An Alternating Optimization Approach for Mixed Discrete Non Linear Programming

Salam Nema, John Goulermas, Graham Sparrow, Phil Cook, Paul Helman

An Alternating Optimization Approach for Mixed Discrete Non Linear Programming

Journal:	*Engineering Optimization*
Manuscript ID:	GENO-2008-0181.R2
Manuscript Type:	Original Article
Date Submitted by the Author:	12-Dec-2008
Complete List of Authors:	Nema, Salam; The University of Liverpool; Knowledge Support Systems Ltd Goulermas, John; The University of Liverpool, Department of Electrical Engineering and Electronics Sparrow, Graham; KSS Retail Cook, Phil; KSS Retail Helman, Paul; KSS Retail
Keywords:	Mixed discrete nonlinear programming, problem decomposition, alternate optimization

An Alternating Optimization Approach for Mixed Discrete Non Linear Programming

S. Nema[1,2], J. Y. Goulermas[1], G. Sparrow[2], P. Cook[2] , and P. Helman[2]

[1] Department of Electrical Engineering and Electronics, Brownlow Hill,
The University of Liverpool, Liverpool, L69 3GJ, UK
emails: {s.nema, j.y.goulermas}@liverpool.ac.uk

[2] Knowledge Support Systems Ltd, Seventh Floor, St James's Buildings,
79 Oxford Street, Manchester, M1 6SS, UK
emails: {graham.sparrow, phil.cook, paul.helman}@ kssretail.com

Abstract

This article contributes to the development of the field of Alternating Optimization (AO) and general Mixed Discrete Non-Linear Programming (MDNLP) by introducing a new decomposition algorithm (AO-MDNLP) based on the Augmented Lagrangian Multipliers method. In the proposed algorithm, an iterative solution strategy is proposed by transforming the constrained MDNLP problem into two unconstrained components or units; one solving for the discrete variables, and another for the continuous ones. Each unit focuses on minimizing a different set of variables while the other type is frozen. During optimizing each unit, the penalty parameters and multipliers are consecutively updated until the solution moves towards the feasible region. The two units take turns in evolving independently for a small number of cycles. The validity, robustness and effectiveness of the proposed algorithm are exemplified through some well known benchmark mixed discrete optimization problems.

Keywords

Mixed discrete nonlinear programming; Alternating optimization; Augmented Lagrangian; Decomposition.

I. Introduction

This work addresses the mixed discrete programming problem, which seeks a global optimum to an optimization formulation with an objective function subject to a set of linear and nonlinear constraints where the decision variables are both continuous and discrete. In the last decade, there has been a dramatic increase in the techniques developed to solve MDNLP problems (Leyffer 2001, He *et al.* 2004,

Rao and Xiong 2005); such techniques have been applied in various domains, ranging from the process industry and engineering, to the financial and management sciences as well as operational research sectors. The challenging difficulty of MDNLP problems is their high nonlinearity and non-differentiability due to the combinatorial nature of the associated discrete-valued variables.

The categories of algorithms for solving MDNLPs can be mainly divided into stochastic and deterministic ones. The stochastic methods are employing randomized searches and aim to tackle the problem of local optimality. Examples include Simulated Annealing (Cardoso *et al.* 1997), Genetic Algorithms (Rao and Xiong 2005, Young *et al.* 2007), Differential Evolution (Lampinen and Zelinka 1999), Particle Swarm Optimization (He *et al.* 2004, Yiqing *et al.* 2007), and other hybrid methods (Juang 2004, Zhong *et al.* 2004, Nema *et al.* 2008). The deterministic ones take a different approach and adopt a systematic way of approaching the optimum; popular examples include the Non-Linear Branch-and-Bound (Borchers and Mitchell 1994, Leyffer 2001), Sequential Linearization (Loh and Papalambros 1991, Lamberti and Pappalettere 2005), the Penalty Function approach (Shin *et al.* 1990, Fu *et al.* 1991), and the Lagrangian Relaxation methods (Anstreicher and Wolkowicz 2000, Dillon and O'Malley 2002).

This article proposes an original method for solving MDNLP problems, based on the generic framework of Alternating Optimization (AO) introduced by Bezdek and Hathaway (2003). AO is a very efficient iterative procedure for solving large problems by alternating between restricted subsets of variables. It has good convergence properties, reduced development times and the ability to reduce the risk of getting trapped in a local minima. Its main drawback, however, is that AO cannot be adapted easily for use with constrained optimization problems. In this article, the AO procedure was applied to the constrained formulation of MDNLP by partitioning and processing each discrete and continuous subset of the mixed decision variables with different, and more suitable to each subset, solvers. The solvers combine a standard Quasi-Newton gradient-based method (Rao 1996, Nocedal and Wright 1999), with a Lagrangian formulation of the MDNLP, together with a Branch-and-Bound search (Borchers and Mitchell 1994, Gallardo *et al.* 2007) for the continuous and discrete variables, respectively.

The rest of the article is organized as follows. Section II.A describes the MDNLP formulation with equality and inequality constraints, Section II.B presents the augmented Lagrangian approach for

constrained optimization, while Sections II.C-D briefly review the principles of the Quasi-Newton and the Branch-and-Bound search methods. Section III.A describes the Alternating Optimization method for unconstrained optimization, and the new algorithm is introduced in Section III.B. Numerical examples and comparisons of the new approach are provided in Section IV. Finally, conclusions and suggestions for further work are presented in Section V.

II. Employed Optimization Models and Algorithms

A. The MDNLP formulation

An MDNLP optimization problem contains continuous, integer and discrete variables, with linear and nonlinear constraints, and also constraints on the value sets of the discrete variables. It can be stated as

$$\min_{x,y} \quad f(\vec{x},\vec{y})$$
$$\text{s.t.} \quad \begin{cases} g_i(\vec{x},\vec{y}) \le 0, & i=1,\ldots,m \\ h_j(\vec{x},\vec{y}) = 0, & j=1,\ldots,l \\ \vec{x} \in X \subseteq \Re^{n_c} \\ \vec{y} \in Y \equiv Y_1 \times \ldots \times Y_{n_d} \end{cases} \tag{1}$$

where $\vec{x}$ is the vector of n_c continuous variables and $\vec{y}$ is the vector of n_d discrete variables within the value set X. The problem also accounts for m inequalities $g(\vec{x},\vec{y})$ and l equalities $h(\vec{x},\vec{y})$. Y_k is the discrete value set of each k^{th} discrete variable y_k. The major difficulty that arises in the MDNLP problems is due to the combinatorial nature of the Y variables, as the number of possible solutions rises exponentially with the discrete variables domain. Therefore, complexity analysis characterises MDNLP problems as Non-Polynomial Complete (Vavasis 1991).

B. The augmented Lagrangian multipliers method

An effective way for solving a continuous optimisation problem with constraints using solvers for unconstrained problems, is to convert it to an equivalent unconstrained one by using the penalty approach, where all constraints $g(\vec{x})$ and $h(\vec{x})$ are converted to extra penalty terms added to the objective function $f(\vec{x})$. Such an advanced penalty method is the Augmented Lagrangian Penalty Function (ALPF) (Bazaraa *et al.* 1993, Nocedal and Wright 1999) which combines the properties of the quadratic penalty function and the Lagrangian formulation of the problem. The ALPF can be expressed as

$$\phi(\vec{x},\vec{\lambda},\vec{r}) = f(\vec{x}^k) + \sum_{i=1}^{m} \beta_i(\vec{x}^k)\cdot\left[\lambda_i^k + r_i^k \beta_i(\vec{x}^k)\right] + \sum_{j=1}^{l} h_j(\vec{x}^k)\cdot\left[\lambda_{m+j}^k + r_{m+j}^k h_j(\vec{x}^k)\right] \quad (2)$$

where $\{\lambda_1,\ldots\lambda_m\}$ and $\{\lambda_{m+1},\ldots\lambda_{m+l}\}$ are the Lagrangian multipliers for the m inequalities and the l equalities, respectively. The r_i represent the positive penalty terms for the corresponding two types of constraints. $\beta_i(\vec{x})$ is a used to convert the inequalities to equalities via setting

$$\beta_i(\vec{x}) = \max\left\{g_i(\vec{x}^k), -\frac{\lambda_i^k}{2r_i^k}\right\} \quad (3)$$

The optimum solution $\vec{x}^*$ is computed as a sequence of iterative unconstrained subproblems with regular updates of the penalties r_i^k and the multipliers λ_i^k at each iteration k. The optimization is initialized with the values of $\lambda_i^0 = \lambda_{i+m}^0 = 0$ and $r_i^0 = r_{i+m}^0 = 1$ as suggested by Rao (1996). Because the correct penalty factors and the Lagrangian multipliers are problem dependent and, thus, unknown, they are continually updated as

$$\begin{aligned} \lambda_i^{k+1} &= \lambda_i^k + 2\cdot r_i^k \cdot \beta_i(\vec{x}^k), && \forall\, i = 1,\ldots,m \\ \lambda_{m+j}^{k+1} &= \lambda_{m+j}^k + 2\cdot r_{m+j}^k \cdot h_j(\vec{x}^k), && \forall\, j = 1,\ldots,l \end{aligned} \quad (4)$$

To make the procedure more efficient, instead of fixing the penalties r_i, an adaptation strategy (Bean and Hadj-Alouane 1992) has been used to regulate the penalty decrease/increase. For instance, if a current point $\vec{x}^k$ violates the i^{th} inequality constraint $g_i(\vec{x}^k)$, r_i^k must be increased to eventually move the final solution to the feasible region. The following heuristic is used to update the penalty parameters

$$r_i^{k+1} = \begin{cases} \frac{6}{5} r_i^k & \text{if } g_i(\vec{x}) > \varepsilon \\ \frac{4}{5} r_i^k & \text{if } g_i(\vec{x}) < \varepsilon \\ r_i^k & \text{otherwise} \end{cases} \qquad \forall i = 1,\ldots,m \quad (5)$$

where ε is the user-defined tolerance for acceptable constraint violations. The same rule applies to updating the equality penalty r_{m+j}^{k+1}, based on the violation condition $|h_j(\vec{x})| > \varepsilon$.

The following termination criterion has been used to examine how close the search approaches to the optimum solution. Firstly, the solution is obtained when the relative error between the augmented function in two successive iterations becomes small, according to

$$\left|\frac{\phi(\vec{x}^{k+1},\vec{\lambda}^{k+1},\vec{r}^{k+1}) - \phi(\vec{x}^k,\vec{\lambda}^k,\vec{r}^k)}{\phi(\vec{x}^{k+1},\vec{\lambda}^{k+1},\vec{r}^{k+1})}\right| \le \varepsilon \quad (6)$$

Since ϕ becomes a nonconvex function, it is important to check for optimality of the obtained solution $\vec{x}^*$, as this is the case when the corresponding multiplier vector $\vec{\lambda}^k$ approaches the optimal one $\vec{\lambda}^*$ (Gill *et al.* 1988). The algorithm was terminated if the current feasible point $\vec{x}^*$ satisfies the Karush-Kuhn-Trucker (KKT) conditions which are necessary for $\vec{x}^*$ to be a global optimum of ϕ

$$\begin{aligned} &\nabla f\left(\vec{x}^*\right)+\sum_{i=1}^{m}\lambda_i\nabla g_i\left(\vec{x}^*\right)+\sum_{j=1}^{l}\lambda_{m+j}\nabla h_j\left(\vec{x}^*\right)\approx\varepsilon \\ &\lambda_i\geq 0,\quad \lambda_i\cdot g_i\left(\vec{x}^*\right)=0,\quad \forall i=1,\ldots,m \end{aligned} \tag{7}$$

C. Unconstrained optimization

After a continuous constrained problem is transformed to a continuous unconstrained one through the ALPF, a standard Quasi-Newton (QN) algorithm (Bazaraa *et al.* 1993) can be employed to efficiently minimize it. Second-order gradient-based algorithms proceed towards the minimum point of a minimizing function $f\left(\vec{x}\right)$ in a sequential manner by updating the current solution in each *(k+1)*th iteration as

$$\vec{x}^{k+1}=\vec{x}^k-H\left(\vec{x}^k\right)^{-1}\cdot\nabla f\left(\vec{x}^k\right) \tag{8}$$

To reduce the computational load of estimating the Hessian H at point $\vec{x}^k$ in each iteration, QN builds up curvature information using first-order derivatives by applying the Sherman-Morrison formula (Nocedal 1999)

$$\begin{aligned} &H^{k+1}=H^k+\frac{\left(y^k-H^k s^k\right)\cdot\left(y^k-H^k s^k\right)^T}{\left(y^k-H^k s^k\right)^T s^k} \\ &\text{where}\quad y^k=\nabla f\left(\vec{x}^{k+1}\right)-\nabla f\left(\vec{x}^k\right) \\ &\qquad\quad s^k=-H^k\left(\vec{x}^k\right)^{-1}\cdot\nabla f\left(\vec{x}^k\right) \end{aligned} \tag{9}$$

where H^0 is usually taken to be the identity matrix.

D. The Branch-and-Bound (BB) algorithm

This is an established generic algorithm for efficiently enumerating and searching parts of optimization problems. The BB method for discrete problems (Nemhauser and Wolsey 1988, Floudas 1995) is based on the mechanisms of separation, relaxation and fathoming in a search tree. Its principle lies in successive decompositions of the original problem to smaller disjoint subproblems until an optimal solution is found.

The algorithm starts by solving first the continuous relaxation problem using a Non-Linear Programming (NLP) solver. If all discrete variables take discrete values the search is stopped. Otherwise, a tree search is performed in the space of the discrete variables. Then the algorithm selects one of those discrete variables which take a non-discrete value and branch on it. Branching generates two new subproblems by adding simple bounds to the NLP relaxation. Then, one of the two new NLP problems is selected and solved. If the discrete variables take non-discrete values then branching is repeated, while if one of the fathoming rules is satisfied, then no branching is required, and the corresponding node is flagged as fully explored. When during the search discrete solutions are found, they can provide upper bounds on the optimal value of the original problem. Once a node has been fathomed the algorithm backtracks to another node which has not been explored until all nodes are fathomed. The general operations of the algorithm are shown in Table 1.

Place the continuous relaxation and set upper bound to infinity.
while there are unexamined subproblems/nodes in the tree
    Select an unexplored node.
    Solve the NLP problem on the discrete variable *y*.
    Obtain lower bound.
    if the solution is optimal and *y* value is fractional:
        Branch on *y*.
    endif
    Solve NLP problem until:
        - The subproblem is infeasible, or
        - A discrete feasible solution is found (record the value of this solution as upper bound), or
        - The lower bound is greater than the objective value of a previous discrete solution.
    Continue branching and solving NLP subproblems.
Endwhile

Table 1. Main Branch-and-Bound operations.

III. The Proposed AO-MDNLP Framework

A. Alternating Optimization (AO)

AO is a generic methodology for locating the solution of an optimization problem by partitioning and treating independently the design variables. It has been shown that the AO method very efficiently converges to at least a local minimum regardless of the initialization (Hathaway and Bezdek 2001, 2003). The principle advantage of AO is that it replaces the optimization of the objective function with a sequence of easier optimizations involving the different partitions of the design variables.

If we assume that we have to minimize a function $f(\vec{x})$ of n variables, the original problem can be partitioned into N autonomous subsets of variables (with n_s variables in each s^{th} subset, with $\sum_{s=1}^{N} n_s = n$) and the process of optimization alternates between these subsets until the global problem is completed. The flowchart in Figure 1 illustrates the operation sequencing of AO, where the strikethrough notation $\vec{\not{x}}_i$ indicates variables that are fixed with respect to the current subproblem at index i. In later sections, the parameter t is used to define the number of cycles to be used during the AO optimization process.

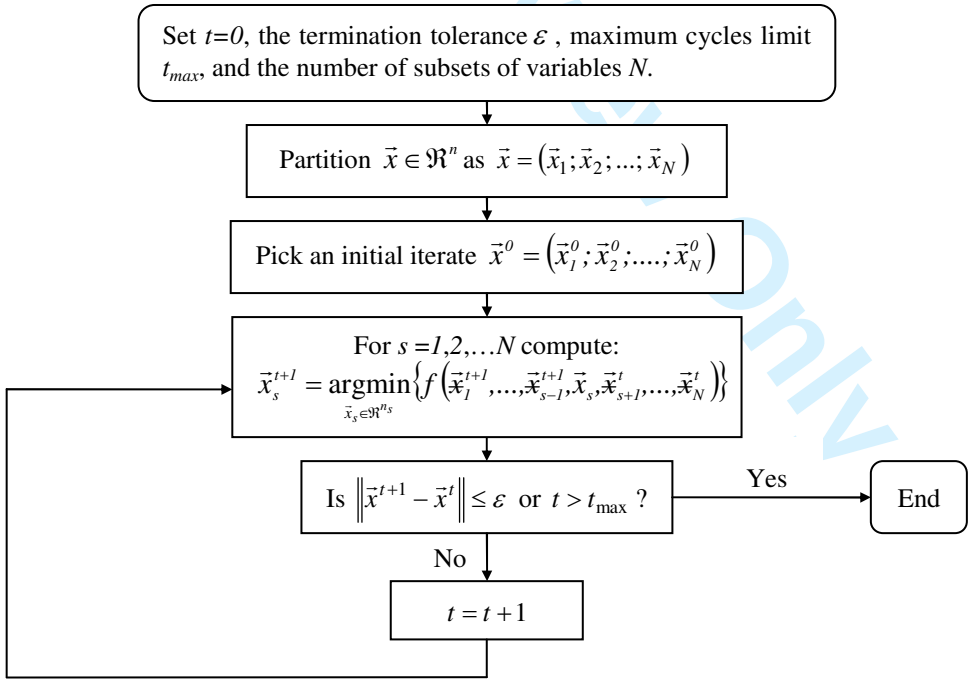


Figure 1. Iteration procedure of Alternating Optimization.

Example: *Hartmann function*

A classic benchmark problem in nonlinear optimization introduced by Dixon and Szego (1978) has been used to demonstrate the application of AO, It minimizes the following

$$f(\vec{x}) = -\sum_{i=1}^{4} \alpha_i \, exp\left[-\sum_{j=1}^{6} B_{ij} \left(x_j - Q_{ij} \right)^2 \right], \qquad \alpha = [1, 1.2, 3, 3.2]'$$

$$B = \begin{bmatrix} 10 & 3 & 17 & 3.05 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 17 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}, \qquad Q = 10^{-4} \times \begin{bmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{bmatrix} \tag{10}$$

This problem has six continuous variables $\vec{x} = (x_1, \ldots, x_6) \in \Re^6$. Suppose we choose $N=2$ arbitrary partitions $\vec{x}_1 = (x_1, x_2, x_3) \in \Re^3$, and $\vec{x}_2 = (x_4, x_5, x_6) \in \Re^3$ of $n_1=n_2=3$ variables each. The minimization can start by setting the initialization point to $\vec{x}^0 = (1,1,1,1,1,1)$, and then minimize $f(\vec{x})$ by alternatively minimizing each subset of the partitioned variables independently.

Cycle t	$\vec{x}_1^t$	$\vec{x}_2^t$	$\left\|\vec{x}^t - \vec{x}^{t-1}\right\|$
0	*(1, 1, 1)*	*(1, 1, 1)*	----
1	*(0.1312, 0.2005, 0.5683)*	*(0.2718, 0.3128, 0.6595)*	*1.6428*
2	*(0.2015, 0.1501, 0.4774)*	*(0.2753, 0.3117, 0.6573)*	*0.1256*
3	*(0.2017, 0.1500, 0.4769)*	*(0.2753, 0.3117, 0.6573)*	*0.0005*
4	*(0.2017, 0.1500, 0.4769)*	*(0.2753, 0.3117, 0.6573)*	*0.0000*

Table 2. Applying AO on the Hartmann function.

Table 2 shows a possible outcome of applying AO on this example. The algorithm converges quickly to the optimum solution requiring only four cycles to satisfy the stopping condition $\left\|\vec{x}^t - \vec{x}^{t-1}\right\| \le 10^{-4}$. This simple example indicates that the AO framework can provide the means for solving many large scale problems that are difficult to process by existing methods, and it leads to easier subproblems with solution spaces much more reduced than the original *n*-dimensional one.

B. The AO-MDNLP algorithm

Based on our observation that MDNLPs have highly structured constraints with mixed variables, This method proposes to partition these MDNLPs by their variables into two subproblems and solve each subproblem as in the example before but using the Lagrangian transformation and also with different and appropriately efficient subsolvers for each subset. This new architecture combines the previously

discussed robust components, namely the AO framework, the ALPF model and the QN and BB algorithms. The rationale behind this variable partitioning is to allow many computationally expensive MDNLP problems to be solved by existing solvers more efficiently. This is possible because the proposed AO-MDNLP method leads to smaller and simpler structured subproblems that are easier to minimize, while the Lagrangian framework supports resolution of the violated constraints across the subproblems using an effective updating strategy.

Because the original problem in Equation (1) consists of the objective function $f(\vec{x},\vec{y})$ and the constraints $g_i(\vec{x},\vec{y})$ and $h_j(\vec{x},\vec{y})$ the constraints (without assuming a specific problem structure) are always associated with both continuous and discrete variables. In order to apply AO, the problem was decomposed into two subproblems; one optimizing the set of $\vec{x}$ and the other the set of $\vec{y}$ variables. To make the handling of the constraints more uniform and also efficient, the ALPF has been used to allow the continuous subproblem to be converted to an unconstrained one. Overall, the proposed method decomposes the MDNLP problem of Equation (1) to two units, where an unconstrained problem is solved at each unit. Unit-*A* fixes all variables $\vec{y}$ and minimizes the ALPF using QN, defined as

$$\begin{aligned}\phi(\vec{x},\vec{y},\vec{\lambda},\vec{r}) &= f(\vec{x}^k,\vec{y}^k)+\sum_{i=1}^{m}\beta_i(\vec{x}^k,\vec{y}^k)\cdot\left[\lambda_i^k+r_i^k\beta_i(\vec{x}^k,\vec{y}^k)\right]+\sum_{j=1}^{l}h_j(\vec{x}^k,\vec{y}^k)\cdot\left[\lambda_{m+j}^k+r_{m+j}^k h_j(\vec{x}^k,\vec{y}^k)\right]\\ \beta_i(\vec{x},\vec{y}) &= \max\left\{g_i(\vec{x}^k,\vec{y}^k),-\frac{\lambda_i^k}{2r_i^k}\right\}\end{aligned} \tag{11}$$

After Unit-*A* performing the full minimization of $\phi(\vec{x},\vec{y},\vec{\lambda},\vec{r})$ with respect to $\vec{x}$, some of the penalty parameters r_i^k and the Lagrangian multipliers λ_i^k are consecutively updated. The unconstrained optimization of $\phi(\vec{x},\vec{y},\vec{\lambda},\vec{r})$ has to be carried out for a sequence of values $\vec{r}$ and $\vec{\lambda}$ until the solution moves towards the feasible region, where the Lagrangian multipliers can be estimated more accurately. The iterative process stops when the augmented function is not changing much between two successive iterations. In the mean time, a test for the satisfaction of the KKT conditions is performed before taking the current solution as an optimum solution.

Subsequently, Unit-*B* takes turn in the optimization process and the continuous variables $\vec{x}$ become

fixed. The unit invokes a Branch-and-Bound algorithm to minimize the discrete variables $\vec{y}$ only, but instead of solving a constrained problem at each node of the BB tree, the augmented function $\phi(\vec{x},\vec{y},\vec{r},\vec{\lambda})$ is minimized for the relaxed component of $\vec{y}$ using QN. This setup is efficient, because at each node the subproblem has n_c less dimensions that the standard unpartitioned BB. The penalty parameters r_i^k and the Lagrangian multipliers λ_i^k have to be consecutively updated at each node in order to find the feasible continuous solution. When solving each subproblem in the BB tree, the following condition must be satisfied before taking the obtained point as a discrete solution

$$\max\left\|y_i^k - d_i^k\right\| \le \varepsilon \tag{12}$$

where y_i^k is the discrete value of the i^{th} discrete variable at the iteration k, and d_i^k is the nearest discrete value for the discrete design variable y_i^k. Once a node has been fully explored, the global search procedures of BB have to be carried out until a discrete solution has been found. The selection method for branching node may significantly affect the performance of BB. Our approach uses the depth-first with backtracking strategy (Ringertz 1988) until all the nodes have been explored.

After convergence of both units, the algorithm composes the final solution by combining the partial final solution generated by each unit. The algorithm terminates with the current solution, if the maximum number of cycles is reached or if the following necessary stopping criteria is met

$$\left\|\left(\vec{x}^{t+1},\vec{y}^{t+1}\right)-\left(\vec{x}^{t},\vec{y}^{t}\right)\right\| \le \varepsilon \tag{13}$$

The overall implementation of the proposed AO-MDNLP is presented in Table 3.

Stage 1: ***Initialization***

Set cycle count $t = 0$, termination tolerance ε, and maximum cycles limit $t_{\max}$.

Pick an initial iterate $\left(\vec{x}^0, \vec{y}^0\right)$, and set $\left(\vec{\lambda}_i^0, \vec{\lambda}_{i+m}^0, \vec{r}_i^{\,0}, \vec{r}_{i+m}^{\,0}\right)$.

Stage2: ***Optimization***

while $(t \le t_{max})$

Form $\phi(\vec{x}, \vec{y}, \vec{\lambda}, \vec{r})$ according to Equation (11).

%Unit-A:

while (the termination criterion in (6) and (7) is not met)

Minimize $\phi(\vec{x}, \vec{y}, \vec{\lambda}, \vec{r})$ using QN method.

Update the parameters $\vec{\lambda}, \vec{r}$ according to Equation (4) and (5).

end while

Record the obtained solution $(\vec{x}^t, \vec{y}^t)$.

%Unit-B:

while (there are unexamined nodes in the BB tree)

Minimize $\phi(\vec{x}, \vec{y}, \vec{\lambda}, \vec{r})$ at each node on the discrete variable y.

end while

Record the obtained solution $(\vec{x}^t, \vec{y}^t)$.

Stage 3: ***Convergence***

if (the necessary stopping condition in (13) is met)

Terminate the algorithm with $(\vec{x}^*, \vec{y}^*)$ as an optimum solution.

else

Increase cycle number as *t=t+1*.

end if

end while

Table 3. Pseudo-Code for the AO-MDNLP algorithm.

IV. Numerical Experimentation

In this section, the performance of the proposed AO-MDNLP was investigated with a number of difficult real-world bench problems from mechanical engineering and chemical process synthesis, frequently employed in the literature. In all experiments, the constraint tolerances $\varepsilon_{g,h} = 10^{-4}$ are used for both equality and inequality constraints. A complete implementation of the AO-MDNLP algorithm has been developed in Matlab 7.4, running on a 2.0GHz Pentium 4 CPU with 1GB of RAM.

A. Results

Experiment 1:

This is a nonconvex problem from Floudas (1995), which involves a process flow sheeting problem. It has two continuous variables and one discrete variable with three linear and nonlinear inequality constraints, and is given by

$$\begin{aligned} min \quad & f(\vec{x},\vec{y}) = -0.7\,y_1 + 5(x_1 - 0.5)^2 + 0.8 \\ subject\ to & \begin{cases} -\exp(x_1 - 0.2) - x_2 \le 0 \\ x_2 + 1.1\,y_1 \le -1.0 \\ x_1 - y_1 \le 0.2 \\ 0.2 \le x_1 \le 1 \\ -2.22554 \le x_2 \le -1 \\ y_1 \in \{0,1\} \end{cases} \end{aligned} \tag{14}$$

The optimum results obtained by the present approach are listed in Table 4.

Cycle t	$\left(\vec{x}_A^t; \vec{y}_A^t\right) = (x_1, x_2, y_1)$	$\left(\vec{x}_B^t; \vec{y}_B^t\right) = (x_1, x_2, y_1)$	$f(\vec{x}, \vec{y})$	$\left\|\left(\vec{x}^t, \vec{y}^t\right) - \left(\vec{x}^{t-1}, \vec{y}^{t-1}\right)\right\|$
0	*(1, 1, 1)*	*(1, 1, 1)*	*1.35*	*----*
1	*(0.5944, -1.4835, 0.4395)*	*(0.5944, -1.4835, 1.00)*	*0.1446*	*2.5164*
2	*(0.9418, -2.0998, 1.00)*	*(0.9418, -2.0998, 1.00)*	*1.0758*	*0.7075*
3	*(0.9418, -2.0998, 1.00)*	*(0.9418, -2.0998, 1.00)*	*1.0758*	*0.0000*

Table 4. Alternating Optimization results of experiment 1.

Experiment 2:

This problem arises in the synthesis of chemical process, and it was investigated by Duran and Grossmann (1986). The goal is to determine the optimal solution of a chemical process system. The problem has three continuous variables and three discrete variables with six linear and nonlinear inequality constraints. The master problem can be formulated as follows

$$\begin{aligned} min \quad & f(\vec{x},\vec{y}) = 5y_1 + 6y_2 + 8y_3 + 10x_1 - 7x_3 - 18\times\log(x_2+1) - 19.2\times\log(x_1 - x_2 + 1) + 10 \\ subject\ to & \begin{cases} 0.8\times\log(x_2+1) + 0.96\times\log(x_1 - x_2 + 1) - 0.8x_3 \ge 0 \\ \log(x_2+1) + 1.2\times\log(x_1 - x_2 + 1) - x_3 - 2y_3 \ge -2 \\ x_2 - x_1 \le 0 \\ x_2 - 2y_1 \le 0 \\ x_1 - x_2 - 2y_2 \le 0 \\ y_1 + y_2 \le 1 \\ 0 \le x_1 \le 2\ , 0 \le x_2 \le 2\ ,\ 0 \le x_3 \le 1 \\ y_1, y_2, y_3 \in \{0,1\} \end{cases} \end{aligned} \tag{15}$$

The optimal solutions obtained using the AO-MDNLP algorithm, are presented in Table 5.

Cycle t	$(\vec{x}_A^t;\vec{y}_A^t)=$ $(x_1,x_2,x_2,\not{y}_1,\not{y}_2,\not{y}_3)$	$(\vec{x}_B^t;\vec{y}_B^t)=$ $(\not{x}_1,\not{x}_2,\not{x}_3,y_1,y_2,y_3)$	$f(\vec{x},\vec{y})$	$\left\|(\vec{x}^t,\vec{y}^t)-(\vec{x}^{t-1},\vec{y}^{t-1})\right\|$
0	*(1, 1, 1, 1, 1, 1)*	*(1, 1, 1, 1, 1, 1)*	*19.5234*	*----*
1	*(1.1542, 0.5502, 1, 0.2751, 0.3020, 0)*	*(1.1542, 0.5502, 1, 0, 1, 1)*	*11.5790*	*1.1073*
2	*(1.3, 0, 0.9995, 0, 1, 1)*	*(1.3, 0, 0.9995, 0, 1, 0)*	*6.0098*	*0.5692*
3	*(1.3, 0, 0.9995, 0, 1, 1)*	*(1.3, 0, 0.9995, 0, 1, 0)*	*6.0098*	*0.0000*

Table 5. Optimal design of process synthesis problem.

Experiment 3:

Consider the optimal design problem of a pressure vessel given in Sandgren (1990). The objective of this problem is to minimize the total cost of materials for forming and welding of a pressure vessel. The design variables of the problem are specified as: $(\vec{x},\vec{y})=(x_1,x_2,y_1,y_2)^T$, which correspond respectively to the sell thickness, spherical head's thickness, shell radius, and shell length, where y_1 and y_2 represent discrete values, integer multiples of 0.0625, while x_1 and x_2 are continuous variables. The mathematical formulation of the problem is

$$\begin{aligned} &min \quad f(\vec{x},\vec{y})=0.6224y_1x_1x_2+1.7781y_2x_1^2+3.1661y_1^2x_2+19.84x_1^2x_1 \\ &subject\ to \begin{cases} 0.0193x_1-y_1\le 0 \\ 0.00954x_1-y_2\le 0 \\ 1{,}296{,}000-\pi x_1^2x_2-\frac{4}{3}\pi x_1^3\le 0 \\ x_2-240\le 0 \\ 0.0625\le y_1\le 6.1875 \\ 0.0625\le y_2\le 6.1875 \\ 10\le x_1\le 200 \\ 10\le x_2\le 200 \end{cases} \end{aligned} \tag{16}$$

The AO cycles results of the pressure vessel design problem are shown in Table 6.

Cycle t	$(\vec{x}_A^t;\vec{y}_A^t)=$ $(x_1,x_2,\not{y}_1,\not{y}_2)$	$(\vec{x}_B^t;\vec{y}_B^t)=$ $(\not{x}_1,\not{x}_2,y_1,y_2)$	$f(\vec{x},\vec{y})$	$\left\|(\vec{x}^t,\vec{y}^t)-(\vec{x}^{t-1},\vec{y}^{t-1})\right\|$
0	*(1, 1, 1, 1)*	*(1, 1, 1, 1)*	*25.4066*	*----*
1	*(159.84, 209.1, 2.733, 1.625)*	*(159.84, 209.1, 2.9375, 1.6250)*	*168,004.3*	*261.8*
2	*(46.19, 180.4, 2.9375, 1.6250)*	*(46.19, 180.4, 2.8750, 1.4375)*	*32,659.5*	*117.2*
3	*(42.0989, 176.6305, 2.8750, 1.4375)*	*(42.0989, 176.6305, 0.8125, 0.4375)*	*6,059.65*	*6.01*
4	*(42.0989, 176.6305 , 0.8125, 0.4375)*	*(42.0989, 176.6305, 0.8125, 0.4375)*	*6,059.65*	*0.000*

Table 6. Optimal design of the pressure vessel.

Experiment 4:

This problem was studied by Duran and Grossmann (1986). It has more continuous and discrete variables; there are 32 possible combinations of the 5 binary variables, of which 11 are feasible as determined by the linear inequality constraint. There are 3 nonlinear inequality constraints and one linear equality constraints. The problem formulation is given below

$$
\begin{aligned}
\min \quad & f(\vec{x},\vec{y}) = 5y_1 + 8y_2 + 6y_3 + 10y_4 + 6y_5 - 10x_1 - 15x_2 - 15x_3 + 15x_4 + 5x_5 - 20x_6 \\
& + \exp(x_1) + \exp(0.83333x_2) - 60 \times \log(x_4 + x_5 + 1) + 140 \\
\text{subject to} \quad & \begin{cases}
-\log(x_4 + x_5 + 1) \le 0 \\
\exp(x_1) - 10y_1 \le 1 \\
\exp(0.833333x_2) - 10y_2 \le 1 \\
1.25x_3 - 10y_3 \le 0 \\
x_4 + x_5 - 10y_4 \le 0 \\
-2x_3 + 2x_6 - 10y_5 \le 0 \\
-x_1 - x_2 - 2x_3 + x_4 + 2x_6 \le 0 \\
-x_1 - x_2 - 0.75x_3 + x_4 + 2x_6 \le 0 \\
x_3 - x_6 \le 0 \\
2x_3 - x_4 - 2x_6 \le 0 \\
-0.5x_4 + x_5 \le 0 \\
-0.2x_4 - x_5 \le 0 \\
y_1 + y_2 = 1 \\
y_4 + x_5 \le 1 \\
0 \le x_1 \le 2 \ , \ 0 \le x_2 \le 2 \ , \ 0 \le x_3 \\
0 \le x_4 \ , \ 0 \le x_5 \le 2 \ , \ 0 \le x_6 \le 3 \\
y_1, y_2, y_3, y_4, y_5 \in \{0,1\}
\end{cases}
\end{aligned}
\tag{17}
$$

This example shows that the global solution can be obtained by the algorithm as shown in Table 7.

Cycle t	$(\vec{x}_A^t; \vec{y}_A^t) = (x_1, x_2, x_3, x_4, x_5, x_6, y_1, y_2, y_3, y_4, y_5)$	$(\vec{x}_B^t; \vec{y}_B^t) = (x_1, x_2, x_3, x_4, x_5, x_6, y_1, y_2, y_3, y_4, y_5)$	$f(\vec{x},\vec{y})$	$\left\|(\vec{x}^t, \vec{y}^t) - (\vec{x}^{t-1}, \vec{y}^{t-1})\right\|$
0	*(1,1,1,1,1,1,1,1,1,1,1)*	*(1,1,1,1,1,1,1,1,1,1,1)*	*74.1025*	----
1	*(1.904, 1.9995, 2.6218, 0.6264, 0.3132, 2.6217, 0.5713, 0.4292, 0.3277, 0.094,0)*	*(1.904, 1.9995, 2.6218, 0.6264, 0.3132, 2.6217, 1,1,0,1,0)*	*6.4246*	*3.1125*
2	*(1.999, 2.121, 0, 2.761, 1.381, 0, 1, 1, 0, 1, 0)*	*(1.999, 2.121, 0, 2.761, 1.381, 0, 0, 1, 1, 1, 0)*	*73.5040*	*4.4122*
3	*(0, 2, 1.0784, 0.652, 0.326, 1.0784, 0, 1, 1, 1, 0)*	*(0, 2, 1.0784, 0.652, 0.326, 1.0784, 0, 1, 1, 1, 0)*	*73.0353*	*3.4498*
4	*(0, 2, 1.0784, 0.652, 0.326, 1.0784, 0, 1, 1, 1, 0)*	*(0, 2, 1.0784, 0.652, 0.326, 1.0784, 0, 1, 1, 1, 0)*	*73.0353*	*0.0000*

Table 7. Alternating optimisation of process synthesis problem.

Experiment 5:

This problem was investigated by Kocis and Grossmann (1988), Costa and Oliviera (2001). It tackles the optimal design of multi-product batch plant with M serial processing stages, where fixed amounts Q_i from N products must be produced. This problem contains a large number of nonlinear inequality constraints; it also has 22 continuous variables and 24 discrete variables. The master problem formulation can be stated as:

$$\begin{aligned} min \quad & f = \sum_{j=1}^{M} \alpha N_j V_j^B \\ subject\ to & \begin{cases} \sum_{i=1}^{N} \frac{Q_i T_{Li}}{B_i} \le H \\ V_j \ge S_{ij} B_i \\ N_J T_{Li} \ge t_{ij} \\ 1 \le N_j \le N_j^u \\ V_j^l \le V_j \le V_j^u \\ T_{Li}^l \le T_{Li} \le T_{Li}^u \\ B_j^l \le B_j \le B_j^u \end{cases} \end{aligned} \tag{18}$$

where, the numerical parameters for the model are chosen as: $M = 6$, $N = 5$, $H = 6000$, $\alpha_j = 250$, $\beta_j = 0.6$, $N_j^u = 3$, $V_j^l = 300$, and $V_j^u = 3000$. The values of $T_{Li}^l, T_{Li}^u, B_j^l$, and B_j^u are given by

$$T_{Li}^l = \max \frac{t_{ij}}{N_j^u}, \quad T_{Li}^u = \max t_{ij}, \quad B_j^l = \frac{Q_i^* T_{Li}}{H}, \quad B_j^u = \min\left(Q_i \min_j V_j^u / S_{ij} \right)$$

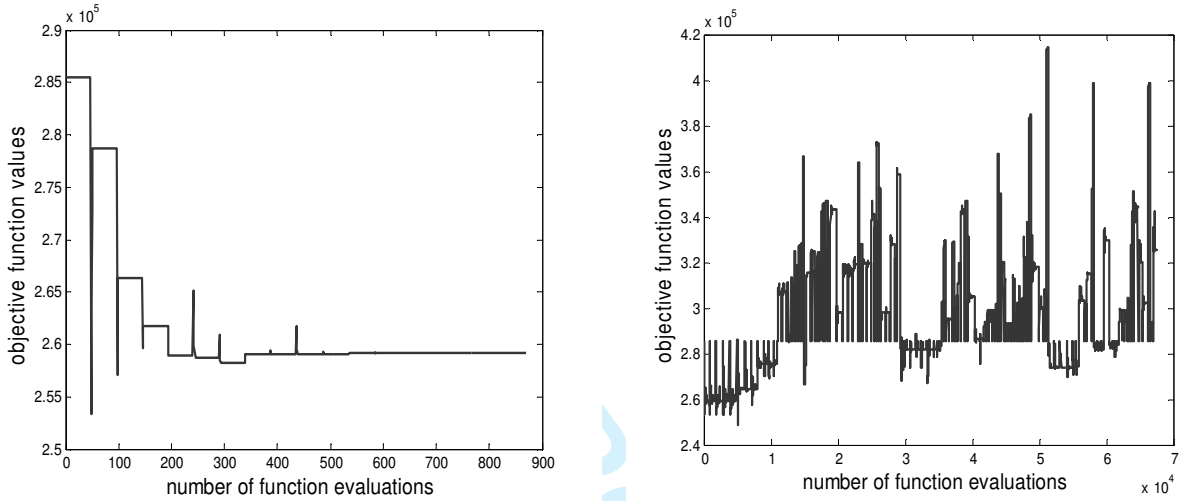
$$where,\ S_{ij} = \begin{bmatrix} 7.9 & 2.0 & 5.2 & 4.9 & 6.1 & 4.2 \\ 0.7 & 0.8 & 0.9 & 3.4 & 2.1 & 2.5 \\ 0.7 & 2.6 & 1.6 & 3.6 & 3.2 & 2.9 \\ 4.7 & 2.3 & 1.6 & 2.7 & 1.2 & 2.5 \\ 1.2 & 3.6 & 2.4 & 4.5 & 1.6 & 2.1 \end{bmatrix}_{N \times M} ,and\ t_{ij} = \begin{bmatrix} 6.4 & 4.7 & 8.3 & 3.9 & 2.1 & 1.2 \\ 6.8 & 6.4 & 6.5 & 4.4 & 2.3 & 3.2 \\ 1.0 & 6.3 & 5.4 & 11.9 & 5.7 & 6.2 \\ 3.2 & 3.0 & 3.5 & 3.3 & 2.8 & 3.4 \\ 2.1 & 2.5 & 4.2 & 3.6 & 3.7 & 2.2 \end{bmatrix}_{N \times M} \tag{19}$$

Table 8 summarizes the optimal results of the batch plant problem, where the optimal solution has been found in four AO cycles with an optimal objective function value of 2.8551×10^5.

The optimal solution (x^*, y^*)
$(\vec{x}, \vec{y}) = [5.9395, 6.6468, 6.5896, 6.4588, 6.2642, 1.1632, 1.2238, 1.8245, 1.2238, 1.3083,$ $0.6931, 0.6931, 1.0986, 0.6931, 0, 0, 8.0064, 7.5452, 7.5882, 7.8706, 7.7528, 7.6544,$ $0,0,0,0,1,1,1,1,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0]_{1\times 46}$

Table 8. Optimal design of batch plant problem.

Note that, not all the AO iterations results for the batch plan problem were included because of its large size. However, the convergence behaviour of the proposed algorithm has been shown in Figure 2, where the AO search process performed in the space of the discrete and continuous variables. The convergence graph Figure 2(a) shows the decrease of the objective function in Unit-*A* while performing the full minimization of $\phi(\vec{x},\vec{y},\vec{\lambda},\vec{r})$ with respect to $\vec{x}$. Figure 2(b) shows the development of the objective function in Unit-B when applying the BB method to minimize $\phi(\vec{x},\vec{y},\vec{\lambda},\vec{r})$ with respect to $\vec{y}$.



(a)Unit 1 evaluation plot (QN method) (b)Unit 2 evaluation plot (BB method)

Figure 2. Objective function evaluations during the AO process.

B. Discussion of results

The performance of the AO-MDNLP algorithm is investigated using five MDNLP problems. Problems 1 to 4 are considered here for the purpose of comparing with the improved Particle Swarm Optimization (PSO) introduced by He *et al.* (2004) and a Hybrid Genetic Algorithm (MDHGA) proposed by Rao and Xiong (2005) which are the state of the art methods for solving MDNLP problems. Problem 3 has been chosen to evaluate the efficiency of the proposed algorithm against other stochastic methods presented in the literature. Problems 5 can be considered as more complex global optimization problems, where AO-MDNLP algorithm is needed to find the optimal values of the discrete and continuous variables. Table 9 summarizes all the obtained results using the proposed approach.

Experiment index	No. of continuous variables	No. of discrete variables	No. of Constraints	No. of AO cycles	Optimal objective function value
1	2	1	3	3	1.0758
2	3	3	6	3	6.0098
3	2	2	4	4	6,059.65
4	6	5	14	4	73.0353
5	22	24	73	4	285,506.5

Table 9. Experimental results of the AO-MDNLP algorithm.

It is important to note that, when the problem is partitioned by its variables, each subproblem is of much smaller scale than the original problem and can be solved in less time with more accuracy than the original problem. This can be exploited in a multi-processor architecture with relaxed synchronisation between the units to enable faster execution. The proposed AO-MDNLP algorithm for handling mixed-variables is found to work efficiently because of using an appropriate solver for optimizing each different type of variables.

For Experiment 1, the optimal solution is *1.0758* which agrees with Floudas (1995). In Experiment 2, the AO-MDNLP converges to the optimum solution after only three cycles; the optimal objective function value of *6.0098* is similar to the best known results reported by Duran and Grossmann (1986).

Experiment	*MDHGA algorithm*		*PSO algorithm*		*AO-MDNLP algorithm*	
	$f^*(\vec{x},\vec{y})$	Function evaluations	$f^*(\vec{x},\vec{y})$	Function evaluations	$f^*(\vec{x},\vec{y})$	Function evaluations
1	1.077	1,221	1.076	1,802	1.0758	690
2	6.15	10,352	6.01	11,589	6.0098	5880
3	7284.02	26,459	6,059.71	28,187	6,059.65	9,765
4	73.124	25,616	73.0468	26,432	73.0353	17,226

Table 10. Comparison of the proposed algorithm performance with PSO and MDHGA.

As shown in Table 10, a comparison is made to evaluate the performance of our approach with the popular PSO and a hybrid GA in terms of both solution accuracy and computational cost. The AO-MDNLP algorithm slightly outperformed both algorithms in terms of solution accuracy. However, the proposed approach provides much better performance in terms of computational cost, as it requires significantly fewer function evaluations to solve each problem.

In order to further assess the efficiency of the proposed algorithm, its results have been compared with those published in the literature such as Evolutionary Algorithm (EA) (Deb 1997), Evolutionary Programming method (EP) (Cao and Wu 1999), and Genetic Algorithm (GA) (Coello and Montes 2001). As shown in Table 11 for Experiment 3, the optimum value of the objective function is only found to be

slightly better than that of the best known solution found by He *et al.* (2004), but with a significant improvement in the number of function evaluations. The number of function evaluations needed is *28,187* in He *et al.* (2004), while in our algorithm the total number of function evaluations required to converge is *9,765.*

Quantity	*MDHGA*	*EP*	*EA*	*GA*	*PSO*	AO-MDNLP
x_1	1.1875	1.000	0.9345	0.8125	0.8125	0.8125
x_2	0.625	0.625	0.5000	0.4375	0.4375	0.4375
x_3	61.4483	51.1958	48.3290	40.097398	42.0984456	42.0989
x_4	27.4037	90.7821	112.6790	176.65404	176.636595	176.6305
g_1	-0.0015	-0.0119	-0.00475	-0.00002	0.00000	0.00000
g_2	-0.0388	-0.1366	-0.038941	-0.035891	-0.0358808	-0.0358
g_3	-963.9357	-13584.5631	-3652.876	-27.886075	0.00000	0.00000
g_4	-212.5963	-149.2179	-127.321	-63.345953	-63.363404	-63.6948
$f(\vec{x}, \vec{y})$	**7,284.02**	**7,108.6160**	**6,410.381**	**6,059.946**	**6,059.714**	**6,059.654**

Table11. Optimal solution of pressure vessel design problem (Experiment 3).

In Experiment 5, the batch plant problem with larger size is used to illustrate the efficiency of the algorithm. It was able to find the optimum solution in only four AO cycles. The optimal objective function obtained was *285,506.5*, with *104,319* function calls. The computational time increases for this problem, but optimal solutions are provided at the end, where other algorithms such as outer approximation method (Duran and Grossmann 1988) fails to give any results as soon as the problem size begins to be larger. Overall, our experiments show that one advantage of the proposed approach is that it is more likely to find the global optimum solution, where it is difficult to achieve in practice. Furthermore, it can cope with problems that involve different search spaces, and makes it possible to solve large-scale optimization problems that may otherwise be computationally difficult and cause the algorithm to fail.

V. Conclusions

In order to improve upon existing optimization methods, this article examines the idea of modifying traditional alternating optimization by introducing an algorithm for solving MDNLP problems. A decomposition technique has been discussed and some computationally efficient procedures have been presented. The key to this technique is an augmented Lagrangian function which preserves separability without violating or using explicit constraints. The proposed approach shows robustness in a diverse

range of problems and that it can be beneficial for cases where the problem has many strongly interacting variables. It should be noted that, this technique allows the use of any method for optimizing each set of variables. The idea should be also extendable to other decomposition strategies; future work could attempt to address further decomposing or portioning subproblems in order to exploit their special structure, so that instead of having two units more units are used to hierarchically decompose the problem. For larger MDNLP problems, the performance of the AO-MDNLP algorithm is still open, where more numerical tests on considerably larger problems can be performed in order to get a more detailed picture of algorithm performance.

References

Anstreicher, K.M., and Wolkowicz, H., 2000. Lagrangian relaxation of quadratic matrix constraints. *SIAM Journal on Matrix Analysis and Applications*, 22, 41–55.

Bazaraa, M.S., Sherali, H.D., and Shetty, C.M., 1993. *Nonlinear Programming: Theory and Algorithms*, 2nd Edition.

Bean, J.C., and Hadj-Alouane, A.B., 1992. *A dual genetic algorithm for bounded integer programs.* University of Michigan, Technical Report.

Bezdek, J.C., and Hathaway, R.J., 2003. Convergence of Alternating Optimization. *Neural, Parallel & Scientific Computations*, 11, 351–368.

Borchers, B., and Mitchell, J.E., 1994. An improved branch and bound algorithm for mixed integer nonlinear programming. *Computers and Operations Research*, 21(4), 359–367.

Cao, Y. J., and Wu, Q. H., 1999. A mixed variable evolutionary programming for optimization of mechanical design. *International Journal of Engineering Intelligent Systems for Electrical Engineering and Communications*, **7(2)**, 77–82.

Cardoso, M.F., Salcedo, R.L, Feyo de Azevedo, S., and Barbosa, D., 1997. A simulated annealing approach to the solution of minlp problems. *Computer Chemical. Engineering*, 21(12), 1349–1364.

Coello, C.A.C., and Montes E.M., 2001. Use of dominance-based tournament selection to handle constraints in genetic algorithms. *Intelligent Engineering Systems through Artificial Neural Networks*, 11, 177–182.

Costa, L., Oliviera, P., 2001. Evolutionary algorithms approach to the solution of mixed integer non-linear programming problems. *Computers & Chemical Engineering*, 25, 257–266.

Deb, K., 1997. A robust optimal design technique for mechanical component design", Evolutionary

Algorithms in Engineering Applications. *Springer-Verlag*, 497–514.

Dillon, J.D., and O'Malley, M.J., 2002. A Lagrangian augmented Hopfield network for mixed integer non-linear programming problems. *Neuro Computing*, 42, 323–330.

Dixon, L.C.W., and Szego, G.P., 1978. *The optimization problem: An introduction.* New York: North Holland.

Duran, M., and Grossmann, I.E., 1986. An Outer approximation algorithm for a class of mixed integer nonlinear programs. *Mathematical programming*, 36, 307–339.

Floudas, C.A., 1995. *Nonlinear and Mixed-Integer Optimization*, Oxford University press.

Fu, J.F., Fenton, R.G., and Cleghorn, W.L., 1991. A mixed integer-discrete-continuous programming method and its application to engineering design optimization. *Engineering Optimization*, 17(4), 263–280.

Gallardo, J.E., Cotta, C., and Fernandez, A.J., 2007. On the Hybridization of Memetic Algorithms With Brach-and-Bound Techniques. *IEEE Transaction Systems, Man, Cybernetics B.*, 37(1), 77–83.

Gill, PE., Murray, W., and Wright, M.H., 1988. *Practical optimization.* Academic, London.

Hathaway, R.J., and Bezdek, J.C., 2001. Local convergence analysis of tri-level alternating optimization. *Neural, Parallel, and Scientific Computation,* 9, 19–28.

He, S., Prempain, E., and Wu, Q.H., 2004. An improved particle swarm optimizer for mechanical design optimization problems. *Engineering Optimization*, 36(5), 585–605.

Juang, C.-F., 2004. A Hybrid of Genetic Algorithms and Particle Swarm Optimization for Recurrent Network Design. *IEEE Transaction Systems, Man, Cybernetics B.*, 34(2), 997–1006.

Kocis, G.R., Grossmann, I.E., 1988. Global optimization of nonconvex mixed-integer nonlinear programming (MINLP) problems in process synthesis. *Industrial & Engineering Chemistry Research*, 27, 1407–1421.

Lamberti, L., and Pappalettere, C., 2005. An efficient sequential linear programming algorithm for engineering optimization. *Journal of Engineering Design*, 16(3), 353–371.

Lampinen, J., and Zelinka, I., 1999. Mixed integer-discrete-continuous optimization by differential evolution. *Proceedings of the 5th International Conference on Soft Computing*, 71–76.

Leyffer, S., 2001. Integrating SQP and branch-and-bound for mixed integer nonlinear programming. *Computational Optimization and Applications*, 18, 295–309.

Loh, H.T., and Papalambros, P.Y., 1991. A sequential linearization approach for solving mixed-discrete nonlinear design optimization problems. *ASME J. Mech. Des.*, 113, 325–334.

Nema, S., Goulermas, J.Y., Sparrow, G., and Cook, P., 2008. A Hybrid Particle Swarm Branch-and-

Bound (HPB) Optimizer for Mixed Discrete Nonlinear Programming. *IEEE Transaction Systems, Man, Cybernetics A.*, 38(6), 1411–1424.

Nemhauser, G., and Wolsey, L., 1988. *Integer and Combinatorial Optimization*. John Wiley and Sons Interscience.

Nocedal, J., and Wright, S.J., 1999. *Numerical Optimization*. Springer-Verlag.

Rao, S.S., 1996. *Engineering Optimization*. 3rd edition, New York: Wiley.

Rao, S.S., and Xiong, Y., 2005. A Hybrid Genetic Algorithm for Mixed-Discrete Design Optimization. *Transaction of the ASME*, 127, 1100–1112.

Ringertz, U.T., 1988. On methods for discrete structural optimization. *Engineering Optimization*, 13(1), 47–64.

Sandgren, E., 1990. Nonlinear integer and discrete programming in mechanical design optimization. *Journal of Mechanical Design*, 112, 223–229.

Shin, D.K., Gurdal, Z., and Griffn Jr, O.H., 1990. A penalty approach for nonlinear optimization with discrete design variables. *Engineering Optimization*, 16(1), 29–42.

Vavasis, S., 1991. *Nonlinear Optimization: Complexity Issues*. Oxford University Press, New York, N.Y.

Yiqing, L., Xigang, Y., and Yongkian, L., 2007. An improved PSO algorithm for solving non-convex NLP/MINLP problems with equality constraints. *Computers and Chemical Engineering*, 31, 153–162.

Young, CT., Zheng, Y., Yeh, CW., and Jang, SS., 2007. Information-guided genetic algorithms approach to the solution of MINLP problems. *Industrial & Engineering Chemistry Research*, 46(5), 1527–1537.

Zhong, W., Liu, J., Xue, M., and Jiao, L., 2004. A multiagents genetic algorithm for global numerical optimization. *IEEE Transaction Systems, Man, Cybernetics B.*, 34(2), 1128–1141.