Artificial Intelligence 221 (2015) 73–114

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Red–black planning: A new systematic approach to partial delete relaxation

Carmel Domshlak [a], Jörg Hoffmann [b,*], Michael Katz [c]

[a] Technion, Haifa, Israel
[b] Saarland University, Saarbrücken, Germany
[c] IBM Research, Haifa, Israel

ARTICLE INFO

Article history:
Received 13 June 2014
Received in revised form 16 December 2014
Accepted 22 December 2014
Available online 5 January 2015

Keywords:
Planning
Heuristic search
Delete relaxation

ABSTRACT

To date, delete relaxation underlies some of the most effective heuristics for deterministic planning. Despite its success, however, delete relaxation has significant pitfalls in many important classes of planning domains, and it has been a challenge from the outset to devise heuristics that take *some* deletes into account. We herein devise an elegant and simple method for doing just that. In the context of finite-domain state variables, we define *red* variables to take the relaxed semantics, in which they accumulate their values rather than switching between them, as opposed to *black* variables that take the regular semantics. Red–black planning then interpolates between relaxed planning and regular planning simply by allowing a subset of variables to be painted red. We investigate the tractability region of red–black planning, extending Chen and Giménez' characterization theorems for regular planning to the more general red–black setting. In particular, we identify significant islands of tractable red–black planning, use them to design practical heuristic functions, and experiment with a range of "painting strategies" for automatically choosing the red variables. Our experiments show that these new heuristic functions can improve significantly on the state of the art in satisficing planning.[1]

1. Introduction

In deterministic, also know as "classical", planning for action selection, the world states are represented by complete assignments to a set of variables, the actions allow for deterministic modifications of these assignments, and the objective is to find a sequence of actions that modifies a given initial assignment to an assignment that satisfies a goal property. In the last two decades, solvers for classical planning have made spectacular advances in their empirical efficiency. A key role in this progress, especially in the context of satisficing planning where no optimality guarantee is required, is played by the *monotonic*, or "delete-free", *relaxation* [4–6].

At a high level, state variables in the monotonic relaxation accumulate their values, rather than switching between them. The key property of such a relaxation is that applying actions under value accumulating semantics does not reduce the

* Corresponding author.
E-mail addresses: dcarmel@ie.technion.ac.il (C. Domshlak), hoffmann@cs.uni-saarland.de (J. Hoffmann), katzm@il.ibm.com (M. Katz).

[1] Many of the results presented herein were previously published in three conference papers [1–3]. The present article discusses these results much more comprehensively, and makes several extensions, including a refinement to the theoretical analysis, detailed proofs, additional experiments, and implementing heuristic functions relying on acyclic black causal graphs instead of arcless ones.

http://dx.doi.org/10.1016/j.artint.2014.12.008
0004-3702/

applicability of actions in the future. As a result, while regular satisficing planning is PSPACE-complete even for simple formalisms, monotonic satisficing planning is polynomial-time [7]. Despite this, plans for the monotonic relaxation – commonly referred to as *relaxed plans* – often yield very useful heuristic functions [8,9], and have been a key ingredient of state-of-the-art satisficing planners (e.g., [5,6,10,11]) since more than a decade.

While some of the most effective heuristics to date are obtained in this manner, the delete relaxation has significant pitfalls. A striking example (see, e.g., [8,12]) is the inability to account for "having to move back" on a road map: Under the relaxation, once we traversed the map once, we are in all locations simultaneously so there never is a need to move back. In effect, if, say, a truck needs to move across a line of road segments to pick up a cargo and then move back to deliver it, then the heuristic value remains constant (equal to the length of the line) until the truck reaches the cargo. In many domains that involve transportation or other types of movement, this leads to huge plateaus, i.e., regions of states with identical heuristic value. Another prominent issue (see, e.g., [13–15]) is "resource persistence", that is, inability to account for the consumption of non-replenishable resources. The monotonic relaxation furthermore ignores any detrimental side effects an action may have on other parts of the plan, trivializing domains with a puzzle nature. For example, monotonic relaxation of Rubic's Cube tasks loses the dependencies across the subcubes.

Given these weaknesses of monotonic relaxation, it has been an actively pursued research challenge from the outset to design heuristics that take *some* deletes into account. This resulted in a wealth of approaches, taking into account conflicts in the relaxed plan [16,10,17,13,18,15], keeping track of (some) side effects [12,19,20], and incorporating TSP solvers responsible for the movements of particular variables in the relaxed plan [21–23]. It has proved daunting, however, to devise frameworks that fully *interpolate* between regular planning and monotonic planning, by providing a choice of *which*, and the ability to scale freely with *how many*, deletes are taken into account. The first such interpolation framework was put forward in 2012, enriching the monotonic relaxation with an explicitly represented set of fact conjunctions, forcing the heuristic to eventually become perfect as that set gets larger [24–26]. We herein propose a much simpler interpolation framework: *we relax only some of the state variables*.

In this framework, which we baptize *red–black planning*, some state variables, called *red*, take the relaxed semantics and accumulate their values, while all other variables, called *black*, keep the regular semantics and thus switch between their values.[2] Consider again our previous example where a truck needs to move across a line of road segments. Say we paint the truck-position variable black, and we paint the cargo-position variable red. A *red–black plan* then needs to include the backward truck moves, and the length of an optimal red–black plan is equal to that of an optimal real plan. The same applies to VisitAll when painting the robot position black. A heuristic function generated this way would be perfect.

The problematic word here of course is the "would". Apart from the quality of the heuristic, we also need to consider the computational effort required to compute it. As red–black planning generalizes monotonic planning – the special case where all variables are painted red – and optimal monotonic planning is NP-hard [7], optimal red–black planning also is (at least) NP-hard. Therefore, in analogy to commonly used relaxed plan heuristics, we will generate an (inadmissible) heuristic function by generating some, not necessarily optimal, red–black plan. Still, the question is *whether there are significant tractable fragments of satisficing red–black planning*.

Fortunately, the answer is "yes". Analyzing the complexity of satisficing red–black planning, we in particular show tractability for planning tasks whose *black causal graph* – the projection of the standard causal graph [27–29] onto the black variables – is acyclic, and whose black variables satisfy a certain invertibility condition. Specifically, we show that any fully relaxed (aka monotonic) plan for a problem in this fragment can be "repaired" into a valid red–black plan with only a polynomial runtime overhead.

Investigating the corresponding heuristic function from a practical perspective, we find that its bias to follow decisions made in the "basis" monotonic plans can be harmful, leading to dramatic over-estimation even in very simple toy benchmarks. Targeting this pitfall, we devise an improved red–black planning algorithm that relies less on monotonic plans, getting rid of much of this over-estimation phenomenon. We fill in important realization details, pertaining in particular to planning with acyclic causal graphs and invertible variables (a sub-procedure of our heuristic function). We devise optimizations enhancing red–black plan applicability, short-cutting the search if the red–black plan is applicable in the original planning task.

To obtain an automatic planning methodology, we finally require *painting strategies* for automatically deciding which variables should adopt each color. We devise a family of such strategies, and analyze their performance. We finally run comparative experiments against the state of the art on the IPC benchmarks, showing that our new heuristic functions bring significant advantages over standard delete-relaxation heuristics, as well as over alternative partial delete-relaxation heuristics, in several domains and overall.

The rest of the paper is structured as follows. In Section 2 we provide the necessary background, and in Section 3 we formally introduce red–black planning as a framework for relaxation. We analyze the complexity of satisficing red–black planning (Section 4), discuss the practical aspects of generating heuristic functions in this context (Section 5), investigate painting strategies (Section 6), and run experiments against the state of the art (Section 7). We conclude with a brief

[2] The aforementioned works of Fox and Long [21,22] and Keyder and Geffner [23] can also be viewed as "un-relaxing" some of the state variables (those controlled by TSP solvers). However, this applies only to restricted subsets of variables, and the TSP treatment is weaker than ours in the sense that it considers only the *set* of variable values that must be visited, ignoring duplicates and ordering. We will get back to this in more detail later, once we formally introduced our framework.