The Bulletin of Society for Mathematical Services and Standards *Online: 2013-03-01*
ISSN: 2277-8020, Vol. 5, pp 31-36
doi:10.18052/www.scipress.com/BSMaSS.5.31

Embedding Sun Graphs in a Single page

B. Mahavir

Department of Mathematics, A.M. Jain College, Chennai - 600 114, India

Keywords: Book embedding, book thickness, page number, VLSI design

Abstract: A *book* consists of a line in the 3-dimensional space, called the spine, and a number of *pages*, each a half-plane with the spine as boundary. A *book embedding* (π, ρ) of a graph consists of a linear ordering of π, of vertices, called the *spine ordering*, along the spine of a book and an assignment ρ, of edges to pages so that edges assigned to the same page can be drawn on that page without crossing. That is, we cannot find vertices u, v, x, y with $\pi(u) < \pi(x) < \pi(v) < \pi(y)$, yet the edges uv and xy are assigned to the same page, that is $\rho(uv) = \rho(xy)$. The *book thickness* or *page number* of a graph G is the minimum number of pages in required to embed G in a book. In this paper we consider the *Sun Graph* or the *Trampoline graph* and obtain the printing cycle for embedding the Sun Graph in a single page. We also give a linear time algorithm for such an embedding.

1. Introduction

The growth of the subject 'graph theory' has been very rapid in recent years, particularly since the domain of its application is extremely varied. Graph algorithms play a very important role in design of various computer networks. Among theproblems one comes across in graph theory, is the embedding of graphs. A particular way of embedding graphs is in the pages of a book. The book embedding of graphs was first introduced by Bernhart and Kainen [1] and since then, many researchers have actively studied it. Determining the book thickness for general graphs is *NP*- hard. But obtaining the book thickness for particular graphs have been found to be possible. The book embeddings have been studied for many classes of graphs. To name a few, we have: Complete Graphs [1, 2], Complete Bipartite Graphs[10], Trees, Grids and X-trees [3], hypercubes [3, 9], incomplete hypercubes [8], iterated line digraphs [6], de Bruijn graphs, Kautz graphs, shuffle-exchange graphs [7], for each of which embedding in books have been studied.

The book embedding problem has many different applications, which include sorting with parallel stacks, single-row routing of printed circuit boards, and the design of fault-tolerant processor arrays [4, 11].

2. Preliminaries

Definition: The Sun Graph (or the trampoline graph) network of order n denoted by S^n is defined as follows. S^1 is K_3. S^n, $n \geq 2$ is obtained from S_{n-1} by adding a new vertex v corresponding to each edge ab on the Hamiltonian cycle of S^n and including the edges (v, a) and (v, b). See Figure 1.

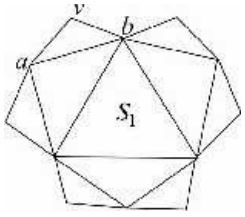

Figure 1 S^3, The Sun Graph of order 3

We give the definitions which are required for the discussion.

Definition: The *interlace* of the finite sequences $(,a_1..,a_r)$ and $(b, _1..,b_r)$ is denoted by $(,a_1..,a_r)$. $\Delta..,(b_1\ b_r)$ and is defined as the sequence given by,

$$(,a_1..,a_r)) .\Delta..,(b_1\ b_r\ ) = (,a_1\ ,b_1..,a,\ _r b_r)$$

For example the interlace of the sequences (1, 3, 5, 7) and (2, 4, 6, 8) is the sequence (1, 2, 3, 4, 5, 6, 7, 8)

Definition: The *progressive interlace* of order k of the sequence $s = (,a.1..,a_i)$ is denoted by

$\Delta^k s$ and is recursively defined as, $\Delta^k s = \Delta()\Delta^{k-1}s)$, where

$$\Delta^1 s = \Delta^1(a_1,...,a_i) = (a_1,...,a_i)\Delta(i+1,...,2k) \text{ and } \Delta^0 s = s$$

Clearly, $|\ \Delta^k s\ | = 2^{k-1}\ |\ s\ |$.

For example if s = (13,14,16,18), then

$\Delta^2 s$

= Δ(Δ13,14,16,18))

= Δ((13,14,16,18)Δ(5, 6, 7, 8))

= Δ(13, 5,14, 6,16, 7,18, 8)

= (13, 5,14, 6,16, 7,18, 8)Δ(9,10,11,12,13,14,1 5,16)

= (13, 9, 5,10,14,11, 6,12,16,13, 7,14,18,15, 8,16)

3. Embedding the Sun graph of order k in a single page

We shall discuss the embedding of the Sun graph of order k.

Lemma 1: The Sun graph S^k, $k \geq 1$ can be embedded in a single page.

Proof: The Sun graph S^k, $k \geq 1$ is outerplanar[3]. Hence it can be embedded in a single page. Hence the lemma.

We shall now find the printing cycle of the book embedding of the Sun graph in a single page.

Theorem 2: The printing cycle S^n is Δ^{n-1} (1, 2, 3),$n \geq 1$embeds S^n, $n \geq 1$ in a single page.

Proof: Let us prove the theorem by induction on n, the order of the Sun graph. When n = 1, S^1 is K_3. The printing cycle (1, 2, 3) = Δ^0 (1, 2, 3) embeds S^1 in a single page. We shall also prove the theorem for the case n = 2. To obtain S^2, we add three more vertices 4, 5 and 6 and constructing three K_3 graphs with vertex sets (1, 4, 2), (2, 5, 3) and (3, 6, 1). The new vertices are added to the spine in the following way. Since 1 and 2 are adjacent on the spine, mark the vertex 4 between 1 and 2. Similarly mark the vertex 5 between 2 and 3. However since 3 and 1 are not adjacent on the spine, extend the spine beyond 3 and mark 6 adjacent to it. This would yield the printing cycle 1, 4,2, 5, 3, 6 =$\Delta^1 s$.

Now let us assume that when $n = k$, the nest and extend operation embeds s^k in a single page with the printing cycle Δ^{k-1} (1, 2, 3). Let us prove that the nest and extend operation would embed $_s k$ +1 in a single page with printing cycle Δ^k (1, 2, 3).

Let $_\Delta k-1(1, 2, 3),=...a,_1, a_2\ _{3.2}k-1$ be the hamilton cycle in s^k. To obtain $_s{}^{k+1}$, we introduce vertices $a_{3.2^{k-1}+1,}\ a_{3.2^{k-1}+2},...,a_{3.2^{k-1}+3.2^{k-1}} = a_{3.2^k}$ so that $(a_i, a_{3.2^{k-1}+i}, a_{i+1})$ is the vertex set of a K_3 graph, for i = 1, 2, ...,$3.2^{k-1} + 3.2^{k-1} - 1 = 3.2^k - 1$. This would interlace the sequences $(a_{1,}a_2,...,a_{3.2^{k-1}-1})$

and $(a_{3.2^{k-1}+1}, a_{3.2^{k-1}+2}, \ldots, a_{3.2^k-1})$ The single remaining vertex $3.2k$ is marked by extending the spine beyond 3.2^{k-1} adjacent to it. The resulting printing cycle this is

$((a_1, a_2, \ldots, a_{3.2^{k-1}}) \Delta (a_{3.2^{k-1}+1}, a_{3.2^{k-1}+2}, \ldots, a_{3.2^k}) = \Delta^k(1,2,3)$

Apparently, this printing cycle would embed s^{k+1} in a single page.
This completes the induction and the theorem is proved.

4. Embedding Algorithm

We now give the algorithm to obtain the printing cycle for embedding the sun graph s^k in a single page.

```
1       ALGORITHM BKEMBEDSUN(a, k)
2       // The algorithm will populate the array a[ ] with the printing cycle of the
3       // sun graph of order k
4       {
5       i = 1; k = 1; m = 3;
6       a[1] = 1; a[2] = 2; a[3] = 3;
7               while(i <= k)
8               {
9                       if(i != 1)
10                      {
11                      l = m;
12                      m = 2*m;
13                      j = m–1;
14                              for(i = l; i > 1; i––)
15                              {
16                              a[j] = a[i];
17                              j = j–2;
18                              }
19                                      t = n/2+1;
20                                  for(i = 2; i < n; i = i+2)
21                                  {
22                                  a[i] = t;
23                                  t++;
24                                  }
25                      }
26              i++;
27              }
28 }
```

Now, a brief description about the algorithm is given.
Line 5 initializes the variables that are used later during the execution of the algorithm with appropriate values.

Line 6 populates *a*[ ] with the printing cycle of S^1 .
Lines 7 to 28 are within the scope of a 'while' loop, which is expected to perform the following. For each value of $i = 1, 2, \ldots, k$, when the 'while' loop iterates once with some value of i, say $i = r$, the array a[1, ..., 3.2^{s-1}] will contain the printing cycle of S^{r-1} .
The 'if' statement in line 9 singles out the case $i = 1$, for which the printing cycle has already been initialized in line 6.

Each time we go through the 'while' loop once, the number of vertices is exactly the double compared to the previous iteration. In line 11, the number of vertices in S^{r-1} is preserved in the variable *l*, while the number of vertices of S^r is assigned to the variable *m* in line 12.
The 'for' loop in lines 14 to 18 moves the elements of a[1, ..., *l*] to a[1, ..., *m*-1] in such a manner that the hamilton cycle of S^{r-1} is placed in a[1, 3, 5, ..., *m*-1] leaving space for the additional vertices in S^r to be placed in a[2, 4, 6, ..., m].
In line 19, variable *t* is initialized with the value of *l*+1, which is the first new vertex added to S^{r-1} to obtain S^r.
The 'for' loop in lines 20 to 24 assigns the successive values *l*+1, ..., 2*l* = *m* to the cells *a*[*i*], *i* = 2, 4, ..., *m*.
Line 26 increments the value if *i* to obtain the printing cycle of the next S^i or to exit the loop when *i* has reached the value *k*+1.

5. Proof of correctness of the embedding algorithm

We now give the proof of correctness of the Algorithm BKEMBEDSUN(a,k).

Theorem 3: Algorithm BKEMBEDSUN(*a*,*k*) gives the printing cycle for embedding S^k in a single page.
Proof: Let us prove the theorem using loop invariants**[ ]**. This method is suitable since the entire execution in the algorithm is within the while( ) loop in the algorithm, outside of which only initialization of the variables used have been done.

The executable statements in the algorithm starts in line 5 by initializing the variable *i* with 1, which will indicate the order of the current Sun graph whose printing cycle is being obtained. Also the variable *m* is initialized with 3, the number of vertices of S^1. At any given instance after line 12, the *m* will contain the number of vertices of the current Sun graph. Now we shall use loop invariants to give the proof of correctness.
Initialization: Initially when *k* = 1, the 'while' loop on entering exits at once due to the 'if' statement after incrementing the variable *i* by 1 in line number 26. But then in line 6, a[1, 2, 3] has already been assigned with the printing cycle of S^1, (1, 2, 3). Hence when *k* = 1, the algorithm gives the printing cycle that will embed S^1 in a single page. Further at the end of this iteration, the value of *m* is 3, the number of vertices in S^1.
Maintenance: Assume that at the beginning of the iteration for which $i = r$, $m = 3 \times 2^{r-2}$, the number of vertices of S^{r-1} and that the array a[1, ..., *m*] contains the printing cycle of S^{r-1}. Let us prove that at the end of this iteration, $m = 3 \times 2^{r-1}$ and that the array a[1, ..., *m*] contains the printing cycle of S^r.
The number of terms involved in the sequence $\Delta^{r-2}(1, 2, 3)$ is $3 \times 2^{r-2}$, the present value of *m* and *l* is assigned this value in line 11. Thus *l* will contain the number of vertices of S^{r-1}. In line 12, *m* is doubled so that its value will be the $2 \times 3 \times 2^{r-2} = 3 \times 2^{r-1}$, the number of vertices in S^r.
The variable *j* will point to the element a[*m*-1], the last element of a[1, ..., *m*] having the odd index.
The *'for' loop* in lines 14 to 18 moves a[*l*] to a[*m*-1], a[l-1] to a[m-3], and so on and finally a[2] to a[3]. Thus (a[1], a[3], ..., a[m-1]) = $\Delta^{r-2}(1, 2, 3)$.
Line 19 initializes *t* with *l*+1, the first new vertex to be added to S^{r-1} in order to obtain the printing cycle of S^r.

The *'for' loop* from in lines 20 to 24 executes for each value of i = 2, 4, ..., m and places in $a[2]$, $a[4]$, ..., $a[m]$, the values l+1, l+2, ..., m.
Hence after the *'for' loop* executes, $a[1, ..., m]$ will contain Δ^{r-2} (1, 2, 3) Δ (l+1, l+2, ..., m) = Δ^{r-1} (1, 2, 3) , the printing cycle of S^r .

Termination: When i reaches the value k+1, the *while*( ) loop exits and a[1, ..., m] will contain the printing cycle obtained at the end of the previous iteration corresponding to $i = k$, which is Δ^{k-1} (1, 2, 3) , the printing cycle of S^k.
Hence Algorithm BKEMBEDSUN(a, k) will produce Δ^{k-1} (1, 2, 3) , the printing cycle of the Sun graph S^k . Hence the theorem is proved.

6. Time Complexity

The following theorem proves that Algorithm BKEMBEDSUN(a, k) is a linear time algorithm on the number of vertices of S^k .

Theorem 4: Algorithm BKEMBEDSUN(a, k) requires O(n) time to execute, where n is the number of vertices in the Sun graph S^k , of order k.

Proof: Executable statements in the algorithm begin in line 5 onwards. Lines 5 and 6 require constant time.
The *while*( ) loop in lines 7 to 28 runs once for each value of i = 1, 2, ..., k.
Lines 9 to 13 require constant time.
The *'for' loop* in lines 14 to 18 makes $m/2 = 3 \times 2^{i-2}$ movements where m is the number of vertices in S^i . The *'for' loop* n lines 20 to 24 makes $m/2 = 3 \times 2^{i-2}$ assignments.
The other statements in the lines 19, 23 and 26 require constant time to execute.
In all, for a particular value of i, the current iteration of the *while*( ) loop will execute in a time proportional to $3 \times 2^{i-2} + 3 \times 2^{i-2} + c$, where c_1 is a constant.
Hence when the *while*( ) loop iterates for i = 2, 3, ..., k, the total time required will be proportional to $3 \times 2 + 3 \times 2^2 + ... + 3 \times 2^{k-1} + c$, where c_2 is a constant.
The time required for executing line 6 is proportional to 3×2^0 .
Hence the total time required for the algorithm to execute is proportional to,

$$3 \times 2^0 + 3 \times 2^1 + 3 \times 2^2 + ... + 3 \times 2^{k-1} + c \text{ , where } c \text{ is a constant}$$

$$= 3 \times (1 + 2^1 + ... + 2^{k-1}) + c_3$$

$$= 3 \times \frac{1 \times (2^k - 1)}{2 - 1} + c_3$$

$$= 3 \times (2^k - 1) + c_3$$

$$= 2 \times (3 \times 2^{k-1}) - 3 + c_3$$

$= 2 \times n + c$, where n is the number of vertices of S^k and c is a constant.
$= O(n)$.
This proves the theorem.

7. Conclusion:

From the above discussion, it is apparent that even for outerplanar graphs, it may be very difficult to obtain the printing cycle that would embed such graphs in a single page. This once again underlines the fact that not many general results exist or are obtained in the book embedding of graphs.

References

[1] Bernhart, F. and Kainen, P.C., The book thickness of a graph. J. Combin. Theory Ser. B. v27. 320-331.

[2] T. C. Biedl, T. Shermer, S. Wiitesided, S. Wismath, Bounds for orthogonal 3-D graph drawing, Journal of Graph Algorithms Appl. 3(1999) 63-79.

[3] F. R. K. Chung , Frank Thomson Leighton , Arnold L. Rosenberg, Embedding graphs in books: a layout problem with applications to VLSI design, SIAM Journal on Algebraic and Discrete Methods, v.8 n.1, p.33-58, January 2, 1987.

[4] F. R. K. Chung, F. T. Leighton, and A. L. Rosenbert, Diogenes – A methodology for designing fault-tolerant processor arrays, 13th International Conference of Fault- Tolerant Computing, 1983, pp. 26-32.

[5] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, Introduction to Algorithms, PHI, Second Edition, 2005.

[6] Hasunuma, T., Embedding iterated line digraphs in books. Networks. v40. 51-62. [7] Hasunuma, T.,Yukio Shibata, Embedding de-Bruijn, Kautz and shuffle-exchange networks in books, Discrete Applied Mathematics, v.78 n.1-3, p.103-116, Oct. 21, 1997.

[7] Jywe-Fei Fang , Kuan-Chou Lai, Embedding the incomplete hypercube in books, Information Processing Letters, v.96 n.1, p.1-6, 16 October 2005.

[8] Konoe, M., Hagihara, K. and Tokura, N., On the pagenumber of hypercubes and cube-connected cycles. IEICE Trans. vJ71-D. 490-500.

[9] Muder, D.J., Weaver, M.L. and West, D.B., Pagenumber of complete bipartite graphs, Journal of Graph Theory. v12. 469-489.

[10] L. Rosenbert, The Diogenes approach to testable fault-tolerant arrays into processors, IEEE Trans. Comput., C-32 (1983), 902-910.