Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 14, no. 5, May 2018, pp. 1030-1039
DOI: 10.23940/ijpe.18.05.p22.10301039

Auto-Tuning for Solving Multi-Conditional MAD Model

Feng Yao[a], Yi Liu[a], Huifen Chen[a], Chen Li[b,c,*], Zhonghua Lu[b], Jinggang Wang[a], Zhiheng Li[a], and Ningming Nie[b]

[a]*State Grid HeNan Electric Power Company, Zhengzhou, 450052, China*
[b]*Computer Network Information Center, Chinese Academy of Sciences, Beijing, 100190, China*
[c]*University of Chinese Academy of Sciences, Beijing, 100049, China*

Abstract

As an important branch of Integer Programming (IP), Mixed Integer Nonlinear Programming (MINLP) has been applied in many fields. As a typical MINLP model, solving the multi-conditional MAD model is a NP-hard problem. In order to solve the model efficiently and rapidly, an auto-tuning of the branch-cut algorithm, which is the solving algorithm of the multi-conditional MAD model, is performed by using the CPLEX solver deployed on the Era supercomputer. The experimental results show that the parallel branch-cut algorithm after auto-tuning can improve the computation speed significantly and can obtain comparable results with the algorithm before auto-tuning, and the parallel efficiencies are better and preponderate over 60% when the number of threads is 2 or 4.

Keywords: auto-tuning; mixed integer nonlinear programming; branch-cut algorithm; parallel computing; CPLEX

(Submitted on February 2, 2018; Revised on March 8, 2018; Accepted on April 17, 2018)

1. Introduction

As an important branch of Integer Programming (IP), Mixed Integer Nonlinear Programming (MINLP) is a kind of Nonlinear Programming problem with continuous and integer variables. In recent years, it has been widely used in unit combination of power market [9], water resources management and sharing [6], portfolio selection [10], process combination and design application under uncertain conditions [1,11], management science and control interaction [2] and logistics-based location optimization [12].

As one of the most difficult problems in mathematical programming, MINLP is NP-hard. The characteristic of the integer variable makes the KKT condition in the nonlinear programming unlikely to form, which means that MINLP cannot be solved in polynomial time [7]. Presently, MINLP is solved by two kinds of algorithms: deterministic algorithm and heuristic algorithm. Deterministic algorithm can obtain the optimal solution or determine whether there is a feasible solution to the problem. It mainly includes Branch & Bound algorithm (B&B) [4], Generalized Benders Decomposition algorithm (GDB) [8], Outer Approximation algorithm (OA) [5], Extended the Cut Plane algorithm (ECP) [13] and hybrid algorithm, which is obtained by integrating B&B with OA [3]. As the basis of other deterministic algorithms, B&B is widely used for solving MINLP. But, when the deterministic algorithm is too time-consuming, heuristic algorithms become the choice to solve the problem quickly. Heuristic algorithms mainly include feasibility pump heuristic algorithm, RINS heuristic algorithm, diving heuristic algorithm, etc.

In recent years, mathematical programming solver has been well developed, and a large number of excellent open source and commercial solvers have emerged. As the most famous solver, CPLEX adopts a branch-cut algorithm framework, which combines the branch and bound algorithm, cutting plane algorithm and heuristic algorithm to solve the MINLP problem. But, solving the given mixed integer nonlinear programming model by CPLEX with default parameters cannot reach optimal efficiency, which needs to tune the branch cut algorithm for different models.

* Corresponding author.
E-mail address: lichen@sccas.cn

This paper mainly achieves an auto-tuning of branch-cut algorithm, which is the solving algorithm of multi-conditional MAD model. The rest of the paper is organized as follows. Section 2 provides an overview of the multi-conditional MAD model. In Section 3, we analyze the branch-cut algorithm frame used by CPLEX and describe some important parameters that are related to the branch-cut algorithm. In Section 4, we detail the relevant numerical experiments and analyze the results. Some conclusions are drawn in Section 5.

2. Multi-conditional MAD model

Set column vector $\mathbf{r} \in R^n$, $\boldsymbol{x} \in R^n$, $\boldsymbol{y} \in R^n$, $\boldsymbol{u} \in R^T$ and variable subscript set $I \subseteq N = \{1,\ldots,n\}$, t ∈ {1,...,T}. Then, the multi-conditional MAD model can be described as Equation (1) - (8):

$$max \sum_{i=1}^{n} \left(r_i(x_i - y_i) - c(x_i) - d(y_i)\right) \tag{1}$$

$$st.\ \ (1-\delta)M \le \sum_{i=1}^{n}(x_i + \gamma y_i) \le M \tag{2}$$

$$\frac{2}{T}\sum_{t=1}^{T} u_t \le \omega M \tag{3}$$

$$u_t \ge \sum_{i=1}^{n}(r_{it} - r_i)(x_i - y_i) \quad t = 1,2,3 \ldots\ldots T \tag{4}$$

$$\sum_{i=1}^{n}(z_i + z_i') = num \tag{5}$$

$$L_i * M * z_i \le x_i \le U_i * M * z_i \quad i = 1,2,3 \ \ldots\ldots n \tag{6}$$

$$L_i' * M * z_i' \le y_i \le U_i' * M * z_i' \quad i = 1,2,3 \ \ldots\ldots n \tag{7}$$

$$z_i * z_i' = 0 \quad i = 1,2,3 \ \ldots\ldots n \tag{8}$$

Where c (·) and d (·) are piecewise linear non-convex functions shown in Equation (9):

$$c(x) = d(x) = \begin{cases} 0.003x, & x \le 1500 \\ 0.0023x + 1.05, & 1500 < x < 1760 \\ 0.002x + 1.578, & x \ge 1760 \end{cases} \tag{9}$$

$z_i, z_i' \in \{0,1\}$. δ is a small positive real number. $0< \gamma <1$, $0<\omega<1$. L_i, U_i, L_i', $U_i' \in R \cup \{\pm\infty\}$. num ∈ Z, M ∈ R.

Multi-conditional MAD model is a typical mixed integer nonlinear programming according to the model structure. Decision variables in the model are Var = $\{x_i$、y_i、z_i、$z_i' \mid$ i=1,2,3...n$\}$, and the decision variable scale is 4n where the number of 0-1 integer decision variables is 2n. The number of constraints in the model is 4n+2T+2.

3. Auto-tuning for solving multi-conditional MAD model

As a high-performance mathematical Programming solver under IBM, CPLEX can quickly and steadily solve a series of programming problems such as Linear Programming (LP), Quadratic Programming (QP), Quadratically Constrained Programming (QCP), Mixed Integer Programming. The solver adopts a branch-cut algorithm framework, which includes pre-solving, LP solver, branch strategy, node selection strategy, cutting plane generator, heuristic strategy set, etc.

The branch-cut algorithm that is used to solve MINLP models in CPLEX is obtained by integrating the branch-bound algorithm with the cut plane algorithm and heuristic algorithm. Based on the branch limit algorithm, the branch-cut algorithm tries to find a monotone decreasing upper bound sequence (Equation (10))

$$\overline{z}_1 \ge \overline{z}_2 \ge \cdots \overline{z}_s \ge z^* \tag{10}$$

and monotone increasing lower bound sequence (Equation (11)) of the problem:

$$\underline{z}_1 \leq \underline{z}_2 \leq \cdots \underline{z}_t \leq z^* \tag{11}$$

Then, it obtains the optimal solution when $\bar{z}_s - \underline{z}_t \leq \epsilon$.

Where the upper bound $\bar{z}_i$ ($i = 1, \cdots, s$) is the dual bound (the primal bound, in the case of a minimization problem) of the problem, which is generally obtained by feasible solutions; the lower bound $\underline{z}_i$ ($i = 1, \cdots, t$) is the primal bound (the dual bound, in the case of a minimization problem) of the problem, which is generally obtained by solving relaxation problems.

In order to speed up the solving process of the problem, the branch-cut algorithm strengthens the dual bound $\bar{z}$ by using the cutting plane algorithm, and it uses the heuristic algorithm to strengthen the primal bound $\underline{z}$.

CPLEX solver provides users with a large number of optional parameters such as parallel scale parameters, algorithm parameters, file read parameters, screen output parameters, statistical parameter, etc. Solver parameters can be set by specific function interfaces or by a parameter file. Some important parameters that are related to the branch-cut algorithm are described as follows:

- **cuts**
  The **cuts** parameter sets types of cuts used during mixed integer nonlinear optimization. CPLEX integrates a variety of cutting plane algorithms, including Clique Cut, Cover Cuts, Mixed Integer Rounding Cut, Gomory Fractional Cut, Flow Cover Cut, Implied Bound Cut and Disjunctive Cut, etc. Users can choose one or more different cutting plane methods to enhance the dual bounds in the branch bound algorithm for different models.

- **presolvenode**
  presolvenode is the parameter related to node presolving. It decides whether the presolving is performed by force and whether node probing is conducted.

- **startalgorithm**
  The **startalgorithm** parameter sets the algorithm to solve the initial relaxation. CPLEX integrates a variety of LP algorithms, including primal simplex algorithm, dual simplex algorithm, network simplex algorithm, barrier algorithm, sifting algorithm and so on, to solve the initial relaxation generated by the branch-cut algorithm. The value of the startalgorithm parameter can be set according to the problem.

- **subalgorithm**
  The **subalgorithm** parameter sets the algorithm to solve subproblems, and the alternative algorithms is the same as algorithms in the **startalgorithm** parameter.

- **branch**
  The **branch** parameter sets the direction of the first branch. The value -1 is used to set the *down branch first*, and the value 1 is used to set the *up branch first*. The value 0 is used to set *automatic*.

- **search**
  The **search** parameter sets the indicator for the search method. There are three methods for the branch-cut algorithm in CPLEX: traditional branch-and-cut search, dynamic search, and a combination of the two. In general, dynamic search is suitable for most problems and can be much faster than the other two.

- **variableselect**
  The **variableselect** parameter sets the variable selection strategy, including minimum integer infeasibility, maximum integer infeasibility, pseudo costs, strong branching and pseudo reduced costs for CPLEX.

- **nodeselect**
  The **nodeselect** parameter chooses the node selection strategy to decide which branch problem is solved first. Alternative strategies include depth-first search, best-bound search, best-estimate search and alternate best-estimate search.

- **fpheur**
  fpheur is the parameter related to the feasibility pump heuristic algorithm. It determines whether the feasibility pump heuristic algorithm is used and what kind of feasibility pump heuristic algorithm is used in the branch-cut algorithm.

- **lbheur**
  The **lbheur** parameter sets the indicator for the local branching heuristic, and if it is set to *no*, the heuristic algorithm is disabled in local branching. For a lot of problems, the heuristic algorithm can't strengthen the primal bound $\underline{z}$ in the local branching, so it will slow down the solving process of the problem if we don't disable the algorithm. At this point, the lbheur parameter can be set to *no* to disable the heuristic algorithm.

- **rinsheur**
  The **rinsheur** parameter sets the frequency of calling the RINS heuristic algorithm, and if it is set to -1, CPLEX will never call the RINS heuristic algorithm. It will reduce the computing efficiency if the RINS heuristic algorithm is invoked in a default frequency because the algorithm may be very suitable for solving a problem. Or, it will not work at all. So, you can disable the algorithm or increase the frequency of calling the algorithm by setting the **rinsheur** parameter to get a high computational efficiency.

- **parallel** and **threads**
  The **parallel** parameter sets the type of parallelism used by CPLEX, and the **threads** parameter that is always used together with the parallel parameter specifies a global thread limit. These two parameters jointly determine the parallel scale when the model is solved. The default setting of the **parallel** parameter allows CPLEX to choose between the deterministic and opportunistic mode depending on the **threads** parameter. If the **threads** parameter is set to its default setting, CPLEX chooses the deterministic mode. If the **threads** parameter is set to one, CPLEX runs sequentially in the deterministic mode in a single thread. Otherwise, if the **threads** parameter is set to a value greater than one, CPLEX chooses the opportunistic mode.

Steps for solving MINLP problems by using CPLEX include preprocessing, branching, node selection, solving LP problem, etc. Each step contains a variety of specific algorithms. The solver will test almost all the algorithms in a certain order when it solves the problem, which may be very inefficient. In order to solve the multi-conditional MAD model efficiently and quickly, parameters in CPLEX need to be tuned. The specific tuning on parameters and the quantitative comparison before and after tuning will be described in the experiment in section 4.

4. Numerical experiment and result analysis

4.1. Experimental environment and parameter settings

We use the mathematical programming solver, CPLEX 12.2, which is deployed on the Era supercomputer to solve multi-conditional MAD model under different parameter values in parallel. Tune parameters of the branch-cut algorithm are used to solve the model. Finally, we achieve a quantitative comparison of solving efficiency of the model before and after tuning, and it gives an analysis of parallel efficiency of the branch-cut algorithm after tuning. The detailed experimental environment is shown in Table 1, and key parameters of the model are shown in Table 2.

Table 1. Experimental environment

Parameter	ConfRev
Operating System	CentOS release 6.4(Final)
Linux Kernel	2.6.32-358.el6
Complier	Intel Compiler 2013_sp1.0.080
Network	56Gbps FDR InfiniBand
Blade Compute Node	2 Intel Xeon E5-2680V2(10 cores|2.8Ghz) 64 GB DDR3 ECC 1866MHz memory

Table 2. Key parameters of the model

Parameter	n	M	T	γ	δ	ω	p_t	num	L_i	U_i	L_i '	U_i '
ConfRev	457,1300	10000	200	0.3	0.005	0.05	1/290	2-15	0.01	1	0.01	1

4.2. Experimental results

To get high efficiency, we tune parameters of the branch-cut algorithm appropriately for different situations when we use CPLEX to solve the multi-conditional MAD model:

- When using heuristic algorithms to search feasible solutions, we find most of them are obtained by using the RINS heuristic algorithm. Therefore, we improve the frequency of calling the RINS heuristic algorithm. At the same time, we disable the heuristic algorithms that have not found a solution and are not invoked by any other algorithm.

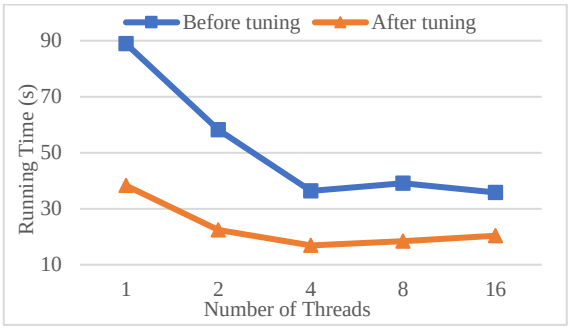
- Choose dual simplex algorithm to solve relaxation problems and subproblems generated during the model solving process.
- Perform presolving by force before nodes are solved.
- Choose best-bound search algorithm as the node selection strategy.
- Set dynamic search as the search method.
- Up branch first is used as the first branch direction.
- To avoid doing too much useless processing in the deeper layers, we limit the maximum depth of branch when optimal solutions are not found in the deeper layers.
- Select Clique Cut to strengthen the dual boundary of the branch-cut algorithm.

The optimal solutions of the model under different parameter values are shown in Table 3. The model calculation time under different parallel scales before and after parameter tuning is shown in Figure 1. The parallel efficiency of the branch-cut algorithm after parameter tuning is shown in Figure 2.

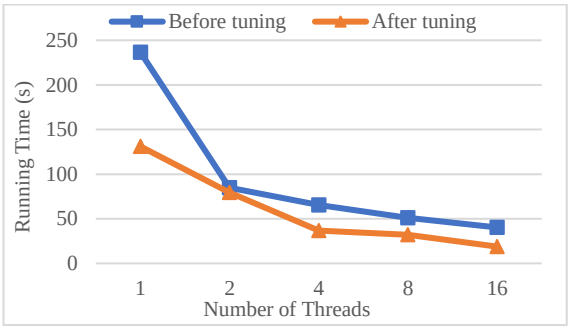
Table 3. The optimal solution of multi-conditional MAD model

n	num	The optimal solution before parameter tuning	The optimal solution after parameter tuning
457	2	109.3114903	109.3114903
457	3	116.7518309	116.7518309
457	4	117.0426611	117.0426611
457	6	117.5375522	117.5375522
457	7	117.5320482	117.5320482
457	8	117.516054	117.516054
457	9	117.4450745	117.4450745
457	11	117.2496022	117.2496022
457	12	117.1461861	117.1461861
457	13	117.0216122	117.0216122
457	14	116.8882934	116.8882934
457	15	116.7595908	116.7595908
1300	2	249.94231166	249.94231166

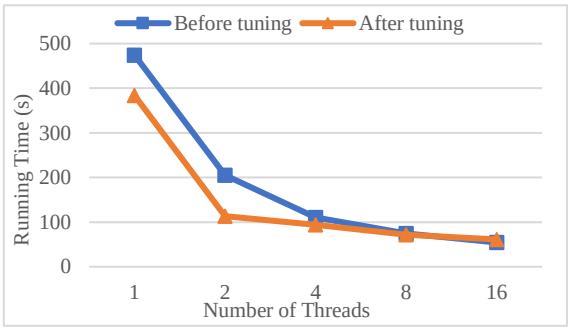
As shown in Table 3, the optimal solutions of the multi-conditional MAD model before and after parameter tuning are exactly the same, that is, the branch-cut algorithm after parameter tuning can obtain comparable results with the algorithm before parameter tuning.



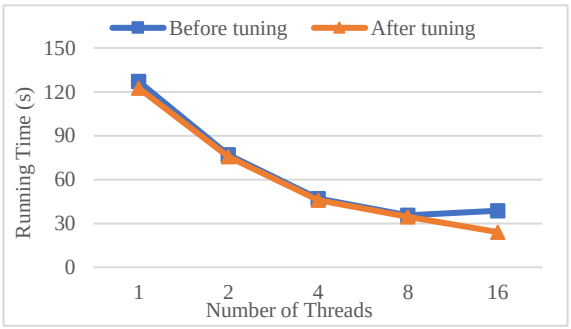
(a) n=457, num=2



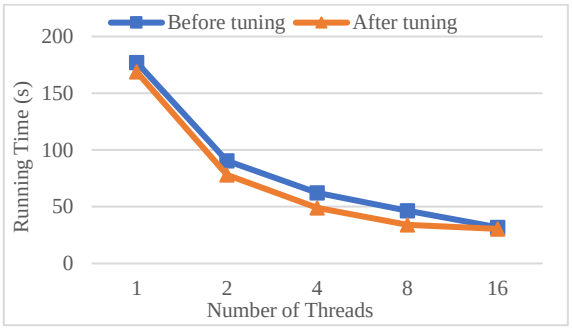
(b) n=457, num=3



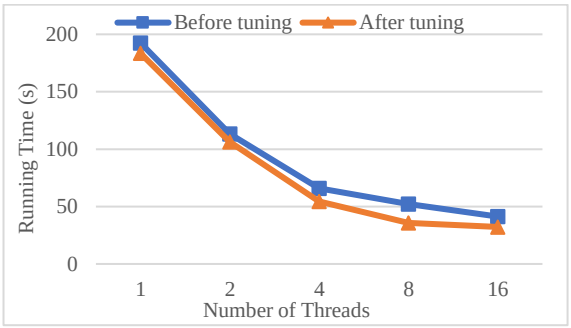
(c) n=457, num=4



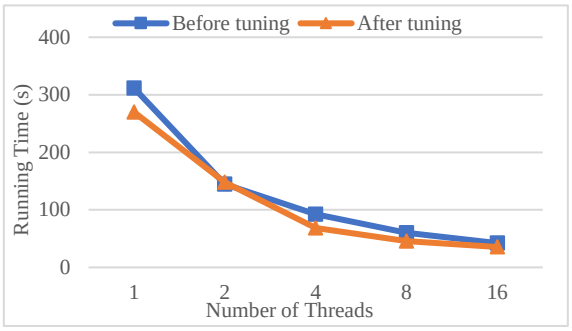
(d) n=457, num=6



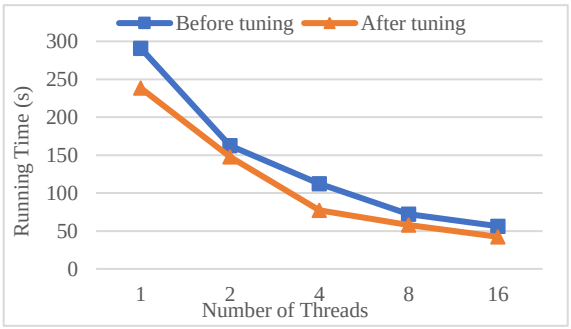
(e) n=457, num=7



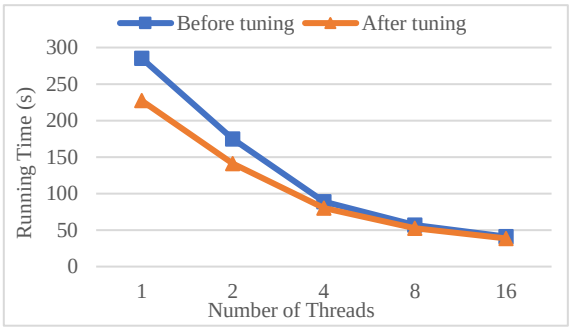
(f) n=457, num=8



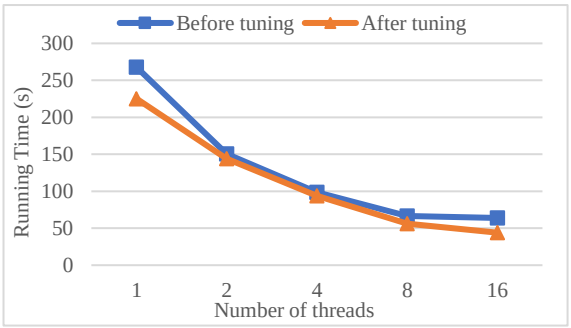
(g) n=457, num=9



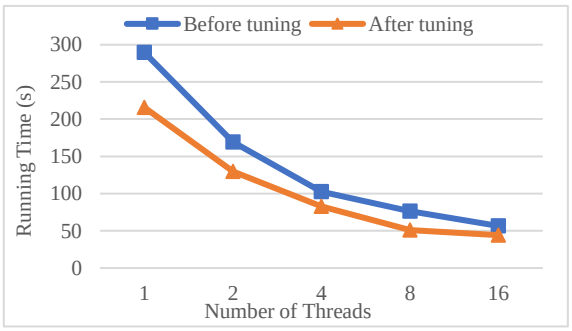
(h) n=457, num=11



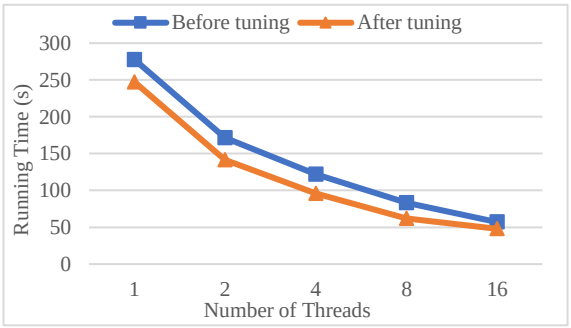
(i) n=457, num=12



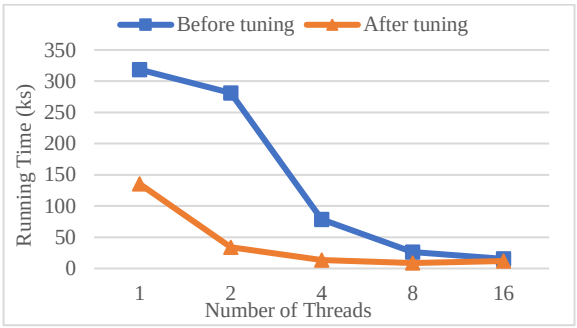
(j) n=457, num=13



(k) n=457, num=14



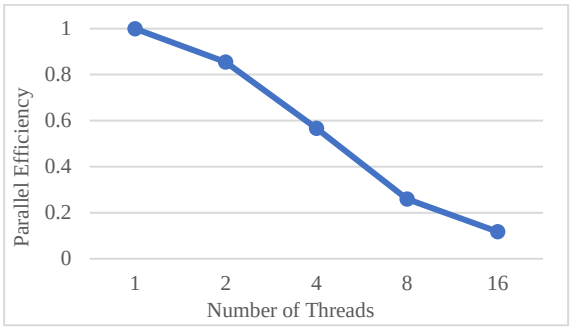
(l) n=457, num=15



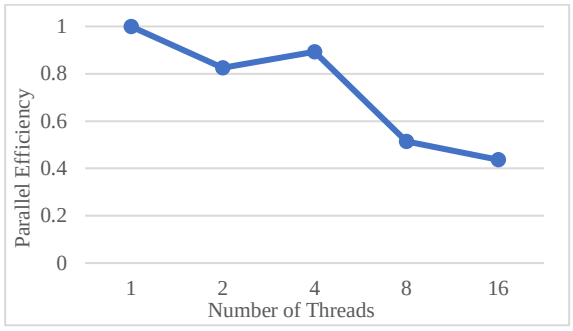
(m) n=1300, num=2

Figure 1. The model calculation time under different parallel scales before and after parameter tuning

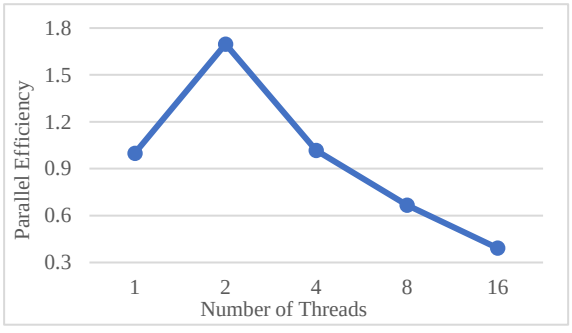
It can be seen from Figure 1 that the parameter tuning has marked effects on model calculation, and computing speed of the model is improved. Compared with running time before tuning, the running time after tuning is reduced by as much as 87.83%, which significantly shortened the solving time of the model. However, the performance of the branch-cut algorithm after parameter tuning is not stable under different parallel scales. For example, when n=1300 and num=2, the single-thread solving time is reduced by 57.48% compared with that before tuning, but when the number of threads is 2 and 4, the running time is reduced by up to 87.83% and 82.5%, and when 16 threads were used, the running time savings dropped to 22.59%. This is caused by the load balancing problem of the parallel branch-cut algorithm. When threads are solving subproblems, if the solution obtained by one thread is worse than the current candidate solution, then the subproblem that is solved by this thread will be pruned instantly and the thread will be idle; However, if the subproblem solved by another thread does not meet the pruning conditions, the subproblem will generate a series of subproblems, and this thread will continue to work for a long time. Load balancing problem results in the instability of the parallel branch-cut algorithm under the same parameter tuning settings.



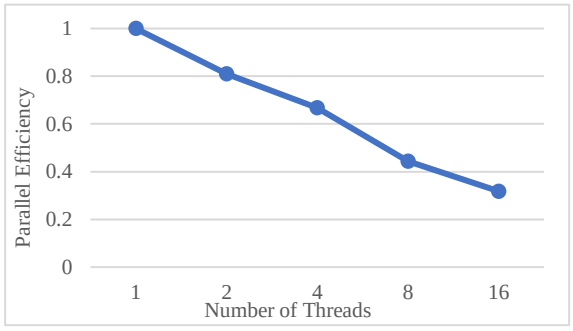
(a) n=457, num=2



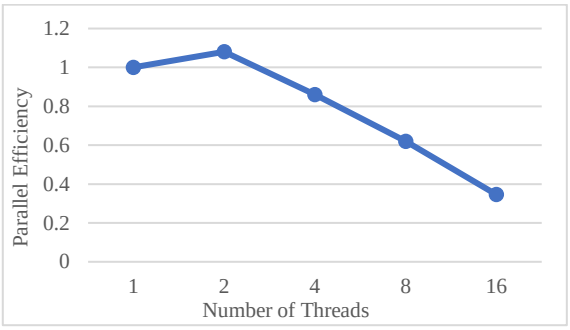
(b) n=457, num=3



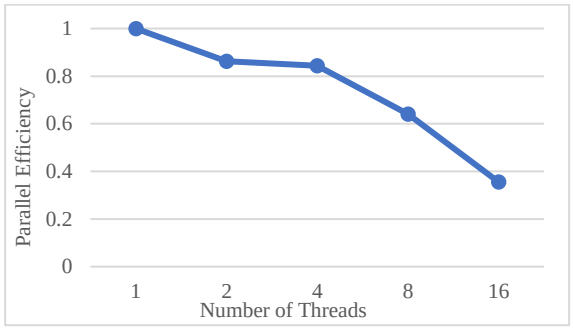
(c) n=457, num=4



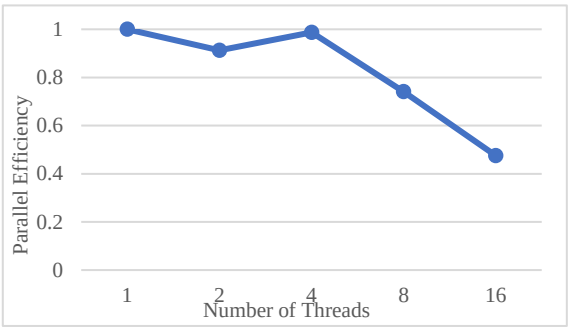
(d) n=457, num=6



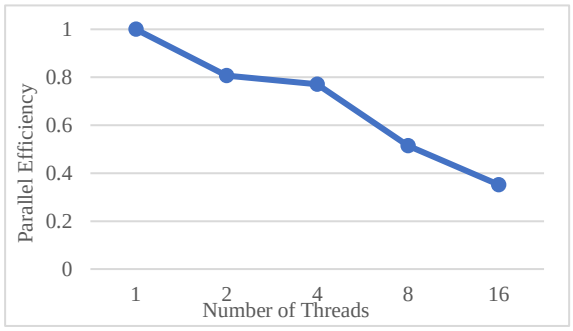
(e) n=457, num=7



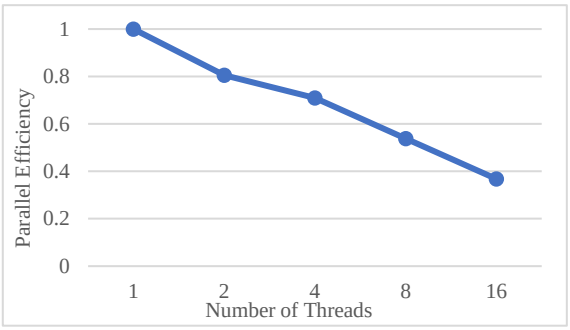
(f) n=457, num=8



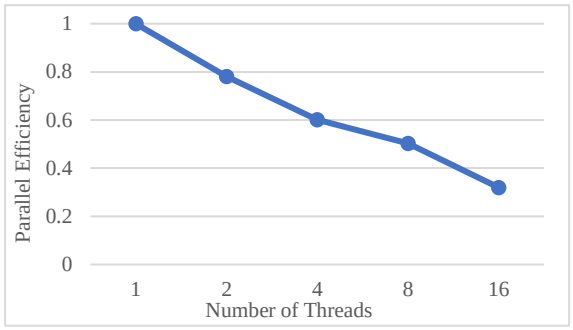
(g) n=457, num=9



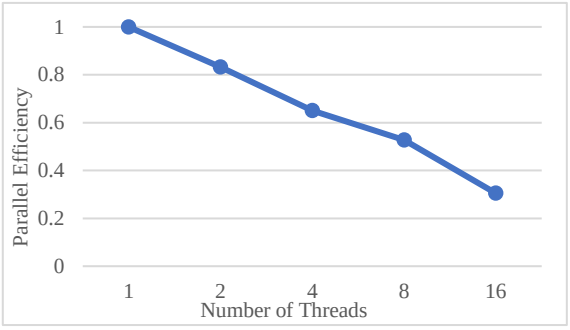
(h) n=457, num=11



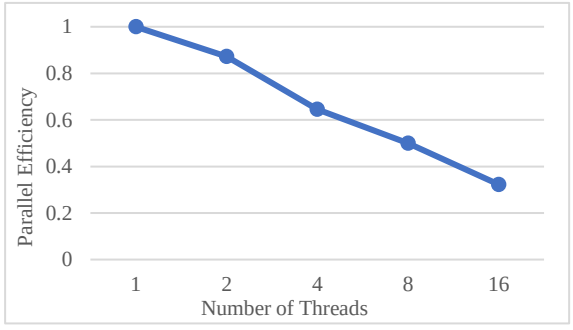
(i) n=457, num=12



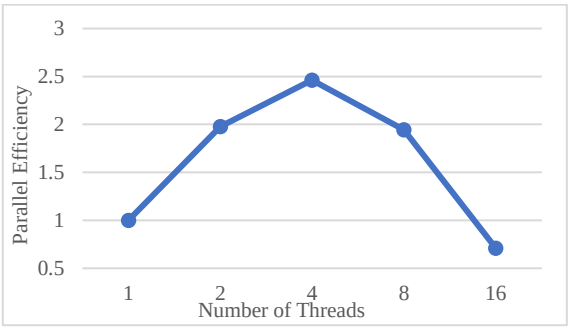
(j) n=457, num=13



(k) n=457, num=14



(l) n=457, num=15



(m) n=1300, num=2

Figure 2. The parallel efficiency of the Branch-Cut algorithm after parameters adjustment under different parallel scales

From Figure 2, we can see that the parallel efficiency declines with an increase of the number of threads. When the number of threads is 2 and 4, the parallel efficiencies are better and preponderate over 60%. When the number of threads increases further, the parallel efficiency begins to decrease substantially. This is because when the number of threads exceeds 4, the overhead of load balancing and communication exceed the benefits of pruning, making it difficult to improve the computational efficiency. It can also be seen from Figure 2 that "super linear" may appear when the branch-cut algorithm solves the model in parallel. For example, when the number of threads in Figure 2(c) is 2, the parallel efficiency reaches 169.8%. The reason is that the parallel branch and the bound algorithm in the parallel branch-cut algorithm is able to reduce the computational load of the model, but only in the case of reduced computational load is it considerable. However, when the advantage of the reduction of calculation is not obvious, the parallel efficiency will decrease significantly with an increase of the number of threads.

5. Conclusions

To solve the multi-conditional MAD model efficiently, we use the CPLEX solver deployed on an Era supercomputer to perform an auto-tuning of the branch-cut algorithm. The comparison experiments show that the parallel branch-cut algorithm after tuning reduces the time it takes to solve the multi-conditional MAD model significantly, and it can obtain comparable results with the algorithm before tuning. Experiments under different parallel scales demonstrate that the parallel efficiencies are better and preponderate over 60% when the number of threads is 2 or 4.

Acknowledgements

This work was partly supported by the National Key R&D Program of China (No. 2017YFB0203102), the State Key Program of National Natural Science Foundation of China (No. 91530324), the National Natural Science Foundation of China (No. 61272193) and the Youth Innovation Promotion Association, CAS (No. 2015375).

References

1. C. S. Adjiman, I. P. Androulakis, and C. A. Floudas, “Global Optimization of MINLP Problems in Process Synthesis and Design,” *Computers & Chemical Engineering*, vol. 21, no. 10, pp. 445-450, 1997
2. R. J. Allgor and P. I. Barton, “Mixed-Integer Dynamic Optimization,” *Computers & Chemical Engineering*, vol. 21, no. 21, pp. 451-456, 1997
3. P. Bonami, L. T. Biegler, A. R. Conn, et al. “An Algorithmic Framework for Convex Mixed Integer Nonlinear Programs,” *Discrete Optimization*, vol. 5, no. 2, pp. 186-204, 2008
4. R. J. Dakin, “A Tree-Search Algorithm for Mixed Integer Programming Problems,” *Computer Journal*, vol. 8, no. 3, pp. 250-255, 1965
5. M. A. Duran and I. E. Grossmann, “An Outer-Approximation Algorithm for a Class of Mixed-Integer Nonlinear Programs,” *Mathematical Programming*, vol. 36, no. 3, pp. 307-339, 1986
6. D. C. Faria and M. J. Bagajewicz, “A New Approach for Global Optimization of a Class of MINLP Problems with Applications to Water Management and Pooling Problems,” *Aiche Journal*, vol. 58, no. 8, pp. 2320-2335, 2012
7. M. R. Garey and D. S. Johnson, “Computers and Intractability: A Guide to the Theory of NP-Completeness,” W. H. Freeman, 1983
8. A. M. Geoffrion, “Generalized Benders Decomposition,” *Journal of Optimization Theory & Applications*, vol. 10, no. 4, pp. 237-260, 1972
9. D. Han, J. Jian, and L. Yang, “Outer Approximation and Outer-Inner Approximation Approaches for Unit Commitment

Problem," IEEE Transactions on Power Systems, vol. 29, no. 2, pp. 505-513, 2014
10. C. Li, Z. H. Lu, J. L. Hu, Y. H. Hu, and J. Wang, "Portfolio Optimization Problem Solving Under Concave Transaction Costs and Cardinality Constraints," *in Proceedings of CCF HPC China 2016*, pp. 739-742, Xi'an, China, October 2016 (In Chinese)
11. M. L. Luyben and C. A. Floudas, "Analysis the Interaction of Design and Control, Part 2: Reactor-Separator-Recycle Systems," *Computers & Chemical Engineering*, vol. 18, no. 10, pp. 971-993, 1994
12. Z. H. Shen, Y. Dong, and X. X. Wang, "Logistics Base Location Optimization Model Based on Mixed Integer Non-linear Program," *Logistics Sci-Tech*, vol. 36, no. 7, pp. 89-93, 2013 (In Chinese)
13. T. Westerlund and F. Pettersson, "An Extended Cutting Plane Method for Solving Convex MINLP Problems," *Computers & Chemical Engineering*, vol. 19, no. 1, pp. 131-136, 1995

Feng Yao received his M.S. degree in Xi'an Jiao Tong University, Xi'an, China, in 2006. Now he is a senior engineer of Henan electric power dispatching control center, State Grid HeNan Electric Power Company. His current research interest includes analysis and control technology of power system.

Yi Liu received his M.S. Degree from Zhengzhou University. Now, he is a Senior Engineer at the Henan Electric Power Dispatching Control Center, State Grid HeNan Electric Power Company. His current research interests include analysis and control technology of power systems and power forecasting technology of new energy.

Huifen Chen received her Ph.D. Degree from Tsinghua University. Now, she is a Senior Engineer at the Henan Electric Power Dispatching Control Center, State Grid HeNan Electric Power Company. Her current research interests include analysis and control technology of power systems and power forecasting technology of new energy.

Chen Li graduated from the School of Control and Computer Engineering, North China Electric Power University, with a Bachelor's Degree. Now she is a Ph.D. student of the Computer Network Information Center, Chinese Academy of Sciences. Her current research interests include high performance computing technology in the financial computing.

Zhonghua Lu received her Bachelor's Degree and Master's Degree from Shaanxi Normal University and her Ph.D. Degree from the Institute of Mathematics, Chinese Academy of Sciences. Now, she is a Professor of Computer Network Information Center at the Chinese Academic of Sciences. Her research interests include applications of mathematics and computation, biomathematics, parallel computing and high-performance computing technology in financial computing.

Jinggang Wang received his M.S. Degree from North China Electric Power University. Now, he is a Senior Engineer at the Henan Electric Power Dispatching Control Center, State Grid HeNan Electric Power Company. His current research interests include analysis technology of power system.

Zhiheng Li received his M.S. Degree from Huazhong University of Science and Technology. Now, he is a Senior Engineer at the Henan Electric Power Dispatching Control Center, State Grid HeNan Electric Power Company. His current research interests include operation analysis technology of power system.

Ningming Nie received her Bachelor's Degree from Xiangtan University, Xiangtan, China, in 2005, and her Ph.D. degree from the Academy of Mathematics and Systems Science, Chinese Academy of Science. Her research interests include high performance computing.