Artificial Intelligence 238 (2016) 166–189

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Tractability-preserving transformations of global cost functions[☆]

David Allouche[a], Christian Bessiere[b], Patrice Boizumault[e], Simon de Givry[a], Patricia Gutierrez[c], Jimmy H.M. Lee[d,*], Ka Lun Leung[d], Samir Loudni[e], Jean-Philippe Métivier[e], Thomas Schiex[a,*], Yi Wu[d]

[a] MIAT, UR-875, INRA, F-31320 Castanet Tolosan, France
[b] CNRS, University of Montpellier, France
[c] IIIA-CSIC, Universitat Autonoma de Barcelona, 08193 Bellaterra, Spain
[d] Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
[e] GREYC, Université de Caen Basse-Normandie, 6 Boulevard du Maréchal Juin, 14032 Caen cedex 5, France

ARTICLE INFO

Article history:
Received 10 February 2015
Received in revised form 14 June 2016
Accepted 18 June 2016
Available online 21 June 2016

Keywords:
Graphical model
Weighted constraint satisfaction problem
Cost function network
Constraint programming
Constraint satisfaction problem
Global cost functions
Decomposition
Computational complexity

ABSTRACT

Graphical model processing is a central problem in artificial intelligence. The optimization of the combined cost of a network of local cost functions federates a variety of famous problems including CSP, SAT and Max-SAT but also optimization in stochastic variants such as Markov Random Fields and Bayesian networks. Exact solving methods for these problems typically include branch and bound and local inference-based bounds.

In this paper we are interested in understanding when and how dynamic programming based optimization can be used to efficiently enforce soft local consistencies on Global Cost Functions, defined as parameterized families of cost functions of unbounded arity. Enforcing local consistencies in cost function networks is performed by applying so-called Equivalence Preserving Transformations (EPTs) to the cost functions. These EPTs may transform global cost functions and make them intractable to optimize.

We identify as *tractable projection-safe* those global cost functions whose optimization is and remains tractable after applying the EPTs used for enforcing arc consistency. We also provide new classes of cost functions that are tractable projection-safe thanks to dynamic programming.

We show that dynamic programming can either be directly used inside filtering algorithms, defining polynomially DAG-filterable cost functions, or emulated by arc consistency filtering on a Berge-acyclic network of bounded-arity cost functions, defining Berge-acyclic network-decomposable cost functions. We give examples of such cost functions and we provide a systematic way to define decompositions from existing decomposable global constraints.

These two approaches to enforcing consistency in global cost functions are then embedded in a solver for extensive experiments that confirm the feasibility and efficiency of our proposal.

☆ This paper is an extended version of [50] and [4].
* Corresponding authors.
E-mail address: thomas.schiex@toulouse.inra.fr (T. Schiex).

http://dx.doi.org/10.1016/j.artint.2016.06.005
0004-3702/© 2016 Elsevier B.V. All rights reserved.

1. Introduction

Cost Function Networks (CFNs) offer a simple and general framework for modeling and solving over-constrained and optimization problems. They capture a variety of problems that range from CSP, SAT and Max-SAT to maximization of likelihood in stochastic variants such as Markov Random Fields or Bayesian networks. They have been applied to a variety of real problems, in resource allocation, bioinformatics or machine learning among others [2,18,28,29,35,60,63].

Besides being equipped with an efficient branch and bound procedure augmented with powerful local consistency techniques, a practical CFN solver should have a good library of global cost functions to model the often complex scenarios in real-life applications.

Enforcing local consistencies requires to apply Equivalence Preserving Transformations (EPTs) such as cost projection and extension [20]. Most local consistencies require to compute minima of the cost function to determine the amount of cost to project/extend. By applying these operations, local consistencies may reduce domains and, more importantly, tighten a global lower bound on the criteria to optimize. This is crucial for branch and bound efficiency. Global cost functions have unbounded arity, but may have a specific semantics that makes available dedicated polynomial-time algorithms for minimization. However, when local consistencies apply EPTs, they modify the cost function and may break the properties that makes it polynomial-time minimizable. We say that a cost function is *tractable* if it can be minimized in polynomial time. The notion of *tractable projection-safety* captures precisely those functions that remain tractable even after EPTs.

In this paper, we prove that any tractable global cost function remains tractable after EPTs to/from the zero-arity cost function ($W_\varnothing$), and cannot remain tractable if arbitrary EPTs to/from r-ary cost functions for $r \geq 2$ are allowed. When $r = 1$, we show that the answer is indefinite. We describe a simple tractable global cost function and show how it becomes intractable after projections/extensions to/from unary cost functions. We also show that flow-based projection-safe cost functions [46] are positive examples of tractable projection-safe cost functions.

For $r = 1$, we introduce *polynomially DAG-filterable* global cost functions, which can be transformed into a filtering Directed Acyclic Graph with a polynomial number of simpler cost functions for (minimum) cost calculation. Computing minima of such cost functions, using a polynomial time dynamic programming algorithm, is tractable and remains tractable after projections/extensions. Thus, polynomially DAG-filterable cost functions are tractable projection-safe. Adding to the existing repertoire of global cost functions, cost function variants of existing global constraints such as Among, Regular, Grammar, and Max/Min, are proved to be polynomially DAG-filterable.

To avoid the need to implement dedicated dynamic programming algorithms, we also consider the possibility of directly using decompositions of global cost functions into polynomial size networks of cost functions with bounded arities, usually ternary cost functions. We show how such *network-decompositions* can be derived from known global constraint decompositions and how Berge-acyclicity allows soft local consistencies to emulate dynamic programming in this case. We prove that Berge-acyclic network-decompositions can also be used to directly build polynomial filtering DAGs.

To demonstrate the feasibility of these approaches, we implement and embed various global cost functions using filtering DAG and network-decompositions in `toulbar2`, an open source cost function networks solver. We conduct experiments using different benchmarks to evaluate and to compare the performance of the DAG-based and network-based decomposition approaches.

The rest of the paper is organized as follows. Section 2 contains the necessary background to understand our contributions. Section 3 analyses the tractability of enforcing local consistencies on global cost functions and characterizes the conditions for preserving tractability after applying EPTs. In Section 4 we define DAG-filtering and in Section 5 we give an example of a polynomial DAG-filterable global cost function. Sections 6 and 7 present network-decomposability and the conditions for preserving the level of local consistency. Section 8 shows the relation between network-decompositions and DAG-filtering. Section 9 provides an experimental analysis of the two approaches on several classes of problems. Section 10 concludes the paper.

2. Background

We give preliminaries on cost function networks and global cost functions.

2.1. Cost function networks

A cost function network (CFN) is a special case of the valued constraint satisfaction problem [62] with a specific cost structure $([0, \dots, \top], \oplus, \leq)$. We give the formal definitions of the cost structure and CFN as follows.

Definition 1 *(Cost Structure [62]).* The *cost structure* $([0, \dots, \top], \oplus, \leq)$ is a tuple defined as:

- $[0, \dots, \top]$ is the interval of integers from 0 to $\top$ ordered by the standard ordering $\leq$, where $\top$ is either a positive integer or $+\infty$.
- $\oplus$ is the addition operation defined as $a \oplus b = min(\top, a + b)$. We also define the subtraction $\ominus$ operator for any a and b, where $a \geq b$, as: