Discrete Optimization 22 (2016) 37–65

Contents lists available at ScienceDirect

Discrete Optimization

www.elsevier.com/locate/disopt

An integer programming approach to optimal basic block instruction scheduling for single-issue processors

Michael Jünger, Sven Mallach*

Universität zu Köln, Institut für Informatik, Albertus-Magnus-Platz, 50923 Köln, Germany

ARTICLE INFO

Article history:
Available online 7 December 2015

MSC:
68M07
68M20
68N20
68W01
90C10
90C27

Keywords:
Instruction scheduling
Integer programming
Branch-and-cut

ABSTRACT

We present a novel integer programming formulation for basic block instruction scheduling on single-issue processors. The problem can be considered as a very general sequential task scheduling problem with delayed precedence constraints. Our model is based on the linear ordering problem and has, in contrast to the last IP model proposed, numbers of variables and constraints that are strongly polynomial in the instance size. Combined with improved preprocessing techniques and given a time limit of ten minutes of CPU and system time, our branch-and-cut implementation is capable to solve all but eleven of the 369,861 basic blocks of the SPEC 2000 integer and floating point benchmarks to proven optimality. This is competitive to the current state-of-the art constraint programming approach that has also been evaluated on this test suite.

1. Introduction

Today most computer programs are written in high-level programming languages and developers rely on compilers in order to generate executable machine code for various operating systems and processor architectures. One of the fundamental subroutines of any compiler is the instruction scheduling phase where the generated machine instructions shall be ordered such that the number of processor clock cycles needed to complete all the operations is minimized. Modern processor architectures are pipelined, i.e., the execution of a single machine instruction is partitioned into several stages [1]. As a result, multiple instructions can be in flight, occupying different stages at the same time. However, in practice, the ideal flow of instructions through the pipeline may be disturbed by several conflicts, especially by such caused by data dependencies between the instructions. Therefore, each precedence relationship is associated with a *latency* that captures the number of clock cycles needed until the result computed by the first instruction is available to its successor. The starting times of successor instructions must obey these latencies to ensure that no conflicts

* Corresponding author.
E-mail addresses: mjuenger@informatik.uni-koeln.de (M. Jünger), mallach@informatik.uni-koeln.de (S. Mallach).

http://dx.doi.org/10.1016/j.disopt.2015.10.003
1572-5286/

occur and all operands are present in logic when they execute. These *latency constraints* make instruction scheduling an $\mathcal{NP}$-hard combinatorial optimization problem [2], even for *single-issue* processors that allow only at most one instruction to be inserted into the pipeline (*issued*) in every clock cycle. Polynomial-time solvability is known only for the very restrictive case that the maximum occurring latency is one clock cycle [3].

In this article, we focus on the exact solution of the basic block instruction scheduling problem for single-issue processors using integer programming. A *basic block* is a set of related instructions without any internal branches such that all the instructions need to be scheduled as a straight-line sequence after entering the basic block and before exiting it again. Production compilers typically rely on a *list scheduling* heuristic, a method maintaining a list of instructions ready to be scheduled and iteratively selecting a ready instruction with the highest among some pre-determined priorities. For the problem under consideration, Bernstein, Rodeh, and Gertner [4] showed that any list schedule is no worse than $2 - \frac{1}{L+1}$ times the optimum where L is the maximum latency occurring. Many computational experiments, e.g. [5–8], reveal near-optimal performance of list scheduling when averaging results over a particular set of instances. The fine-grained results in [6,7] show however, that the number of basic blocks where list scheduling does not find optimal schedules grows significantly with increasing size of the instances. Moreover, provably optimal schedulers are desirable to enable quality measures and as well in settings where the runtime performance of the final programs is critical or where longer compile times are tolerable. This is the case, e.g., for embedded and digital signal processing applications or, in general, software that is pre-compiled only once before (mass) delivery.

Early branch-and-bound [9,10], integer programming (IP) [11–13] and constraint programming (CP) [5] approaches were limited to small sets of instances with roughly up to 50 instructions. The most recent contribution to attack the instruction scheduling problem with integer programming was given by Wilken, Liu and Heffernan [8] in 2000. They were the first to optimally schedule a larger set of basic blocks with up to 1000 instructions by applying some search space reduction techniques and problem-specific cutting plane separation. However, their experiments were restricted to instances with latencies in the range between zero and two clock cycles and later it was shown that the method is not as successful on more realistic instances with larger and varying latencies [14]. Even more, its scalability is limited since the numbers of variables and constraints are pseudo-polynomial as they depend on (an upper bound on) the makespan. An important result of their work is however that the reduction techniques are essential to be able to schedule real-world instances to optimality. One year later, van Beek and Wilken [15] proposed a constraint programming approach that could optimally schedule the instances used for the experiments by Wilken, Liu and Heffernan even faster. After Heffernan and Wilken then proposed a set of methods to even more effectively reduce the search space of basic block instances [16] in 2005, Malik, McInnes, and van Beek [6] were able to improve their CP approach to solve the problem also for multiple-issue processors on an even larger set of instances (about 350,000 basic blocks with up to 2600 instructions). While the previous solvers from [8,15] could not solve hundreds of these instances to optimality [14], there is only one instance that could not be solved by the CP solver within a time limit of ten minutes of CPU and system time for single-issue processors in our experiments. Notably, also in [6], the authors emphasize that their search space reductions are key to the success of their solver.

In this paper, we present the first integer programming approach that is competitive to the CP method of Malik, McInnes and van Beek for single-issue processors. It is also the first IP model that is based on the linear ordering problem. With the exception of scheduling models that employ exactly one general integer ('completion time') variable per instruction (see Section 5), it is also the first model whose numbers of variables and constraints are strongly polynomial in the size of the instance. Our corresponding implementation is able to solve all but eleven instances of the mentioned benchmark set to optimality within ten minutes of CPU and system time and is faster on some particular instances. We highlight the most important existing search space reduction techniques that are indeed not specific to CP, and we found that