CAHIER DU LAMSADE

Laboratoire d'Analyse et Modélisation de Systèmes pour l'Aide à la Décision
(Université Paris-Dauphine)
Unité de Recherche Associée au CNRS n° 825

COMPARISON OF SIMULATED ANNEALING AND GENETIC ALGORITHM WITH A HYBRID METHOD FOR GENERAL INTEGER PROBLEMS

CAHIER N° 147
juillet 1997

Halim M'SILTI [1]
Pierre TOLLA [1]
Arnaud SCHAAL [1]

received: June 1997.

[1] *LAMSADE, Université Paris-Dauphine, Place du Maréchal De Lattre de Tassigny, 75775 Paris Cedex 16, France (e-mail: {msilti,tolla,schall}@lamsade.dauphine.fr).*

CONTENTS

Comparaison d'une méthode du recuit simulé et d'un algorithme génétique au sein d'une méthode hybride de résolution de problèmes linéaires en nombres entiers généraux.

Résumé

Dans un précédent article [Schaal et al., 1997], nous avons présenté un schéma hybride, appelé Mip-Genint (Mixed Integer Problems - GENetic INTerior), combinant des méthodes de points intérieurs (IPM), des algorithmes génétiques (GA) et un générateur de coupes (CG) permettant de résoudre des problèmes linéaires en nombres entiers généraux. Dans cet article, nous proposons d'évaluer l'efficacité d'une méthode de recuit simulé (SA) lorsque celle-ci est implantée à la place de l'algorithme génétique. Nous comparons les résultats obtenus en utilisant la méthode du recuit simulé, avec et sans les points d'ancrages recentrés, avec les résultats obtenus en utilisant l'algorithme génétique. La motivation de cette étude est née de l'observation de l'inefficacité de l'opérateur de croisement de l'algorithme génétique dans certains cas, cette inefficacité pouvant provenir de la nature probabiliste de cet opérateur.

mots-clefs : problèmes linéaires en nombres entiers généraux, méthodes de points intérieurs, algorithmes génétiques, recuit simulé, coupes économiques.

Comparison of simulated annealing and genetic algorithm within a hybrid method for general integer problems.

Abstract

In a previous paper [Schaal et al., 1997], we have presented an hybrid scheme, named Mip-Genint (Mixed Integer Problems - GENetic INTerior), combining interior point method (IPM), a genetic algorithm (GA) and a cut generator (CG) for solving general integer programming problems. We propose here an evaluation of a simulated annealing algorithm (SA) in place of the genetic algorithm. We compare results obtained by the simulated annealing algorithm, with or without the use of centered anchor points, with the results obtained using the genetic algorithm. This study is due to the observation that the crossover operator of the genetic algorithm is inefficient for some problems. This inefficiency could be due to the probabilistic nature of this operator.

keywords: general linear integer programming, interior point methods, genetic algorithm, simulated annealing, economic cuts.

1 Introduction

In this paper, we propose a new version of the hybrid heuristic framework previously presented [Schaal, 1997, Schaal et al., 1997], named Mip-Saint[1]. The results obtained with the genetic algorithm in this method have shown the interest of our method to solve general integer problems. The continuous optimization techniques find quickly points named "anchor points". The SA algorithm explores the integer-solution space around the anchor points in order to find "satisfactory" solutions. Cuts enable us to determine "centered anchor points" inside the feasible space.

This framework is designed to solve the following general linear integer programming problems:

$$(IP_\kappa)\begin{cases} \min & c^{\mathrm{T}}.x \\ s.c. & A.x + f = b \\ & 0 \le x \le u \\ & 0 \le f \le \kappa \\ & x \in \mathbb{Z}^n \\ & f \in \mathbb{R}^m \end{cases}$$

where κ are bounds imposed on the slacks to introduce the uncertainty inherent to the constraints definition [Schaal, 1997]. The relaxations (LP) and (LP_{cut}) of (IP_κ) have solutions with nil slacks.The solutions of these relaxed problems make it easier to find the feasible integral solutions of (IP_κ).

$$(LP)\begin{cases} \min & c^{\mathrm{T}}.x \\ s.c. & A.x = b \\ & 0 \le x \le u \\ & x \in \mathbb{R}^n \end{cases} \qquad (LP_{cut})\begin{cases} \min & c^{\mathrm{T}}.x \\ s.c. & A.x = b \\ & 0 \le x \le u \\ & c^{\mathrm{T}}.x = cut \\ & x \in \mathbb{R}^n \end{cases}$$

Based on the constraint satisfaction methods [Schiex et al., 1997, Fargier et al., 1996], Mip-Saint tries to satisfy most of the constraints within a reduced search computation time. This method does not require manipulating the integral feasible solutions (cf. fig. 1). The method consist of two steps:

- step 1: computation of the first anchor point x_c^* (interior point method, LP),
- step 2:
  - 2.1: computation of integral solutions (simulated annealing algorithm, IP_κ),
  - 2.2: computation of centered anchor points x_r^* (interior point method, LP_{cut}).

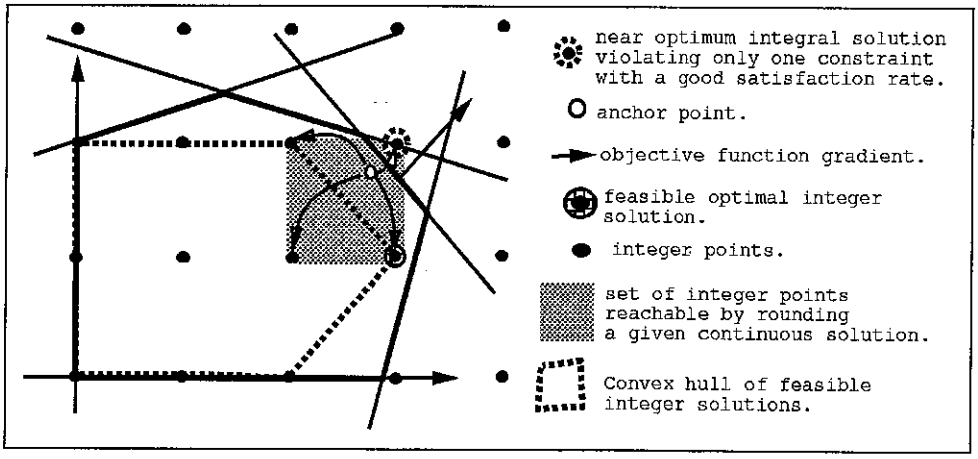

Figure 1: computing of integral components

[1]Mixed Integer Simulated Annealing INTerior

We present a comparative study of Mip-Saint, with or without the help of centered anchor points, and Mip-Genint when applied to 50 problems, randomly generated, having 50 to 1,000 variables and 50 to 95 constraints with dense constraint matrix where more than 70% matrix coefficients are different of zero. This paper is organized as follows. In section 2, we outline the methods and techniques used in Mip-Saint; we present ties with other existing methods at the end of this section. In section 3, we provide the results of the computational experiments. In section 4, we conclude this paper and present further developments; results tables are grouped at the end of the paper.

2 Hybrid organization

It is well-known that most of the existing methods to solve linear programming problems are organized in a hybrid manner including continuous and integral overlapped techniques. Nevertheless, we must stress that there exists a major difference between the traditional methods and both Mip-Genint and Mip-Saint method. This difference results from the frequency calls of the continuous method (stage 2.2). In fact, a new cut is generated and a new centered anchor point is computed only after the convergence of the simulated annealing algorithm to a non-satisfactory solution. In the most cases, the search in the neighbourhood of x_c^* does not lead[2] to a "satisfactory" solution; however, centered anchor points improve the convergence. The cut generator uses the objective function to construct a new constraint and to modify its level (right hand side). This cut is included in the formulation of (LP_{cut}) problem solved by infeasible-interior-point algorithm (see chapters 2, 3, 4 and 6 of [Beasley, 1996]) . The stopping criterion of the interior point algorithm (primal feasibility) is weakened as the anchor points are to be rounded. The optimality of solutions, constructed by Mip-Saint, is checked by the examination of $c^{\mathrm{T}}.(x - x_c^*)$.

In Mip-Saint, solution quality is improved by the simulated annealing algorithm in order to satisfy the constraints by exploring neighbourhood $V(\bullet)$ of the anchor points x_c^* and x_r^*. We propose to deal with solutions satisfying most of the constraints and "close[3]" to the feasible boundary. [Fox, 1993] has proposed to use a tabu list of precedents encountered solutions in order to accelerate the algorithm. [Koakutsu et al., 1995] have proposed to use a combination of genetic algorithm and simulated annealing method which combines objectives degradations of the simulated annealing and crossing over solutions in the genetic algorithm.

2.1 Simulated annealing

The simulated annealing algorithm was first applied by [Kirkpatrick et al., 1983] to the optimization problems; it was applied to integer programming problems by [Johnson et al., 1991], [Dowsland, 1993] and [Czyzak et al., 1994]. This algorithm is based on the analogy of the physical entropy principle. This method asymptotically converges towards optimal solutions under sufficient assumptions and allows to simplify regularity hypothesis imposed to objective f and constraints function [Goffe, 1995]. Furthermore, this method is adapted to handle objectives with many local extrema and not well-defined. This algorithm is a hill-climbing method where objective degradations are allowed after local moves in the integral solution space.

One solution is accepted or rejected under a probability law parametrized by the temperature θ^k which lowers with the time factor. Higher the temperature, more important the probability of accepting a non-improving solution is. This allows the algorithm from leaving local minima. The temperature cooling schedule might allow rizing θ^k when no improvements are made during a

[2] Some rounding algorithms are polynomial [Lakshminarayanan and Chandrasekaran, 1994], but only for covering or packing problems ; none general purpose algorithm are polynomial.

[3] Measured by the relative distance between the solution and the unsatisfied constraints.

certain amount of time. Algorithm of simulated annealing, including centralization of the solution, is given in figure 2. The centerd anchor points have an impact on the efficiency of the simulated annealing process as we can see in tables and figures presented at the end of this paper. The operator, which finds a new solution y in the vicinity of the old solution x, enlarges the scope of the explored solution space by altering, each time, one component of the current x solution. The probability of accepting the y solution from the current x solution, taking into account the objective function f, is given by:

$$Proba(\text{accepting the move from x to y}) = \min\{1, exp(-\frac{f(y) - f(x)}{\theta^k})\}.$$

```
BEGIN                                                    {Initialisation}
      θ ← θ0;
      j ← 0;
      k ← 0;
      choose x^0;
      x* ← x^0;
      REPEAT
      BEGIN
            θ ← g(k, θ);
            REPEAT
            BEGIN
                  IF ( reinitialization criterion is verified ) THEN
                  BEGIN
                        Compute a centered anchor point x*_r;
                        x^j ← x*_r;
                  END;
                  Find y ∈ V(x^j);
                  Compute p = min{ 1, exp(-(f(y)-f(x^j))/θ^k) };
                  Choose x^{j+1} ← y with p probability;
                  x^{j+1} ← x^j with 1 - p probability;
                  j ← j + 1;
                  IF f(x^j) < f(x*) THEN x* ← x^j;
            END
            UNTIL (stopping criterion verified );
            k ← k + 1;
      END
      UNTIL ( θ = 0);
END;
```

Figure 2: Simulated annealing algorithm

2.2 Economic cut and centering

The economic cut $c^T .x \geq \alpha \times c^T .x_c^*$, parameterized by α which represents the depth of the cut, finds different anchor points. These points are computed by taking a step further inside the feasible region, increasing the economic level by α (fixed to 1.005 in Mip-Saint). This cut produces a degeneracy in the linear problem which guarantees the convergence of the interior point method to the center of the optimal face. Therefore, the computation of a solution with the primal dual algorithm ends in the interior of the initial feasible region.

2.3 Motivation of simulated annealing vs. genetic algorithm

The major motivation of the use of a simulated annealing method in stage 2.1 of our hybrid framework is the inefficiency of the crossover operator pointed out in previous experiments [Schaal, 1997,

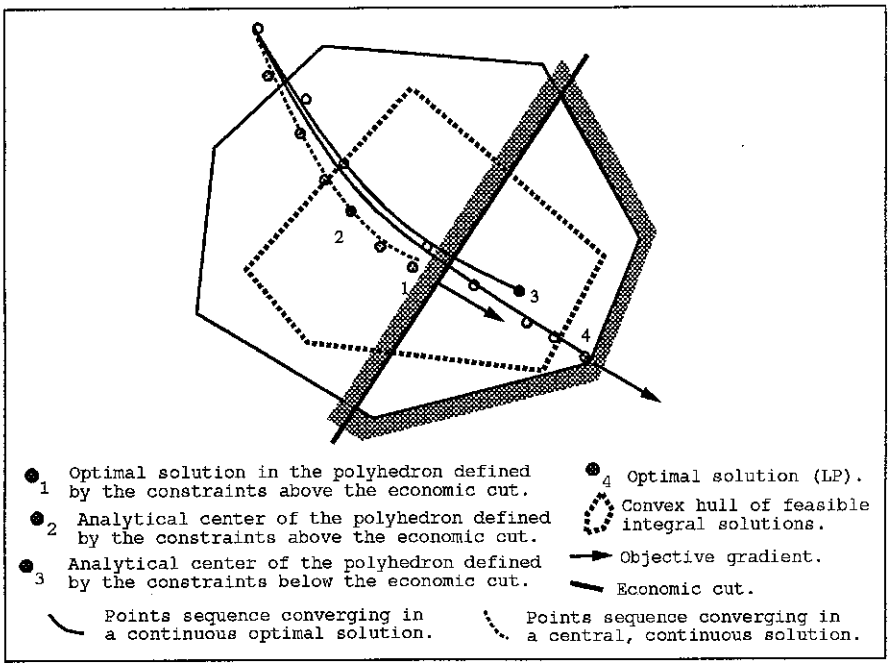

Figure 3: Economic cut and centering

Schaal et al., 1997]. The main reason that could motivate this observation is directely connected to crossover operators and the mating principles used to construct couples of parents. Traditional crossover operators (one point or multiple point versions) implemented in genetic algorithm are pure probabilistic operators. Therefore sharing the information between two different solutions remains too chaotic. The use of linear combination, in the context of crossover operators, has shown unefficient in the first study we made [Schaal, 1997] as we were unable to choose properly couples to mate.

A different approach that takes into account the inefficiency of crossover was proposed by Glover. He noted that our approach shares several features in common with the scatter search framework [Glover, 1977]. The latter procedure provides an evolutionary procedure based on a different foundation than genetic algorithms. These features include the following:

1. the generation of anchor points (called reference points in scatter search) which are "weighted centers" of regions of interest;

2. the use of generalized rounding to restore integer feasibility for integer variables that receive fractional values.

The scatter search approach combines solutions by using linear combinations as the basic operators. These operators are applied strategically rather than by relying on randomized processes as in genetic algorithms, and include the prescription of joining more than two solutions at a time. This strategic orientation, which provides the ability to examine parts of the space within and beyond the convex hull of points previously examined. This orientation makes it possible to focus, more strongly, on elite solutions without significant risk of inbreeding. This latter feature also accords with part of the motivation behind our use of economic cuts and centering. More details of comparisons and contrasts between scatter search and genetic algorithms are provided in [Michalewicz, 1995] and in [Glover, 1994, Glover, 1996].

3 Computational results

In table 1 we collect information on ten instances of five problems that have been used for our experimental study. All problems have been randomly generated; for all instances, by construction, a solution (x, f) to problem IP_0 exists. Such problems are not well represented in the Miplib mixed integer optimization test examples where 72.88% elements are pure $0-1$ problems or are sparse. Our method is devoted to solve problems with dense matrix and integer valued variables with bounds higher than 1.

The evaluation criteria, presented in table 2, are based on the assessment of the number of satisfied constraints, the slacks variables f and the residual r_j where: $r_j = \max\{(A.x-b)_j; 0\}, \forall j \in 1..m$. Following Gondzio contribution (Lamsade, 1997) MIP-SAINT find efficient efficient approximate solutions (see fig. 1) and is not designed to deal with unfeasibility of the problem solved. Experience schows that this approach is adapted to solve industrial problems [Schaal et al., 1993] where we can accept a solution with relative duality gaps reasonably small.

As some problem instances have very good initial anchor points, therefore, we should initialize the first solution using x_c^* by randomly choosing between: $E(x_c^*) - 1$, $E(x_c^*) + 1$ and $E(x_c^*)$. Tables 3 to 7 present results of Mip-Genint and table 8 gives the optimality gap of solutions found for $(IP)_0$ problem in the P50 familly. Tables 9 to 4 present results obtained with Mip-Saint when centerd anchor points are not used. Finally, tables 14 to 18 present results obtained with Mip-Saint when using centerd anchor points. These results are summarized in figures 4 and 5 which present the number of constraints (#C) satisfied. All results were obtained on a SUN-SPARC 5 workstation using gnu C compiler.

4 Conclusion

The centered anchor points improve the local search and allow Mip-Saint to:
• find solutions quickly,
• improve these solutions in order to find satisfactory rate of one thousandth.
The comparison of results between Mip-Saint and Mip-Genint confirms the role of crossover operator in solving some of the problems used. However, even if a strong superiority exists for the genetic algorithm when centering is not used, the gap reduces when centering is allowed and the only difference existing between the two approaches lies in the number of solutions generated when the algorithm is stopped. Possible improvements to our hybrid scheme are:
• implementing embedded operators, taking into account the problem features,
• parallelizing the method implemented in stage 2.1 of Mip-Saint,
• marrying our approach with the scatter search framework, as mentioned in section 2.3, which has a number of features that are capable of being exploited by our design.

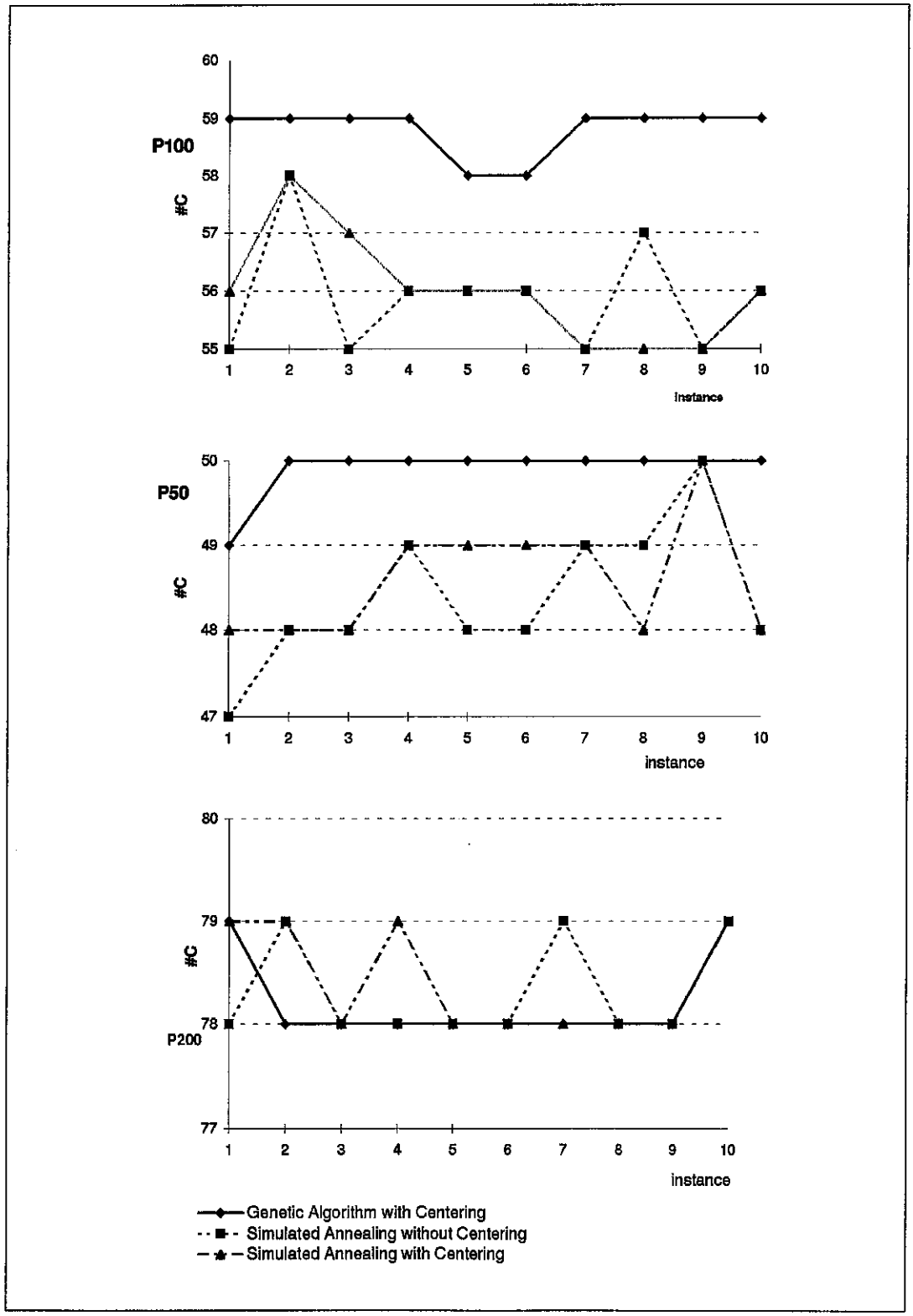

Figure 4: *Mip-Genint vs. Mip-Saint*

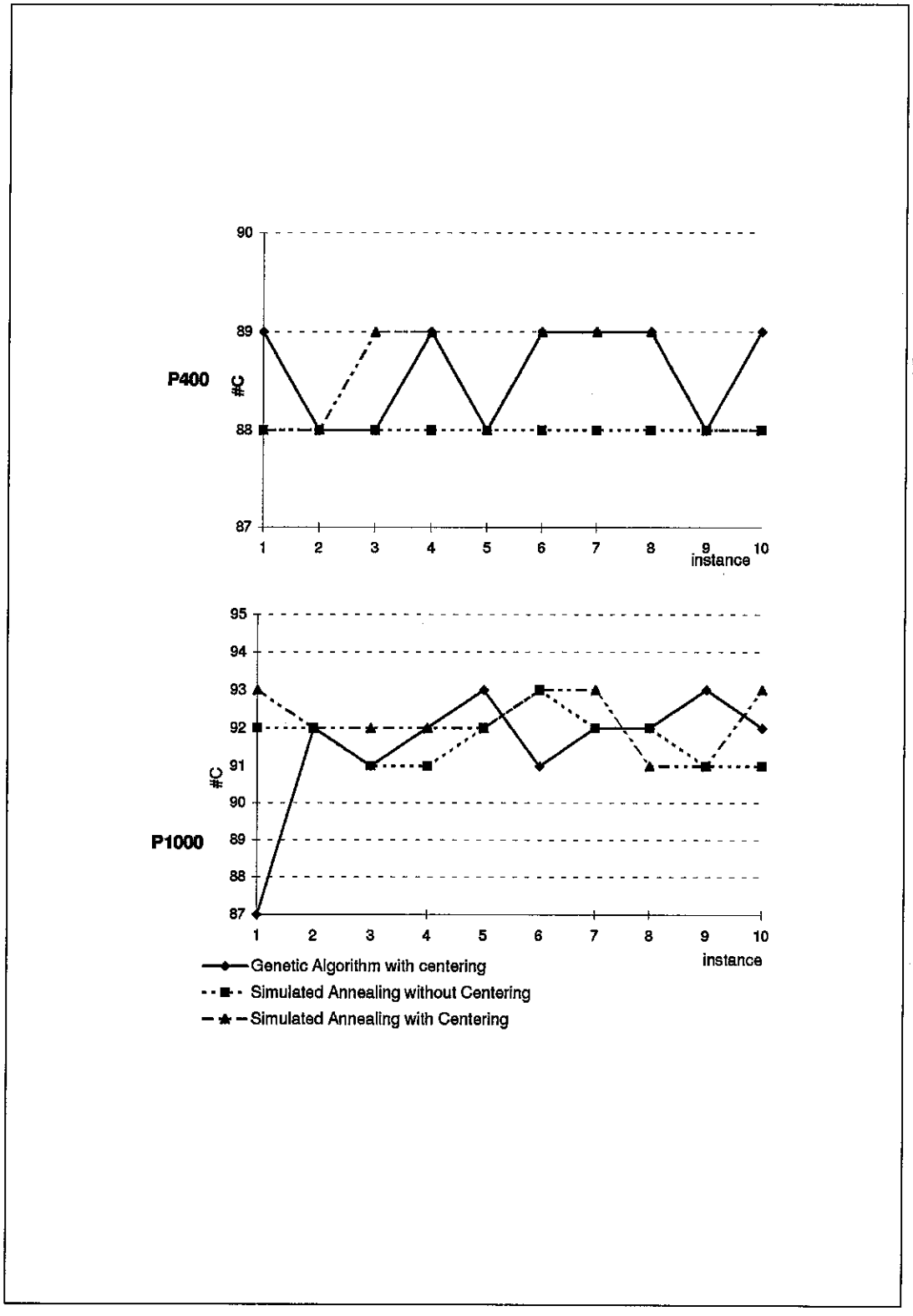

Figure 5: *Mip-Genint vs. Mip-Saint*

Acknowledgements

The authors would like to thank Prof. Fred Glover for many constructive observations including ties between our approach and the scatter search that lead to a better presentation of our results and further developments of Mip-Genint. We also thank Elizabeth Auzan, Dominique François and Adel Guitouni for valuable remarks concerning the English structure of this paper.

References

[Beasley, 1996] Beasley, J., editor (1996). *Advances in linear and integer programming.* Oxford science publications, Oxford. Oxford lecture series in mathematics and its applications, 4.

[Czyzak et al., 1994] Czyzak, P., Hapke, M., and Jaszkiewicz, A. (1994). Application of the pareto-simulated annealing to the multiple criteria shortest path problem. Technical report, Institute of Computing Science, Poznan University of Technology, Poznan, Poland.

[Dowsland, 1993] Dowsland, K. (1993). Some experiments with simulated annealing techniques for packing problems. *European Journal of Operations Research*, 68:389–399.

[Fargier et al., 1996] Fargier, H., Dubois, D., and Prade, H. (1996). Problèmes de satisfaction de contraintes flexibles : une approche égalitariste. *Revue d'Intelligence Artificielle*, 9(3):311–354.

[Fox, 1993] Fox, B. (1993). Integrating and accelerating tabu search, simulated annealing, and genetic algorithms. *Annals of Operations Research*, 41.

[Glover, 1977] Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8(1):156–166.

[Glover, 1994] Glover, F. (1994). Tabu search for nonlinear and parametric optimization (with links to genetic algorithms). *Discrete Applied Mathematics*, 49:231–255.

[Glover, 1996] Glover, F. (1996). Tabu search and adaptive memory programming – advances, applications and challenges. In Barr, H. and Kennington, editors, *Interfaces in Computer Science and Operations Research*, pages 1–75. Kluwer Academic Publishers.

[Goffe, 1995] Goffe, W. (1995). Simulated annealing - global optimization method. Abstract of algorithms, Econometrics laboratory, University of Californie, Berkeley.

[Johnson et al., 1991] Johnson, D., Aragon, C., McGeoch, L., and Schevon, C. (May-June 1991). Optimixation by simulated annealing: an experimental evaluation; part ii, graph colouring and number partitioning. *Operations Research*, 39(3):379–406.

[Kirkpatrick et al., 1983] Kirkpatrick, S., Gellat, C., and Vecchi, M. (1983). Optimization by simulated annealing. *Science*, 220:671–681.

[Koakutsu et al., 1995] Koakutsu, S., Kang, M., and Wei-Ming Dai, W. (1995). Genetic simulated annealing and application to non-slicing floorplan design. Technical Report UCSC-CRL-95-52, Baskin Center for Computer Engineering & Information Sciences, University of California, Santa-Cruz.

[Lakshminarayanan and Chandrasekaran, 1994] Lakshminarayanan, S. and Chandrasekaran, R. (1994). A rounding algorithm for integer programs. *Discrete Applied Mathematics*, 50:267–282.

[Michalewicz, 1995] Michalewicz, Z. (1995). Heuristic methods for evolutionary computation techniques. *Journal of Heuristics*, 1:177–206.

[Schaal, 1997] Schaal, A. (1997). Une approche hybride de résolution de problèmes linéaires en nombres entiers : méthodes intérieures et meta heuristiques. PhD Thesis to be presented.

[Schaal et al., 1993] Schaal, A., M'Silti, H., and Tolla, P. (1993). Modélisation et système interactif de stocks liés.

[Schaal et al., 1997] Schaal, A., M'Silti, H., and Tolla, P. (1997). Une approche hybride de résolution de problèmes linéaires en nombres entiers. Technical report, Lamsade, University of Paris-Dauphine. An English and updated version is submitted to Journal of Heuristics.

[Schiex et al., 1997] Schiex, T., Fargier, H., and Verfaillie, G. (1997). Problèmes de satisfaction de contraintes valués. À paraître dans la Revue d'Intelligence Artificielle.

problem family	number of variables	number of constraints	u_{max}	c_{max}
P1000	1,000	95	100	100
P400	400	90	100	100
P200	200	80	100	100
P100	100	60	100	100
P50	50	50	100	100

Table 1: problems description

notations	meaning
res,	results of the method
DV	the method diverges and does not found any solution
CV	the method converges in a solution
u_{max}	maximum value of u bounds
c_{max}	maximum cost coefficient
inst.	instance of the problem
$\#C$	number of satisfied constraints
$\| r \|_1$	amplitude of the "non satisfaction" of the constraints
$\#A$	number of modifications of the best solution
$\overline{(\frac{r}{b})} = \frac{1}{m} \times \sum_{j=1}^{m} \frac{r_j}{b_j}$	" satisfaction rate" of constraints
$\| f \|_\infty$	maximum value of slacks components
IM	iteration where best individual was found
cpu1	processing time (seconds) needed to compute the best solution
cpu2	processing time (seconds) needed to exceed the maximum generation number

Table 2: notations used

inst.	#C	$\| r \|_1$	$\overline{(\frac{r}{b})}$	$\| f \|_\infty$	$c(x_e^*)$	$c(x_c^*)$	cpu1	cpu2
0	49	1	0,000667	96	61902	61914	14,93	56,48
1	50	0	0	0	60845	60803,3	19,43	54,28
2	50	0	0	0	69952	69841,7	13,79	54,59
3	50	0	0	0	63727	63506,1	12,16	53,14
4	50	0	0	0	57852	57724,9	13,81	56,36
5	50	0	0	0	63511	63332,6	12,35	58,67
6	50	0	0	0	76386	75977,6	4,93	50,52
7	50	0	0	0	56807	56705,6	38,53	59,34
8	50	0	0	0	61886	61813,4	0	65,26
9	50	0	0	0	65984	65770	10,04	51,67

Table 3: Mip-Genint : P50-Centered anchor points computed

inst.	#C	$\| r \|_1$	$\overline{(\frac{r}{b})}$	$\| f \|_\infty$	$c(x_e^*)$	$c(x_c^*)$	cpu1	cpu2
0	59	72	0.004364	4,667	112,240	114,107	451.06	1439.89
1	59	1	0.000141	238	86,395	86,662.6	175.39	1364.02
2	59	127	0.011199	13,303	118,100	115,129	1174.18	1448.63
3	59	117	0.026712	10,375	103,343	109,960	151.24	1438.52
4	58	133	0.026883	7,059	120,505	127,865	66.94	1407.21
5	58	106	0.041549	4,578	112,714	116,202	121.74	1384.48
6	59	41	0.007593	4,211	89,489	94,706.2	652.50	1375.16
7	59	86	0.012045	4,674	102,697	102,163	1095.22	1402.60
8	59	52	0.036111	6,046	95,283	100,860	87.53	1449.49
9	59	2	0.000694	256	126,319	126541	527.66	1419.60

Table 4: Mip-Genint : P100-Centered anchor points computed

inst.	#C	$\| r \|_1$	$\overline{(\frac{r}{b})}$	$\| f \|_\infty$	$c(x_e^*)$	$c(x_c^*)$	cpu1	cpu2
0	79	254	0.005188	27,501	181,186	173,949	191.56	307.75
1	78	336	0.011240	22,663	210,582	209,466	33.91	292.09
2	78	194	0.012082	28,775	151,509	163,864	22.64	284.18
3	78	362	0.009582	29,758	197,819	197,384	39.29	317.18
4	78	236	0.013143	25,830	187,109	184,814	87.81	291.03
5	78	564	0.021147	62,523	148,578	166,255	82.76	296.24
6	78	33	0.003573	3,695	198,735	197,569	135.71	296.82
7	78	56	0.005099	4,332	200,531	201,161	20.98	294.64
8	78	242	0.013105	23,468	171,478	167,897	85.91	285.38
9	79	217	0.007351	31,671	200,806	206,297	134.27	305.40

Table 5: Mip-Genint : P200-Centered anchor points computed

inst.	#C	$\| r \|_1$	$\overline{(\frac{r}{b})}$	$\| f \|_\infty$	$c(x_e^*)$	$c(x_c^*)$	cpu1	cpu2
0	89	501	0.023194	58,067	383,168	348,043	967.60	981.84
1	88	533	0.008025	55,458	402,109	388,219	620.72	981.46
2	88	101	0.006310	8,879	324,130	331,991	188.67	915.60
3	89	681	0.015474	54,746	378,377	354,172	810.57	902.53
4	88	51	0.001404	4,359	367,667	366,981	197.16	972.51
5	89	341	0.009666	45,041	355,236	325,471	909.80	929.37
6	89	652	0.020698	349,662	167,196	n,c,	11,441.50	12,889.43
7	89	384	0.014175	57,723	374,195	348,132	914.78	937.41
8	88	311	0.006293	53,637	340,209	314,587	1,043.77	1,051.26
9	89	371	0.016229	42,065	400,372	361,448	564.07	907.03

Table 6: Mip-Genint : P400-Centered anchor points computed

inst.	#C	$\| r \|_1$	$\overline{(\frac{r}{b})}$	$\| f \|_\infty$	$c(x_e^*)$	$c(x_c^*)$	cpu1	cpu2
0	87	2828	0.084211	1,305,639	0	n,c,	0	2,918.53
1	92	396	0.009010	28,458	710,935	726,760	708.37	1,403.16
2	91	784	0.044094	82,405	922,429	799,440	7,886.70	8,992.18
3	92	1162	0.062877	71,845	886,311	738,223	8,112.58	8,125.62
4	93	947	0.022324	85,780	786,639	653,674	8,408.69	9,599.45
5	91	1130	0.071981	97,896	821,374	710,752	8,685.62	8,734.22
6	92	449	0.013588	33,610	705,644	719,061	798.78	1,397.02
7	92	1073	0.018339	66,527	752,408	646,595	2,406.83	2,503.87
8	93	71	0.001709	7,644	653,604	658,817	2,467.36	2,522.30
9	92	341	0.023868	42,698	726,294	743,095	1,089.28	1,409.57

Table 7: Mip-Genint : P1000-Centered anchor points computed

inst.	relative gap value
1	0,00068582
2	0,00157929
3	0,00347841
4	0,00220182
5	0,00281687
6	0,00537527
7	0,00178818
8	0,00117450
9	0,00325376

Table 8: Mip-Genint : P50: majoration of the optimality gap for (IP_0)

inst.	#C	$\| r \|_1$	$\overline{(\frac{r}{b})}$	$\| f \|_\infty$	$c(x_e^*)$	$c(x_c^*)$	cpu1	cpu2
0	47	3	0.001129	185	61685	61914	0	17.45
1	48	162	0.104919	11698	41309	60803.3	0.73	17.16
2	48	269	0.102772	14837	48352	69841.7	12.25	16.98
3	49	213	0.017317	10347	48676	63506.1	4.79	16.84
4	48	170	0.019851	6681	54018	57724.9	0.49	16.88
5	48	236	0.104269	10818	49892	63332.6	0.33	16.85
6	49	229	0.026023	20836	39110	75977.6	1.76	17.03
7	49	119	0.012081	9002	43045	56705.6	6.98	16.84
8	50	0	0	0	61786	61813.4	0	17.89
9	48	248	0.096160	11362	51547	65770	5.55	16.96

Table 9: Mip-Saint: P50-Centered anchor points not computed

inst.	#C	$\| r \|_1$	$\overline{(\frac{r}{b})}$	$\| f \|_\infty$	$c(x_e^*)$	$c(x_c^*)$	cpu1	cpu2
0	55	424	0.054782	16057	106912	114107	58.61	81.08
1	58	2	0.000512	206	86456	86662.6	0	75.28
2	55	509	0.412343	23781	91710	115129	35.30	70.54
3	56	591	0.062302	27630	105035	109960	22.07	76.75
4	56	542	0.111047	22016	121485	127865	25.69	75.24
5	56	630	0.081401	21935	126165	116202	29.87	75.77
6	55	400	0.127585	16860	101454	94706.2	59.45	72.53
7	57	541	0.083691	19619	106478	102163	60.65	66.78
8	55	458	0.132604	14833	98473	100860	4.35	104.21
9	56	6	0.000711	168	126340	126541	0	88.49

Table 10: Mip-Saint: P100-Centered anchor points not computed

inst.	#C	$\| r \|_1$	$\overline{(\frac{r}{b})}$	$\| f \|_\infty$	$c(x_e^*)$	$c(x_c^*)$	cpu1	cpu2
0	78	644	0.032484	44673	197224	173949	18.83	474.94
1	79	427	0.013792	49771	230590	209466	198.83	388.15
2	78	797	0.043722	41306	197963	163864	20.67	360.52
3	78	768	0.021196	40870	199769	197384	24.43	445.58
4	78	348	0.036567	44161	227592	184814	18.80	437.76
5	78	811	0.218815	45497	198608	166255	19.65	398.59
6	79	625	0.031250	48274	220914	197569	124.97	413.46
7	78	585	0.071031	30340	238643	201161	44.17	396.69
8	78	416	0.019053	35502	198684	167897	21.46	415.81
9	79	450	0.015244	45357	208701	206297	106.60	541.67

Table 11: Mip-Saint: P200-Centered anchor points not computed

inst.	#C	$\| r \|_1$	$\overline{(\frac{r}{b})}$	$\| f \|_\infty$	$c(x_e^*)$	$c(x_c^*)$	cpu1	cpu2
0	88	633	0.202875	79050	411856	348043	744.11	2014.20
1	88	622	0.140178	87906	482325	388219	141.71	1873.73
2	88	1082	0.051498	68055	469114	331991	89.33	1606.93
3	88	1060	0.027894	103910	409052	354172	89.42	1683.19
4	88	750	0.048213	64253	478653	366981	93.85	1753.50
5	88	674	0.012415	81902	421722	325471	109.77	1649.10
6	88	931	0.041389	148816	359332	-19629	108.77	13990.46
7	88	618	0.122516	76665	435437	348132	930.43	1740.57
8	88	683	0.087909	60593	430649	314587	435.07	2154.56
9	88	504	0.014266	59767	452813	361448	390.28	1927.98

Table 12: Mip-Saint: P400-Centered anchor points not computed

inst.	#C	$\| r \|_1$	$\overline{(\frac{r}{b})}$	$\| f \|_\infty$	$c(x_e^*)$	$c(x_c^*)$	cpu1	cpu2
0	92	1724	0.072747	202846	1.16003e+06	-50408	22898.30	85671.38
1	92	1563	0.039506	99090	1.16631e+06	726760	12840.80	14130.53
2	91	959	0.078719	67964	1.02159e+06	799440	677.03	15294.65
3	91	1579	0.126194	135386	1.0474e+06	738223	2288.25	14592.83
4	92	1844	0.032328	109693	1.0444e+06	653674	5282.04	15161.63
5	93	1480	0.034731	135340	1.11325e+06	710752	6427.64	13816.83
6	92	1562	0.055210	140882	1.18096e+06	719061	1769.13	14920.57
7	92	1641	0.057464	164130	1.08133e+06	646595	8432.37	14587.03
8	91	5	0.001301	418	658530	658817	0	11729.74
9	91	1091	0.038650	74412	877050	743095	2192.23	15231.70

Table 13: Mip-Saint: P1000-Centered anchor points not computed

inst.	#C	$\| r \|_1$	$\overline{(\frac{r}{b})}$	$\| f \|_\infty$	$c(x_e^*)$	$c(x_c^*)$	cpu1	cpu2
0	48	131	0.040857	7180	52551	61914	5.65	17.43
1	48	267	0.057681	10751	43029	60803.3	1.34	17.23
2	48	205	0.027274	8309	58766	69841.7	4.94	37.08
3	49	336	0.027317	11900	56258	63506.1	5.76	17.20
4	49	253	0.022896	9049	45829	57724.9	3.84	17.26
5	49	223	0.020744	10619	41370	63332.6	13.83	37.03
6	49	335	0.038068	15801	50110	75977.6	8.98	36.95
7	48	273	0.026818	13122	36807	56705.6	4.70	37.34
8	50	0	0	0	61786	61813.4	0	18.30
9	48	345	0.068012	13422	47044	65770	15.23	37.38

Table 14: Mip-Saint: P50-Centered anchor points computed

inst.	#C	$\| r \|_1$	$\overline{(\frac{r}{b})}$	$\| f \|_\infty$	$c(x_e^*)$	$c(x_c^*)$	cpu1	cpu2
0	56	367	0.072841	13305	108101	114107	49.49	83.50
1	58	2	0.000512	206	86456	86662.6	0	77.54
2	57	525	0.128843	19771	109466	115129	47.27	131.40
3	56	600	0.080019	14706	120501	109960	26.67	78.83
4	56	494	0.075415	24482	120298	127865	33.24	75.96
5	56	566	0.124209	22459	114849	116202	41.62	140.43
6	55	381	0.150203	14738	104460	94706.2	50.69	74.74
7	55	472	0.103250	22438	98801	102163	34.20	68.78
8	55	446	0.106431	13271	97722	100860	46.13	184.39
9	56	6	0.000711	168	126340	126541	0	158.66

Table 15: Mip-Saint: P100-Centered anchor points computed

inst.	#C	$\| r \|_1$	$\overline{(\frac{r}{b})}$	$\| f \|_\infty$	$c(x_e^*)$	$c(x_c^*)$	cpu1	cpu2
0	79	477	0.009743	44500	190435	173949	743.84	898.05
1	79	581	0.021360	46843	230412	209466	121.29	755.63
2	78	506	0.026060	46728	188732	163864	129.12	359.37
3	79	423	0.009666	45583	220702	197384	134.18	852.94
4	78	321	0.041696	32981	227604	184814	165.33	845.06
5	78	533	0.022982	50804	189199	166255	33.65	767.66
6	78	339	0.017587	48412	225360	197569	35.95	801.94
7	78	702	0.053239	50455	234532	201161	39.98	760.89
8	78	486	0.029500	40520	178650	167897	75.87	799.42
9	79	561	0.019004	50591	211365	206297	190.38	537.89

Table 16: Mip-Saint: P200-Centered anchor points computed

inst.	#C	$\| r \|_1$	$\overline{(\frac{r}{b})}$	$\| f \|_\infty$	$c(x_e^*)$	$c(x_c^*)$	cpu1	cpu2
0	88	932	0.035025	90780	394478	348043	249.28	2017.57
1	88	851	0.014748	59944	487210	388219	144.57	1884.77
2	89	778	0.016466	59878	453446	331991	882.74	2852.46
3	89	682	0.015496	72342	464493	354172	123.62	2117.88
4	88	773	0.015368	55397	480973	366981	102.54	1971.66
5	89	613	0.032904	67748	426962	325471	693.49	1853.67
6	89	1012	0.056222	136875	371455	-19629	876.69	14284.91
7	89	594	0.020561	74307	498328	348132	693.12	1757.91
8	88	497	0.010013	68885	411242	314587	91.60	2163.33
9	88	846	0.109124	78439	475765	361448	89.74	1933.66

Table 17: Mip-Saint: P400-Centered anchor points computed

inst.	#C	$\| r \|_1$	$\overline{(\frac{r}{b})}$	$\| f \|_\infty$	$c(x_e^*)$	$c(x_c^*)$	cpu1	cpu2
0	93	1199	0.033691	201427	1.14316e+06	-50408	3066.99	135024.79
1	92	1234	0.028961	90967	1.22011e+06	726760	9711.04	14263.55
2	92	1932	0.479501	109536	1.17753e+06	799440	11732.27	23664.36
3	92	1282	0.082181	97842	1.04825e+06	738223	2314.25	20879.51
4	92	1729	0.029217	121986	1.00985e+06	653674	4522.74	15011.68
5	93	1820	0.042325	202156	1.0953 e+06	-50551	27492.29	132883.63
6	93	2001	0.035794	157337	1.16911e+06	719061	4987.74	22780.26
7	91	1703	0.072009	98421	1.07205e+06	646595	1381.50	14625.70
8	91	5	0.001301	418	658530	658817	0	12133.55
9	93	1654	0.058475	106282	1.09767e+06	743095	22575.21	23569.44

Table 18: Mip-Saint: P1000-Centered anchor points computed