Chapter 9
A Good Recipe for Solving MINLPs

Leo Liberti, Giacomo Nannicini, and Nenad Mladenović

Abstract Finding good (or even just feasible) solutions for Mixed-Integer Nonlinear Programming problems independently of the specific problem structure is a very hard but practically useful task, especially when the objective and/or the constraints are nonconvex. We present a general-purpose heuristic based on Variable Neighbourhood Search, Local Branching, Sequential Quadratic Programming and Branch-and-Bound. We test the proposed approach on the MINLPLib, discussing optimality, reliability and speed.

9.1 Introduction

The mathematical programming formulation $\min\{f(x) \mid g(x) \leq 0\}$ can be ascribed to four different categories: Linear Programming (LP) if f, g are linear forms and $x \in \mathbb{R}^n$ are continuous variables, Mixed-Integer Linear Programming (MILP) if some of the variables are integer, Nonlinear Programming (NLP) if there are some nonlinear functions in f, g and the variables are continuous, Mixed-Integer Nonlinear Programming (MINLP) if f, g involve nonlinear functions and the vector x includes some integer variables; problems are also categorized according to the convexity of objective function and constraints. In general, solving LPs and convex NLPs is considered easy, and solving MILPs, nonconvex NLPs and convex MINLPs (cMINLPs) is considered difficult. Solving nonconvex MINLPs involves difficulties arising from

Leo Liberti · Giacomo Nannicini
LIX, École Polytechnique, Palaiseau, France
e-mail: `{liberti,giacomon}@lix.polytechnique.fr`

Nenad Mladenović
Brunel University, London, UK and Institute of Mathematics, Academy of Sciences, Belgrade, Serbia
e-mail: `nenad.mladenovic@brunel.ac.uk,nenad@turing.mi.sanu.ac.yu`

both nonconvexity and integrality, and it is considered the hardest problem of all. From the modelling point of view, however, nonconvex MINLPs are the most expressive mathematical programs — it stands to reason, then, that general-purpose MINLP solvers should be very useful. Currently, optimal solutions of MINLPs in general form are obtained by using the spatial Branch-and-Bound (sBB) algorithm [2, 29, 38, 39]; but guaranteed optima can only be obtained for relatively small-sized MINLPs. Realistically-sized MINLPs can often have thousands (or tens of thousands) of variables (continuous and integer) and nonconvex constraints. With such sizes, it becomes a great challenge to even find a feasible solution, and sBB algorithms become almost useless. Some good solvers targeting cMINLPs exist in the literature [1, 4, 6, 16, 17, 27]; although they can all be used on nonconvex MINLPs as well (forsaking the optimality guarantee), in practice their mileage varies wildly with the instance of the problem being solved, resulting in a high fraction of "false negatives" (i.e. feasible problems for which no feasible solution was found). The Feasibility Pump (FP) idea was recently extended to cMINLPs [5], but again this does not work so well when applied to nonconvex MINLPs unmodified [35].

In this chapter, we propose an effective and reliable MINLP heuristic, called the Relaxed-Exact Continuous-Integer Problem Exploration (RECIPE) algorithm. The MINLPs we address are cast in the following general form:

$$\left.\begin{array}{rrcl} \min\limits_{x\in\mathbb{R}^n} & & f(x) & \\ \text{s.t.} & l \leq & g(x) & \leq u \\ & x^L \leq & x & \leq x^U \\ & x_i \in \mathbb{Z} & & \forall\, i \in Z \end{array}\right\} \tag{9.1}$$

In the above formulation, x are the decision variables (x_i is integer for each $i \in Z$ and continuous for each $i \notin Z$, where $Z \subseteq \{1, \ldots, n\}$). $f : \mathbb{R}^n \to \mathbb{R}$ is a possibly nonlinear function, $g : \mathbb{R}^n \to \mathbb{R}^m$ is a vector of m possibly nonlinear functions (assumed to be differentiable), $l, u \in \mathbb{R}^m$ are the constraint bounds (which may be set to $\pm\infty$), and $x^L, x^U \in \mathbb{R}^n$ are the variable bounds.

RECIPE puts together a global search phase based on Variable Neighbourhood Search (VNS) [22] and a local search phase based on a Branch-and-Bound (BB) type heuristic. The VNS global phase rests on neighbourhoods defined as hyperrectangles for the continuous and general integer variables [30] and by Local Branching (LB) for the binary variables [15]. The local phase employs a BB solver for convex MINLPs [17], and applies it to (possibly nonconvex) MINLPs, making therefore effectively a heuristic. A local NLP solver, which implements a Sequential Quadratic Programming (SQP) algorithm [20], supplies an initial constraint-feasible solution to be employed by the BB as starting point. RECIPE is an efficient, effective and reliable general-purpose algorithm for solving complex MINLPs of small and medium scale.

The original contribution of this chapter is the way a set of well-known and well-tested tools are combined into making a very powerful global optimization method. This chapter does not contribute theoretical knowledge but rather the description of a practically useful algorithm whose easy implementation rests on existing off-the-shelf software tools complemented by relatively few lines of code. It turns out that RECIPE, when acting on the whole MINLPLib library [9], is able to find optima equal to or better than the best solutions reported in the literature for 55% of the instances. The closest competitor is SBB+CONOPT [10, 13], which matches or surpasses the best solutions listed in MINLPLib on only 37% of the instances. We improve the best known solutions in 7% of the cases.

The rest of this chapter is organized as follows. In Section 9.2 we describe the basic component algorithms on which RECIPE is based. Section 9.3 presents the overall approach. In Section 9.4 we discuss computational results obtained over MINLPLib, focusing on optimality, reliability and speed. Section 9.5 concludes the chapter.

9.2 The Basic Ingredients

This section describes the four main components used in RECIPE, which are:

- the global search phase: Variable Neighbourhood Search;
- the binary variable neighbourhood definition technique: Local Branching;
- the constraint and integral feasibility enforcing local solution algorithm: Branch-and-Bound for cMINLPs;
- the constraint feasibility enforcing local solution algorithm: Sequential Quadratic Programming.

9.2.1 Variable Neighbourhood Search

VNS relies on iteratively exploring neighbourhoods of growing size to identify better local optima [22, 23, 24]. More precisely, VNS escapes from the current local minimum x^* by initiating other local searches from starting points sampled from a neighbourhood of x^* which increases its size iteratively until a local minimum better than the current one is found. These steps are repeated until a given termination condition is met. This can be based on CPU time, number of non-improving steps and other configurable parameters.

VNS has been applied to a wide variety of problems both from combinatorial and continuous optimization [3, 7, 12, 26, 31, 32, 37]. Its early applications to continuous problems were based on a particular problem structure. In the continuous location-allocation problem, the neighbourhoods are defined according to the meaning of problem variables (assignments of facilities to customers, positioning of yet unassigned facilities and so on) [7]. In bilinearly constrained bilinear problems the neighbourhoods are defined in terms of the applicability of the successive linear programming approach, where the problem variables can be partitioned so that fixing the variables in either set yields a linear problem; more precisely, the neighbourhoods of size k are defined as the vertices of the LP polyhedra that are k pivots away from the current vertex [22]. The first VNS algorithm targeted at problems with fewer structural requirements, namely, box-constrained nonconvex NLPs, was given in [36] (the paper focuses on a particular class of box-constrained NLPs, but the proposed approach is general). Its implementation is described in [11]. Since the problem is assumed to be box-constrained, the neighbourhoods arise naturally as hyperrectangles of growing size centered at the current local minimum x^*. The same neighbourhoods were used in [30], an extension of VNS to constrained NLPs.

9.2.2 Local Branching

LB is an efficient heuristic for solving difficult MILP problems [15]. Given an integer $k > 0$, the LB search explores k-neighbourhoods of the incumbent x^* by allowing at most k of the integer variables to change their value; this condition is enforced by means of the *local branching constraint*:

$$\sum_{i \in \bar{S}} (1 - x_i) + \sum_{i \notin \bar{S}} x_i \leq k, \tag{9.2}$$

where $\bar{S} = \{i \leq n \mid i \in Z \wedge x_i^* = 1\}$, which defines a neighbourhood of radius k with respect to the binary variables of (9.1), centered at a binary solution with support $\bar{S}$. LB updates the incumbent as it finds better solutions. When this happens, the LB procedure is called iteratively with $\bar{S}$ relative to the new incumbent. We remark that LB was successfully used in conjunction with VNS in [25].

9.2.3 Branch-and-Bound for cMINLPs

Solving cMINLPs (i.e. MINLPs where the objective function and constraints are convex — the terminology is confusing as all MINLPs are actually nonconvex problems because of the integrality constraints) is conceptually not much more difficult than solving MILPs: as the relaxed problem is convex, obtaining lower bounds is easy. The existing tools, however, are still far from the quality attained by modern MILP solvers. The problem is usually solved by BB, where only the integer variables are selected for branching. A restricted (continuous) convex NLP is formed and solved at each node, where the variable ranges have been restricted according to the node's definition. Depending on the algorithm, the lower bounding problem at each node may either be the original problem with relaxed integrality constraints [10, 17] (in which case the BB becomes a recursive search for a solution that is both integer feasible and a local optimum in continuous space), or its linear relaxation by outer approximation [1, 4, 14, 16]. In the former case, the restricted NLP is solved to optimality at each node by using local NLP methods (which converge to the node's global optimum when the problem is convex) such as SQP (see Section 9.2.4), in the latter it is solved once in a while to get good incumbent candidates.

Another approach to solving MINLPs, which can be applied to convex and pseudoconvex objective and constraints alike, is taken in [40, 41, 42], where a cutting planes approach is blended in with a sequence of MILP subproblems (which only need to be solved to feasibility).

These approaches guarantee an optimal solution if the objective and constraints are convex, but may be used as a heuristic even in presence of nonconvexity. Within this chapter, we employ these methods in order to find local optima of general (nonconvex) MINLPs. The problem of finding an initial feasible starting point (used by the BB local NLP subsolver) is addressed by supplying the method with a constraint feasible (although not integer feasible) starting point found by an SQP algorithm (see Section 9.2.4).

9.2.4 Sequential Quadratic Programming

SQP methods find local solutions to nonconvex NLPs. They solve a sequence of quadratic approximations of the original problem subject to a linearization of its constraints. The quadratic approximation is obtained by a convex model of the objective function Hessian at a current solution point, subject to a linearization of the (nonlinear) constraints around the current point. SQP methods are now at a very advanced stage [20], with corresponding implementations being able to warm- or cold-start. In particular, they deal with the problem of infeasible linear constraints (this may happen as the linearization around a point of a set of feasible nonlinear constraints is not always feasible), as well as the feasibility of the starting point with respect to the nonlinear constraints. This case is dealt with by elastic programming [21]. In particular, `snopt` does a good job of finding a constraint feasible point out of any given initial point, even for reasonably large-scale NLPs. By starting a local MINLP solver from a constraint feasible starting point, there are better chances that an integer feasible solution may be found.

9.3 The RECIPE Algorithm

Our main algorithm is a heuristic exploration of the problem solution space by means of an alternating search between the relaxed NLP and the exact MINLP. This is a two-phase global optimization method. Its local phase consists in using the SQP algorithm for solving relaxed (nonconvex) NLPs locally; next, the BB algorithm is used for solving exact (nonconvex) MINLPs to feasibility. The global phase of the algorithm is given by a VNS using two separate neighbourhoods for continuous and general integer variables and for binary variables. The former neighbourhoods have hyper-rectangular shape; the latter are based on a LB constraint involving all binary variables.

We consider a (nonconvex) MINLP P given by formulation (9.1), with its continuous relaxation $\bar{P}$. Let $B = \{i \in Z \mid x_i^L = 0 \wedge x_i^U = 1\}$ be the set of indices of the binary variables, and $\bar{B} = \{1, \ldots, n\} \setminus B$ the set of indices of others, including general integer and continuous variables. Let $Q(\bar{x}, k, k_{\max})$ be its reformulation obtained by adding a local branching constraint

$$\sum_{i \in B} (\bar{x}_i(1 - x_i) + (1 - \bar{x}_i)x_i) \leq \left\lceil k \frac{|B|}{k_{\max}} \right\rceil, \tag{9.3}$$

where $\bar{x}$ is a (binary) feasible solution (e.g. obtained at a previous iteration), $k_{\max} \in \mathbb{N}$ and $k \in \{1, \ldots, k_{\max}\}$. At each VNS iteration (with a certain associated parameter k), we obtain an initial point $\tilde{x}$, where $\tilde{x}_i$ is sampled in a hyperrectangular neighbourhood of radius k for $i \in \bar{B}$ (rounding where necessary for $i \in Z \setminus B$) and $\tilde{x}_i$ is chosen randomly for $i \in B$. We then solve the continuous relaxation $\bar{P}$ locally by means of an SQP method using $\tilde{x}$ as a starting point, and obtain $\bar{x}$ (if $\bar{x}$ is not feasible with respect to the constraints of P, then $\bar{x}$ is re-set to $\tilde{x}$ for possibly having a better choice). We then use a BB method for cMINLPs in order to solve $Q(\bar{x}, k, k_{\max})$, obtaining a solution x'. If x' improves on the incumbent x^*, then x^* is replaced by x' and k is reset to 1. Otherwise (i.e. if x' is worse than x^* or if $Q(\bar{x}, k, k_{\max})$ could not be solved), k is increased in a VNS-like fashion. The algorithm is described formally in Algorithm 1.

9.3.1 Hyperrectangular Neighbourhood Structure

We discuss here the neighbourhood structure for $N_k(x)$ for the RECIPE algorithm.

Consider hyperrectangles $H_k(x)$, centered at $x \in \mathbb{R}^n$ and proportional to the hyperrectangle $x^L \leq x \leq x^U$ given by the original variable bounds, such that $H_{k-1}(x) \subset H_k(x)$ for each $k \leq k_{\max}$. More formally, let $H_k(x^*)$ be the hyperrectangle $y^L \leq x \leq y^U$ where, for all $i \notin Z$,

Algorithm 1: The RECIPE algorithm.

```
INPUT: Neighbourhoods $N_k(x)$ for $x \in \mathbb{R}^n$;
          maximum neighbourhood radius $k_{\max}$;
          number $L$ of local searches in each neighbourhood.
OUTPUT: Best solution found $x^*$.
Set $x^* = x^L/2 + x^U/2$
while (!time-based termination condition) do
  Set $k \leftarrow 1$
  while ($k \le k_{\max}$) do
    for ($i = 1$ to $L$) do
      Sample a random point $\tilde{x}$ from $N_k(x^*)$.
      Solve $\bar{P}$ using an SQP algorithm from initial point $\tilde{x}$ obtaining $\bar{x}$
      if ($\bar{x}$ is not feasible w.r.t. the constraints of $P$) then
        $\bar{x} = \tilde{x}$
      end if
      Solve $Q(\bar{x}, k, k_{\max})$ using a BB algorithm from initial point $\bar{x}$ obtaining $x'$
      if ($x'$ is better than $x^*$) then
        Set $x^* \leftarrow x'$
        Set $k \leftarrow 0$
        Exit the FOR loop
      end if
    end for
    Set $k \leftarrow k + 1$.
  end while
end while
```

$$\begin{aligned} y_i^L &= x_i^* - \frac{k}{k_{\max}}(x_i^* - x_i^L) \\ y_i^U &= x_i^* + \frac{k}{k_{\max}}(x_i^U - x_i^*), \end{aligned}$$

for all $i \in Z \setminus B$,

$$\begin{aligned} y_i^L &= \lfloor x_i^* - \frac{k}{k_{\max}}(x_i^* - x_i^L) + 0.5 \rfloor \\ y_i^U &= \lfloor x_i^* + \frac{k}{k_{\max}}(x_i^U - x_i^*) + 0.5 \rfloor, \end{aligned}$$

and for all $i \in B$, $y^L = 0$ and $y^U = 1$.

We let $N_k(x) = H_k(x) \backslash H_{k-1}(x)$. This neighbourhood structure defines a set of hyperrectangular nested shells with respect to continuous and general integer variables. Let τ be the affine map sending the hyperrectangle $x^L \le x \le x^U$ into the unit L_∞ ball (i.e., hypercube) $\mathcal{B}$ centered at 0, i.e., $\mathcal{B} = \{x : |x_i| \le 1 \forall i\}$. Let $r_k = \frac{k}{k_{\max}}$ be the radii of the balls $\mathcal{B}_k$ (centered at 0) such that $\tau(H_k(x)) = \mathcal{B}_k$ for each $k \le k_{\max}$. In order to sample a random vector $\tilde{x}$ in $\mathcal{B}_k \backslash \mathcal{B}_{k-1}$ we proceed as in Algorithm 2.

The sampled point $\tilde{x}$ will naturally not be feasible in the constraints of (9.1), but we can enforce integral feasibility by rounding $\tilde{x}_j$ to the nearest integer for $j \in Z$, i.e. by setting $\tilde{x}_j \leftarrow \lfloor \tilde{x}_j + 0.5 \rfloor$. This will be rather ineffective with the binary variables x_j, which would keep the same value $\tilde{x}_j = x^*_j$ for each $k \leq \frac{k_{\max}}{2}$. Binary variables are best dealt with by solving the LB reformulation Q in Algorithm 1.

9.4 Computational Results

Algorithm 1 presents many implementation difficulties: the problem must be reformulated iteratively with the addition of a different LB constraint at each iteration; different solvers acting on different problem formulations must be used. All this must be coordinated by the outermost VNS at the global level. We chose AMPL [19] as a scripting language because it makes it very easy to interface to many external solvers. Since AMPL cannot generate the reformulation Q of P iteratively independently of the problem structure, we employed a C++ program that reads an AMPL output `.nl` file in flat form [29] and outputs the required reformulation as an AMPL-readable `.mod` file.

The `minlp_bb` solver [27] was found to be the MINLP solver that performs best when finding feasible points in nonconvex MINLPs (the comparison was carried out with the default-configured versions of `filMINT` [1] and `BonMin` [6]). The SQP solver of choice was `snopt` [21], found to be somewhat more reliable than `filtersqp` [18]: on the analysed test set, `snopt` achieves, on average, better results at finding feasible solution in a short CPU time. All computational results have been obtained on an Intel Xeon 2.4 GHz with 8 GB RAM running Linux.

RECIPE rests on three configurable parameters: $k_{\max}$ (the maximum neighbourhood radius), L (the number of local searches starting in each neighbourhood) and the maximum allowed user CPU time (not including the time taken to complete the last local search). After some practical experimentation on a reduced subset of instances, we set $k_{\max} = 50$, $L = 15$ and the maximum CPU time to 10h. These parameters were left unchanged over the whole test set, yielding good results without the need for fine-tuning.

Algorithm 2: Sampling in the shell neighbourhoods.

```
INPUT: $k, k_{\max}$.
OUTPUT: A point $\tilde{x}$ sampled in $H_k(x)\backslash H_{k-1}(x)$.
Sample a random direction vector $d \in \mathbb{R}^n$
Normalize $d$ (i.e., set $d \leftarrow \frac{d}{||d||_\infty}$)
Let $r_{k-1} = \frac{k-1}{k_{\max}}$, $r_k = \frac{k}{k_{\max}}$
Sample a random radius $r \in [r_{k-1}, r_k]$ yielding a uniformly distributed point in the shell
Let $\tilde{x} = \tau^{-1}(rd)$
```

9.4.1 MINLPLib

The MINLPLib [9] is a collection of Mixed Integer Nonlinear Programming models which can be searched and downloaded for free. Statistics for the instances in the MINLPLib are available at `http://www.gamsworld.org/minlp/minlplib/minlpstat.htm`. The instance library is available at `http://www.gamsworld.org/minlp/minlplib.htm`. The MINLPLib is distributed in GAMS [8] format, so we employed an automatic translator to cast the files in AMPL format.

At the time of downloading (Feb. 2008), the MINLPLib consisted of 265 MINLP instances contributed by the scientific and industrial OR community. These were all tested with the RECIPE algorithm implementation described above. We had 20 unsuccessful runs due to some AMPL-related errors (the model contained some unusual AMPL operator not implemented by some of the solvers/reformulators employed in RECIPE). The instances leading to AMPL-related failure were:

`blendgap`, `dosemin2d`, `dosemin3d`, `fuzzy`, `hda`, `meanvarxsc`, `pb302035`, `pb302055`, `pb302075`, `pb302095`, `pb351535`, `pb351555`, `pb351575`, `pb351595`, `water3`, `waterful2`, `watersbp`, `waters`, `watersym1`, `watersym2`.

The performance of RECIPE was evaluated on the 245 runs that came to completion. The results are given in Tables 9.1, 9.2 (solved instances) and 9.3 (unsolved instances). Table 9.1 lists results where the best solution found by RECIPE was different by at least 0.1% from that listed in MINLPLib. The first column contains the instance name, the second contains the value f^* of the objective function found by the RECIPE algorithm and the third the corresponding CPU usage measured in seconds of user time; the fourth contains the value $\bar{f}$ of the objective function reported in the official MINLPLib table and the fifth contains the name of the corresponding GAMS solver that found the solution. Table 9.2 lists instance names where the best values found by RECIPE and listed in MINLPLib are identical.

9.4.1.1 Optimality

RECIPE found feasible solutions for 163 instances out of 245 (66%). Relative to this reduced instance set, it found the best known solution for 121 instances (74%), gave evidence of the unboundedness of three instances (1%), and improved the best known objective value for 12 instances (7%). In the other cases it found a local optimum that was worse than the best known solution.

Improved solutions were found for the following instances:

Table 9.1 Computational results on MINLPLib. Values denoted by * mark instances with unbounded values in the optimal solution.

instance	RECIPE		known solution	
	f^*	*CPU*	$\bar{f}$	Solver
`csched2a`	**-165398.701331**	75.957500	-160037.701300	BonMin
`eniplac`	-131926.917119	113.761000	**-132117.083000**	SBB+CONOPT
`ex1233`	160448.638212	3.426480	**155010.671300**	SBB+CONOPT
`ex1243`	118489.866394	5.329190	**83402.506400**	BARON
`ex1244`	211313.560000	7.548850	**82042.905200**	SBB+CONOPT
`ex1265a`	15.100000	9.644530	**10.300000**	BARON
`ex3`	**-53.990210**	1.813720	68.009700	SBB+CONOPT
`ex3pb`	**-53.990210**	1.790730	68.009700	SBB+CONOPT
`fo7_2`	22.833307	23.710400	**17.748900**	AlphaECP
`fo7`	24.311289	25.423100	**20.729700**	AlphaECP
`fo9`	38.500000	46.296000	**23.426300**	AlphaECP
`fuel`	17175.000000	1.161820	**8566.119000**	SBB+CONOPT
`gear4`	1.968201	9.524550	**1.643400**	SBB+CONOPT2
`lop97ic`	4814.451760	3047.110000	**4284.590500**	-
`lop97icx`	**4222.273030**	1291.510000	4326.147700	SBB+CONOPT
`m7`	220.530055	17.275400	**106.756900**	AlphaECP
`minlphix`	**209.149396***	4.849260	316.692700	SBB+`snopt`
`nuclear14b`	**-1.119531**	7479.710000	-1.113500	SBB+CONOPT
`nuclear24b`	**-1.119531**	7483.530000	-1.113500	SBB+CONOPT
`nuclear25`	**-1.120175**	1329.530000	-1.118600	SBB+CONOPT
`nuclearva`	-1.008822	167.102000	**-1.012500**	SBB+CONOPT2+`snopt`
`nuclearvb`	-1.028122	155.513000	**-1.030400**	SBB+CONOPT2+`snopt`
`nuclearvc`	**-1.000754**	176.075000	-0.998300	SBB+CONOPT2+`snopt`
`nuclearvd`	**-1.033279**	202.416000	-1.028500	SBB+CONOPT2+`snopt`
`nuclearve`	-1.031364	193.764000	**-1.035100**	SBB+CONOPT2+`snopt`
`nuclearvf`	**-1.020808**	200.154000	-1.017700	SBB+CONOPT2+`snopt`
`nvs02`	**5.964189**	1.925710	5.984600	SBB+CONOPT3
`nvs05`	28.433982	4.215360	**5.470900**	SBB+CONOPT3
`nvs14`	**-40358.114150**	2.070690	-40153.723700	SBB+CONOPT3
`nvs22`	28.947660	4.849260	**6.058200**	SBB+CONOPT3
`o7_2`	125.907318	23.262500	**116.945900**	AlphaECP
`o7`	160.218617	24.267300	**131.649300**	AlphaECP
`oil`	-0.006926	389.266000	**-0.932500**	SBB+CONOPT(fail)
`product`	-1971.757941	2952.160000	**-2142.948100**	DICOPT+CONOPT3/CPLEX
`st_e13`	2.236072	0.548916	**2.000000**	BARON
`st_e40`	52.970520	0.930858	**30.414200**	BARON
`stockcycle`	120637.913333	17403.200000	**119948.688300**	SBB+CONOPT
`super3t`	-0.674621	38185.500000	**-0.685965**	SBB+CONOPT
`synheat`	186347.748738	3.534460	**154997.334900**	SBB+CONOPT
`tln7`	19.300000	1000.640000	**15.000000**	BARON
`risk2b`	$-\infty^*$	45.559100	-55.876100	SBB+CONOPT3
`risk2bpb`	$-\infty^*$	48.057700	-55.876100	SBB+CONOPT3

`csched2a`:	$f^* = -165398.701331$	(best known solution: -160037.701300)
`ex3`:	$f^* = -53.990210$	(best known solution: 68.009700)
`ex3pb`:	$f^* = -53.990210$	(best known solution: 68.009700)
`lop97icx`:	$f^* = 4222.273030$	(best known solution: 4326.147700)
`minlphix`:	$f^* = 209.149396$	(best known solution: 316.692700)
`nuclear14b`:	$f^* = -1.119531$	(best known solution: -1.113500)
`nuclear24b`:	$f^* = -1.119531$	(best known solution: -1.113500)
`nuclear25`:	$f^* = -1.120175$	(best known solution: -1.118600)
`nuclearvc`:	$f^* = -1.000754$	(best known solution: -0.998300)
`nuclearvd`:	$f^* = -1.033279$	(best known solution: -1.028500)
`nuclearvf`:	$f^* = -1.020808$	(best known solution: -1.017700)
`nvs02`:	$f^* = 5.964189$	(best known solution: 5.984600)
`nvs14`:	$f^* = -40358.114150$	(best known solution: -40153.723700)
`risk2b`:	$f^* = -\infty$	(best known solution: -55.876100)
`risk2bpb`:	$f^* = -\infty$	(best known solution: -55.876100).

Table 9.2 Instances for which RECIPE's optima are the same as those reported in MINLPLib.

alan	ex1224	gbd	nvs06	parallel	st_e32	tln2
batchdes	ex1225	gear2	nvs07	prob02	st_e36	tln4
batch	ex1226	gear3	nvs08	prob03	st_e38	tln5
cecil_13	ex1252a	gear	nvs09	prob10	st_miqp1	tln6
contvar	ex1252	gkocis	nvs10	procsel	st_miqp2	tloss
csched1a	ex1263a	hmittelman	nvs11	pump	st_miqp3	tls2
csched1	ex1263	johnall	nvs12	qap	st_miqp4	util
csched2	ex1264a	m3	nvs13	ravem	st_miqp5	
du-opt5	ex1264	m6	nvs15	ravempb	st_test1	
du-opt	ex1265	meanvarx	nvs16	sep1	st_test2	
enpro48	ex1266a	nuclear14a	nvs17	space25a	st_test3	
enpro48pb	ex1266	nuclear14	nvs18	space25	st_test4	
enpro56	ex4	nuclear24a	nvs19	spectra2	st_test6	
enpro56pb	fac1	nuclear24	nvs20	spring	st_test8	
ex1221	fac2	nuclear25a	nvs21	st_e14	st_testgr1	
ex1222	fac3	nuclear25b	nvs23	st_e15	st_testph4	
ex1223a	feedtray2	nvs01	nvs24	st_e27	synthes1	
ex1223b	feedtray	nvs04	oaer	st_e29	synthes2	
ex1223	gastrans	nvs03	oil2	st_e31	synthes3	

All new best solutions were double-checked for constraint, bounds and integrality feasibility besides the verifications provided by the local solvers, and were all found to be integral feasible; 11 out of 12 were constraint/bound feasible to within a 10^{-5} absolute tolerance, and 1 (`csched2a`) to within 10^{-2}. The 3 instances marked by * in Table 9.1 (`minlphix`, `risk2b`, `risk2bpb`) gave solutions x^* with some of the components at values in excess of 10^{18}. Since `minlphix` minimizes a fractional objective function and there are no upper bounds on several of the problem variables, the optimum is attained when the variables appearing in the denominators tend towards $+\infty$. We solved `risk2b` and `risk2bpb` several times, setting increasing upper bounds to the unbounded variables: this yielded decreasing values of the objective function, suggesting that these instances are really unbounded (hence the $-\infty$ in Table 9.1).

On 82 instances out of 245 listed in Table 9.3, RECIPE failed to find any local optimum within the allotted time limit. Most of these failures are due to the difficulty of the continuous relaxation of the MINLPs: there are several instances where the SQP method (`snopt`) does not manage to find a feasible starting point, and in these cases the convex MINLP solver (`minlp_bb`) also fails. On a smaller number of instances, `minlp_bb` is not able to find integral feasible solutions even though constraint feasible solutions are provided by `snopt`.

9.4.1.2 Reliability

One interesting feature of RECIPE is its reliability: in its default configuration it managed to find solutions with better or equal quality than those

Table 9.3 Instances unsolved by RECIPE.

4stufen	elf	fo9_ar2_1	no7_ar2_1	nuclear49	st_e35	tltr
beuster	fo7_ar2_1	fo9_ar25_1	no7_ar25_1	o7_ar2_1	st_test5	uselinear
deb10	fo7_ar25_1	fo9_ar3_1	no7_ar3_1	o7_ar25_1	st_testgr3	var_con10
deb6	fo7_ar3_1	fo9_ar4_1	no7_ar4_1	o7_ar3_1	super1	var_con5
deb7	fo7_ar4_1	fo9_ar5_1	no7_ar5_1	o7_ar4_1	super2	waste
deb8	fo7_ar5_1	gasnet	nous1	o7_ar5_1	super3	water4
deb9	fo8_ar2_1	m7_ar2_1	nous2	o8_ar4_1	tln12	waterx
detf1	fo8_ar25_1	m7_ar25_1	nuclear104	o9_ar4_1	tls12	waterz
eg_all_s	fo8_ar3_1	m7_ar3_1	nuclear10a	ortez	tls4	windfac
eg_disc2_s	fo8_ar4_1	m7_ar4_1	nuclear10b	qapw	tls5	waterx
eg_disc_s	fo8_ar5_1	m7_ar5_1	nuclear49a	saa_2	tls6	
eg_int_s	fo8	mbtd	nuclear49b	space960	tls7	

reported in the MINLPLib on 136 instances over 245 (55%) and at least a feasible point in a further 11% of the cases. On the same set of test instances, the closest competitor is SBB+CONOPT, which matches or surpasses the best solutions in MINLPLib in 37% of the cases, followed by BARON with 15% and by AlphaECP with 14% (these percentages were compiled by looking at `http://www.gamsworld.org/minlp/minlplib/points.htm` in June 2008).

9.4.1.3 Speed

The total time taken for solving the whole MINLPLib (including the unsolved instances, where the VNS algorithm terminates after exploring the neighbourhoods up to $k_{\max}$ or when reaching the 10 hours time limit, whichever comes first) is roughly 4 days and 19 hours of user CPU time. RECIPE's speed is very competitive with that of sBB approaches: tests conducted using the $oo\mathcal{OPS}$ solver [28, 29, 34] as well as BARON on some complex MINLPs showed that sBB methods may take a long time to converge. Naturally, the trade-off for this speed is the lack of an optimality guarantee.

9.5 Conclusion

This chapter describes a heuristic approach to solving nonconvex MINLPs based on the mathematical programming formulation. Our approach, called RECIPE, combines several existing exact, approximate and heuristic techniques in a smart way, resulting in a method that can successfully solve many difficult MINLPs without hand-tuned parameter configuration. Such a reliable solver would be particularly useful in industrial applications where the optimum quality is of relative importance and the optimization layer is hidden from user intervention and is therefore "just supposed to work".

Acknowledgements We are very grateful to Prof. Tapio Westerlund for carefully checking all the computational results and informing us of some misprints on the MINLPLib website.

References

1. K. Abhishek, S. Leyffer, and J. Linderoth. FilMINT: An outer-approximation based solver for nonlinear mixed-integer programs. Technical Report ANL/MCS-P1374-0906, Argonne National Laboratory, 2007.
2. C.S. Adjiman, I.P. Androulakis, and C.A. Floudas. Global optimization of MINLP problems in process synthesis and design. *Computers & Chemical Engineering*, 21:S445–S450, 1997.
3. M. Aouchiche, J.M. Bonnefoy, A. Fidahoussen, G. Caporossi, P. Hansen, L. Hiesse, J. Lacheré, and A. Monhait. VNS for extremal graphs 14: The AGX 2 system. In Liberti and Maculan [33], pages 281–308.
4. 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. Technical Report RC23771, IBM Corporation, 2005.
5. P. Bonami, G. Cornuéjols, A. Lodi, and F. Margot. A feasibility pump for mixed integer nonlinear programs. Technical Report RC23862 (W0602-029), IBM Corporation, 2006.
6. P. Bonami and J. Lee. `BONMIN` user's manual. Technical report, IBM Corporation, June 2007.
7. J. Brimberg and N. Mladenović. A variable neighbourhood algorithm for solving the continuous location-allocation problem. *Studies in Location Analysis*, 10:1–12, 1996.
8. A. Brook, D. Kendrick, and A. Meeraus. GAMS, a user's guide. *ACM SIGNUM Newsletter*, 23(3-4):10–11, 1988.
9. 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), 2003.
10. ARKI Consulting and Development. *SBB Release Notes*, 2002.
11. M. Dražic, V. Kovačević-Vujčić, M. Čangalović, and N. Mladenović. Glob — a new VNS-based software for global optimization. In Liberti and Maculan [33], pages 135–154.
12. M. Dražić, C. Lavor, N. Maculan, and N. Mladenović. A continuous variable neighbourhood search heuristic for finding the tridimensional structure of a molecule. *European Journal of Operational Research*, 185:1265–1273, 2008.
13. A. Drud. CONOPT: A GRG code for large sparse dynamic nonlinear optimization problems. *Mathematical Programming*, 31:153–191, 1985.
14. M. Duran and I. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36:307–339, 1986.
15. M. Fischetti and A. Lodi. Local branching. *Mathematical Programming*, 98:23–37, 2005.
16. R. Fletcher and S. Leyffer. Solving mixed integer nonlinear programs by outer approximation. *Mathematical Programming*, 66:327–349, 1994.
17. R. Fletcher and S. Leyffer. Numerical experience with lower bounds for MIQP branch-and-bound. *SIAM Journal of Optimization*, 8(2):604–616, 1998.
18. R. Fletcher and S. Leyffer. User manual for `filter`. Technical report, University of Dundee, UK, March 1999.
19. R. Fourer and D. Gay. *The AMPL Book*. Duxbury Press, Pacific Grove, 2002.
20. P. Gill, W. Murray, and M.A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal of Optimization*, 12(4):979–1006, 2002.

21. P.E. Gill. *User's guide for SNOPT version 7.* Systems Optimization Laboratory, Stanford University, California, 2006.
22. P. Hansen and N. Mladenović. Variable neighbourhood search: Principles and applications. *European Journal of Operational Research*, 130:449–467, 2001.
23. P. Hansen and N. Mladenović. Variable neighbourhood search. In P. Pardalos and M.G.C. Resende, editors, *Handbook of Applied Optimization.* Oxford University Press, Oxford, 2002.
24. P. Hansen and N. Mladenović. Variable neighbourhood search. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics.* Kluwer, Dordrecht, 2003.
25. P. Hansen, N. Mladenović, and D. Urošević. Variable neighbourhood search and local branching. *Computers & Operations Research*, 33(10):3034–3045, 2006.
26. C. Lavor, L. Liberti, and N. Maculan. Computational experience with the molecular distance geometry problem. In J. Pintér, editor, *Global Optimization: Scientific and Engineering Case Studies*, pages 213–225. Springer, Berlin, 2006.
27. S. Leyffer. User manual for `minlp_bb`. Technical report, University of Dundee, UK, March 1999.
28. L. Liberti. *Reformulation and Convex Relaxation Techniques for Global Optimization.* PhD thesis, Imperial College London, UK, March 2004.
29. L. Liberti. Writing global optimization software. In Liberti and Maculan [33], pages 211–262.
30. L. Liberti and M. Dražic. Variable neighbourhood search for the global optimization of constrained NLPs. In *Proceedings of GO Workshop, Almeria, Spain*, 2005.
31. L. Liberti, C. Lavor, and N. Maculan. Double VNS for the molecular distance geometry problem. In P. Hansen, N. Mladenović, J.A. Moreno Pérez, editors, *Proceeding of the 18th Mini Euro Conference on Variable Neighbourhood Search*, Tenerife, Spain, 2005.
32. L. Liberti, C. Lavor, N. Maculan, and F. Marinelli. Double variable neighbourhood search with smoothing for the molecular distance geometry problem. *Journal of Global Optimization*, accepted for publication.
33. L. Liberti and N. Maculan, editors. *Global Optimization: from Theory to Implementation.* Springer, Berlin, 2006.
34. L. Liberti, P. Tsiakis, B. Keeping, and C.C. Pantelides. *ooOPS*. Centre for Process Systems Engineering, Chemical Engineering Department, Imperial College, London, UK, 2001.
35. A. Lodi. Personal communication, 2007.
36. N. Mladenović, J. Petrović, V. Kovačević-Vujčić, and M. Čangalović. Solving a spread-spectrum radar polyphase code design problem by tabu search and variable neighbourhood search. *European Journal of Operational Research*, 151:389–399, 2003.
37. J. Puchinger and G.R. Raidl. Relaxation guided variable neighbourhood search. In *Proc. of Mini Euro Conference on Variable Neighbourhood Search, Tenerife, Spain*, 2005.
38. E.M.B. Smith and C.C. Pantelides. A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs. *Computers & Chemical Engineering*, 23:457–478, 1999.
39. M. Tawarmalani and N.V. Sahinidis. Global optimization of mixed integer nonlinear programs: A theoretical and computational study. *Mathematical Programming*, 99:563–591, 2004.
40. T. Westerlund. Some transformation techniques in global optimization. In Liberti and Maculan [33], pages 45–74.
41. T. Westerlund and R. Pörn. Solving pseudo-convex mixed integer optimization problems by cutting plane techniques. *Optimization and Engineering*, 3:235–280, 2002.
42. T. Westerlund, H. Skrifvars, I. Harjunkoski, and R. Pörn. An extended cutting plane method for a class of non-convex MINLP problems. *Computers & Chemical Engineering*, 22(3):357–365, 1998.