# A linear time algorithm for the Hamiltonian path problem on a series-parallel graph

Q. S. Wu* R. C. T. Lee† K. M. Chao‡

November 23, 1999

## Abstract

Given a graph $G = (V, E)$, a Hamiltonian path is a path that traverses each vertex of $G$ exactly once. The problem of determining whether a graph $G$ has a Hamiltonian path is a well-known NP-complete problem. In this paper, we shall present a linear time algorithm for solving the Hamiltonian path problem on a series-parallel graph.

## 1 Introduction

Let $G = (V, E)$ be a graph, where $V = \{v_1, v_2, \ldots, v_n\}$ is the vertex set. A *path* is an edge set $\{(v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k)\}$ where $v_1, v_2, \ldots, v_k$ are distinct. The *length* of a path is the cardinality of the edge set. A *Hamiltonian path* of a graph $G = (V, E)$ is a path whose length is $n - 1$. The *Hamiltonian path problem* is to determine whether a given graph $G$ has a Hamiltonian path or not. In Fig. 1, the graphs (a)(b)(c) have Hamiltonian paths, while the graphs (d)(e) do not. Some special graphs, such as cycles $C_n$ or complete graphs $K_n$, always have Hamiltonian paths [2]; some others, such as trees, can be easily determined that they do not have a Hamiltonian path [1]. However, in general, it is NP-complete to determine whether a given graph $G$ has a Hamiltonian path or not [3].

Series-parallel graphs are a special class of graphs on which several NP-complete problems are found to have efficient algorithms. Especially, many vertex-type problems such as the weighted perfect domination problem [10], the weighted efficient domination problem [8], the minimum vertex cover problem and the minimum feedback vertex set problem [9] are found to have polynomial time algorithms on series-parallel graphs. On the other hand, to find efficient algorithms for edge-type problems, sophisticated design is needed [4].

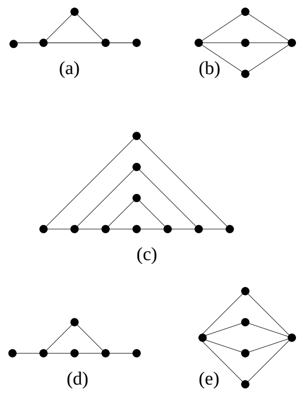


Figure 1: Examples for Hamiltonian and non-Hamiltonian series-parallel graphs.

The Hamiltonian circuit problem is a similar problem which is also NP-complete [3]. People usually regard these two problems as equivalent because they can be reduced to each other on general graphs. On series-parallel graphs, the Hamiltonian circuit problem can be solved in polynomial time [5]. However, the reduction in [3] on general graphs cannot be applied directly on series-parallel graphs to obtain a polynomial time algorithm for the Hamiltonian path problem. In this paper, we describe a linear algorithm to solve the Hamiltonian path problem on series-parallel graphs.

**Definition 1** *A series-parallel graph $G$ is recursively defined as follows [7]:*

1. *An edge $(u, v)$ with two distinct vertices $u$ and $v$ is a series-parallel graph with terminals $u$ and $v$.*

2. *The graph constructed by the series connection or the parallel connection of two series-parallel graphs $G_1$ and $G_2$ is a series-parallel graph with terminals $u_1$ and $v_2$.*

*Dept. of Computer Science, National Tsing Hua University, Hsinchu, Taiwan 30043, Republic of China, e-mail address: solomon@cs.nthu.edu.tw

†National Chi-Nan University, Puli, Nontou Hsien, Taiwan 545, e-mail address: rctlee@ncnu.edu.tw

‡Dept. of Life Science, National Yang-Ming University, Taipei, Taiwan 112, email address: kmchao@ym.edu.tw

3. *Only graphs constructed by a finite number of applications of connections mentioned in (2) are series-parallel graphs.*

## 2 Basic Rules Used in Our Algorithm

An edge set is said to contain exactly $k$ paths if it can be partitioned into $k$ vertex-disjoint sets, and each of them is a path. An edge $(u,v)$ is said to *cover* vertices $u$ and $v$. A vertex $v$ is said to be covered by a path (an edge set) if some edge in the set covers $v$. A graph $G=(V,E)$ is covered by a path $P$ if $P \subseteq E$ and for all $v \in V$, $v$ is covered by $P$.

Let $G=(V,E)$ be a series-parallel graph with terminals $u$ and $v$. We define seven types of edge sets:

- $H_1(G)$: An edge set containing exactly one path, which covers all vertices in $V$. The vertex $u$ is covered by exactly one edge in this set, and the vertex $v$ is also covered by exactly one edge in this set.
- $H_2(G)$: An edge set containing exactly one path, which covers all vertices in $V$. The vertex $u$ is covered by exactly one edge in this set, and the vertex $v$ is covered by exactly two edges in this set.
- $H_3(G)$: An edge set containing exactly one path, which covers all vertices in $V$. The vertex $u$ is covered by exactly two edges in this set, and the vertex $v$ is covered by exactly one edge in this set.
- $H_4(G)$: An edge set containing exactly one path, which covers all vertices in $V$. The vertex $u$ is covered by exactly two edges in this set, and the vertex $v$ is covered by exactly two edges in this set.
- $H_5(G)$: An edge set containing exactly one path, which covers all vertices in $V-\{u\}$. The vertex $v$ is covered by exactly one edge in this set.
- $H_6(G)$: An edge set containing exactly one path, which covers all vertices in $V-\{v\}$. The vertex $u$ is covered by exactly one edge in this set.
- $H_7(G)$: An edge set containing exactly two paths, and the union of these two paths covers all vertices in $V$. Exactly one edge in this edge set covers $u$, and exactly one edge in this edge set covers $v$.

Among all of the seven types of edge sets, $H_i(G)$ is a Hamiltonian path of $G$ for $i=1,2,3$ and 4. $H_5(G)$ is a path which covers $V-\{u\}$. $H_6(G)$ is a path which covers $V-\{v\}$. $H_7(G)$ contains two paths, whose union covers $V$. To produce Hamiltonian paths, $H_5(G), H_6(G)$ and $H_7(G)$ are also needed as shown below.

First, we show how $H_i(G)$'s can be combined serially.

$$
\begin{array}{lll}
\text{Rule}S_1 : H_1(N) & := & H_1(G_1) \cup H_1(G_2) \\
\text{Rule}S_2 : H_2(N) & := & H_1(G_1) \cup H_2(G_2) \\
\text{Rule}S_3 : H_3(N) & := & H_3(G_1) \cup H_1(G_2) \\
\text{Rule}S_4 : H_4(N) & := & H_3(G_1) \cup H_2(G_2) \\
\text{Rule}S_5 : H_5(N) & := & H_5(G_1) \cup H_1(G_2) \\
 & \| & H_1(G_2), \text{if } G_1 \text{ is an edge} \\
\text{Rule}S_6 : H_6(N) & := & H_1(G_1) \cup H_6(G_2) \\
 & \| & H_1(G_1), \text{if } G_2 \text{ is an edge} \\
\text{Rule}S_7 : H_7(N) & := & H_1(G_1) \cup H_7(G_2) \\
 & \| & H_7(G_1) \cup H_1(G_2) \\
 & \| & H_1(G_1) \cup H_5(G_2), \text{if } G_1 \text{ is an edge} \\
 & \| & H_6(G_1) \cup H_1(G_2), \text{if } G_2 \text{ is an edge}
\end{array}
$$

Note that when computing the set union $A \cup B$, if either $A$ or $B$ is UNDEF, the result is UNDEF. Besides, the operator $\|$ is a conditional operator meaning OR. The action to the left of $\|$ is executed. If its result to the left is not UNDEF, the action to the right of $\|$ is ignored. If the left side is UNDEF, the right side is then executed and the value of the entire expression is the result of actions in the right side. In other words, $\|$ is a conditional OR operation that does not execute its right side if the left one succeeds [6]. For readability, we write $H_i := A \cup B \| C \cup D \| E \cup F$ as
$$
\begin{array}{lll}
H_i & := & A \cup B \\
 & \| & C \cup D \\
 & \| & E \cup F.
\end{array}
$$

Similarly, $H_i(G)$'s can also be combined in parallel as follows. Let $N$ be a parallel connection of two series-parallel graph $G_1$ and $G_2$.

$$
\begin{array}{lll}
\text{Rule } P_1 : H_1(N) & := & H_1(G_2), \text{if } G_1 \text{ is an edge} \\
 & \| & H_1(G_1), \text{if } G_2 \text{ is an edge} \\
\text{Rule } P_2 : H_2(N) & := & H_1(G_1) \cup H_5(G_2) \\
 & \| & H_5(G_1) \cup H_1(G_2) \\
 & \| & H_2(G_2), \text{if } G_1 \text{ is an edge} \\
 & \| & H_2(G_1), \text{if } G_2 \text{ is an edge} \\
\text{Rule } P_3 : H_3(N) & := & H_1(G_1) \cup H_6(G_2) \\
 & \| & H_6(G_1) \cup H_1(G_2) \\
 & \| & H_3(G_2), \text{if } G_1 \text{ is an edge} \\
 & \| & H_3(G_1), \text{if } G_2 \text{ is an edge} \\
\text{Rule } P_4 : H_4(N) & := & H_1(G_1) \cup H_7(G_2) \\
 & \| & H_7(G_1) \cup H_1(G_2) \\
 & \| & H_2(G_1) \cup H_6(G_2) \\
 & \| & H_6(G_1) \cup H_2(G_2) \\
 & \| & H_3(G_1) \cup H_5(G_2) \\
 & \| & H_5(G_1) \cup H_3(G_2) \\
 & \| & H_4(G_2), \text{if } G_1 \text{ is an edge} \\
 & \| & H_4(G_1), \text{if } G_2 \text{ is an edge} \\
\text{Rule } P_5 : H_5(N) & := & H_5(G_2), \text{if } G_1 \text{ is an edge} \\
 & \| & H_5(G_1), \text{if } G_2 \text{ is an edge} \\
\text{Rule } P_6 : H_6(N) & := & H_6(G_2), \text{if } G_1 \text{ is an edge} \\
 & \| & H_6(G_1), \text{if } G_2 \text{ is an edge} \\
\text{Rule } P_7 : H_7(N) & := & H_5(G_1) \cup H_6(G_2) \\
 & \| & H_6(G_1) \cup H_5(G_2) \\
 & \| & H_7(G_2), \text{if } G_1 \text{ is an edge} \\
 & \| & H_7(G_1), \text{if } G_2 \text{ is an edge}
\end{array}
$$

## 3 Algorithm HPSP

**Definition 2** *Let $G=(V,E)$ be a series-parallel graph with terminals u and v. Let $T=(V_T,E_T)$ be a binary tree in which every internal vertex has exactly two sons and there are $|E|$ leaves.*

*Let the vertices of $T$ be labeled as follows:*

1. *The root of $T$ is labeled $(u,v)$.*
2. *For each edge $(x,y) \in E$, there exists exactly one leaf labeled $(x,y)$ in $T$.*
3. *For each internal vertex $t \in V_T$ labeled $(x,y)$, the roots of the left and right subtrees of $t$ are labeled either $(x,y),(x,y)$ or $(x,z),(z,y)$ respectively, where $z \in V$.*

*We call $T$ the* parse tree *of $G$ [7].*

Our algorithm is based upon the dynamic programming approach which recursively applies the rules described above to merge two edge sets into a new edge set, from the bottom of the parse tree to the root. Let $T$ be a parse tree.

We first define some procedures which will be used in the main algorithm.

### Procedure INIT

Input: A subgraph $N$ which only contains an edge $(x, y)$.
Output: $H_i(N), i = 1, 2, \ldots, 7$.
Method: $H_1(N) := \{(x, y)\}$.
$H_i(N) :=$ UNDEF, for $i = 2, 3, 4, 5, 6, 7$.
End. $\diamondsuit$

### Procedure SCON

Input: A series-parallel graph $N$ that is a series connection of two series-parallel graphs $G_1$ and $G_2$, whose terminals are $(x, z)$ and $(z, y)$, respectively.
Output: $H_i(N), i = 1, 2, \ldots, 7$.
Method: For i = 1 to 7
    Apply Rule $S_i$ to obtain $H_i(N)$.
Next
End. $\diamondsuit$

### Procedure PCON

Input: A series-parallel graph $N$ that is a parallel connection of two series-parallel graphs $G_1$ and $G_2$, whose terminals are $(x, y)$ and $(x, y)$.
Output: $H_i(N), i = 1, 2, \ldots, 7$.
Method: For i = 1 to 7
    Apply Rule $P_i$ to obtain $H_i(N)$.
Next
End. $\diamondsuit$

After traversing the parse tree $T$ in bottom-up order, and computing all $H_i(G)$, we will be able to determine whether $G$ is Hamiltonian or not.

### Procedure SCAN

Input: $H_i(G)$ for $i = 1, 2, 3, 4$.
Output: A Hamiltonian path in $G$ if there exists one. Otherwise report that $G$ has no Hamiltonian path.
Method: For i = 1 to 4
    If $H_i(G)$ exists Then
        Report $H_i(G)$ as the solution.
        Return
    End If
End For
Report that $G$ has no Hamiltonian path.
End. $\diamondsuit$

With these procedures defined, our algorithm is presented as below.

### Algorithm HPSP

**Input:** A series-parallel graph $G = (V, E)$ with terminals $u$ and $v$.
**Output:** A Hamiltonian path in $G$ if there exists one. Otherwise report that $G$ has no Hamiltonian path.
**Method:**
Step 1: Construct a parse tree $T$ of $G$.
Step 2: $T' \leftarrow T$.
Step 3: Do While $(T' \neq \emptyset)$;
    Choose a leaf node $N$ of $T'$ and delete it from $T'$.
    Case 1. $N$ corresponds to an edge $(x, y)$:
        Apply INIT$(x, y)$.
    Case 2. $N$ corresponds to a series connection of two subgraphs $G_1$ and $G_2$, whose terminals are $x, z$ and $z, y$ respectively:
        Apply SCON.
    Case 3. $N$ corresponds to a parallel connection of two subgraphs $G_1$ and $G_2$, whose terminals are $x, y$ and $x, y$ respectively:
        Apply PCON.
    end do while;
Step 4: Apply SCAN to $H_i(G)$ for $i = 1, \ldots, 4$.
End. $\diamondsuit$

## 4 Correctness of Algorithm HPSP

We have presented our algorithm. In the following, we shall prove the correctness of the algorithm. First of all, let us define some notations.

**Definition 3** *If $G$ consists of exactly one edge and two vertices, then $H_0(G) := \emptyset$ and we say $G$ has an $H_0$ type edge set. Otherwise, $H_0(G) :=$ UNDEF.*

This notation is introduced to allow a symmetric representation in the following lemmas. For example, let $N$ be a parallel connection of $G_1$ and $G_2$. If $G_1$ is a single edge, then $H_i(N) = H_i(G_2), \forall i \in \{1, \ldots, 7\}$. Now we can also write it as $H_i(N) = H_0(G_1) \cup H_i(G_2), \forall i \in \{1, \ldots, 7\}$. All of Rule $S_i$'s and Rule $P_i$'s are in the form of $H_i(N) = H_j(G_1) \cup H_k(G_2)$ for some $j \in \{0, 1, 2, \ldots, 7\}$ and $k \in \{0, 1, 2, \ldots, 7\}$.

**Definition 4** *Given a vertex $x$, we use the notation $\deg_P(x)$ to represent the number of edges in $P$ which cover the vertex $x$.*

**Lemma 1** *If $N$ is a series connection of two series-parallel graphs $G_1$ and $G_2$, then $N$ has an $H_i$ type* *edge set if and only if there exist some $j$ and $k$ such that $H_i(G) = H_j(G_1) \cup H_k(G_2)$ in Rule $S_i$, and $G_1$ has an $H_j$ type edge set and $G_2$ has an $H_k$ type edge set.*

**Lemma 2** *If $N$ is a parallel connection of two series-parallel graphs $G_1$ and $G_2$, then $N$ has an $H_i$ type edge set if and only if there exist some $j$ and $k$ such that $H_i(G) = H_j(G_1) \cup H_k(G_2)$ in Rule $P_i$, and $G_1$ has an $H_j$ type edge set and $G_2$ has an $H_k$ type edge set.*

With Lemma 1 and Lemma 2, we are now ready for the proof of our main theorem.

**Theorem 1** *A series-parallel graph $G = (V, E)$ has an $H_i$ type edge set if and only if $H_i(G)$ obtained by our algorithm exists.*

**Proof.** By Lemma 1 and Lemma 2.

**Theorem 2** *If there is a Hamiltonian path on a given series-parallel graph, then our algorithm will find it.*

**Theorem 3** *According to Theorem 2, Algorithm HPSP is correct in finding a Hamiltonian path on a given series-parallel graph $G = (V, E)$ in time $O(|V|)$.*

**Proof.** In [7], it was shown that Step 1 takes $O(|V|)$ time. In each iteration of Step 3, the execution of INIT, SCON, or PCON takes O(1) time. There are $O(|E|)$ leaves in the parse tree $T$. Since $T$ is a binary tree, there are $O(|E|)$ nodes in $T$, Step 2 and Step 3 both require $O(|E|)$ time. The SCAN in Step 4 takes $O(1)$ time. Therefore, Steps 1–4 totally take $O(|E|)$ time. Since in planar graphs, $|E| \leq 3|V| - 6$, we obtain the time complexity of HPSP as $O(|V|)$.

## 5 Conclusion and Future Work

In this paper, we have shown that the Hamiltonian path problem can be solved in linear time on series-parallel graphs. It will be interesting to extend our method to other special graphs such as chordal graphs, partial 2-trees, generalized series-parallel graphs, etc. The Hamiltonian path problem we considered in this paper is a decision problem. It is also interesting to consider the Hamiltonian-completion problem, which is to find the minimum number of edges to be added into a given graph to make it have a Hamiltonian path.

## References


[1] F. T. Boesch, S. Chen and J. A. M. McHugh "On covering the points of a graph with point disjoint paths," *Lecture Notes in Math.* Vol. 46, Springer: Berlin, (1974) 201–212.

[2] G. Chartrand and O. R. Oellermann, *Applied and Algorithmic Graph Theory*, McGraw-Hill: Singapore, (1993) 229–237.

[3] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman: San Francisco, CA, (1979) 199–200.

[4] S. Isobe, X. Zhou and T. Nishizeki, "A Polynomial-Time Algorithm for Finding Total Colorings of Partial k-Trees," *International Journal of Foundations of Computer Science,* Vol. 10 No. 2 (1999) 171–194.

[5] D. S. Johnson, "The NP-Completeness Column: an Ongoing Guide," *Journal of Algorithms* **6** (1985) 434–451.

[6] B. W. Kernighan and R. Pike, *The Unix Programming Environment*, Prentice-Hall, Inc. Englewood Cliffs, NJ, (1984) 143–144.

[7] T. Kikuno, N. Yoshida and Y. Kakuda, "A linear algorithm for the domination number of a series-parallel graph," *Discrete Applied Mathematics* **5** (1983) 299–311.

[8] C.L. Lu, M.T. Ko and C.Y. Tang, "Weighted Perfect and Efficient Edge Domination in Graphs," manuscript, Department of Computer Science, National Tsing Hua University, Hsinchu 300, Taiwan.

[9] K. Takamizawa, T. Nishizeki, and N. Saito, "Linear-time computability of combinatorial problems on series-parallel graphs," *Journal of the Association for Computing Machinery,* **29** (1982) 623-641.

[10] C. C. Yen and R. C. T. Lee, "A linear time algorithm to solve the weighted perfect domination problem in series-parallel graphs", *European Journal of Operational Research* (1994) 192–198.