Science of Computer Programming 98 (2015) 484–515

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Approximate by thinning: Deriving fully polynomial-time approximation schemes

Shin-Cheng Mu[a,*], Yu-Han Lyu[b], Akimasa Morihata[c]

[a] *Institute of Information Science, Academia Sinica, Taiwan*
[b] *Department of Computer Science, Dartmouth College, USA*
[c] *Graduate School of Arts and Sciences, University of Tokyo, Japan*

ARTICLE INFO

Article history:
Received 5 May 2011
Received in revised form 23 June 2014
Accepted 4 July 2014
Available online 18 July 2014

Keywords:
Approximation algorithms
Program derivation

ABSTRACT

The *fully polynomial-time approximation scheme* (FPTAS) is a class of approximation algorithms for optimisation problems that is able to deliver an approximate solution within any chosen ratio in polynomial time. By generalising Bird and de Moor's *Thinning Theorem* to a property between three orderings, we come up with a datatype-generic strategy for constructing fold-based FPTASs. Greedy, thinning, and approximation algorithms can thus be seen as a series of generalisations. Components needed in constructing an FPTAS are often natural extensions of those in the thinning algorithm. Design of complex FPTASs is thus made easier, and some of the resulting algorithms turn out to be simpler than those in previous works.

1. Introduction

Program derivation aims to develop and advocate a methodology with which programs can be formally constructed from their specifications. A particularly fruitful series of results was Bird and de Moor's relational, datatype-generic approach for solving optimisation problems [1]. An optimisation problem is specified as a composition $\mathit{max}\,(\trianglelefteq) \circ \mathit{solutions}$ where *solutions* generates all solution candidates, and $\mathit{max}\,(\trianglelefteq)$ picks an optimal solution under a total preorder $\trianglelefteq$ (the notations will be explained in Section 2). It was shown that if *solutions* can be written as a relational fold whose step relation is monotonic with respect to $\trianglelefteq$, there is a greedy algorithm solving the problem. The monotonicity condition might not always hold, but for many problems the function *solutions* provides some hint as to how to find a non-total sub-ordering of $\trianglelefteq$ for which the monotonicity condition does hold. In this case we have a so-called "thinning" algorithm (the "fold" counterpart of dynamic programming). Bird and de Moor's work not only hinted at a methodology to construct efficient algorithms to optimisation problems whose proofs are developed together with the programs, but also made the relationship between different classes of algorithms formal and clear.

Consider, however, the following optimisation problem. Given is a tree, as shown in Fig. 1(a), in which each node is either a *supplier*, drawn as a square, or a *demander*, drawn as a circle, each labelled by a number. Think of each supplier as a power plant, and demanders as consumers connected to the power plant, directly or indirectly, through the edges. The goal is to partition the tree into subtrees such that each tree either contains no supplier at all, or contains exactly one supplier whose amount of electricity generated is large enough to fulfil all demands in the subtree. The goal is to maximise fulfilled demands. In Fig. 1(b), the tree is partitioned into 5 parts. The leftmost partition consists of one supplier labelled 9

* Corresponding author.
E-mail addresses: scm@iis.sinica.edu.tw (S.-C. Mu), yuhanlyu@cs.dartmouth.edu (Y.-H. Lyu), morihata@graco.c.u-tokyo.ac.jp (A. Morihata).

http://dx.doi.org/10.1016/j.scico.2014.07.001
0167-6423/© 2014 Elsevier B.V. All rights reserved.

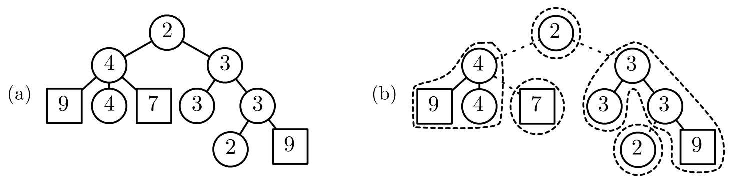

Fig. 1. (a) A tree to be partitioned. (b) A partition with fulfilment 17.

and two demanders labelled 4, and totally 8 units of demand are fulfilled. The rightmost partition fulfils 9 units of demand from three nodes labelled 3. The three singleton trees respectively consist of one isolated supplier and two demanders, and nothing is fulfilled. The total fulfilment of this partition is thus 17, which is also the maximum possible for this tree.

Despite being a simplification of an obvious graph version, this problem of *partitioning trees of supply and demand*, discussed by Ito et al. [2], is already NP-hard.[1] If one agrees to sacrifice a little precision for speed, however, there exists a polynomial-time algorithm that computes a partition whose fulfilment is guaranteed to be at worst within a specified ratio of the optimal partition. Can we derive such an approximation algorithm for this problem?

Curiously, however, a formal treatment of approximation algorithms appears to be an incomplete area in the jigsaw puzzle of theories of program derivation. The present paper is an attempt to start filling in some of the missing pieces.

Approximation algorithms One may be interested in approximation algorithms out of practical concern: by sacrificing some precision, one may find provably near-optimal solutions to NP-hard problems in polynomial time. Besides, approximation algorithms are of interest to theoreticians as well. It is believed that some NP-hard problems can be approximated within any factor, some can be approximated by a constant, yet some are impossible to approximate within any constant at all, unless P = NP. Such classification could therefore reveal important clues to tackle some long-standing theoretical issues.

The *fully polynomial-time approximation scheme* (FPTAS) [4–7] is a class of approximation algorithms that delivers an approximate solution no worse than the optimal solution within any chosen ratio in time polynomial in the size of the input and the inverse of the ratio (precise definition to be given in Section 4). Structures of some FPTASs are surprisingly similar to the thinning strategy of Bird and de Moor specialised to lists, which lead us to believe that the formal development can be extended to approximation algorithms, and that FPTASs can be naturally extended to more datatypes.

Contributions As it turns out, if it is known how to solve an optimisation problem using a thinning strategy, an FPTAS can often be constructed as a fold by extending the ordering used for thinning. Our contributions include:

- We propose a generalisation of the Thinning Theorem, from one preorder to three relations, to prove correctness of fold-based FPTASs. We have thus filled in another piece of the algorithm taxonomy, seeing the route from greedy, to thinning, to approximation algorithms as a series of generalisations.
- We present an algebraic formalism for a certain class of approximation algorithms whose correctness is otherwise not trivial to prove. The formalisation is datatype generic, which means that it deals with not only problems that take lists as inputs, but all regular datatypes. Our formalism is, to the best of our knowledge, the first datatype-generic formulation of FPTASs. Yet it is simpler in structure than some existing FPTAS formulations.
- The formalism suggests a methodology for developing FPTASs, often considered a non-trivial task, by extending the ordering used in the corresponding thinning algorithm in a natural and sometimes the only possible way.
- We demonstrate derivations of several approximation algorithms, some of which turn out to be simpler than those in previous works.

Paper structure In Section 2 we give an introduction to the relational approach to optimisation problems and the thinning strategy, using 0-1 knapsack as an example. Section 3 presents our generalisation of the Thinning Theorem, which we use to model FPTASs in Section 4. We apply our theorems to three examples, chosen to demonstrate the variety of problems we can deal with: in Section 5 we solve the approximate version of 0-1 knapsack, a task scheduling problem, before solving the tree partition problem in Section 6. Code and supplementary proofs accompanying this paper are available online [8].

This article is an extended version of the authors' previous work [9]. A new "generalised thinning" theorem, which is more general than the one in our previous work, is presented in Sections 3. We owe both the theorem and its proof to a reviewer. Proof of the theorem is surprisingly clear, and the antecedents of the theorem can be discovered during the proof. Operational explanation of the theorems, which tends to be rather vague, is now replaced by more precise commuting diagrams. Correspondingly, plenty of explanatory texts in Sections 3 and 4 are rewritten. The notations for "lifted orderings" in Section 4 are changed to a set of new notations, hopefully clearer. The discussion on future work in Section 7 is also extended.

[1] The real problem one would wish to solve is when the supply and demand nodes are placed in a graph. However, to solve a graph problem whose difficulty increases with the treewidth, a measure of connectivity, it is common in the algorithm community to start solving a tree version with a small treewidth to gain more insight. See Kleinberg and Tardos [3, Section 10.4] for more discussions.