An ant colony optimization inspired algorithm for the Set Packing Problem with application to railway infrastructure

Xavier GANDIBLEUX[1], Julien JORGE[1], Sébastien ANGIBAUD[1]
Xavier DELORME[2] and Joaquin RODRIGUEZ[3]

(1) LINA - Laboratoire d'Informatique de Nantes Atlantique
Université de Nantes
2 rue de la Houssinière BP92208, F-44322 Nantes cedex 03 – FRANCE
Xavier.Gandibleux@univ-nantes.fr, {jorge,angibaus}@ensinfo.univ-nantes.fr

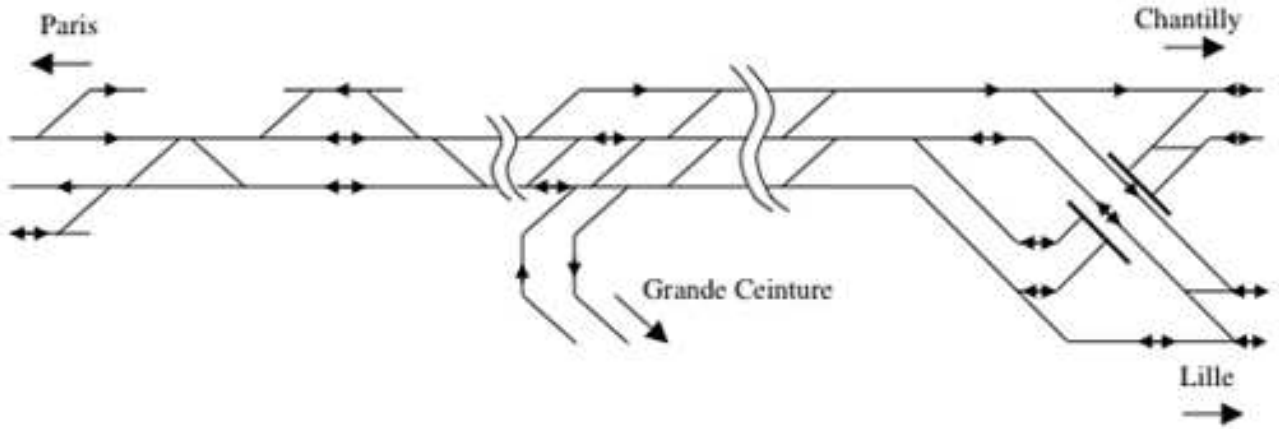
(2) Centre Génie Industriel et Informatique
Ecole des Mines de Saint-Etienne
158 cours Fauriel, F-42023 Saint-Etienne cedex 2 – FRANCE
delorme@emse.fr

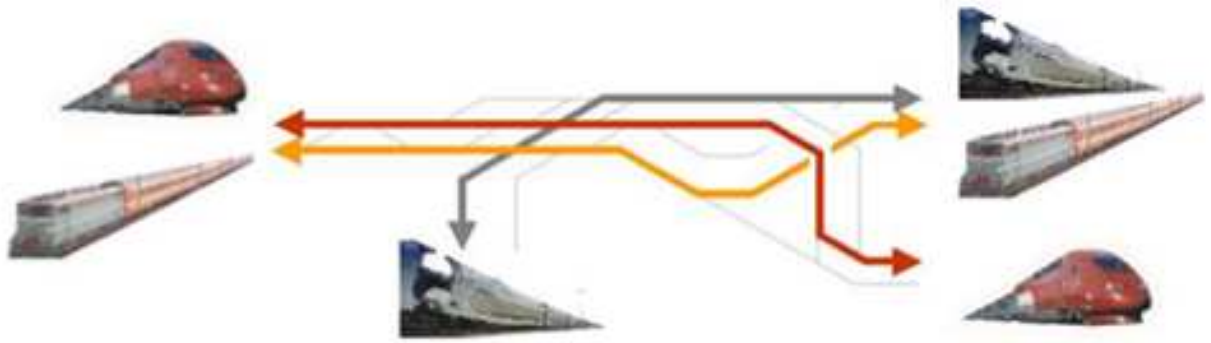
(3) INRETS - Institut National de Recherche sur les Transports et leur Sécurité
ESTAS - Évaluation des Systèmes de Transports Automatisés et de leur Sécurité
20 rue Élisée Reclus, F-59650 Villeneuve d'Ascq – FRANCE
joaquin.rodriguez@inrets.fr

MIC2005
The Sixth Metaheuristics International Conference
Vienna, Austria, August 22–26, 2005

Motivations: A railway planning problem (RPP)

- Planning the construction or reconstruction of infrastructures
- Capacity of one component / junctions of a rail system
- Junction Pierrefitte-Gonesse, north of Paris





The RPP as a Set Packing Problem (SPP)

Given

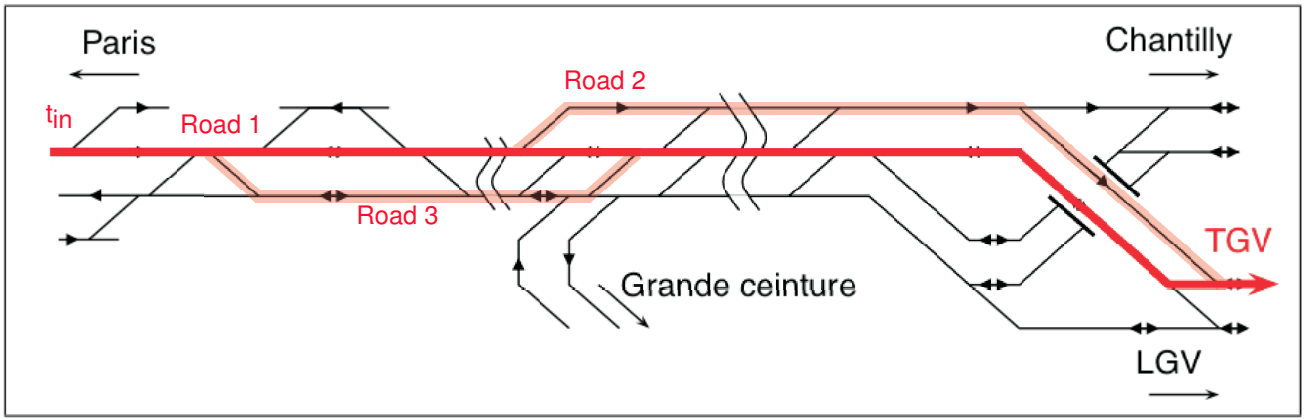
- a finite set $I = \{1, \ldots, n\}$ of items
- $\{T_j\}, j \in J = \{1, \ldots, m\}$, a collection of m subsets of I

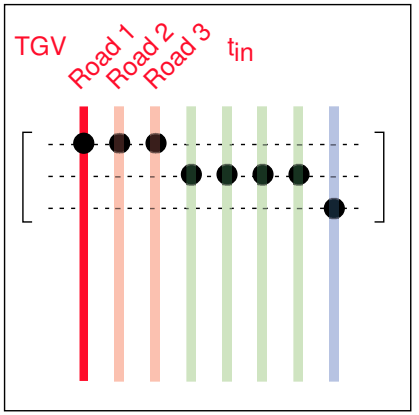
a packing is a subset $P \subseteq I$ such that $|T_j \cap P| \leq 1, \forall j \in J$ which

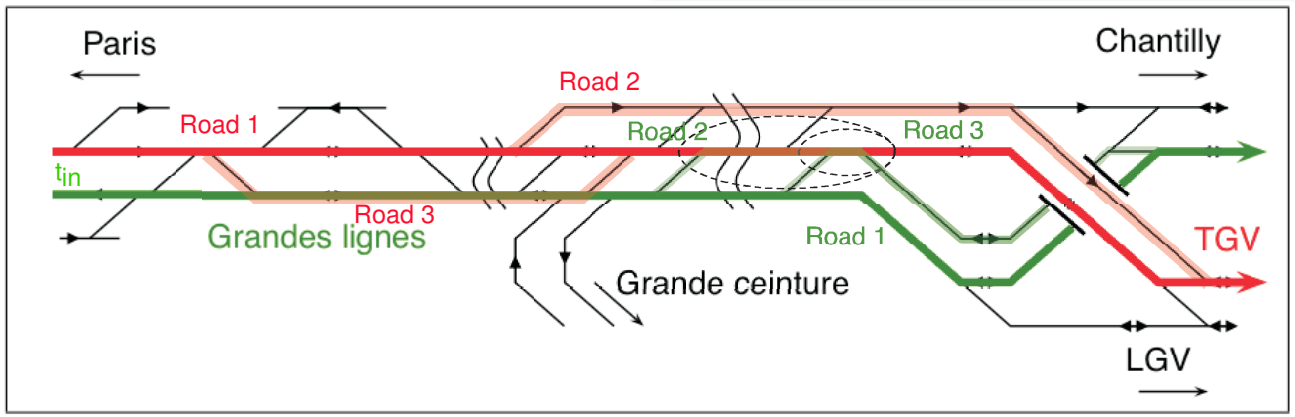
$$\left[\begin{array}{ll} Max\ z(x) = & \sum_{i \in I} c_i x_i \\ & \sum_{i \in I} t_{i,j} x_i \leq 1, \forall j \in J \\ & x_i \in \{0,1\} \quad , \forall i \in I \\ & t_{i,j} \in \{0,1\} \quad , \forall i \in I, \forall j \in J \end{array}\right] \qquad (SPP)$$

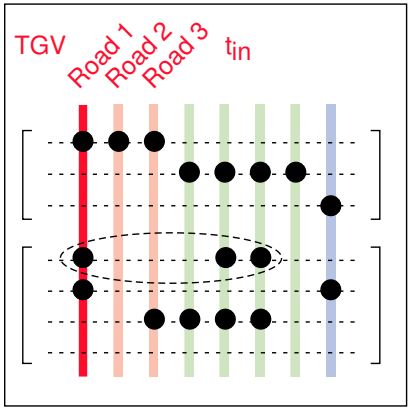
- Strongly NP-Hard (Garey and Johnson 1979)
- Node Packing Problem : $\sum_{i \in I} t_{i,j} = 2, \forall j \in J$

From the RPP to the SPP

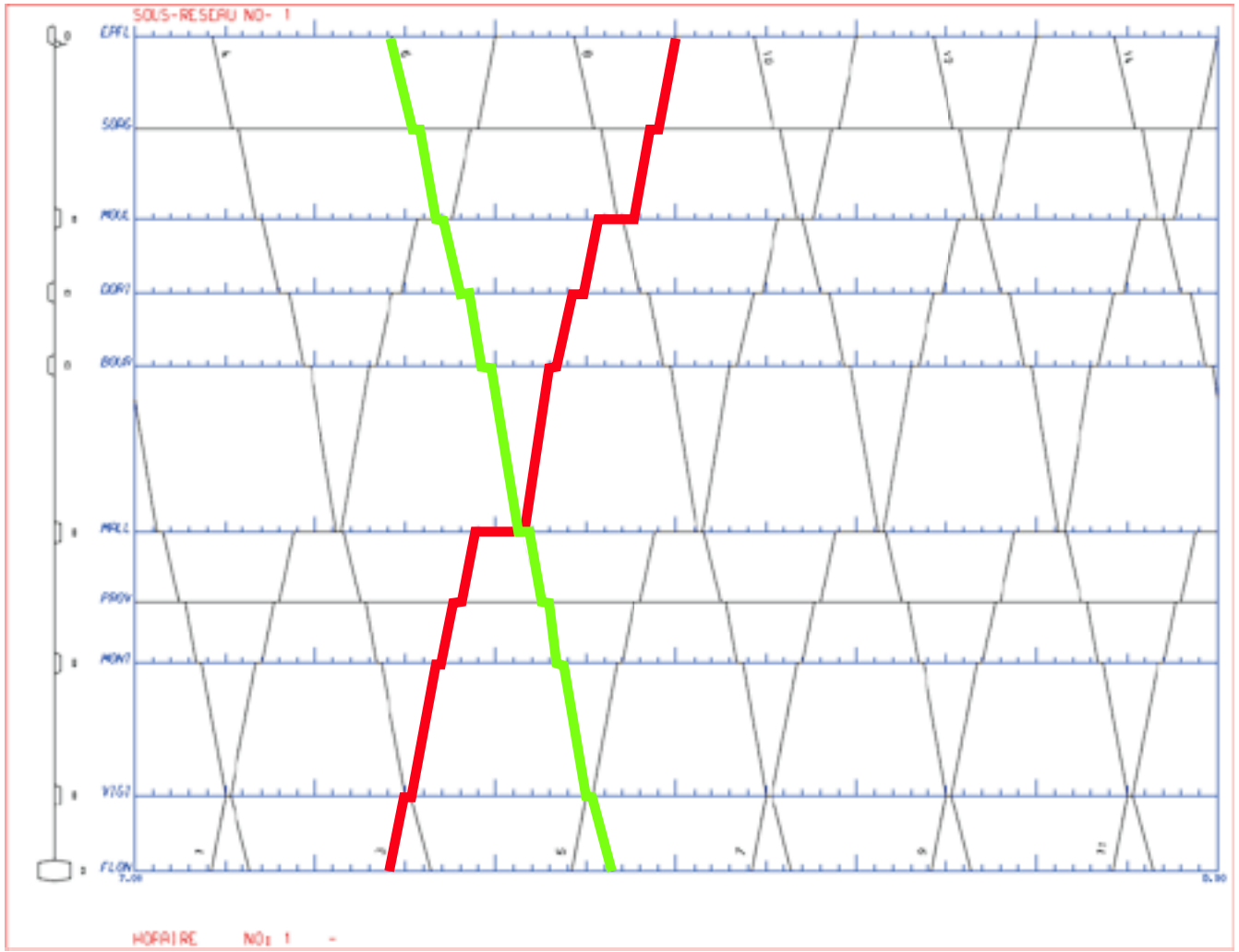








A solution to the RPP



The literature about the SPP

- **Theoretical results and Exact algorithms :**
  - polyhedral theory, facets (Padberg 1973)
  - a Branch & Cut algorithm (Nemhauser 1999, Rossi 2001)
  - a lagrangian relax. with subgradient opt. (Rönnqvist 1995)
  - use a general solver (as LINDO) (Kim 1997)
- **Heuristics and meta-heuristics :**
  - nothing on the market in 2000 and until today (?) except our works
- **Practical applications :**
  - a cutting stock problem (Rönnqvist 1995)
  - a ship scheduling problem (Kim 1997)
  - bounds for a RCPSP (Mingozzi 1998)
  - a ground holding problem (Rossi 2001)
  - a railway planning problem (Zwaneveld 1996, Delorme 2003)

Aim of our investigations

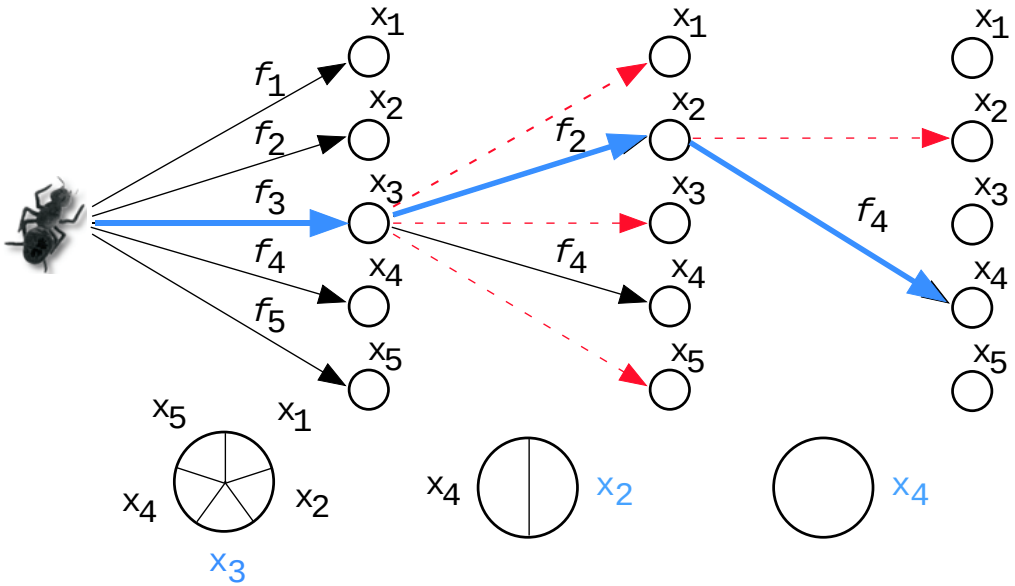
To design efficient algorithms for
solving the large size instances of SPP,
without using the specific structure of the RPP

- **Heuristics and meta-heuristics** :
  - **GRASP** with a path-relinking technique, a learning process and a dynamic tuning of parameters (Delorme 2004)
  - **ACOv1** with a territory disturbance strategy (Gandibleux 2004)

MIC 2005
THE 6TH METAHEURISTICS INTERNATIONAL CONFERENCE
AUGUST 22-26, 2005, VIENNA, AUSTRIA

II. Description and revision of our ACO algorithm(s) for the SPP

ACO for SPP: construction of a solution



x_1	x_2	x_3	x_4	x_5
.	.	.	.	.
1	0	1	0	1
1	1	0	0	0
0	1	0	0	1
0	0	0	1	0

x_1	x_2	x_3	x_4	x_5
0	.	1	.	0
~~1~~	0	1	0	~~1~~
1	1	0	0	0
0	1	0	0	1
0	0	0	1	0

x_1	x_2	x_3	x_4	x_5
0	1	1	.	0
~~1~~	0	~~1~~	0	~~1~~
~~1~~	1	0	0	0
~~0~~	1	0	0	~~1~~
0	0	0	1	0

x_1	x_2	x_3	x_4	x_5
0	1	1	1	0
~~1~~	0	~~1~~	~~0~~	~~1~~
~~1~~	1	0	0	0
~~0~~	1	0	0	~~1~~
~~0~~	0	0	1	0

ACOv1 → ACOv2 → ACOv3 – ACOv3f

Characteristics:

- Each variable has its own pheromone level
- A constructing solution process alternating exploration mode and exploitation mode (SACO, Stuztle 1998)
- A local search based on 1-1 exchanges for the WSPP
- A territory disturbance strategy inspired by a warming up strategy, well-known for the SA

Drawbacks / limits:

- Impossible to solve the RPP instances
- A basic stopping condition
- Good quality of solutions, bad CPU times

Goal for a revised version:

- To deal with large size instances
- To have a more ad-hoc stopping condition

ACOv1 → ACOv2 → ACOv3 – ACOv3f

Characteristics:

- Management of constraints: a dynamic structure (AVL trees)
- A local search based on 1-2 exchanges for the USPP
- A first implementation of a dynamic stopping condition
- Interesting (new best known) solutions for the RPP, CPU times not competitive with GRASP

Drawbacks / limits:

- Excessive CPUt for the biggest instances (indirect consequence of the dynamic structure)
- Not satisfying version of the stopping condition (not successful integrated with the exploration/exploitation strategy).

Goal for a revised version:

- A drastic reduction of the CPUt
- An improvement of the local searches
- A revision of the stopping condition

ACOv1 → ACOv2 → ACOv3 – ACOv3f

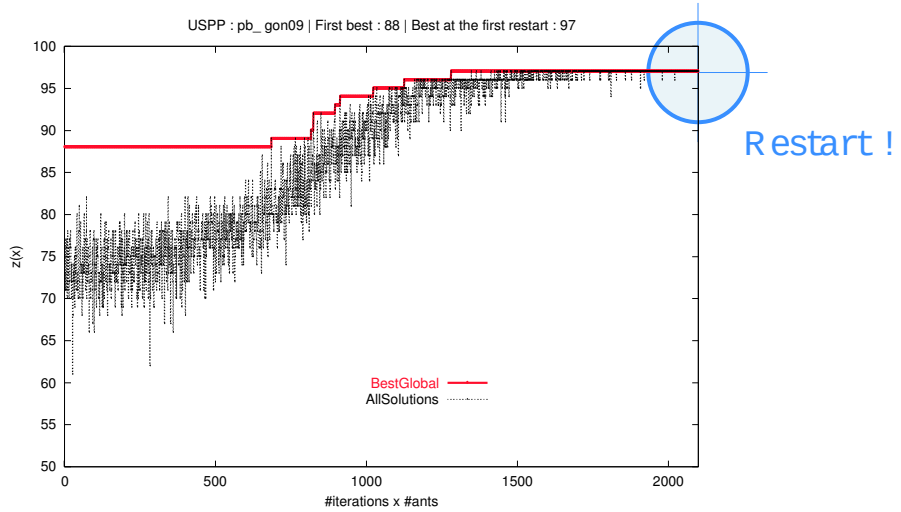
New/revised components:

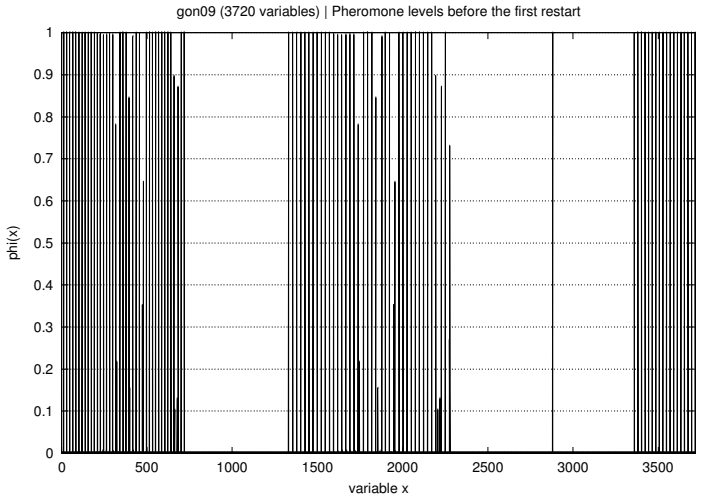
- Ad-hoc data structures and subroutines (the core activity of the algorithm has been redesigned with very efficient routines and data structures regarding to the computational complexity)
- LS based on 1-1 exchanges for the WSPP revised (more aggressive), LS based on 1-2 exchanges for the WSPP integrated
- Stopping condition redesigned (integrated with the exploration/exploitation strategy and based on the interest of the search and diversity of new solutions). ACOv3f: algorithm stopped when the 1st restart is triggered

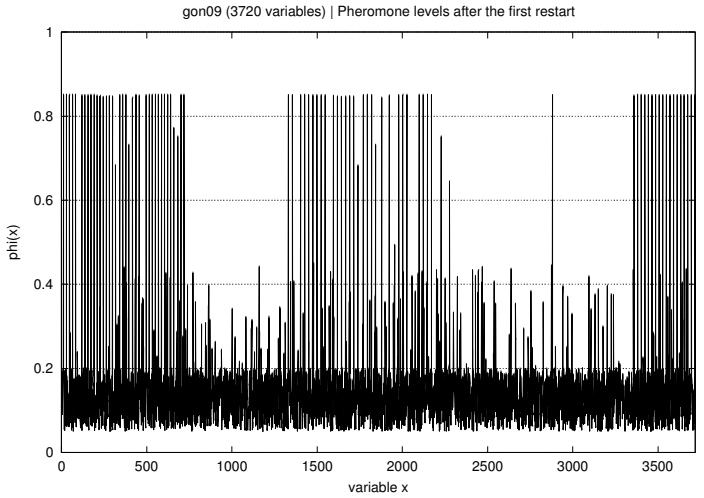
⇒ CPUt drastically reduced
⇒ Search activity increased (more LS, more iterations)
⇒ Automatic stopping condition
⇒ Mature and competitive heuristic for the W/U SPP
⇒ None tuning is required

ACOv3: first convergence and restart

illustrated with gon09 (USPP[RPP], 3720 vars, 482887 csts)

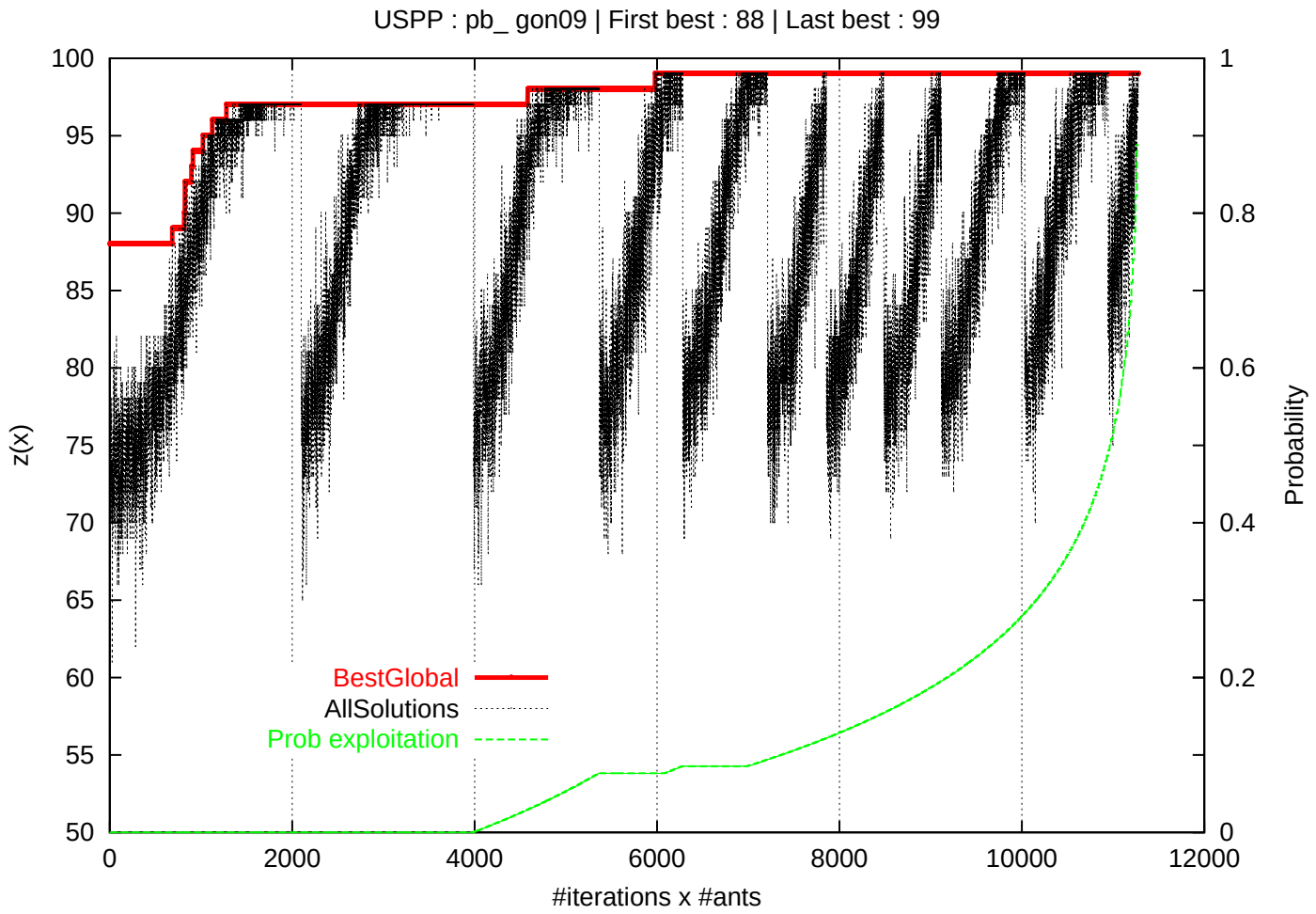






$z(x)$ for ACOv3

illustrated with gon09 (USPP[RPP], 3720 vars, 482887 csts)



MIC 2005
THE 6TH METAHEURISTICS INTERNATIONAL CONFERENCE
AUGUST 22-26, 2005, VIENNA, AUSTRIA

III. Numerical experiments

Data set: 79 instances

100 variables (USPP / WSPP [1,20]):
- 100 / 300 / 500 constraints
- density of matrix between 2.0% and 3.1%

200 variables (USPP / WSPP [1,20]):
- 200 / 600 / 1000 constraints
- density of matrix between 1.0% and 2.6%

500 variables (USPP / WSPP [1,20]):
- 500 / 1500 / 2500 constraints
- density of matrix between 0.7% and 2.3%

1 000 variables (USPP / WSPP [1,20]):
- 1000 / 5000 constraints
- density of matrix between 0.58% and 2.65%

2 000 variables (USPP / WSPP [1,20]):
- 2 000 / 10 000 constraints
- density of matrix between 0.55% and 2.56%

RPP instances (USPP):
- 465 ... 3 720 variables; 8 661 ... 482 887 constraints
- density of matrix between 0.05% and 0.44%

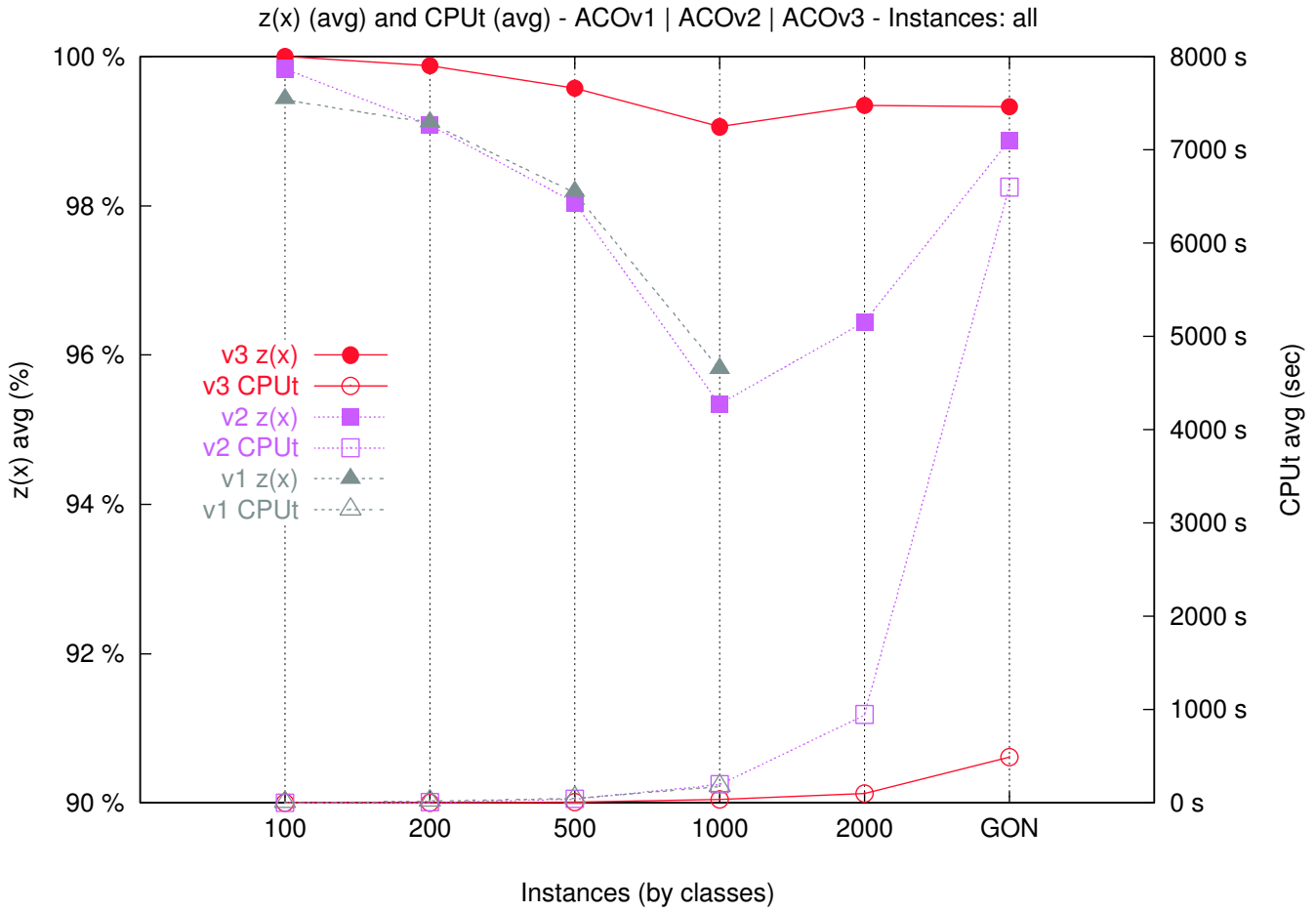
Constant values of the algorithm (parameters)

`maxIter`	auto	Number of iterations
`maxAnt`	15	Number of ants for each iteration
`phiInit`	1.0	Initial pheromone assigned to a variable
`rhoE`	0.9	Pheromone evaporation rate
`rhoD`	phiInit * (1.0 - rhoE)	Pheromone deposition rate
`phiNul`	0.001	Level of pheromone considered as zero
`iterStagnant`	2	Declare the procedure stagnant when no improvement is observed
`restartBefExploit`	2	Number of restarts before enabling the exploitation mode

Experiment information

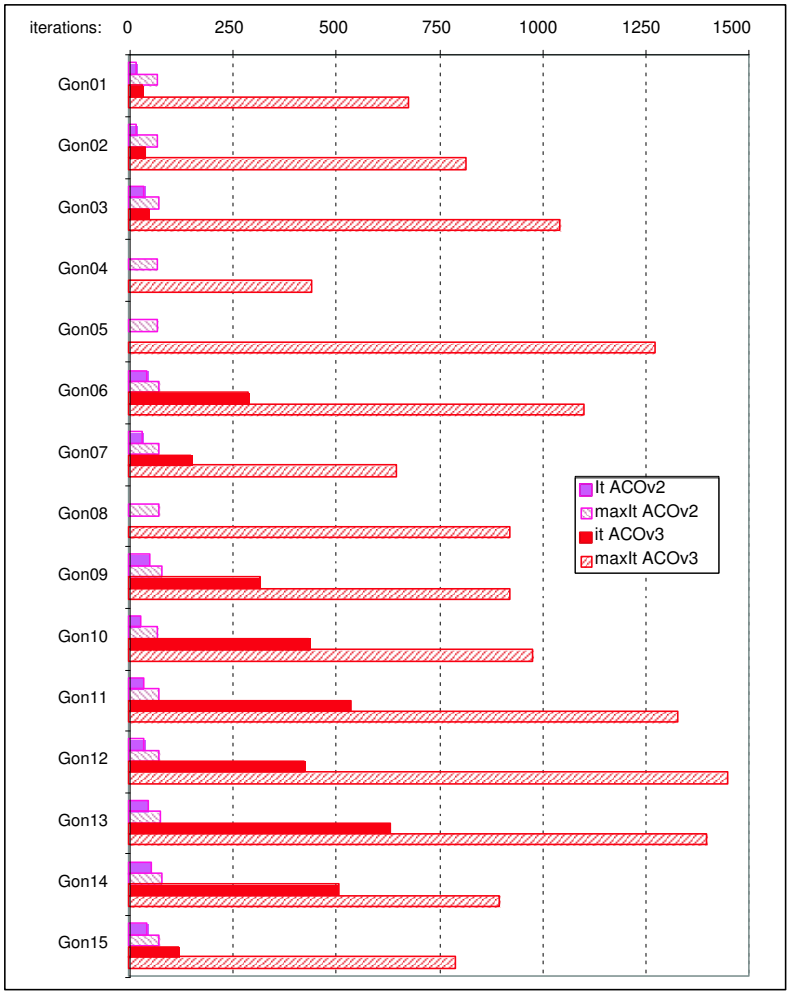
- Numerical instances
  - 100...3 720 variables; 100...482 887 constraints
  - WSPP/USPP(incl. RPP)
- Experimental environment
  - Apple PowerBook; PowerPC G4 1GHz; RAM 512Mb; Mac OS X 10.3.9
- ACO (all versions)
  - C language; gcc 3.3 and -O3
- GRASP (Delorme 2004)
  - Ada language; gnat 3.3

ACO v1 / v2 / v3



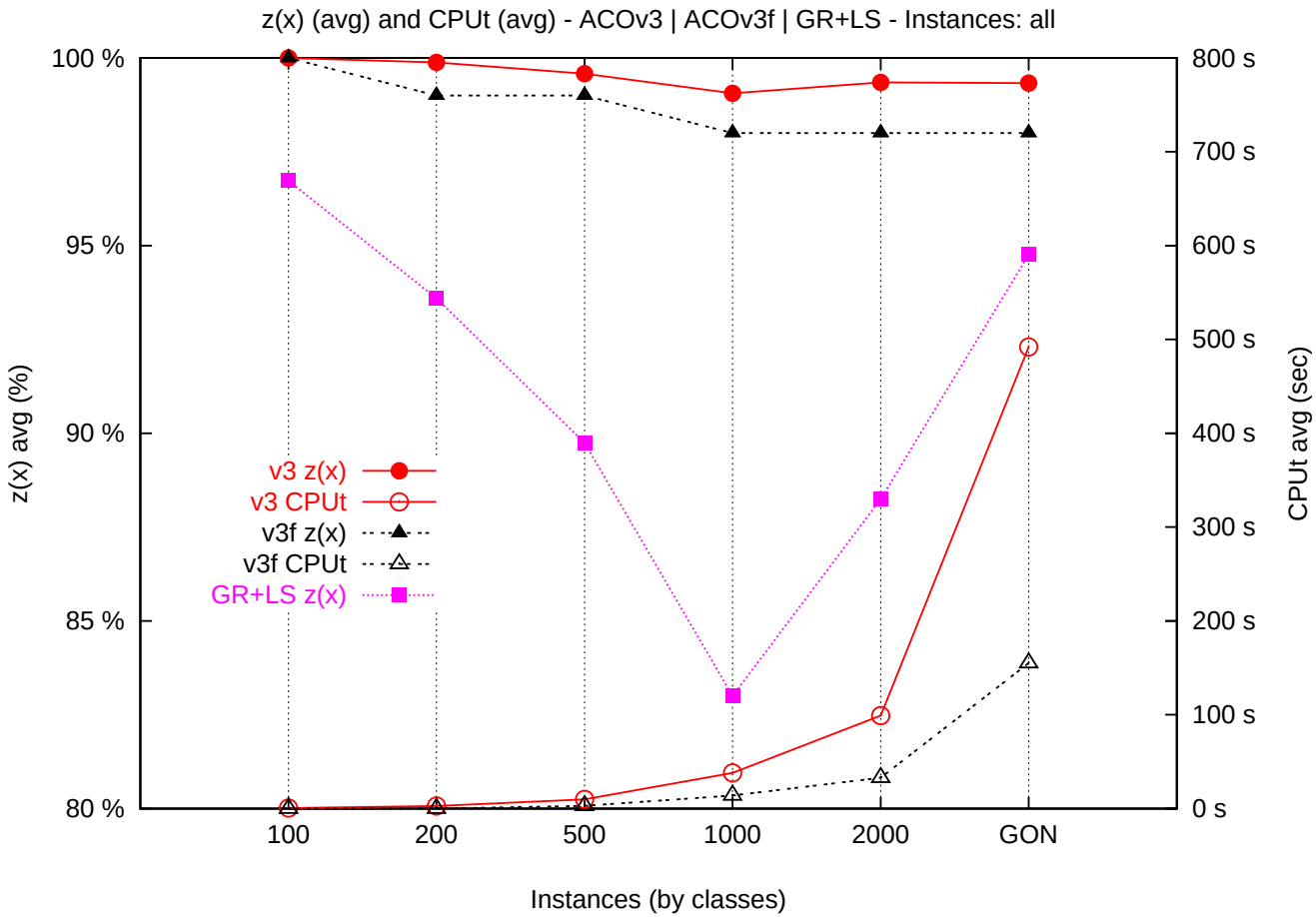
agregated (avg) results / 16 runs / y_1: $z(x)$ (%) vs best known / y_2: CPUt

ACO v2 / v3



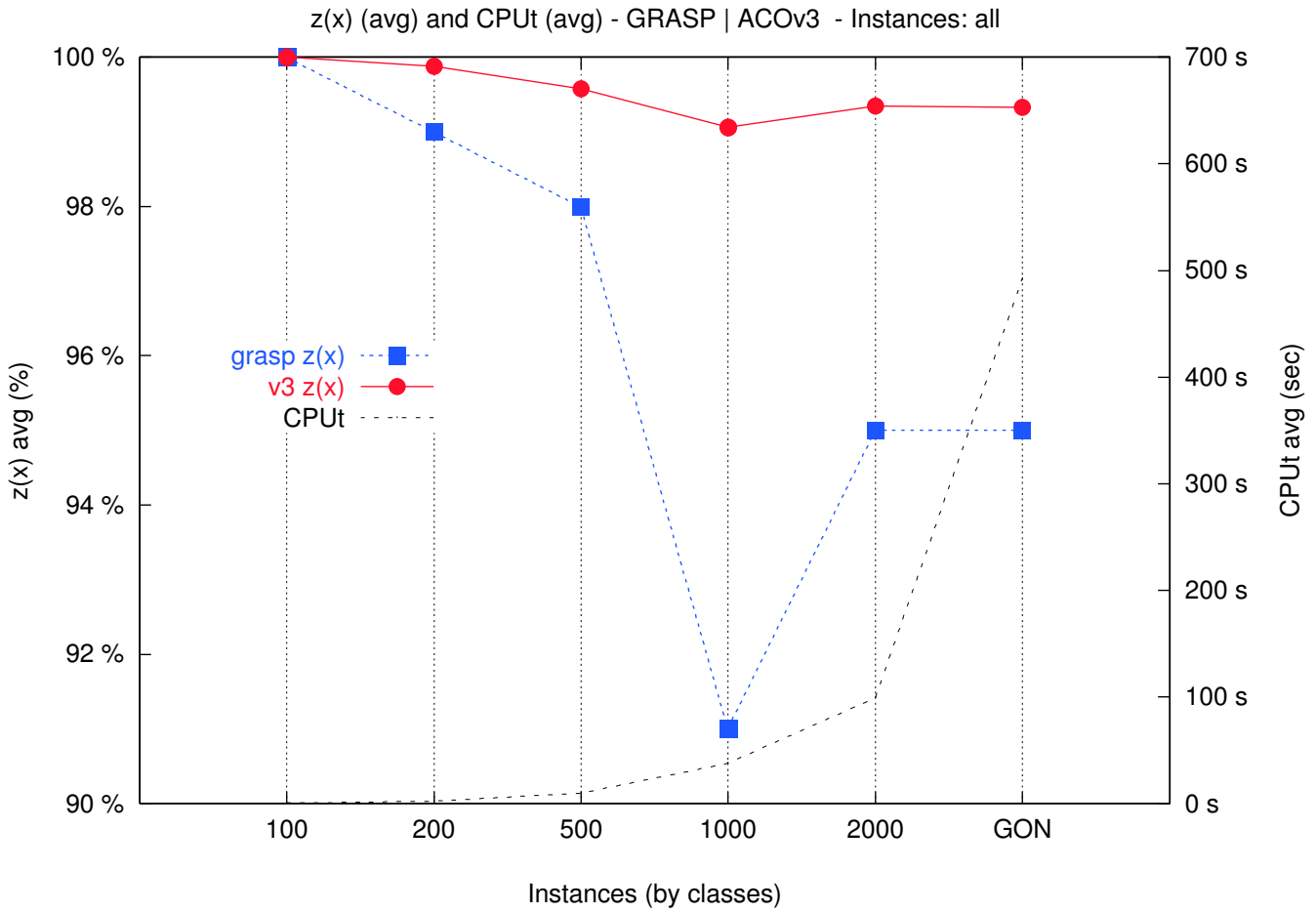
agregated (avg) results / 16 runs / RPP instances

ACO v3 / v3f / GR+LS



agregated (avg) results / 16 runs / y_1: $z(x)$ (%) vs best known / y_2: CPUt

GRASP / ACO v3 (1/2)



agregated (avg) results / 16 runs / y_1: $z(x)$ (%) vs best known / y_2: CPUt

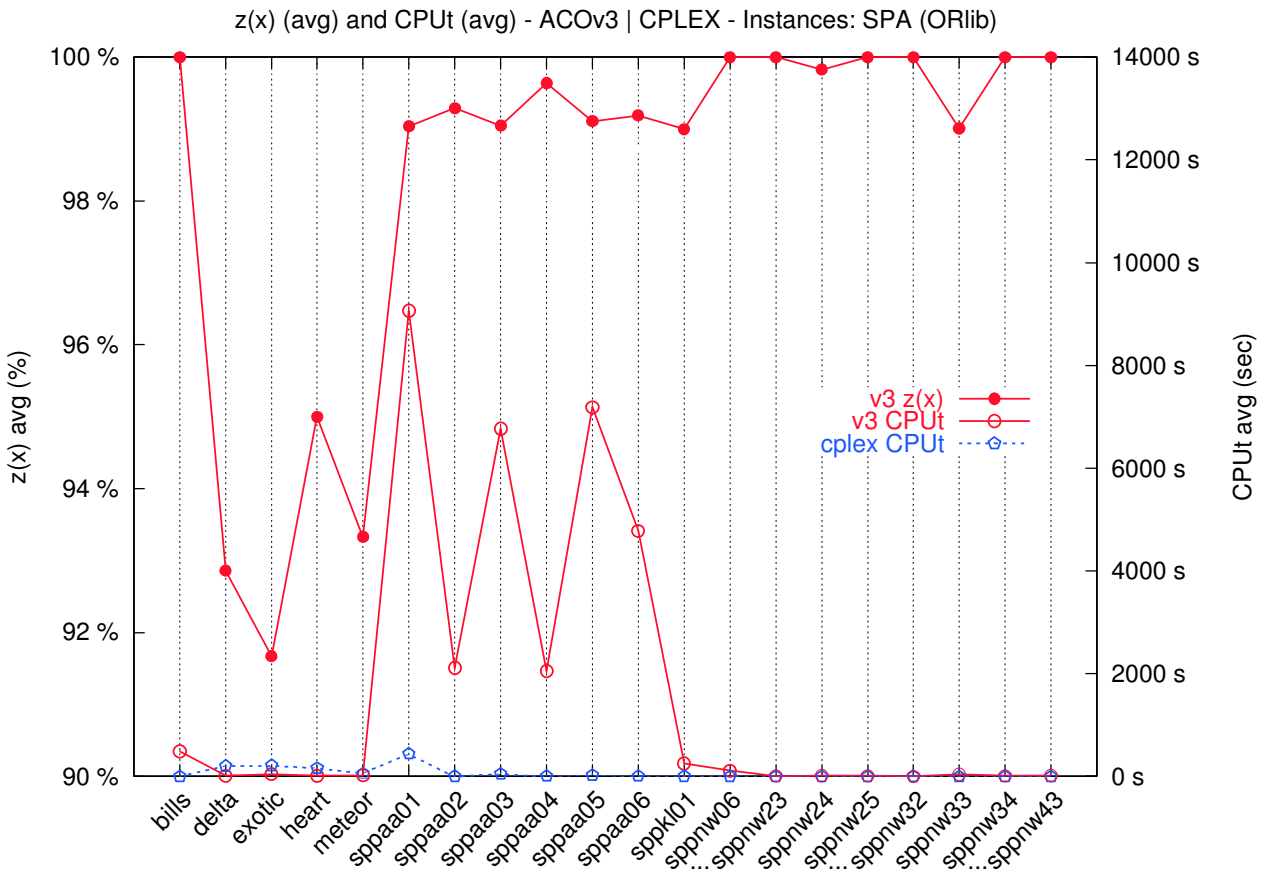
GRASP / ACO v3 (2/2)

RPP instances			avg Quality (%)			avg CPUt (s)	
name	# var.	# con.	ACOv3	ACOv3f	GRASP	ACOv3 [GRASP]	ACOv3f
Gon01	465	8661	100.00	100.00	96.67	10.38	3.25
Gon02	620	11676	100.00	100.00	95.21	25.50	7.62
Gon03	1240	23736	100.00	100.00	93.96	175.00	58.00
Gon04	1240	50705	100.00	100.00	98.80	80.62	28.75
Gon07	1620	79514	99.44	99.09	96.07	187.69	62.38
Gon05	1860	55938	99.34	99.34	98.97	485.44	162.81
Gon06	1860	119009	98.55	97.90	94.32	258.44	75.25
Gon14	2160	140632	99.63	98.12	95.08	360.33	110.00
Gon10	2400	186861	100.00	99.21	97.25	407.67	129.33
Gon15	2503	185523	98.39	98.08	96.47	468.67	155.67
Gon11	2683	228972	99.62	98.47	97.32	606.00	163.00
Gon12	2880	311228	98.12	97.75	92.13	635.00	178.67
Gon13	3210	380292	99.26	98.14	94.81	734.67	212.33
Gon08	3720	155985	100.00	100.00	99.08	2256 .00	789.25
Gon09	3720	482887	97.62	95.92	89.46	689.00	195.67

Best % — Δ with ACOv3 : ≤3% ≤ 8% > 8 % — Δ ACOv3/v3f : > 3× faster

RPP results / 16 runs / quality (%) vs best known

ACOv3 and Set partitionning data (ORlib) viewed as SPP (20 instances)



3 runs / y_1: performances (%) vs optimal solution (cplex) / y_2: CPUt

NB: optimal solutions have been computed with cplex 8.1 under a Pentium M 1.8Ghz

Some remarkable facts

- Cplex 8.2 on PIII 800Mhz needs
  - up to **156109.36 seconds** yet for the class 200
  - cannot deal with some 500, 1000, 2000 and GON instances
- # instances where the best known solution is
  - found at least one time: **74** (93.67%)
  - always found: **43** (54.43%)
- Number of new best known solutions found: **9** (ACOv1: 3 – before: 6)
- Average percentage from the best known solution: **99.59%** i.e.
  - 1.28% better than ACOv1 (for instances 100-1000)
  - 0.39% better than ACOv3f
- Average CPUt comsumed by ACOv3: **110.27 seconds** i.e.
  - 2.23 times less than ACOv1
  - 2.90 times more than ACOv3f

IV. Conclusion – Perspectives

- ACOv3 is a mature approximation method for solving U/W SPP
  - Competitive with the previous existing method (GRASP)
  - None tuning is requested
  - ACOv3f is a compromise alternative for practical applications
- Possible improvement:
  - revisit the stopping criterion (less iterations)
- Possible forthcoming works:
  - a version dedicated for the RPP (exploiting the specific structure of the matrix constraints)
  - a procedure which set on variables (preprocessing or during the resolution process)

Questions – Discussion