GAMS MINLP Solver Comparisons and Some Improvements to the AlphaECP Algorithm

Toni Lastusilta

Process Design and Systems Engineering Laboratory
Department of Chemical Engineering
Division for Natural Sciences and Technology
Åbo Akademi University

Åbo, Finland 2011

If you can't explain it simply, you don't understand it well enough.

Albert Einstein

ISBN 978-952-12-2653-3

Electronic version
Åbo 14.11.2011

Preface

The work related to this thesis was carried out during the period 2007-2011 at the Process Design and Systems Engineering Laboratory, at the Department of Chemical Engineering at Åbo Akademi University. Most importantly, I want to express my gratitude to my supervisor and head of the Laboratory Prof. Tapio Westerlund for his excellent support and for giving me this research opportunity. Furthermore, I want to give a special thanks to Michael Bussieck at GAMS Development Corporation for his support and insights. I also want to thank Dr Lazaros Papageorgiou for his mentoring during my visit to University College of London and Frank Pettersson for his excellent introductory course to this research field.

I am grateful to all my colleagues at the Process Design and Systems Engineering Laboratory for forming a positive and motivating working atmosphere. I want to thank Anders Skjäl, Andreas Lundell and Cataldo de Blasio for our interesting conversations.

The financial support from the Academy of Finland, Research Institute of Åbo Akademi Foundation and the Finnish Technology Development Centre (TEKES) are gratefully acknowledged. Without their economical support this thesis could not have been completed.

During my work I have gained invaluable skills in optimization and its implementation, as well as, a logical and systematic approach towards problem solving. Moreover, the experiences gained during my visit to University College of London and Universidad Simón Bolívar has broadened my perspectives on life. However, with this study I see that I may bring the industrial world and academia one step closer to each other, because even the best invention is useless if no one uses it.

Finally I want to thank my family and friends who has been there for me during my studies. Especially, I want to thank Harri Aarnio for his constructive questions and, also, Tuomo Lyyrtö and Ari for providing me with well needed distractions during the intensive times of my research. I also want to acknowledge the support of my dear friends Peggy Reich and Gunta Kursiša.

Åbo, September 2011

Toni Lastusilta

Abstract

This PhD thesis studies the solving capability of some solvers for optimization problems, as well as presenting some improvements to one of the solvers.

The optimization problems that foremost are studied can be described by a model of Mixed Integer Non-Linear Programming (MINLP) class. The class of MINLP problems can be characterized as difficult and can be found in several industries and it is, therefore, well motivated to study which solvers effectively solve MINLP models. Only since 2003 have methods become available to solve these problems to global optimality, i.e. a guaranteed best possible solution, on optimization platforms like GAMS. Even though a method can solve a model to global optimality, it is of little use if the solution is not at hand when a decision needs to be made. Therefore, a solver which provides a good solution in reasonable time can, in some cases, be more useful. The study shows that all compared solvers have their strengths and weaknesses and that no solver is generally better than all the others.

The solver improvements are implemented in GAMS/AlphaECP, which is based on the Extended Cutting Plane (ECP) method. The improvements significantly increase solving capability and can be characterized as sensible. The enhancements include the integration of AlphaECP with another solving method, the implementation of a heuristic which can be activated when the solver cannot otherwise solve a problem and the use of existing information in an improved way.

Svenskt sammandrag

I denna doktorsavhandling undersöks lösningsförmågan hos ett antal lösare för optimeringsproblem, samt vissa förbättringar till en av lösarna presenteras.

Optimeringsproblemen som främst undersöks kan beskrivas med hjälp av en modell som innehåller ickelinjäriteter och heltalsvariabler, m.a.o. en Mixed Integer Non-Linear Programming (MINLP) modell. Klassen av MINLP problem kan karaktäriseras som svårlöst och förekommer inom ett flertal industriområden. Det är därför välmotiverat att undersöka vilka metoder som mest effektivt löser MINLP modeller. Först efter år 2003 har metoder att lösa ett MINLP problem till globalt optimum, d.v.s. den bästa möjliga lösningen, blivit tillgängliga i optimeringssystem såsom GAMS. Även om en lösare kan lösa ett problem till globalt optimum är den värdelös om lösningen inte är tillhands när ett beslut måste fattas. Därför kan en lösare som ger en bra lösning inom rimlig tid, i vissa fall vara mer användbar. Undersökningen visar att alla lösare har sina styrkor och svagheter och att det inte finns någon metod som är bättre än de andra för alla MINLP problem.

Lösaren som förbättringarna utförts på heter GAMS/AlphaECP och baserar sig på skärplansmetoden Extended Cutting Plane (ECP). Förbättringarna ökar avsevärt lösarens lösningsförmåga och kan karaktäriseras som heuristiska. Förbättringarna omfattar främst integreringen av AlphaECP med en annan lösningsmetod, implementeringen av en heuristik som kan aktiveras ifall lösaren inte annars klarar av att lösa problemet och användningen av den tillgängliga informationen på ett förbättrat sätt.

Contents

List of figures

List of tables

CHAPTER 1

Introduction

In this thesis we study Mixed Integer Non-Linear Programming (MINLP) problems. We have mainly worked on two topics: comparison of the performance of some MINLP solvers in the General Algebraic Modeling System (GAMS) and some improvements to the AlphaECP solver. GAMS is a high-level modeling system for mathematical programming and optimization and currently includes 9 MINLP solvers.

The purpose of section 1.4 and 1.5 is to give a loose, but easy-to-understand, description of how MINLP problems can be solved. This introduction aims to support the understanding of the performance differences in section 2.4.

The General Algebraic Modeling System (GAMS) [GAMS, 2011] optimization platform is used. However, other platforms are also available: AIMMS [Bisschop, 2011], AMPL [Fourer et al., 1990] and LINDO [LINDO, 2011]. We chose GAMS because of its good selection of solvers and ease of finding large test sets of problems.

1.1 List of publications

This thesis is partly based on the following publications:

I) T. Lastusilta, M. R. Bussieck, and T. Westerlund. Comparison of Some High-Performance MINLP Solvers. In *Chem. Eng. Trans.*, volume 11, pages 125–130, 2007

II) T. Lastusilta, M. R. Bussieck, and T. Westerlund. An Experimental Study of GAMS/AlphaECP MINLP Solver. *Ind. Eng. Chem. Res.*, 48:7337–7345, 2009a

III) T. Lastusilta, L.G. Papageorgiou, and T. Westerlund. A Comparative Study of Solving the Problem of Module Identification in a Complex Network. In *Chem. Eng. Trans.*, volume 24, pages 319–324, 2011

IV) T. Lastusilta and T. Westerlund. A Comparative Study of Solving Quadratic Assignment Problems Using Some Standard MINLP Solvers. In *SIMULTECH 2011 1st International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, pages 409–412, 2011

In addition to the above mentioned papers, the paper Lastusilta et al. [2009b] has been published during the research period. This paper presents a framework to solve an industrial scheduling problem. The framework was developed to solve a scheduling problem for a Finnish supplier of packaging material. An MILP model was used in conjunction with a moving time window to enable the scheduling of a large set of orders.

- T. Lastusilta, O. Frankenhaeuser, F. Pettersson, and T. Westerlund. An Optimization Framework for Solving a Large Scale Scheduling Problem. In *European Symposium on Computer Aided Process Engineering*, volume 19, pages 525–529, 2009b

1.2 Motivation for the research

MINLP optimization problems can be found from all disciplines of science and applied science: economics in social science, engineering in applied science, operation research as an interdisciplinary science, etc.. The research areas include: finance, structural optimization, production management, logistics, and parameter estimation [Floudas, 2000]. Optimization applications investigated at Åbo Akademi University are, for example: allocation problems which include plant layout, scheduling and trim loss [Westerlund, 2005], [Jernström, 2006], [Harjunkoski, 1997]; pump configuration [Pettersson, 1994]; chromatographic separation [Emet, 2004]; heat exchanger network problems [Björk, 2002]. The solving methods available can be divided into three groups within global optimization: deterministic, stochastic and heuristic. Deterministic methods are considered in this thesis and, typically, an appropriate deterministic method can provide information about the solution quality. Stochastic algorithms, for example a Monte-Carlo simulation, or a heuristic approach [Silver, 2004], for example an evolutionary algorithm, can typically not provide information about the solution quality. Particularly for large-scale problems, but also for other difficult problems, even finding a solution, is sometimes difficult. A heuristic approach might, in this case, be more useful when taking into consideration a reasonable solution time. For example, in Farkas et al. [2006] the problem of distillation column synthesis is solved by a Case-Based Reasoning approach and in Drezner et al. [2005] a review of heuristic approaches applied on Quadratic Assignment Problems (QAP) is given. Clearly, the heuristic solution approaches are also an important field of study.

An important and challenging class of optimization problems is the class of Mixed Integer Non-Linear Programming (MINLP) problems. Only in the last five years have the theoretical advances to solve MINLP problems to global optimality resulted in solvers that are available on standard optimization platforms. Traditionally, MINLP solvers have only been able to solve convex MINLP problems (a subset) to proven global optimality, otherwise they have worked as a local solver, i.e. if a solution is found, it is the best one in some sense, but can also be seen as a heuristic solver[1]. Currently, the global MINLP solvers solve a sequence of convex subproblems in order to solve an MINLP problem and typically require much more computational effort than a local MINLP solver. Especially for a large scale MINLP problem, for example an industrial application, the computational effort is already a key issue and the solution time required by a global solver can easily become unacceptably long. Furthermore, no method has outperformed the other methods in all cases of convex or global MINLP optimization. The research in methods concerning convex MINLP optimization, as well as, other heuristic approaches has, therefore, its place in the scientific community.

The solver comparisons hold information about which MINLP solvers are good at solving a particular type of problem. They might also provide a rough estimate on the solution time in relation to problem size for a particular type of problem[2]. Note that

[1] A convex MINLP method can in some sense be considered as heuristic, since it does not provide any information about the solution quality and, moreover, the solution can be starting point dependent.

[2] The relation can only be compared for similar problems because the type of non-convexity may

the way the solver performance is measured can give a very different perception of the solving capability of a particular solver and, therefore, it is important to state how the comparison is performed. For example, a local MINLP solver that finds a solution quickly, but is unable to improve the solution if more time is available, performs well if a short time limit is used, but not so well with a longer time limit compared to other solvers. To solve a particular optimization problem on a standard optimization platform there are two important phases. The first phase is to create a model and the second phase is to choose an appropriate solver by understanding the strengths of different solvers. In this thesis we do not consider the modeling aspect, but concentrate on the solvers and solving techniques. One goal of this thesis is to point out the main differences in some MINLP solvers in GAMS and give an overview of their performance. This can assist a novice modeler to solve a particular problem by choosing a well suited solver, without years of experience in optimization. Another goal is to present some algorithmic advances in AlphaECP, an MINLP solver, and demonstrate the improved performance.

1.2.1 The objective of the research

This thesis aims to reveal if there exist a commercial MINLP solver that is universally faster and finds a higher quality solution than any of the other solvers. It intends to uncover the difficulties of performing a fair comparison and, furthermore, explain the major enhancements implemented in the AlphaECP algorithm. The research is carried out by performing solver comparisons with very extensive problem libraries.

1.3 Definitions

We now present some terminology which is used throughout this thesis:

Point Variable levels.

Fixing of variable levels Variable levels are considered as constants and can not be changed by the sub-solver.

Constraint A function that needs to satisfy a particular value.

Feasible region All combinations of variable levels that satisfy all constraints, also called solution space.

Start point Initial variable levels given to a solver.

Cutting Plane A linearization of a non-linear function.

Cut Abbreviation for cutting plane.

Cut generation point The variable levels used to derive a cut.

Violation The magnitude by which a constraint is violated.

Maximum violation The violation value for the most violating constraint at a specific point.

---

heavily impact the solution time.

Domain violation A function cannot be evaluated, for example, $\sqrt{-1}$.

Best solution known The best solution known for a model and not necessarily found by any solver in a particular comparison.

For the optimization problems considered in this thesis, an optimization model has been derived and the terms optimization problem and optimization model are used interchangeably. Also, we are always considering minimization problems, if not mentioned otherwise. Recall that a maximization problem can easily be transformed into a minimization problem.

1.4 An introduction to MINLP solution approaches

The optimization problems considered directly or indirectly in this thesis can be classified and denoted as follows:

LP Linear Programming.

MILP Mixed Integer Linear Programming.

NLP Non-Linear Programming.

MINLP Mixed Integer Non-Linear Programming.

RMILP Relaxed Mixed Integer Linear Programming.

RMINLP Relaxed Mixed Integer Non-Linear Programming.

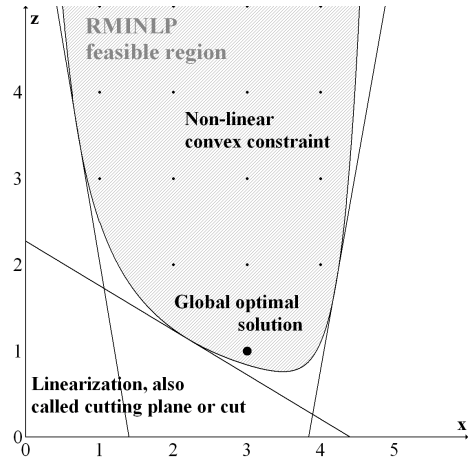

Figure 1.1: An Outer-Approximation with linearizations.

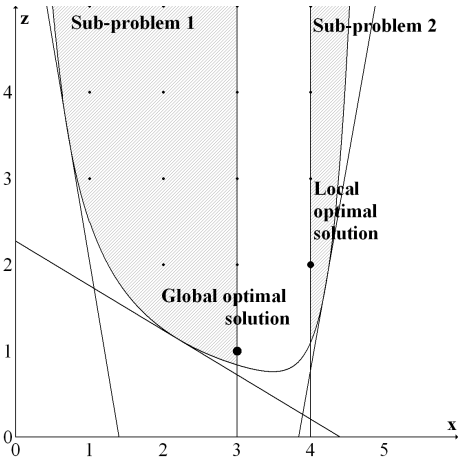

Figure 1.2: A possible division of the problem in two subproblems.

Let us consider the following MINLP problem:

$$
\begin{aligned}
&\min\ f(x,y) && \text{(objective function)}\\
&\text{subject to}\\
&\quad h_i(x,y) = 0 && i \in I \text{ (equality constraint)}\\
&\quad g_j(x,y) \leq 0 && j \in J \text{ (inequality constraint)}\\
&\quad x \in \mathbb{R}^n\\
&\quad y \in \mathbb{Z}^m
\end{aligned}
\tag{1.4.1}
$$

where I and J are sets of equality and inequality constraints and x and y are continuous and discrete vectors, respectively. A function, $f(x,y)$, $h(x,y)$, or $g(x,y)$, can be either linear or non-linear, however, in an MINLP problem at least one function is non-linear and the model contains at least one integer variable. The objective function can be made linear by moving the non-linearity into a constraint, for example, by stating min z subject to $z = f(x,y)$.... For demonstration purposes we consider in this section that we minimize z subject to some constraints. Note also that an integer variable can be expressed as a combination of binary variables.

An LP problem is a convex problem and can be solved in an efficient manner, for example, with the simplex method, see Vanderbei [2001]. An MILP problem is non-convex, but if the discontinuity, i.e. the integer requirement, is relaxed, the resulting RMILP=LP problem is convex. Loosely speaking, relaxing means that some restriction(s) are disregarded, for example, when the integer requirement is relaxed then the integer variables may take continuous values at a solution. An MILP problem can be solved with the Branch-and-Bound (BB) method by solving a sequence of convex LP problems, see Vanderbei [2001]. Note that the BB method, in a worst case scenario, needs to divide the problem into twice as many subproblems as there are combinations of the values of the integer variables. Consider figure 1.1, when z is minimized and if we require z and x to take integer values, then the feasible solutions are represented by the black dots within the feasible region of the integer relaxed MINLP problem. In figure 1.2 we can see a possible division of the problem in figure 1.1 into two subproblems which can be solved separately.

A non-convex MINLP problem can be solved by solving a sequence of convex MINLP problems. For an MINLP model the non-convexity may be in the functions and therefore, the non-convexity of the integer requirement for MINLP models is generally disregarded in the discussion and, likewise, in this text. Hence, a non-linear function can be either convex or non-convex and the problem in figure 1.1 is a convex MINLP problem. Furthermore, if all functions are convex, the model is convex, otherwise non-convex. To informally describe convexity we can consider the problem of finding the lowest point (minimum). The lowest point of a bowl (convex) can be found by following a drop of oil poured into the bowl. Let us now consider an NLP model, for example, an integer relaxed MINLP model. The convex NLP solvers are based on the fact that we can always search in the direction which the oil flows and stop when the oil stops, i.e. local optimization. It is worth pointing out that two local solvers may find different solutions when started

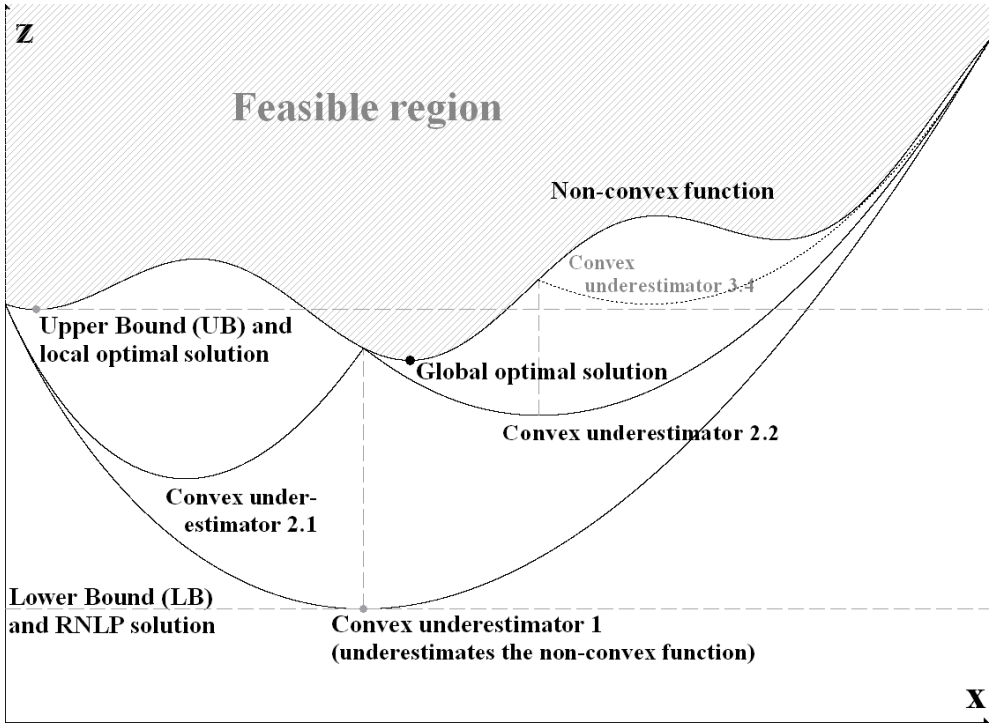

Figure 1.3: Global optimization BB illustration.

from the same starting point because the path to a local optimum is different. However, if we attempt to find the lowest valley between hills (non-convex) the approach does not guarantee that we will find the lowest valley, see figure 1.3. One approach to overcome this is to solve a non-convex NLP problem by using convex underestimators and concave overestimators that overestimate the feasible region, i.e. consider a larger feasible region. To describe a non-convex function sufficiently well, the convex approximation can be enhanced at a point by dividing the problem into two subproblems and forming more exact underestimators for each problem, i.e. a BB method, see figure 1.3. The vertical lines show how the problem is divided or branched into convex subproblems or nodes and each node solved separately. Notice also that if the proposed Upper Bound (UB) in figure 1.3 is found, the region approximated by the convex underestimator 3.4 can be disregarded or fathomed, because we already know a better solution, i.e. the UB. Furthermore, the UB enables us to sometimes reduce the variable bounds, because we are not interested in the situation $z > UB$, i.e. after solving a relaxed problem the Branch-and-Reduce (BR) can be applied. For a description of the resulting solution procedure see [Soland and Falk, 1969], for an illustrative non-general description, and McCormick [1976], for a more general formulation. Note that the global optimization BB method is, in general computationally costly because several convex subproblems need to be solved. A general description of global optimization techniques can be found in Floudas [2000].

The local MINLP solver uses different methods to converge to a solution, but mainly solving subproblems of type LP, MILP and/or NLP. Many of the solvers use a local NLP sub-solver to solve subproblems with fixed or relaxed integer variables. The local NLP solvers usually require the Hessian matrix (the second order derivatives of all constraints) or an approximation of it. This means, depending on the cost of this computation and how frequently the NLP solver is called, a good choice of MINLP solver can be made. Recall also, that any local NLP solver may fail to find a solution, due to the chosen starting point or non-smooth functions. An important non-convexity, that can make a crucial difference in solving a problem, is the treatment of non-linear equality constraints. A non-linear equality constraint is always non-convex, but if the functions are convex [3] , many NLP solvers find good solutions. If, however, linearizations are introduced, they are likely to result in no solution being found. The reason is that two linearization for a constraint, made at two points close to each other, where the constraint values are positive and negative respectively, typically reduce the feasible region exaggeratedly. On the other hand, linearizations are computationally very inexpensive to calculate and usually reduce the relaxed solution space efficiently. An overview of MINLP solving techniques can be found in Adjiman et al. [1998] and Grossmann and Kravanja [1995] and include Branch-and-Bound (BB), Outer Approximation (OA), Generalized Benders Decomposition (GBD) and Extended Cutting Plane (ECP). A more thorough explanation of GBD and OA is in Floudas [1995] and of ECP in Westerlund et al. [1998].

1.5 A short description of some GAMS MINLP solvers

The GAMS optimization platform was chosen due to its wide range of excellent solvers and good availability of test models. The GAMS system became commercial in 1987 and, currently (2011), includes 9 MINLP solvers: 8 deterministic and 1 heuristic. In this section a rough overview of 7 GAMS solvers is given. The heuristic solver OQNLP will not be described and neither is KNITRO, because the MINLP solving capability is designed for convex MINLP models of moderate size [GAMS, 2011]. The descriptions aim to give an understanding of the main differences between the MINLP algorithms and provide sufficient background to support the discussion in section 2.4. It is not meant to be a complete description of the algorithms and therefore I want to apologize to the method developers for any unmentioned important details. As a first source, for more information about the solvers, the solver manuals GAMS, 2011, as well as, Bussieck and Vigerske [2011] can be consulted.

1.5.1 SBB

SBB (Simple Branch-and-Bound) implements a Branch and Bound algorithm where NLP problems are solved in every node. The algorithm starts by relaxing the integer variables and solving an NLP problem. If the solution satisfies the integer requirement, the algorithm terminates, otherwise the branching procedure is applied. After each NLP solution

[3] If the constraint $h(x, y) \leq 0$ is non-linear and convex, then the constraint $h(x, y) = 0$ is non-convex.

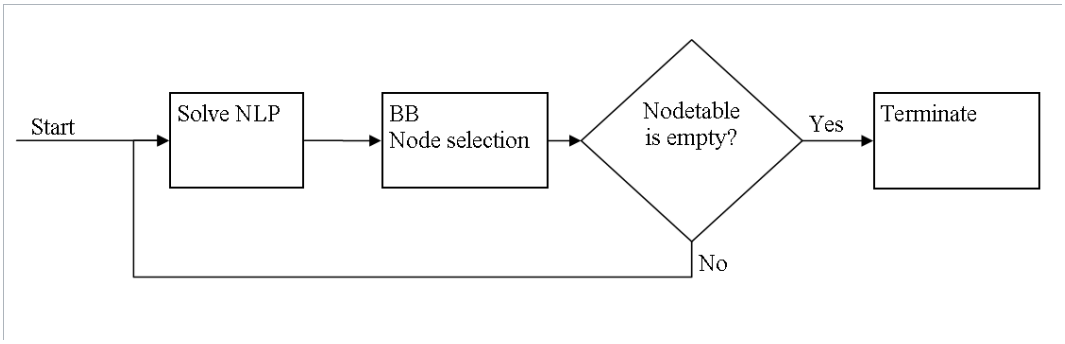

Figure 1.4: Simplified SBB algorithm.

two nodes are added to the BB tree by tightening the bound for some variable(s), for example, if $y = 0.3$ then require in the first node that $y \geq 1$ and in the second node that $y \leq 0$. In the next step a node from the node table is selected and an NLP problem is solved, starting from the previous NLP solution. If an MINLP solution is found, then some nodes from the BB tree might be removed[4], i.e. if the objective function value for the node is worse than the MINLP solution. The algorithm terminates when the node table is empty. More information regarding the solution procedure can be found, for example, in Leyffer [2001].

SBB solves mainly NLP problems and is therefore well suited for models with fewer discrete variables but more difficult nonlinearities. SBB has a fair chance of finding solutions for non-convex models, but global optimality can only be guaranteed for convex problems.

Solver Type: Convex MINLP solver.
Developer: ARKI Consulting and Development A/S.
Introduced in GAMS: 2001.

1.5.2 DICOPT

The algorithm in DICOPT is based on three key ideas: Outer Approximation (OA), Equality Relaxation (ER) and Augmented Penalty (AP). The OA algorithm decomposes the problem into a discrete linear problem (MILP) and a continuous non-linear problem (NLP), see figure 1.5. The master MILP problem is a linear approximation of the MINLP problem, where the MINLP approximation is improved during the iteration procedure. First the NLP problem is solved by fixing the integer variables and a UB can be obtained. Then, linear approximation(s) of non-linear functions are added to the MILP problem and by solving the resulting problem an LB is obtained. If the LB>UB then the integer variables are fixed and an NLP problem is solved. The procedure is repeated until the LB≤UB, when the solution, i.e. UB, can be returned.

[4]The term fathomed can also be used.

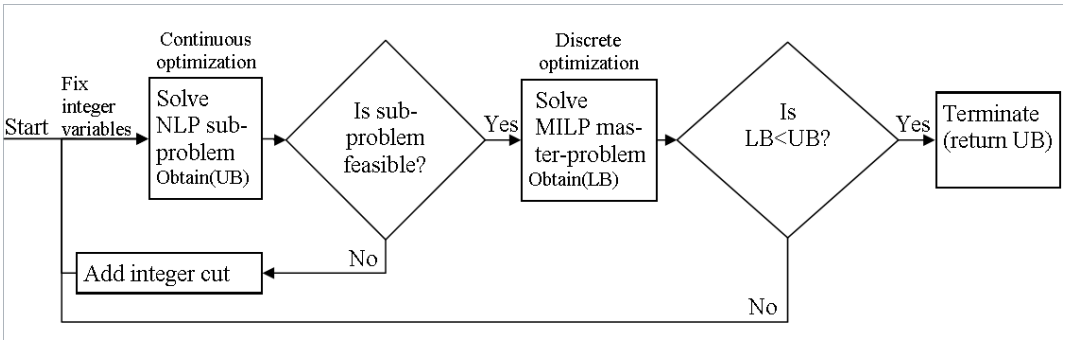

Figure 1.5: Simplified DICOPT algorithm by Kocis and Grossmann [1989].

Additionally, the following can be noted. A continuous NLP problem can be obtained by simply relaxing the integer requirement. The equality constraints can be relaxed after each NLP call into $\leq$, if the associate Lagrange multiplier is positive or else to a $\geq$ constraint. Furthermore, for some fixed y the NLP might be infeasible and in order to avoid this, the non-linear constraints are allowed to violate by adding penalty variables for the non-linear constraints. The violation(s) is/are penalized in the objective function and if a feasible solution exists, the penalty variables will be driven to zero. If a feasible solution does not exist then the continuous variable levels that minimize the summed violation of the inequality constraints will be found. However, in this case an integer cut, instead of linearizations, is added to the MILP problem that prevents finding the same integer combination again. A more detailed description of the algorithm can be found in Floudas [1995] and Kocis and Grossmann [1989].

DICOPT was based on the assumption that MILP models can be solved efficiently and designed so that it minimizes the number of NLP subproblems solved, compared to, for example, SBB.

Solver Type: Convex MINLP solver.
Developers: J. Viswanathan and Ignacio E. Grossmann at Carnegie Mellon University.
Introduced in GAMS: 1993.

1.5.3 AlphaECP

AlphaECP or αECP is based on the Extended Cutting Plane (ECP) method [Westerlund et al., 1998] and a simplified view of the algorithm can be seen in figure 1.6. First the linear part of the problem is solved by excluding the non-linear constraints from the problem. After the resulting MILP model is solved, the NLP solver can be called with fixed integer levels from the MILP solution in the hope of finding an MINLP solution. If the point satisfies all the constraints an MINLP solution is found. However, if some of the constraints are not satisfied, at least one unsatisfied nonlinear constraint is linearized and the resulting MILP model is solved. A linearization is a valid underestimator of a

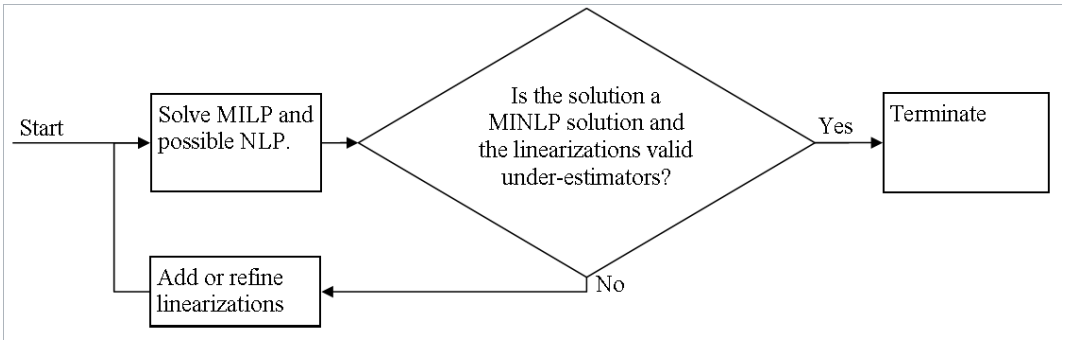

Figure 1.6: Simplified AlphaECP algorithm.

convex constraint, see figure 1.1. However, it is not a valid underestimator of a pseudoconvex constraint[5], but can be refined into such. The refinement procedure speeds up the convergence compared to if a valid underestimator of a pseudoconvex constraint would be immediately formed. The algorithm terminates when the MILP solution is also an MINLP solution and all linearizations are valid under-estimators of pseudoconvex constraints. For an illustrative presentation of the ECP method, see Lastusilta [2007].

Solver Type: Pseudoconvex MINLP solver
Developer: Tapio Westerlund at Åbo Akademi University.
Introduced in GAMS: 2007.

1.5.4 BARON

> The Branch And Reduce Optimization Navigator derives its name from its combining interval analysis and duality in its "reduce" arsenal with enhanced "Branch and Bound" concepts as it winds its way through the hills and valleys of complex optimization problems in search of global solutions. [Sahinidis, 2000]

The BARON (Branch And Reduce Optimization Navigator) algorithm is based on the Branch and Bound (BB) concept to solve non-convex MINLP problems to global optimality, see figure 1.8 and the more accurate figure 1.7. Let (P) be the MINLP problem and (R) an Outer-Approximation (OA) of (P), see figure 1.1 OA example. Solving (R) provides a valid Lower Bound (LB), while (P) provides a local Upper Bound (UB). If UB-LB is sufficiently small the algorithm terminates. Otherwise the BB subdivides the feasible region into more accurate approximations, for which a new LB closer to UB is obtained and possibly a lower UB (figure 1.3 represents the main idea). The procedure is repeated until UB and LB become sufficiently close to each other. In order

[5]Note that a pseudoconvex constraint is a slightly more general constraint class, hence, a convex constraint is also pseudoconvex.

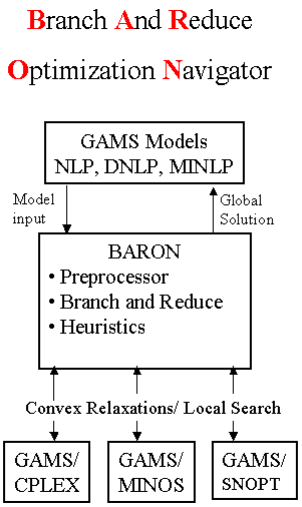

Figure 1.7: Schematic GAMS/BARON connection diagram [Website: BARON].

to make the algorithm efficient, several techniques are used, of which the most important are mentioned here. BARON has a set of different function relaxations that create outer-approximations[6], i.e. convex/concave envelopes which can be linearized. Note that nonlinear programs are harder to solve and are associated with more numerical issues than linear programs [Tawarmalani and Sahinidis, 2004]. Range reduction is incorporated to enhance the convergence speed, i.e. to find tighter variable bounds by, for example, using the convexity and optimality of (R). A detailed description of the algorithm can be found in Tawarmalani and Sahinidis [2002] and Sahinidis [2000].

BARON can guarantee global optimality under fairly general assumptions. These include finite lower and upper bound on variables and their expressions. Note that if a variable is bounded it does not necessary result in a bounded expression, for example, $1/x, x \in [0, 100]$ results in an infinite expression value if $x = 0$. BARON supports the following nonlinear functions: multiplication, division, e^x, $\ln(x)$, x^α for real α, β^x for real β, x^y, and $|x|$.

Solver Type: Global MINLP solver (for a limited set of nonlinear functions).
Developer: Nick Sahinidis at Carnegie Mellon University (partly developed at the University of Illinois at Urbana-Champaign).
Introduced in GAMS: 2001.

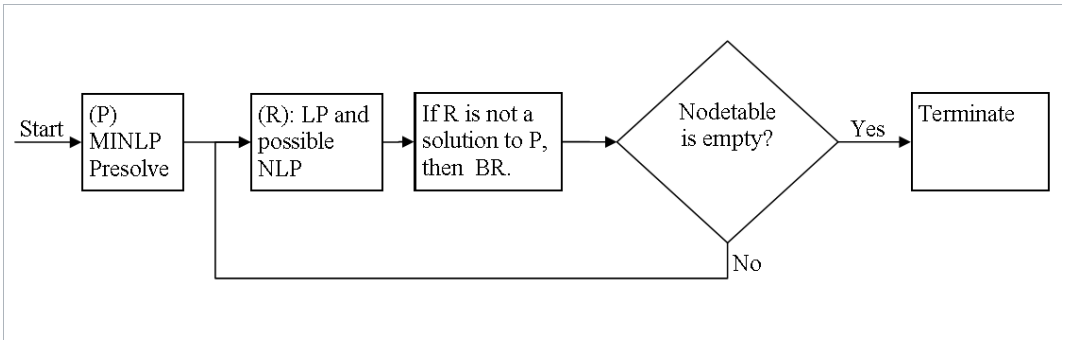

Figure 1.8: Simplified LindoGlobal and BARON algorithm.

1.5.5 LindoGlobal

LindoGlobal is based on the following ideas:

> a) converting the original nonlinear/nonconvex into several linear/convex subproblems, b) using a Convex, Interval, and Algebraic(CIA) analysis, and c) a Branch-and-Bound technique to exhaustively search over these subproblem for the global solution. [Schrage and Gau, 2003]

LindoGlobal, as well as BARON, uses the Branch-and-Bound approach to decompose the non-convex model into several convex sub-models. The algorithm starts with a presolving step where problem (P), see figure 1.8, is solved with local search procedures by starting the search from different starting points. A solution can obviously be used in the BR step, as well as, be returned as a solution. In the first step the problem (R) is solved, which is a relaxation of (P). This means a non-convex function $f(x)$ is relaxed by a convex underestimation $g_1(x) \leq f(x)$ and a concave overestimation $g_2(x) \geq f(x)$, which can, in turn, be approximated by linearizations. If the solution to (R) is also a solution to (P) the solver can terminate and return the global optimal solution, otherwise the BR is applied. BR subdivides the feasible region into more accurate approximations and places the new nodes in a node table. When all nodes have been solved or fathomed (inferior to already obtained MINLP solutions) the algorithm terminates. More information about the solving procedure can be found in Schrage and Gau [2003] and Schrage and Lin [2009].

LindoGlobal and BARON are presented with the same simplistic algorithm diagram in figure 1.8, however, despite the basic idea is the same the algorithms differ significantly. LindoGlobal supports a large set of nonlinear functions and furthermore, relaxations, node selection, range reduction and node selection might be significantly different.

Solver Type: Global MINLP solver (currently limits the model size to 3000 variables and 2000 constraints)

[6] An outer-approximation refers to how the MINLP model is approximated, while the outer approximation algorithm refers to a solution approach.

Developer: Lindo Systems, Inc.
Introduced in GAMS: 2007.

1.5.6 COIN-OR solvers Bonmin and Couenne

COIN-OR (COmputational INfrastructure for Operations Research) is open-source software for the operations research community [Lougee-Heimer, 2003]. One part of the project is to host the GAMS interface and in this section only the MINLP solvers of the COIN-OR infrastructure are very briefly described: Bonmin (Basic Open-source Nonlinear Mixed Integer programming) and Couenne (**C**onvex **O**ver and **U**nder **En**velopes for **N**onlinear **E**stimation). The reader is referred to the COIN-OR manual [GAMS, 2011] and Website: COIN-OR for further information.

Bonmin has six different algorithms, however, the default algorithm, B-BB, is similar to SBB. The other algorithms are: B-OA, B-QG, B-Hyb, B-ECP and B-iFP. B-OA is similar to DICOPT, while B-QG is an outer-approximation based branch-and-cut algorithm based on solving a continuous linear model at each node of the search tree, improving the linear model by outer approximation, and branching on integer variables. B-Hyb is a hybrid of B-BB and B-QG. B-ECP uses Kelley's outer-approximation [Kelley, 1960] based branch-and-cut algorithm, while B-iFP is an iterated feasibility pump algorithm.

Couenne is a global optimization solver for MINLP problems.

> Couenne solves convex and nonconvex MINLPs by an LP based spatial branch-and-bound algorithm that is similar to the algorithm used in BARON. The implementation extends Bonmin by routines to compute valid linear outer approximations for nonconvex problems and methods for bound tightening and branching on nonlinear variables. [GAMS, 2011].

Solver Type: Convex MINLP solver (Bonmin)
Developers: Collaboration between Carnegie Mellon University and IBM Research.
Introduced in GAMS: 2007.

Solver Type: Global MINLP solver (Couenne)
Developers: Collaboration between Carnegie Mellon University and IBM Research, and later also Lehigh University.
Introduced in GAMS: 2009.

CHAPTER 2

The performance of some MINLP solvers

When solving an MINLP problem, the type of non-linearity and the number of integer variables can significantly impact upon the solving capability of a solver. Paying attention to the model characteristics and observing the solving capability of a solver can give insight into a suitable solver choice for a particular problem. An easily observed characteristic is the application area of the problem, which might provide indicative information for a good solver choice. When difficulties arise in solving a model it is sometimes worthwhile to analyze the non-linearity in detail in order to improve the model. However, in other cases it might be easier to simply try another solver.

2.1 Problem libraries

Problem libraries can be used to measure the performance of solvers.[1] In this chapter two model libraries, MinlpLib and Convex MinlpLib, are used to measure the performance of MINLP solvers. Additionally, GlobalLib is used in paper II and 50 problems from QapLib in paper IV. In total 875 models has been used in the comparisons of this thesis: MinlpLib(270), Convex MinlpLib(140), GlobalLib(415) and QapLib (50).

1. MinlpLib, with 270 models, both convex and non-convex [Website: MINLP Library].
2. Convex MinlpLib with 140 convex models [2][Website: Convex MINLP Test Problems]. Note that the values *best solution known* was obtained by collecting them from this comparison.

[1] For a collection of test problem libraries see Website: Global Optimization Test Libraries.

[2] Contains at the time of writing 142 models, but problems Syn10M.gms and Syn30M03M.gms had a faulty solve statement, which was discovered after the test run and is not included in the comparison.

3. GlobalLib, with 415 NLP models [3] , both convex and non-convex [Website: GLOBAL Library].

4. QapLib, with 136 Quadratic Assignment problems, however, only 50 of them were used [Website: QAP Library][4].

Some characteristics of MinlpLib, which was introduced in Bussieck et al. [2003], can be seen in table 2.1. Note that model meanvarxsc.gms contained semi-continuous

Table 2.1: MinlpLib statistics.

	The number of problems with following characteristics:				
Interval	constraints	non-linear constraints	non-linear equality constraints	variables	integer variables
0	0	0	160	0	0
1-49	111	218	61	107	179
50-99	33	4	5	21	30
100-499	83	27	32	89	33
500-999	14	11	5	23	21
1000-4999	22	9	6	23	4
5000-9999	5	1	1	4	0
10000-99999	2	0	0	3	3
Sum	270	270	270	270	270

variables which could not be handled by all solvers. The Convex MinlpLib refers to 140 convex MINLP problems collected from Website: Convex MINLP Test Problems for which some characteristics can be seen in table 2.2. In table 2.3 and in table 2.4 some characteristics of GlobalLib and QapLib, respectively, can be found. Note that in table 2.4 all problems include only one non-linear constraint and the related columns are therefore left out.

2.2 Comparison setup

The test computer is an Intel core i7 with 4 cores of 2,8 GHz and 6 GB of memory. A time limit per model is set for the solvers to perform the optimization. In the comparison a time limit of 1000 seconds per problem is used, except for the performance profiles for AlphaECP in figures 2.1, 2.2 and 2.3, when a time limit of 6 hours per problem is used. If the solver has not stopped within 10 minutes after the time limit has been exceeded, the solution process is terminated by force. In this case the solver will not produce a solution file. The GAMS solvers were used with default settings, since it is assumed

[3] Only 403 models can be solved by AlphaECP. 4 models contained equilibrium constraints and are excluded from the comparison. Additionally, 8 problems contained equation or variable types that AlphaECP can not handle and those problems are excluded from table 2.3.

[4] Additionally, two problems were excluded from the comparison because of a system or memory limitation and are therefore, not included in table 2.4.

Table 2.2: Convex MinlpLib statistics.

	The number of problems with following characteristics:				
Interval	constraints	non-linear constraints	non-linear equality constraints	variables	integer variables
0	0	0	140	0	0
1-49	4	115	0	9	46
50-99	10	21	0	13	30
100-499	55	4	0	67	62
500-999	20	0	0	29	2
1000-4999	51	0	0	22	0
5000-9999	0	0	0	0	0
10000-99999	0	0	0	0	0
Sum	140	140	140	140	140

Table 2.3: GlobalLib statistics.

	The number of problems with following characteristics:				
Interval	constraints	non-linear constraints	non-linear equality constraints	variables	integer variables
0	0	1	210	0	403
1-49	289	308	120	255	0
50-99	31	19	15	29	0
100-499	20	22	19	36	0
500-999	7	13	12	10	0
1000-4999	41	39	26	48	0
5000-9999	11	1	1	18	0
10000-99999	4	0	0	7	0
Sum	403	403	403	403	403

that the solver developer has chosen, as default, the best settings for an arbitrary model. If customized settings were used, it would be very time consuming when solving large problem and solver sets. Furthermore, we would need to have a deep understanding of the underlying algorithm for all solvers so the best settings for each solver could be chosen for the particular test batch. Otherwise the comparison might be disadvantageous for some solvers because of a lack of knowledge of the best settings for the considered test batch.

2.3 Performance indicators

To measure the solver performance for a batch of problems the following indicators can be used: number of found solutions, solution quality and solution time. An optimization model may or may not have a solution and if a feasible (possible) solution is found, then it

Table 2.4: QapLib statistics.

Interval	**The number of problems with following characteristics:** constraints	bilinear terms	variables	integer variables
0	0	0	0	0
1-49	0	0	0	0
50-99	28	0	0	0
100-999	20	0	16	16
1000-9999	0	0	32	32
10000-99999	0	0	0	0
100000-999999	0	16	0	0
1000000-9999999	0	18	0	0
10000000-99999999	0	14	0	0
Sum	48	48	48	48

is not necessarily a local or global optimal solution. The notation "best solution known" is self explanatory, but note that it is not necessarily found by any solver during the test run. Solution quality can be defined as the difference between a found solution and the best solution known. A tolerance that determines if we have found the best solution is used. If the absolute solution value of the best solution known is ≤ 1 an absolute tolerance 10^{-6} is used, else a relative tolerance. Solver performance can be measured as the number of found solutions over time, furthermore, the solution quality can also be observed. Presenting the performance of a solver in a performance profile, see figure 2.1, can however, result in a distorted view if all the desired information is not available.

There are mainly two issues when the GAMS platform is used for a solver comparison. Firstly, only one solution and the respective solution time is easily retrieved from the system and, secondly, not all MINLP solvers always respect a given time limit sufficiently well. It would be desirable to have information about all intermediate solutions and their solution times. When only one solution for a model is available it is very likely to cause a distorted performance profile, for example, when a solver finds the global optimal solution within seconds, but takes a very long time to verify the solution. In this case the solver shows a good performance with a short time limit, but a less good performance with a long time limit. Currently, the GAMS solution report is produced at the final step in the optimization and contains only one solution and its respective solution time. To overcome this issue, continuous performance profiles are not presented for the MINLP solver comparison, but rather the solver performance at different time points, where the solver has been given the time point specific time limit. Obviously, this is computationally much more costly since the same models have to be repeatedly solved with different time limits. However, currently no better way is known by the author to overcome the distortion issue. To illustrate the issue AlphaECP is used, since it provides the intermediate solution values and their solution time in a log file. Figures 2.1-2.3 presents the performance of AlphaECP when the problems in the MinlpLib are solved with a 6 hour time limit per problem. In figure 2.1 we can see the performance

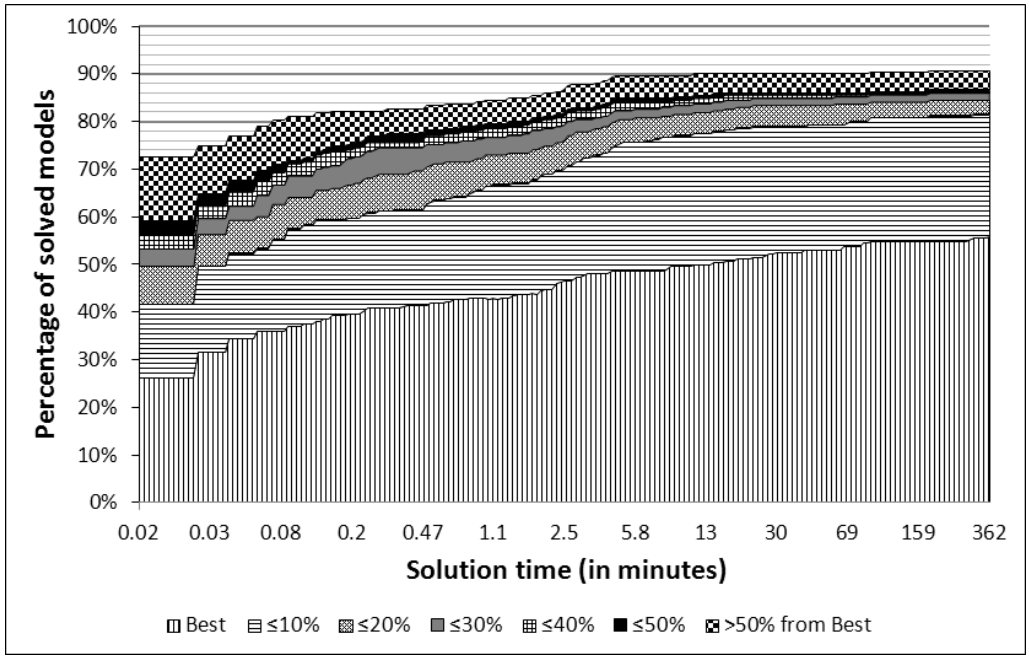

Figure 2.1: Performance profile for AlphaECP when intermediate solutions are taken into consideration.

profile of AlphaECP when information regarding intermediate solutions is available and in figure 2.2 when only the final solution is available. In figure 2.3 the measurement error between the two profiles can be seen. The global MINLP solvers, but possibly also other local MINLP solvers, suffer from a distorted performance profile if only the final solution is available.

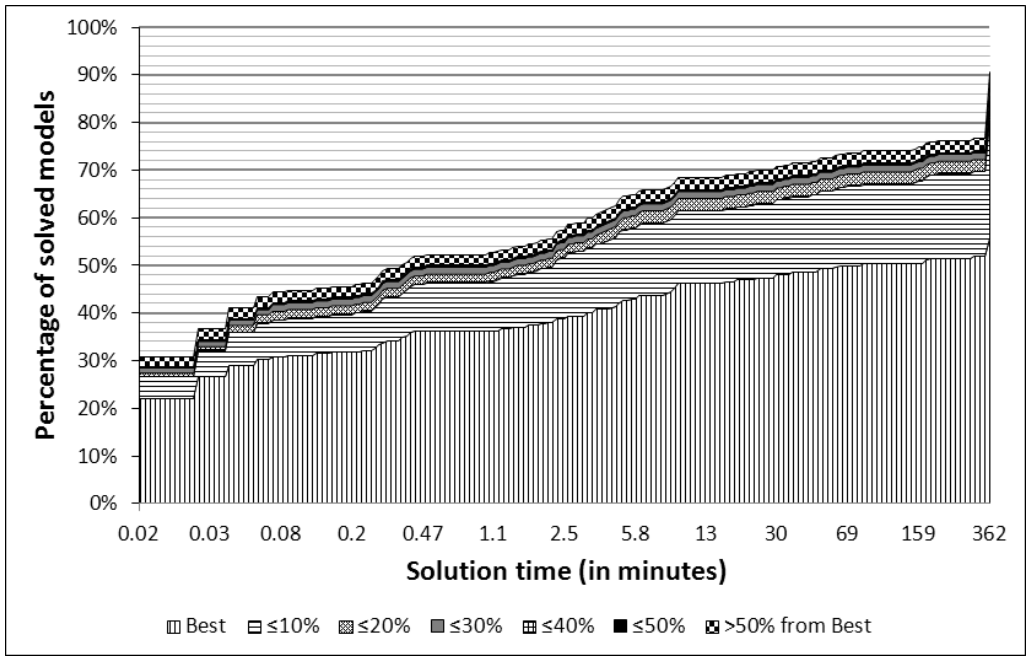

Figure 2.2: Performance profile for AlphaECP when intermediate solutions are **not** taken into consideration.

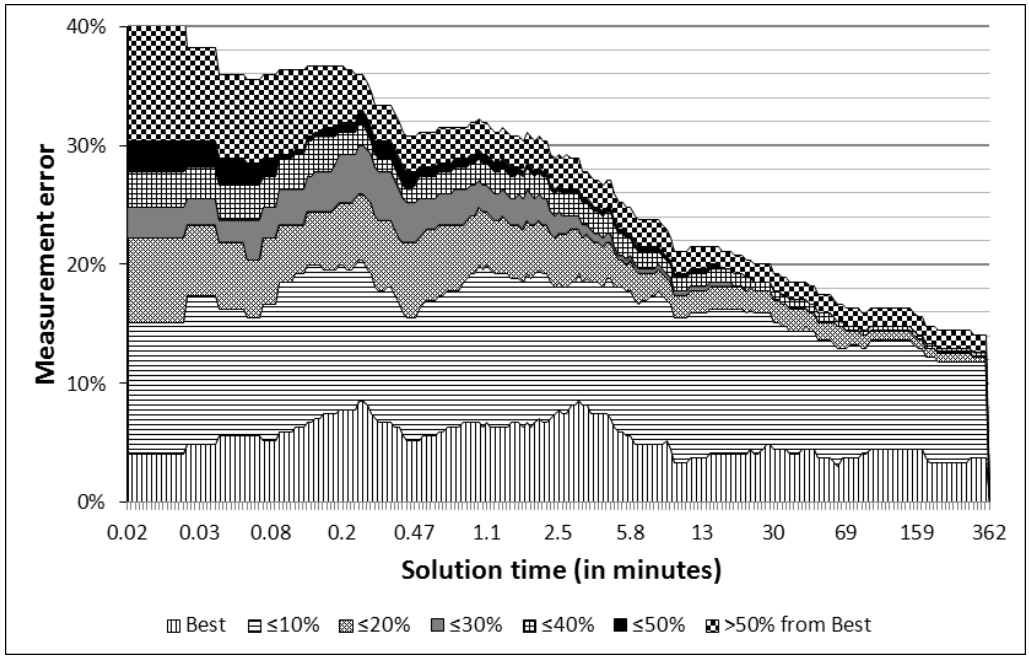

Figure 2.3: The measurement error caused by **not** taking the intermediate solutions into consideration.

Another issue that occurs when undertaking a solver comparison on the GAMS platform is that some solvers do not respect the time limit sufficiently well. For example, for SBB this issue occurs because the reading/writing of files in SBB is not included in the time consumption measurement [Bonami et al., 2008]. If a time limit is given, but the solver does not stop at the limit, it is possible to stop the optimization by force or complete the optimization. For comparison purposes both options are unsatisfactory. If we stop the solver by force, an earlier found solution is not reported, but on the other hand a solution found after the time limit is exceeded, does not result in a fair comparison. Hence, a compromise must be made and, here, we chose to let the solver exceed the time limit with 10 minutes before terminating the solver by force.

2.4 Solver comparisons

Many of the GAMS/MINLP solvers are currently convex MINLP solvers. A local MINLP solver can, in some sense, be seen as a heuristic solver when a non-convex MINLP model is solved. It can converge to a different local optimal solution depending on the choice of sub-solver, start point or solver settings. The models in the convex MinlpLib and MinlpLib are solved with a 1000 seconds ($\sim$ 17 minutes) time limit per problem.

In figures 2.4 and 2.5 we can observe the ability of the compared solvers to find solutions when different time limits are given to the solvers. In figure 2.4, which consisted of 140 convex models, we can see that some local solvers outperformed the global solvers BARON, LindoGlobal and Couenne. Observe that AlphaECP can, already, find a solution to all problems within 10 seconds and, furthermore, that over 80% of the problems are solved to the best known value when the $\sim$ 17 minutes time limit per problem is reached [5]. This implies that a global solver for non-convex models does not need to be the best choice to solve a convex MINLP problem!

In figures 2.6 and 2.7 we can see the time consumption of the solvers. The notation *GAMS time* refers to the time consumption reported by GAMS. The notation *Additional* refers to the additional actual time it took to complete the test run. Hence, the total time to complete the test run is the sum of *GAMS time* and *Additional*. **Note that the external time check used an interval of 30 seconds and, therefore, the total error in the "Additional" measurement can be up to 70 minutes in figure 2.6 and 135 minutes in figure 2.7, which is about the lowest total time used at time point 10 seconds in both figures.**

[5] AlphaECP did not find for all models a higher quality solution than some other solver (not shown in figure 2.4).

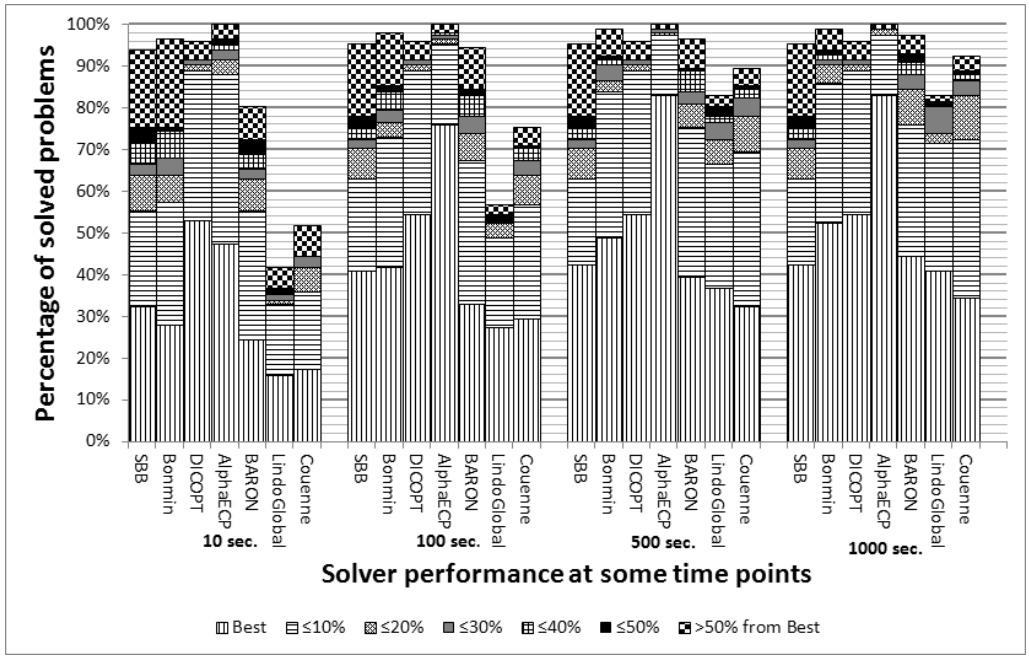

Figure 2.4: Convex MinlpLib: a solution quality comparison.

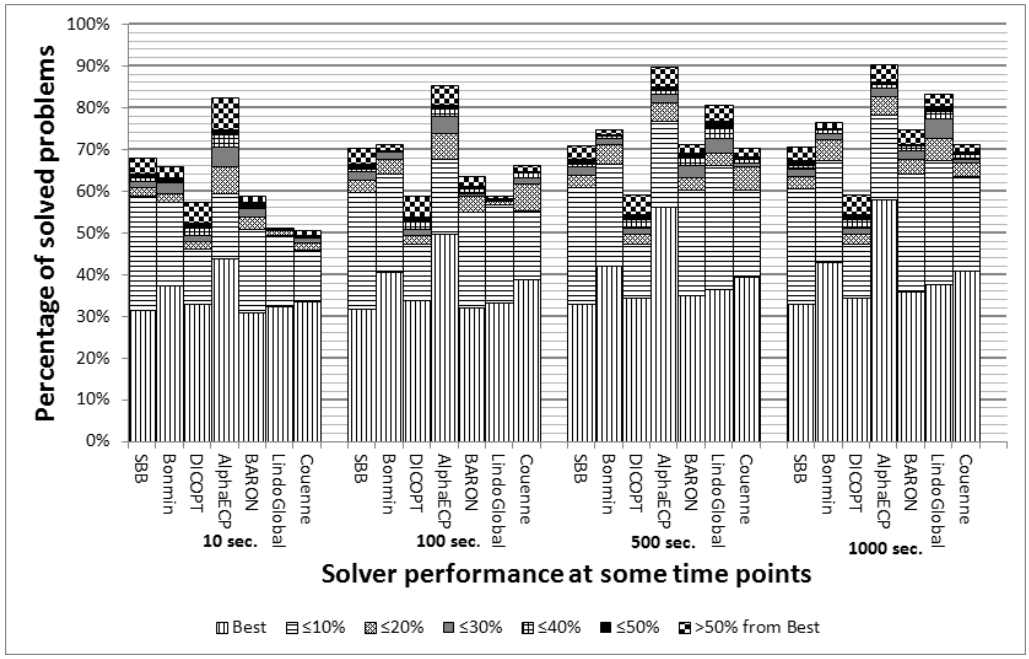

Figure 2.5: MinlpLib: a solution quality comparison.

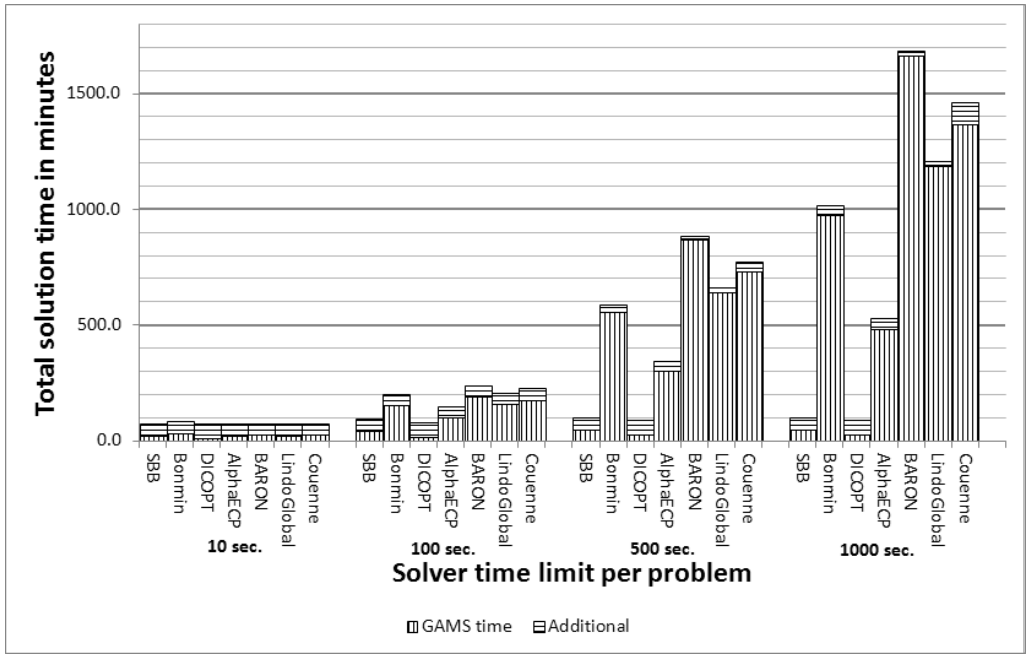

Figure 2.6: Convex MinlpLib: a solver solution time comparison.

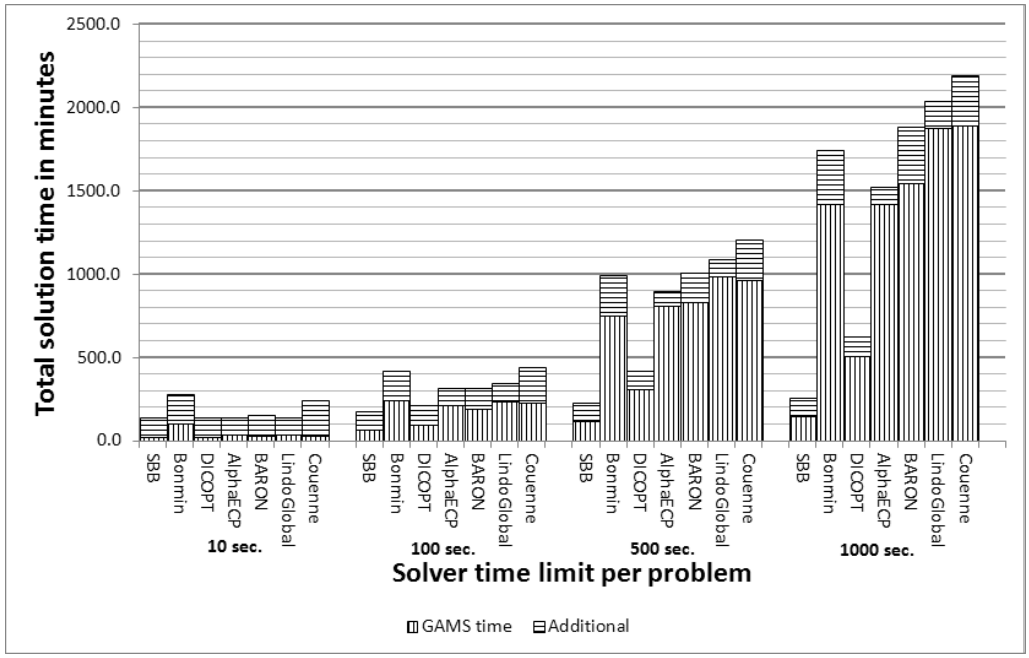

Figure 2.7: MinlpLib: a solver solution time comparison.

CHAPTER 3

GAMS/AlphaECP algorithm development

The GAMS/AlphaECP algorithm is a local MINLP solver implemented in C language and available in the General Algebraic Modeling System. An early release and illustration of the main ideas can be found in Lastusilta [2007].

3.1 An introduction to AlphaECP

The ECP method is an extension of Kelley's cutting plane method for convex NLP problems, see Kelley [1960]. The ECP method was introduced by Westerlund and Pettersson [1995] and finalized by Westerlund and Pörn [2002]. In the first phase it was developed for solving convex MINLP problems and in the second phase to solve MINLP problems with pseudoconvex constraint Westerlund et al. [1998]. In the final phase it was developed to solve pseudoconvex MINLP problems, i.e. problems with a pseudoconvex objective function. Note that the ECP method requires only the solution of an MILP subproblem, the evaluation of the non-linear constraint and one gradient evaluation in each iteration.

In addition to the described ECP method the following functionality has been added to AlphaECP in order to improve the performance: an NLP solver can be called; a cut reselecting heuristic can be used and the solver is able to take advantage of multiple MILP solutions. The NLP solver is typically able to find an accurate solution in a short time and therefore, improves the performance especially for problems containing mainly continuous variables in the non-linear constraints. It also improves the probability of finding a solution for problems containing non-linear equality constraints. The cutting plane reselecting heuristic is not required for pseudoconvex MINLP problems, but improves the probability of solving non-convex MINLP problems. The capability of selecting an MILP solution, after an MILP solver call, improves the quality of cut-

ting planes and, furthermore, slightly increases the convergence speed at the end of the iteration procedure.

3.2 The enhancements to AlphaECP

In this section the algorithm changes to the ECP method described in Westerlund and Pörn [2002] is discussed. The focus was set on improving the solution capability for non-convex MINLP even though nothing more than feasibility can be said about the solution quality, if a solution is found. The main changes discussed are the following:

1. The use of an NLP sub-solver.
2. A cutting plane reselecting heuristic.
3. The use of multiple MILP solutions.

The essentials parts of the algorithm will now be described. The GAMS/AlphaECP version 2.04.01 algorithm, which was released in GAMS version 23.7.1, can be seen in flowchart 3.1. The general idea of the algorithm is to solve an MINLP problem by solving a sequence of MILP problems. An iteration is defined by the loop seen in the flowchart 3.1, i.e. typically solving an MILP problem and adding cuts. An MILP problem in AlphaECP typically consists of the linear part of the MINLP problem and the generated cutting planes. When AlphaECP calls an NLP sub-solver, the NLP problem consists of the original problem where the integer variables have been fixed to variable levels, obtained from the MILP solver solution. To verify optimality, which guarantees optimality for problems with pseudoconvex constraints, *alpha* values are increased to their upper limits, *alphamax*. When the *alpha* values are increased, the approximated feasible region of the MINLP problem is likely to grow and better solutions might be found. The default strategy guaranteeing optimality for a problem with pseudoconvex constraints and a convex objective function will be given [1] . In figure 3.1, the rectangular holders denote processes and the diamond-shaped holders decisions. (*) denotes that the decision Yes is taken only after the question is true for a specified number of consecutive iterations. The reasons why we are unable to add cuts can be: all gradient values are below the accepted tolerance for violating constraints or the function can not be evaluated, for example, $\sqrt{-1}$. The MILP solver parameter MIPoptcr denotes the accepted relative gap between the MILP solution and its integer relaxed solution.

The use of an NLP sub-solver enables AlphaECP to frequently find an MINLP solution early in the solution process. AlphaECP calls an NLP solver with fixed integer variables levels obtained from the MILP solution. Typically, the solution found by the NLP solver is accurate and the solution time short with respect to the MILP solution time. Therefore, it is usually beneficial to call the NLP sub-solver several times during the AlphaECP solution procedure. Cutting planes generated early in the solution procedure

[1] The changes required to solve problems with a pseudoconvex objective function can be found in Westerlund and Pörn [2002]. Here, they are not taken into consideration because they are not essential for discussing the enhancements to the algorithm.

typically reduces the MILP feasible region of MILP subproblems efficiently. However, when closing in on a solution a cut of a non-convex constraint, especially of an equality constraint, might result in an infeasible subproblem. Hence, the pure ECP method is not well suited for solving problems with many non-linear equality constraints. Since MINLP problems frequently include non-linear equality constraints it is understandable that an NLP sub-solver considerable improves the performance of AlphaECP. A cutting plane reselecting heuristic is, by default, called in the following cases:

a) If the MILP solver would return an infeasible solution.

b) The maximum violation is not decreasing and cutting planes are repeatedly moved close to their generation point.

c) When the violation is not decreasing and domain violations are repeatedly encountered.

The first case is obvious, instead of converging to an infeasible MINLP solution, a reselection of cuts is done. The second case is an early recognition of a probably infeasible MINLP. The updating (or movement) of cuts so that almost the same point can be found, is necessary to be able to make small changes in the variable levels. The chosen tolerance is 10^{-6}, defined as the maximum perpendicular distance between a cut and its generation point. Especially when the problem has non-linear equality constraints the high tolerance has shown itself to enable the solver to find better solutions. The drawback in this approach is that the convergence becomes computationally costly when a large number of almost identical MILP subproblems need to be solved. Furthermore, many times the convergence process results in an infeasible MINLP solution. Therefore, early recognition of a probably infeasible MINLP solution is implemented. Instead of completing the slow procedure we reselect cuts, since it is likely that the algorithm converges to an infeasible MINLP solution. In the third case we reselect the cuts when all constraints cannot be evaluated and we identify that the cuts do not reduce the maximum violation. In other words, the MILP solution point is not effectively moving closer to an MINLP feasible one.

Before the cut reselecting heuristic is described, some necessary notations are defined. Generating a cut refers to linearizing a non-linear constraint at a specific point. Including cuts refers to adding a cut or cuts to the next MILP problem. The cut or cuts are generated at the considered point. Cutting away refers to a linearization that prevents the MILP solver finding again the point that is cut away. A point is considered reachable if it is possible for the MILP solver to return the considered point as a solution. The cut reselection is done as follows:

1. From start no cuts are included in the MILP problem. The MILP problem consists of the linear constraints defined by the problem.

2. Choose the best point from the list of possible initial points, i.e. the previously found MILP solutions. Choose a cut violation level, defining whether a cut should

be included in the MILP problem at a considered point if its corresponding constraint violation is above the level value. The cut violation level is obtained from one of the violations at the best point. Then, include cuts at the best point.

3. Choose the closest still reachable point to the previous point, determined by the Euclidean norm. Repeat the procedure until all points are unreachable.

4. If no feasible MILP solution can be found, modify the cut level 6 times, each time increasing the cut level.

5. If no feasible MILP solution can be found, discard the best point from the possible initial points and repeat the procedure until the list of possible points is empty.

The list of possible initial points is that of the points found by the MILP sub-solver during the solution procedure. The initial point can be an MINLP solution, a point where all constraints can be evaluated, or a point where at least some cuts can be successfully generated. The best point is the best MINLP solution, if one has been found, otherwise the point that has the lowest *maxviol*, i.e. the lowest violation on the most violating constraint. When the cut level is changed it significantly modifies the final cut selection. The idea is that, at first, several cuts at each chosen MILP solution are included in the next MILP problem, whereas at the sixth modification, only a few cuts per point are included. Naturally at each considered and still reachable point, at least one cut needs to be included in order to ensure that the point can not be found again.

The use of multiple MILP solutions refers to selecting one MILP solution when the MILP solvers return several solutions. It is easy to convince oneself that an improved performance can be expected when the MILP point can be selected and the additional computational effort is small. Note that the most time consuming step is, in general, to solve the MILP subproblem and therefore, the solver can usually return several solutions with a relatively low additional computational effort. The point is selected from the obtained MILP solutions according to the following criteria:

1. Choose the point with **no** violations and best objective value.

2. Choose the point with the smallest gap between the most and least violating constraints, at a point where all the constraints can be evaluated.

3. Choose the point with the smallest gap between the most and least violating constraints and accept domain violations.

The following performance improvements can be noted: An MINLP solution might be found faster, a domain violating point can be avoided and a point with the smallest gap can be chosen. The idea behind choosing the point with the smallest gap is the following. If the gap is small, then the maximum violation is also small. If the gap is small, but the maximum violation is large, then all cuts effectively reduce the solution space. Furthermore, the likelihood that the whole solution space is cut away because of

cuts on nonlinear equality constraints[2] is reduced. The alpha updating will effectively and gradually enlarge the solution search space, i.e. the quality of cuts is improved.

In figures 3.2 and 3.3 we can observe the performance improvement of the algorithm changes discussed in the beginning of section 3.2. In this comparison the 270 problems found in the MinlpLib are solved with a 1000 second time limit per problem.

[2]If two cuts are generated close to each other for the same nonlinear equality constraint, where at one point the nonlinear constraint value is positive and the other negative, then the whole feasible region is likely to be cut away for a problem with several nonlinear constraints.

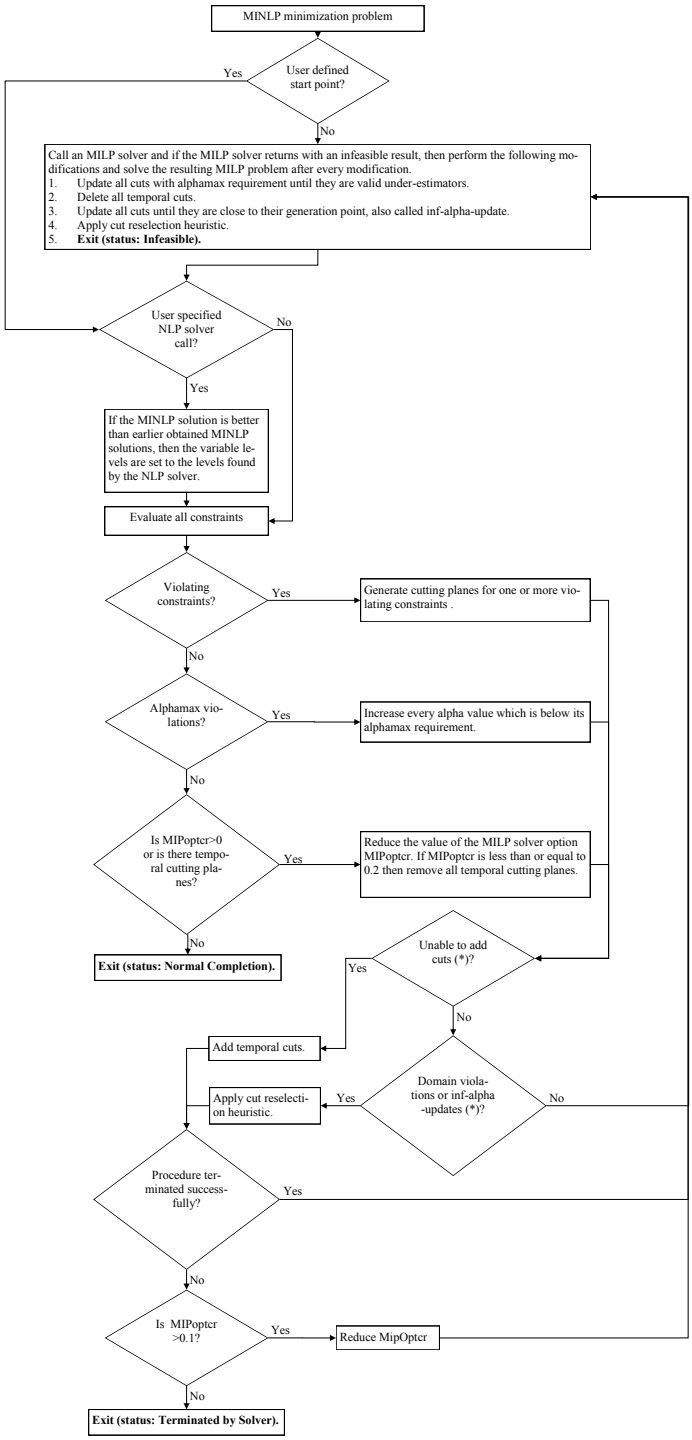

Figure 3.1: AlphaECP flowchart

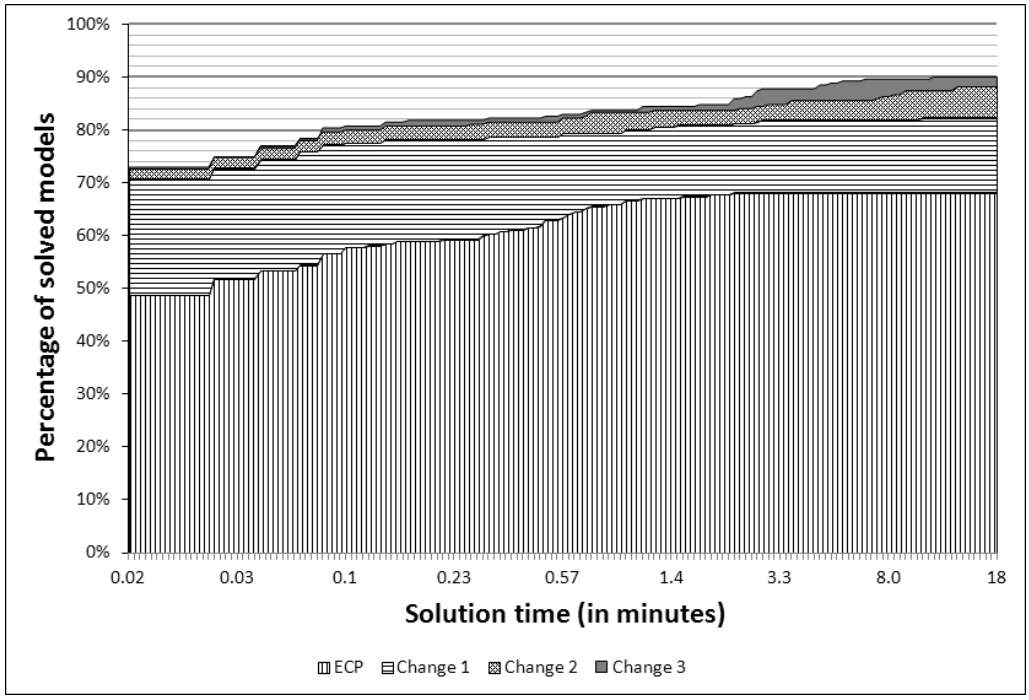

Figure 3.2: AlphaECP improvement: the percentage of found solutions.

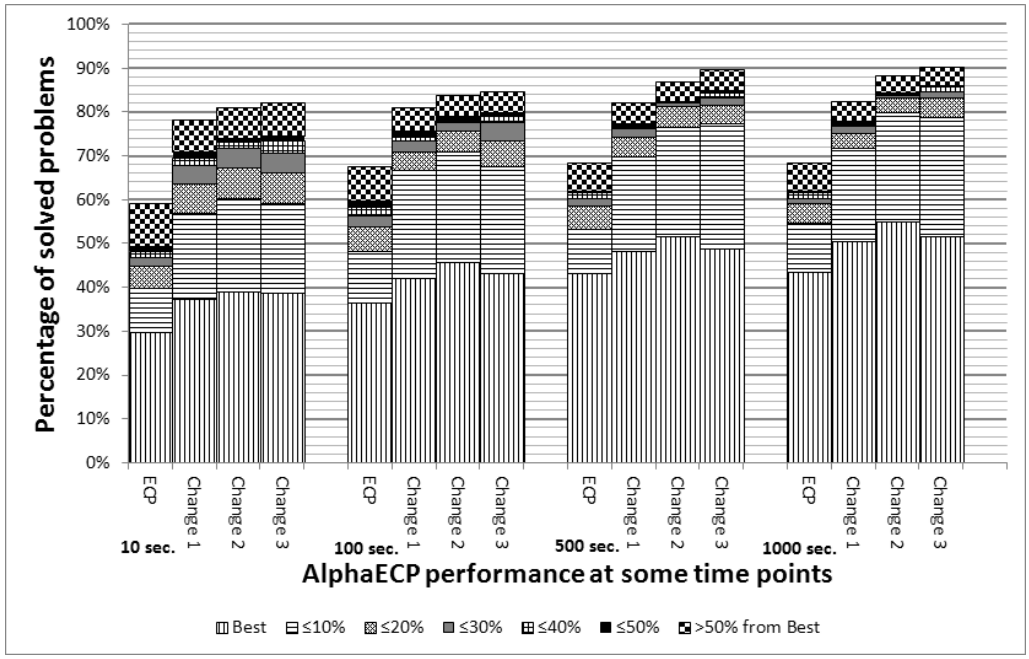

Figure 3.3: AlphaECP improvement: a solution quality comparison.

CHAPTER 4

Notes for the papers

This chapter is intended to give an overview of the papers which form an important part of this thesis. The papers are numbered in chronological order. Papers 1, 2, 3 and 4 consider solver benchmarking and analyze solving capability by comparing solver performance. Furthermore, algorithm advances and algorithmic settings are analyzed for AlphaECP. The main findings during the evolutionary process of writing these papers are presented in chapters 2 and 3. However, these papers include additional and problem-type specific information, which enrich the view of the strengths and weaknesses of some MINLP solvers. Additionally, this chapter specifies and corrects some points in the included papers. In figure 4.1 the considered papers can be seen and an indicative overview of the topics considered, in relation to each other, is given. The symbol (∗) denotes where this PhD thesis could be placed if appendix B is disregarded. The arrows denote the order in which the papers can be read if the aim is to study MINLP solver comparisons.

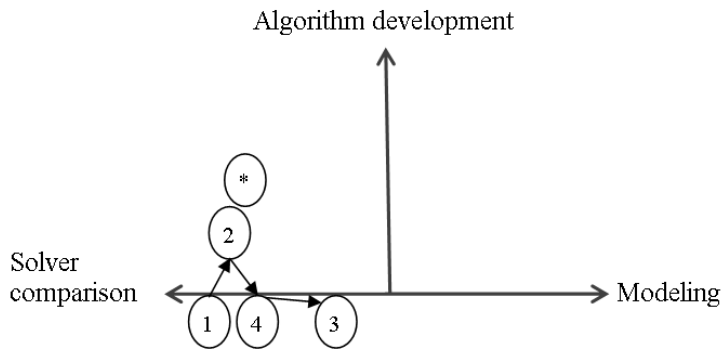

Figure 4.1: Publication content diagram.

4.1 Notes to paper I

This paper introduces AlphaECP, an Extended Cutting Plane implementation, on the General Algebraic Modeling System (GAMS) by presenting an MINLP solver comparison of four solvers. Furthermore, it presents a discovery that each of the four solvers is able to contribute to the overall solving capability of the optimization platform. The conclusion is based on a batch run with 250 models with a 1000 seconds ($\sim$ 17 minutes) time limit per problem. Moreover, note that the terms "optimal" and "global optimal" are used instead of the more correct term "best solution known".

4.2 Notes to paper II

A comparison of six MINLP solvers is presented, as well as, a closer analysis of AlphaECP. The analysis considers the use of different sub-solvers, algorithmic advances and the capability to solve NLP problems. The NLP model library [Website: GLOBAL Library] was used and consisted of 415 models. The MINLP comparison comprises of solving 265 models, 15 additional models to the comparison in Paper I, with both a 1000 seconds ($\sim$ 17 minutes) and a 6 hour time limit per problem. Note that, in general, there is not only one but many factors contributing to an improved solving capability of an optimization system. For example, the computing power of the test computer, the operating system, the optimization platform and/or solver improvements.

4.3 Notes to paper III

The solving capability of seven Mixed Integer Quadratic Constraint Programming (MIQCP) solvers is compared[1] for the problem of module identification in a complex network. Five problem instances are studied where, for each, instance four models were tested, including both convex and non-convex models.

4.4 Notes to paper IV

The solving capability of local MINLP methods for Quadratic Assignments Problems (QAP) is presented. Four MINLP solvers are compared with a compact non-convex MIQCP formulation for 50 instances where a time limit of 1 hour per problem is used. No new best solutions were found. It should be noted however, that the one hour time limit per instance may have been too low to give the solvers a reasonable chance to solve a difficult combinatorial QAP problem.

[1] MIQCP is a special case of MINLP.

CHAPTER 5

Conclusions and discussion

In this thesis the performance of some MINLP solvers has been studied[1]. The issues in preparing a fair comparison have been discussed and illustrated. Some advances of the GAMS/AlphaECP MINLP solver have been presented and analyzed by using performance profiles.

This work aims to give a survey of the solving capability of different MINLP solvers and an understanding of the strengths and weaknesses of solvers. Furthermore, it may aid a novice MINLP modeler to make a good solver selection by pointing out the fundamental difficulties in solving MINLP problems and the key differences in the solution approaches. To solve an academic or industrial MINLP problem, the use of a well-suited method and a good model can provide a more efficient way for solving the problem. Solver comparisons can support the solver choice and thus save time, but also point out unique strengths of some solving methods. Moreover, the study shows that no solver is universally faster and provide a higher quality solution than the other solution approaches existing on the GAMS optimization platform. However, due to the vast increase in computing power and new algorithms, a comparison is only valid for a short time. For this reason, systematic and continuously performed solver comparisons should be considered. Moreover, the issues pointed out in this thesis, when making a comparison, would be worthwhile tackling in order to improve solver comparisons. Furthermore, this could indirectly make it easier to choose a good solution approach for a particular model.

Despite the fact that the future will most probably be dominated by global MINLP solvers, the solution techniques are most likely based, at least partly, on finding local optimal solutions. Many global MINLP solvers are based on solving sequences of convex MINLP problems. A convex MINLP problem is again, for example in the ECP method,

[1] The latest solver comparisons are presented at:

- T. Lastusilta. MINLP Solver Comparisons, September 2011. URL `http://users.abo.fi/tlastusi/minlp_solver_comparisons/`

solved as a sequence of MILP or LP problems. Therefore, we might very well see successful new local optimization techniques be integrated into global solvers by using different frameworks. Note that a good heuristic approach might well be worthwhile if there is significant speedup in the solution procedure of the global solver and especially if the global solver can take further advantage of the work done by the heuristic procedure.

Bibliography

C.S. Adjiman, C.A. Schweiger, and C.A. Floudas. Mixed-Integer Nonlinear Optimization in Process Synthesis, 1998.

J. Bisschop. *AIMMS Optimization Modeling*, April 2011.

K.M. Björk. *A Global Optimization Method with some Heat Exchanger Network Applications.* PhD thesis, Åbo Akademi University, Biskopsgatan 8, 20500 Åbo, Finland, 2002.

P. Bonami, L.T. Biegler, A.R. Conn, G. Cornuéjols, I.E. Grossmann, C.D. 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.

M.R. Bussieck and S. Vigerske. MINLP Solver Software, April 2011. URL `http://www.fermentas.com/techinfo/nucleicacids/maplambda.htm`.

M.R. Bussieck, A.S. Drud, and A. Meeraus. MINLPLib - A Collection of Test Models for Mixed-Integer Nonlinear Programming. *INFORMS Journal on Computing*, 15(1): 114–119, 2003.

Z. Drezner, P. Hahn, and É. Taillard. Recent Advances for the Quadratic Assignment Problem with Special Emphasis on Instances that are Difficult for Meta-Heuristic Methods. *Annals of Operations Research*, 139:65–94, 2005.

S. Emet. *A Comparative Study of Solving Some Nonconvex MINLP problems.* PhD thesis, Åbo Akademi University, Biskopsgatan 8, 20500 Åbo, Finland, 2004.

T. Farkas, Y. Avramenko, A. Kraslawski, Z. Lelkes, and L. Nyström. Selection of a Mixed-Integer Nonlinear Programming (MINLP) Model of Distillation Column Synthesis by Case-Based Reasoning. *Ind. Eng. Chem. Res.*, 45:1935–1944, 2006.

C.A. Floudas. *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications.* Oxford University Press, 1995. ISBN 0195100565.

C.A. Floudas. *Deterministic Global Optimization: Theory, Methods, and Applications*, volume 37 of *Nonconvex Optimization and Its Applications.* Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000. ISBN 0-7923-6014-1.

R. Fourer, D.M. Gay, and B.M. Kernighan. A Modeling Language for Mathematical Programming. *Management Science*, 36:519–554, 1990.

GAMS, 2011. *GAMS - The Solver Manuals*. GAMS Development Corporation, Washington, DC, USA, July 2011.

I.E. Grossmann and Z. Kravanja. Mixed-Integer Nonlinear Programming Techniques for Process Systems Engineering. *Computers Chem. Engng.*, 19:189–204, 1995.

I. Harjunkoski. *Application of MINLP Methods to a Scheduling Problem in the Paper-Converting Industry*. PhD thesis, Åbo Akademi University, Biskopsgatan 8, 20500 Åbo, Finland, 1997.

P. Jernström. *Multi-Product Multi-Purpose Machine Scheduling in Dynamic Environments*. PhD thesis, Åbo Akademi University, Biskopsgatan 8, 20500 Åbo, Finland, 2006.

J. E. Kelley. The Cutting Plane Method for Solving Convex Programs. *Journal of SIAM*, VIII(4):703–712, 1960.

G.R. Kocis and I.E. Grossmann. Computational experience with DICOPT: Solving MINLP problems in process systems engineering. *Computers Chem. Engng.*, 13(3): 307–315, 1989.

T. Lastusilta. An Implementation of the Extended Cutting Plane Method in GAMS. Master's thesis, Åbo Akdemi University, 2007.

T. Lastusilta. MINLP Solver Comparisons, September 2011. URL `http://users.abo.fi/tlastusi/minlp_solver_comparisons/`.

T. Lastusilta and T. Westerlund. A Comparative Study of Solving Quadratic Assignment Problems Using Some Standard MINLP Solvers. In *SIMULTECH 2011 1st International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, pages 409–412, 2011.

T. Lastusilta, M. R. Bussieck, and T. Westerlund. Comparison of Some High-Performance MINLP Solvers. In *Chem. Eng. Trans.*, volume 11, pages 125–130, 2007.

T. Lastusilta, M. R. Bussieck, and T. Westerlund. An Experimental Study of GAMS/AlphaECP MINLP Solver. *Ind. Eng. Chem. Res.*, 48:7337–7345, 2009a.

T. Lastusilta, O. Frankenhaeuser, F. Pettersson, and T. Westerlund. An Optimization Framework for Solving a Large Scale Scheduling Problem. In *European Symposium on Computer Aided Process Engineering*, volume 19, pages 525–529, 2009b.

T. Lastusilta, L.G. Papageorgiou, and T. Westerlund. A Comparative Study of Solving the Problem of Module Identification in a Complex Network. In *Chem. Eng. Trans.*, volume 24, pages 319–324, 2011.

S. Leyffer. Integrating SQP and Branch-and-Bound for Mixed Integer Nonlinear Programming. *Computational Optimization and Applications*, 18:295–309, 2001.

LINDO, 2011. *LINDO API 7.0 User Manual*. LINDO Systems Inc., Chicago, Illinois, USA, April 2011.

R. Lougee-Heimer. The Common Optimization INterface for Operations Research. *IBM Journal of Research and Development*, 47:57–66, 2003.

G.P. McCormick. Computability of global solutions to factorable nonconvex programs: Part I - Convex underestimating problems. *Mathematical Programming*, 10:147–175, 1976.

F. Pettersson. *Mixed Integer Non-Linear Programming Applied on Pump Configurations*. PhD thesis, Åbo Akademi University, Biskopsgatan 8, 20500 Åbo, Finland, 1994.

N.V. Sahinidis. Branch And Reduce Optimization Navigator User's Manual Version 4.0, June 2000. URL `http://archimedes.cheme.cmu.edu/baron/manuse.pdf`.

L. Schrage and C. Gau. Implementation and Testing of a Branch-and-Bound Based Method for Deterministic Global Optimization: Operations Research Applications. In C.A. Floudas and P.M. Pardalos, editors, *Frontiers in Global Optimization*, pages 145–164. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2003. ISBN 978-1-4020-7699-2.

L. Schrage and Y. Lin. The global solver in the LINDO API. *Optimization Methods & Software*, 24:657–668, 2009.

E.A. Silver. An overview of heuristic solution methods. *Journal of the Operational Research Society*, 55:936–956, 2004.

R.M. Soland and J.E. Falk. An algorithm for separable nonconvex programming problems. *Manag. Sci.*, 15:550–569, 1969.

M. Tawarmalani and N.V. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*. Kluwer Academic Publishers, 2002. ISBN 978-1-4020-1031-6.

M. Tawarmalani and N.V. Sahinidis. Global Optimization of Mixed-Integer Nonlinear Programs: A Theoretical and Computational Study. *Math. Program.*, 99:563–591, 2004.

R.J. Vanderbei. *Linear Programming: Foundations and Extensions, 2nd Edition*. Springer, 2001. ISBN 0792373421.

Website: BARON. Schematic GAMS/BARON connection diagram, September 2011. URL `http://www.gams.com/solvers/gamsbaron_connect.gif`.

Website: COIN-OR. COmputational INfrastructure for Operations Research, September 2011. URL `http://www.coin-or.org/`.

Website: Convex MINLP Test Problems. CMU-IBM Open source MINLP Project, August 2011. URL `http://egon.cheme.cmu.edu/ibm/page.htm`. Refering to the convex MINLP models in GAMS format found at the website.

Website: GLOBAL Library. GLOBAL World, July 2011. URL `http://www.gamsworld.org/minlp/`.

Website: Global Optimization Test Libraries. Global Optimization Test Problems, September 2011. URL `http://www.mat.univie.ac.at/~neum/glopt/test.html`.

Website: MINLP Library. MINLP Library, July 2011. URL `http://www.gamsworld.org/minlp/`.

Website: QAP Library. QAPLIB - A Quadratic Assignment Problem Library, July 2011. URL `http://www.seas.upenn.edu/qaplib/`.

J. Westerlund. *Aspects of N-dimensional allocation*. PhD thesis, Åbo Akademi University, Biskopsgatan 8, 20500 Åbo, Finland, 2005.

T. Westerlund and F. Pettersson. An Extended Cutting Plane Method for Solving Convex MINLP Problems. *Computers Chem. Engng. Sup.*, 19:131–136, 1995.

T. Westerlund and R. Pörn. Solving Pseudo-Convex Mixed Integer Optimization Problems by Cutting Plane Techniques. *Optimization and Engineering*, 3:253–280, 2002.

T. Westerlund, Skrifvars H., I. Harjunkoski, and R. Pörn. An Extended Cutting Plane Method for Solving a Class of Non-Convex MINLP Problems. *Computers Chem. Engng.*, 22:357–365, 1998.

Appendices

APPENDIX A

Test run model names

The models used in the performed test run without postfix ".gms":

[Website: MINLP Library]
4stufen, alan, batch, batchdes, beuster, blendgap, cecil_13, chp_partload, contvar, csched1, csched1a, csched2, csched2a, deb10, deb6, deb7, deb8, deb9, detf1, dosemin2d, dosemin3d, du-opt, du-opt5, eg_all_s, eg_disc2_s, eg_disc_s, eg_int_s, elf, eniplac, enpro48, enpro48pb, enpro56, enpro56pb, ex1221, ex1222, ex1223, ex1223a, ex1223b, ex1224, ex1225, ex1226, ex1233, ex1243, ex1244, ex1252, ex1252a, ex1263, ex1263a, ex1264, ex1264a, ex1265, ex1265a, ex1266, ex1266a, ex3, ex3pb, ex4, fac1, fac2, fac3, feedtray, feedtray2, fo7, fo7_2, fo7_ar25_1, fo7_ar2_1, fo7_ar3_1, fo7_ar4_1, fo7_ar5_1, fo8, fo8_ar25_1, fo8_ar2_1, fo8_ar3_1, fo8_ar4_1, fo8_ar5_1, fo9, fo9_ar25_1, fo9_ar2_1, fo9_ar3_1, fo9_ar4_1, fo9_ar5_1, fuel, fuzzy, gasnet, gastrans, gbd, gear, gear2, gear3, gear4, gkocis, hda, hmittelman, johnall, lop97ic, lop97icx, m3, m6, m7, m7_ar25_1, m7_ar2_1, m7_ar3_1, m7_ar4_1, m7_ar5_1, mbtd, meanvarx, meanvarxsc, minlphix, netmod_dol1, netmod_dol2, netmod_kar1, netmod_kar2, no7_ar25_1, no7_ar2_1, no7_ar3_1, no7_ar4_1, no7_ar5_1, nous1, nous2, nuclear104, nuclear10a, nuclear10b, nuclear14, nuclear14a, nuclear14b, nuclear24, nuclear24a, nuclear24b, nuclear25, nuclear25a, nuclear25b, nuclear49, nuclear49a, nuclear49b, nuclearva, nuclearvb, nuclearvc, nuclearvd, nuclearve, nuclearvf, nvs01, nvs02, nvs03, nvs04, nvs05, nvs06, nvs07, nvs08, nvs09, nvs10, nvs11, nvs12, nvs13, nvs14, nvs15, nvs16, nvs17, nvs18, nvs19, nvs20, nvs21, nvs22, nvs23, nvs24, o7, o7_2, o7_ar25_1, o7_ar2_1, o7_ar3_1, o7_ar4_1, o7_ar5_1, o8_ar4_1, o9_ar4_1, oaer, oil, oil2, ortez, parallel, pb302035, pb302055, pb302075, pb302095, pb351535, pb351555, pb351575, pb351595, prob02, prob03, prob10, procsel, product, product2, pump, qap, qapw, ravem, ravempb, risk2b, risk2bpb, saa_2, sep1, space25, space25a, space960, spectra2, spring, stockcycle, st_e13, st_e14, st_e15, st_e27, st_e29, st_e31, st_e32, st_e35, st_e36, st_e38, st_e40, st_miqp1, st_miqp2, st_miqp3, st_miqp4, st_miqp5, st_test1, st_test2, st_test3,

st_test4, st_test5, st_test6, st_test8, st_testgr1, st_testgr3, st_testph4, super1, super2, super3, super3t, synheat, synthes1, synthes2, synthes3, tln12, tln2, tln4, tln5, tln6, tln7, tloss, tls12, tls2, tls4, tls5, tls6, tls7, tltr, uselinear, util, var_con10, var_con5, waste, water3, water4, waterful2, waters, watersbp, watersym1, watersym2, waterx, waterz, windfac.

Website: Global Optimization Test Libraries
BatchS101006M, BatchS121208M, BatchS151208M, BatchS201210M, CLay0203H, CLay0203M, CLay0204H, CLay0204M, CLay0205H, CLay0205M, CLay0303H, CLay0303M, CLay0304H, CLay0304M, CLay0305H, CLay0305M, FLay02H, FLay02M, FLay03H, FLay03M, FLay04H, FLay04M, FLay05H, FLay05M, FLay06H, FLay06M, fo7, fo7_2, fo8, fo9, o7, o7_2, RSyn0805H, RSyn0805M, RSyn0805M02H, RSyn0805M02M, RSyn0805M03H, RSyn0805M03M, RSyn0805M04H, RSyn0805M04M, RSyn0810H, RSyn0810M, RSyn0810M02H, RSyn0810M02M, RSyn0810M03H, RSyn0810M03M, RSyn0810M04H, RSyn0810M04M, RSyn0815H, RSyn0815M, RSyn0815M02H, RSyn0815M02M, RSyn0815M03H, RSyn0815M03M, RSyn0815M04H, RSyn0815M04M, RSyn0820H, RSyn0820M, RSyn0820M02H, RSyn0820M02M, RSyn0820M03H, RSyn0820M03M, RSyn0820M04H, RSyn0820M04M, RSyn0830H, RSyn0830M, RSyn0830M02H, RSyn0830M02M, RSyn0830M03H, RSyn0830M03M, RSyn0830M04H, RSyn0830M04M, RSyn0840H, RSyn0840M, RSyn0840M02H, RSyn0840M02M, RSyn0840M03H, RSyn0840M03M, RSyn0840M04H, RSyn0840M04M, SLay04H, SLay04M, SLay05H, SLay05M, SLay06H, SLay06M, SLay07H, SLay07M, SLay08H, SLay08M, SLay09H, SLay09M, SLay10H, SLay10M, Syn05H, Syn05M, Syn05M02H, Syn05M02M, Syn05M03H, Syn05M03M, Syn05M04H, Syn05M04M, Syn10H, Syn10M02H, Syn10M02M, Syn10M03H, Syn10M03M, Syn10M04H, Syn10M04M, Syn15H, Syn15M, Syn15M02H, Syn15M02M, Syn15M03H, Syn15M03M, Syn15M04H, Syn15M04M, Syn20H, Syn20M, Syn20M02H, Syn20M02M, Syn20M03H, Syn20M03M, Syn20M04H, Syn20M04M, Syn30H, Syn30M, Syn30M02H, Syn30M02M, Syn30M03H, Syn30M04H, Syn30M04M, Syn40H, Syn40M, Syn40M02H, Syn40M02M, Syn40M03H, Syn40M03M, Syn40M04H, Syn40M04M.

APPENDIX B

Papers I-IV

I

II

III

IV

ISBN 978-952-12-2653-3

Electronic version
Åbo 14.11.2011