Available online at www.sciencedirect.com

SCIENCE DIRECT

Journal of Computer and System Sciences 71 (2005) 406–439

JOURNAL OF COMPUTER AND SYSTEM SCIENCES

www.elsevier.com/locate/jcss

A complete and equal computational complexity classification of compaction and retraction to all graphs with at most four vertices and some general results☆

Narayan Vikas

School of Computing Science, Simon Fraser University, Burnaby, British Columbia, Canada V5A 1S6

Received 15 December 2003

Available online 23 May 2005

Abstract

A very close relationship between the compaction, retraction, and constraint satisfaction problems has been established earlier providing evidence that it is likely to be difficult to give a complete computational complexity classification of the compaction and retraction problems for reflexive or bipartite graphs. In this paper, we give a complete computational complexity classification of the compaction and retraction problems for all graphs (including partially reflexive graphs) with four or fewer vertices. The complexity classification of both the compaction and retraction problems is found to be the same for each of these graphs. This relates to a long-standing open problem concerning the equivalence of the compaction and retraction problems. The study of the compaction and retraction problems for graphs with at most four vertices has a special interest as it covers a popular open problem in relation to the general open problem. We also give complexity results for some general graphs. The compaction and retraction problems are special graph colouring problems, and can also be viewed as partition problems with certain properties. We describe some practical applications also.

Keywords: Computational complexity; Graph; Colouring; Homomorphism; Retraction; Compaction; Partition

☆ A version presenting the results of this paper appears in Proceedings of the Tenth Annual International Computing and Combinatorics Conference (COCOON), 2004.

E-mail address: vikas@cs.sfu.ca.

0022-0000/$ - see front matter © 2004 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcss.2004.07.003

1. Introduction

A version presenting the results of this paper appears in [29]. We first introduce the following definitions and problems, and then describe the motivation and results.

1.1. Definitions

The pair of vertices forming an edge in a graph are called the *endpoints* of the edge. An edge vv with the same endpoints in a graph is called a *loop*, and the vertex v is said to have a loop. A *reflexive graph* is a graph in which every vertex has a loop. An *irreflexive graph* is a graph which has no loops. Any graph, in general, is a *partially reflexive graph*, meaning that its individual vertices may or may not have loops. Thus reflexive and irreflexive graphs are special partially reflexive graphs. A *bipartite graph* G is a graph whose vertex set can be partitioned into two distinct subsets G_A and G_B, such that each edge of G has one endpoint in G_A and the other endpoint in G_B; we say that (G_A, G_B) is a bipartition of G. Thus a bipartite graph is irreflexive by definition. When we do not mention the terms reflexive, irreflexive, or bipartite, the corresponding graph may be assumed to be a partially reflexive graph. An edge is said to be *incident* with a vertex v in a graph if v is an endpoint of the edge. A vertex u is said to be *adjacent* to a vertex v in a graph if uv is an edge of the graph; if u is adjacent to v then v is also adjacent to u. If a vertex u is adjacent to a vertex v in a graph then u is said to be a *neighbour* of v, and v is said to be a neighbour of u in the graph. The *neighbourhood* of a vertex v in a graph, denoted as $Nbr(v)$, is the set of all neighbours of v in the graph (note that for a loop vv, we have $v \in Nbr(v)$). A graph in which each pair of distinct vertices are adjacent is called a *complete graph*. We denote an irreflexive complete graph with k vertices by K_k. A bipartite graph, with bipartition (G_A, G_B), in which every vertex in G_A is adjacent to every vertex in G_B is called a *complete bipartite graph*. A *path* of length $k-1$ is a graph containing k distinct vertices, say $v_0, v_1, v_2, \ldots, v_{k-1}$, such that $v_0v_1, v_1v_2, \ldots, v_{k-2}v_{k-1}$ are all the non-loop edges of the graph, $k \geqslant 1$; we may write such a path as $v_0v_1v_2 \ldots v_{k-1}$, where the vertex v_0 is called the *origin* or the *first vertex* of the path, the vertex v_{k-1} is called the *terminus* or the *last vertex* of the path, and the vertices $v_1, v_2, \ldots, v_{k-1}$ are called the *internal vertices* of the path. A *cycle* of length k, called a *k-cycle*, is a graph containing k distinct vertices, say $v_0, v_1, v_2, \ldots, v_{k-1}$, such that $v_0v_1, v_1v_2, \ldots, v_{k-2}v_{k-1}, v_{k-1}v_0$ are all the non-loop edges of the graph, $k \geqslant 3$; we may write such a cycle as $v_0v_1v_2 \ldots v_{k-1}v_0$. A *square* will be used as a synonym for a 4-cycle. A *triangle* will be used as a synonym for a 3-cycle. A *walk* of length n in a graph is a sequence of vertices $v_0v_1v_2 \ldots v_n$ not necessarily distinct such that v_iv_{i+1} is an edge of the graph, for all $i = 0, 1, 2, \ldots, n-1, n \geqslant 0$; we say that such a walk is from v_0 to v_n. For a graph G, we use $V(G)$ and $E(G)$ to denote the vertex set and the edge set of G respectively. The *size* of a graph is the number of vertices plus the number of edges in the graph. We define min S and max S to give the minimum and the maximum element respectively in a set S. When a set S is an argument of a mapping f, we define $f(S) = \{f(s) | s \in S\}$. If a set has only one vertex, we may just write the vertex instead of the set.

Let G be a graph. A vertex v of G is said to be an *isolated vertex* of G, if v is not adjacent to any other vertex v' of G, $v \neq v'$ (note that an isolated vertex may have a loop). Two vertices u and v of G are said to be *connected* in G, if there exists a path from u to v in G; otherwise u and v are said to be *disconnected* in G. The *distance* between a pair of vertices u and v in G, denoted as $d_G(u, v)$ or $d_G(v, u)$, is the length of a shortest path from u to v in G, if u and v are connected in G; we define $d_G(u, v)$ (and $d_G(v, u)$) to be infinite, if u and v are disconnected in G. The *diameter* of G is the maximum