ARTICLE IN PRESS

Available online at www.sciencedirect.com

ScienceDirect

Discrete Applied Mathematics ▮▮▮ (▮▮▮▮) ▮▮▮–▮▮▮

DISCRETE APPLIED MATHEMATICS

www.elsevier.com/locate/dam

The general σ all-ones problem for trees☆

Xueliang Li[a], Chao Wang[b], Xiaoyan Zhang[c]

[a] *Center for Combinatorics and LPMC–TJKLC, Nankai University, Tianjin 300071, PR China*
[b] *College of Software, Nankai University, Tianjin 300071, PR China*
[c] *School of Mathematics and Computer Science, Institute of Mathematics, Nanjing Normal University, Nanjing 210097, PR China*

Received 13 March 2006; received in revised form 30 April 2007; accepted 26 August 2007

Abstract

The general σ all-ones problem is defined as follows: Given a graph $G = (V, E)$, where V and E denote the vertex-set and the edge-set of G, respectively. Denote C as an initial subset of V. The problem is to find a subset X of V such that for every vertex v of $G\backslash C$ the number of vertices in X adjacent to v is odd while for every vertex v of C the number is even. X is called a solution to the problem. When $C = \emptyset$, this problem is the so-called σ all-ones problem. If a vertex is viewed to be adjacent to itself, the σ all-ones problem is addressed as the σ^+ all-ones problem. The σ^+ all-ones problem has been studied extensively. However, the σ all-ones problem has received much less attention. Unlike the σ^+ all-ones problem, which has solutions for any graphs, the σ all-ones problem may not have solutions for many graphs, even for some very simple graphs like C_3 and P_5. So, it becomes an interesting question to find polynomial time algorithms to determine if for a given tree the problem has solutions. And if it does, to find a solution to the minimum σ all-ones problem. In this paper we present two algorithms of linear time to solve the general σ all-ones problem for trees. The first one is good for counting the number of solutions if solutions do exist, and the second one is good for solving the minimum σ all-ones problem. Furthermore, we can modify the algorithm slightly to solve the general minimum σ all-ones problem.

MSC: 05C05; 05C70; 05C85; 68R10; 68Q25; 90C27

Keywords: (minimum) (general) σ all-ones problem; Tree; Algorithm; Linear time

1. Introduction

A cellular automaton is a discrete dynamical system that consists of an arrangement of basic components called cells together with a transition rule. In this paper, we study cellular automaton based on the σ-rule on graphs. In what follows we always assume that $G = (V, E)$ is a finite connected simple undirected graph. Each vertex (cell) assumes one of the two states, viz. 0 or 1. An assignment of states to the vertices will be called a *configuration*. The behavior of the automaton is determined by a local transition rule: the state of vertex (cell) v at time $t + 1$ depends only on the states of the vertices (cells) in the neighborhood of v at time t. We consider the local rule σ of a very simple algebraic nature: states are elements over GF(2) and the next state of a vertex (cell) v is the sum of the states of all neighboring vertices (cells). Analogously, for the σ^+-rule, the state of a vertex v at time $t + 1$ is the sum of the states at time t of

☆ Research supported by NSFC, PCSIRT, the "973" program and Jiangsu Planned Projects for Postdoctoral Research Funds (0602023C).
E-mail addresses: lxl@nankai.edu.cn (X. Li), wangchao@nankai.edu.cn (C. Wang), xiaoyanice@yahoo.com.cn (X. Zhang).

0166-218X/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2007.08.042

the neighbors as well as the state of the vertex v itself. The local rule is applied in parallel everywhere to obtain the global rule operating on configurations. If X is a configuration at time t, Y is the configuration at time $t+1$ by σ- or σ^+-rule, then Y is called the successor of X (with respect to σ or σ^+), and conversely, X is called a predecessor of Y. One of the basic problems in the study of the evolution of configurations is to determine whether a given configuration has a predecessor. This problem is referred as the *predecessor existence problem* (PEP) [10]. For known results on the complexity of this problem, see [14,16]. As is shown in [14], PEP is naturally in NP for finite cellular automata. From [10] we know that PEP is solvable in polynomial time over GF(2) for rules σ and σ^+. In [10] the author also showed that a modified version of PEP is NP-hard where the predecessor configuration is required to have the minimum number of cells with non-zero states. This problem is called the *bounded predecessor existence problem* (BPEP). This problem remains NP-complete even if the target configuration is fixed to be $\mathbf{1}$, the vector with all components equal to 1.

For the σ^+-rule, when the target configuration is fixed to be $\mathbf{1}$, the corresponding problem is also called *all-ones problem* [11]. An equivalent version of the all-ones problem was proposed by Peled in [9], where it was called the lamp lighting problem. The all-ones problem has been extensively studied recently, see Sutner [13,15], Barua and Ramakrishnan [1] and Dodis and Winkler [3]. Using linear algebra, Sutner [12] proved that the all-ones problem always has a solution for any graph, and gave some results on counting the number of predecessors for $n \times n$ grid graphs. Lossers [7] gave another beautiful proof also by using linear algebra. A graph-theoretic proof was given by Eriksson et al. [5]. Galvin [6] gave a linear time graph-theoretic algorithm to find solutions for trees. In [10], Sutner proved that the minimum all-ones problem is NP-complete [8] in general. In [2] we gave an algorithm of linear time to solve the minimum all-ones problem for a tree, and counted the number of solutions to the all-ones problem for a given tree.

For the σ-rule, when the target configuration is fixed to be $\mathbf{1}$, we call the corresponding problem the σ *all-ones problem*. Moreover, when the target configuration is fixed to be an arbitrary configuration $\mathbf{1}-\mathbf{b}$, the corresponding problem is called *general* σ *all-ones problem*, where $\mathbf{b}$ is an arbitrary configuration. The σ all-ones problem is a special case of the general σ all-ones problem for which $\mathbf{b}$ is taken to be $\mathbf{0}$, where $\mathbf{0}$ is a vector with all components equal to 0. Unlike the σ^+-rule, it is not always true that the σ all-ones problem has solutions for all graphs. For example, if we take G as a triangle or a path on five vertices, then the σ all-ones problem does not have any solution for this G. On the other hand, from the Odd Set problem in [4] we can easily show that the BPEP for the σ-rule with the target configuration $\mathbf{1}$ remains NP-complete even if G is bipartite. So, it becomes an interesting question to find polynomial time algorithms for trees, a special kind of bipartite graphs. Actually, in this paper we find a decision algorithm of linear time to determine whether a solution exists for the general σ all-ones problem for a tree, and from this algorithm we can deduce a counting algorithm which gives a simple formula to count the number of solutions. We also obtain another linear time algorithm, and from this algorithm we can find a solution with minimal number of cells of non-zero state in a tree if solutions do exist, i.e., give a solution to the minimum general σ all-ones problem.

In other words, the general σ all-ones problem is try to find a configuration $l : V \longrightarrow \mathrm{GF}(2)$, such that

$$\sum_{v \in N(u)} l(v) + b_v = 1, \tag{1}$$

holds for every $u \in V$, where $N(u)$ is the neighborhood of the vertex u and b_v is 1 or 0 as given. In what follows, if $W = \{v_1, \ldots, v_s\}$, then $N(W)$ is defined to be the set $N(v_1) \cup \cdots \cup N(v_s)$, as usual. We call (1) the state equation of the vertex u. If there exists such a function l, then for $u \in V$ we say that $l(u)$ is the truth value of u, and $\{v \in V : l(v) = 1\}$ is a solution to the general σ all-ones problem. For $u \in V$, if we can get $l(u)$ from the state equations of some vertices, we say that u can be assigned a truth value, or $l(u)$ can be determined. In fact, we can formulate the general σ all-ones problem by algebraic language: to find solutions for the equation system

$$A_G X + \mathbf{b} = \mathbf{1},$$

where A_G denotes the adjacency matrix of the graph G and $\mathbf{b}$ is a vector with component b_v equal to 1 or 0 as given. So, one can employ the Gaussian elimination method to see if there is a solution. However, we hope to have a more intuitive way to judge if a solution exists. In other words, we would like to work directly on the given graphs. Moreover, the Gaussian elimination method does not help in solving the minimum general σ all-ones problem.

Please cite this article as: X. Li, et al., The general σ all-ones problem for trees, Disc. Appl. Math. (2007), doi: 10.1016/j.dam.2007.08.042

2. The decision algorithm and the counting of solutions

In this section, we give a decision algorithm of linear time to determine whether there exists a solution to the general σ all-ones problem for a given rooted tree T. Moreover, from this algorithm we can count the number of solutions if they do exist. First, we introduce some notation as follows:

$$D = \{v \in V(T) : l(v) \text{ has been determined}\},$$

$$U = \left\{ u \in V(T) : \text{ for all } v \in N(u), l(v) \text{ has been determined, and } \sum_{v \in N(u)} l(v) + b_u = 1 \right\},$$

$$L = \{v \in V(T) : d_v = 1 \text{ where } d_v \text{ is the degree of } v \text{ in } T\},$$

$$F = N(L).$$

The main idea of this decision algorithm is simple. By the special structure of trees, some vertices can be assigned a truth value first. For example, the truth value of every vertex in F can be determined in order to make the state equation of each vertex in L hold. Thus, we have $D_1 = F$ and $U = L$ first. Note that here we mention L instead of the set of leaves because the root may belong to L if its degree is 1, i.e., it has only one child.

In this section we will give an inductive algorithm. We can determine the truth values of some new vertices from the state equations of the vertices in $N(D_1)\backslash U$ by using the truth values of the vertices in D_1. We get D_2 by adding the new vertices into D_1. At the same time, the state equations of more vertices hold than before. Hence, U is changed to be a larger set. Keep this operation until no new vertices are produced. In the process, if the state equation of some vertex cannot hold, the algorithm stops and outputs NO. Otherwise, the algorithm outputs YES. The correctness of this algorithm lies in Theorem 2.2.

The Decision Algorithm for the General σ All-ones Problem for Trees.

Input: A rooted tree T with n vertices $V = \{v_1, \ldots, v_n\}$ and a (0, 1)-vector $\mathbf{b} = (b_{v_1}, b_{v_2}, \ldots, b_{v_n})$.

Step 0: For each $v \in L$, assign the truth value $1 - b_v$ to the only vertex $v \in F$ which is adjacent to v, i.e., set $l(w) = 1 - b_v$. If there are several vertices $v_1, v_2, \ldots, v_k$ in L which has the same adjacent vertex $w \in F$ and $b_{v_1}, b_{v_2}, \ldots, b_{v_k}$ are not the same, then we cannot assign the truth value to w, and there is no solution. Exit and output NO.

Step 1: Set $D_0 = \emptyset$, $D_1 = F$, $U = L$. For each $v \in V(T)\backslash U$, set $e(v) = d_v$ and $A(v) = N(v)$. $e(v)$ will be the number of all the undetermined vertices adjacent to v and $A(v)$ will be the set of all the undetermined vertices adjacent to v in the following steps. Set $f(v) = 1 - b_v$ and $i = 1$.

Step 2: If there are new vertices that are determined, i.e., $D_i\backslash D_{i-1} \neq \emptyset$, for each new vertex $w \in D_i\backslash D_{i-1}$, consider all its adjacent vertices whose state equations have not been satisfied, i.e., consider each vertex $u \in N(w)\backslash U$.

Case 1: $e(u) \geqslant 3$, i.e., there are more than three undetermined vertices adjacent to u. Then delete w from $A(u)$, and decrease $e(w)$ by 1, because$w$ has been determined. In addition, decrease $f(u)$ by $l(w)$.

Case 2: $e(u) = 2$, i.e., $A(u) = \{w, v_0\}$. If $l(v_0)$ has not been determined then assign $f(u) - l(w)$ to $l(v_0)$, and add u and v_0 to U and D_i, respectively. Otherwise, consider whether $l(v_0)$ is equal to $f(u) - l(w)$. If $l(v_0) = f(u) - l(w)$, add u to U. Else, there is no solution. Exit and output NO.

Step 3: Set $i = i + 1$. If $D_i\backslash D_{i-1} \neq \emptyset$, go to Step 2. Otherwise, go to Step 4.

Step 4: Output YES, D and U.

At the end of the decision algorithm, if $D = V$, the truth value of every vertex has been determined, which means that the solution is unique. Otherwise, $D \neq V$ and $U \neq V$, i.e., the truth value of every vertex in $V\backslash D$ has not been determined and the state equation of every vertex in $V\backslash U$ is unsatisfied. We say that the state equation (1) of the vertex u is unsatisfied if the truth values of at least two vertices in $N(u)$ have not been determined. Furthermore, the reason why these equations are unsatisfied is that the truth value of every vertex in $V\backslash D$ has not been determined.

Please cite this article as: X. Li, et al., The general σ all-ones problem for trees, Disc. Appl. Math. (2007), doi: 10.1016/j.dam.2007.08.042

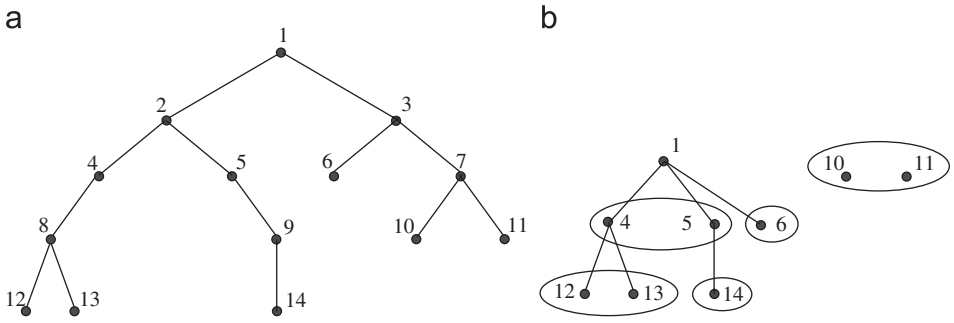

Fig. 1. (a) A given rooted tree with vertex set $\{1, 2, \ldots, 14\}$. (b) Constructed forest $\mathscr{F}$.

We can consider $l(v)$ for $v \in V\backslash D$ as variables to be determined and $\sum_{v\in N(u)} l(v) + b_v = 1$ for $u \in V\backslash U$ as restricted equation system of these variables. Corresponding to the restricted equation system, we can construct a forest $\mathscr{F}$ containing group structures. $\mathscr{F}$ can be used to count the number of solutions. The construction method is as follows.

Initially, set $\mathscr{F} = \emptyset$. For all $u \in V\backslash U$ whose state equation is (1), suppose $N(u) \cap (V\backslash D) = \{v_1, \ldots, v_k\}$, $k \geqslant 2$. By our decision algorithm, we know that $v_1, \ldots, v_k$ are brother vertices, or some vertex v_i is the grandfather vertex of the other vertices in the original rooted tree T. For the former case, we put $v_1, \ldots, v_k$ into one group B. For the latter case, we put $v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_k$ into one group B and connect v_i to every vertex in B, which makes v_i the parent vertex of every vertex in B. Then we call v_i the parent vertex of B and B the child group of v_i. For any one of the above two cases, we add B into $\mathscr{F}$. After considering all of restricted equations, we get the forest $\mathscr{F}$. It is easy to see that the total number of groups in $\mathscr{F}$, denoted by $B(\mathscr{F})$, is $|V\backslash U|$. Next we state the method as the following algorithm by using the concept of layers of the original tree T.

Denote the root of T as t. For all $u \in V(T)$, the minimum distance between t and u is denoted by $d(t, u)$. Denote $g(u)$ as the grandfather vertex of u. Set

$$\bar{B}_T(u) = \{v \in T : v \text{ and } u \text{ have the same parent vertex in } T\}.$$

Note that $u \in \bar{B}_T(u)$.

The Construction Algorithm for $\mathscr{F}$.

Input: A rooted tree T, and the set D obtained by executing our decision algorithm.

Step 0: If $D = V$, there exists a unique solution, exit. Else, suppose $V\backslash D = C_{j_1} \cup C_{j_2} \cup \cdots \cup C_{j_s}$, where $C_{j_i} = \{v \in V\backslash D : d(t, v) = j_i\}$, $1 \leqslant i \leqslant s$, $j_1 < \cdots < j_s$. Note that j_1 may be equal to 0. Set $\mathscr{F} = \emptyset$.

Step 1: Do the following operations until $C_{j_1} = \emptyset$.
Choose $v \in C_{j_1}$. If $\bar{B}_T(v) \cap C_{j_1} = \{v\}$, then add v to $\mathscr{F}$ and delete v from C_{j_1}. Otherwise, let $B = \bar{B}_T(v) \cap C_{j_1}$, add B to $\mathscr{F}$ and delete all vertices in B from C_{j_1}.

Step 2: For $k = 2$ to s, do the following operations until $C_{j_k} = \emptyset$.
Choose $v \in C_{j_k}$. Set $B = \bar{B}_T(v) \cap C_{j_k}$, then add B to $\mathscr{F}$ and delete all vertices in B from C_{j_k}. If $g(v)$ belongs to $C_{j_1} \cup \cdots \cup C_{j_{k-1}}$, then connect $g(v)$ with all the vertices in B to make $g(v)$ the parent vertex of B.

Step3: Output $\mathscr{F}$.

After $\mathscr{F}$ has been constructed, we give the definition of layers in $\mathscr{F}$. All the roots of the trees in $\mathscr{F}$ can be considered in the top layer or the first layer of $\mathscr{F}$ because they have no parent in $\mathscr{F}$. And all the children vertices of those in the top layer can be considered in the second layer of $\mathscr{F}$, etc.

Example 2.1. For a given rooted tree in Fig. 1(a) with $\mathbf{b} = (0, 0, \ldots, 0)$, by executing our decision algorithm, we get that $D = \{2, 3, 7, 8, 9\}$, $U = \{1, 4, 5, 6, 10, 11, 12, 13, 14\}$, and $l(3) = l(7) = l(8) = l(9) = 1$, $l(2) = 0$. The equation

Please cite this article as: X. Li, et al., The general σ all-ones problem for trees, Disc. Appl. Math. (2007), doi: 10.1016/j.dam.2007.08.042

system consisting of all the unsatisfied state equations is

$$\begin{cases} l(1)+l(4)+l(5)=1, \\ l(1)+l(6)+1=1, \\ 1+l(10)+l(11)=1, \\ l(4)+l(12)+l(13)=1, \\ l(5)+l(14)=1. \end{cases}$$

The constructed forest $\mathscr{F}$ is in Fig. 1(b).

The purpose of constructing the forest $\mathscr{F}$ is to count the number of solutions to the general σ all-ones problem. The counting method is given as follows.

The counting method

For each vertex, we set a boolean variable Judge-State and an integer variable Num. Let the Judge-State of each leaf of $\mathscr{F}$ be YES, and that of any other vertex be NO. Set 1 to the Num of each vertex.

If the Judge-State of every vertex in some group B with parent vertex f is YES, then let $\mathrm{Num}(f) = \mathrm{Num}(f) * 2^{|B|-1}\prod_{v\in B}\mathrm{Num}(v)$, and delete B from $\mathscr{F}$. Once some vertex of state NO does not have children, change its state into YES. Go on until there are only m isolated vertices $v_1, \ldots, v_m$ and s groups $B_1, \ldots, B_s$ left in the top layer. Let

$$N = 2^{m+\sum_{i=1}^{s}|B_i|-s}\prod_{i=1}^{m}\mathrm{Num}(v_i)\prod_{i=1}^{s}\prod_{v\in B_i}\mathrm{Num}(v). \tag{2}$$

Then we claim that N is the number of solutions to the general σ all-ones problem for the tree T.

Definition 2.1. For any $v \in V(\mathscr{F})$, denote $Q(v) = \{w \in V(\mathscr{F}) :$ there exists a sequence of vertices $v = v_0, v_1, \ldots, v_k = w$ of $V(\mathscr{F})$ such that v_i is the child of v_{i-1} in $\mathscr{F}$, $1 \leqslant i \leqslant k, k \geqslant 0\}$ as the set of all descendent vertices of v including itself in $\mathscr{F}$.

Definition 2.2. For any $w \in V(\mathscr{F})$, denote $EQ(w) = \{\sum_{v\in N(u)} l(v) + b_u = 1 : u \in V\backslash U$ and $(V\backslash D)\cap N(u) \subseteq Q(w)\}$ as the equation system corresponding to $Q(w)$. Note that all the vertices corresponding to the variables in $EQ(w)$ are the descendent vertices of w in $\mathscr{F}$.

Definition 2.3. Suppose $v \in V(\mathscr{F})$. If v does not have children, then we say that the Num of v has been determined. If v has children, we say that the Num of v has been determined by using our counting method from the bottom layer up to delete groups, successively, until v does not have children and its Num has been changed.

Lemma 2.1. *If the* Num *of* $u \in V(\mathscr{F})$ *has been determined, then* Num(u) *is the number of solutions to the equation system* $EQ(u)$ *when* u *has been assigned a truth value.*

Proof. By induction on the number of layers. Suppose u has k child groups $B_1, \ldots, B_k$ which has vertices $v_{11}, \ldots, v_{1m_1}$; $v_{21}, \ldots, v_{2m_2}$; $\ldots$; $v_{k1}, \ldots, v_{km_k}$, respectively. Assume that the Num of v_{ij}, $1 \leqslant i \leqslant k$, $1 \leqslant j \leqslant m_i$ has been determined, then the equation system $EQ(u)\backslash\left(\bigcup_{\substack{1\leqslant i\leqslant k\\ 1\leqslant j\leqslant m_i}} EQ(v_{ij})\right)$ is

$$\begin{cases} l(u)+l(v_{11})+\cdots+l(v_{1m_1})=a_1, \\ l(u)+l(v_{21})+\cdots+l(v_{2m_2})=a_2, \\ \vdots \\ l(u)+l(v_{k1})+\cdots+l(v_{km_k})=a_k, \end{cases}$$

where $a_i \in \{0, 1\}$, $1 \leqslant i \leqslant k$, has been determined by the original restricted equation system. It is easy to see that there are 2^{m_i-1} different ways to assign truth values to $v_{i1}, \ldots, v_{im_i}$ when u has been assigned a truth value, $i \in \{1, 2, \ldots, k\}$. By the induction hypothesis, when u has been assigned a truth value, the number of solutions to $EQ(v_{ij})$

is $\mathrm{Num}(v_{ij})$. So, the number of solutions to the equation system $EQ(u)$ is $\prod_{1\leqslant i\leqslant k}\left(2^{m_i-1}\prod_{1\leqslant j\leqslant m_i}\mathrm{Num}(v_{ij})\right)=\prod_{1\leqslant i\leqslant k}\left(2^{|B_i|-1}\prod_{v\in B_i}\mathrm{Num}(v)\right)$, which is exactly the Num of u calculated by using our counting method. The proof is complete. □

Lemma 2.2. *The final result obtained by our counting method is equal to the number of solutions to the equation system consisting of all the unsatisfied state equations.*

Proof. Suppose only m isolated vertices $v_1,\ldots,v_m$ and s groups $B_1,\ldots,B_s$ in the top layer of $\mathscr{F}$ are left after deleting groups by using our counting method, successively. By Lemma 2.1, the Num of each vertex v in $\{v_1,\ldots,v_m\}$ and $B_1,\ldots,B_s$ is equal to the number of solutions to $EQ(v)$. After deleting the equation system $\left(\bigcup_{i=1}^{m}EQ(v_i)\right)\cup\left(\bigcup_{i=1}^{s}\bigcup_{v\in B_i}EQ(v)\right)$ from the equation system of all the unsatisfied equations, the remaining equation system is

$$\sum_{v\in B_i}l(v)=a_i,\quad 1\leqslant i\leqslant s. \tag{3}$$

It is easy to see that for $w\in\{v_1,\ldots,v_m\}$, $l(w)$ can be 1 or 0 arbitrarily. Meanwhile, there are totally $2^{|B_i|-1}$ different ways to assign the truth values to the vertices of B_i under the restriction of (3), $1\leqslant i\leqslant s$. Hence, the total number of solutions to the entire equation system is

$$\begin{aligned}N&=2^m\prod_{i=1}^{s}2^{|B_i|-1}\prod_{i=1}^{m}\mathrm{Num}(v_i)\prod_{i=1}^{s}\prod_{v\in B_i}\mathrm{Num}(v)\\&=2^{m+\sum_{i=1}^{s}|B_i|-s}\prod_{i=1}^{m}\mathrm{Num}(v_i)\prod_{i=1}^{s}\prod_{v\in B_i}\mathrm{Num}(v),\end{aligned}$$

which is exactly the final result obtained by our counting method. □

Surprisingly, although (2) looks like a complicated formula, there exists a very simple formula for calculating the number of solutions to the general σ all-ones problem for a tree T, i.e., $2^{|U|-|D|}$. Next we give its proof.

Theorem 2.1. $N=2^{|U|-|D|}$.

Proof. For any $v\in V(\mathscr{F})$, after its Num has been determined, by our counting method we can claim that

$$\mathrm{Num}(v)=2^{|Q(v)\setminus\{v\}|-|\{B:B\subseteq Q(v)\setminus\{v\}\}|}. \tag{4}$$

In fact, at the time we delete every group B, we transfer $2^{|B|-1}$ and the Num of every vertex in B that has been determined to the parent vertex of B. Thus, it is easy to prove (4) by induction.

Our proof proceeds by induction on the number of layers. From bottom up, suppose v does not have child groups, i.e., v is a leaf. Then (4) obviously holds. Suppose v has s child groups $B_1,\ldots,B_s$, which has vertices $v_{11},\ldots,v_{1m_1}; v_{21},\ldots,v_{2m_2};\ldots; v_{s1},\ldots,v_{sm_s}$, respectively. By the induction hypothesis, we have that $\mathrm{Num}(v_{ij})=2^{|Q(v_{ij})\setminus\{v_{ij}\}|-|\{B:B\subseteq Q(v_{ij})\setminus\{v_{ij}\}\}|}$, $1\leqslant i\leqslant s$, $1\leqslant j\leqslant m_i$. Then

$$\begin{aligned}\mathrm{Num}(v)&=2^{\sum_{i=1}^{s}|B_i|-s}\prod_{i=1}^{s}\prod_{w\in B_i}\mathrm{Num}(w)\\&=2^{\sum_{i=1}^{s}|B_i|-s}\prod_{i=1}^{s}\prod_{w\in B_i}2^{|Q(w)\setminus\{w\}|-|\{B:B\subseteq Q(w)\setminus\{w\}\}|}\\&=2^{\sum_{i=1}^{s}|B_i|+\sum_{i=1}^{s}\sum_{w\in B_i}|Q(w)\setminus\{w\}|}\times 2^{-(s+\sum_{i=1}^{s}\sum_{w\in B_i}|\{B:B\subseteq Q(w)\setminus\{w\}\}|)}\\&=2^{|Q(v)\setminus\{v\}|-|\{B:B\subseteq Q(v)\setminus\{v\}\}|}.\end{aligned}$$

Hence our claim holds. □

Suppose only m isolated vertices $v_1, \ldots, v_m$ and s groups $B_1, \ldots, B_s$ in the top layer are left after deleting groups by using our counting method, successively. Then

$$\begin{aligned}
N &= 2^{m+\sum_{i=1}^{s}|B_i|-s}\prod_{i=1}^{m}\mathrm{Num}(v_i)\prod_{i=1}^{s}\prod_{w\in B_i}\mathrm{Num}(w)\\
&= 2^{m+\sum_{i=1}^{s}|B_i|-s}\prod_{i=1}^{m}2^{|Q(v_i)\setminus\{v_i\}|-|\{B:B\subseteq Q(v_i)\setminus\{v_i\}\}|}\prod_{i=1}^{s}\prod_{w\in B_i}2^{|Q(w)\setminus\{w\}|-|\{B:B\subseteq Q(w)\setminus\{w\}\}|}\\
&= 2^{m+\sum_{i=1}^{m}|Q(v_i)\setminus\{v_i\}|+\sum_{i=1}^{s}|B_i|+\sum_{i=1}^{s}\sum_{w\in B_i}|Q(w)\setminus\{w\}|}\\
&\quad\times 2^{-(s+\sum_{i=1}^{m}|\{B:B\subseteq Q(v_i)\setminus\{v_i\}\}|+\sum_{i=1}^{s}\sum_{w\in B_i}|\{B:B\subseteq Q(w)\setminus\{w\}\}|)}\\
&= 2^{|V(\mathscr{F})|-|B(\mathscr{F})|}\\
&= 2^{|V\setminus D|-|V\setminus U|}\\
&= 2^{|U|-|D|}.
\end{aligned}$$

Theorem 2.2. *The decision algorithm for a rooted tree T with n vertices is correct and its time complexity is* $\mathrm{O}(n)$.

Proof. When our decision algorithm stops at some vertex v and outputs NO, the state equation of v can not hold, and thus the general σ all-ones problem for T does not have solutions. Otherwise, there are two cases. One is that $D = V$, the other is that $D \neq V$ and $U \neq V$. For the former case, the truth value of every vertex has been determined and every state equation holds, which means that the general σ all-ones problem for T has a unique solution. For the latter case, the number of solutions is $2^{|U|-|D|} > 0$ by Theorem 2.1, which means that the σ all-ones problem for T has solutions. Thus our decision algorithm is correct.

It can be seen that the time complexity of our decision algorithm is $\mathrm{O}(n)$ by counting the total number of executions of Cases 1 and 2 in the algorithm. This number is at most

$$\sum_{i\geqslant 1}\sum_{w\in D_i\setminus D_{i-1}}|N(w)|\leqslant\sum_{w\in V}|N(w)|=\sum_{w\in V}d_w=2|E(T)|=2(n-1).$$

Clearly, the time complexity of executing Cases 1 and 2 is $\mathrm{O}(1)$. Hence the total time complexity of our decision algorithm is $\mathrm{O}(n)$. □

Note that in our algorithm, the choice of the root is immaterial. In fact, any vertex of T may be the root and our decision algorithm will obtain the same output. Our counting method will get the same number of solutions because $|V|$ and $|U|$ are the same.

3. The algorithm for the minimum general σ all-ones problem for trees

From our Theorem 2.1 we know that for a tree T if the general σ all-ones problem has solutions, then the number of solutions could be exponentially large. So, for the minimum general σ all-ones problem we cannot efficiently get an optimal solution simply by exhausting all the solutions. Polynomial time algorithms are needed to tackle it. In this section we give another algorithm to determine whether there exists a solution to the σ all-ones problem for a rooted tree T, and moreover this algorithm can find a solution for the minimum σ all-ones problem for trees if solutions do exist. Furthermore, the algorithm can be modified slightly to solve the minimum general σ all-ones problem. But, unlike the algorithm in Section 2, it is of no help for the counting of solutions.

At first, we sketch the rough idea for our algorithm. Similar to the algorithm for the minimum all-ones problem for trees in [2], we first set a triple (P_v, Q_v, S_v) for every vertex v of T, where P_v and Q_v are sets of vertices and S_v is a value over GF(2). Here we name S_v as the *influenced state value* of v. Then from bottom to up, we could determine the triple of each vertex of T inductively. At last, we could get the solution to the minimum σ all-ones problem from the triples of the remaining few vertices. The algorithm is as follows. Perhaps the following figure can serve as an example to help checking this algorithm.

Please cite this article as: X. Li, et al., The general σ all-ones problem for trees, Disc. Appl. Math. (2007), doi: 10.1016/j.dam.2007.08.042

Algorithm for the Minimum σ All-ones Problem for Trees.

Input: A tree T rooted at t and a triple (P_v, Q_v, S_v) for every vertex v of T.
Step 0: Initially, for every vertex v of T, set $(\{v\}, \emptyset, 0)$ for v.
Step 1: Use SUBPROCEDURE successively to change the triple for every vertex of T layer by layer from bottom up, till no two-layer tree exists. Then the following two cases are left:
Case 1: Only t is left. Suppose that its triple is (P, Q, S).
If $S = 0$, no solution exists, then stop the algorithm.
Else, choose the one with smaller cardinality in $\{P, Q\}$ to be a solution to the minimum σ all-ones problem.
Case 2: Only one-layer tree with root t is left. Add an artificial vertex t_0 to be the parent of t and denote T_0 as the new tree. Set the triple $(\{t_0\}, \emptyset, 0)$ to t_0. Operate on the two-layer tree with root t_0 by using SUBPROCEDURE. If the triple of t_0 has been changed into $(P_{t_0}, Q_{t_0}, S_{t_0})$, then Q_{t_0} is a solution to the minimum σ all-ones problem.

The SUBPROCEDURE, which is employed by our minimum σ all-ones algorithm, is given as follows.
Consider a two-layer tree with root r. This SUBPROCEDURE deletes all the grandchild vertices of r and their parent vertices from the two-layer tree, and changes the triple of r at the same time.

SUBPROCEDURE.

Input: A two-layer tree with root r.
Step 1: Repeat
Choose a grandchild vertex w of r with the parent vertex f.
Employ SUBSUB.
Until all the grandchild vertices of r and their parent vertices are deleted.

The SUBSUB, which is employed by SUBPROCEDURE, is given as follows.
Consider a two-layer tree with root r. Suppose w is a grandchild vertex of r, f is the parent vertex of w. The following SUBSUB deletes the one-layer tree with root f.

SUBSUB.

Input: A two-layer tree with root r, a grandchild vertex w of r and the parent vertex f of w.
Step 1: If the influenced state values of w and its brother vertices are different, then no solution exists, stop our algorithm for the minimum σ all-ones problem.
Else, the influenced state values of w and its brother vertices are the same. We denote S_w as the influenced state value. If $1 - S_w = 1$, change (P_r, Q_r, S_r) into $(P_r \cup P_f, Q_r \cup P_f, S_r)$; else, change (P_r, Q_r, S_r) into $(P_r \cup Q_f, Q_r \cup Q_f, S_r)$. Suppose all the triples of w and its brothers are $(P_1, Q_1, S_1), \ldots, (P_m, Q_m, S_m)$. We employ the so-called Minimum Odd (Even) Sum Algorithm to get the solution D_w (E_w) on $\{(P_1, Q_1), \ldots, (P_m, Q_m)\}$.
Case 1: $S_f = 0$. Change (P_r, Q_r, S_r) into $(P_r \cup E_w, Q_r \cup D_w, (S_r + 1 - S_w))$.
Case 2: $S_f = 1$. Change (P_r, Q_r, S_r) into $(P_r \cup D_w, Q_r \cup E_w, (S_r + 1 - S_w))$.

Finally, delete the one-layer tree with root f from the two-layer tree with root r.

Remark 3.1. The Minimum Odd (Even) Sum Problem and its algorithm can be found in [2]. The application of this algorithm on sets can be found in Step 2 of our algorithm for the minimum σ^+ all-ones problem for trees. The details are omitted.

In order to prove the correctness of our algorithm, we shall use the following notations and terminology for further discussion. The proof may be some tedious.

Definition 3.1. For a rooted tree with root t, the pseudo σ all-ones problem is to find a subset C of vertices such that for every vertex v except for t, the number of vertices in C adjacent to v is odd, while the number of vertices in C adjacent

Please cite this article as: X. Li, et al., The general σ all-ones problem for trees, Disc. Appl. Math. (2007), doi: 10.1016/j.dam.2007.08.042

to t is arbitrary. Then C is called a solution to the pseudo σ all-ones problem. The minimum pseudo σ all-ones problem is defined similarly.

According to our algorithm for the minimum σ all-ones problem, the influenced state value of every vertex has been uniquely determined if the pseudo σ all-ones problem has solutions.

Definition 3.2. Suppose r is not a leaf vertex of the rooted tree T. Denote the set of the children of r as C_r. Denote T_r as the subtree of T with root r. For $c_1, \ldots, c_s \in C_r$, let $T_{r-\{c_1,\ldots,c_s\}} = T_r \backslash (T_{c_1} \cup \cdots \cup T_{c_s})$ represent the subtree obtained by deleting the subtrees with roots $c_1, \ldots, c_s$ from T_r. Let $T_{r,\{c_1,\ldots,c_s\}} = T_{r-(C_r\backslash\{c_1,\ldots,c_s\})}$ represent the subtree obtained by deleting the subtrees rooted at the vertices in $C_r\backslash\{c_1, \ldots, c_s\}$ from T_r. Note that $T_{r,\{c_1,\ldots,c_s\}} = T_r$ when $C_r = \{c_1, \ldots, c_s\}$.

Definition 3.3. Suppose v is a vertex of T.

(1) If v does not have grandchildren, then v is a leaf or a parent vertex of a leaf, $(\{v\}, \emptyset, 0)$ is the triple assigned to v. Now, we say that v has been assigned.
(2) If v has grandchildren and all its children and grandchildren have been assigned, then employ SUBPROCEDURE to delete all the grandchildren of v and their parent vertices. (P_v, Q_v, S_v) obtained by the SUBPROCEDURE is the triple assigned to v. Now, we say that v has been assigned.

Lemma 3.1. *Suppose that v is any vertex of T. When our algorithm is executed up to v, suppose v has children $f_1, \ldots, f_k, \ldots, f_t$, where $f_1, \ldots, f_k$ are not leaf vertices and the others are leaves. When our algorithm deletes the subtrees with roots $f_1, \ldots, f_k$, it changes the triple of v into (P_v, Q_v, S_v). Then P_v represents the solution containing v to the minimum pseudo σ all-ones problem for $T_{v,\{f_1,\ldots,f_k\}}$. Q_v represents the solution not containing v to the minimum pseudo σ all-ones problem for $T_{v,\{f_1,\ldots,f_k\}}$. And S_v represents the number modulo 2 for the vertices adjacent to v in the solution to the pseudo σ all-ones problem for $T_{v,\{f_1,\ldots,f_k\}}$.*

Proof. We prove this lemma by induction on the number of layers of T. When our algorithm is executed up to v, $f_1, \ldots, f_k$ and their children have been assigned. Suppose after executing SUBSUB to delete f_1 and its children, f_2 and its children, $\ldots$, f_k and its children, the triple of v has been changed into $(P_v^{(1)}, Q_v^{(1)}, S_v^{(1)})$, $(P_v^{(2)}, Q_v^{(2)}, S_v^{(2)})$, $\ldots$, $(P_v^{(k)}, Q_v^{(k)}, S_v^{(k)})$,successively. Note that the original triple of v is $(P_v^{(0)}, Q_v^{(0)}, S_v^{(0)}) = (\{v\}, \emptyset, 0)$ and the last triple $(P_v^{(k)}, Q_v^{(k)}, S_v^{(k)}) = (P_v, Q_v, S_v)$. We claim that $\{v\} \cup (P_v^{(i)} \backslash P_v^{(i-1)})$ or $(Q_v^{(i)} \backslash Q_v^{(i-1)})$, $1 \leqslant i \leqslant k$, represents a solution to the minimum pseudo σ all-ones problem for T_{v,f_i} under the condition that v belongs to the solution or not, and $S_v^{(i)} - S_v^{(i-1)}$ represents the number modulo 2 for the vertices adjacent to v in the solution to the pseudo σ all-ones problem for T_{v,f_i}.

We only show the claim for the case $i = 1$. The other cases are similar. Assume that f_1 has children $w_1, \ldots, w_m$. When we begin to execute SUBSUB to delete f_1 and $w_1, \ldots, w_m$, suppose the triples of f_1 and $w_1, \ldots, w_m$ are $(P_{f_1}, Q_{f_1}, S_{f_1})$ and $(P_{w_1}, Q_{w_1}, S_{w_1}), \ldots, (P_{w_m}, Q_{w_m}, S_{w_m})$, respectively. By the induction hypothesis, P_{w_j} or Q_{w_j}, $1 \leqslant j \leqslant m$, represents a solution to the minimum pseudo σ all-ones problem for T_{w_j} under the condition that w_j belongs to the solution or not. S_{w_j} represents the number modulo 2 for the vertices adjacent to w_j in the solution to the pseudo σ all-ones problem for T_{w_j}. P_{f_1} or Q_{f_1} represents the minimum solution to the pseudo σ all-ones problem for $T_{f_1-\{w_1,\ldots,w_m\}}$ under the condition that f_1 belongs to the solution or not. And S_{f_1} represents the number modulo 2 for the vertices adjacent to f_1 in the solution to the pseudo σ all-ones problem for $T_{f_1-\{w_1,\ldots,w_m\}}$. The triple of v is $(P_v^{(0)}, Q_v^{(0)}, S_v^{(0)}) = (\{v\}, \emptyset, 0)$ at this time. Next we show that the steps in SUBSUB are necessary and correct.

1. The influenced state values of $w_1, \ldots, w_m$ are different. Note that S_{w_j} is the number modulo 2 of the vertices adjacent to w_j in the solution to the pseudo σ all-ones problem for T_{w_j}, $1 \leqslant j \leqslant m$. In order to guarantee that the state equation of w_j holds, we need adjust f_1 to belong to the solution C_{v,f_1} to the pseudo σ all-ones problem for T_{v,f_1} or not, that is,

$$\chi_{C_{v,f_1}}(f_1) + S_{w_j} = 1, \quad j = 1, \ldots, m, \tag{5}$$

where $\chi_{C_{v,f_1}}(f_1)=0$, if $f_1 \notin C_{v,f_1}$; $\chi_{C_{v,f_1}}(f_1)=1$, otherwise. Hence, when the influenced state values of $w_1,\ldots,w_m$ are different, the pseudo σ all-ones problem for T_{v,f_1} does not have solutions, neither the σ all-ones problem for T.

2. The influenced state value of every vertex in $\{w_1,\ldots,w_m\}$ is S_w. We still demand

$$\chi_{C_{v,f_1}}(f_1)+S_w=1 \tag{6}$$

to guarantee that the state equation of every w_j holds. But (6) means that whether or not f_1 belongs to the solution C_{v,f_1} to the pseudo σ all-ones problem depends on the value of $1-S_w$. If $1-S_w=1(0)$, then $f_1 \in (\notin) C_{v,f_1}$. Obviously, in order to get a solution to the minimum pseudo σ all-ones problem for T_{v,f_1}, we must union P_{f_1} (Q_{f_1}) with $P_v^{(0)}$ and $Q_v^{(0)}$. Now, the state equation of every vertex in $\{w_1,\ldots,w_m\}$ holds. In order to guarantee that the state equation f_1 also holds, we must have

$$\chi_{C_{v,f_1}}(v)+S_{f_1}+\sum_{j=1}^{m}\chi_{C_{v,f_1}}(w_j)=1.$$

In order to obtain the optimal solution, if $\chi_{C_{v,f_1}}(v)+S_{f_1}=1$, we need to get the solution E_w to the Minimum Even Sum Problem on $\{(P_{w_1},Q_{w_1}),\ldots,(P_{w_m},Q_{w_m})\}$. Otherwise, we need to get the solution D_w to the Minimum Odd Sum Problem. And then we also need to union E_w or D_w into $P_v^{(0)}$ and $Q_v^{(0)}$ correspondingly to get $P_v^{(1)}$ and $Q_v^{(1)}$, respectively. In addition, after $\chi_{C_{v,f_1}}(f_1)$ has been determined, the corresponding influenced state value of v will be changed into $S_v^{(0)}+1-S_w=S_v^{(0)}+\chi_{C_{v,f_1}}(f_1)$, which is exactly $S_v^{(1)}$.

After executing SUBSUB, $P_v^{(1)}=\{v\}\cup(P_v^{(1)}\backslash P_v^{(0)})$ or $Q_v^{(1)}=(Q_v^{(1)}\backslash Q_v^{(0)})$ represents a solution to the minimum pseudo σ all-ones problem for T_{v,f_1} under the condition that v belongs to the solution or not, and $S_v^{(1)}-S_v^{(0)}$ represents the number modulo 2 for the vertices adjacent to v in the solution to the pseudo σ all-ones problem for T_{v,f_1}. Thus, the steps in SUBSUB are necessary and correct, then the claim holds.

So,

$$P_v=P_v^{(k)}=\{v\}\cup(P_v^{(k)}\backslash P_v^{(0)})=\{v\}\cup(P_v^{(k)}\backslash P_v^{(k-1)})\cup(P_v^{(k-1)}\backslash P_v^{(k-2)})\cdots\cup(P_v^{(1)}\backslash P_v^{(0)})$$

or

$$Q_v=Q_v^{(k)}=(Q_v^{(k)}\backslash Q_v^{(0)})=(Q_v^{(k)}\backslash Q_v^{(k-1)})\cup(Q_v^{(k-1)}\backslash Q_v^{(k-2)})\cdots\cup(Q_v^{(1)}\backslash Q_v^{(0)})$$

represents the solution containing v or not to the minimum pseudo σ all-ones problem for $T_{v,\{f_1,\ldots,f_k\}}$. And

$$S_v=S_v^{(k)}=(S_v^{(k)}-S_v^{(0)})=(S_v^{(k)}-S_v^{(k-1)})+(S_v^{(k-1)}-S_v^{(k-2)})+\cdots+(S_v^{(1)}-S_v^{(0)})$$

represents the number modulo 2 for the vertices adjacent to v in the solution to the pseudo σ all-ones problem for $T_{v,\{f_1,\ldots,f_k\}}$. □

Theorem 3.1. *The algorithm for the minimum σ all-ones problem for a given rooted tree T outputs an optimal solution and the time complexity is linear.*

Proof. In Step 1, when we call SUBPROCEDURE successively until no two-layer tree exists, the following two cases are left.

Case 1: Only one vertex is left, i.e., the root t of T. Suppose (P, Q, S) is the final triple of t. P is the optimal solution containing t and Q is the optimal solution not containing t to the minimum pseudo σ all-ones problem for T. If $S=1$, the number of the vertices adjacent to t in either P or Q is odd, then both P and Q are solutions to the σ all-ones problem for T. Thus, the one in $\{P, Q\}$ with smaller cardinality is an optimal solution to the minimum σ all-ones problem for T. If $S=0$, the number of the vertices adjacent to t in either P or Q is even. Since the influenced state value of every vertex has been uniquely determined if the pseudo σ all-ones problem has solutions, the influenced state value does not depend on whether or not the solution is an optimal one. Therefore, the number of vertices adjacent to t in any solution to the pseudo σ all-ones problem will be even. Then the σ all-ones problem does not have solutions.

Please cite this article as: X. Li, et al., The general σ all-ones problem for trees, Disc. Appl. Math. (2007), doi: 10.1016/j.dam.2007.08.042

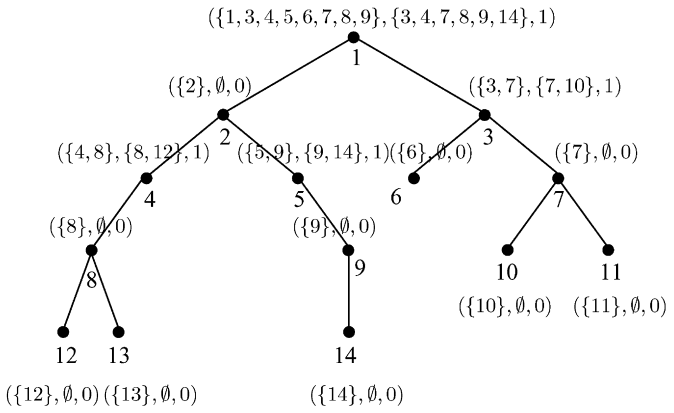

Fig. 2. An example of our algorithm for the minimum σ all-ones problem for trees. The solution is {3, 4, 7, 8, 9, 14}.

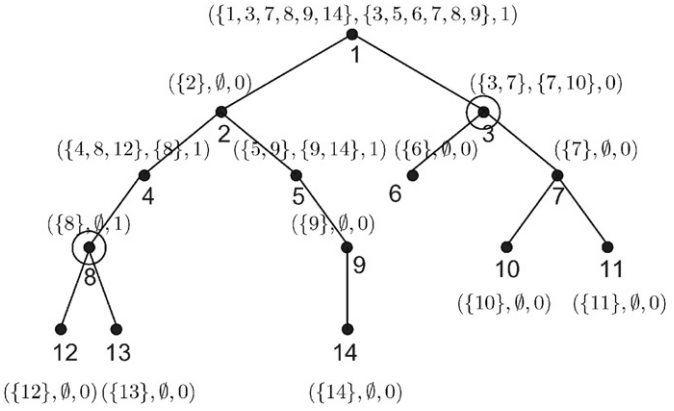

Fig. 3. An example of our algorithm for the minimum general σ all-ones problem for trees with the restriction that all the initial states of vertices are 0 except the states of vertices 3 and 8.

Case 2: Only one-layer tree with root t is left. When our algorithm for the minimum σ all-ones finishes the execution of Case 2, we get $(P_{t_0}, Q_{t_0}, S_{t_0})$ where t_0 is an artificial vertex mentioned in Case 2 of the algorithm. By Lemma 3.1, Q_{t_0} is a solution not containing t_0 to the minimum pseudo σ all-ones problem for T_0. Clearly, Q_{t_0} is an optimal solution to the minimum σ all-ones problem for T.

For the time complexity, one can see that our algorithm proceeds layer by layer on T from bottom up. For every layer our algorithm uses time linear in the number of vertices in the layer, even though sometimes the algorithm of linear time for the Minimum Odd or Even Sum problem has to be used. It is then easy to see that the total time complexity used by our algorithm is still linear. □

An example to show our algorithm at work is given in Fig. 2 as before. The rooted tree T in Fig. 2 has 14 vertices $\{1, 2, \ldots, 14\}$. Initially, our algorithm set a triple $(\{v\}, \emptyset, 0)$ for every vertex v. Then, from bottom up, the triple for every vertex which has grandchildren can be assigned. In the end, by comparing $|\{1, 3, 4, 5, 6, 7, 8, 9\}| = 8$ and $|\{3, 4, 7, 8, 9, 14\}| = 6$, we get an optimal solution $\{3, 4, 7, 8, 9, 14\}$ to the minimum σ all-ones problem for the rooted tree. The details about the triple (P_v, Q_v, S_v) for every vertex v are recorded in Fig. 2.

In fact, we can change the algorithm for the minimum σ all-ones problem for trees slightly to solve the minimum general σ all-ones problem $AX + \mathbf{b} = 1$ where $\mathbf{b}$ is the initial configuration. We need only to set $(\{v\}, \emptyset, \mathbf{b}_v)$ instead of $(\{v\}, \emptyset, 0)$ for every vertex v of T initially in Step 0 while the other parts of the algorithm remains unchanged.

It is easy to see that the new algorithm is guaranteed to terminate with an optimal solution for the minimum general σ all-ones problem. An example is as follows (Fig. 3). If we restrict all the initial states of vertices to be 0 except the states

Please cite this article as: X. Li, et al., The general σ all-ones problem for trees, Disc. Appl. Math. (2007), doi: 10.1016/j.dam.2007.08.042

of vertices 3 and 8, we can get the triple of vertex 1 being ({1, 3, 7, 8, 9, 14}, {3, 5, 6, 7, 8, 9}, 1) after we execute the new algorithm. So $P = \{1, 3, 7, 8, 9, 14\}$ and $Q = \{3, 5, 6, 7, 8, 9\}$ can be both the solution to the minimum σ all-ones problem.

Acknowledgment

The authors are very grateful to the referees for their helpful comments and suggestions, which helped to improve the presentation of the paper.

References

[1] R. Barua, S. Ramakrishnan, σ-game, σ^+-game and two-dimensional additive cellular automata, Theoret. Comput. Sci. 154 (1996) 349–366.
[2] W.Y.C. Chen, X.L. Li, C. Wang, X.Y. Zhang, The minimum all-ones problem for trees, SIAM J. Comput. 33 (2) (2004) 379–392.
[3] Y. Dodis, P. Winkler, Universal configurations in light-flipping games, in: Proceedings of the 12th Annual ACM/SIAM Symposium on Discrete Algorithms (SODA), January 2001, pp. 926–927.
[4] R.G. Downey, M.R. Fellows, A. Vardy, G. Whittle, The parametrized complexity of some fundamental problems in coding theory, SIAM J. Comput. 29 (2) (1999) 545–570.
[5] H. Eriksson, K. Eriksson, J. Sjöstrand, Note on the lamp lighting problem, Adv. in Appl. Math. 27 (2001) 357–366.
[6] F. Galvin, Solution to problem 88-8, Math. Intelligencer 11 (2) (1989) 31–32.
[7] O.P. Lossers, Solution to problem 10197, Amer. Math. Monthly 100 (8) (1993) 806–807.
[8] C.H. Papadimitriou, K. Steiglitz, Combinatorial Optimizations: Algorithms and Complexity, Prentice-Hall Inc., Englewood Cliffs, NJ, 1982.
[9] U. Peled, Problem 10197, Amer. Math. Monthly 99 (2) (1992) 162.
[10] K. Sutner, Additive automata on graphs, Complex Systems 2 (6) (1988) 649–661.
[11] K. Sutner, Problem 88-8, Math. Intelligencer 10 (3) (1988).
[12] K. Sutner, Linear cellular automata and the Garden-of-Eden, Math. Intelligencer 11 (2) (1989) 49–53.
[13] K. Sutner, The σ-game and cellular automata, Amer. Math. Monthly 97 (1990) 24–34.
[14] K. Sutner, On the computational complexity of finite cellular automata, J. Comput. System Sci. 50 (1995) 87–97.
[15] K. Sutner, σ-automata and Chebyshev-polynomials, Theoret. Comput. Sci. 230 (2000) 49–73.
[16] T. Yaku, The constructibility of a configuration in a cellular automaton, J. Comput. System Sci. 7 (1973) 481–496.