Future Generation Computer Systems 29 (2013) 520–527

Contents lists available at SciVerse ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A new light-based solution to the Hamiltonian path problem

Javad Salimi Sartakhti, Saeed Jalili *, Ali Gholami Rudi

SCS lab, Computer Engineering Department, Electrical and Computer Engineering Faculty, Tarbiat Modares University, Tehran, Iran

ARTICLE INFO

Article history:
Received 23 December 2011
Received in revised form
16 July 2012
Accepted 20 July 2012
Available online 4 August 2012

Keywords:
Unconventional computing
Optical computing
Light-based computing
Hamiltonian path
NP-complete problem

ABSTRACT

The exponential running time for algorithms designed to solve NP-complete problems in conventional computers, mostly, makes it almost impossible solving large instances of such problems in reasonable time. Unconventional computers may be the answer to this problem. In this paper we use light to solve a Hamiltonian path problem in polynomial time. At first, the solution space of the problem is generated and then invalid solutions are eliminated from it. Our approach is based on filters which remove paths that are not Hamiltonian from the set of possible solutions. We perform polynomial preprocessing steps to produce the filters and find Hamiltonian paths of a graph based on the rays of light passing through them. Finally we report the design of filters and use light to solve the Hamiltonian path problem.

1. Introduction

NP-complete problems are one of the main challenges of today's computers. Many studies have been carried out to find efficient algorithms for solving NP-complete problems. Although efficient algorithms for special cases of the problems and algorithms have been proposed to find approximate solutions [1,2], it is believed that, at least on conventional computers, no polynomial algorithm exists to solve these problems. Interestingly unconventional computers do not share some of these limitations and can solve NP-complete problems in polynomial time.

The research on unconventional computing tries to find revolutionary new algorithms and physical implementations. Unconventional computers have efficiently solved many of hard-to-solve problems for conventional computers. To name a few, Quantum computing [3], DNA computing [4], Soap bubbles [5], Gearbased computers [6], Adiabatic algorithms [3] are some unconventional approaches.

Although DNA computing is more dominant in solving NP-complete problems [7–9], light-based computing is another motivating and interesting area for researchers. Various properties of light, such as massive parallelism, enable us to solve some real world problems more efficiently than conventional computers [10]. The first attempt for the optical computers was made in 1929 by Tauschek who obtained a patent on Optical Character Recognition (OCR) machines which were used as a template for matching characters [11].

Afterwards, researchers have used light properties to offer efficient solutions for problems that cannot be solved efficiently by conventional computers. Some examples are discrete Fourier Transform Computation performed in constant time [12,13], very fast processors for vector-matrix multiplications and the first continuous wave all-silicon laser which is used in some real applications [14].

Recently, researchers have presented light-based solutions and devices to deal with the problems which could not be efficiently handled by conventional computers. They are categorized as NP-Complete or NP-Hard problems. For example, the 3-SAT problem [15,16], the exact cover problem [17], the Hamiltonian path problem [15,18], the subset sum problem [19,20], the vertex cover problem, the maximum clique problem, 3D-matching, and the traveling salesman problem [15] are many NP-complete problem instances that have been recently solved using physical properties of light. Also, some researchers use various optical components for digital computing [21,22].

2. Problem definition

Given a graph $G = (V, E)$, the path P is Hamiltonian if it passes every vertex exactly once. In other words, all paths of length $|V| - 1$ in a graph G are Hamiltonian, if all vertices in the path are distinct. For instance, in Fig. 1 the path $v_0\ v_1 v_2\ v_3\ v_4$ is a Hamiltonian path as well as $v_0\ v_1\ v_4 v_3\ v_2$.

The problem of finding the Hamiltonian path of a graph or deciding whether a graph has a Hamiltonian path is NP-complete [23]. There are two types of Hamiltonian path solutions

* Corresponding author.
E-mail addresses: salimi.sartakhti@gmail.com (J.S. Sartakhti), sjalili@modares.ac.ir (S. Jalili), gholamirudi@modares.ac.ir (A.G. Rudi).

0167-739X/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2012.07.008

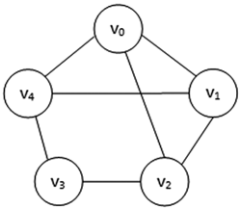

Fig. 1. Graph G that includes some Hamiltonian paths such as $v_0 v_1\ v_2\ v_3\ v_4$.

in conventional computers: the first one is a non-exact algorithm and the second one is an exact algorithm. Some non-exact solutions such as [24,25] have been presented for the Hamiltonian path problem, but they do not necessarily find exact solutions. Also, although in exact algorithms several methods have been represented for the Hamiltonian path problem [26], a polynomial resource solution for it has not been found yet. In unconventional computers, some solutions are proposed to the Hamiltonian path problem using optical computing and DNA computing approaches [4,15,18]. In this paper we propose a new light-based solution to the Hamiltonian path problem.

The rest of this paper is organized as follows: In Sections 3 and 4 we present the design and implementation models of our solution to the Hamiltonian path problem, respectively. Solving a Hamiltonian path problem instance using our solution is explained in Section 5 while the complexity analysis of our solution is presented in Section 6. The conclusion of this study is addressed in Section 7.

3. Design model

In this section we propose a new optical solution to the Hamilton path problem. First, the solution model is presented in three steps. In step 1, we explain how to represent the solution space of the Hamiltonian path problem as binary sequences; in steps 2 and 3, we design some filters to eliminate the sequences that do not satisfy Hamiltonian path conditions.

Fig. 2 shows our design model to find all Hamiltonian paths. In our model strategy, the potential solution space of the problem is firstly generated and then all non-Hamiltonian paths are eliminated from the space by some designed filters.

Step 1. *Hamiltonian path representation*.

In the graph $G = (V, E)$ where $|V| = N$, each vertex can be represented by $\lceil \log(N) \rceil$ bits. A Hamiltonian path in graph G is specified by N vertices. Therefore, a binary sequence of $N \times \lceil \log(N) \rceil$ bits can represent the Hamiltonian path. For example, to represent each vertex of graph G in Fig. 1, three bits are required. Thus, to indicate the solution space of the problem, all binary sequences of length 15 bits should be generated.

Each potential Hamiltonian path is represented by the following binary sequence:

$$\underbrace{S_{N\log(N)-1}\cdots S_{(N-1)\log(N)+1}S_{(N-1)\log(N)}}_{P_{N-1}}\cdots\underbrace{S_{2\log(N)-1}\cdots S_{\log(N+1)}S_{\log(N)}}_{P_1}\underbrace{S_{\log(N)-1}\cdots S_1 S_0}_{P_0}\cdots.$$

The above binary sequence is composed of N parts where P_i indicates the ith vertex in the Hamiltonian path of the graph G. In general, there are $2^{N\times\lceil\log(N)\rceil}$ possible value assignments to a sequence of length $N \times \lceil \log(N) \rceil$. Each value assignment can represent a possible solution of the Hamiltonian path problem. For example the "100011010001000" sequence indicates Hamiltonian path $v_0 \to v_1 \to v_2 \to v_3 \to v_4$ for graph G in Fig. 1.

Next, we should eliminate all solutions indicating non-Hamiltonian paths, from the solution space of the problem. We classify non-Hamiltonian paths into three types; each one is addressed in one of the steps 2–4.

Step 2. *Out of bound sequence filtering*.

Having generated binary sequences, we design some filters to eliminate all binary sequences which contain at least one vertex out of G's vertices (invalid vertices). We call these sequences 'out of bounds sequences'. If a sequence contains at least one invalid vertex, it does not represent a real Hamiltonian path. For a graph $G = (V, E)$ such that $|V| = N \neq 2^{\lceil\log(|V|)\rceil}$, some generated sequences certainly contain vertices that do not belong to V (e.g., in Fig. 1, $|V| = 5 \neq 2^3$, so $\{v_5, v_6, v_7\} \notin V$). So, out of bounds sequence filters eliminate all binary sequences that contain at least one of $\{v_n,\ v_{n+1},\ \ldots,\ v_{2^{\lceil\log(n)\rceil}}\}$ vertices. For example, in Fig. 1, to represent all potential Hamiltonian paths we need binary sequences of length $= 5 * 3 = 15$ bits. Since $\{v_5, v_6, v_7\}$ do not belong to graph vertices, all binary sequences including "101", "110" and "111" vertices are out of bounds sequences.

Definition 1. Suppose M is a set of generated binary sequences corresponding to potential Hamiltonian paths of graph $G = (V, E)$. $\Phi(i, X,\ M)$ returns all sequences belonging to M, where $P_i \neq X$, $(0 \leq i < N, 0 \leq X < 2^{\lceil\log(N)\rceil}$ and $X \in \mathbb{N})$. The result of Φ are all binary sequences which do not contain the vertex X in part i.

Definition 2. The intersection of two filters Φ_1 and Φ_2 is defined as an operator which allows sequences that have neither been eliminated by Φ_1 or Φ_2 to pass. This operator is indicated by $\cap$ in this paper.

Algorithm 1 uses the Φ filter to eliminate all out of bounds sequences, according to Definitions 1 and 2.

Algorithm 1: Out of Bound Sequences (OBS) filter.

```
Procedure OBS filter (M){
/* M contains all generated binary sequences of length  N × ⌈log (N) ⌉ */
1.  Ω = M
2.  For i= N to 2^⌈log(N)⌉  /* {v_N, v_N+1, ..., v_2^⌈log (N)⌉} */
3.      For each part j  of the binary sequences
4.          Ω←Φ(j, i, M) ∩ Ω
5.      end For
6.  end For
7.  Return  Ω  }
```

In Algorithm 1, for each invalid vertex i.e., v_i, and a part of the sequences i.e., P_j, a filter is produced which eliminates binary sequences including v_i in its part P_j and then applies the intersection operation on the filter and the result filter of the previous iteration.

Step 3. *Filtering sequences with duplicate vertices*.

In this step, sequences with duplicate vertices are eliminated. Since a Hamiltonian path should not include duplicate vertices, the sequences with duplicate vertices represent paths that are not Hamiltonian, so they must be eliminated.

Definition 3. The negation of Φ is defined as $\overline{\Phi}(i, X,\ M)$ in which the ith part of all sequences belonging to set M is not equal to X must be eliminated. The result of this filter is binary sequences that their $P_i = X$.

Definition 4. The union of two filters Φ_1 and Φ_2 is defined as an operator which allows passing sequences that are not eliminated in at least one of Φ_1 and Φ_2 filters. This operator is indicated by $\Phi_1 \cup \Phi_2$.

Algorithm 2 indicates how to eliminate all sequences including duplicate vertices, with respect to Definition 4.

Algorithm 2: sequences with duplicate vertices filter (SVF)

```
Procedure  SVF(M){
/* M is output of Algorithm 1. */
1.  Ω = M
2.  For each vertex of G such as v_i
3.      For each part j from 0 to N-2
4.          For each part k from j+1 to N-1
5.              Ω ←(Φ(j, i, M) ∪ Φ(k, i, M)) ∩ Ω
6.          end For
7.      end For
8.  end For
9.  return  Ω }
```