Fast Algorithms for Detecting Circular VAR Flows in Large Power-Flow Models

Yonghong Chen
Midwest ISO
Indianapolis IN

Xing Liu and Vaithianathan "Mani" Venkatasubramanian
School of EECS, Washington State University
Pullman WA 99164-2752

Abstract

Circular VAR flows are mainly caused by improper coordination of reactive power control devices such as transformers in the power system. Circular VAR flow results in wastage of reactive power resources, and could lead to higher line currents as well as unnecessary switching of reactive power controls. In large power system, locating the presence of circular VAR flows is a challenging task. By treating the reactive power-flow problem as a directed graph, two new algorithms are proposed in the paper for detection of circular VAR flows. The first algorithm extends the standard graph theory approach by finding all the strongly connected components of a graph. The second algorithm is based on certain simple yet powerful necessary conditions for circular VAR loops in power systems that are proposed in this paper. The two algorithms are shown to be extremely fast by detecting all circular VAR loops in two large-scale power-flow models.

Keywords*: Loop flows, Directed graphs, depth-first search, electric power system, power-flow studies.*

I. Introduction

Circular VAR flow corresponds to a circular flow of reactive power across several transmission line branches. Typically, circular VAR flows arise from improper coordination of parallel and/or neighboring transformer banks in the power system. The circular VAR flows whenever present result in wastage of reactive power resources while also leading to higher line currents and increased active power losses. In practice, transmission system operators routinely inspect the VAR flows on specific transformer banks and known transmission paths to prevent circular loop flows. If present, the problem is mitigated by better coordination of tap changer settings.

The circular VAR flow problem is in general quite complicated for realistic large power system models, because of the nonlinear nature of the power-flow problem. It is nearly impossible for operators to manually find all VAR loops present in the system, especially when the loops involve multiple transmission line branches.

In the doctoral dissertation [1], a methodology was presented for fast detection and elimination of circular VAR flows in the large power system. The algorithm presented in [1] is based on the depth-first-search (DFS) algorithm from graph theory [2]. The results from the dissertation [1] were cited and briefly discussed in [3]. The recent paper [4] also proposes graph theory based methods for detection of circular VAR flows using a different matrix based approach illustrated on small power-flow test cases.

In this paper, we propose *fast* algorithms for detection of circular VAR flows in realistic *large* power-flow models. Specifically, we propose two

1530-1605/08 $25.00 © 2008 IEEE

powerful new algorithms: the first one is an extension of the standard graph theory approach for finding strongly connected components and this algorithm was earlier presented in the doctoral dissertation [1] of the first author of this paper. The second algorithm is a new algorithm proposed here which uses certain simple necessary conditions for the existence of circular VAR flows in large-scale power-flow models. Both algorithms use DFS as the search engine and hence are compatible with sparsity savings.

The algorithms are illustrated on two realistic large scale power-flow cases from the Korean system (1,400 buses) and from the eastern American inter-connection (16,000 buses). The algorithms find a surprisingly large number of circular VAR loops (more than 1300 loops) in a standard eastern American planning power-flow case. They also take a fraction of a second to find them on a normal laptop computer. While most of these VAR loops in this example contain small VAR loop flows, a significant number of them (100+) have at least 10 MVAR of loop-flows in each of the loops which should not be ignored during operation. The fast algorithms provide a way to include the detection and control of circular VAR flows in any standard power-flow program.

The algorithms presented here for circular VAR flows can be easily extended for circular MW flows by formulating the directed graphs appropriately.

II. Definition and basic theory

Let us consider a set of buses say from bus 1 to bus n such that there exists a transmission line between any bus i and bus i+1. Suppose the reactive power-flows on each transmission line satisfy the conditions $Q_{i,i+1}>0$ and $Q_{i+1,i}<0$, then, we say that there is circular VAR flow or circulating VAR flow in the loop connecting the buses 1 through n.

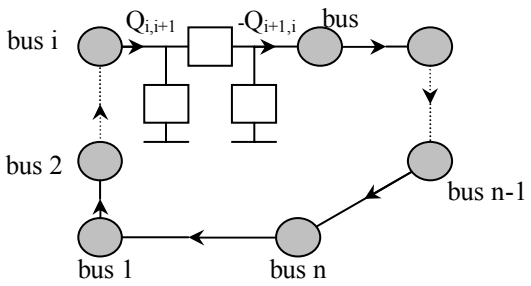

Figure 1. Illustration of a circular VAR flow

The smallest absolute value of the reactive power-flows $Q_{i,i+1}$ and $Q_{i+1,i}$ for i=1, n-1, is defined as the measure of the circular VAR flow for the entire loop connecting buses 1 to n. Our aim is to find the existence of any such circular VAR loop in a given power-flow model.

A. Necessary conditions for circular VAR flows

Necessary conditions for the existence of circular VAR flows in lossless lines were presented in [4]. In the real system, the transmission lines and transformers always have resistances so that conditions based on lossless line assumptions do not hold on many branches. In this paper, we present more simplistic conditions for the existence of circular VAR flows that will prove to be very useful in actual implementations later.

Let us consider the case of buses 1 to n introduced above in Figure 1 with the circular VAR flow as shown. Let us denote the bus voltage magnitude for bus i to be V_i as usual. In a typical power transmission line, it is well-known [5] that whenever $V_i>V_j$, the reactive power will flow from bus i to bus j, that is, $Q_{ij}>0$ and $Q_{ji}<0$. This is not always the case for many branches because of various reasons including asymmetry in admittance bus elements from presence of transformer banks. Moreover, in the actual power system, the voltage magnitudes of adjacent buses may

be nearly equal, which renders the assumptions in the inference on Q flows mentioned above to be invalid. Even with these qualifying remarks, the common rule-of-thumb, *"if $V_i > V_j$, then, the reactive power will flow from bus i to bus j, that is, $Q_{ij} > 0$ and $Q_{ji} < 0$"* holds on most of the transmission lines. That is, we normally assume that *the reactive power flows along the voltage gradient*. However, for the existence of the circular VAR loop, we need the following simple and powerful necessary condition which must directly contradict the heuristic rule above for at least one of the branches in the loop.

Necessary Condition: We assume without loss of generality that no two bus voltage magnitudes are ever equal. Then, in any circular VAR loop, there is at least one transmission line between say buses i and j, which satisfies $V_i > V_j$, and the reactive power flows from bus j to bus i, that is, $Q_{ij} < 0$ and $Q_{ji} > 0$.

Proof: The proof is by contradiction. Suppose the necessary condition is not satisfied at all the branches in a loop. By definition of the circular VAR loop in Figure 1, we must have that reactive power must be flowing from bus i to bus i+1 along every transmission branch in the loop for every i from 1 to n. Note that bus n+1 is in fact bus 1 being in a loop. Since the necessary condition is not applicable at each of the branches, the branch reactive power flows imply $V_1 > V_2 > V_3 > \ldots > V_{n-1} > V_n > V_1$. Clearly, we have a contradiction. ■

The necessary condition gives a powerful criterion for locating the existence of circular VAR flows. In any power-flow model, the branches which satisfy the necessary condition that the reactive power flows against the voltage gradient will be denoted *critical branches* in this paper since this kind of branch gives us an efficient way to detect the loops. We need to emphasize the fact that this is only a necessary condition and is not a sufficient condition for finding circular VAR loops. In other words, many of the critical branches may not result in any circular VAR flow, while a few critical branches may lead to more than VAR loops as well. Many of these critical branches typically have small reactive power-flows possibly resulting in small circular VAR flows which may not be significant. However, the branches with significant reactive power-flows across the voltage gradient (that satisfy the necessary condition) that are present in VAR loops are indeed the critical branches for managing the circular VAR flows in the large power system.

In Section IV, we present a new algorithm which speeds up and simplifies the circular VAR detection by locating the critical branches in the system first.

III. SCC-TREE Algorithm

In this section, we present the first algorithm for circular VAR detection by combining the theory of strongly connected components (SCC) from graph theory with a tree search algorithm for finding loops within strongly connected components. Let us consider a directed graph with a set of nodes (also called vertices in graph theory) V and a set of branches (also called edges in graph theory) E. A subset of nodes U of V is said to be strongly connected if there exists a directed path from node i to node j for any two nodes i and j in U.

Circular loop (such as the circular VAR loops above) is an example of a strongly connected component, though a strongly connected component may contain multiple circular loops or no loops at all (trivial case of isolated nodes). In the following graph in Figure 2 from [2], there are three strongly connected components, namely {a, b, e}, {c, d}, {f, g}. Details can be seen in [2].

For the power-flow model, we will assume that each bus i denotes a node i. The direction of Q flow from say bus i to bus j denotes the directed branch in the graph from node i to node j if $Q_{ij} > 0$ *and* $Q_{ji} < 0$. If the reactive power is either flowing into the line from

the two ends, or if it is flowing out of the line at both ends, there is *no* corresponding branch in the graph owing to the inconsistency of VAR flow directions at the two ends. In the following text, we will refer to buses, vertices and nodes interchangeably. Similarly, the names, transmission line, branch and edge all denote the same component in a graph.

A. Circular VAR flow in the graph

As a first step, we represent the power-flow model as the directed graph say G=(V, E) [1]. V is the set of all vertices or buses or nodes in the system and E is the set of all directed branches or edges connecting the nodes in V.

If there is any loop in the directed graph, the loop corresponds to a circular VAR flow for the power-flow case, and the minimum reactive power-flow among the branches in the loop is called the circular VAR flow for this loop as defined earlier. The total circular VAR flow for the system is defined as the summation of the magnitudes of all the individual loop circular VAR flows.

Assuming G is a directed graph, G^T is the transpose of G (All the line directions in G are opposite to those in G^T). The following linear-time (i.e., time proportional to number of buses plus number of branches) algorithm from [2] computes the strongly connected components of a directed graph G=(V, E) using two depth-first searches (DFS), one on G and one on G^T. First, we introduce the DFS algorithm from graph theory [2].

B. Depth First Search algorithm

The strategy followed by depth-first search is to search "deeper" in the graph whenever possible [2]. In depth-first search, edges are explored out of the most recently discovered vertex *v* that still has unexplored edges leaving it. When all of *v*'s edges have been explored, the search "back-tracks" to explore edges leaving the vertex from which *v* was discovered. This process continues until we have discovered all the vertices that are reachable from the original source vertex. If any undiscovered vertices remain, then one of them is selected as a new source and the search is repeated from that source. This entire process is repeated until all vertices are discovered.

Whenever a vertex v is discovered during a scan of the adjacency list of an already discovered vertex u, depth-first search records this event by setting $v's$ predecessor field $\pi[v]$ to *u*. The procedure DFS below records when it discovers vertex *u* in the variable $d[u]$ and when it finishes vertex *u* in the variable $f[u]$. These time-stamps are integers between 1 and twice of the number of vertices. For every vertex *u*, we have $d[u] < f[u]$. Vertex *u* is WHITE before time $d[u]$, GRAY between time $d[u]$ and time $f[u]$, and BLACK thereafter. The following pseudo-code from [2] is the basic depth-first-search algorithm. The variable time is a global variable used for time stamping. The details of the algorithm can be seen in [2].

```
DFS ( G )
1   for each vertex u ∈ V [ G ]
2       do color [ u ] ← WHITE
3          π [ u ] ← NIL
4   time ← 0
5   for each vertex u ∈ V [ G ]
6       do if color [ u ] = WHITE
7          then DFS – VISIT ( u )
```

```
DFS – VISIT (u)
1  color [u] ← GRAY
2  d [u] ← time ← time + 1
3  for each v ∈ Adj [u]
4     do if color [v] = WHITE
5        then π [v] ← u
6           DFS – VISIT (v)
7  color [u] ← BLACK
8  f [u] ← time ← time + 1
```

Strongly-Connected-Components (G) [2]

1. *Call DFS(G) to computer finishing times f[u] for each vertex u*
2. *Computer G^T*
3. *Call DFS(G^T), but in the main loop of DFS, consider the vertices in order of decreasing f[u] (as computed in line 1)*
4. *Output the vertices of each tree in the depth-first forest of step 3 as a separate strongly connected component.*

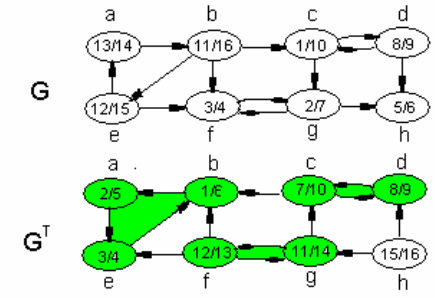

Figure 2. An example of computing strongly connected components using DFS from [2]

Figure 2 shows the process of the two depth-first searches. For each node *u*, the starting and finishing time are shown as $d[u]/f[u]$. In [2], it is proved that the above algorithm correctly computes the strongly connected components of a directed graph G. The computational burden for the algorithm is proved to be linearly proportional to the number of buses plus the number of branches in the graph. This linear computational complexity renders the algorithm useful for large-scale power-flow models. Once the strongly connected components have been computed, all the sub-loops or connected multiple loops can be identified by a simple search step presented in the next subsection that only deals with a few branches of the strongly connected component.

C. TREE algorithm

As we know from graph theory, a SCC may consist of many loops or may be trivial (an isolated node). For finding all the loops present in a nontrivial SCC, we need to develop new algorithms. The basic idea is still by applying DFS to the SCC. From the top of a tree to its bottom, if any of branches has the path that can go back to the top, it is a loop. Since every SCC would be a part of the whole graph and relatively smaller, this method would not take much time to find every loop.

Let us assume there are n SCC in G=(V,E) and SCC[i] is the set of all vertices that belong to the i^{th} SCC, i=1,2,…,n. The pseudo-code of TREE algorithm is given below:

TREE(SCC)

1. *# of tree i← 1.*
2. *# of loop j← 0.*
3. *m← 1*
4. *For each $u_1 \in SCC[m]$*
5. *FID_LOOP(u_1)*
6. *m← m+1*
7. *go to 4 if $m \le n$*

FIND_LOOP(u)

1. *do if v is bottom of tree*
2. *theni←i+1*
3. *tree[i]← u*
4. *for each $v \in Adj[u]$ and $v \in SCC[m]$, tree[i] ← v*
5. *do if v has existed in tree[i]*
6. *then go to 2*
7. *do if $v=u_1$*
8. *then j←j+1,record loop[j] ← tree[i]*
9. *FIND_LOOP(v)*

In fact, TREE algorithm presented above is an exhaustive search. But it won't miss any loops. Although most of circular VAR flows are tiny and may be uninteresting, this algorithm can have the ability to find all the loops. In this context, it provides a powerful algorithm for finding *all* the loops in the graph. This completes the discussion of SCC-TREE algorithm. We present examples later in Section V.

IV. CB (critical branch) Algorithm

In this section, we assume that we have constructed the directed graph for the power-flow model, and that we have also identified the critical branches or edges as defined in Section II. This algorithm is a new contribution of this paper, and provides a powerful technique for associating specific critical branches (related transformer banks) with circular VAR flows that exist in a large-scale power-flow model. While the SCC-TREE algorithm in Section IV is by exhaustive search of the whole power network, the CB algorithm presented in this section is useful for identifying all the circular VAR loops that may be associated with specific network branches (and related transformer bank controls). The algorithm can be used for controlling and eliminating circular VAR flows. These control algorithms for coordination of transformer bank settings to eliminate circular VAR flows will be discussed in a future publication. This paper presents the algorithms for finding the circular VAR flows which will serve as the computational engines for the control coordination algorithms later.

Normally, it is necessary to start from every node and every branch to search the whole graph. This way is absolute exhausted one and can guarantee any loop won't be lost. On the other hand, this method will spend plenty of time to complete this task. By the definition of critical branches and the necessary conditions for circular VAR flows, we know that we don't need to think about whole nodes and branches in the system. In our purposed approach, we exploit the necessary condition of Section II: each loop must contain at least one critical branch. Therefore, if we find each critical branch, we will be able to detect all the loops in the graph. Obviously there is no exception that a circular VAR flow doesn't include any critical branch. So the CB algorithm can also guarantee that all loops can be found.

Based on this advantage, we don't find SCC at all and don't need to search all vertices in graph. We can start TREE algorithm only from every critical branch by doing some modifications in the algorithm used in SCC-TREE algorithm since now we face the whole graph. The algorithm proceeds as follows.

Let us consider a critical branch say from node i to node j. Starting from node j, carry out the TREE search according to the algorithm in Section III, until we exhaust all the deeper nodes and return to node j. During the TREE search, every time that we encounter a branch that starts from say bus k and ends in bus i, the path from bus j to bus k (from TREE) together with branch from bus k to bus i, and the branch from bus i to bus j, constitute a loop. We record the loop and continue along the TREE search. When we finish the entire TREE search for bus j, we move to the next critical branch. When all critical branches have been analyzed, we have located all the loops in the graph.

Some of the loops may involve more than one critical branch and hence, they may be discovered several times during the progress of this algorithm. Since the speed of the DFS algorithm is proportional to the size of V plus the size of E, and this new algorithm repeats a complete DFS (in the worst case) for each critical branch, the computational complexity of the new algorithm in this section will be proportional to the sum of the number of buses and the number of buses times the number of critical branches.

In some cases, the CB algorithm may turn out to be an exhaustive search like the SCC-TREE algorithm of Section III. For simplicity, the SCC-TREE algorithm can be considered a "top-down" approach,

while the CB algorithm is more like a "bottom-up" approach.

At first glance, the new algorithm appears much slower than that of Section III. In reality, the DFS associated with one bus j will be much shorter and faster than the DFS for all the buses as in Section III. Therefore, we will see that the algorithm in this section IV can be slower or faster than the algorithm in Section III, depending on the number of critical branches in the graph. Also, the second algorithm provides an independent method for computing the loops. Consistency of results between the two algorithms is very helpful to verify in an on-line computational environment.

V. Simulations on large power-flow models

A. Korean test system

The first case is a Korea power system case. The whole system will be checked for the presence of any circular VAR flow loops in this system. This case comes from an actual planning model with no alterations made for this study. This case includes more than 1,400 buses and more than 2,600 branches. Out of the 2,600 branches, 433 branches satisfy the necessary condition of Section II and qualify as critical branches.

Both algorithms in Sections III and IV find three circular VAR loops discussed below.

Table 1: Circular VAR flow loop #1

Bus1	Bus2	IDs	V1(pu)	V2(pu)	Q12(MVAR)
3400	3411	1	1.02925	1.00188	-54.23
3411	3400	2	1.00188	1.02925	-70.35

In Table 1, we find Q_{flow} = 54.23 MVAR for the circular VAR loop. This circular VAR flow exists in two buses which are connected with each other by two parallel transformers. Different tap settings in the parallel transformers results in the circular VAR flow which is a well-known problem. For transformer 1, the tap ratio is 1.0375 pu, while the tap ratio of transformer 2 is 1.0125pu. From bus 3400 to bus 3411, $V_1 > V_2$ but $Q_{12} < 0$. Hence, this branch is the critical branch in the notation of Section IV.

In Table 2, details of circular VAR flow loop #2 are shown and in the same time, it is numerical proof for the claim on loss and asymmetrical branches since all the branches in this loop are transmission lines that are symmetrical.

Table 2. Circular VAR flow loop #2

Bus1	Bus2	IDs	V1(pu)	V2(pu)	Q12(MVar)
1565	1535	1	0.99661	1.0015	-1.023
1565	1535	2	0.99661	1.0015	-1.023
1535	1510	1	1.0015	1.00179	-1.724
1535	1510	2	1.0015	1.00179	-1.724
1510	1545	1	1.00179	0.99796	-1.228
1510	1545	2	1.00179	0.99796	-1.228
1545	1590	1	0.99796	0.99565	-1.85
1545	1590	2	0.99796	0.99565	-1.85
1590	1565	1	0.99565	0.99661	-6.682
1590	1565	2	0.99565	0.99661	-6.682

For loop #2, the circular VAR loop flow, the minimum absolute value of the Q12 column is 1.023 MVAR. For this loop, interestingly, there is no transformer or any other asymmetrical branch in the entire loop. In the same way, notice that from 1510 and 1545, $V_1 > V_2$ but $Q_{12} < 0$ and $Q_{21} = 0.98 > 0$. This is a critical branch.

Table 3. Circular VAR flow loop #3

Bus1	Bus2	IDs	V1(pu)	V2(pu)	Q12(MVar)
4400	4410	1	1.02183	1.00113	44.48
4400	4410	2	1.02183	1.00113	43.75
4400	4410	3	1.02183	1.00113	-84.37
4410	4400	4	1.00113	1.02183	-39.05
4410	4400	5	1.00113	1.02183	-38.4
4410	4400	6	1.00113	1.02183	90.43

The circular VAR flow for this loop is denoted as 38.4 MVAR from Table 3.

The simulations were carried out on a "regular" laptop with 1.6 GHz Pentium processor and 1 GB of memory. In both cases, we assume that the data has been read from a power-flow solved case and the branch Q flows have been computed for each branch from the bus data. The file reading and Q flow computations were carried out using simple code and the two tasks take about 0.30 seconds for the 1,400 bus Korean test data. The computational times for detecting the loops for two algorithms are 0.01 seconds and 0.06 seconds for the algorithms from Sections III and IV respectively.

B. Eastern inter-tie power-flow data

The second case is a summer peak case for the eastern system. It is quite large system that includes more than about 16,000 buses and more than 26,000 branches. This case also comes from a real planning model and has not been modified in any way. Naturally, this system is so complicated that we can encounter all kinds of loop flows. Since there are many circular VAR flows, many of which are small, we only show the details of some major loop flows below.

There are 22,909 branches with consistent Q flows at the two ends and there are 22,909 directed branches in the VAR flow graph. For this power-flow model, we find a rather large number 1,367 circular VAR flow loops. Table 4 summarizes the MVAR flow values in the loops. While most of the loops (1,156) have circular VAR flows of less than 5 MVAR and may be ignored, there is still a significant number (211) loops with a total circular MVAR flow of about 4,000 MVAR in them that is leading to wastage of reactive power in the system. This also leads to unnecessary line currents that result in higher MW losses and more fatigue on power equipment.

Table 4. Summary of Circular VAR flows

Circular VAR flow range (MVAR)	# of VAR loops	Sum of VAR flows (MVAR)
⩾1	1040	9421.899
⩾5	236	6094.1
⩾10	134	4868.63
⩾20	50	3057.29
⩾30	26	2263.421

The next four tables 5 through 8 summarize the four major VAR loops in the power-flow case.

Table 5: Circular VAR flow loop #1

Bus1	Bus2	IDs	Q12 (MVar)
32441	32289	1	-247.73
32289	32441	2	-217.89

This loop has Q_{flow} = 217.89 MVAR.

Table 6: Circular VAR flow loop #2

Bus1	Bus2	IDs	Q12 (MVar)
63404	63581	1	-184.77
63581	63404	2	-632.58

The circular VAR flow for the loop is Q_{flow} = 184.77 MVAR.

Table 7: Circular VAR flow loop #3

Bus1	Bus2	IDs	Q12 (MVar)
77950	77405	1	-681.18
77405	77950	2	-157.58

The loop VAR flow is Q_{flow} = 157.58 MVAR.

Table 8: Circular VAR flow loop #4

Bus1	Bus2	IDs	Q12 (MVar)
37551	36382	1	-122.02
36382	37551	2	-102.10
36382	37551	3	-103.14

The circular VAR flow is Q_{flow} = 102.10 MVAR.

The circular VAR loops in each of the above four worst cases is caused by incorrect ratios of parallel transformers between two buses. It is a well-known fact that operators need to be careful while coordinating the taps of parallel transformers. Also, the first four circular VAR flow loops in this power-flow case themselves lead to more than 660

MVAR of reactive resource wastage which must be avoided. In addition to these four loops, there are many more circular VAR flows in the power-flow case which consist of multiple transmission lines, as summarized in Table 4. The two algorithms presented in the paper provide efficient methods to compute all such flows in a real-time computational environment.

We next discuss the computational time taken by the two algorithms for this 16,000 bus eastern inter-tie power-flow model. These times are the actual times taken by the processor to compute all the MVAR loop flows from test simulations on a "regular" laptop as mentioned earlier.

In actual systems, there may not be any need to compute the circular VAR loops with small insignificant loop flows. Accordingly, the two algorithms can be made faster by pre-specifying an acceptable lower threshold for detecting loop flows in the power-flow case. In Table 9, the lower bound is specified in the first column, and the results along each row show the number of loops and the times taken for that lower bound value. It is easily seen that the algorithms are quite fast even while computing all the loops, and they become increasingly fast as the number of branches in the graph becomes smaller.

Table 9: Comparison of computational times in one case

Circular VAR flow lower bound	# of loops	# of critical branches	SCC-TREE algorithm (seconds)	CB algorithm (seconds)
1	1040	4673	0.48	5.91
5	236	2938	0.3	1.27
10	134	1912	0.02	0.81
20	50	1013	0.02	0.44
30	26	672	0.02	0.29

Table 9 also illustrates the claim in Section IV that the time taken by the new algorithm of Section IV is proportional to the number of critical branches. When the lower Q flow bound is increased to 30 MVAR, there are still 672 critical branches in the eastern system data. The computation for the new algorithm is much faster at only 0.29 seconds compared to the value of 5.91 seconds for the first row of Table 9 with 4673 critical branches. However, for the eastern system, SCC-TREE algorithm is faster than CB algorithm but the time taken by CB algorithm is still acceptable. Another point we should notice is that the time consumed by algorithms highly depends on the status of the computer. That is, even a small unrelated event can affect the computing process and can then result in some error in the computational time. The results on computational time reported in Table 9 are found by averaging the time taken over a few repeated runs in a typical 1.6 GHz Pentium laptop as noted earlier.

Based on these test results, it appears that the CB algorithm may be comparable in speed or be faster than the standard algorithm of Section III only when there are a small number of critical branches in the system. For further understanding of the time consumed by both algorithms, another four eastern inter-tie power-flow cases are tested (lower bound is 10Mvar and 50Mvar). Both algorithms are extremely fast in finding all the loops, while the SCC-TREE algorithm is consistently the faster of the two algorithms.

Table 10: Comparison of computational times

(10 Mvar lower bound used)

# of case	# of loops	SCC-TREE algorithm (seconds)	CB algorithm (seconds)
1	140	0.03	0.75
2	140	0.03	0.75
3	143	0.02	0.72
4	145	0.03	0.72

Table 11: Comparison of computational times
(50 Mvar lower bound used)

# of case	# of loops	SCC-TREE algorithm (seconds)	CB algorithm (seconds)
1	11	0.02	0.15
2	11	0.02	0.15
3	12	0.04	0.16
4	13	0.04	0.15

VI. Future work

If there are many circular VAR flows in a real power system, it would be useful to know of what control actions (such as coordination of transformer banks and/or switching capacitor/reactor banks) need to be taken to eliminate them. Efficient optimization methods can be formulated and applied to prioritize the candidate controls. Resent research reported in the doctoral dissertation [1] that resulted in a prototype implementation at Bonneville Power Administration (briefly discussed in [3]) presents one such approach.

The algorithms presented in this paper are useful for finding out whether and where a power system has problems with circular VAR flows. When circular VAR flows exist and are located, some available and feasible control devices can be analyzed around these loops. For example, many circular VAR flows are caused by inconsistent tap settings of parallel (or nearby) transformers. Adjusting the taps of one of the parallel (or nearby) transformers can delete the loop from the system. Such control problems are easily facilitated by the VAR flow detection algorithms of this paper.

VII. Conclusions

In this paper, a simple powerful necessary condition for the existence of circular VAR flows is proposed. Two new algorithms are proposed for computation of circular VAR loop flows in large-scale power system models. Test results on realistic large scale power-flow cases show that the algorithms are extremely fast even for very large power systems. The methods provide the means to compute and correct circular VAR (and possibly MW) flows in on-line computational environments for the real power system.

VIII. Acknowledgements

The authors thank Power System Engineering Research Center, Consortium for Rapid Technology Solutions, Entergy, and Tennessee Valley Authority for their support of the research reported in this paper.

IX. References

[1] Yonghong Chen, “Development of Automatic Slow Voltage Control for Large Power Systems”, *Ph.D. dissertation*, Washington State University, 2001.

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, *Introduction to Algorithms*, MIT press, 1994.

[3] K. Tomsovic, D.E. Bakken, V. Venkatasubramanian, and A. Bose, “Designing the next generation of real-time control, communication, and computations for large power systems”, *Proceedings of the IEEE,* Volume 93, Issue 5, May 2005 pp. 965 – 979.

[4] P. Wei, Y. Ni, F. Wu, “Load flow tracing in power systems with circulating power”, *International Journal of Electrical Power & Energy Systems*, Vol. 24, Issue 10, December 2002.

[5] J. J. Grainger, and W. D. Stevenson, *Power System Analysis*, McGraw-Hill, Inc., 1994.