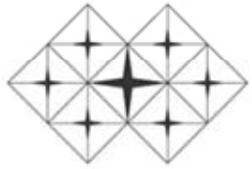
Nurse Rostering with Constraint Programming – An Overview

H. Simonis
IC-Parc, Imperial College London
Parc Technologies Ltd

Reusing some slides from [Simonis97]

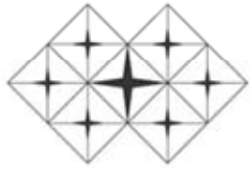
Overview of presentation

- Constraint Programming in a nutshell
- The Nurse Rostering problem
- Using Global Constraints
- Soft Constraints
- User interaction
- User-driven specification
- Case study
- References and systems



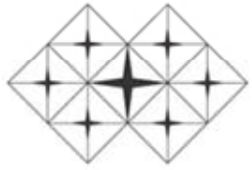
Constraint Programming in a nutshell

- Declarative description of problems with
  - *Variables* which range over (finite) sets of values
  - *Constraints* over subsets of variables which restrict possible value combinations
  - A *solution* is a value assignment which satisfies all constraints
- Constraint propagation/reasoning
  - Removing inconsistent values for variables
  - Detect failure if constraint can not be satisfied
  - Interaction of constraints via shared variables
  - Incomplete
- Search
  - User controlled assignment of values to variables
  - Each step triggers constraint propagation



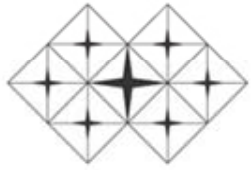
History

- Application view
  - Graphics: Sutherland 63, Borning 81
  - Vision: Waltz 72
  - Simulation: Sussmann & Steele 80
- Problem structure view
  - Consistency techniques: Montanari 74, Mackworth 77, Freuder 78
  - Constraint logic programming: Colmerauer 84, Lassez & Jaffar 87, Dincbas & al 86
- Methods view
  - OR techniques: Lauriere 78
  - Global constraints: Aggoun & Beldiceanu 93



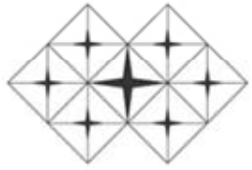
Nurse Rostering Problem (Typical)

- In hospitals, nurses work in three shift system
- Problem solved by ward (typical 20-30 nurses)
- Each nurse works at most one shift per day
  - M (morning), A (afternoon) or N (night) shift
- R (rest) periods must be allocated equally to each nurse
- The demand on each day for each shift is varying, but known
  - depends on number of patients, procedures
- The work assignment must follow a set of rules
  - many rules are strict, others are preferences
- A solution consists in an assignment for each nurse on each day to a value M,A,N,R which satisfies all rules
- Rule sets are changing over time



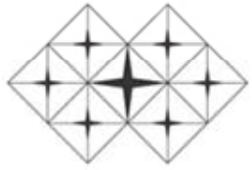
Objective(s)

- Find a feasible solution
- Often, as well:
  - Balance work load of nurses
  - Do not balance work load of nurses
  - Satisfy as many preferences as possible
  - Relax as few constraints as possible



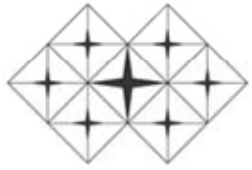
Example rule set

- The demand of service required must be satisfied for each day
- In the scheduling period, there must be 4 days rest for each person
- In any sequence of 6 days, there must be at least one rest day
- In any sequence of 5 days, there may be at most 2 rest days
- In the scheduling period, a person is not allowed to work more than 4 night shifts
- Nobody is allowed to work 4 night shifts in a row
- A day working night shift may not be followed by a day working morning shift
- It is not allowed to work night shifts, switch one day to a day shift and then work nightshift again
- There can be no bridge days, one day of work in between two days off



Model

- Variables
  - Variables describe the work of each person on each day
  - Four possible values
    - 0 Rest
    - 1 Morning
    - 2 Afternoon
    - 3 Night
  - Variables arranged in array of lines and columns
    - Lines = work for one person
    - Columns = work on one day



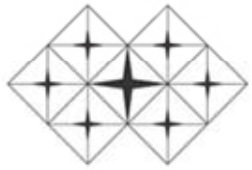
Typical solution

Quit | About | Help | Run | **Strategy** e | i

Problem 16 **Cost** 0 **Backtracking Steps** 298 **Execution Time [msec]** 3225

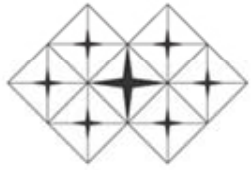
N	R	N	A	R	N	N	R	M	M	M	R	M	A
N	R	N	A	R	N	A	M	N	R	R	M	M	M
A	R	A	N	N	R	N	A	R	A	M	A	N	R
M	M	A	N	R	R	M	A	N	R	A	A	A	R
M	N	R	N	A	R	M	A	R	N	A	R	A	A
M	N	R	M	N	N	N	R	R	M	M	M	A	R
M	A	R	M	N	N	N	R	N	A	M	R	R	M
R	M	M	R	M	A	R	N	A	M	N	A	R	A
R	M	A	R	M	A	R	N	A	A	R	A	N	A
R	M	A	R	M	M	R	N	A	A	R	N	N	N

4	4	1	2	3	1	2	1	1	3	4	2	2	2
1	1	4	2	1	2	1	3	3	4	2	4	3	4
2	2	2	3	3	4	4	3	3	1	1	1	3	1

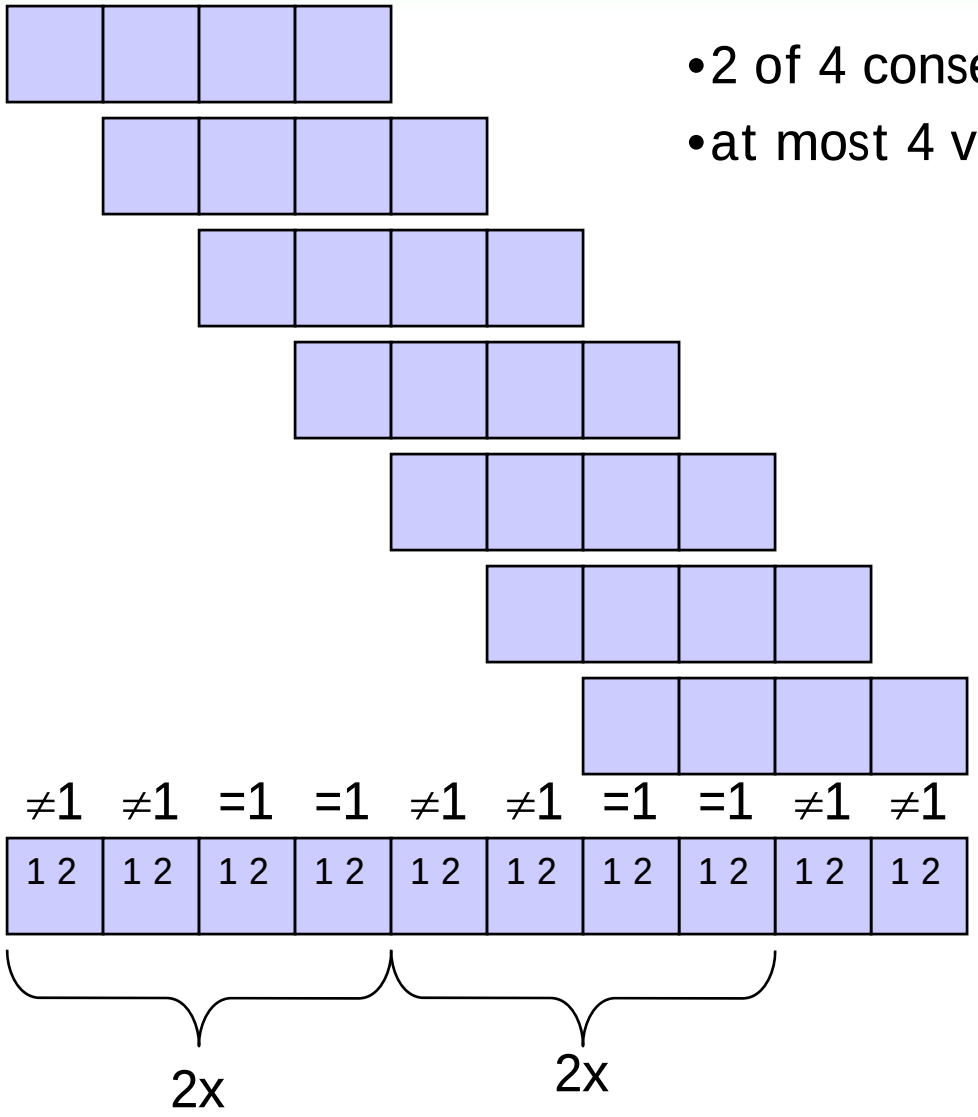


How to express the constraints?

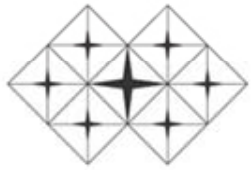
- Simplistic
  - Combination of primitive constraints
    - Nobody is allowed to work 4 night shifts in a row
    - X1=N & X2=N & X3=N => X4 \= N
  - Some constraint require excessive disjunction/conjunction
    - In any sequence of 5 days, there may be at most 2 rest days
  - May need switch to 0/1 variables
    - X1R, X1M, X1A, X1N with domain 0/1
      - instead of X1 with domain R,M,A,N
      - X1R+X2R+X3R+X4R+X5R <= 2
- Problems
  - Very limited propagation, no global view
  - Massive number of primitive constraints



Global reasoning: Example

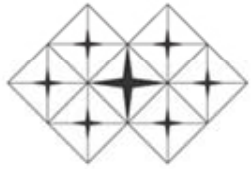


- 2 of 4 consecutive variables have value 1
- at most 4 variables have value 1

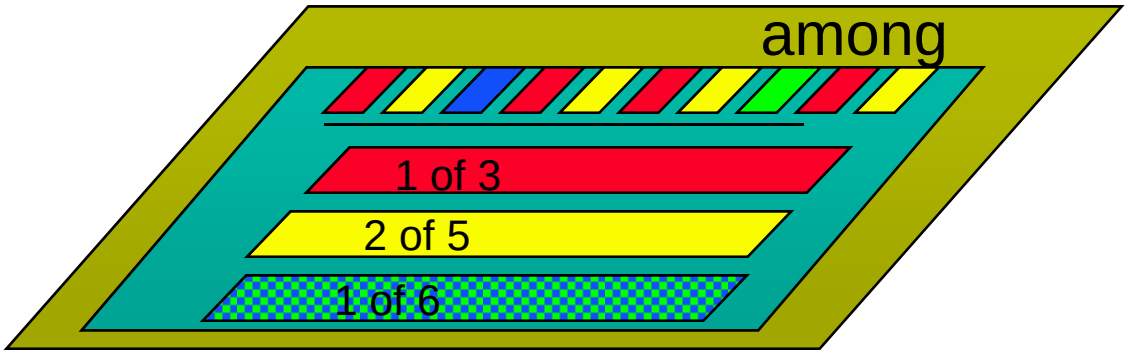


Global constraints

- Work on sets of variables
  - Global conditions, not local constraints
- Semantic methods
  - Operations Research
  - Spatial algorithms
  - Graph theory
  - Network flows
- Building blocks (high-level constraint primitives)
  - Multi-purpose
  - As general as possible
  - Usable with other constraints
  - Very strong propagation
  - Acceptable algorithmic complexity

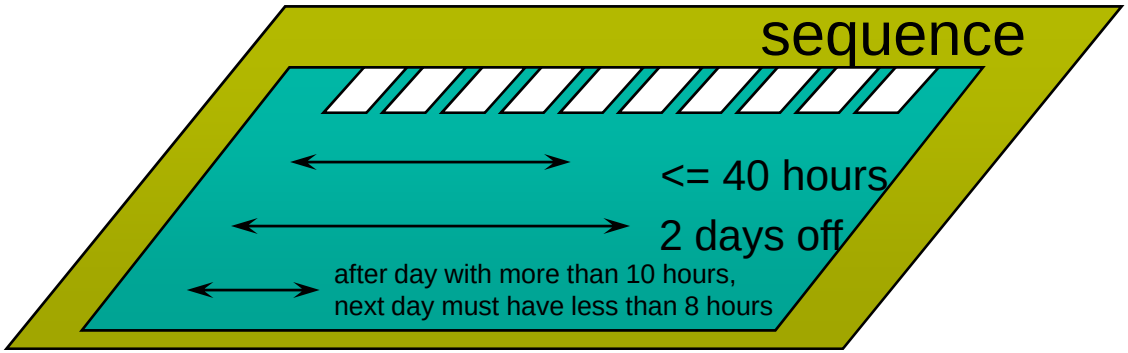


The Among global constraint (CHIP)

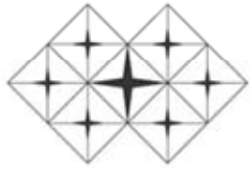


- Among constraint
  - How often do values occur in (sub)sequences
  - based on counting arguments
  - network flow
  - interaction between sequences
- Applications
  - production sequencing, time tabling, coloring problems, set covering

The Sequence global constraint (CHIP)

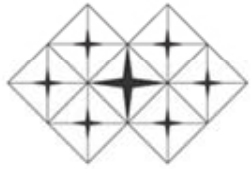


- Sequence constraint
  - constraints on pattern inside sequences
  - combinatorial pattern matching
  - counting arguments
- Applications
  - Time tabling, personnel assignment,
  - work rules, scheduling with daily working time limits



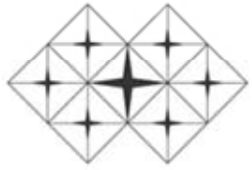
Model

- Constraints on day
  - Restrict the number of times a value is chosen on a day
  - atmost(N,[X1,X2..Xn],[V],all)
    - The value V is taken N times among the variables X1..Xn
    - N fixed in this case by demand profile
    - Propagation event all: propagate whenever possible
  - Redundant among constraints
    - combinations of work pattern
    - rest days required (for perfect problems)
- Constraints on persons
  - among(1,6,6,4,4,[X1,X2..Xk],[0],all),
    - in any sequence of 6 days, at least 1 and at most 6 days off
    - at least and at most four days off altogether
  - among(0,2,5,4,4,[X1,X2..Xk],[0],all),
    - in any sequence of 5 days, at least 0 and at most 2 days off



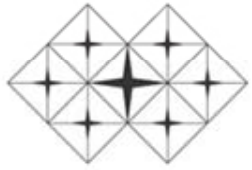
Model

- Constraints on person (cont'd)
  - **among(0,3,4,0,4,[X1,X2..k],[3],all)**
    - **in any sequence of four days, at most 3 night shifts**
    - **altogether at most 4 night shifts**
  - **sequence([0,0,2],[X1,X2..k],[[[sum,1,#=,[3]],[sum,1,#=,[1]]]],all)**
    - **in any sequence of two consecutive variables, there must at least 0 and at most 0 (never) be a pattern which starts with one variable having value 3 (= Night) followed by one variable having value 1**
  - **sequence([0,0,3],[X1,X2..k],**
    [[[sum,1,#=,[3]],[sum,1,#=,[1,2]],[sum,1,#=,[3]]],
    [[[sum,1,#=,[0]],[sum,1,#=,[1,2,3]],[sum,1,#=,[0]]]],all)
    in any sequence of three variables there can not be a pattern of either the form
    N[MA]N
    nor of the form
    R[MAN]R



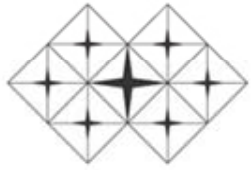
Problems

- Which global constraints?
  - Balance between
    - expressiveness
    - reasoning power
    - implementation effort
- How to implement them?
  - Use system which has them as primitives
    - CHIP, Solver
  - Roll your own
    - see Bourdais2003 for algorithms



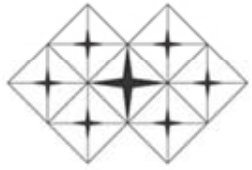
Remaining question(s)

- How to assign values (search part)
  - Very important for stability and efficiency
  - See case study
- How to improve propagation



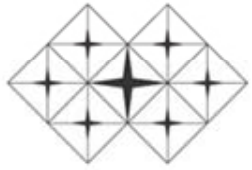
Soft constraints (Meyer auf'm Hofe 97, Abdennadher 99)

- Depending on rule set, problem may be infeasible
  - 'no' is not an answer
- One man's constraint is another man's preference
- Stratification of constraints
  - systematic
  - legal
  - fairness
  - preference
- Not all users agree which constraints belong to which category



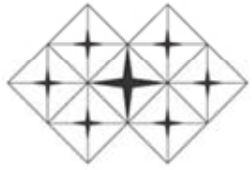
Problems

- Hard to propagate information if you don't know which constraints to use
- Global constraints require hard constraints
  - but: new work by Meyer auf'm Hofe
- How to compare two solutions which violate different types of constraints
  - weights and levels somewhat arbitrary

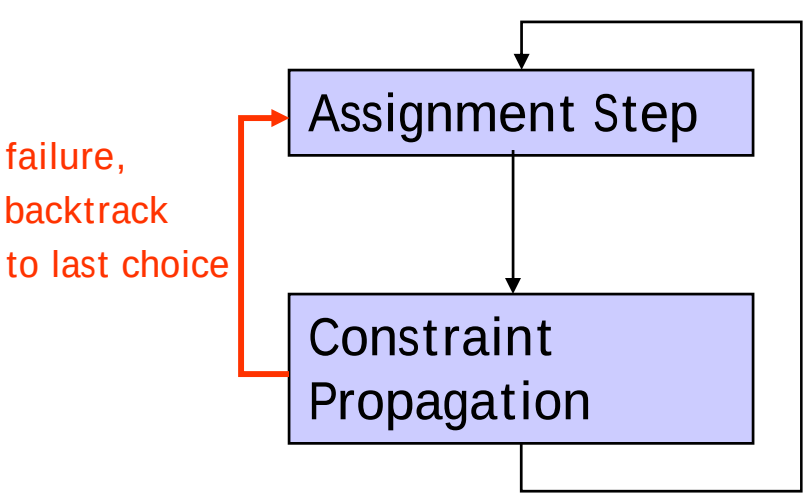


User Interaction (Weil, Heus 95)

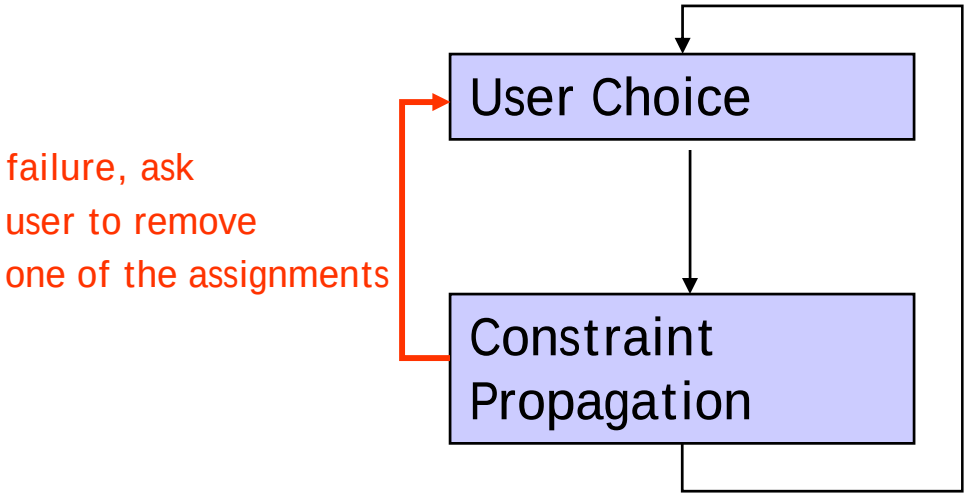
- Staff satisfaction is major factor in
  - staff retention
  - decrease in absenteeism
- Important to take wishes of staff into account
  - difficult to achieve with black-box solver
  - wishes can not be hard constraints
- Power balance in hospital
  - who controls the roster generation
  - negotiation tool for supervisor



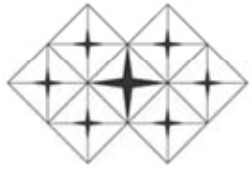
Solver Paradigm



Black box solver

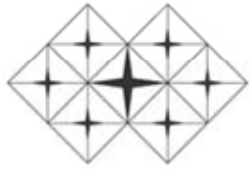


Interactive solver



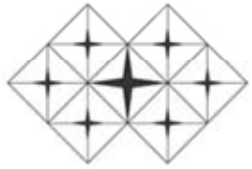
Problems

- Propagation must be very good
  - failure detection
  - user experience
- User interface must present state of solver
  - domain representation
  - domain size
- Backtracking not always chronological
  - solver support
- Solver must support multi-threading
  - problem in some languages



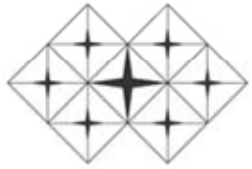
User driven specification (Chan et al, 1998)

- Rule sets constantly evolving
  - regulations
  - work load
- Fixed programs not flexible enough
- User wishes to add/disable/remove constraints
  - temporary relaxation of rules in crisis
  - special rules for some staff
- Constraints become part of user interface



Example: Gymnaste

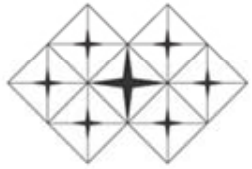
- Rule editor to define new constraints
  - based on near-natural syntax
  - graphical user interface
- Enable/Disable flag for each constraint
  - handling as soft constraints
  - preference
  - apply to certain persons only
- Must match flexible constraint language in system
  - global constraint
  - generator of primitive constraints



Case study

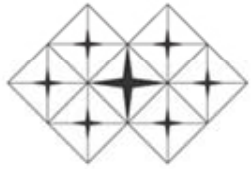
Problem description

- 3 data sets of 100 randomly generated demand profiles
- Rule set as defined above
- 10 persons/14 days
- perfect problems, no spare capacity
- Set X: demand per day 7..8 persons
  - demand per shift 1..4
  - one problem unsolvable
- Set Y: demand per day 6..9 persons
  - demand per shift 1..4
  - one problem unsolvable
- Set Z: demand per day 5..10 persons
  - demand per shift 1..4
  - at least 4 problems unsolvable



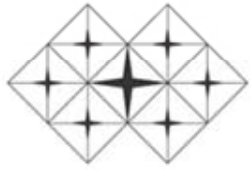
Search strategies

- Combination of several concepts
  - variable selection
  - value selection
  - search method
- Systematic analysis of combinations
- No custom strategy used
  - depending on demand
  - depending on already assigned values



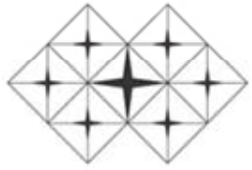
Variable selection

- static by line
  - top to bottom, left to right
- static by column
  - left to right, top to bottom
- most constrained
  - variables sorted by column
- static order by demand and most constrained
  - sort variable columns by decreasing demand
  - use most constrained on this list of variables
  - labeling starts in “difficult” part



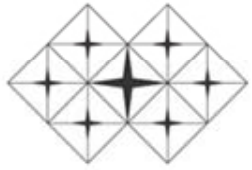
Value selection

- Only 4 possible values
- Perfect problems
- Static order
  - indomain
- Randomized order
  - random starting point for indomain
  - values not independent
- Domain splitting
  - enforce a rest day (value 0) or a working day (values 1,2,3)
  - use of disequality inside labeling
  - default in some CLP systems



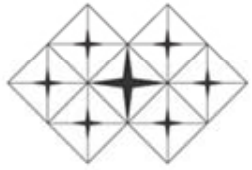
Search method

- Full search
  - limited by number of backtrack allowed (5000)
- Partial search
  - quadratic credit
  - 5 backtrack in local search
  - also limited by number of backtrack (5000)



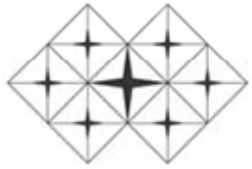
Strategies

- Static variable order, full search
  - A: by line
  - B: by line, randomized
  - C: by column
  - D: by column, randomized
- Dynamic variable order, full search
  - E: most constrained
  - F: most constrained, randomized
  - G: most constrained on demand order
  - H: most constrained on demand order, randomized



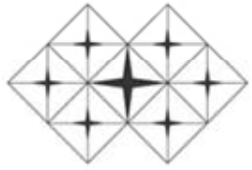
Strategies 2

- Dynamic variable order, partial search
  - I: most constrained
  - J: most constrained, randomized
  - K: most constrained on demand order
  - L: most constrained on demand order, randomized
- Static variable order, partial search
  - M: by line
  - N: by line, randomized
  - O: by column
  - P: by column, randomized
- Dynamic order, partial search, domain splitting
  - Q: most constrained
  - R: most constrained on demand order



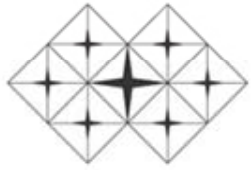
Evaluation

- Count the number of problems solved in each data set
- Randomized solver only run once
- All tests use the same number of choices (5000)



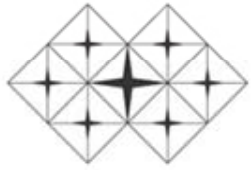
Success rate

		1000 backtracks			5000 backtracks		
	Strategy	Data set 1 demand 7..8 per day	Data set 2 demand 6..9 per day	Data set 3 demand 5..10 per day	Data set 1 demand 7..8 per day	Data set 2 demand 6..9 per day	Data set 3 demand 5..10 per day
A	by line	0	0	0	0	0	0
B	by line, randomized	0	0	0	1	2	0
C	by column	9	16	11	9	18	13
D	by column, randomized	44	38	34	41	38	39
E	Most constrained	86	67	39	86	71	43
F	Most constrained, randomized	57	46	26	45	39	42
G	static sort, most constrained	57	46	39	63	53	41
H	static sort, most constrained, randomized	60	52	32	63	50	34
I	partial search, most constrained	98	90	69	99	97	83
J	partial search, most constrained, randomized	97	94	70	99	97	85
K	partial search, static sort, most constrained	98	90	63	99	96	77



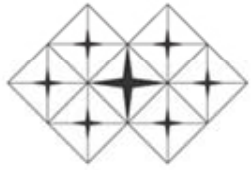
Success rate, part 2

L	partial search, static sort, most constr, randomized	98	94	69	99	99	85
M	By line, partial search	0	0	0	0	0	0
N	By line, partial search, randomized	4	2	1	20	12	5
O	By column, partial search	13	13	12	21	18	13
P	By column, partial search, randomized	93	87	71	99	96	86
Q	Most constrained, partial search, R-MAN split	98	92	72	99	97	86
R	Most constrained, static order, partial search, R-MAN split	99	92	68	99	96	80

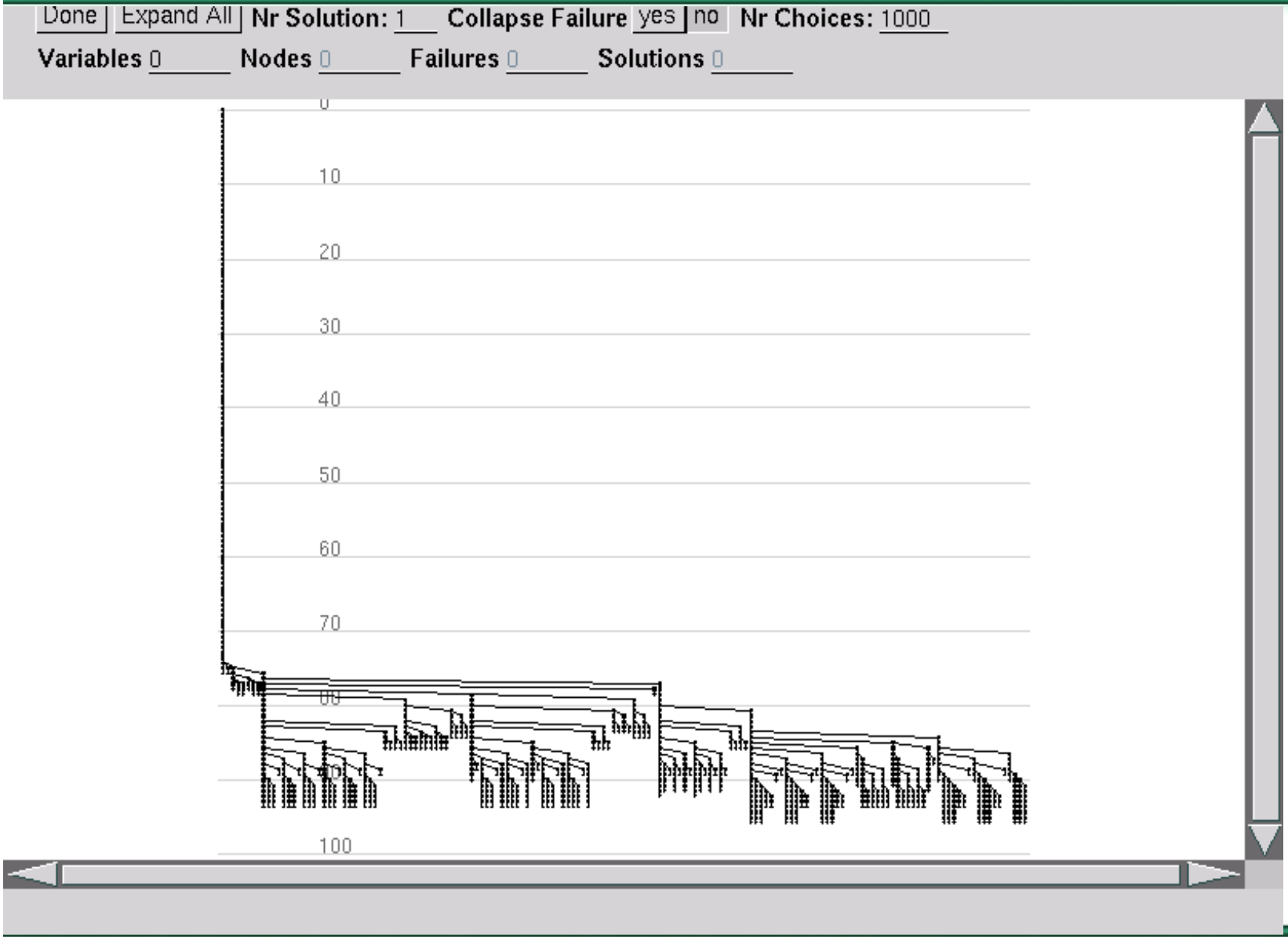


Search performance

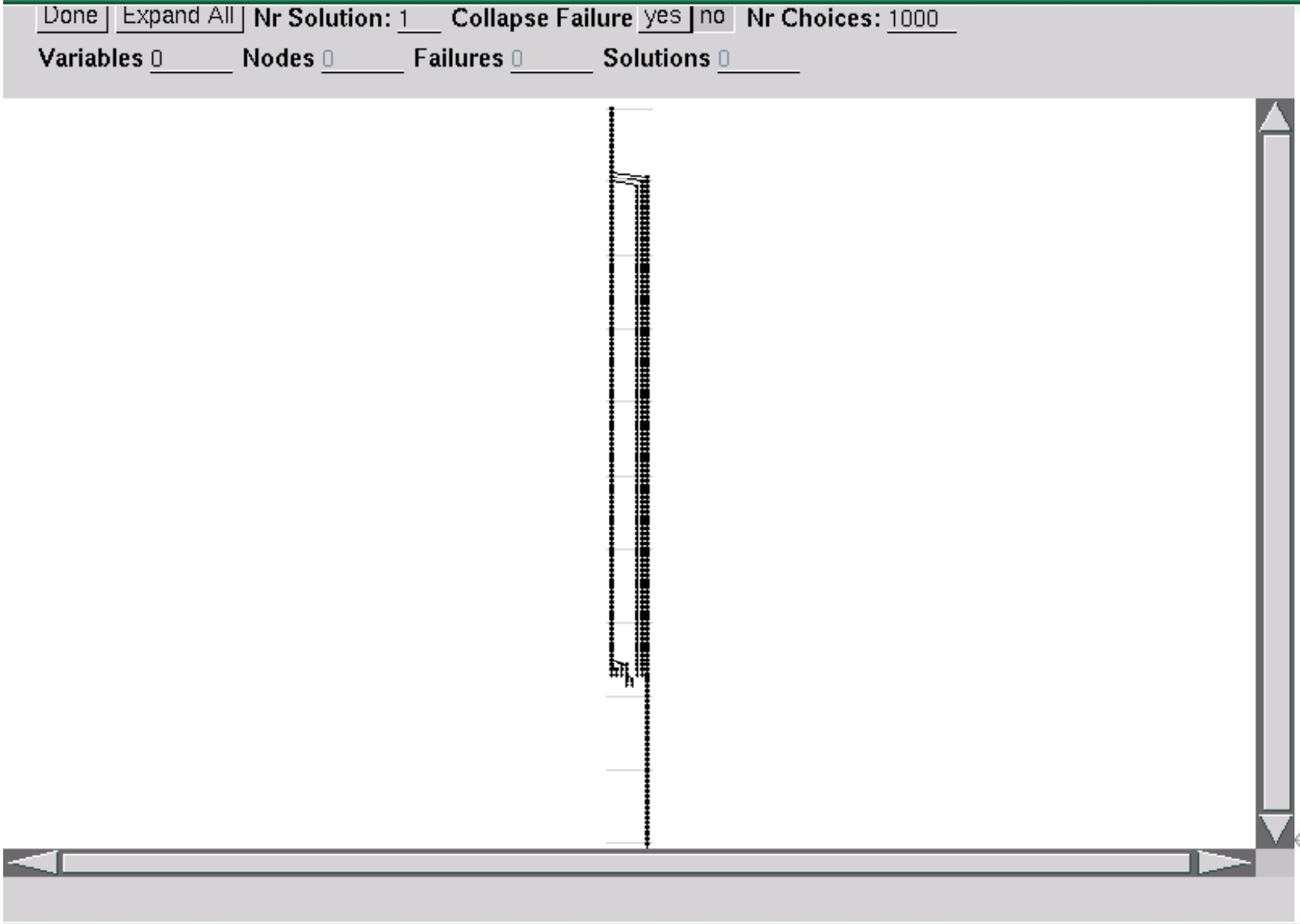
- Search tree analysis for full and partial search
  - strategy E: most constrained
  - strategy I: most constrained and partial search
- Plotting number of choices needed to find solutions
  - x-axis: number of backtracks allowed
  - y-axis: number of problems solved
  - different plots for data sets X, Y and Z
  - shown for 1000 and 5000 backtrack horizon



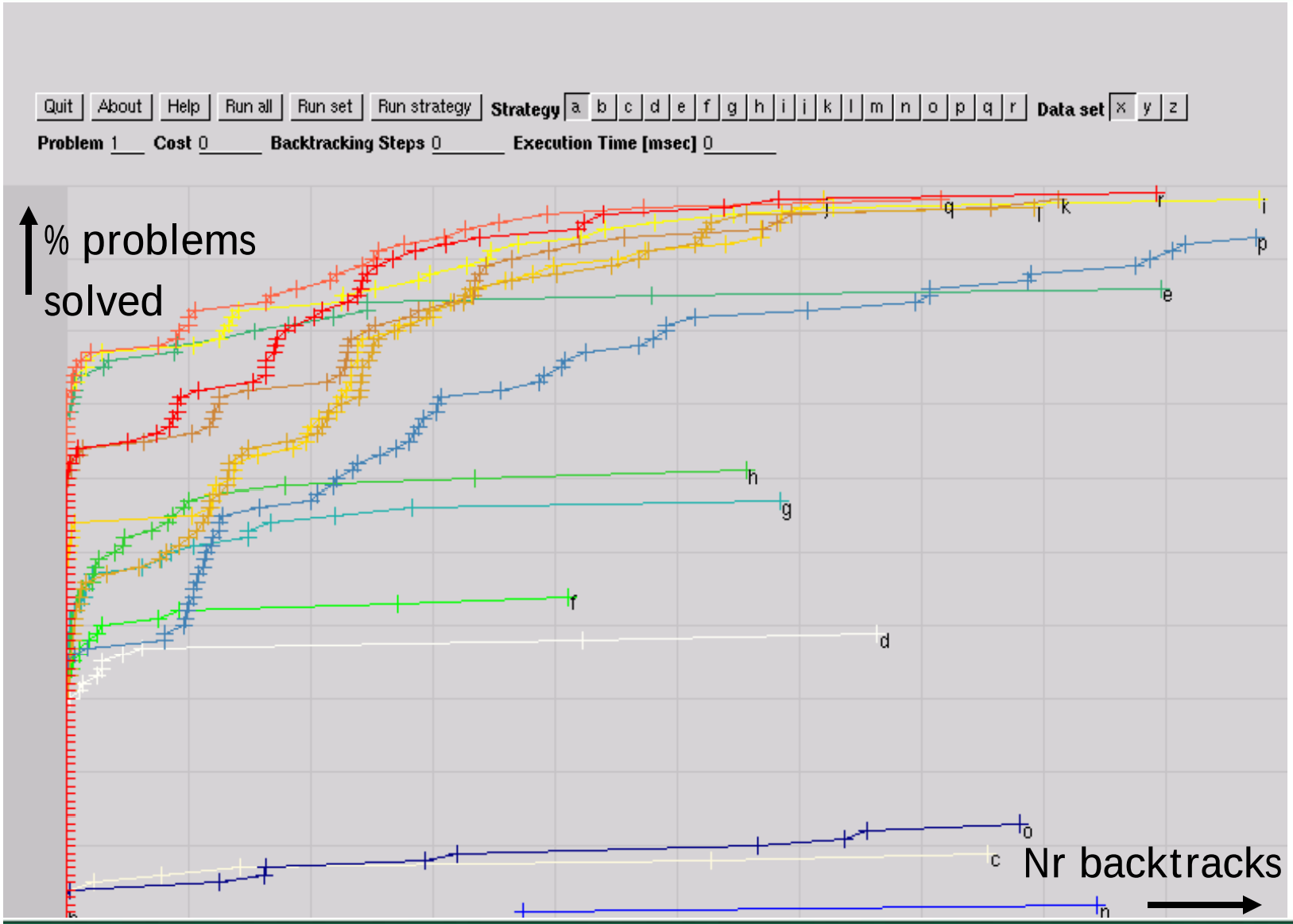
Complete search failure, late backtracking



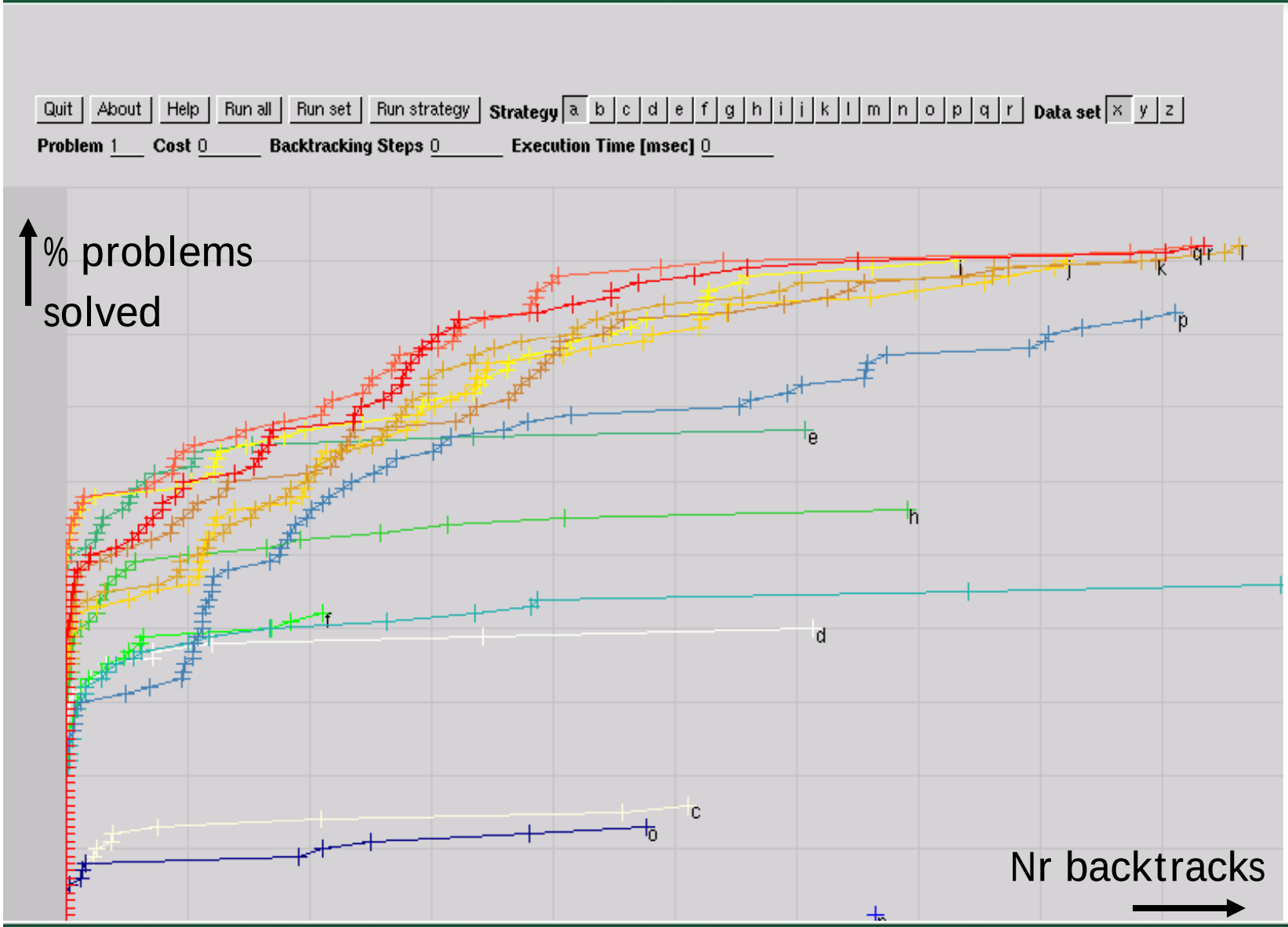
Partial search success, early restart



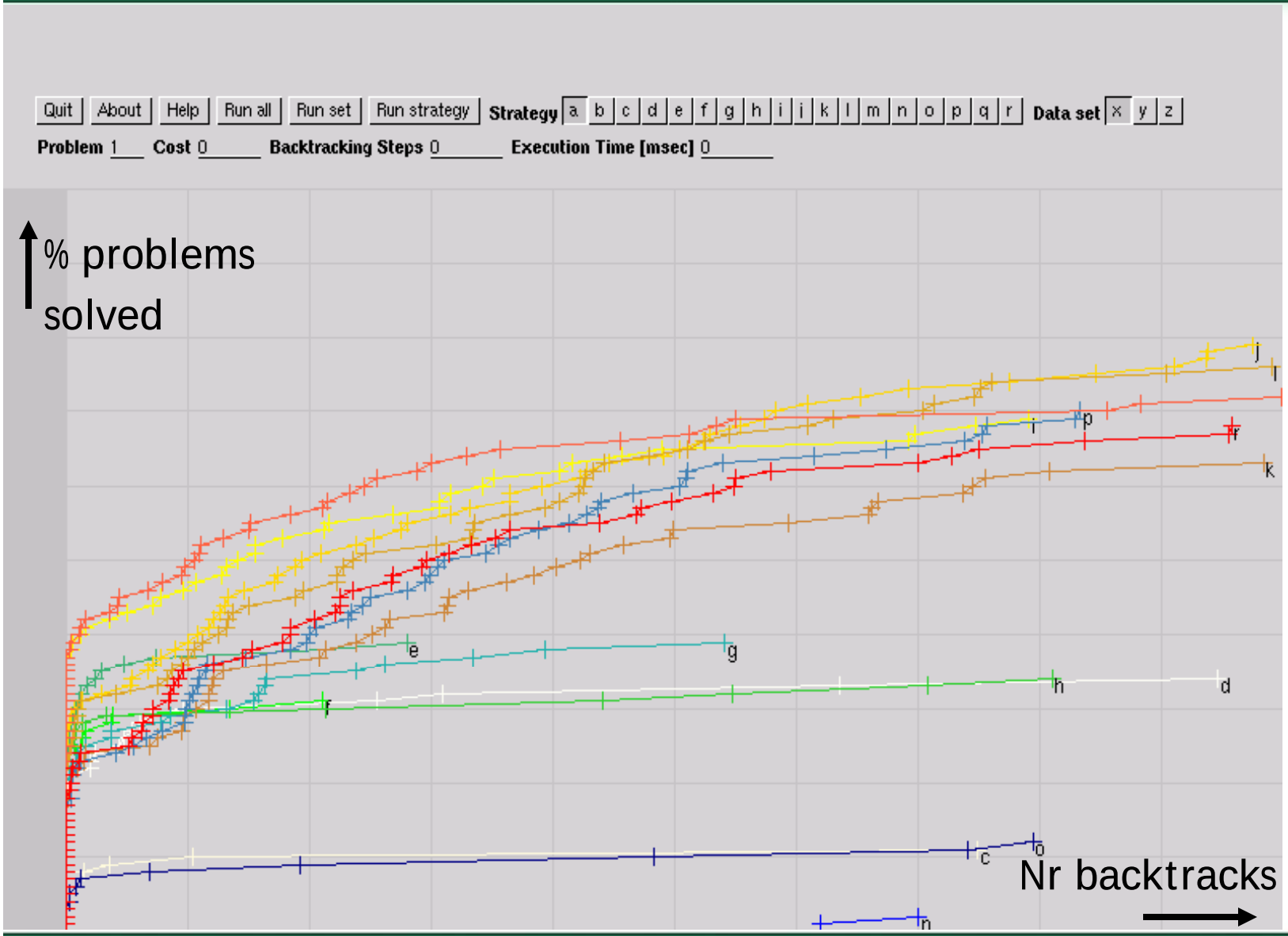
Data set X, backtrack plot,1000 choices



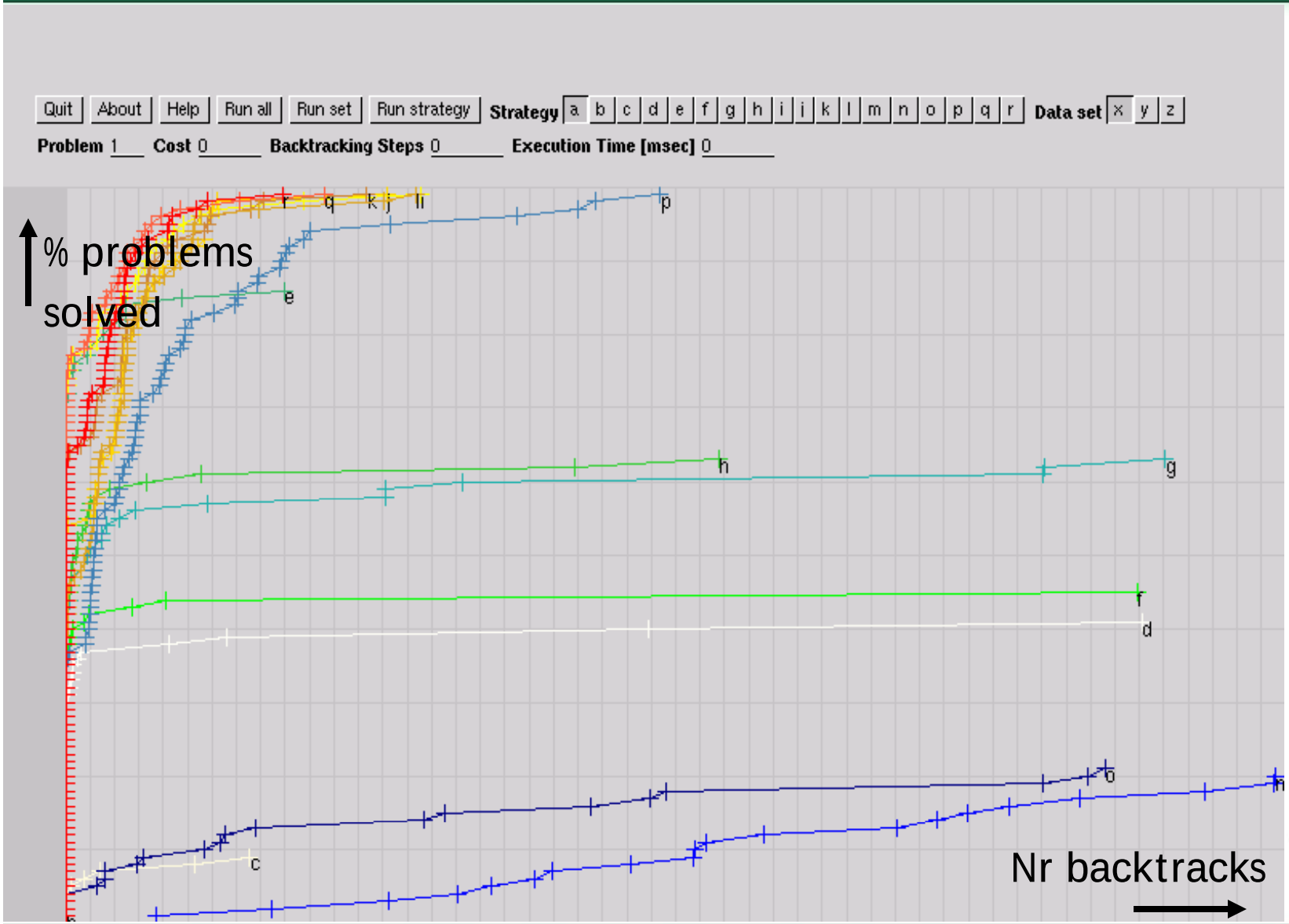
Data set Y, backtrack plot,1000 choices



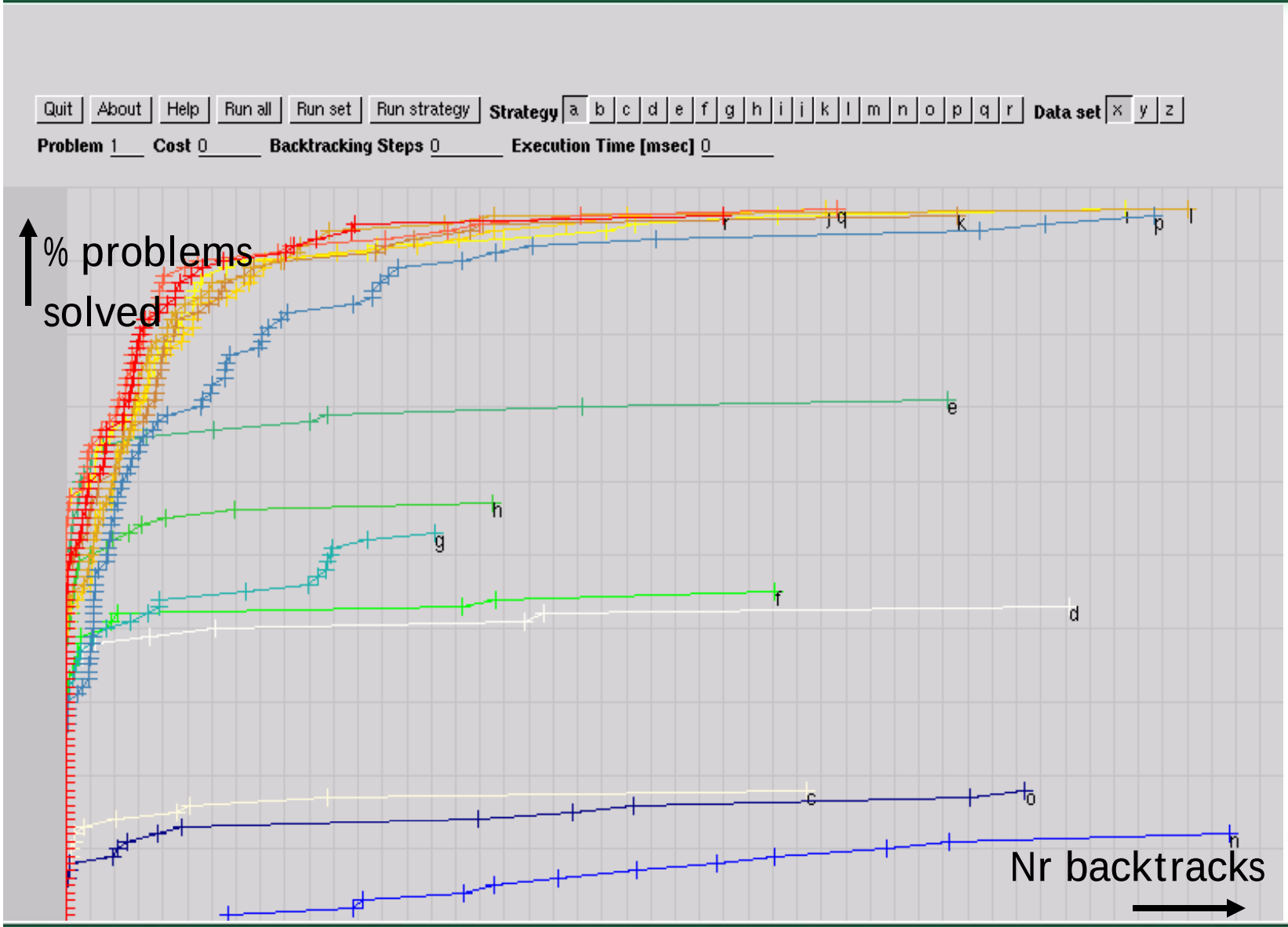
Data set Z, backtrack plot,1000 choices



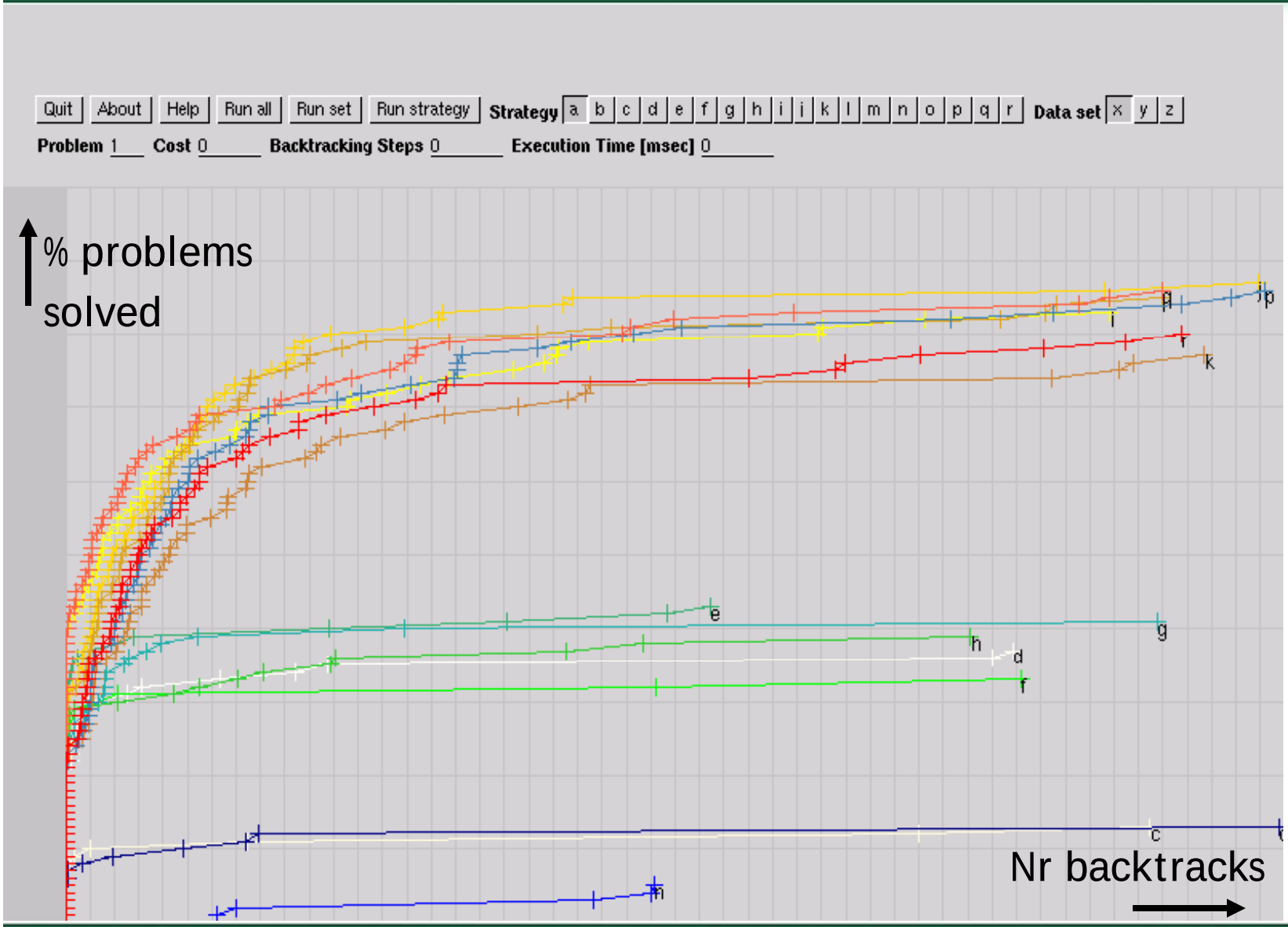
Data set X, backtrack plot,5000 choices



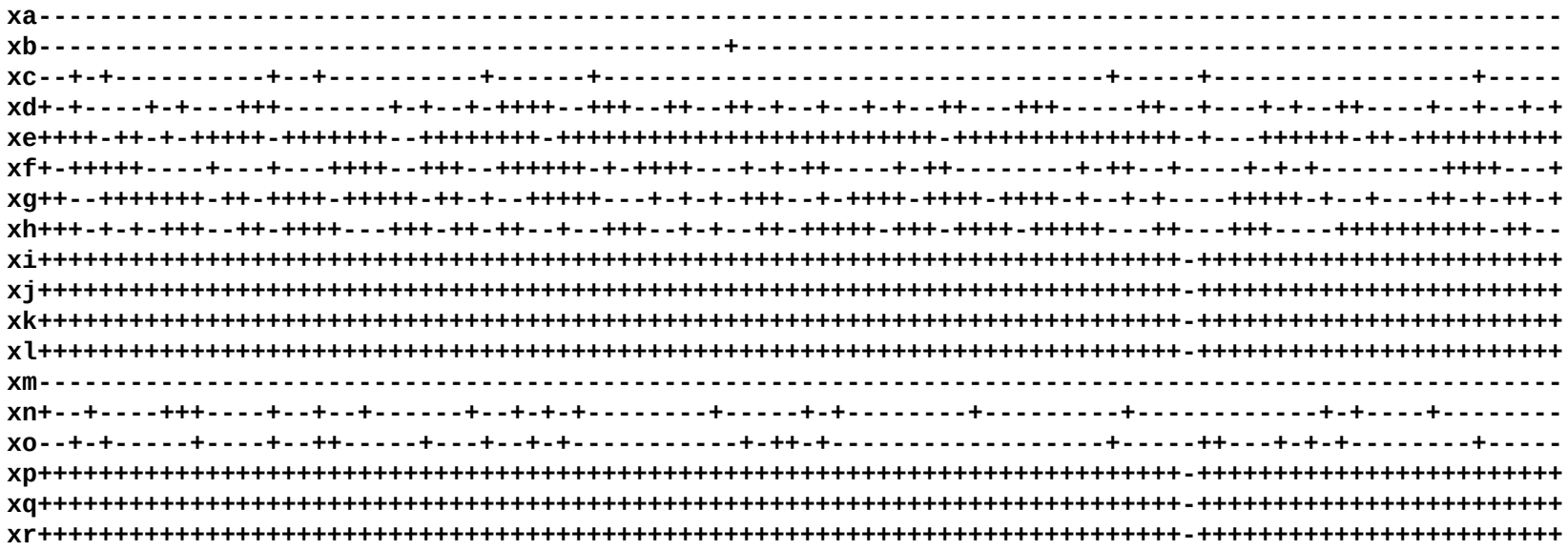
Data set Y, backtrack plot,5000 choices



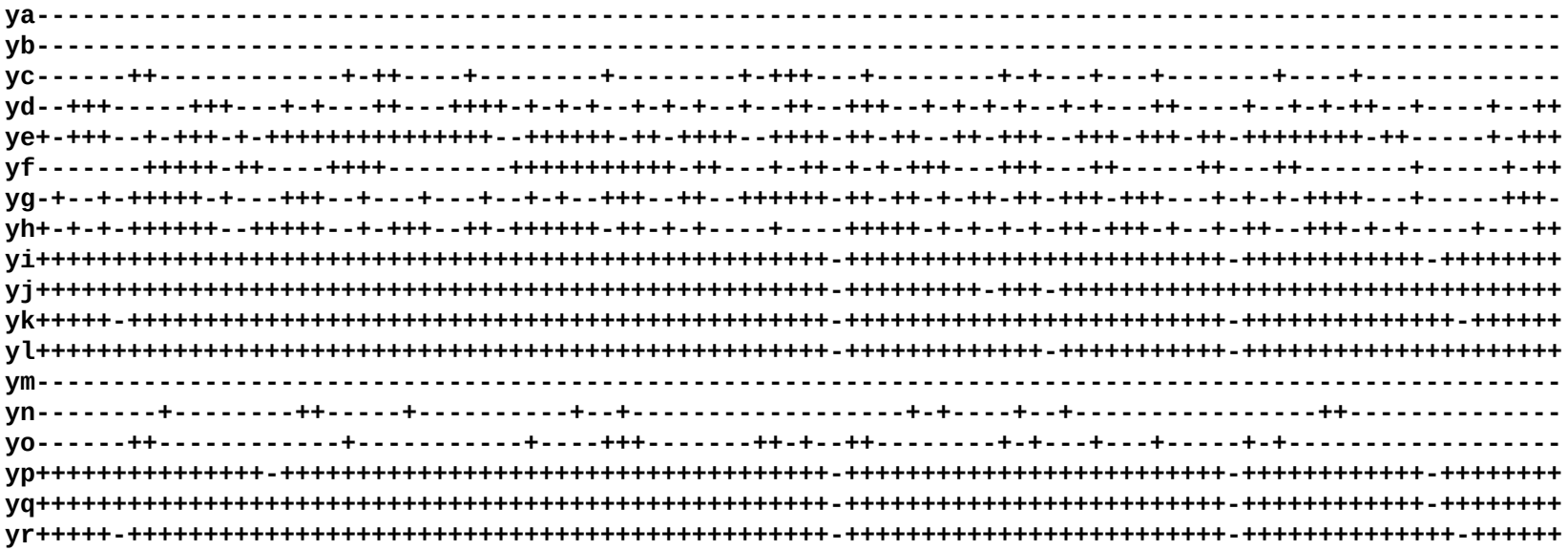
Data set Z, backtrack plot,5000 choices

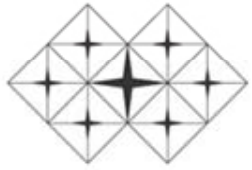


Solutions found, data set X

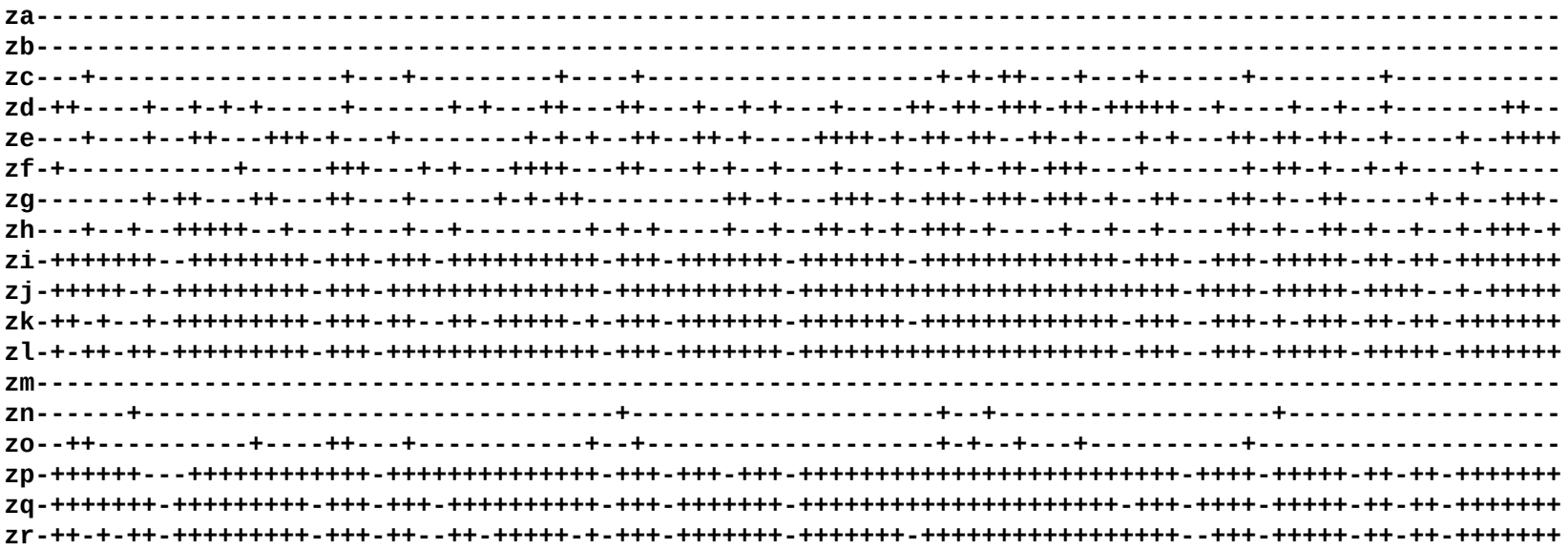


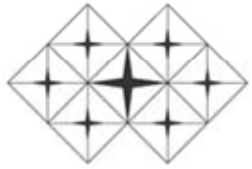
Solutions found, data set Y





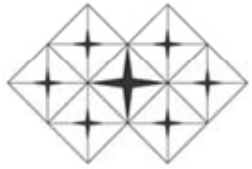
Solutions found, data set Z





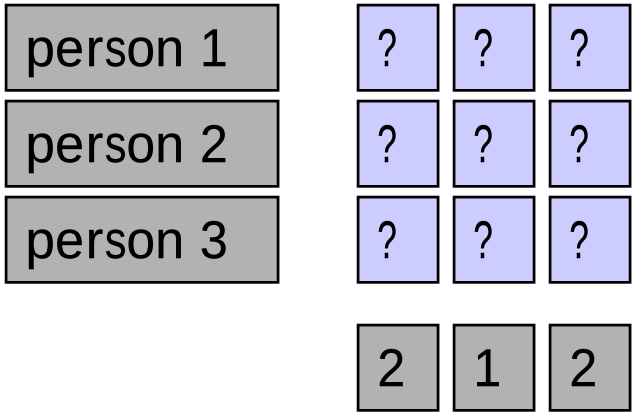
Evaluation

- Rather complex constraints easily expressed
- Propagation still limited
  - does not detect inconsistency in some simple cases
  - constraints on lines and columns do not interact enough
  - needs redundant constraints for better results
- Search strategy plays important role
  - no single best method
  - combination of different methods achieves near 100% results
- Partial search very significant improvement
  - only if strategy is already working
- Benefit of increased search visible for some strategies
- Effect of demand profile clearly visible

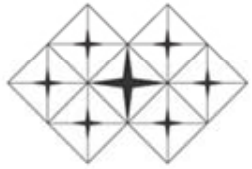


Missed propagation

- Interaction row/column constraints
  - each person should have two days of type M
  - demand for type M on days 2,1,2
  - no spare capacity
  - easy to see that this is infeasible
    - but requires global view
    - interaction of constraints

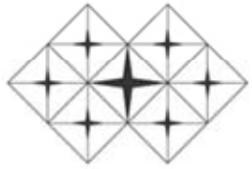


  - can be solved with LP
    - requires hybrid solver



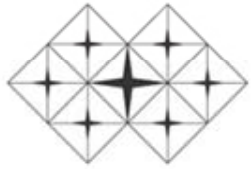
References

- S. Darmoni et al. Horoplan: computer-assisted nurse scheduling using constraint-based programming. Twelfth International Congress of the European Federation for Medical Informatics, 1994, Lisboa, Portugal.
- G. Weil, K. Heus, F. Puget, M. Poujade. Solving the Nurse Scheduling Problem Using Constraint Programming, IEEE Engineering in Medicine and Biology, July-August 1995.
- B. Cheng, J. Lee, J. Wu. Speeding Up Constraint Propagation By Redundant Modelling, CP96, Cambridge, USA.
- H. Simonis. Standard Models for Finite Domain Constraint Solving. Tutorial PACT 97, London, UK.
- H. Meyer auf'm Hofe. ConPlan/SIEDAplan: Personnel Assignment as a Problem of Hierarchical Constraint Satisfaction, PACT97, London , UK.
- P. Chan, K. Heus, G. Weil. Nurse Scheduling with Global Constraints in CHIP: GYMNASTE, PACT 1998, London , UK.
- S. Abdennadher, H. Schlenker. INTERDIP – An Interactive Constraint Based Nurse Scheduler, PACLP 1999, London, UK.
- A. Chun et al. Nurse Rostering at the Hospital Authority of Hong-Kong. AAAI Innovative Applications of Artificial Intelligence, 2000.
- P. Chan, G. Weil. Cyclical Timetabling Using Constraint Logic Programming, PACLP 2000, Manchester, UK.
- P. Chan. La planification du personnel: acteurs, actions et termes multiples pour une planification operationelle des personnes, PhD Thesis Universite Joseph Fourier, Grenoble 1. TIMC-IMAG, Octobre 2002.
- S. Bourdais, P. Galinier, G. Pesant. HIBISCUS: A Constraint Programming Application to Staff Scheduling in Health Care, CP2003, Kinsale, Ireland.



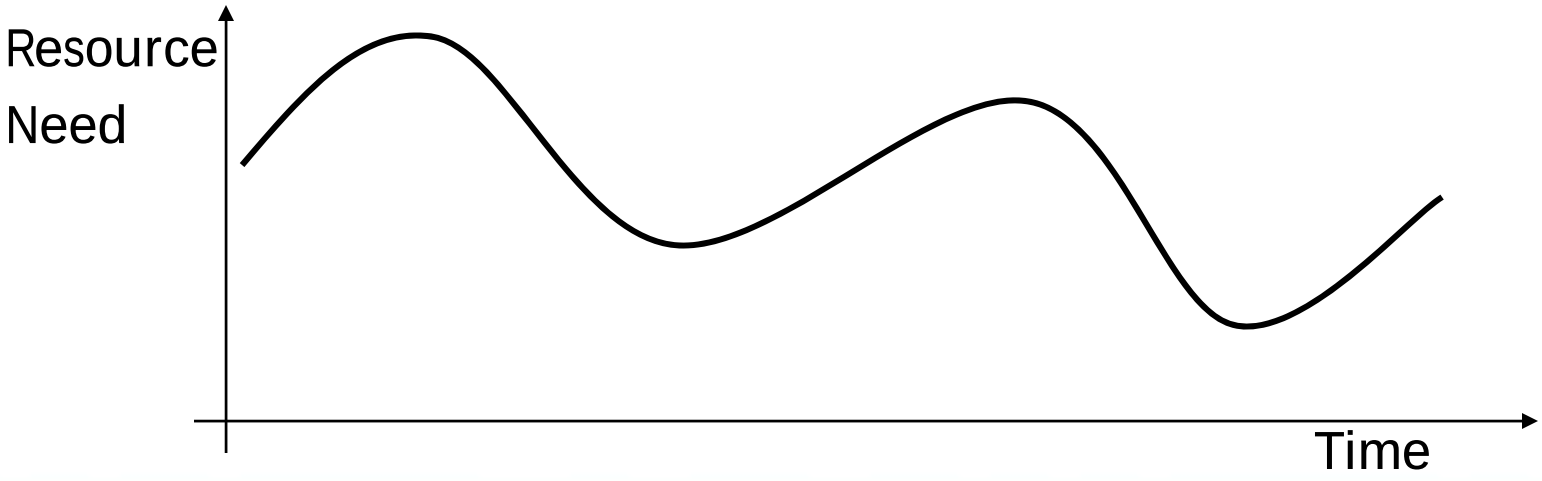
But…

- Are we solving the right problem?
- Scale
  - solving the whole hospital rostering in one go
  - assumes that personnel can be moved between wards
  - may be difficult to implement
  - large possible gains in resource utilisation
- Other views of personnel allocation



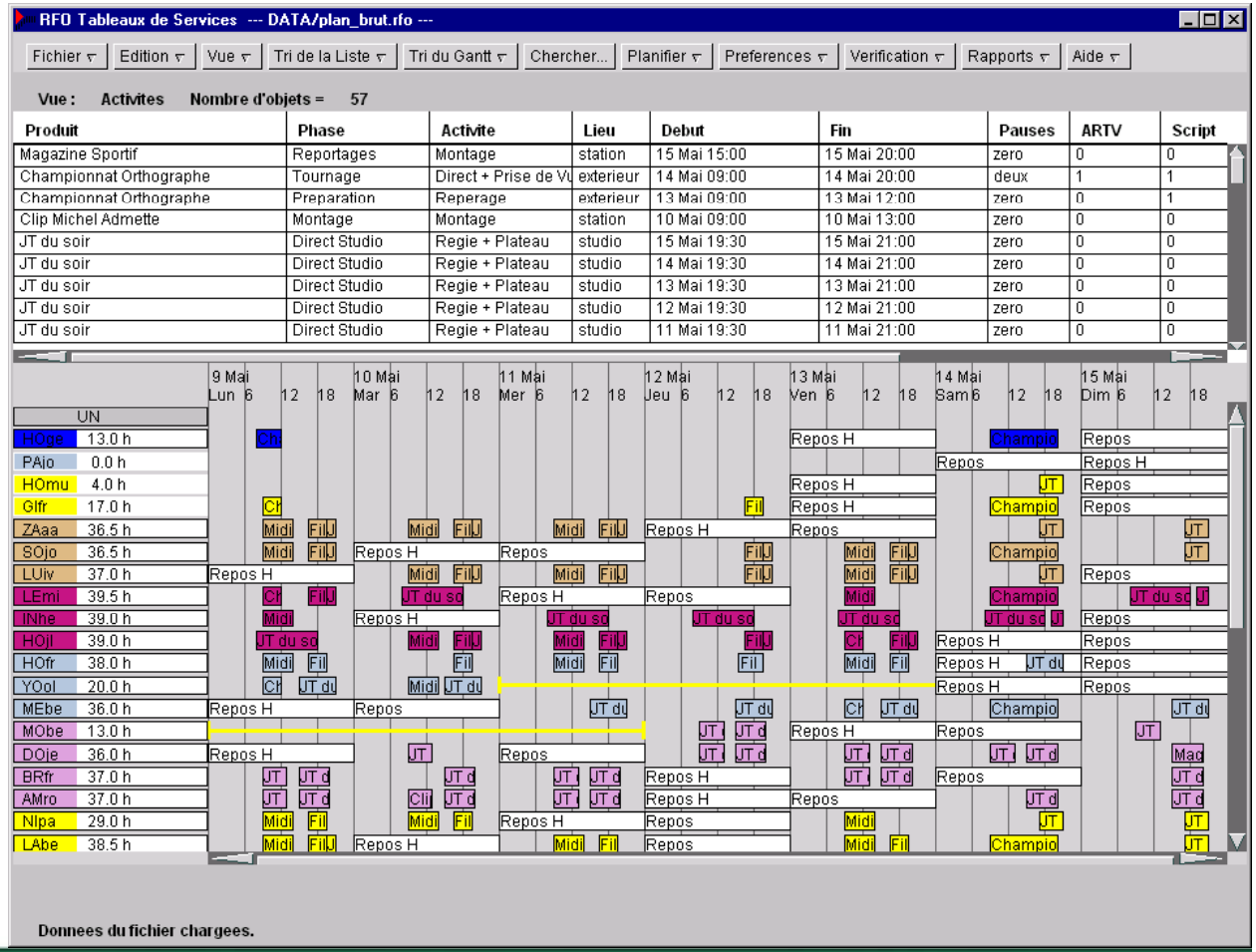
Workload matching

- Cover a demand profile with personnel starting work at different times
- Typical: call center staffing
- Existing application:
  - Carte bleue call center (ATOS, France)



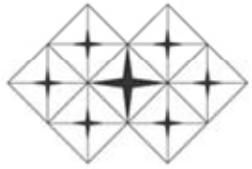
Task coverage

- Assign personnel to cover tasks
  - tasks fixed in time
  - required qualifications
- Existing application OptiService (COSYTEC, France)
  - for TV stations (RFO, TSR, Canal+)
  - cover all tasks (features, news, editing, transmission)
  - hierarchy of qualifications
  - planning period 1 month – 1 year



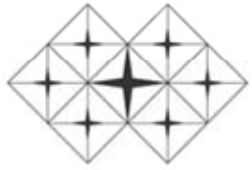
Yearly work-balance (Chan 2002)

- Allocate working time based on yearly working time limits
  - total working time: N hours, K days off, L weekend off
  - daily demands: M hours, P persons
- Hierarchical problem decomposition
  - yearly plan
  - monthly work load
  - monthly roster
  - important to plan ahead
- Existing application (COSYTEC)
  - supermarket checkout personnel
  - triggered by 35 hour week legislation in France



Products based on Constraints

- COSYTEC: Gymnaste
  - http://www.cosytec.com/constraint_programming/cases_studies/health_care.htm
- COSYTEC: MOSAR
  - http://www.cosytec.com/constraint_programming/cases_studies/administration.htm
- EquiTime: EquiTime, EffiTime
  - http://www.equitime.fr
- SIEDA: Dienstplan
  - http://www.sieda.com



Summary

- Constraint Programming is useful basis for rostering
  - expressive
  - powerful
- Emphasis on different aspects leads to different solutions
- Several commercial rostering applications based on CP