What are the problems solved by this paper?

To better understand the dynamic behavior of a program, it is useful to record the paths (or traces) that are followed during the execution of the program. Previous path profiling techniques captured only acyclic paths which are short snippets of the execution of the program and end at the boundaries of loops and procedures.

This paper shows a technique by which the entire control flow of the program can be captured — including loop iterations and inter-procedural paths. However, such a complete trace can be very large and difficult to process. Hence, this paper introduces a technique by which the complete trace can be compressed into a compact but useful form. This form is called the Whole Program Path (WPP).

The main aim for profiling the program execution is to identify the fragments of code in a program which consume most of the execution time, so that the compiler can target these fragments and optimize them. This paper shows how such hot sub-paths can be identified in the WPP.

What are the approaches attempted by this paper?

The WPP is created in two steps. In the first step, an instrumented program is executed to record the acyclic paths that are executed. A path can be uniquely identified by using an accumulator that is incremented by some predetermined amount along a select set of edges in the CFG. This identifier is stored along with the trace of the paths. Edges leading to basic blocks containing procedure calls terminate an acyclic path, so that the call site can be recorded in the trace.

The resulting trace is compressed using a string compression algorithm called SEQUITUR. Since SEQUITUR does not produce minimal representation (grammar) of a string, a minor enhancement to the basic algorithm is proposed, which provides better compression. A tool called PPCompress (which uses SEQUITUR) is used to compress the acyclic path trace and obtain the WPP.

An algorithm is proposed to find the hot sub-paths using the WPP.

What are the main conclusions of this paper?

- It is found that the technique to find WPP is feasible even for commercial applications.
- The overheads associated with the instrumentation and the WPP processing are acceptable.
- An advantage is that the traces include calls to library routines.
- If the program is multithreaded, a separate WPP is created for each thread.
- The compression ratio gives a measure of the program’s control-flow regularity.
- The number of hot sub-paths with long lengths (length = number of acyclic paths in the hot sub-path) is not very large. This is a good thing, since compilers can focus on smaller fragments of code for optimizations.
• However, this was less true of commercial applications, and so the compiler should have a 'larger perspective' while performing optimizations on such programs.

• The earlier observation that data compression gives an upper bound on the performance of branch prediction is found to be valid.

• These techniques can be valuable for the global and inter-procedural optimizations that are required to support highly speculative or VLIW microprocessors.