Available online at www.sciencedirect.com
ScienceDirect
Applied Mathematics Letters 20 (2007) 880–884
Applied Mathematics Letters
www.elsevier.com/locate/aml

A hybrid neural network approach to bilevel programming problems

Kuen-Ming Lan[a], Ue-Pyng Wen[a], Hsu-Shih Shih[b], E. Stanley Lee[c,*]

[a] *Department of Industrial Engineering and Engineering Management, National Tsing Hua University, Hsinchu 300, Taiwan*
[b] *Graduate Institute of Management Sciences, Tamkang University, Tamsui, Taipei 25137, Taiwan*
[c] *Department of Industrial and Manufacturing Systems Engineering, Kansas State University, Manhattan, KS 66506, USA*

Received 13 June 2006; accepted 19 July 2006

Abstract

A combined neural network and tabu search hybrid algorithm is proposed for solving the bilevel programming problem. To illustrate the approach, two numerical examples are solved and the results are compared with those in the literature.

Keywords: Bilevel programming; Multilevel programming; Neural network; Tabu search

1. Introduction

Multilevel programming [1] problems play an important role in decentralized planning for an organization in which decision makers are arranged at hierarchical levels. This decision-making process is made from the top down with some interactions among decision units of the organization. Multilevel problems have been applied in many areas such as government policy, economic systems, finance, transportation and network design, and are particularly applicable in conflict resolution.

The multilevel programming problem is not easy to solve. In fact, the problem has been proved to be NP hard. One way to solve this problem is to use the Karush–Kuhn–Tucker conditions to reduce the problem to a one-level complementary slackness problem. To solve this nonlinear slackness problem, Fortuny and McCarl [2] reformulated it into a mixed 0–1 integer programming problem. In this work, this mixed integer problem is solved by the combined use of tabu search and a neural network.

2. Bilevel programming problem

Consider the bilevel problem with linear coefficients [1,3,4]:

$$P1: \quad \max_{x} F(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{ax} + \boldsymbol{by} \quad \text{where } \boldsymbol{y} \text{ solves}$$
$$\max_{y} f(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{cx} + \mathbf{dy}$$

* Corresponding author. Tel.: +1 785 532 3730; fax: +1 785 532 3738.
E-mail address: eslee@ksu.edu (E.S. Lee).

0893-9659/$ - see front matter
doi:10.1016/j.aml.2006.07.013

$$\text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y} \le \boldsymbol{r}$$
$$\boldsymbol{x}, \boldsymbol{y} \ge 0$$

where $\boldsymbol{a}, \boldsymbol{c}, \boldsymbol{x} \in \Re^{n_1 \times 1}$, $\boldsymbol{b}, \boldsymbol{d}, \boldsymbol{y} \in \Re^{n_2 \times 1}$, $\boldsymbol{r} \in \Re^{m \times 1}$, $\boldsymbol{A}$ and $\boldsymbol{B}$ are $m \times n_1$ and $m \times n_2$ matrices, respectively.

Various approaches have been proposed for solving this problem. One of them uses the Karush–Kuhn–Tucker conditions to reduce the problem to the following one-level problem:

$$P2: \quad \max_{x,y,w} F(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{a}\boldsymbol{x} + \boldsymbol{b}\boldsymbol{y}$$
$$\text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y} + \boldsymbol{u} = \boldsymbol{r}$$
$$\boldsymbol{w}\boldsymbol{B} - \boldsymbol{t} = \boldsymbol{d}$$
$$\boldsymbol{y}\boldsymbol{t} = 0$$
$$\boldsymbol{w}\boldsymbol{u} = 0$$
$$\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}, \boldsymbol{u}, \boldsymbol{t} \ge 0, \quad \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t} \in \Re^{n \times 1} \quad \text{and} \quad \boldsymbol{w}, \boldsymbol{u}, \boldsymbol{r} \in \Re^{m \times 1}$$

The above $P2$ is nonlinear with complementary constraints: $\boldsymbol{y}\boldsymbol{t} = 0$ and $\boldsymbol{w}\boldsymbol{u} = \boldsymbol{0}$. Fortuny and McCarl [2] reformulated the complementary slackness conditions by introducing a 0–1 vector $\boldsymbol{\eta}$. Thus, a mixed integer 0–1 programming problem can be obtained as

$$P3: \quad \max_{x,y,w} F(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{a}\boldsymbol{x} + \boldsymbol{b}\boldsymbol{y}$$
$$\text{s.t.} \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y} + \boldsymbol{u} = \boldsymbol{r}$$
$$\boldsymbol{w}\boldsymbol{B} - \boldsymbol{t} = \boldsymbol{d}$$
$$\boldsymbol{w} \le M(1 - \eta_1)$$
$$\boldsymbol{y} \le M(1 - \eta_2)$$
$$\boldsymbol{u} \le M\boldsymbol{\eta}_1$$
$$\boldsymbol{t} \le M\boldsymbol{\eta}_2$$
$$\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}, \boldsymbol{u}, \boldsymbol{t} \ge 0, \quad \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t} \in \Re^{n \times 1}, \boldsymbol{w}, \boldsymbol{u}, \boldsymbol{r} \in \Re^{m \times 1}, \quad \text{and} \quad \boldsymbol{\eta}_1, \boldsymbol{\eta}_2 \in \{0, 1\}.$$

3. The proposed hybrid procedure

If the number of 0–1 variables is large, problem $P3$ cannot be solved easily. In this work, a hybrid neural network combined with tabu search will be used to solve this problem. The suggested core solver relies on the network by Rodriquez-Vazquez et al. [5], and the 0–1 variables are selected by a tabu strategy. The binary variables selected by tabu strategy form the input for the neural network optimization. If the solution is infeasible, the binary variables are marked as forbidden; otherwise, further binary variables are generated for the next iteration. The process continues until either the tabu maximum terminal size is reached or there is no better solution in the tabu list. The procedure can be summarized in the following five steps.

Step 1. (*Initialization*)

1a. Set the tabu maximum terminal size $= N$, the tabu list $=$ TL, inventory list $=$ IL, and NA (No admittance) $= 1$.

1b. Set a tabu index $I = 1$ for tabu iteration, and the necessary neural network parameters, including learning rate $= \mu$, maximum time for stopping condition $= T$, maximum number of iterations for neural network $= MI$, and penalty constant for the constraints $= K$.

Step 2. (*The Selection of Tabu strategy*)

2a. Randomly generate the input data $\eta_{[I]}$ which is the discrete variables set from IL′ which is a complementary infeasible variables set IL. Go to 2b.

2b. Check the input data $\eta_{[I]}$, go to Step 3 if $\eta_{[I]}$ is not in TL; Otherwise, repeat the sub-step 2a.

Step 3. (*Neural network computation*)

Utilize the input data $\eta_{[I]}$ to solve $P3$ through the neural network by Rodriquez-Vazquez et al. [5] for obtaining $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}, \boldsymbol{u}$ and $\boldsymbol{t}$. Go to Step 4.

Step 4. (*Decision rule*)

4a. Check if the variables $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}, \boldsymbol{u}$ and $\boldsymbol{t}$ are satisfactory. If all constraints in $P2$ are satisfactory, compute $F_{[I]} = \boldsymbol{ax} + \boldsymbol{by}$ and go to sub-step 4b; Otherwise, go to sub-step 4c.

4b. If $I \leq N$ and $\eta_{[I]}$ is not in TL, record the current $\eta_{[I]}$ and the $F_{[I]}$ in the TL. Set $I = I + 1$ and go to the sub-step 2a; Otherwise, go to Step 5.

4c. Record the input data $\eta_{[I]}$ in the IL, and let $I = I + 1$. Go to sub-step 2a.

Step 5. (*Termination*)

If $I \leq N$ and NA $\leq$ TL, let NA = NA + 1 and $I = I + 1$. Go to Step 2a; otherwise, go to termination. The near-optimal solution is reached with the objective value F^* for problem $P3$.

The above procedure is illustrated by solving two well-known examples.

4. Numerical examples

We will demonstrate the hybrid approach by solving two well-known examples.

Example 1. A bilevel programming problem [3].

$$\max_{x_1} f_1 = -2x_1 + 11x_2 \quad \text{(upper level)}$$

where x_2 solves

$$\max_{x_2} f_2 = -x_1 - 3x_2 \quad \text{(lower level)}$$

s.t.

$$x_1 - 2x_2 \leq 4,$$
$$2x_1 - x_2 \leq 24,$$
$$3x_1 + 4x_2 \leq 96,$$
$$x_1 + 7x_2 \leq 126,$$
$$-4x_1 + 5x_2 \leq 65,$$
$$x_1 + 4x_2 \geq 8,$$
$$x_1, x_2 \geq 0.$$

After applying the Karush–Kuhn–Tucker transformation and the concept due to Fortuny and McCarl [2], the above problem reduced to a problem similar to Problem $P3$. The above-proposed algorithm was used to solve this problem.

The selection of the parameters for the neural network and the tabu list are important factors, which influence the performance of the procedure. The penalty parameter is chosen as 1 for simplification, and the learning rate is set as 0.001 or 0.0001. The stopping condition is set at 10^{-6}, or $|\text{norm}(\boldsymbol{x}(k+1)) - \text{norm}(\boldsymbol{x}(k))| < 10^{-6}$. The maximum number of iterations is 50,000 and the length of the tabu list is chosen as 7, and it is empty initially. The MATLAB [6] software with a personal computer (CPU: Intel Pentium 3.2 GHz, RAM: 512 MB) was used to solve the problem. The best solution obtained is $(x_1*, x_2*) = (17.4500, 10.9080)$ with $(f_1*, f_2*) = (85.085500, -50.1740)$. The results are very near those of Shih et al. [7]. The results based on different penalty values are listed in Table 1.

In Table 1, a couple of solutions are listed under different penalty values. For $K = 1$, the tabu list is empty, because the solution will not converge to a stable point after 50,000 iterations. Stable solutions were obtained at around 50,000 iterations with penalty value $K = 2, 5, 10$. When $K = 20$, the tabu list is also null and the neural network trajectory begins to oscillate.

Example 2. A bilevel decentralized system with a center at the upper level and three independent divisions at the lower level [8].

$$\max_{x_1} f_1 = x_1 + x_2 + 2x_3 + x_4 \quad \text{(upper level)}$$

Table 1
Computational results with various penalty values, K, Example 1

K	Tabu list	Binary variable set	Solution
1	–	–	No feasible solution
2	List 1	[0 0 0 1 1 0 0]	$x_1 = 0.0000$, $x_2 = 2.0000$, $f_1 = 22.0000$
2	List 2	[0 1 1 1 1 1 0]	$x_1 = 5.3028$, $x_2 = 17.2410$, $f_1 = 179.057900$
2	List 3	[0 1 1 0 0 1 1]	$x_1 = 9.8780$, $x_2 = 16.5900$, $f_1 = 162.709500$
2	List 4	[1 0 1 1 1 1 0]	$x_1 = 17.4500$, $x_2 = 10.9080$, $f_1 = 85.085500$
5	List 1	[0 0 0 1 1 0 0]	$x_1 = 0$, $x_2 = 2.0000$, $f_1 = 22.0000$
5	List 2	[1 0 1 1 1 1 0]	$x_1 = 17.4500$, $x_2 = 10.9080$, $f_1 = 85.085500$
10	List 1	[0 0 0 1 1 0 0]	$x_1 = 0$, $x_2 = 2.0000$, $f_1 = 22.0000$
10	List 2	[1 0 1 1 1 1 0]	$x_1 = 17.4500$, $x_2 = 10.9080$, $f_1 = 85.085500$
20	–	–	No feasible solution

Note: Tabu list set at 7.

where x_2, x_3, and x_4 solve

$$\begin{cases} \max\limits_{x_2} f_{21} = x_1 + 3x_2 + x_3 + x_4; & \text{(lower level)} \\ \max\limits_{x_3} f_{22} = x_1 + x_2 + 3x_3 + x_4; & \text{(lower level)} \\ \max\limits_{x_4} f_{23} = x_1 + x_2 + x_3 + 3x_4; & \text{(lower level)} \end{cases}$$

s.t.

$$3x_1 + 3x_2 \le 30, \qquad 2x_1 + x_2 \le 20,$$
$$x_3 \le 10, \qquad x_3 + x_4 \le 15,$$
$$x_4 \le 10, \qquad x_1 + 2x_2 + 2x_3 + x_4 \le 40,$$
$$x_1, x_2, x_3, \text{and } x_4 \ge 0.$$

Following the same procedure of Example 1, the bilevel decentralized problem will be first reformulated as an unconstrained optimization problem, and then solved using the MATLAB [6] software on a personal computer with the length of tabu list set at 7. With $M = 1000, K = 5$ and $\mu = 0.0001$, the solution obtained is $(x_1*, x_2*, x_3*, x_4*) = (5.0004, 4.9996, 10.0000, 5.0000)$ with objective $f_1* = 35.0002$ and $(f_{21}*, f_{22}*, f_{23}*) = (34.9992, 45.0000, 35.0000)$. The solution is quite close to that of Anandalingam's [8]. A table similar to Table 1 was obtained with different penalty functions. However, due to space limitations, this table is omitted. No feasible solutions were obtained for $K = 1$ or 20.

5. Discussions

Besides tabu search, other algorithms such as the branch and bound and the genetic algorithms could also be used to form the desired hybrid approach. In a previous work [7], an augmented Lagrange neural network and a branch and bound tree were combined to solve the multilevel programming problem. The problem is that the branch and bound tree usually requires exhaustive enumeration. For Example 1 with seven binary variables, the computation may take 2^7 tests to obtain the solution using the branch and bound algorithm. The tabu strategy requires the same tests in the worst cases; however, in general, the best solution is obtained with only 40–60 tabu iterations. Furthermore, the current approach requires less than 15×10^4 iterations in the neural network computation compared to the sometimes 10^6 iterations in the previous paper [7].

We also tried to use the genetic algorithm and neural network combination to solve the multilevel problem. We discovered that the initial population size is the key to computational efficiency. With the same example, Example 1, the solution is not easily obtained if the initial population size is chosen as 10. In general, in order to generate enough chromosomes, the population size must equal to or larger than half of the total variables sets ($2^7/2$). Thus, 2^6 iterations will be required to generate the initial population size. Compared to tabu search, the genetic algorithm is not an attractive option.

In conclusion, the proposed approach is fairly efficient compared to the previous branch and bound hybrid approach [7]. The number of iterations is much less and the time needed for 0–1 variables search is also short. The approach would be good for solving general bilevel programming problems.

Acknowledgement

The authors are grateful to National Science Council, Taiwan, ROC, for the financial support under the grant numbers: NSC 93-2213-E-032-014 and NSC 94-2213-E-032-004.

References

[1] E.S. Lee, H.S. Shih, Fuzzy and Multi-Level Decision Making: An Interactive Computational Approach, Springer-Verlag, London, UK, 2001.
[2] J. Fortuny-Amat, B. McCarl, A representation and economic interpretation of a two-level programming problem, J. Oper. Res. Soc. 32 (1981) 783–792.
[3] U.P. Wen, S.T. Hsu, Linear bi-level programming problems—A review, J. Oper. Res. Soc. 42 (1991) 125–133.
[4] J.F. Bard, Practical Bilevel Optimization, Kluwer, London, 1998.
[5] A. Rodriquez-Vazquez, R. Dominguez-Castro, A. Rueda, J.L. Huertas, E. Sanchez-Sinencio, Switched-capacitor neural networks for linear programming, Electron. Lett. 24 (1988) 496–498.
[6] MATLAB (Release 11), The MathWorks Inc., Natick, MA, 2000.
[7] H.S. Shih, U.P. Wen, E.S. Lee, K.M. Lan, H.C. Hsiao, A neural network approach to multiobjective and multilevel programming problems, Comput. Math. Appl. 48 (2004) 95–108.
[8] G. Anandalingam, A mathematical programming model of decentralized multi-level systems, J. Oper. Res. Soc. 39 (1988) 1021–1033.