**International Refereed Journal of Engineering and Science (IRJES)**
*ISSN (Online) 2319-183X, (Print) 2319-1821*
*Volume 3, Issue 2 (February 2014), PP.16-24*

# A Neighbourhood Search Approach For Solving Large Scale Mixed-Integer Non Linear Programming Problems

Erwin, Setiawan Tanadi, Meidiana Tanadi, Herman Mawengkang
*Graduate School of Mathematics, University of Sumatera Utara*
*FMIPA USU, Medan Indonesia - 20155*

**Abstract:-** The mixed-integer nonlinear programming problems addressed in this paper are large-scale, highly combinatorial and highly nonlinear problems. They have a structure characterized bya subset of variables restricted to assume discrete values, which are linear and separable from the continuous variables. The strategy of releasing nonbasic variables from their bounds, combined with the "active constraint" method, has been developed. This strategy is used to force the appropriate non-integer basic variables to move to their neighbourhood integer points. Succesful implementation of these algorithms was achieved on various test problems.

**Keywords:-** Nonlinear programming, active constraints, direct search, integer programming, neighbourhood search

## I. INTRODUCTION

Mixed Integer Nonlinear Programming (MINLP) refers to mathematical programming with continuous and discrete variables and nonlinearities in the objective function and constraints. The special class of Mixed-Integer nonlinear programming problem which is addressed in this paper is to assume discrete values, which are linear and separable from the continuous variables.

The basic form of an MINLP problem when represented mathematically can be written as follows:

$$\min Z = f(x, y) \tag{1}$$

$$s.t.\ g(x, y) \leq 0\ j \in J \tag{2}$$

$$x \in X, y \in Y \tag{3}$$

where *f(·), g(·)* are *convex, differentiable* functions, *J* is the index set of inequalities, and *x* and *y* are the continuous and discrete variables, respectively. The set *X* is assumed to be a convex compact set, e.g. $X = \{x \mid x \in \boldsymbol{R}^n, Dx < d, x^L < x < x^U\}$; the discrete set *Y* corresponds to a polyhedral set of integer points, $Y = \{y \mid y \in \boldsymbol{Z}^m, Ay < a\}$ , and in most real world problems is restricted to 0-1 values, $y \in \{0,1\}^m$ .

There are various applications for the MINLP model, including the process industry and the financial engineering, management science and operations research sectors. It includes problems in process flow sheets, portfolio selection, batch processing in chemical engineering (consisting of mixing, reaction, and centrifuge separation), and optimal design of gas or water transmission networks. Other areas of interest include the automobile, aircraft, and VLSI manufacturing areas. An impressive collection of MINLP applications can be found in [1] and [2]. The needs in such diverse areas have motivated research and development in MINLP solver technology, particularly in algorithms for handling large-scale, highly combinatorial and highly nonlinear problems.

Methods for solving MINLPs include innovative approaches and related techniques taken and extended from MIP, such as, Outer Approximation (OA) methods [2,3,4], Branch-and-Bound (B&B) [5,6,7], Extended Cutting Plane methods [8], and Generalized Bender's Decomposition (GBD) [9] for solving MINLPs have been discussed in the literature since the early 1980's. These approaches generally rely on the successive solutions of closely related NLP problems. For example, B&B starts out forming a pure continuous NLP problem by dropping the integrality requirements of the discrete variables (often called the relaxed MINLP or RMINLP).

Moreover, each node of the emerging B&B tree represents a solution of the RMINLP with adjusted bounds on the discrete variables.

Heuristic approaches to solving MINLPs include Variable Neighbourhood Search [10], automatically tuned variable fixing strategies [11], Local Branching [12], feasible neighbourhood search [12], Feasibility Pump [13,14,15], heuristics based on Iterative Rounding [16]. Recently [17] propose a MINLP heuristic called the Relaxed-Exact-Continuous-Integer Problem Exploration (RECIPE) algorithm. The algorithm puts together a global search phase based on Variable Neighbourhood Search [10] and a local search phase based on a MINLP heuristic. In heuristic approaches, however, one of the main algorithmic difficulties connected to MINLPs is to find a feasible solution. From the worst-case complexity point of view, finding a feasible MINLP solution is as hard as finding a feasible Nonlinear Programming solution, which is NP-hard [12].

Due to the fact that the functions in MINLPs are not smooth, therefore in this paper we use a direct search method, known as unconstrained optimization techniques that do not explicitly use derivatives. More information regarding to direct search method in optimization can be found in [8].

In this paper we address a strategy of releasing nonbasic variables from their bounds, combined with the “active constrained” method and the notion of superbasic for efficiently tackling a particular class of MINLP problems.

The rest of this paper is organized as follows. In Section 2 we give a brief notion of neighbourhood search. The basic approach of the proposed method is presented in Section 3. How to derive the proposed method is given in Section 4. The algorithm is presented in Section 5. Section 6 addresses a computational experience. The conclusions can be found in Section 7.

## II. NEIGHBOURHOOD SEARCH

It should be noted that, generally, in integer programming the reduced gradient vector, which is normally used to detect an optimality condition, is not available, even though the problems are convex. Thus we need to impose a certain condition for the local testing search procedure in order to assure that we have obtained the “best” suboptimal integer feasible solution.

Scarf [18] has proposed a quantity test to replace the pricing test for optimality in the integer programming problem. The test is conducted by a search through the neighbours of a proposed feasible point to see whether a nearby point is also feasible and yields an improvement to the objective function.

Let $[\beta]_k$ be an integer point belongs to a finite set of neighbourhood $N([\beta]_k)$. We define a neighbourhood system associated with $[\beta]_k$, that is, if such an integer point satisfies the following two requirements

1. If $[\beta]_j \in N([\beta]_k)$ then $[\beta]_k \in [\beta]_j), j \neq k.$
2. $N([\beta]_k) = [\beta]_k + N(0)$

With respect to the neighbourhood system mentioned above, the proposed integerizing strategy can be described as follows.

Given a non-integer component, $x_k$, of an optimal vector , $x_B$. The adjacent points of $x_k$, being considered are $[x_k]$ and $[x_k] + 1$. If one of these points satisfies the constraints and yields a minimum deterioration of the optimal objective value we move to another component, if not we have integer-feasible solution.

Let $[x_k]$ be the integer feasible point which satisfies the above conditions. We could then say if $[x_k] + 1 \in N([x_k])$ implies that the point $[x_k] + 1$ is either infeasible or yields an inferior value to the objective function obtained with respect to $[x_k]$. In this case $[x_k]$ is said to be an “optimal” integer feasible solution to the integer programming problem. Obviously, in our case, a neigbourhood search is conducted through proposed feasible points such that the integer feasible solution would be at the least distance from the optimal continuous solution.

## III. THE BASIC APPROACH

Before we proceed to the case of MINLP problems, it is worthwhile to discuss the basic strategy of process for linear case, i.e., Mixed Integer Linear Programming (MILP) problems.
Consider a MILP problem with the following form

$$Minimize\ P = c^T x \tag{4}$$

$$Subject\ to\ Ax \le b \tag{5}$$

$$x \ge 0 \tag{6}$$

$$x_j\ integer\ for\ some\ j \in J \tag{7}$$

A component of the optimal basic feasible vector $(x_B)_k$, to MILP solved as continuous can be written as

$$(x_B)_k = \beta_k - \alpha_{k1}(x_N)_1 - \cdots - \alpha_{kj^*}(x_N)_j - \cdots - \qquad \alpha_{k,n-m}(x_N)_{n-m} \tag{8}$$

Note that, this expression can be found in the final tableau of Simplex procedure. If $(x_B)_k$ is an integer variable and we assume that $\beta_k$ is not an integer, the partitioning of $\beta_k$ into the integer and fractional components is that given

$$\beta_k = [\beta_k] + f_k, 0 \le f_k \le 1 \tag{9}$$

suppose we wish to increase $(x_B)_k$ to its nearest integer, $([\beta] + 1)$. Based on the idea of suboptimal solutions we may elevate a particular nonbasic variable, say $(x_N)_{j^*}$, above its bound of zero, provided $\alpha_{kj^*}$, as one of the element of the vector $\alpha_{j^*}$, is negative. Let $\Delta_{j^*}$ be amount of movement of the non variable $(x_N)_{j^*}$, such that the numerical value of scalar $(x_B)_k$ is integer. Referring to Eqn. (8), $\Delta_{j^*}$ can then be expressed as

$$\Delta_{j^*} = \frac{1-f_k}{-\alpha_{kj^*}} \tag{10}$$

while the remaining nonbasic stay at zero. It can be
seen that after substituting (9) into (10) for $(x_N)_{j^*}$ and taking into account the partitioning of $\beta_k$ given in (10), we obtain

$$(x_B)_k = [\beta] + 1 \tag{11}$$

Thus, $(x_B)_k$ is now an integer.

It is now clear that a nonbasic variable plays an important role to integerize the corresponding basic variable. Therefore, the following result is necessary in order to confirm that must be a non-integer variable to work with in integerizing process.

**Theorem 1.** *Suppose the MILP problem (4)-(7) has an optimal solution, then some of the nonbasic variables.* $(x_N)_j, j = 1, \dots, n$*, must be non-integer variables.*

**Proof:**

Solving problem as a continuous of slack variables (which are non-integer, except in the case of equality constraint). If we assume that the vector of basic variables $x_B$ consists of all the slack variables then all integer variables would be in the nonbasic vector $x_N$ and therefore integer valued.

## IV. DERIVATION OF THE METHOD

It is clear that the other components, $(x_B)_{i\neq k}$, of vector $x_B$ will also be affected as the numerical value of the scalar $(x_N)_{j^*}$ increases to $\Delta_{j^*}$. Consequently, if some element of vector $\alpha_{j^*}$, i.e., $\alpha_{j^*}$ for $i \neq k$, are positive, then the corresponding element of $x_B$ will decrease, and eventually may pass through zero. However, any component of vector *x* must not go below zero due to the non-negativity restriction. Therefore, a formula, called the minimum ratio test is needed in order to see what is the maximum movement of the nonbasic $(x_N)_{j^*}$ such that all components of *x* remain feasible. This ratio test would include two cases.

1. A basic variable $(x_B)_{j\neq k}$ decreases to zero (lower bound) first.
2. The basic variable, $(x_B)_k$ increases to an integer.

Specifically, corresponding to each of these two cases above, one would compute

$$\theta_1 = \min_{i\neq k|\alpha_{j^*}>0}\left\{\frac{\beta_i}{\alpha_{j^*}}\right\} \tag{12}$$

$$\theta_2 = \Delta_{j^*} \tag{13}$$

How far one can release the nonbasic $(x_N)_{j^*}$ from its bound of zero, such that vector $x$ remains feasible, will depend on the ratio test $\theta^*$ given below

$$\theta^* = \min(\theta_1, \theta_2) \tag{14}$$

Obviously, if $\theta^* = \theta_1$, one of the basic variable $(x_B)_{i\neq k}$ will hit the lower bound before $(x_B)_k$ becomes integer. If $\theta^* = \theta_2$, the numerical value of the basic variable $(x_B)_k$ will be integer and feasibility is still maintained. Analogously, we would be able to reduce the numerical value of the basic variable $(x_B)_k$ to its closest integer $[\beta_k]$. In this case the amount of movement of a particular nonbasic variable, $(x_N)_{j'}$, corresponding to any positive element of vector $\alpha_{j'}$, is given by

$$\Delta_{j'} = \frac{f_k}{\alpha_{kj'}} \tag{15}$$

In order to maintain the feasibility, the ratio test $\theta^*$ is still needed.

Consider the movement of a particular nonbasic variable, $\Delta$, as expressed in Eqns.(10) and (15).The only factor that one needs to calculate is the corresponding element of vector $\alpha$. A vector $\alpha_j$ can be expressed as

$$\alpha_j = B^{-1}a_j, j = 1 \ldots, n-m \tag{16}$$

Therefore, in order to get a particular element of vector $\alpha_j$ we should be able to distinguish the corresponding column of matrix $[B]^{-1}$. Suppose we need the value of element $\alpha_{kj^*}$, letting $v_k^T$ be the $k$-th column vector of $[B]^{-1}$, we then have

$$v_k^T = e_k^T B^{-1} \tag{17}$$

Subsequently, the numerical value of $\alpha_{kj*}$ can be obtained from

$$\alpha_{kj^*} = v_k^T a_{j^*} \tag{18}$$

in Linear Programming (LP) terminology the operation conducted in Eqns. (17) and (18) is called the pricing operation. The vector of reduced costs $d_i$ is used to measure the deterioration of the objective function value caused by releasing a nonbasic variable from its bound. Consequently, in deciding which nonbasic should be released in the integerizing process, the vector $d_i$ must be taken into account, such that deterioration is minimized. Recall that the minimum continuous solution provides a lower bound to any integer-feasible solution. Nevertheless, the amount of movement of particular nonbasic variable as given in Eqns. (10) or (15), depends in some way on the corresponding element of vector $\alpha_j$. Therefore it can be observed that the deterioration of the objective function value due to releasing a nonbasic variable $(x_N)_{j^*}$ so as to integerize a basic variable $(x_B)_k$ may be measured by the ratio

$$\left|\frac{d_k}{\alpha_{kj^*}}\right| \tag{19}$$

where $|a|$ means the absolute value of scalar a.

In order to minimize the detoriation of the optimal continuous solution we then use the following strategy for deciding which nonbasic variable may be increased from its bound of zero, that is,

$$\min_j \left\{\left|\frac{d_k}{\alpha_{kj^*}}\right|\right\},\ j = 1, \dots, n-m \tag{20}$$

From the "active constraint" strategy and the partitioning of the constraints corresponding to basic $(B)$, superbasic $(S)$ and nonbasic $(N)$ variables, we can write

$$\begin{bmatrix} B & S & N \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} x_b \\ x_N \\ x_S \end{bmatrix} = \begin{bmatrix} b \\ b_N \end{bmatrix} \tag{21}$$

or

$$Bx_b + Sx_N + Nx_S = b \tag{22}$$

$$x_N = b_N \tag{23}$$

The basis matrix $B$ is assumed to be square and nonsingular, we get

$$x_B = \beta - Wx_S - \alpha x_N \tag{24}$$

Where

$$\beta = B^{-1}b \tag{25}$$

$$W = B^{-1}S \tag{26}$$

$$\alpha = B^{-1}N \tag{27}$$

Expression (23) indicates that the nonbasic variables are being held equal to their bound. It is evident through the "nearly" basic expression of Eqn. (24), the integerizing strategy discussed in the previous section, designed for MILP problem can be implemented. Particularly, we would be able to release a nonbasic variable from its bound, Eqn.(23) and exchange it with a corresponding basic variable in the integerizing process, although the solution would be degenerate. Furthermore, the Theorem (1) above can also be extended for MINLP problem.

**Theorem 2.** *Suppose the MINLP problem has a bounded optimal continuous solution, then we can always get a non-integer $y_i$ in the optimum basic variable vector.*

***Proof.***

1. If these variables are nonbasic, they will be at their bound. Therefore they have integer value.
2. If a $y_j$ is superbasic, it is possible to make $y_j$ basic and bring in a nonbasic at its bound to replace it in the superbasic.

However, the ratio test expressed in (14) cannot be used as a tool to guarantee that the integer solution found still remains in the feasible region.

## 4.1 Pivoting

Currently, we are in a position where particular basic variable, $(x_B)_k$ is being integerized, thereby a corresponding nonbasic variable, $(c_N)_{j^*}$, is being released from its bound of zero. Suppose the maximum movement of $(x_N)_{j^*}$ satisfies

$$\theta^* = \Delta_{j^*} \tag{28}$$

such that $(x_B)_k$ is integer valued. To exploit the manner of changing the basis in linear programming, we would be able to move $(x_N)_{j^*}$ into $B$ (to replace $(x_B)_k$) and integer-valued $(x_B)_k$ into *S* in order to maintain the integer solution. We now have a degenerate solution since a basic variable is at its bound. The integerizing process continues with a new set of $[B, S]$. In this case, eventually we may end up with all of the integer variables being superbasic.

**Theorem 3**. *A suboptimal solution exists to the MILP and MINLP problem in which all of the integer variables are superbasic.*

***Proof.***

1. If all of the integer variables are in *N*, then they will be at bound.
2. If an integer variable is basic it is possible to either
   - interchange it with a superbasic continuous variable, or
   - make this integer variable superbasic and bring in a nonbasic at its bound to replace it in the basis which gives a degenerate solution.

The other case which can happen is that a different basic variabels $(x_B)_{i\neq k}$ may hit its bound before $(x_B)_k$ becomes integer. Or in other words, we are in a situation where

$$\theta^* = \Delta_1 \tag{29}$$

In this case we move the basic variable $(x_B)_j$ into $N$ and its position in the basic variable vector would be replaced by nonbasic $(x_B)_{j^*}$. Note that $(x_B)_k$ is still a non-integer basic variable with a new value.

## V. THE ALGORITHM

After solving the relaxed problem, the procedure for searching a suboptimal but integer-feasible solution from an optimal continuous solution can be described as follows.

Let

$$x = [x] + f, \quad 0 \leq f \leq 1 \tag{30}$$

be the (continuous) solution of the relaxed problem, $[x]$ is the integer component of non-integer variable $x$ and $f$ is the fractional component.

**Stage 1.**

**Step 1.** Get row $i^*$ the smallest integer infeasibility, such that $\delta_{i^*} = \min\{f_i, 1 - f_i\}$

**Step 2.** Do a pricing operation

$v_{i^*}^T = e_{i^*}^T B^{-1}$

**Step 3.** Calculate $\sigma_{ij} = v_{i^*}^T a_j$

With $j$ corresponds to

$$\min_j \left\{ \left| \frac{d_j}{\sigma_{ij}} \right| \right\} \tag{31}$$

Calculate the maximum movement of nonbasic j at lower bound and upper bound

Otherwise go to next non-integer nonbasic or superbasic $j$ (if available). Eventually the column $j^*$ is to be increased form LB or decreased from UB. If none go to next $i^*$.

**Step 4.**

Solve $B\alpha_{j^*} = \alpha_{j^*}$ for $\alpha_{j^*}$

**Step 5.** Do ratio test for the basic variables in order to stay feasible due to the releasing of nonbasic $j$ from its bounds.

**Step 6.** Exchange basis

**Step 7.** If row $i^* = \{\emptyset\}$ go to Stage 2, otherwise

Repeat from step 1.

**Stage 2.** Do integer lines search to improve the integer feasible solution

## VI. COMPUTATIONAL EXPERIENCE 1: A PROCESS SYSTEM SYNTHESIS PROBLEM

### 6.1 Mathematical Statement the Problem.

This synthesis problem is the one of simultaneously determining the optimal structural and operating parameters for a process so as to satisfy a given design specification. The decision variables are defined as follows.

$y$ is a binary variable which is associated with each process unit (piece of equipment) to denote its potential existence in the final optimal configuration, and

$x$ are the continuous variables which represent process parameters such as flow rates of materials.
Generally, the objective is to minimize the annual costs, including both investment and operation costs.

Minimize

$$F = 5y_1 + 8y_2 + 6y_3 + 10y_4 + 6y_5 + 7y_6 + 4y_7 + 5y_8 - 10x_3 - 15x_5 + 15x_{10} + 80x_{17} + 25x_{19} + 35x_{21} - 40x_9 + 15x_{14} - 35x_{25} + \exp(x_3) + \exp\left(\frac{x_5}{1.2}\right) - 65\ln(x_{10} + x_{17} + 1) - 90\ln(x_{19}1) - 80\ln(x_{21} + 1) + 120 \quad (32)$$

**Subject to**

$$-1.5\ln(x_{19} + 1) - \ln(x_{21} + 1) - x_{14} \le 0 \quad (33)$$
$$-\ln(x_{10} + x_{17} + 1) \le 0 \quad (34)$$
$$-x_3 - x_5 + x_{10} + 2x_{17} + 0.8x_{19} + 0.8x_{21} - 0.5x_9 - x_{14} - 2x_{25} \le 0 \quad (35)$$
$$-x_3 - x_5 + 2x_{17} + 0.8x_{19} + 0.8x_{21} - 2x_9 - x_{14} - 2x_{25} \le 0 \quad (36)$$
$$-2x_{17} - 0.8x_{09} - 0.8x_{21} + 2x_9 + x_{14} + 2x_{25} \le 0 \quad (37)$$
$$-0.8x_{19} - 0.8x_{21} + x_{14} \le 0 \quad (38)$$
$$-x_{17} + x_9 + x_{25} \le 0 \quad (39)$$
$$-0.4x_{19} - 0.4x_{21} + 1.5x_{14} \le 0 \quad (40)$$
$$0.16x_{19} + 0.16x_{21} - 1.2x_{14} \le 0 \quad (41)$$
$$x_{10} - 0.8x_{17} \le 0 \quad (42)$$
$$-x_{10} + 0.4x_{17} \le 0 \quad (43)$$
$$\exp(x_3) - 10y_1 \le 1 \quad (44)$$
$$\exp(x_3) - 10y_2 \le 1 \quad (45)$$
$$x_9 - 10y_3 \le 0 \quad (46)$$
$$0.8x_{19} + 0.8x_{21} - 10y_4 \le 0 \quad (47)$$
$$2x_{17} - 2x_9 - 2x_{25} - 10y_5 \le 0 \quad (48)$$

$x_{19} - 10y_6 \le 0$ (49)

$x_{21} - 10y_7 \le 0$ (50)

$x_{10} + x_{17} - 10y_8 \le 0$ (51)

$y_1 + y_2 = 1$ (52)

$y_4 + y_5 \le 1$ (53)

$-y_4 + y_6 + y_7 = 0$ (54)

$y_3 - y_8 \le 0$ (55)

$0 \le y_j \le 1$ and integer for $j = 1, \cdots, 8$ (56)

$l \le x \le u$ (57)

$x = x_j: (j = 3,5,10,17,19,21,9,14,25) \in R^9$ (58)

$l^T = (0,0,0,0,0,0,0,0,0), \quad u^T = (2,2,1,2,2,2,2,1,3)$ (59)

The above formulation contains 8 binary variables, 9 bounded continuous variables, 23 inequality constraints. Nonlinearities appear in the objective function and in four inequalities.

## 6.2 Discussion of the Results

We solved this problem using PC with processor Intel(R) Core (TM) i5-2300 CPU @ 280 GHZ and RAM 4.00GB. The continuous optimal solution was obtained by using NLP software. Only one binary variable is integer-valued (at its lower bound) in the continuous solution. A binary variable $y_7$ is in superbasic set with non integer value. We then moved this variable to its closest integer by using truncation strategy and kept it superbasic. We must check the feasibility of the corresponding basic variables due to this movement. We integerized the remaining non-integer binary variables by using our proposed integerizing strategy. Both the continuous and the integer results of the synthesis problem can be seen in **Table 1**.

**Table 1. The Results of the Synthesis Problem.**

| Variable | Activity in | Activity after |
|---|---|---|
| | Cont.Soln. | integ. Process |
| $x_3$ | 1.90293 | 0.0 |
| $x_5$ | 2.0 | 2.0 |
| $x_{10}$ | 0.52752 | 0.46784 |
| $x_{17}$ | 0.65940 | 0.58480 |
| $x_{19}$ | 2.0 | 2.0 |
| $x_{21}$ | 1.08333 | 0.0 |
| $x_9$ | 0.65940 | 0.0 |
| $x_4$ | 0.41111 | 0.26667 |
| $x_{25}$ | 0.0 | 0.58480 |
| $y_1$ | 0.57055 | 0.0 |
| $y_2$ | 0.42945 | 1.0 |
| $y_3$ | 0.06594 | 0.0 |
| $y_4$ | 0.30833 | 1.0 |
| $y_5$ | 0.0 | 0.0 |
| $y_6$ | 0.2 | 1.0 |
| $y_7$ | 0.10833 | 0.0 |
| $y_8$ | 0.11869 | 1.0 |
| Obj.value(F) | 15.08219 | 68.00974 |

**Our objective result is in agreement with the result obtained by [5].**

## VII. CONCLUSIONS

This paper has presented a direct search method for achieving integer-feasibility for a class of mixed-integer nonlinear programming problems in a relatively short time. The direct search approach used the strategy of releasing nonbasic variable from their bounds, combined with the "active constraint" method and the notion of superbasic. After solving a problem by ignoring the integrality requirements, this strategy is used to force the appropriate non-integer basic variables to move to their neighborhoods integer points.

A study of the criteria for choosing a nonbasic variable to work with in the integerizing strategy has also been made. The number of integerizing steps would be finite if the number of integer variables contained in the problem is finite. However, it should be noted that the computational time for the integerizing process does not necessarily depend on the number of integer variables, since many of the integer variables may have an integer value at the continuous optimal solution.

The new direct search method has been shown to be successful on a range of problems, while not always able to achieve global optimality. In a number of cases to obtain the suboptimal point is acceptable, since the exponential complexity of the combinatorial problems in general precludes branch-and-bound, except on small to medium problems.

Computational testing of the procedure presented this paper has demonstrated that it is a viable approach for large problems.

## REFERENCES

[1]. I.E. Grossmann and N.V. Sahinidis, Special Issue on Mixed-integer Programming and its Application to Engineering, Part I, Optim. Eng., 3 (4), Kluwer Academic Publishers, Netherlands, (eds) 2002.

[2]. I.E. Grossmann and N.V. Sahinidis), Special Issue on Mixed-integer Programming and its Application to Engineering, Part II, Optim. Eng., 4 (1), Kluwer Academic Publishers, Netherlands, (eds) 2002.

[3]. M. A. Duran and I. E Grossmann, An Outer-Approximation Algorithm For a class of Mixed-Integer Nonlinear Programs, Mathematical Programming 36, 1986, 307.

[4]. Roger Fletcher and Sven Leyffer, Solving Mixed Integer Nonlinear Programming by Outer Approximation, Mathematical Programming, Vol. 66, 1994, 327.

[5]. G. Nannicini, P. Belotti. Rounding-based heuristics for non-convex MINLPs. In: P. Bonami, L. Liberti, A. Miller, A. Sartenear (eds). Proceedings of the European Workshop on MINLP. CIRM, Marseille, France 2009.

[6]. O.K. Gupta and A. Ravindran, Branch and Bound Experiments in Convex Nonlinear Integer Programming, Manage Sci., 1985, 31 (12), 1533–1546.

[7]. P. Belotti, J. Lee, L. Liberti, F. Margot, A. Wachter, Branching and Bounds Tightening Techniques for non-convex MINLP, Optimization Methods and Software, 24(4) (2009) 597-634.

[8]. Tamara G. Kolda, Robert M. Lewis, Virginia Torezon. Optimization by Direct Search: New Perspective on Some Classical and Modern Methods. SIAM Review, 45 (1) (2003) 385-482.

[9]. A.M. Geoffrion, A Generalized Benders Decomposition, J. Op-tim. Theory Appl., 10 (4), 1972, 237–260.

[10]. L. Liberti, G. Nannicini, N. Mladenovic. A good recipe for solving MINLPs. In: V. Maniezzo, T. Stutzle, S.Voss(eds.) Metaheuristics: Hybridizing metaheuristics and Mathematical Programming. Annals of Information Systems, vol. 10 (Springer 2009) 231-245.

[11]. T. Berthold and A. Gleixner. Undercover-Primal MINLP Heuristic. In P. Bonami, L. Liberti, A. Miller and A. Sartenaer (eds.) Proceedings of the European Workshop on MINLP, Marseille, 2010, 103-113.

[12]. H. Mawengkang and B. A Murtagh, Solving Nonlinear Integer Programs with Large-Scale Optimization Software. Annals of Operations Research, 1986, 5425-437.

[13]. C. D'Ambrosio, A. Frangioni, L. Liberti, A.Lodi, Experiments with a Feasibility Pump approachfor non-convex MINLPs, In: P. Festa(ed) Proceedings of the 9th Synposiumon Experimental Algorithms (SEA 2010), Lecture Notes in Computer Science, vol. 6049. Springer, Berlin, 2010.

[14]. C. D'Ambrosio, A. Frangioni, L. Liberti, A. Lodi, A storm of Feasibility Pump for non-convex MINLP. Tech. Rep. OR-10-13, DEIS, Universita di Bologna 2010.

[15]. M. Fischetti, F. Glover, A. Lodi. The Feasibility Pump. Mathematical Programming A 104(1) (2005), 91-104.

[16]. G. Nannicini and P. Belotti, Local Branching for MINPs. Technical Report workingpaper, CMU, 2009.

[17]. L. Liberti, N. Miladenovie, G. Nannicini. A Recipe for Finding Good Solutions to MINLPs. Mathematical Programming Computation, August 2011.

[18]. H. E. Scarf, Testing for Optimality in the Absence of Convexity. In: W. P. Heller, R. M Starr and D. A. Starett (Eds) Cambridge University Press, 1986, pp 117-134.