) \mapsto (D_{x\prime}, D_y)$$

  s.t.

$$D_{x\prime} = \{a \in D_x \mid \exists b \in D_y, a < b\}$$

- Arc_consistence 2 :

$$(D_x, D_y) \mapsto (D_x, D_{y\prime})$$

  s.t.

$$D_{y\prime} = \{b \in D_y \mid \exists a \in D_x, a < b\}$$

A Generic Algorithm to reach fixpoint

for **constraint propagation** : ordering on size of domains
work on a CSP

$F = \{$ set of propagation functions$\}$
$X =$ initial CSP
$G = F$
While $G \neq \emptyset$
    choose $g \in G$
    $G = G - \{g\}$
    $G = G \cup update(G, g, X)$
    $X = g(X)$
EndWhile

Solving CSP

Methods

- Complete methods
- **Incomplete methods**

Second Approach

Incomplete methods

- heuristics algorithms
- Metaheuristics

two families

- Local search
  - Simulated annealing [Kirkpatrick et al, 1983]
  - Tabu Search [Glover, 1986]
  - ...
- Evolutionary Algorithms
  - Genetic Algorithms [Holland, 1975]
  - Genetic programming [Koza, 1992]
  - ...

Incomplete methods

Definitions

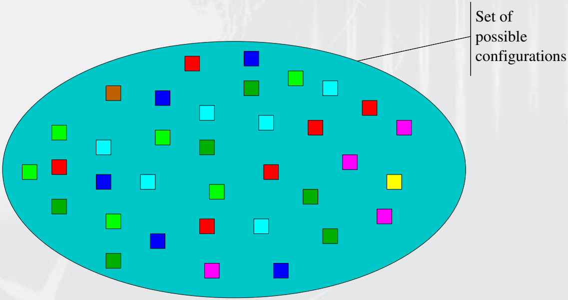
- Explore a $D_1 \times D_2 \times \cdots \times D_n$ search space
- Move from neighbor to neighbor (resp. generation to generation) thanks to an evaluation function
  - Intensification
  - Diversification

Properties :

- focus on some “promising” parts of the search space
- does not answer to unsat. problems
- no guaranteed
- “fast” to find a “good” solution

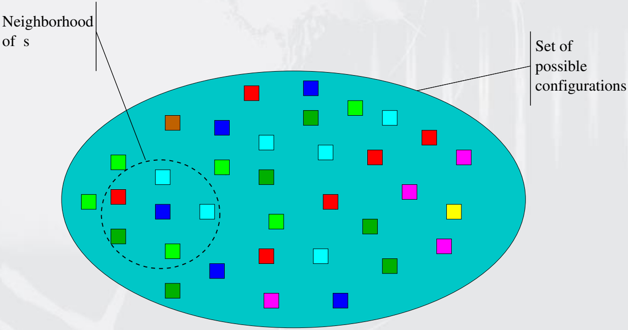
Local search (1)

- Search space : set of possible configurations
- Tools : neighborhood and evaluation function



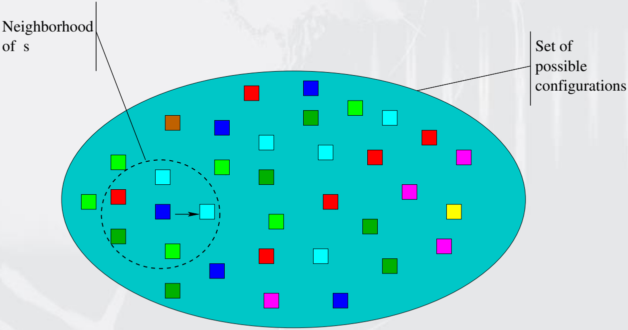
Local search (2)

- Search space : set of possible configurations
- Tools : neighborhood and evaluation function



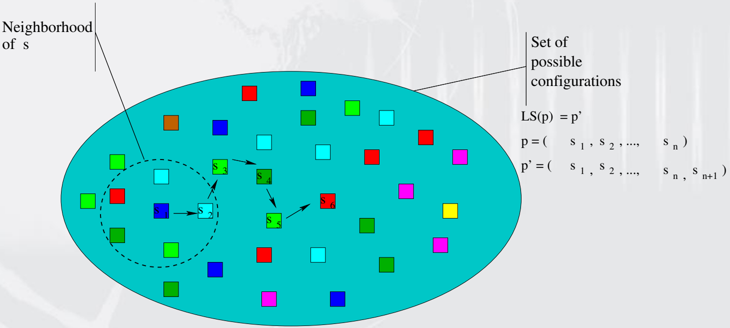
Local search (3)

- Search space : set of possible configurations
- Tools : neighborhood and evaluation function



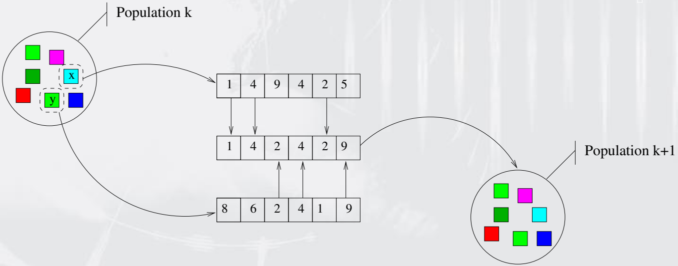
Local search (4)

- Search space : set of possible configurations
- Tools : neighborhood and evaluation function



Genetic algorithms (1)

- Search space : set of possible configurations
- Tools : population, **crossing**, mutations, and evaluation function

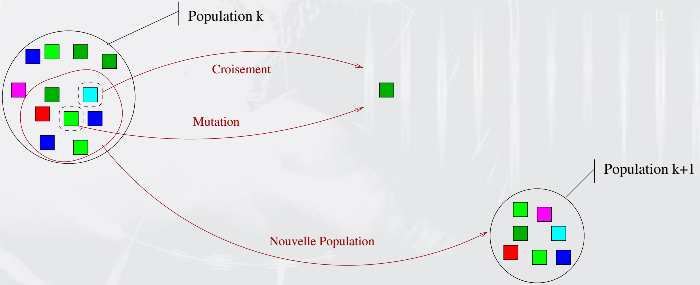


Genetic algorithms (2)

- Search space : set of possible configurations
- Tools : population, crossing, **mutations**, and evaluation function

Genetic algorithms (3)

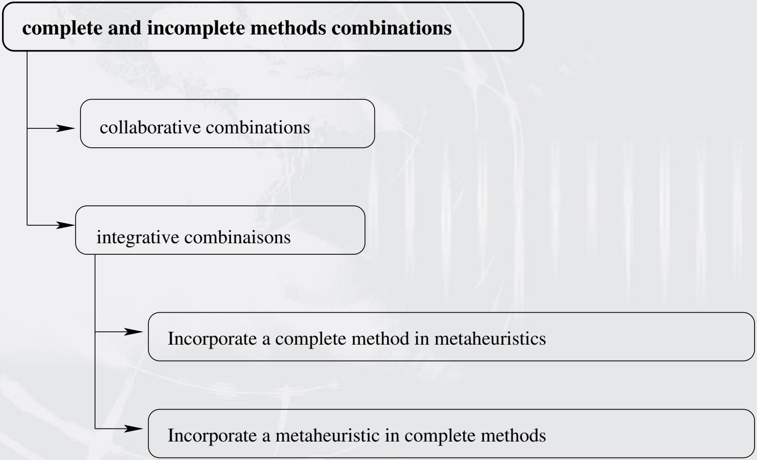
- Search space : set of possible configurations
- Tools : population, crossing, mutations, and **evaluation function**



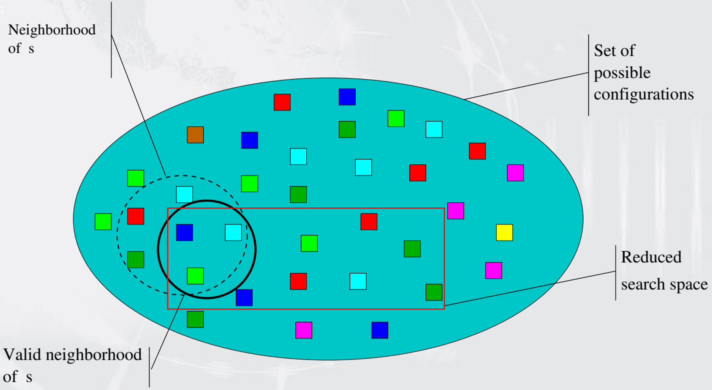
Hybridization : getting the best of the both

- Efficiency : faster complete solver
- Quality : better solutions (better optimum)
- Generally :
  - Ad-hoc systems (designed from scratch)
  - Dedicated to a class of problems
  - Master-slave approaches (LS for CP, CP for LS)

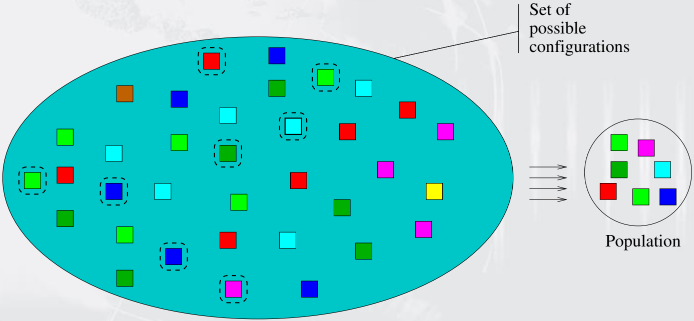
Hybridization : Overview



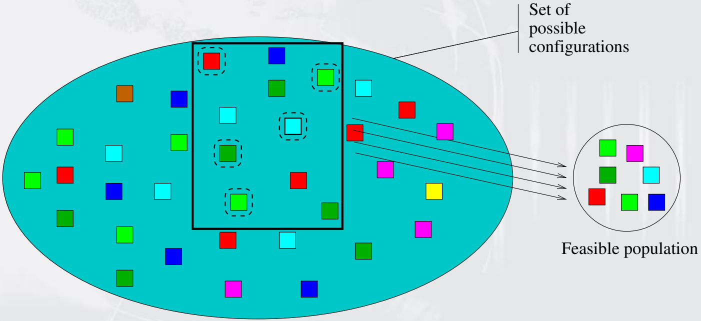
CP for LS


Neighborhood of s
Set of possible configurations
Reduced search space
Valid neighborhood of s

CP for GA (1)



CP for GA (2)



Hybridization : getting the best of the both

- Idea :
  - fine grain hybridization
  - finer strategies
  - every technique at the same level
  - one algorithm squeleton
  - easier to modify, extend, compare, ...

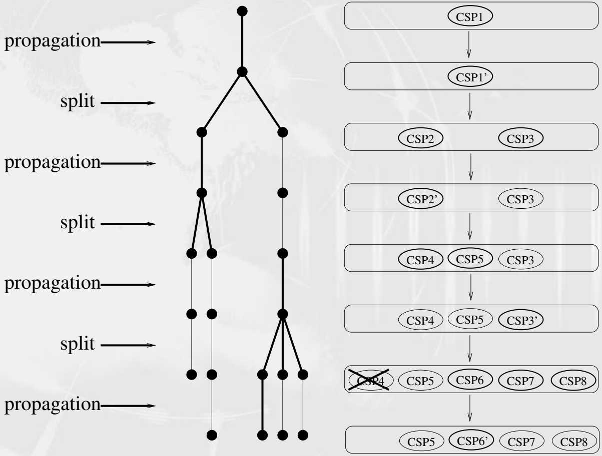
Our purpose :

- Use of an existing theoretical model for CSP solving
- Definition of the solving process

Integrate :

- **Split**
- LS
- GA

Search Tree

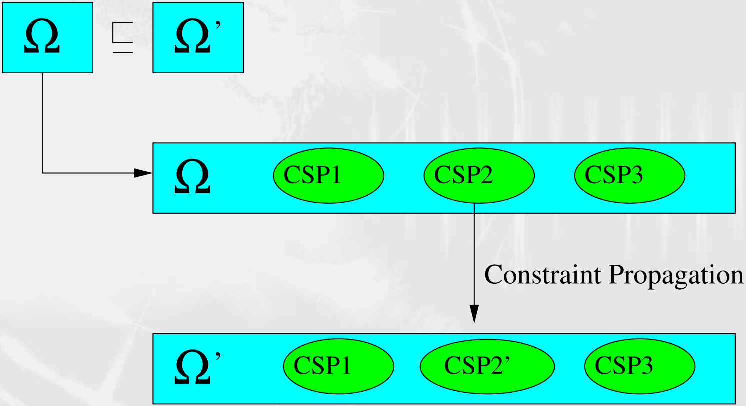

propagation
split
propagation
split
propagation
split
propagation
CSP1
CSP1'
CSP2
CSP3
CSP2'
CSP3
CSP4
CSP5
CSP3
CSP4
CSP5
CSP3'
CSP4
CSP5
CSP6
CSP7
CSP8
CSP5
CSP6'
CSP7
CSP8

Idea 1 : Integrate split in GI

- use the CP framework algorithm
- split and reduction at the same level
- work on union of CSP

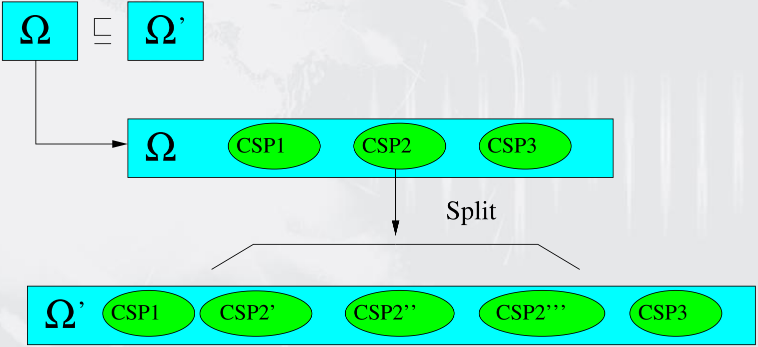
Theoretical model for CSP solving (2)

Reduction : by constraint propagation :



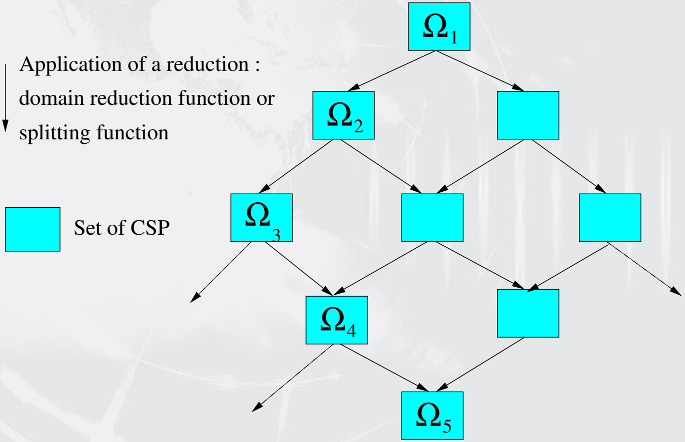
Theoretical model for CSP solving (3)

Reduction by domain splitting :



Theoretical model for CSP solving (1)

Partial ordering :



Terminates with a fixed point : set of solutions or inconsistent CSP

Our purpose :

- Use of an existing theoretical model for CSP solving
- Definition of the solving process

Integrate :

- Split
- **LS**
- GA

Idea 2 : Integrate LS and CP

Local Search

- In theory : infinite
- In practice : number of iteration

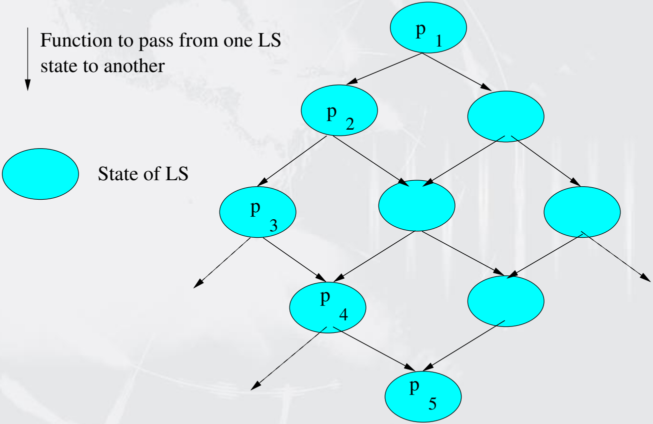
Characteristics of a LS path

- notion of solution
- Operational (computational) point of view : **path = samples**
- maximum length

Goal :

- LS ordering
- integrate LS in CSP

LS as moves on a partial ordering



Terminates with a fixed point : local minimum, maximum number of iteration

Integrate samples for LS

- A sample :
  - Depends on a CSP
  - Corresponds to a LS state
- an SCSP contains (LS+domain reduction) :
  - A set of domains ;
  - A (list of) sample for local search (a LS path).
- an σCSP contains (LS+domain reduction+split) :
  - A set of SCSP

Theoretical model for hybridization (3)

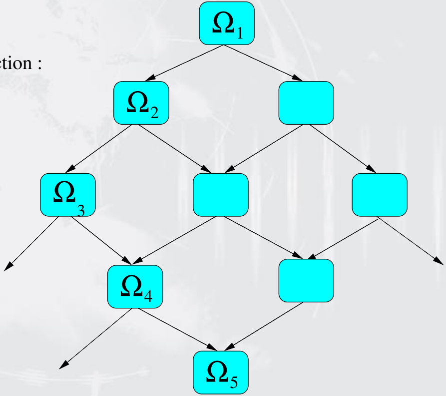
Ordering over σCSP

Application of a reduction :

function :

- domain reduction,
- splitting,
- or LS step.

Set of SCSP



➡ A solution before the end of the process, given by LS

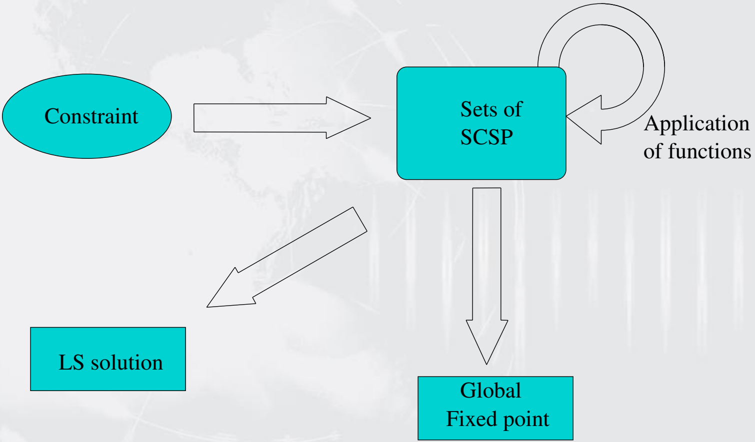
➡ Terminates with a fixed point : set of solutions or inconsistent SCSP

Algorithm

Always the GI algorithm

- In the chaotic iteration framework
- Finite domains (sets of SCSP)
- Monotonic functions (LS move, domain reduction, splitting)
- Decreasing functions
- $\rightarrow$ termination and computation of fixed point (solution of CP)
- Practically : can stop before with a LS solution

General process CP+LS



Strategies through reduction functions

- Reduction / Split / LS functions
- Choose a / several SCPS to apply functions
- Choose a / several domains (Prop. - Split)
- Tests : Random - Depth first - FC

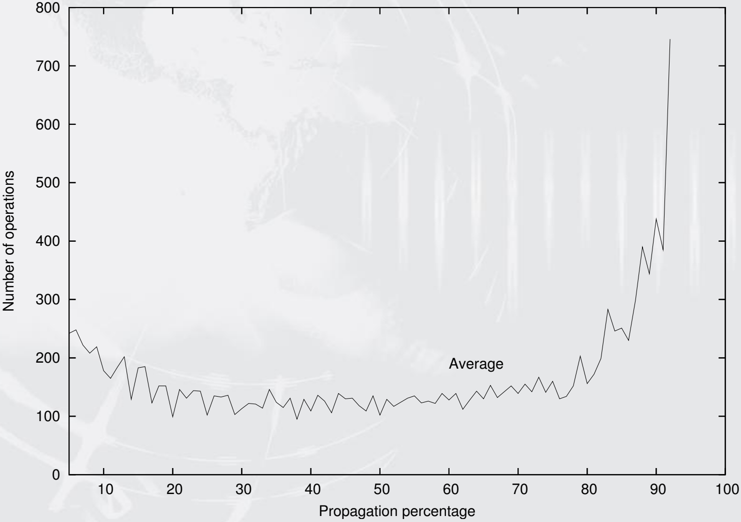
Strategies : ratio LS-CP

- Ratio of LS and CP functions applied : (red%,split%,ls%)
  - 10% of split compared to reduction
  - CP alone : (90,10,0)
  - LS alone : (0,0,100)
- LS strategy : tabu
- Choose : LS, reduction, or split but keep the given ratios

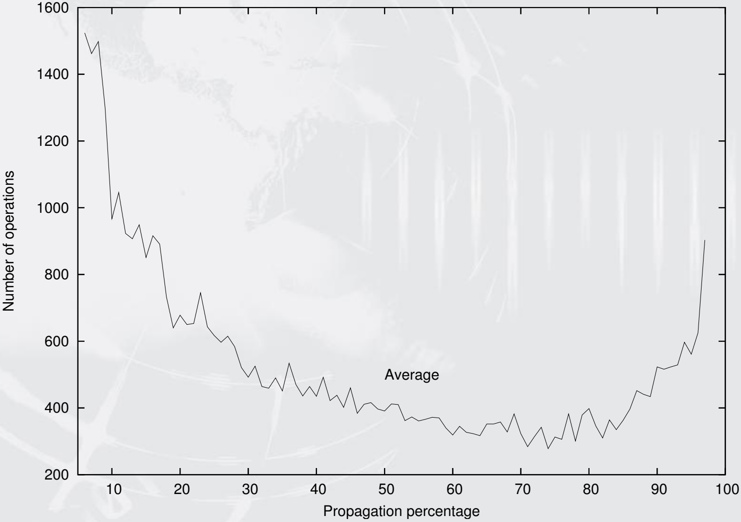
Implementation

- C++
- Generic Algorithm
- choose function with percentages

Langford number 4 2 cooperation area



SEND + MORE = MONEY cooperation area



Ranges

Problem	S+M	LN42	Zebra	M. square	Golomb
Rate FC	70 - 80	15 - 25	60 - 70	30 - 45	30 - 40

Best range of propagation rate (α) to compute a solution

Strategy	Method	S+M	LN42	M. square	Golomb
Random	LS	1638	383	3328	3442
	CP+LS	1408	113	892	909
	CP	3006	1680	1031	2170
D-First	LS	1535	401	3145	3265
	CP+LS	396	95	814	815
	CP	1515	746	936	1920
FC	LS	1635	393	3240	3585
	CP+LS	22	192	570	622
	CP	530	425	736	1126

Avg. nb. of operations to compute a first solution

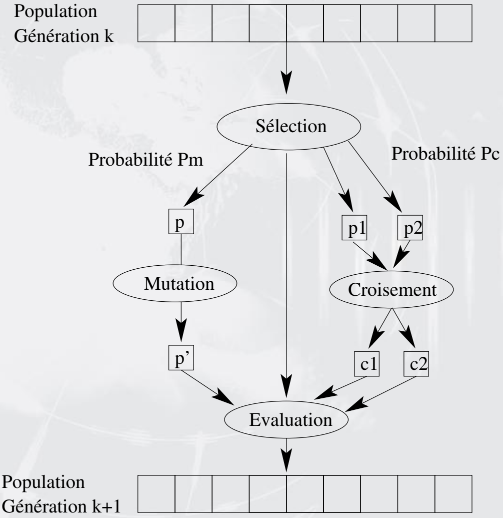
Our purpose :

- Use of an existing theoretical model for CSP solving
- Definition of the solving process

Integrate :

- Split
- LS
- **GA**

Idea 3 : GA = moves in a partial ordering



GA ordering

Characteristics of a GA path

- notion of solution
- maximum length
- Operational (computational) point of view : **path = generations**

Here a macroscopic point of view :

1 iteration = 1 generation

Integrate generations for GA in CSPs

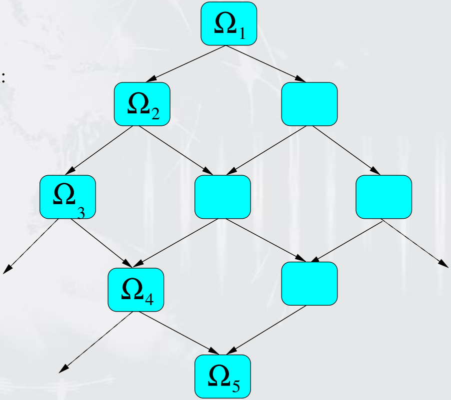
- a generation :
  - depends on a CSP
  - corresponds to a GA search state
- an ogCSP contains (GA+domain reduction) :
  - a set of domains ;
  - a (list of) population(s) for GA (a GA path).
- an OGCSP contains (GA+domain reduction+split) :
  - a set of ogCSPs

Theoretical model for hybridization CP+GA

Ordering over OGCSPs

Application of a reduction: function:
- domain reduction
- split
- or GA.

set of ogCSP



→ an optimal solution given by GA before the end of the process

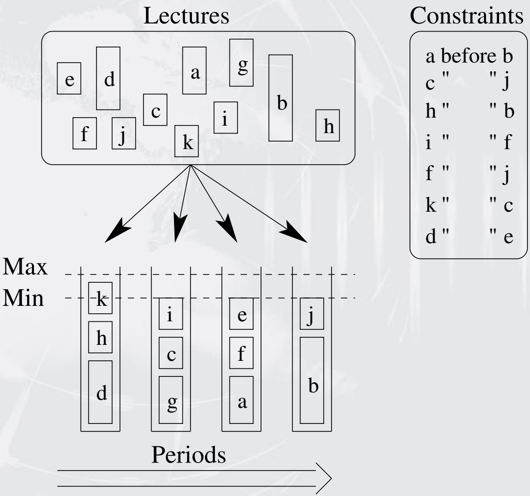
→ fixed point: set of solutions or inconsistency

Algorithm

Always the GI algorithm

- In the chaotic iteration framework
- Finite domains (set of OGCSP)
- Monotonic functions (GA move, domain reduction, splitting)
- Decreasing functions
- $\rightarrow$ termination and computation of fixed point (solution of CP)
- Practically : can stop before with an optimum for GA

Application BACP (to optimize loads of periods)

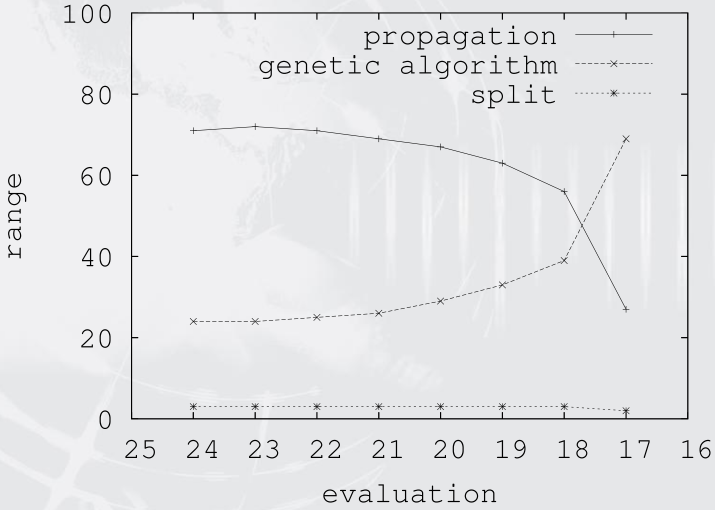


Experimentations

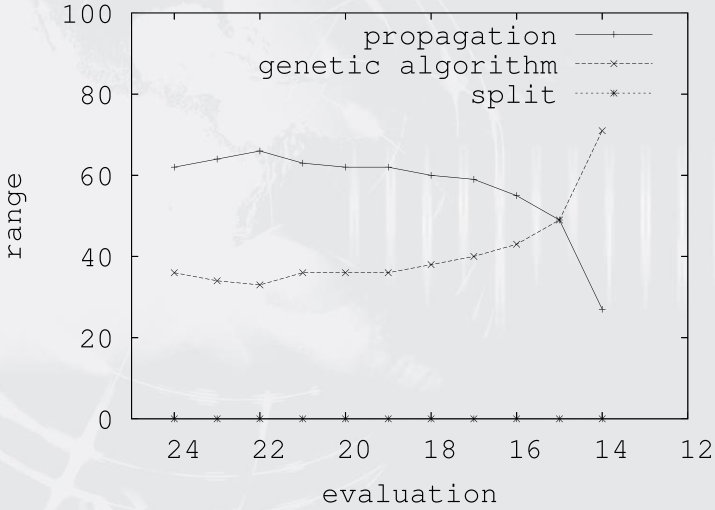
On optimization problems

- Here, ratios are (45,5,50)
- Mesure the real impact on the resolution

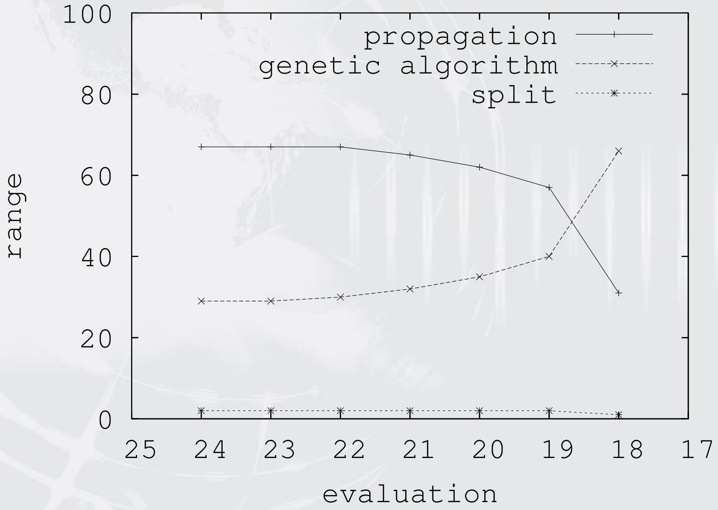
BACP8


100
80
60
40
20
0
range
25 24 23 22 21 20 19 18 17 16
evaluation
propagation
genetic algorithm
split

BACP10


100
80
60
40
20
0
range
propagation
genetic algorithm
split
24
22
20
18
16
14
12
evaluation

BACP12

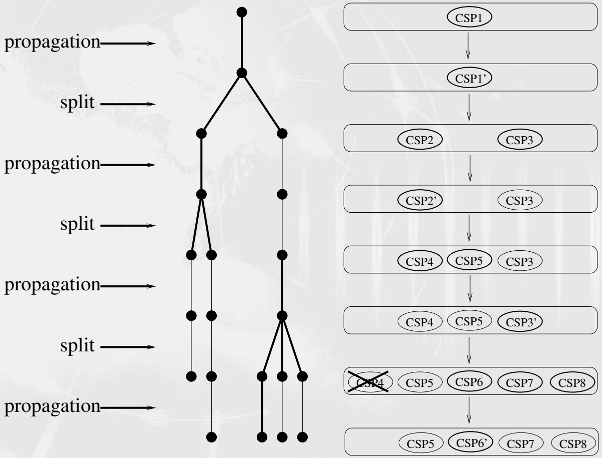

propagation
genetic algorithm
split
range
evaluation
0
20
40
60
80
100
25
24
23
22
21
20
19
18
17

Motivation

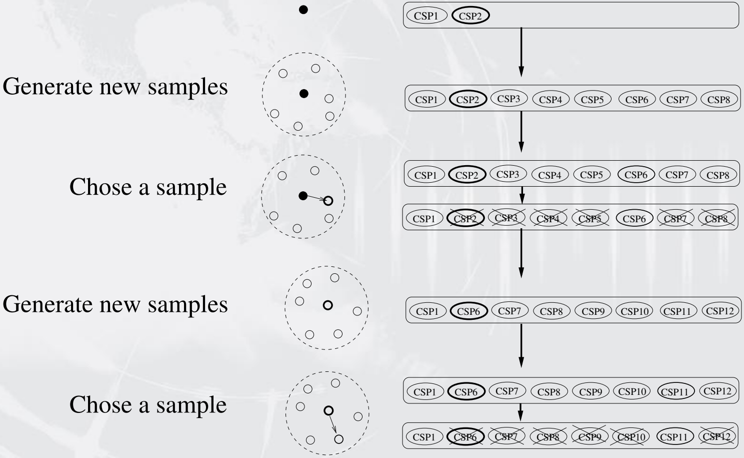
- LS + CP : a search path
- GA + CP : a sequence of generations
- But : GA + CP + LS ?

$\Longrightarrow$ Notion of sample $\neq$ generation
$\Longrightarrow$ Need to consider sample and individual at the same level : CSP level

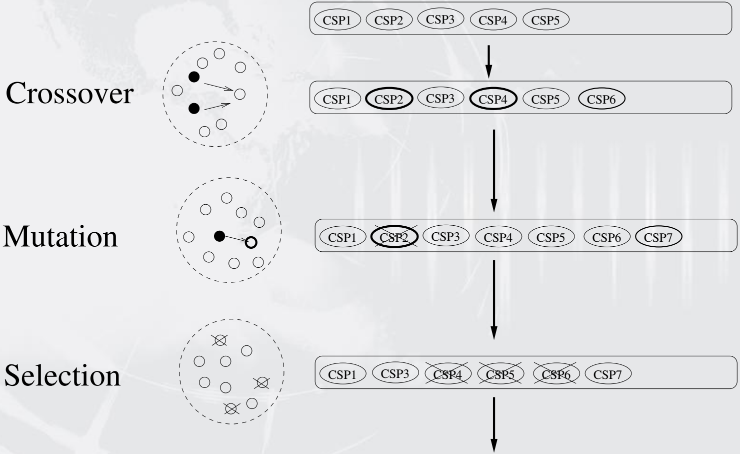
Search Tree


propagation
split
propagation
split
propagation
split
propagation
CSP1
CSP1'
CSP2
CSP3
CSP2'
CSP3
CSP4
CSP5
CSP3
CSP4
CSP5
CSP3'
CSP4
CSP5
CSP6
CSP7
CSP8
CSP5
CSP6'
CSP7
CSP8

Example : Local Search



Example : Genetic algorithms



Idea 4 : break up methods

Basic components

- reducing is a search component
- splitting is a search component
- generating a neighborhood is a search component
- moving to sample is a search component

Basic behaviours

- splitting and generating a neighborhood
- reducing, selecting and moving to sample

Basic fonctions

Sampling $\mathcal{S}$:

$$\mathcal{P}(\langle X, C, \mathcal{P}(D_1) \times \cdots \times \mathcal{P}(D_n)\rangle) \rightarrow \mathcal{P}(\langle X, C, \mathcal{P}(D_1) \times \cdots \times \mathcal{P}(D_n)\rangle)$$

$$\{\phi_1, \ldots, \phi_n\} \mapsto \{\phi_1, \ldots, \phi_n, \phi_{n+1}\}$$

s.t. $\exists \phi_i$ with $\phi_i \sqsubseteq \phi_{n+1}$

Reducing $\mathcal{R}$:

$$\mathcal{P}(\langle X, C, \mathcal{P}(D_1) \times \cdots \times \mathcal{P}(D_n)\rangle) \rightarrow \mathcal{P}(\langle X, C, \mathcal{P}(D_1) \times \cdots \times \mathcal{P}(D_n)\rangle)$$

$$\{\phi_1, \ldots, \phi_i, \ldots, \phi_n\} \mapsto \{\phi_1, \ldots, \phi_i', \ldots, \phi_n\}$$

Where $\phi_i' = \emptyset$ or $\phi = \langle X, C, D_i\rangle$ and $\phi' = \langle X, C, D_i'\rangle$ s.t. $D_i' \subseteq D_i$.

Reduction functions (1)

Domain reduction (DR)

$$\{\phi_1, \ldots, \phi_i, \ldots, \phi_n\} \rightarrow^{DR} \{\phi_1, \ldots, \phi'_i, \ldots, \phi_n\}$$

Where $DR = \mathcal{R}^m$ with $m > 0$

Split (SP)

$$\{\phi_1, \ldots, \phi_i, \ldots, \phi_n\} \rightarrow^{SP} \{\phi_1, \ldots, \phi_i^1, \ldots, \phi_i^m, \ldots, \phi_n\}$$

Where $SP = \mathcal{S}^m\mathcal{R}$

Local Search (LS)

$$\{\phi_1, \ldots, \phi_i, \ldots, \phi_n\} \rightarrow^{LS} \{\phi_1, \ldots, \phi'_i, \ldots, \phi_n\}$$

Where $LS = \mathcal{S}^m\mathcal{R}^{m-1}$

Reduction functions (2) Genetic Algorithms

Crossover

$$\{\phi_1, \ldots, \phi_n\} \rightarrow^{CR} \{\phi_1, \ldots, \phi_n, \phi_{n+1}\}$$

Where $CR = \mathcal{S}$

Mutation

$$\{\phi_1, \ldots, \phi_i, \ldots, \phi_n\} \rightarrow^{MU} \{\phi_1, \ldots, \phi_i', \ldots, \phi_n\}$$

Where $MU = \mathcal{SR}$

Selection

$$\{\phi_1, \ldots, \phi_n\} \rightarrow^{SE} \{\phi_1, \ldots, \phi_n'\}$$

Where $SE = \mathcal{R}^m$.

Properties

- a strategy is a sequence of words $\in \{DR, SP, LS, SE, CR, MU\}$
- is it finite sequences ?
- need conditions to avoid loops

Conclusion

- A generic model for hybridizing complete (CP) and incomplete (LS and GA) methods
- Implementation of modules working on the same structure
- Complementarity of methods : hybridization

Future

- synergies between :
  - complete + incomplete methods,
  - CP + AI + OR + . . .
- learning strategies, rules based system with a language
- providing more tools in a generic environment
- Design of strategies

Hybridization of complete and incomplete methods to solve CSP

Tony Lambert

LERIA, Angers, France

LINA, Nantes, France

October 27th, 2006

Rapporteurs :	François FAGES	Directeur de Recherche, INRIA Rocquencourt
	Bertrand NEVEU,	Ingénieur en Chef des Ponts et Chaussées, HDR
Examinateurs :	Steven PRESTWICH,	Associate Professor, University College Cork
	Jin-Kao HAO,	Professeur à l'Université d'Angers
Directeurs de thèse :	Éric MONFROY,	Professeur à l'Université de Nantes
	Frédéric SAUBION,	Professeur à l'Université d'Angers