No. 734

TALLINNA TEHNIKAÜLIKOOLI TOIMETISED
TRANSACTIONS OF TALLINN TECHNICAL UNIVERSITY

UDC 681.3.06

Leo Võhandu, Rein Kuusik

CLIQUES AND ALGORITHMS WITH HIDDEN PARALLELISM

Abstract

The problem of hidden parallelism of algorithms based on theory of monotone systems is discussed. A new fast algorithm for separation of all maximal cliques on finite non-oriented graph is described. An example of algorithm's work is presented.

1. Introduction

Parallel algorithms are in focus today, but there remains much to be done with sequential algorithms. One of the new possibilities belongs to the class of a so-called hidden parallelism. Everybody knows hopefully that the efficiency of generic algorithms is grounded on that notion [1]. Acting sequentially, we work in reality in parallel with 3^K schemes.

Are there any other principles of the same power as reproduction and crossing-over, still not mutilating original objects? The answer is yes. In our paper we describe a new possibility to create algorithms with hidden parallelism for NP-hard graph problems. As its foundation, we use the theory of monotone systems developed at our Department [1] mostly for data analysis and hypothesis generation. Usually the theory of graphs is used in data analysis, now we turn the tables. As an example, we will handle in this paper the problem of finding all maximal cliques in a graph without any repetitions. Our algorithm is polynomial in clique number.

[1] The short edition of the scheme already appeared in the Proceedings of Tallinn Tech. Univ., in 1971 publication, see note 2 below. Unfortunately, unknown to the author seminal publication from 1971, scientific community already occupied the name "Monotone System", which was later given to the scheme in 1975. Thus, the reader should not be confused with applications of well-known dynamic monotone systems, especially in biological models, random sampling, chemical reaction networks, etc., in addition to by "Monotone System" name utilized in "Reliability Theory".

2. The Monotone System

Let us have a finite set E and a mapping $P: E \times 2^E \to R$, satisfying the following condition:

$$x \in A \subseteq B \subseteq E \to P(x,A) \leq P(x,B) \tag{1}$$

The condition (1) is a condition of monotonicity and the couple (E,P), where P is a mapping satisfying that condition, is called a monotone on the set E.

The theory of monotone systems has been developed mostly in the articles [2-6]. [2] Its use has been very efficient in different fields of data analysis and AI. [3] Here we represent the minimal amount of information needed about monotone systems.

With every monotone system (E,P) one can connect another map $F: 2^E \to R$, which is defined by $F(A) = \min P(x,A)$ if $x \in A$. This mapping F is a minimal function of a given monotone system.

A kernel of the monotone system is a subset $W \subset E$ on which the minimum function of the monotone system obtains its maximum. In general, a monotone system can have many kernels, but all its kernels belong to the unique maximal kernel.

For our goals it is very important that this maximal kernel can be found with the help of a greedy algorithm.

We will give the description of that algorithm's idea. First, we find $e_1 \in E$, so that $a_1 = P(e_1,E) = F(E)$. After that we find $e_2 \in E \setminus \{e_1\}$, so that $a_2 = P(e_2, E \setminus \{e_1\}) = F(E \setminus \{e_1\})$.

---

[2] Seminal article J. E. Mullat, "On a certain maximum principle for certain set-valued functions," Tr. Tallin. Politech. Inst., Ser. A, No. 313, 37-44 (1971), http://www.datalaundering.com/download/modular.pdf .

[3] **A**rtificial **I**ntelligence

Continuing in this way, we can order the entire set $E : E = \{e_1, e_2, ..., e_n\}$ and we also get at the same time a sequence of real numbers $a_1, a_2, ..., a_n$.

It is known that if a_k is maximal in this number sequence, then the subset $\{a_k, ..., a_n\}$ is the kernel of the given monotone system, and all kernels of that system are given in the same way [2].

Such a freedom in the mapping choice makes this structuring very powerful. To analyze graph structures, we have efficiently used as a standard mapping a so-called frequency mapping [7].

3. Graphs, Frequency Maps and Ordering of Vertices

For handling a clique problem, we need to analyze only finite symmetric graphs $G = (V, E)$, where V is a set of graph vertices and E is a set of edges of the graph. One can represent the graph as an adjacency matrix $A = (a_{ij})$, where $a_{ij} = a_{ji} = 1$ if the vertices i and j are connected, 0 otherwise.

To define the frequency mapping, we use for a while the adjacency matrix A as a $n \times n$ data matrix. For every column we count the data table frequencies of zeros and ones $\{f_{0j}, f_{1j}\}$, where j is the column number (i.e., vertex number). For every row (vertex) of this data table we will define as a weight the row values sum w_i of frequencies $f(a_{ij})$ in table A: $w_i = \sum f(a_{ij})$.

It is easy to prove that with this weight function the vertices of a graph build up a monotone system. (Excluding of a vertex and its connections from a graph will only diminish the weights of other vertices).

As an elementary example, we take a simple graph.

A	1	2	3	4	5	6	w_i					
1	1	1	1	0	0	0	20*	-	-	-	-	-
2	1	1	1	1	1	0	24	20	16*	-	-	-
3	1	1	1	0	1	1	24	20	18	14	10*	-
4	0	1	0	1	1	0	20	18*	-	-	-	-
5	0	1	1	1	1	1	24	22	18	14*	-	-
6	0	0	1	0	1	1	20	18	16	14	10	6*
0	3	1	1	3	1	3						
1	3	5	5	3	5	3						

The left side of the scheme contains the adjacency matrix, the column frequencies are indicated below, and on the right side of the first column (w_i) there are the weights vertices. The (corrected) weights with asterisks show the chosen minimal elements. (In the case of equal values we take the element with the last biggest drop in its value history. If there is no history, or its histories are the same, we just take the first object.)

Ordering the vertices of graphs in elimination order, we get

A	1	4	2	5	3	6
1	1	0	1	0	1	0
4	0	1	1	1	0	0
2	1	1	1	1	1	0
5	0	1	1	1	1	1
3	1	0	1	1	1	1
6	0	0	0	1	1	1

Using this order, one can easily write out the maximal cliques of G: $\{5,3,6\}$, $\{2,5,3\}$, $\{4,2,5\}$, and also $\{1,2,3\}$.

This ordering algorithm is not the quickest possible, and it needs reordering of the adjacency matrix, i.e., special scanning, to write out all maximal cliques. For smaller graphs it is good enough, especially taking into account the common need for a nice structural representation of graphs.

4. A Quick Algorithm for Maximal Cliques

To separate all maximal cliques of a graph (V,E), we use its adjacency matrix $X(I)$ (index I shows the number of its generation, $X(0)$ corresponds to G). Here we use as the weight of vertex its degree in the subgraph $G(I)$ corresponding to the adjacency matrix $X(I)$. The weights (degrees) of the vertices will be saved in the vector $F(I)$. The vertices belonging to the clique in the generation I are saved in the vector $CLIQUE(I)$. In the generation I we only analyze vertices with the non-zero weights. The vector FP contains the frequencies of admissible vertices.

Algorithm

P1. **Initializing**. $I = 0$, $CLIQUE(0) = \{\}$, $FP\ FP = \{\}$, $CONTROL = \{\}$.

P2. **Calculating weights** of vertices $F(0)$. Set $F(0)$ to $CONTROL$.

P3. **Checking for a clique**. If all weights of the vertices to be analyzed are equal to their number $-1[\mathrm{N(I)}-1]$ THEN

BEGIN a clique is found. Output all vertices as a clique. Set all weights in $F(I) = 0$ END

P4. **Control for back-tracking**. If all weights in $F(I)$ are zeros THEN

BEGIN $I \leftarrow I - 1$ IF $I = -1$ GOTO FINISH ELSE GOTO P3 END

P5. **Selection of the maximal weight**.
$CLIQUE(I+1) \leftarrow CLIQUE(I)$. IF $I = 0$ THEN
BEGIN
IF all weights in $F(0)$ are zeros and minus ones THEN
BEGIN set $CONTROL$ to $F(0)$, set free for further analysis the submatrix of $X(0)$ defined by vertices with the non-zero weights in $CONTROL$, GOTO P3 END.
END
Find the vertex J with the maximal weight in $F(I)$ and add it to the $CLIQUE(I+1)$.

P6. **Elimination and recalculation**. Eliminate the vertex J and recalculate weights in $F(I)$ (set zero to the weight (J) and diminish its neighbours weights by one).
IF $I = 1$ THEN recalculate weights in $CONTROL$.

P7. **Formation of a new generation**. $I \leftarrow I + 1$. Exclude the submatrix $X(I)$ corresponding to the vertex J. (All admissible neighbours of the vertex J belong to the matrix $X(I)$ plus its neighbours minus one weight in $F(I-1)$).
Calculate the weights for $F(I)$ and FP. (For $F(I)$ we take into account connections of the original adjacency matrix, for FP the admissible connections of $X(I-1)$ only.)
Checking for a clique. IF the weight of some vertex K in $F(I)$ is zero, we have found a clique. Output $CLIQUE(I)$ and the vertex K.

P8. Elimination of generations. Ban in the matrix $X(I-1)$ the non-zero admissible elements of the matrix $X(I)$ and those of the matrix J with its neighbours $X(I)$. Recalculate weights in $F(I-1)$, that is – subtract FP from $F(I-1)$.
Set free all elements in $X(I)$ for further analysis.
IF $I = 1$ THEN
BEGIN
IF the non-zero weights of vertices in $CONTROL$ are equal to their weights in $F(1)$ THEN assign zero to their weights in $CONTROL$ and recalculate all vertices weights in $CONTROL$ connected with those vertices in $X(1)$.
IF for a vertex P in $F(I-1)$ weight $(P) = 0$ and $CONTROL(P) \neq 0$ THEN set weight $(P) = -1$ in $F(I-1)$.
END

P9. **Minimization of the number of generations**. IF in $F(I)$ there exist vertices p with the non-zero weights which are one less that the number of vertices being analysed in the current generation $[N(I)-1]$, add them to $CLIQUE(I)$, make their weights in $F(I)$ to zero and recalculate weights of their neighbours.
Checking for a clique. IF the number of vertices in $\{P\}$ is equal to the number of vertices being analysed in the generation (I), THEN
BEGIN a clique has been found. Output $CLIQUE(I)$, GOTO P4 END
IF the recalculated weight of some neighbours of $\{P\}$ K is zero, THEN
BEGIN a clique has been found. Output $CLIQUE(I)$ and K END

P10. GOTO P3
FINISH

5. Example

Let us have a non-oriented graph $G(6,9)$ with an adjacency matrix $X(0)$:

$X(0)$	1	2	3	4	5	6
1	0	1	1	0	0	0
2	1	0	1	1	1	0
3	1	1	0	1	0	1
4	0	1	1	0	1	1
5	0	1	0	1	0	0
6	0	0	1	1	0	0
$F(0)$	2	4	4	4	2	2
CONTROL	2	4	4	4	2	2

In the vector $F(0)$ we have the degrees of vertices (Step P2). Now we take $CLIQUE(1) \leftarrow CLIQUE(0)$ and find the vertex (or the first of them) with the maximal degree. In our case $J = 2$. So we add the vertex 2 to the $CLIQUE(1)$ (step P5). In step P6 we eliminate that vertex by making its weight in $F(0)$ equal to zero, diminish its neighbours weights by one and get $F(0)$: $1\,0\,3\,3\,1\,2$.

As $I = 1$, we do the same with $CONTROL$: $1\,0\,3\,3\,1\,2$.

Now we go to the next generation by step P7 ($I \leftarrow I + 1$) and create a submatrix $X(1)$ corresponding to vertex 2. As all neighbours of vertex 2 are admissible, we form $X(1)$ from vertices $1, 3, 4, 5$ and calculate the new weights vectors $F(1)$ and FP:

$X(1)$	1	3	4	5
1	0	1	0	0
3	1	0	1	0
4	0	1	0	1
5	0	0	1	0
$F(1)$	1	2	2	1
FP	1	2	2	1

Further we ban the non-zero elements of $X(1)$ and those of vertex 2 in $X(0)$ (step P8):

$X(0)$	1	2	3	4	5	6
1	0	-	-	0	0	0
2	-	0	-	-	-	0
3	-	-	0	-	0	1
4	0	-	-	0	-	1
5	0	-	0	-	0	0
6	0	0	1	1	0	0

We then recalculate weights in $F(0)$: $(F(0)-FP)$

$F(0)$	1	0	3	3	1	2
$-FP$	1		2	2	1	
New $F(0)$	0	0	1	1	0	2

As $I=1$, we now compare the weights of vertices in $CONTROL$ and $F(1)$ (stepP8). We see that vertices 1 and 5 have equal weights in both vectors. We assign zero to them in $CONTROL$ and diminish accordingly the weights in $CONTROL: 0\,0\,2\,2\,0\,2$ (vertex 1 is connected with vertex 3, vertex 5 with vertex 4 in matrix $X(1)$. As we do not have elements in $X(1)$ minus one, we go to step P4.

We now put $CLIQUE(2) \leftarrow CLIQUE(1)$ (step P%) and find the vertex with the maximal degree in $F(1)$. In our case $J=3$. We put vertex 3 into $CLIQUE(2)$, eliminate it by zeroing its weight in $F(1)$ and diminish neighbours weights by one (step P6):

$$F(1): \quad 0 \quad 0 \quad 1 \quad 1\,.$$

At step P7 we go to the next generation ($I \leftarrow I+1$), create a submatrix $X(2)$ corresponding to vertex 3 and calculate new weight vectors $F(2)$ and FP:

$X(2)$	1	4
1	0	0
4	0	0
$F(2)$	0	0
FP	0	0

As weights of vertices 1 and 4 are zeros, we can output new cliques $\{2,3,1\}$, $\{2,3,4\}$ (step P7).

As $FP=0$, we have only to ban elements of vertex 3 in $X(1)$ (elements $(1,3)$, $(4,3)$, $(3,1)$, $(3,4)$ (step P8):

$X(1)$	1	3	4	5
1	0	-	0	0
3	-	0	-	0
4	0	-	0	-1
5	0	0	1	0

And do not have to recalculate weights in $F(1)$.

As all weights in $F(2)$ are equal to zero, we now backtrack to step P4 ($I \leftarrow I-1$) and analyze the vector $F(1):0011$ (step P3).

The zeros in $F(1)$ indicate that these vertices are already eliminated from the analysis. We can now use as a short-cut a well-known fact that in a graph having a clique with K vertices there are at least K degrees no less that $K-1$. From $F(1)$ we can easily see that there is a clique $\{2,3,4\}$ in $X(1)$ (step P3).

Further eliminate vertices 4 and 5 by making their weights to zero. As $F(1)=0$, we backtrack to step P3 ($I \leftarrow I-1$) and analyze the vector $F(0):001102$.

We now take $CLIQUE(1) \leftarrow CLIQUE(0)$ and find the vertex with the maximal degree. In our case $J=6$. So we add vertex 6 to $CLIQUE(1)$ (step P5). In step P6 we eliminate that vertex by making its weigh in $F(0)$ equal to zero, diminish its neighbours weights by one and get $F(0):000000$.

As $I=1$, we recalculate $CONTROL:001100$.

We then form a new generation ($I \leftarrow I+1$) (step P7) and create a submatrix $X(1)$ corresponding to vertex 6. As all neighbours of vertex 6 are admissible, we form $X(1)$ from vertices $3, 4$ and calculate the new weight vector $F(1)$. (In $X(1)$ the sign minus denotes an existing but non-admissible connection between vertices.)

$X(1)$	3	4
1	0	-
4	-	0
$F(1)$	1	1
FP	0	0

($FP = 0$, because the elements $(3,4)$ and $(4,3)$ in the matrix $X(1)$ are non-admissible.)

Step P8. We ban all admissible elements of $X(1)$ and also the elements $(6,3), (6,4), (3,6), (4,6)$ in the matrix $X(0)$.

$X(0)$	1	2	3	4	5	6
1	0	-	-	0	0	0
2	-	0	-	-	-	0
3	-	-	0	-	0	-
4	0	-	-	0	-	-
5	0	-	0	-	0	0
6	0	0	-	-	0	0

We do not have to recalculate weights in $F(0)$ as $FP = 0$ in $X(1)$.

As $I = 1$, we next compare vertices weights in $CONTROL$ and $F(1)$. We see that vertices 3 and 4 have equal weights in both vectors. We assign zero to them in $CONTROL$. Since they do not have neighbours with a non-zero weight in $CONTROL$, there is nothing to diminish. New weights in $CONTROL$ are: $0\,0\,0\,0\,0\,0$.

Step P9. We analyze vector $F(1):11$.

Elimination of those vertices gives us vector $F(1)=0$. Using again our clique-degree short cut (two vertices with the weight 1), we see that there is a clique $\{6,3,4\}$.

Since all weights in $F(1)$ are now equal to zero, we backtrack to step $P4$ ($I \leftarrow I-1$) and analyze vector $F(0)$. $F(0)=0$ means that all maximal cliques have been found.

FINISH

References

1. Goldberg, D. *Generic Algorithms in Search. Optimization and Machine Learning*. Addison-Wecley, 1989.
2. Mullat, J. 'Extremal Subsystems of Monotonic Systems.' *Automation and Remote Control*, 1976, No. 5, pp. 130-139**,** **http://www.datalaundering.com/download/extrem01.pdf**; No. 8, pp. 169-178**,** **http://www.datalaundering.com/download/extrem02.pdf** .
3. Mullat, J., Vyhandu, L. 'Monotonic Systems in Scene Analysis.' Symposium. *Mathematical Processing of Crtographic Data*. Tallinn, 1979, pp. 63-66, **http://www.datalaundering.com/cardgraf.pdf** .
4. Vyhandu, L. 'Fast Methods in Exploratory Data Analysis.' *Trans. of Tallinn Tech. Univ*., 1989, No. 705, pp. 3-13.
5. Kuusik, R. 'Application of Theory of Monotonic Systems for Decision Trees Generation.' *Trans. of Tallinn Tech. Univ*., 1989, No. 705, pp. 47-58.
6. Kuusik, R. 'Hypotheses Generator for Qualitative Data.' Trans. of Tallinn Tech. Univ., 1987, No. 645, pp. 141-148 (in Russian).
7. Vyhandu, L. 'Express Methods of Data Analysis.' *Trans. of Tallinn Tech. Univ*., 1979, No. 464, pp. 21-35.

UDK 681.3.06

Leo Võhandu, Rein Kuusik

Klikid ja varjatud paralleelsusega algorithmid

Kokkuvõte

Artiklis küsitletakse nn. "varjatud paralleelsust" monotoonsete süsteemide teorial basseeruvates algoritmides. Esitatakse kiire algoritm kõikide maksimaalsete klikkide leidmiseks orienteeimata graafis. Algoritmi töö on illustreeritud näitega.