Latin American Journal of Solids and Structures
www.lajss.org

A Heuristic Algorithm Based on Line-up Competition and Generalized Pattern Search for Solving Integer and Mixed Integer Non-linear Optimization Problems

Abstract
The global optimization of integer and mixed integer non-linear problems has a lot of applications in engineering. In this paper a heuristic algorithm is developed using line-up competition and generalized pattern search to solve integer and mixed integer non-linear optimization problems subjected to various linear or nonlinear constraints. Due to its ability to find more than one local or global optimal points, the proposed algorithm is more beneficial for multi-modal problems. The performance of this algorithm is demonstrated through several non-convex integer and mixed integer optimization problems exhibiting good agreement with those reported in the literature. In addition, the convergence time is compared with LCAs' one demonstrating the efficiency and speed of the algorithm. Meanwhile, the constraints are satisfied after passing only a few iterations.

Keywords
Global optimization, integer optimization, mixed integer optimization, multi-modal problems, constraint optimization.

Behrooz Shahriari [a*]
M. R. Karamooz Ravari [b]
Shahram Yousefi [c]
Mahdi Tajdari [d]

[a,c] Department of Mechanical and Aerospace Engineering, Malek-Ashtar University of Technology, Isfahan, 84145-115, Iran.
[b] Department of Mechanical Engineering, Graduate University of Advanced Technology, Kerman, 76311-33131, Iran.
[d] Department of Mechanical Engineering, Islamic Azad University, Arak Branch, Arak, Iran.

Corresponding author email:
* shahriari@mut-es.ac.ir

http://dx.doi.org/10.1590/1679-78252293

Received 15.07.2015
In revised form 25.09.2015
Accepted 13.10.2015
Available online 09.11.2015

1 INTRODUCTION

A vast number of optimization problems deal with integer variables. Due to the complexity of either the objective function or the constraints, these problems can be a real challenge. The presence of nonlinearities in the objective and constraint functions might imply non-convexity in mixed integer nonlinear programming (MINLP) problems, i.e. the potential existence of multiple local solutions. A mixed integer optimization problem can be formulated as:

$$\begin{array}{ll} \min \quad f(\mathbf{x}) & \\ S.T: & \\ g_i(\mathrm{x}) \le 0 & i = 1,...,p \\ h_j(\mathrm{x}) = 0 & j = 1,...,q \\ x_k \in R & k = 1,...,s \\ x_k \in Z & k = s+1,...,n \\ l_r \le x_r \le u_r & r = 1,...,n \end{array}$$

where n is the number of variables, $\mathbf{x}$ is the vector of variables, and (l_r, u_r) are the boundaries of the variable $\mathrm{x_r}$.

The solution methods are classified into three major classes and named as: relaxation methods, search heuristics, and pattern search methods (Abramson, 2002).

Relaxation methods such as outer approximation(Duran and Grossman, 1986; Fletcher and Leyffer, 1994), generalized bender decomposition(Geoffrion, 1972), branch and bound methods (Dakin, 1965; Leyffer, 1998; Kesavan and Barton, 2000), and extended cutting plane (Kelley, 1960; Marchand et al., 2002; Wang, 2009), involve solving several sub-problems. The solution process needs the linearization of some sub-problems which requires cost function and constraints to be differentiable.

Search heuristics are methods designed to find global optima without using derivative information by systematically searching the solution space (Abramson, 2002). These methods often are based on the principles of natural biological evolution. The most relevant algorithms to solve MINLP are simulated annealing (Gidas, 1985), Tabu search (Glover, 1990; Glover, 1994), and evolutionary algorithms such as Genetic Algorithm (Holland, 1962; Costa and Oliveira, 2001; Deep et al., 2009), Evolutionary Strategy (Costa and Oliveira, 2001), Evolutionary Programming (Fogel et al., 1966), Ant Colony Optimization (Schluter et al., 2009), Particle Swarm Optimization (Coelho, 2009), and Differential Evolution (Ponsich and Coello, 2009; Lin et al., 2004).

Pattern search methods were proposed to minimize a continuous function without any knowledge of its derivative. The class of generalized pattern search (GPS) methods was introduced for solving unconstrained non-linear programming (Box, 1975), and was used to optimize mixed integer constraint non-linear optimization problems (Audet and Dennis, 2001).

The line-up competition algorithm (LCA) which is categorized in evolutionary algorithms was proposed to optimize non-linear (Yan and Ma, 2001) and mixed integer non-linear optimization problems (Yan et al., 2004).

In this paper, an algorithm based on line-up competition and generalized pattern search is proposed to optimize integer and mixed integer non-linear problems. Using this algorithm more than one optimal point could be obtained, which makes it appropriate for multi-modal problems. The present algorithm is simple, easy to implement, and fast. The rest of the paper is organized as follows. In section 2, the proposed algorithm is described and all the required steps are mathematically formulated. Section 3 is allotted to the numerical implementation of the algorithm. Finally, the performance of this algorithm is tested through several examples in section 4.

2. ALGORITHM

This section of the paper is allocated to the description of the proposed algorithm. First, in subsection 2.1,an overall perspective of the algorithm is presented, and the steps which should be followed are explained. In subsections 2.2 to 2.7, these steps are mathematically formulated and their details are discussed.

2.1. Outline on the Present Algorithm

In the present algorithm, a uniform mesh is first generated over the solution space as the initial population. This uniform mesh guarantees that the initial population covers the whole search space not leading to loss any area of the space. In the rest of the paper, each point of the mesh is called a "family". These families are ranked to form a line-up according to the values of their objective functions, i.e. the best family is placed in the first position in the line-up, while the worst is placed in the final position. Based on the position of each family in the line-up, a search space is allocated to each family. In the next step each family produces $2n$ children in its allocated search space, where n is the number of variables. These children are produced using generalized pattern search method which cause that all the directions on the corresponding search space get covered. The members of each family compete with each other, as well as their father, and the best one survives as the father of the next generation. This algorithm can be described as follow:

1. Generate mesh points on the search space and compute the value of the objective function for each family.
2. Rank the mesh points to form a line-up according to their objective function values. For a minimization problem the line-up is an ascending sequence and vice versa.
3. Allocate a search space to each family according to their position in the line-up. The best family (first in the line-up) has the smallest search space, while the worst (final in the line-up) has the biggest search space.
4. Produce $2n$ children using generalized pattern search method. Then, the children compete with each other, as well as their father, and the best one survives as the next generation's father.
5. Update the search space according to the f-th first family. The search space will be expanded if there is at least one improvement in the f-th first family and will be contracted if there is no improvement in the f-th first family. Notice that the value of$f$ is defined by the user and helps to find more than one optimal point.
6. If the stopping criterion isn’t satisfied return to (3).

The above-mentioned steps are described in mathematical terms in the following subsections.

2.2. Mesh Generation

To start the optimization process, it is necessary to generate an initial population. In this paper, the initial population is generated using a regular mesh over the search space. It means that some deterministically generated points are distributed in each direction corresponding to each variable.

Let's define vector **m**, whose arrays, *m1* to *mn*, denote the number of mesh points which should be generated in (l_1,u_1) to (l_n,u_n), respectively. The mesh points related to *mj* are generated in (l_j,u_j) utilizing the following equation.

$$\begin{cases} X_{jt_j} = l_j + \dfrac{t_j - 1}{m_j - 1}(\Delta_j) & \text{j} = 1,\ldots,\text{s} \quad ,\text{t}_\text{j} = 1,\ldots,\text{m}_\text{j} \\ X_{jt_j} = INT[l_j + \dfrac{t_j - 1}{m_j - 1}(\Delta_j)] & \text{j} = s+1,\ldots,\text{n} \quad ,\text{t}_\text{j} = 1,\ldots,\text{m}_\text{j} \end{cases} \tag{1}$$

where $\Delta_j = u_j - l_j$, X_{jt_j} is the t_j-th value in (l_j,u_j), and *INT* denotes the integer operator. Notice that *mj*-s control the number of mesh points.

A matrix, **M**, containing the mesh points is defined, which has *n* rows and *C* columns where *C* is calculated as:

$$C = \prod_{r=1}^{n} m_r \tag{2}$$

Each column of **M** denotes a mesh point in the search space. The arrays of this matrix (*mip*) are generated using Eq.3.

$$M_{ip} = \begin{cases} X_{1t} & if \quad \dfrac{(t-1)C}{m_1} + 1 \le p \le \dfrac{tC}{m_1} \quad \text{and } i = 1 \\ X_{it} \quad if & \dfrac{(\text{a}_\text{i}\text{-1})\text{C}}{\prod_{\text{q}=1}^{\text{i-1}} \text{m}_\text{q}} + \dfrac{(t-1)C}{\prod_{q=1}^{i} m_q} + 1 \le p \le \dfrac{(\text{a}_\text{i}\text{-1})\text{C}}{\prod_{\text{q}=1}^{\text{i-1}} \text{m}_\text{q}} + \dfrac{tC}{\prod_{q=1}^{i} m_q} \quad \text{and i} > 1 \end{cases} \tag{3}$$

$$\text{p} = 1,\ldots,\text{C} \quad ,\text{t} = 1,\ldots,\text{m}_\text{i} \quad ,\text{i} = 1,\ldots\text{n}$$

in which $a_i = 1,\ldots,\prod_{q=1}^{i-1} m_q$. Figure 1 shows the generated mesh for n=3, **m**=$[3\ 2\ 3]^\text{T}$, **l**=$[0\ 2\ 1]^\text{T}$, and **u**=$[1\ 4\ 10]^\text{T}$, with the use of Eq. 3.

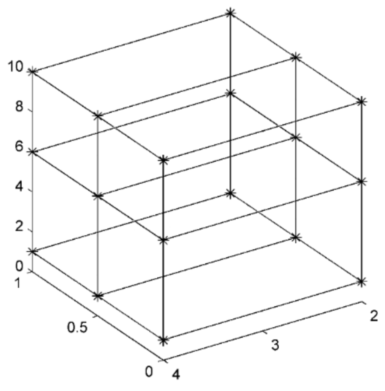


Figure 1: Generated mesh for n=3, **m**=$[3\ 2\ 3]^\text{T}$, **l**=$[0\ 2\ 1]^\text{T}$, and **u**=$[1\ 4\ 10]^\text{T}$ using Eq. 3.

The mesh matrix, **M**, for this example is as follow:

$$\mathbf{M}=\begin{bmatrix} 0 & 0 & 0 & 0.5 & 0.5 & 0.5 & 1 & 1 & 1 & 0 & 0 & 0 & 0.5 & 0.5 & 0.5 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 \\ 1 & 6 & 10 & 1 & 6 & 10 & 1 & 6 & 10 & 1 & 6 & 10 & 1 & 6 & 10 & 1 & 6 & 10 \end{bmatrix}$$

2.3. Ranking the Families

The columns of the mesh matrix, **M**, should be ranked based on their objective function values. Considering $\mathrm{W}=\begin{bmatrix} w_1 & w_2 & ... & w_c \end{bmatrix}$ as a vector, whose arrays are the ranked objective function values, **V** would be a matrix where its columns are the mesh points corresponding to the arrays of **W.**

2.4. Allocation of the Search Space

Allocation of the search space is based on the position of each family in the line-up. In other words, the best family has the smallest search space and the worst has the biggest one. The search space for the *j*-th mesh point is a rectangular region. The lower and upper boundaries of *i*-th variable for *j*-th family (mesh point) in *k*-th generation are calculated using Eq. 4 and Eq. 5, respectively:

$$\begin{aligned} L_{ij}^{(k)} &= \max\left(\left(V_{ij}^{(k)}-\frac{\Delta_i^{(k)}\cdot j}{2C}\right),l_i\right) && \mathrm{i}=1,...,\mathrm{s} \qquad ,\mathrm{j}=1,...,\mathrm{C} \\ L_{ij}^{(k)} &= INT\left[\max\left(\left(V_{ij}^{(k)}-\frac{\Delta_i^{(k)}\cdot j}{2C}\right),l_i\right)\right] && \mathrm{i}=s+1,...,\mathrm{n} \qquad ,\mathrm{j}=1,...,\mathrm{C} \end{aligned} \tag{4}$$

$$\begin{aligned} U_{ij}^{(k)} &= \min\left(\left(V_{ij}^{(k)}+\frac{\Delta_i^{(k)}\cdot j}{2C}\right),u_i\right) && \mathrm{i}=1,...,\mathrm{s} \qquad ,\mathrm{j}=1,...,\mathrm{C} \\ U_{ij}^{(k)} &= INT\left[\min\left(\left(V_{ij}^{(k)}+\frac{\Delta_i^{(k)}\cdot j}{2C}\right),u_i\right)\right] && \mathrm{i}=s+1,...,\mathrm{n} \qquad ,\mathrm{j}=1,...,\mathrm{C} \end{aligned} \tag{5}$$

2.5. Production of Children

In each family *2n* children are produced utilizing GPS(2n). Let's define children generator matrix, D, as follow:

$$\mathrm{D}=\begin{bmatrix} \mathrm{I}_n & \text{-}\mathrm{I}_n \end{bmatrix} \tag{6}$$

where $\mathbf{I}_\mathrm{n}$ is unit matrix. Now, the arrays of the children matrix of s-th family in k-th generation are produced using the following equation:

$$CH_{ij}^{(s,k)}=\begin{cases} V_{is} & D_{ij}=0 \\ L_{is} & D_{ij}=-1 \\ U_{is} & D_{ij}=1 \end{cases} \tag{7}$$

After generating the children, they compete with each other and their father. The winner of the competition will be the father of that family for the next generation. Notice, that if the father wins in the competition, he again stands as the father of the next generation. But, if one of the children wins the competition, the father will be replaced by the best child.

2.6. Update the Search Space

The search space will be expanded if there is at least one improvement in the *f*-th first family as:

$$\Delta_j^{(k+1)} = \min\left(\alpha\Delta_j^{(k)}, \Delta_j\right) \qquad \mathrm{j} = 1,\ldots,\mathrm{n} \tag{8}$$

where α is the expansion factor and α>1.The search space will be contracted if there is no improvement in the *f*-th first family as:

$$\Delta_j^{(k+1)} = \beta\Delta_j^{(k)} \qquad \mathrm{j} = 1,\ldots,\mathrm{n} \tag{9}$$

where β is the contraction factor and 0<β<1.

2.7. Stopping Criterion

The stopping criterion can be explained by Eq.10:

$$\frac{\left|\Delta^{(k)}\right|}{\left|X_{best}^{(k)}\right|} < \varepsilon_1 \qquad and \qquad Max\left(V_f^{(k)}\right) < \varepsilon_2 \tag{10}$$

where $Max\left(V_f^{(k)}\right)$ is the maximum constraint violation of the *f*-th first family, ε_1 is the convergence tolerance parameter, and ε_2 is constraint violation tolerance.

3. IMPLEMENTATION

3.1. Constraint Handling

Constraint handling in optimization problems is a real challenge. In this paper, a pseudo cost function approach is used to replace the original objective function with a pseudo cost function, which is a weighted sum of the original objective function and the constraint violations (Vanderplaats, 1999). Therefore, the pseudo cost function acts as the objective function of the new unconstrained optimization problem. Eq. 11 shows the mathematical explanation of a classical pseudo cost (static penalty function (Back et al., 1997)) function.

$$F = f + P\left(\left|h_1\right| + \left|h_2\right| + \ldots + \left|h_p\right| + \max\left(0, g_1\right) + \max\left(0, g_2\right) + \ldots + \max\left(0, g_m\right)\right) \tag{11}$$

in which P is penalty factor.

3.2. Algorithm Parameters Choice

There are several parameters in the present algorithm which should be chosen by the user. It is important to use appropriate values for these parameters because they influence the performance of the algorithm. These parameters are vector **m**, expansion factor α, contraction factor β, *f* (as described previously), penalty factor p, and stopping criterion tolerances ε_1 and ε_2.

The larger value for the arrays of **m** makes it possible to find the global optimum point more accurately. However, it increases the computational time. It's suitable to choose the value of its arrays according to the optimization cost function, constraints, and boundaries. In other words, for highly non-linear problems, the arrays of **m** might be increased. Also, for a wide boundary of the *j*-th variable, a larger value for m_j is suitable. In this paper m_j=2 is used for all examples except example 12, and the obtained results are winsome, so m_j=2 seems suitable, but in problems with lots of variables, e.g. example 12 m_j=1 can be used.

Expansion and contraction factors influence the quality of the solution and computational time. A larger value for both expansion and contraction factors provides better results, but the computational time increases. Meanwhile, larger values for β provide better global search. Based on our computing experiences, for a simple problem,$\beta \geq 0.4$ is sufficient for finding the global optimal solution. While for a difficult problem, $\beta \geq 0.9$ is recommended. For both simple and difficult problems, α=1 is efficient. Note that the word "simple" refers to problems with a small number of local and global minima, while the word "difficult" refers to noisy problems with several local and global minima. In other words, if there are several minima in the solution space, a small value of β will cause a fast contraction of the search space leading to the loss of some optima.
A larger quantity for the*f* parameter provides a better global search and helps to find several global and local optimal points (if they exist), but the time of convergence to a global solution is possibly much longer. So, the value of this parameter might be chosen according to the non-linearity of the cost function and constraints.

Penalty factor is another important parameter which can affect the performance of the algorithm. The value of this parameter should be large enough in comparison with the cost function value. A value about 10^{10} times greater than the values of the cost function is suitable for this parameter. In addition, when there are several constraints with a different order of magnitudes, all the constraints should be normalized.

Smaller stopping criterion tolerances make the convergence slow and their choice depends on the desired precision.

4. NUMERICAL RESULTS

The performance of the present optimization algorithm is tested in twelve integer and mixed integer non-linear optimization problems taken from the literature. These optimization problems are the test problems for mixed integer non-linear programming (Yan et al, 2004; Costa and Oliveira, 2001; Deep et al, 2009).A more thorough list of the test problems can be found in (Schluter et al, 2009). Table 1 shows the algorithm parameters for each example.

	α	β	f	ε_1	ε_2	P(penalty factor)
Example 1	1	0.4	8	10^{-4}	10^{-6}	10^{10}
Example 2	1	0.4	8	10^{-4}	10^{-6}	10^{10}
Example 3	1	0.9	10	10^{-3}	10^{-6}	10^{4}
Example 4	1	0.9	10	1	10^{-6}	unconstraint
Example 5	1	0.85	10	1	10^{-6}	unconstraint
Example 6	1	0.85	10	1	10^{-6}	10^{9}
Example 7	1	0.95	10	1	10^{-6}	10^{9}
Example 8	1	0.96	8	0.01	10^{-6}	unconstraint
Example 9	1	0.95	10	10^{-3}	10^{-6}	10^{9}
Example 10	1	0.95	10	1	10^{-6}	10^{9}
Example 11	1	0.95	10	10^{-3}	10^{-6}	10^{9}
Example 12	1	0.97	10	10^{-3}	10^{-8}	10^{20}

Table 1: Algorithm parameters for each example.

Example 1: This example is taken from (Yan et al., 2004) and also given in (Costa and Oliveira, 2001; Deep et al., 2009).

$$\begin{aligned}
&\min\ f(\mathrm{x}) = -0.7x_3 + 5(x_1 - 0.5)^2 + 0.8\\
&S.T:\\
&-\exp(x_1 - 0.2) - x_2 \le 0\\
&x_2 + 1.1x_3 \le -1\\
&x_1 - 1.2x_3 \le 0.2\\
&0.2 \le x_1 \le 1\\
&-2.22554 \le x_2 \le -1\\
&x_3 \in \{0,1\}
\end{aligned}$$

The global optimum point is $\mathrm{x}_{\mathrm{opt}} = \begin{bmatrix} 0.9419 & -2.1 & 1 \end{bmatrix}$ with $f(\mathrm{x}_{\mathrm{opt}}) = 1.0765$. The result is in full agreement with other studies. Figure 2 (a) and (b) show the history of pseudo cost function value and convergence parameter, respectively. Referring to this figure, a fast convergence to the global optimal point is achieved.

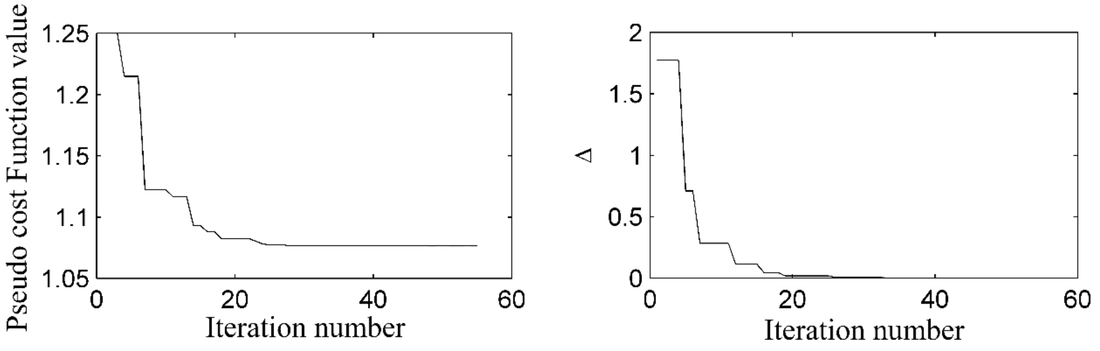


Figure 2: Example 1, (a) pseudo cost function history, (b) Convergence parameter history.

Figure 3 shows the variation of the convergence time with parameter α for several values of β. According to this figure, α=1 and β=0.4 are the best values for fast convergence to the optimal point.

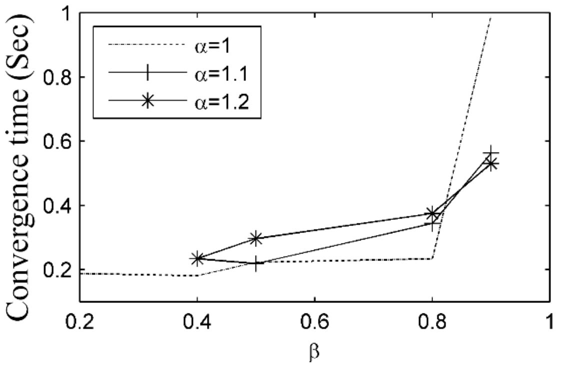


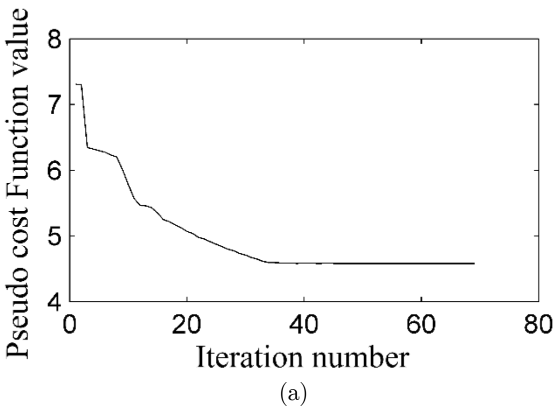
Figure 3: Example 1, convergence time vs. β for three values of α.

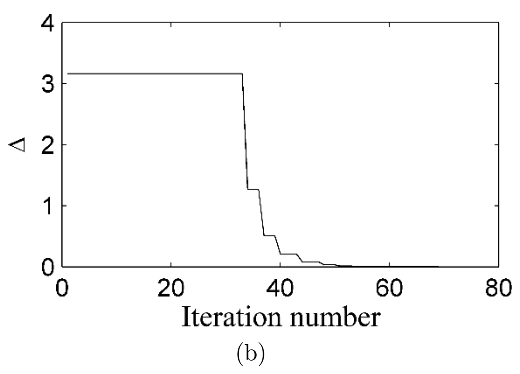
Example 2: This example is taken from (Yan et al., 2004).

$$
\begin{aligned}
&\min\ f(\mathrm{x}) = (x_1-1)^2 + (x_2-2)^2 + (x_3-3)^2 + (x_4-1)^2 + (x_5-2)^2 + (x_6-1)^2 - \ln(x_7+1)\\
&S.T:\\
&x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \le 5\\
&x_1^2 + x_2^2 + x_3^2 + x_6^2 \le 5.5\\
&x_1 + x_4 \le 1.2\\
&x_2 + x_5 \le 1.8\\
&x_3 + x_6 \le 2.5\\
&x_1 + x_7 \le 1.2\\
&x_2^2 + x_5^2 \le 1.64\\
&x_3^2 + x_6^2 \le 4.25\\
&x_3^2 + x_5^2 \le 4.64\\
&x_i \ge 0 \qquad\qquad i = 1,2,3\\
&x_i \in \{0,1\} \qquad i = 4,\dots,7
\end{aligned}
$$

The global optimum point is $\mathrm{x}_{\mathrm{opt}} = \begin{bmatrix}0.2 & 0.8 & 1.9079 & 1 & 1 & 0 & 1\end{bmatrix}^T$ with $f(\mathrm{x}_{\mathrm{opt}}) = 4.5796$. The obtained optimum point is in full agreement with that reported in(Yan et al., 2004). The pseudo cost function, convergence parameter, and maximum constraint violation history of the problem are depicted in figure 4 (a)-(c), respectively, showing fast convergence to the optimum point. The maximum constraints violation is almost zero in all iterations, meaning that the generated mesh covers the search space appropriately.

Figure 5 compares the convergence time for several values of α and β. Referring to this figure, α=1 and 0.5≤β≤0.8 are the best values for fast convergence to the optimum point.





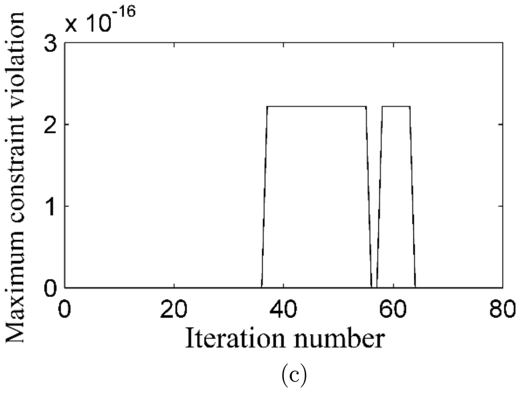


Figure 4: Example 2, (a) pseudo cost function history, (b) Convergence parameter history, (c) Maximum constraint violation history.

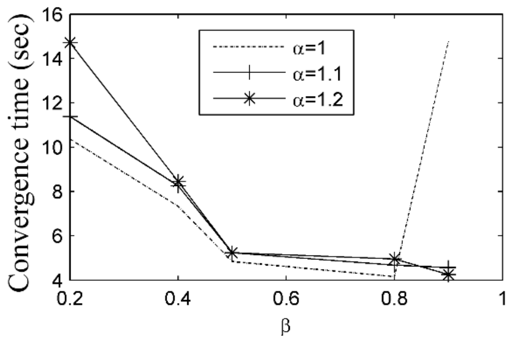


Figure 5: Example 2, convergence time vs. β for three values of α.

Example 3: This is a synthesis problem of a process system, taken from (Yan et al., 2004).

$$\min\ f(\mathrm{x}) = \sum_{i=1}^{4}(a_i \exp(x_{(i+4)}) - b_i \ln(11x_i))$$

$S.T:$

$x_1 + x_2 = e$

$$\begin{aligned}
&x_1 + x_2 - x_3 - x_4 = 0\\
&x_5 + x_6 \geq 4\\
&x_7 + x_8 \geq 4\\
&4x_i \leq x_{(i+4)} \qquad i = 1,\dots,4\\
&x_i \geq 0 \qquad i = 1,\dots,4\\
&x_i \in \{0,1,2,3,4\} \qquad i = 5,\dots,8
\end{aligned}$$

where $a_1 = 2.1$, $a_2 = 0.1$, $a_3 = 4.1$, $a_4 = 0.1$, $b_1 = 3$, $b_2 = 1$, $b_3 = 3$, $b_4 = 4$ and $e = 1$. The obtained optimum point is $\mathrm{x}_{\mathrm{opt}} = \begin{bmatrix} 0.25 & 0.75 & 0.25 & 0.75 & 1 & 3 & 1 & 3 \end{bmatrix}$ with $f(\mathrm{x}_{\mathrm{opt}}) = 4.2498$ which is in full agreement with the data coming from (Yan et al., 2004).

Figure 6 (a)-(c) show the pseudo cost function, convergence parameter, maximum constraint violation history of the problem, respectively. Since there are 2 equal constraints, finding the feasible region would require that some iterations to be passed. Accordingly, all the constraints are satisfied after 26 iterations.

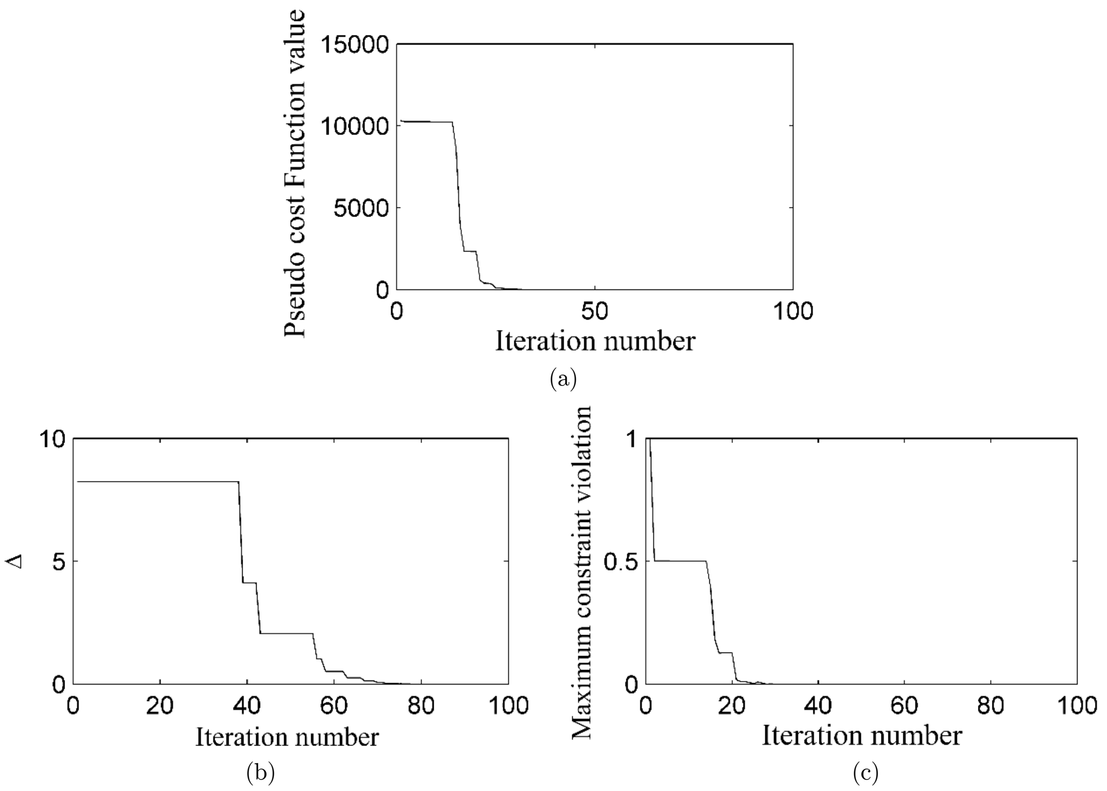


Figure 6: Example 3, (a) pseudo cost function history, (b) Convergence parameter history, (c) Maximum constraint violation history.

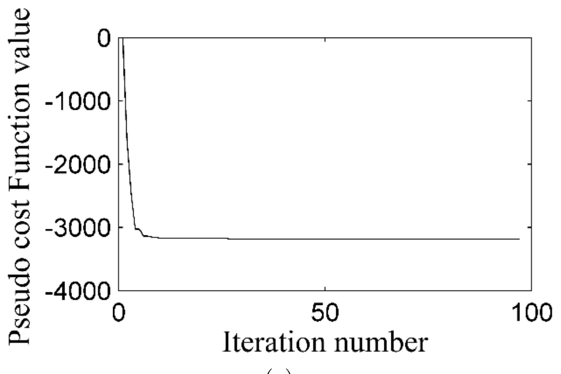
Example 4: This example is taken from (Tian et al., 1998).

$$\min\ f(\mathrm{x}) = (x_1-3)^2\cos(\pi.x_1) + (x_2-6)\sin(\frac{\pi}{4}.x_2) + \frac{(x_3-2.5)^2}{x_2+2} + (x_3+2)^3\exp(-x_4)$$
$$S.T:$$
$$x_i \in \{0,1,2,\ldots,60\} \qquad i=1,\ldots,4$$

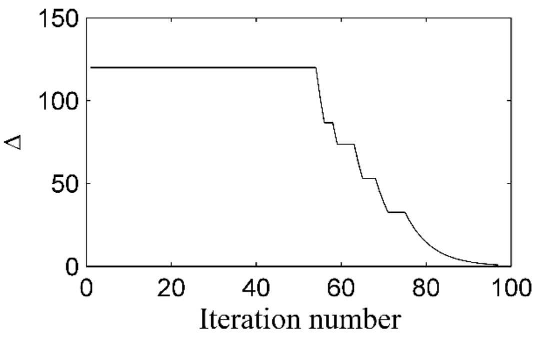
In this problem, all the variables are limited to have integer values. There are many local and global optimum solutions for this problem. Using an appropriate value for parameter *f*, several optimum points can be found. Table 2 shows 10 optimum points and their corresponding optimum values which are obtained by this algorithm. All the optimum points have the same value and the algorithm finds them with only one run through 95 iterations (Figure 7) which takes only 1.812 sec to converge.

	Optimum point	Optimum value
Optimum point 1	$[59\ \ 54\ \ 2\ \ 57]^T$	-3183.9955
Optimum point 2	$[59\ \ 54\ \ 3\ \ 60]^T$	-3183.9955
Optimum point 3	$[59\ \ 54\ \ 2\ \ 51]^T$	-3183.9955
Optimum point 4	$[59\ \ 54\ \ 2\ \ 52]^T$	-3183.9955
Optimum point 5	$[59\ \ 54\ \ 2\ \ 41]^T$	-3183.9955
Optimum point 6	$[59\ \ 54\ \ 3\ \ 60]^T$	-3183.9955
Optimum point 7	$[59\ \ 54\ \ 2\ \ 55]^T$	-3183.9955
Optimum point 8	$[59\ \ 54\ \ 2\ \ 53]^T$	-3183.9955
Optimum point 9	$[59\ \ 54\ \ 2\ \ 54]^T$	-3183.9955
Optimum point 10	$[59\ \ 54\ \ 2\ \ 42]^T$	-3183.9955

Table 2: Example 4 optimal points.



(a)



(b)

Figure 7: Example 4, (a) pseudo cost function history, (b) Convergence parameter history.

Example 5: This example is taken from (Tian et al., 1998).

$$\min\ f(\mathrm{x}) = (x_1 - 2.5)^2 (x_2 + 12.6)^2 (x_3 + 25.4) + \frac{(x_3 - 4.5)^2 \exp(x_2 - 6.5)}{x_4 + 18.4} +$$
$$x_4^3 (x_5 + 10.8)^2 \sin\left(\frac{\pi}{10} x_5 (x_6 + 1)\right)$$
$$S.T:$$
$$x_i \in \left\{-10, -9, \ldots, 9, 10\right\} \qquad i = 1, \ldots, 6$$

This problem contains many global and local optimum points, of which 6 are found after only 45 iterations (Figure 8). Table 3 presents the optimal points and their values.

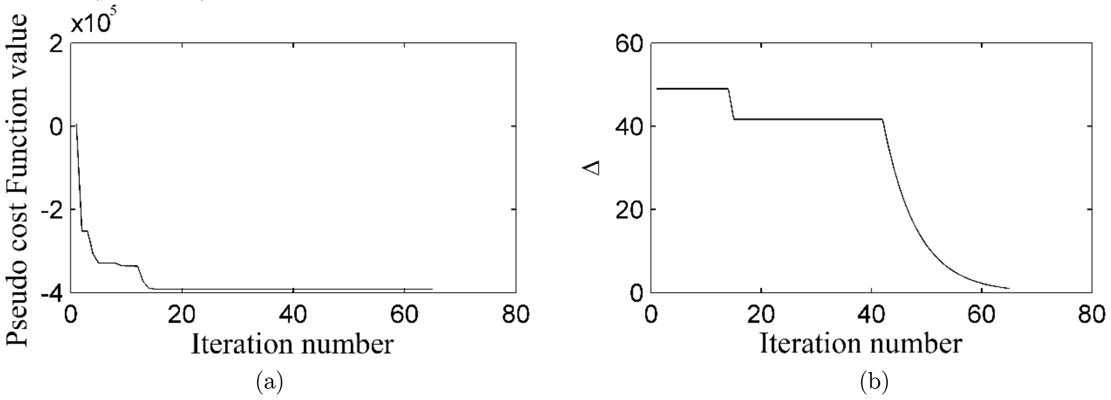


Figure 8: Example 5, (a) pseudo cost function history, (b) Convergence parameter history.

	Optimum point	Optimum value
Optimum point 1	$\begin{bmatrix}3 & -10 & -10 & 10 & 9 & -6\end{bmatrix}^T$	-392013.974
Optimum point 2	$\begin{bmatrix}2 & -10 & -10 & 10 & 9 & -6\end{bmatrix}^T$	-392013.974
Optimum point 3	$\begin{bmatrix}2 & -10 & -10 & -10 & 9 & 4\end{bmatrix}^T$	-392013.974
Optimum point 4	$\begin{bmatrix}-1 & -10 & -10 & 10 & 9 & -6\end{bmatrix}^T$	-390764.726
Optimum point 5	$\begin{bmatrix}3 & -10 & -10 & -10 & 9 & -7\end{bmatrix}^T$	-372826.1706
Optimum point 6	$\begin{bmatrix}2 & -10 & -10 & -10 & 9 & -7\end{bmatrix}^T$	-372826.1706

Table 3: Example 5 optimal points.

Example 6: This example is taken from (Costa and Oliveira, 2001) and also reported in (Deep et al., 2009).

$$\max\ f(\mathrm{x}) = -5.357854x_1^2 - 0.835689x_3x_4 - 37.29329x_4 + 40792.141$$
$$S.T:$$

$$85.334407 + 0.0056858x_3x_5 + 0.0006262x_2x_4 - 0.0022053x_1x_3 \leq 92$$
$$80.51249 + 0.0071317x_3x_5 + 0.0029955x_4x_5 + 0.0021813x_1^2 \leq 110$$
$$9.300961 + 0.0047026x_1x_3 + 0.0012547x_1x_4 + 0.0019085x_1x_2 \leq 25$$
$$27 \leq x_i \leq 45 \qquad i = 1,2,3$$
$$x_4 \in \{78,79,\ldots,102\}$$
$$x_5 \in \{33,34,\ldots,45\}$$

The global optimum is $x_1 = 27$, $x_4 = 78$ for any combination of x_2 and x_5 . The optimum point is obtained after only 2 iterations which exhibits fast convergence.

Example 7: This example is taken from (Deep et al., 2009).

$$\min\ f(\mathrm{x}) = x_1x_7 + 3x_2x_6 + x_3x_5 + 7x_4$$
$$x_1 + x_2 + x_3 \geq 6$$
$$x_4 + x_5 + 6x_6 \geq 8$$
$$x_1x_6 + x_2 + 3x_5 \geq 7$$
$$4x_2x_7 + 3x_4x_5 \geq 25$$
$$3x_1 + 2x_3 + x_5 \geq 7$$
$$3x_1x_3 + 6x_4 + 4x_5 \leq 20$$
$$4x_1 + 2x_3 + x_6x_7 \leq 15$$
$$x_i \in \{0,1,\ldots,4\} \qquad i = 1,2,3$$
$$x_i \in \{0,1,2\} \qquad i = 4,5,6$$
$$x_7 \in \{0,1,\ldots,6\}$$

Using the present algorithm, 3 global and 2 local optimal points are obtained after 45 iterations, which take 2.484 sec to converge. Table 4 shows the optimum points and their corresponding optimum values.

	Optimum point	Optimum value
Optimum point 1	$\begin{bmatrix}0 & 2 & 4 & 0 & 2 & 1 & 6\end{bmatrix}$	14
Optimum point 2	$\begin{bmatrix}0 & 2 & 4 & 0 & 2 & 1 & 5\end{bmatrix}$	14
Optimum point 3	$\begin{bmatrix}0 & 2 & 4 & 0 & 2 & 1 & 4\end{bmatrix}$	14
Optimum point 4	$\begin{bmatrix}2 & 4 & 0 & 0 & 2 & 1 & 2\end{bmatrix}$	16
Optimum point 5	$\begin{bmatrix}1 & 3 & 2 & 0 & 2 & 1 & 4\end{bmatrix}$	17

Table 4: Example 7 optimal points.

Example 8: This example is taken from (Deep et al., 2009).

$$\min\ f(\mathrm{x}) = \sum_{i=1}^{9}\left[\exp\left(-\frac{(u_i - x_2)^{x_1}}{x_3}\right) - 0.01i\right]^2$$

$$u_i = 25 + \left(-50\ln(0.01i)\right)^{2/3}$$
$$S.T:$$
$$0 \le x_1 \le 5$$
$$x_2 \in \left\{0,1,\dots,25\right\}$$
$$x_3 \in \left\{1,2,\dots,100\right\}$$

The global optimum point is $\mathrm{x}_{\mathrm{opt}} = \begin{bmatrix}1.5 & 25 & 50\end{bmatrix}^T$ with $f(\mathrm{x}_{\mathrm{opt}}) = 0$. It is in full agreement with the results reported in other studies.

Example 9: This example is taken from (Costa and Oliveira, 2001).

$$\min\ f(\mathbf{x}) = 5x + 7x_3 + 6x_4 + 7.5x_5 + 5.5x_6$$
$$x_5 + x_6 = 1$$
$$z_1 = 0.9\left(1 - e^{-0.5x_3}\right)x_1$$
$$z_1 = 0.8\left(1 - e^{-0.4x_4}\right)x_2$$
$$x_1 + x_2 - x = 0$$
$$z_1 + z_2 = 10$$
$$x_3 \le 10x_5$$
$$x_4 \le 10x_6$$
$$x_1 \le 20x_5$$
$$x_2 \le 20x_6$$
$$x_i \ge 0 \quad i = 1,\dots,4$$
$$x_i \in \left\{0,1\right\} \quad i = 5,6$$

The global optimum point is $\mathrm{x}_{\mathrm{opt}} = \begin{bmatrix}13.4252 & 0 & 3.5162 & 0 & 1 & 0\end{bmatrix}^T$ with $f(\mathrm{x}_{\mathrm{opt}}) = 99.2396$. The optimum value of the cost function is reported as $f(\mathrm{x}_{\mathrm{opt}}) = 99.245209$ in (Fogel et al., 1966). As seen, the proposed method provides a better value for the cost function in comparison to the one reported by Fogle et al.

Example 10: This is an integer cubic problem which is taken from (Dickman and Gilman, 1989).

$$\max\ f(\mathbf{x}) = -(x_1x_2x_3 + x_1x_4x_5 + x_2x_4x_6 + x_6x_7x_8 + x_2x_5x_7)$$
$$12 - (2x_1 + 2x_4 + 8x_8) \le 0$$

$$41-(11x_1+7x_4+13x_6)\le 0$$
$$60-(6x_2+9x_4x_6+x_7)\le 0$$
$$42-(3x_2+5x_5+7x_8)\le 0$$
$$53-(6x_2x_7+9x_3+5x_5)\le 0$$
$$13-(4x_3x_7+x_5)\le 0$$
$$2x_1+4x_2+7x_4+3x_5+x_7\le 69$$
$$9x_1x_8+6x_3x_5+4x_3x_7\le 47$$
$$12x_2+x_2x_8+2x_3x_6\le 73$$
$$x_3+4x_5+2x_6+9x_8\le 31$$
$$x_i\in\{0,1,\ldots,7\}\qquad i=1,3,4,6,8$$
$$x_i\in\{0,1,\ldots,15\}\quad i=2,5,7$$

There are eight design variables and ten inequality constraints. The optimum point is $\mathrm{x}_{\mathrm{opt}}=\begin{bmatrix}5 & 4 & 1 & 1 & 6 & 3 & 2 & 0\end{bmatrix}^T$ with $f(\mathrm{x}_{\mathrm{opt}})=-110$, which is in full agreement with (Dickman and Gilman, 1989).

Example 11: This example is taken from (Costa and Oliveira, 2001) and also given in (Lin et al., 2004). The problem contains three integer variables and seven continuous ones subjected to 18 inequality constraints.

$$\min\ f=\sum_{j=1}^{M}\alpha_j N_j V_j^{\beta_j}$$
$$S.T:$$
$$\sum_{i=1}^{N}\frac{Q_iT_{Li}}{B_i}\le H$$
$$V_j\ge S_{ij}B_i$$
$$N_jT_{Li}\ge t_{ij}$$
$$1\le N_j\le N_j^u$$
$$V_j^l\le V_j\le V_j^u$$
$$T_{Li}^l\le T_{Li}\le T_{Li}^u$$
$$B_i^l\le B_i\le B_i^u$$

Where, for the specific problem considered in this paper, M=3, N=2, H=6000, a_j=250, β_j=0.6, N_j^u=3, V_j^l=250 and V_j^u=2500. The values of the other parameters are given as follows:

$$T_{Li}^l=\max\frac{t_{ij}}{N_j^u}$$

$$T_{Li}^{u} = \max t_{ij}$$

$$B_i^l = \frac{Q_i}{H} T_{Li}$$

$$B_i^u = \min_i (Q_i, \min_j (\frac{V_j^u}{S_{ij}}))$$

$$\mathbf{t} = \begin{bmatrix} 8 & 20 & 8 \\ 16 & 4 & 4 \end{bmatrix}$$

$$\mathbf{S} = \begin{bmatrix} 2 & 3 & 4 \\ 4 & 6 & 3 \end{bmatrix}$$

The obtained global optimum point is $\mathrm{x}_{\mathrm{opt}} = \begin{bmatrix} 1 & 1 & 1 & 480 & 720 & 960 & 240 & 120 & 20 & 16 \end{bmatrix}^T$ with $f(\mathrm{x}_{\mathrm{opt}}) = 38499.8$, which is in full agreement with that which was reported in the other studies.

Example 12: This example addresses the optimal design of multiproduct batch plants which is taken from (Goyal and Ierapetritou, 2004) and also given in (Kocis and Grossmann, 1988). This MINLP problem consists of 100 variables, from which 60 are binary ones. The problem contains 217 constraints and a nonlinear objective function. The problem data can be found in (Goyal and Ierapetritou, 2004).

$$\min \; f = \sum_{j=1}^{M} \alpha_j \exp(n_j + \beta_j v_j)$$

$$S.T:$$

$$v_j \geq \ln(S_{ij}) + b_i \qquad i = 1,...,N \quad j = 1,...,M$$

$$n_j + t_{Li} \geq \ln(t_{ij}) \qquad i = 1,...,N \quad j = 1,...,M$$

$$\sum_{i=1}^{N} Q_i \exp(t_{Li} - b_i) \leq H$$

$$n_j = \sum_{k=1}^{N_j^U} \ln(k) Y_{kj} \qquad j = 1,...,M$$

$$\sum_{k=1}^{N_j^U} Y_{kj} = 1 \qquad j = 1,...,M$$

$$0 \leq n_j \leq \ln\left(N_j^U\right) \qquad j = 1,...,M$$

$$\ln\left(V_j^L\right) \leq v_j \leq \ln\left(V_j^U\right) \qquad j = 1,...,M$$

$$\ln\left(T_{Li}^L\right) \leq t_{Li} \leq \ln\left(T_{Li}^U\right) \qquad j = 1,...,M$$

$$\ln\left(B_i^L\right) \leq b_i \leq \ln\left(B_i^U\right) \qquad j = 1,...,M$$

The optimal value of the objective function is 2.68×10^6, and is in full agreement with the value reported in(Goyal and Ierapetritou, 2004).

5. CONCLUSION AND SUMMARY

In this paper, an algorithm was proposed for the solution of constrained, integer and mixed integer, non-linear optimization problems. In this algorithm a deterministic search over the solution space is performed to find the optimal solution. The performance of the proposed algorithm was tested through several integer and mixed integer (including multi-modal) optimization problems. The obtained results were compared with those reported in the literature demonstrating efficiency and fast convergence. One of the most important advantages of this method is the ability to find more than one optimal point with only one run of the computer program. So, this algorithm is suitable for multi-modal optimization problems.

References

Abramson, M. A., (2002). Pattern search algorithms for mixed variable general constrained optimization problems. Dissertation, Houston, Texas.

Audet, C., Dennis, J. E., (2001). Pattern search algorithm for mixed variable programming, SIAM J. Optimiz. 11:53-594.

Back, T., Fogel, D. B., & Michalewicz, Z. (1997). Handbook of evolutionary computation. IOP Publishing Ltd.

Box, G. E. P., (1975). Evolutionary operation: A method for increasing industrial productivity, J. Appl. Statist. 6:81-101.

Coelho, L. D. S., (2009). An efficient particle swarm approach for mixed-integer programming in reliability–redundancy optimization applications, Reliab. Eng. Syst. Saf. 94:830–837.

Costa, L., Oliveira, P., (2001). Evolutionary algorithms approach to the solution of mixed integer non-linear programming problems, Comput. Chem. Eng. 25:257-266.

Dakin, R. J., (1965). A tree-search algorithm for mixed integer programming problems, Comput. J. 8:250-255.

Deep, K., Singh, K. P., Kansal, M. L., Mohan, C., (2009). A real coded genetic algorithm for solving integer and mixed integer optimization problems, Appl. Math. Comput. 212:505-518.

Dickman, B. H., Gilman, M. J., (1989). Technical Note: Monte Carlo Optimization, J. Optim. Theory Appl. 60:149-157.

Duran, M. A., Grossman, I. E., (1986). An outer approximation algorithm for a class of mixed integer nonlinear programs, Math. Prog. 36:306-339.

Fletcher, R., Leyffer, S., (1994). Solving mixed integer programs by outer approximation, Math. Prog. 66:327-349.

Fogel, L. J., Owens, A. J., Walsh, M. J., (1966). Artificial Intelligence through simulated evolution, John Wiley and Sons(New York).

Geoffrion, A. M., (1972). Generalized benders decomposition, J. Optim. Theory Appl. 10:237-260.

Gidas, B., (1985). Non-stationary Markov chains and convergence of the annealing algorithm, J. Statist. Phys. 39:73-131.

Glover, F., (1990). Tabu search, ORSA J. Comp. 1:4-32.

Glover, F., (1994). Tabu search for nonlinear and parametric optimization (with links to genetic algorithms), Discrete Appl. Math. 49:231-255.

Goyal, V., Ierapetritou, M. G., (2004). Computational studies using a novel simplicial-approximation based algorithm for MINLP optimization, Comput. Chem. Eng. 28:1771-1780.

Holland, J. H., (1962). Outline for a logical theory of adaptive systems, J. ACM 3:297-314.

Kelley, J. E., (1960). The cutting plane method for solving convex programs, J. Soc. Indust. and Appl. Math. 8:703-712

Kesavan, P., Barton, P. I., (2000). Generalized branch-and-cut framework for mixed-integer nonlinear optimization problems, Comput. Chem. Eng. 24:1361-1366.

Kocis, G. R., Grossmann, I. E., (1988). Global Optimization of Nonconvex Mixed-Integer Nonlinear Programming (MINLP) Problems in Process Synthesis, Ind. Eng. Chem. Res. 27:1407-1421.

Leyffer, S., (1998). Integrating SQP and branch-and-bound for mixed integer nonlinear programming, Technical Report NA/182, Dundee University.

Lin, Y. C., Hwang, K. S., Wang, F. S., (2004). A mixed-coding scheme of evolutionary algorithms to solve mixed-integer nonlinear programming problems, Comput. Math. Appl. 47:1295-1307.

Marchand, H., Martin, A., Weismantel, R., (2002). Laurence Wolsey, Cutting planes in integer and mixed integer programming, Discrete Appl. Math. 123:397-446.

Ponsich, A., Coello, C. A., (2009). Differential Evolution performances for the solution of mixed-integer constrained process engineering problems, Applied Soft Computing. doi:10.1016/j.asoc.2009.11.030.

Schluter, M., Egea, J. A., Banga, J. R., (2009). Extended ant colony optimization for non-convex mixed integer non-linear programming, Comput. Oper. Res. 36:2217-2229.

Tian, P., Ma, J., Zhang, D. M., (1998). Nonlinear integer programming by Darwin and Boltzmann mixed strategy, Eur. J. Oper. Res. 105:224-235.

Vanderplaats, G. N. (1999). Numerical optimization techniques for engineering design, Vanderplaats Research & Development. Inc., Colorado Springs, CO.

Wang, L., (2009). Cutting plane algorithms for the inverse mixed integer linear programming problem, Oper. Res. Lett. 37:114-116.

Yan, L. X., Ma, D. X., (2001). Global optimization of nonconvex nonlinear programs using line-up competition algorithm, Comput. Chem. Eng. 25:1601-1610.

Yan, L., Shen, K., Hu, S., (2004). Solving mixed integer nonlinear programming problems with line-up competition algorithm, Comput. Chem. Eng. 28:2647-2657.