Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

TIMO BERTHOLD

Primal MINLP Heuristics in a nutshell

ZIB-Report 13-42 (August 2013)

Herausgegeben vom
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Takustraße 7
D-14195 Berlin-Dahlem

Telefon: 030-84185-0
Telefax: 030-84185-125

e-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

Primal MINLP Heuristics in a nutshell

Timo Berthold*

21/Aug/2013

Abstract

Primal heuristics are an important component of state-of-the-art codes for mixed integer nonlinear programming (MINLP). In this article we give a compact overview of primal heuristics for MINLP that have been suggested in the literature of recent years. We sketch the fundamental concepts of different classes of heuristics and discuss specific implementations. A brief computational experiment shows that primal heuristics play a key role in achieving feasibility and finding good primal bounds within a global MINLP solver.

1 Introduction

Optimization problems that feature, at the same time, nonlinear functions as constraints and integrality requirements for the variables are arguably among the most challenging problems in mathematical programming. This article gives an overview on existing heuristic approaches to find good feasible solutions for these so-called *MINLPs*.

Definition 1.1 (MINLP) *A* mixed integer nonlinear program (MINLP) *is an optimization problem of the form*

$$\begin{array}{lll} \min & c^{\mathsf{T}} x & \\ \text{s.t.} & g_i(x) \leq 0 & \textit{for all } i \in \mathcal{M} \\ & x_j \in \mathbb{Z} & \textit{for all } j \in \mathcal{I}, \end{array}$$

where $\mathcal{I} \subseteq \mathcal{N} := \{1, \ldots, n\}$ *is the index set of the integer variables,* $c \in \mathbb{R}^n$*, and* $g_i : \mathbb{R}^n \to \mathbb{R}$ *for* $i \in \mathcal{M} := \{1, \ldots, m\}$*.*

There are many subclasses of MINLPs; in this article, we will be particularly concerned with the following: *convex MINLPs*, for which all constraint functions $g_i, i \in \mathcal{M}$, are convex, *mixed integer quadratically constrained programs (MIQCPs)*, for which all constraint functions are quadratic, *mixed integer linear programs (MILPs)*, for which all constraint functions are linear, *nonlinear programs (NLPs)*, for which all variables are continuous, and *linear programs (LPs)*, for which the constraints are linear and all variables are continuous.

For MILPs, it is well-known that general-purpose *primal heuristics* like the feasibility pump [2, 18, 20] are able to find high-quality solutions for a wide range of problems. A *primal heuristic* is, roughly speaking, an incomplete algorithm that aims at finding high-quality feasible solutions

*Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany, `berthold@zib.de`

quickly. In general, it is neither guaranteed to be successful, nor does it provide any additional information, such as a dual bound on the solution quality.

For MINLPs, research in the last five years has shown an increasing interest in primal heuristics [6, 8, 9, 10, 12, 13, 15, 22, 24, 25]. The goal of this article is to provide a brief overview on the cited work. We focus on methods that have been developed for the application inside a global solver such as Baron, Bonmin, Couenne, or Scip. In such an environment, it is often worth sacrificing success on a number of instances for a significant saving in average running time. One way to do so are "fast fail" strategies that take the most crucial decisions in the beginning and in a defensive fashion such that if the heuristic aborts, it will not have consumed much running time. Furthermore, we restrict ourselves to primal heuristics that have been specifically developed and tested for MINLPs; we do not cover the manifold ideas to apply metaheuristics to global optimization problems.

We partition our survey by the main concepts on which the reviewed algorithms are based. Nonlinear extensions of the feasibility pump [18] are discussed in Section 2, large neighborhood search heuristics are introduced in Section 3, other ideas, such as rounding and diving, are treated in Section 4. Section 5 presents a computational evaluation of primal heuristics implemented within the MINLP solver Scip. Finally, conclusions are drawn in Section 6.

2 Feasibility Pumps

The fundamental idea of all Feasibility Pump [18] algorithms is to construct two sequences of points that hopefully converge to a feasible solution of a given mathematical programming problem. One sequence consists of points that are feasible for a continuous relaxation (e.g., an NLP relaxation of an MINLP), but possibly integer infeasible. The other sequence consists of points that are integral (for the integer variables), but might violate the imposed constraints. The next point of one sequence is always generated by minimizing the distance to the last point of the other sequence, using different distance measures in both cases (e.g., the ℓ_1 and the ℓ_2 norm). We refer to the process of constructing an integral point from a constraint feasible point as the *rounding step* and to the process of finding a new point that fulfills the continuous relaxation as the *projection step*.

Bonami et al. [12] and Bonami and Gonçalves [13] present the first two versions of a Feasibility Pump for MINLPs. Both teams of authors consider convex MINLPs and implement their ideas in Bonmin [11].

The paper [13] is probably the closest to the original Feasibility Pump for MILPs. It performs a simple rounding to the nearest integer in the rounding step and solves a convex NLP relaxation with an ℓ_1 objective for the projection step.

In [12], the authors suggest using an ℓ_2 norm as objective for the projection step. The most signigficant difference to [13], however, is the implementation of the rounding step. Instead of performing an instant rounding to the nearest integer, they solve a MILP relaxation based on an outer approximation [17] of the underlying MINLP. This has an important effect w.r.t. the main weakness of Feasibility Pump algorithms: cycling. For convex MINLPs, it is always possible to avoid cycling by adding a no-good cut to the auxiliary MILP.

The particular difficulty addressed by D'Ambrosio et al. in [15] is that of handling the nonconvex NLP relaxation when adapting the algorithm of [12] to nonconvex constraints. The authors suggest using a stochastic multi-start approach, feeding the NLP solver with multiple randomly generated starting points, and solving the NLP to local optimality. In the event that this does not lead to a feasible solution, a final NLP is solved, in which the integer variables are fixed and the original objective is reinstalled on the continuous variables. To avoid cycling, their algorithm provides the MILP solver with a tabu list of previously used solutions.

3 Large Neighborhood Search

The main idea of *large neighborhood search* (LNS) is to define a neighborhood of "good" solution candidates centered at a particular reference point – typically the incumbent solution. The neighborhood is explored by solving an auxiliary MINLP, which is constructed by restricting the feasible region of the original MINLP by additional constraints and variable fixings. LNS is a common paradigm for MILP heuristics, e.g., RINS [16], which defines a neighborhood by fixing variables which coincide in the incumbent and the LP optimum, or Local Branching [19], which searches the neighborhood of solutions that differ in at most k variables from the incumbent.

Bonami and Gonçalves describe an extension of the RINS heuristic to convex MINLPs [13]. They use an optimum of the NLP relaxation as a second reference solution besides the incumbent.

Nannicini, Belotti, and Liberti introduce a Local Branching heuristic for nonconvex MINLPs [25]. It solves a MILP that is derived from a linear relaxation of the original MINLP, the integrality constraints, and a Local Branching constraint. Subsequently, an NLP local search is performed by fixing the integer variables to the values from the Local Branching MILP's incumbent (which is not necessarily feasible for the original MINLP) and solving the resulting continuous problem.

In [9], Berthold et al. suggest a generic way of generalizing LNS heuristics from MILP to MINLP, for the first time presenting nonlinear versions of Crossover [4, 26] and the DINS [21] heuristic.

Berthold presents RENS [6], an LNS algorithm that optimizes over the set of feasible roundings of a relaxation solution. To this end, integer variables that take an integral value in the relaxation solution are fixed to that value, for others, the bounds are changed to the two nearest integers.

In [8], Berthold and Gleixner introduce *Undercover*, an LNS start heuristic for MINLP that explores a linear subproblem which is obtained by fixing as small a subset of variables as possible. The set of variables to be fixed is determined by solving a vertex covering problem. Although general in nature, this approach works best for MIQCPs.

The RECIPE algorithm described in [22] falls into the category of variable neighborhood search heuristics: it iteratively explores different neighborhoods, updating the neighborhood definition after each iteration.

4 Rounding, Diving, and MILP heuristics

Rounding, diving, and propagation heuristics are kind of "folklore": Most solvers and many custom codes use them, but there are few publications on this topic.

Bonami and Gonçalves present computational results for NLP-based diving heuristics [13]. Their algorithm solves a convex NLP relaxation, fixes several variables (with variable selection rules referred to as Fractional Diving and Vectorlength Diving in [4]), and iterates this process. They further tested solving a final sub-MINLP as soon as all fractional variables exclusively belong to linear constraints. Mahajan et al. [23] suggest a diving algorithm that uses quadratic programming relaxations.

Nannicini and Belotti present *iterative rounding* [24], which is a mixture of diving and variable neighborhood search. It solves a series of auxiliary MILPs to generate integer points near an initial optimal solution of an NLP relaxation. In each iteration, the feasible region of the MILP gets contracted further by outer approximation and no-good cuts.

A popular approach for solving MINLPs is to use an outer approximation generated by linearization of convex constraints and linear underestimation of nonconvex constraints. Having an outer approximation at hand, one might employ MILP primal heuristics to the outer approximation LP plus the integrality constraints. In particular for heuristics that are computationally very cheap, such as rounding and propagation heuristics [3], this is a valid strategy. Applying MILP heuristics to such a "MILP relaxation" typically produces points that are integral, valid for the LP outer approximation, but violate one or more nonlinear constraints. Such points are natural candidates for an NLP local search as it is, e.g., described in [10, 22, 25]: the integer variables are fixed to their value in the (infeasible) reference solution and the resulting NLP is solved to local optimality.

5 Computational Results

To evaluate the impact of primal heuristics on the performance of a global MINLP solver, we conducted a computational experiment in which we compare the performance of the MINLP solver SCIP [1] when running with and without primal heuristics. We used SCIP version 3.0.1 compiled with SOPLEX 1.7.1 [28] as LP solver and IPOPT 3.11 [27] as NLP solver. SCIP does not run all of the described algorithms by default. It features Undercover, nonlinear versions of RENS and Crossover, an NLP local search, and many MILP heuristics, including a Feasibility Pump (for an overview, see [5]). As a test set, we chose the MINLPLIB [14], excluding instances which feature nonlinear functions that SCIP 3.0.1 cannot handle, e.g., trigonometric functions. The results were obtained on a cluster of 64bit Intel Xeon X5672 CPUs at 3.20GHz with 12 MB cache and 48 GB main memory, running an OPENSUSE 12.3 with a GCC 4.7.2 compiler. We imposed a time limit of one hour. Detailed results can be found in the Appendix.

Similar to the situation in MILP, the impact of primal heuristics on the overall running time was negligible. Both versions differed by less than one percent in shifted geometric mean. Furthermore, both variants solved 170 of the 252 test instances to optimality. The major difference occurs when considering the primal bound. For those instances which could not be solved within the time limit, the SCIP version without heuristics found a feasible solution in 35 cases, the one using primal heuristics in 58. The primal bound at termination was better for 48 instances when using primal heuristics, only for two instances it was worse. Consequently, the average *primal integral* [7] of both runs differed by about 50%.

6 Conclusion

Altogether, the results show that primal heuristics are essential to improve the primal bound observed during search, while they do not deteriorate the overall optimization process. Within a complete MINLP solver, heuristics give a user the immediate advantage of finding good solutions early, in particular for hard instances that are not solved within a given time limit.

The situation of computational MINLP today is, in many respects, comparable to that of computational MILP in the early nineties. Just as primal heuristics have become a substantial ingredient of nowadays MILP solvers, we can expect them to remain an active field of research for the MINLP community in the nearby future.

Acknowledgements

This research has been supported by the DFG Research Center MATHEON *Mathematics for key technologies* in Berlin. Many thanks go to Ambros M. Gleixner for proof-reading a first draft of this paper.

References

[1] T. Achterberg. SCIP: Solving Constraint Integer Programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

[2] T. Achterberg and T. Berthold. Improving the Feasibility Pump. *Discrete Optimization*, Special Issue 4(1):77–86, 2007.

[3] T. Achterberg, T. Berthold, and G. Hendel. Rounding and propagation heuristics for mixed integer programming. In D. Klatte, H.-J. Lüthi, and K. Schmedders, editors, *Operations Research Proceedings 2011*, pages 71–76. Springer Berlin Heidelberg, 2012.

[4] T. Berthold. Primal heuristics for mixed integer programs. Diploma thesis, Technische Universität Berlin, 2006.

[5] T. Berthold. Heuristics of the branch-cut-and-price-framework SCIP. In J. Kalcsics and S. Nickel, editors, *Operations Research Proceedings 2007*, pages 31–36. Springer-Verlag, 2008.

[6] T. Berthold. RENS – the optimal rounding. ZIB-Report 12-17, Zuse Institute Berlin, 2012. Submitted for publication.

[7] T. Berthold. Measuring the impact of primal heuristics. ZIB-Report 13-17, Zuse Institute Berlin, 2013. Accepted for publication in Operations Research Letters.

[8] T. Berthold and A. M. Gleixner. Undercover: a primal MINLP heuristic exploring a largest sub-MIP. *Mathematical Programming*, 2013. Online first publication.

[9] T. Berthold, S. Heinz, M. E. Pfetsch, and S. Vigerske. Large neighborhood search beyond MIP. In L. D. Gaspero, A. Schaerf, and T. Stützle, editors, *Proceedings of the 9th Metaheuristics International Conference (MIC 2011)*, pages 51–60, 2011.

[10] T. Berthold, S. Heinz, and S. Vigerske. Extending a CIP framework to solve MIQCPs. In J. Lee and S. Leyffer, editors, *Mixed Integer Nonlinear Programming*, volume 154 of *The IMA Volumes in Mathematics and its Applications*, pages 427–444. Springer, 2011.

[11] P. Bonami, L. Biegler, A. Conn, G. Cornuéjols, I. Grossmann, C. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wächter. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5:186–204, 2008.

[12] P. Bonami, G. Cornuéjols, A. Lodi, and F. Margot. A feasibility pump for mixed integer nonlinear programs. *Mathematical Programming*, 119(2):331–352, 2009.

[13] P. Bonami and J. Gonçalves. Heuristics for convex mixed integer nonlinear programs. *Computational Optimization and Applications*, 51:729–747, 2012.

[14] M. Bussieck, A. Drud, and A. Meeraus. MINLPLib – a collection of test models for mixed-integer nonlinear programming. *INFORMS Journal on Computing*, 15(1):114–119, 2003.

[15] C. D'Ambrosio, A. Frangioni, L. Liberti, and A. Lodi. A storm of feasibility pumps for nonconvex MINLP. *Mathematical Programming*, 136:375–402, 2012.

[16] E. Danna, E. Rothberg, and C. L. Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming*, 102(1):71–90, 2004.

[17] M. A. Duran and I. E. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36(3):307–339, 1986.

[18] M. Fischetti, F. Glover, and A. Lodi. The feasibility pump. *Mathematical Programming*, 104(1):91–104, 2005.

[19] M. Fischetti and A. Lodi. Local branching. *Mathematical Programming*, 98(1-3):23–47, 2003.

[20] M. Fischetti and D. Salvagnin. Feasibility pump 2.0. *Mathematical Programming Computation*, 1:201–222, 2009.

[21] S. Ghosh. DINS, a MIP improvement heuristic. In M. Fischetti and D. P. Williamson, editors, *Integer Programming and Combinatorial Optimization, 12th International IPCO Conference, Proceedings*, volume 4513 of *LNCS*, pages 310–323. Springer, 2007.

[22] L. Liberti, N. Mladenović, and G. Nannicini. A recipe for finding good solutions to MINLPs. *Mathematical Programming Computation*, 3:349–390, 2011.

[23] A. Mahajan, S. Leyffer, and C. Kirches. Solving mixed-integer nonlinear programs by QP-diving. Preprint ANL/MCS-2071-0312, Argonne National Laboratory, Mathematics and Computer Science Division, 2012.

[24] G. Nannicini and P. Belotti. Rounding-based heuristics for nonconvex MINLPs. *Mathematical Programming Computation*, 4(1):1–31, 2012.

[25] G. Nannicini, P. Belotti, and L. Liberti. A local branching heuristic for MINLPs. *ArXiv e-prints*, 2008.

[26] E. Rothberg. An evolutionary algorithm for polishing mixed integer programming solutions. *INFORMS Journal on Computing*, 19(4):534–541, 2007.

[27] A. Wächter and L. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[28] R. Wunderling. *Paralleler und objektorientierter Simplex-Algorithmus*. PhD thesis, Technische Universität Berlin, 1996.

Appendix

Table 1: Impact of primal heuristics on overall solving process for MINLPLib instances. When an instance could not be solved to optimality, the primal bound after one hour is given.

	default		no heuristics	
Instance	Nodes	Time	Nodes	Time
4stufen	∞		∞	
alan	4	0.1	8	0.1
batchdes	4	0.1	3	0.1
batch	6	0.2	15	0.1
beuster	120277		∞	
cecil_13	-115593		-115562	
chp_partload	∞		∞	
contvar	∞		∞	
csched1a	-29484.1		-30249.9	
csched1	1066	1.2	2991	1.6
csched2a	-127600		-129127	
csched2	-150059		-119167	
detf1	12.8818		∞	
du-opt	232	0.8	294	0.5
du-opt5	75	0.3	74	0.2
eg_all_s	9.54458		∞	
eg_disc2_s	19.9971		∞	
eg_disc_s	12.8718		∞	
eg_int_s	100000		∞	
elf	580	0.9	822	0.8
eniplac	45	0.4	86	0.1
enpro48	248	1.1	9640	2.5
enpro48pb	3853	2.4	78	0.6
enpro56	124	1.0	33903	6.2
enpro56pb	85	1.0	263428	
ex1221	1	0.1	1	0.1
ex1222	1	0.1	1	0.1
ex1223a	1	0.1	1	0.1
ex1223b	1	0.1	6	0.1
ex1223	1	0.1	6	0.1
ex1224	6	0.1	18	0.1
ex1225	1	0.1	1	0.1
ex1226	1	0.1	5	0.1
ex1233	155011		155095	
ex1243	29	0.5	128	0.4
ex1244	392	1.2	325	0.7
ex1252a	1067	5.1	1048	10.9
ex1252	2368	10.5	1627	11.8
ex1263	360	0.8	406	0.7
ex1263a	308	0.3	190	0.1
ex1264	79	0.2	227	0.3
ex1264a	272	0.3	67	0.1
ex1265	132	0.3	89	0.3
ex1265a	105	0.1	112	0.1
ex1266	64	0.6	48	0.6

Table 1: Impact of primal heuristics on overall solving process for MINLPLib instances. When an instance could not be solved to optimality, the primal bound after one hour is given.

	default		no heuristics	
Instance	Nodes	Time	Nodes	Time
ex1266a	166	0.2	51	0.1
ex3	5	0.2	5	0.1
ex3pb	5	0.2	5	0.1
ex4	8	0.7	92	0.6
fac1	5	0.1	5	0.1
fac2	3.31837e+08		9.33091e+08	
fac3	6	0.2	15	0.1
feedtray2	1	0.2	107	0.7
feedtray	-13.2521		∞	
fo7_2	55625	23.9	58912	21.8
fo7_ar2_1	54859	22.5	28535	10.3
fo7_ar25_1	37990	16.3	50996	18.4
fo7_ar3_1	57084	23.5	49246	17.5
fo7_ar4_1	49263	22.3	52006	22.5
fo7_ar5_1	26616	14.2	31567	14.0
fo7	167645	74.7	173509	74.5
fo8_ar2_1	284758	116.8	240573	90.3
fo8_ar25_1	383103	145.4	250042	95.7
fo8_ar3_1	475974	109.9	69260	33.7
fo8_ar4_1	58378	29.0	116895	48.9
fo8_ar5_1	50974	30.3	138664	61.1
fo8	395824	211.8	387718	180.1
fo9_ar2_1	2267077	856.1	2557641	985.8
fo9_ar25_1	4699479	1925.4	6650732	2769.7
fo9_ar3_1	809661	347.7	129742	65.2
fo9_ar4_1	512918	319.9	1106804	623.9
fo9_ar5_1	1424595	708.2	803051	395.8
fo9	965136	537.5	1973156	1145.9
fuel	4	0.1	5	0.1
fuzzy	∞		∞	
gasnet	490	2.5	29431	185.9
gastrans	3	0.2	15	0.1
gbd	1	0.1	1	0.1
gear2	1009	0.7	1032	0.3
gear3	126	0.4	429	0.1
gear4	4159	1.1	349	0.1
gear	126	0.2	429	0.1
ghg_1veh	7.78141		3821915	1663.0
ghg_2veh	7.7709		8.20276	
ghg_3veh	7.76609		∞	
gkocis	1	0.1	4	0.1
hda	-5671.47		∞	
hmittelman	1	0.1	1	0.1
johnall	2	76.9	1	8.1
lop97ic	4610.73		5183.26	
lop97icx	4259.37		4341.74	
m3	14	0.1	19	0.1

Table 1: Impact of primal heuristics on overall solving process for MINLPLib instances. When an instance could not be solved to optimality, the primal bound after one hour is given.

	default		no heuristics	
Instance	Nodes	Time	Nodes	Time
m6	1117	1.7	3202	1.3
m7_ar2_1	5399	2.8	12292	3.9
m7_ar25_1	2456	1.5	1856	0.9
m7_ar3_1	7810	5.0	13694	4.8
m7_ar4_1	1121	1.7	1761	1.2
m7_ar5_1	9883	5.2	10575	4.2
m7	6951	4.6	2541	1.5
mbtd	8.91665		∞	
meanvarx	2	0.2	5	0.1
meanvarxsc	2	0.2	13	0.1
minlphix	-26110.7		259.52	
netmod_dol1	-0.560008		-0.560008	
netmod_dol2	171	46.7	172	47.8
netmod_kar1	345	6.0	268	5.4
netmod_kar2	345	6.0	268	5.2
no7_ar2_1	36320	19.4	28389	13.5
no7_ar25_1	95347	50.4	93286	44.2
no7_ar3_1	293714	137.2	384556	158.4
no7_ar4_1	217836	104.5	169311	75.5
no7_ar5_1	103159	54.0	138984	63.8
nous1	1.56707		1.82194	
nous2	4868	3.9	6424	4.1
nuclearva	∞		∞	
nuclearvb	∞		∞	
nuclearvc	∞		∞	
nuclearvd	∞		∞	
nuclearve	∞		∞	
nuclearvf	∞		∞	
nuclear25	-1.10613		∞	
nuclear25a	-1.08439		∞	
nuclear25b	-1.05757		∞	
nuclear49	∞		∞	
nuclear49a	∞		∞	
nuclear49b	-1.11491		∞	
nuclear14	∞		∞	
nuclear14a	∞		∞	
nuclear14b	-1.10637		∞	
nuclear10a	∞		∞	
nuclear10b	-1.15015		∞	
nuclear104	∞		∞	
nvs01	13	0.1	16	0.1
nvs02	1	0.1	1	0.1
nvs03	1	0.1	1	0.1
nvs04	1	0.1	13	0.1
nvs05	760	1.6	5985635	1372.6
nvs06	11	0.1	31	0.1
nvs07	1	0.1	1	0.1

Table 1: Impact of primal heuristics on overall solving process for MINLPLib instances. When an instance could not be solved to optimality, the primal bound after one hour is given.

	default		no heuristics	
Instance	Nodes	Time	Nodes	Time
nvs08	1	0.1	9	0.1
nvs09	2440089	762.5	-4.03417	
nvs10	1	0.1	1	0.1
nvs11	3	0.1	9	0.1
nvs12	5	0.1	13	0.1
nvs13	8	0.1	19	0.1
nvs14	1	0.1	1	0.1
nvs15	7	0.1	8	0.1
nvs16	5	0.1	4	0.1
nvs17	47	0.1	89	0.1
nvs18	23	0.1	52	0.1
nvs19	90	0.2	179	0.2
nvs20	105	0.5	384	0.8
nvs21	24	0.1	38	0.1
nvs22	12	0.2	48	0.1
nvs23	122	0.4	187	0.3
nvs24	114	0.4	187	0.3
o7_2	1464180	693.0	1168428	507.0
o7_ar2_1	428307	112.7	179099	82.7
o7_ar25_1	615855	341.7	629362	323.7
o7_ar3_1	1007602	520.0	1077211	512.3
o7_ar4_1	1857564	1015.3	1780200	882.3
o7_ar5_1	693767	333.0	787092	372.0
o7	4104431	2122.5	2673150	1276.1
o8_ar4_1	243.071		243.071	
o9_ar4_1	236.138		236.138	
oaer	1	0.1	3	0.1
oil2	-0.73326		∞	
oil	-0.932494		∞	
ortez	34	0.5	35	0.1
parallel	924.163		615454	2712.5
pb302035	4.28828e+06		4.8551e+06	
pb302055	4.33141e+06		5.51106e+06	
pb302075	4.55301e+06		6.35817e+06	
pb302095	5.92856e+06		6.58091e+06	
pb351535	5.63016e+06		6.34354e+06	
pb351555	5.28901e+06		6.16652e+06	
pb351575	6.84379e+06		8.49873e+06	
pb351595	7.54834e+06		1.02975e+07	
prob02	1	0.1	1	0.1
prob03	1	0.1	1	0.1
procsel	1	0.1	2	0.1
product2	1	3.1	∞	
product	10366	23.6	16144	41.2
pump	1036	6.5	1103	9.6
qapw	392424		405072	
qap	415192		415484	

Table 1: Impact of primal heuristics on overall solving process for MINLPLib instances. When an instance could not be solved to optimality, the primal bound after one hour is given.

	default		no heuristics	
Instance	Nodes	Time	Nodes	Time
ravem	42	0.8	163	0.5
ravempb	42	1.1	152	0.5
risk2b	184	0.6	467	0.3
risk2bpb	8	0.1	11	0.2
saa_2	12.8818		∞	
sep1	21	0.4	35	0.1
space25	∞		∞	
space25a	638.828		∞	
space960	∞		∞	
spectra2	14	0.8	67	0.7
spring	83	0.5	135	0.1
st_e13	1	0.1	3	0.1
st_e14	1	0.1	6	0.1
st_e15	1	0.1	1	0.1
st_e27	1	0.1	1	0.1
st_e29	6	0.1	18	0.1
st_e31	1891	1.3	1098	0.7
st_e32	7935	10.7	11331	13.5
st_e35	64867.7		14628	15.3
st_e36	333	0.8	524	0.3
st_e38	3	0.1	15	0.1
st_e40	29	0.1	22	0.1
st_miqp1	1	0.1	1	0.1
st_miqp2	1	0.1	5	0.1
st_miqp3	5	0.1	3	0.1
st_miqp4	1	0.1	1	0.1
st_miqp5	1	0.1	1	0.1
stockcycle	42555	207.8	69920	337.4
st_test1	1	0.1	1	0.1
st_test2	1	0.1	1	0.1
st_test3	1	0.1	1	0.1
st_test4	1	0.1	1	0.1
st_test5	1	0.1	1	0.1
st_test6	1	0.1	1	0.1
st_test8	1	0.1	1	0.1
st_testgr1	30	0.1	55	0.1
st_testgr3	14	0.1	9	0.1
st_testph4	1	0.1	1	0.1
super1	∞		∞	
super2	∞		∞	
super3	∞		∞	
super3t	-0.653203		∞	
synheat	154997		∞	
synthes1	5	0.1	5	0.1
synthes2	5	0.1	6	0.1
synthes3	7	0.1	495622	46.9
tln2	1	0.1	1	0.1

Table 1: Impact of primal heuristics on overall solving process for MINLPLib instances. When an instance could not be solved to optimality, the primal bound after one hour is given.

	default		no heuristics	
Instance	Nodes	Time	Nodes	Time
tln4	4298	1.8	3168	1.2
tln5	349271	173.0	16705	8.2
tln6	15.3		15.3	
tln7	15.1		15.7	
tln12	91.3		118.8	
tloss	120	0.2	203	0.2
tls2	18	0.1	13	0.1
tls4	18131	34.1	36715	65.7
tls5	10.3		10.9	
tls6	15.9		16	
tls7	17.2		22.1	
tls12	∞		∞	
tltr	11	0.2	23	0.2
uselinear	∞		∞	
util	82	0.2	379	0.1
waste	656.325		742.053	
water4	926.947		1002.75	
waterx	957.908		2854.37	
waterz	929.784		1027.87	
feas. sol.	228		205	
better obj.	48		2	
all optimal				
sh. geom. mean	842	9.6	988	9.5
arithm. mean	149 273	72.4	152 835	73.2