A New Result on the Complexity of Heuristic Estimates for the $A^\star$ Algorithm*

Othar Hansson and Andrew Mayer
Computer Science Division
University of California
Berkeley, CA 94720

Marco Valtorta
Department of Computer Science
University of South Carolina
Columbia, SC 29208

Abstract

Relaxed models are abstract problem descriptions generated by ignoring constraints that are present in base-level problems. They play an important role in planning and search algorithms, as it has been shown that the length of an optimal solution to a relaxed model yields a monotone heuristic for an $A^\star$ search of a base-level problem. Optimal solutions to a relaxed model may be computed algorithmically or by search in a further relaxed model, leading to a search that explores a hierarchy of relaxed models.

In this paper, we review the traditional definition of problem relaxation and show that searching in the abstraction hierarchy created by problem relaxation will not reduce the computational effort required to find optimal solutions to the base-level problem, unless the relaxed problem found in the hierarchy can be transformed by some optimization (e.g., subproblem factoring). Specifically, we prove that any $A^\star$ search of the base-level using a heuristic h_2 will *largely dominate* an $A^\star$ search

*This research was made possible by support from Heuristicrats Research Inc. and the National Aeronautics and Space Administration. The first two authors may be reached at {othar, mayer}@heuristicrat.com. The third author may be reached at mgv@cs.scarolina.edu.

of the base-level using a heuristic h_1, if h_1 must be computed by an $A^\star$ search of the relaxed model *using* h_2.

1 Introduction

We extend a result by Valtorta [17] on the complexity of heuristic estimates for the $A^\star$ algorithm in a way that makes it applicable to some problems that have arisen in recent work in planning and search [6, 12]. We start by giving an informal description of the new result that contrasts it with the original one. Definitions and proofs will be given in the body of the paper.

Both our paper and [17] present complexity results in the state-space model of problem-solving. Both use the number of node expansions performed by the algorithm as a measure of the cost of solving a (state-space) problem using a (heuristic or blind) search algorithm. (By a "blind" search algorithm we mean one that uses no cost estimates or uninformative cost estimates.) Both consider the cost of solving a problem as the total cost of computing the heuristic plus the cost of using $A^\star$ with that heuristic. In both cases, the heuristics are viewed as solutions of relaxed subproblems [13], also known as auxiliary problems [5][1].

Valtorta's 1984 work was motivated by the attempt to spur research on automating the efficient computation of heuristics by showing that, with (certain classes of) inefficient heuristics, $A^\star$ was no better than brute-force search (Dijkstra's algorithm). This paper is motivated by the need to determine how best to use (in a sense to be explained later) a set of available heuristics.

Let P, P', P'' be three problems such that $P'' \Leftarrow P' \Leftarrow P$, where the relation $\Leftarrow$ is the auxiliarity relation. Informally, this notation indicates that P is the base problem, P' is a relaxed subproblem of P, and P'' is a relaxed subproblem of P'. A natural question is that of how to choose a relaxed subproblem (of which there are many) so that the cost of computing the heuristic (by solving the relaxed subproblem) plus the cost of solving the base problem (by $A^\star$ using the heuristic)

[1]The relaxed subproblems of [13] and the auxiliary problems of [5] are special cases of the abstracted problems of [12, 14]. We do not claim that our results apply for the more general definition of [12, 14], except in the special cases where that definition coincides with the older ones.

is minimum. This is the question considered in [16] and [17], whose key result we summarize after the introduction of necessary notation.

Let $A \succeq B$ denote the fact that algorithm A "largely dominates" algorithm B, i.e., in a certain technical sense (to be explained later), A is at least as efficient as B.

Let $A^\star(P \leftarrow Alg(P'))$ denote the $A^\star$ algorithm, applied to the problem P, informed by heuristics derived from the relaxed model P', where instances of P' are solved by an algorithm. If instances of P' are solved by an $A^\star$ search, using P'', we use the notation $A^\star(P \leftarrow A^\star(P' \leftarrow Alg(P'')))$. The pattern of this notation allows us to abbreviate this as $A^\star(P \leftarrow P' \leftarrow P'')$, i.e., as a hierarchy of relaxed models, where the last relaxed model is solved algorithmically. (See Figure 1.)

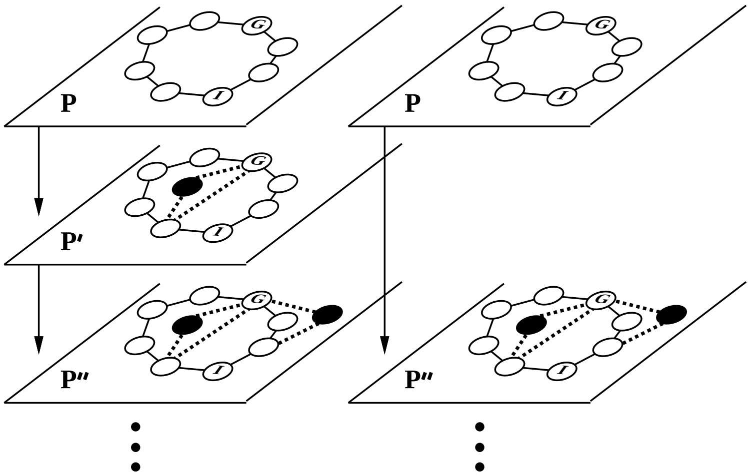

Figure 1: Relaxation hierarchies used in Algorithms $A^\star(P \leftarrow P' \leftarrow P'')$ and $A^\star(P \leftarrow P'')$

Let P^∞ denote an element of the class of most-relaxed models, i.e., a state-space graph in which there is a path of zero cost between every pair of states. $A^\star(P \leftarrow P^\infty)$ is said to be a blind or brute-force algorithm, as it uses a completely uninformative heuristic. Since Dijkstra's algorithm is optimal in a large class of blind search algorithms, as shown in [17], by $A^\star(P \leftarrow P^\infty)$ we mean $A^\star$ with the tie-breaking rule (for node expansions) that makes it equivalent to Dijkstra's algorithm.

The original result [17] is:

$$\forall P' \Leftarrow P \quad A^\star(P \leftarrow P' \leftarrow P^\infty) \preceq A^\star(P \leftarrow P^\infty). \tag{1}$$

The new result (which summarizes both Theorems 1 and 2) is:

$$\forall P'' \Leftarrow P' \Leftarrow P \quad A^\star(P \leftarrow P' \leftarrow P'') \preceq A^\star(P \leftarrow P''), \tag{2}$$

and as a simple corollary,

$$\forall P^n \Leftarrow P^{n-1} \Leftarrow \ldots \Leftarrow P' \Leftarrow P \quad A^\star(P \leftarrow P' \leftarrow \ldots \leftarrow P^{n-1} \leftarrow P^n) \preceq A^\star(P \leftarrow P^n). \tag{3}$$

In words, while Valtorta showed that there is no efficient way of solving a problem with $A^\star$ when the cost of computing the heuristic using a blind method is counted, we now show that the most efficient way of using a hierarchy of relaxed models, culminating in a heuristic that can be computed algorithmically, is to ignore the hierarchy and use that heuristic directly. Generating a relaxed model is useful only when an efficient algorithm accompanies it.

The rest of the paper is organized as follows: In the second section we provide background material on state-space search, the $A^\star$ algorithm, and relaxed models. The aim of the third section is to motivate the need for our new result, by describing an instance from the recent literature to which it can be usefully applied. In the fourth section, we prove our main result (corresponding to (2) above). The fifth section, which concludes the paper, is devoted to an interpretation of the main result, including some remarks on the measure of cost ("largely dominates") used to evaluate the comparative efficiency of state-space search algorithms.

2 Background

2.1 State-Space Search

The state-space approach to problem-solving considers a problem as a quadruple, $(S, O \subset S \times S, I \in S, G \subset S)$. S is the set of possible *states* of the problem. O is the set of *operators*, or transitions from state to state. I is the one *initial state* of a problem instance, and G is the set of *goal states*. Search problems can be represented as a state-space graph, where the states are nodes, and the operators are directed, weighted arcs between nodes (the weight associated with each operator, O_i, is the cost of applying it, $C(O_i)$). The problem consists of determining a sequence of operators, $O_1, O_2, ..., O_n$ that, when applied to I, yields a state in G. Such a sequence is called a *solution path* (or *solution*), with *length* n and *cost* $\sum_{i=1}^{n} C(O_i)$. A solution with minimum cost is called *optimal*.

2.2 $A^\star$ Algorithm

Solutions to a given problem may be found by brute force search over the state-space. However, as the sizes of the state-spaces of most problems are prohibitively large, the only hope of finding an optimal solution in reasonable time is to use an intelligent method of guiding a search through the state-space. Typically, such methods take the form of branch-and-bound, wherein partial solutions (equivalently, classes of solutions) are enumerated ("branch"), and perhaps eliminated from future consideration by an estimate of solution cost ("bound"). One such method, the celebrated $A^\star$ algorithm (originally in [7]; also [13]) orders the search by associating with each node n two values: $g(n)$, the length of the shortest-path from the initial state to n, and $h(n)$, an estimate of the length of the shortest-path from n to any goal state (the actual length is $h^*(n)$). In brief, $A^\star$ is an ordered best-first search algorithm, that always examines the successors of the "most promising" node in the search tree, based on the evaluation function, $f(n) = g(n) + h(n)$. Thus the number of nodes expanded is a function of the heuristic function used.

The following definitions will simplify the discussion:

A heuristic function, $h(n)$, is said to be *admissible* if $\forall n[h(n) \leq h^*(n)]$.

A heuristic function, $h(n)$, is said to be *monotone* if $\forall n, n'[f(n) \leq f(n')$ (where n' is a successor of n)] (recall that $f(n)$ is determined by $h(n)$). Monotonicity implies admissibility [13].

$A^\star$ terminates when it expands (examines) a goal state for the first time. If it uses an admissible heuristic, $A^\star$ will have found the optimal path to this goal state, i.e., $f(n) = g(n) + h(n) = h^*(I)$. As it orders the nodes by $f(n) = g(n) + h(n)$, $A^\star$ with a monotone heuristic *surely expands* all nodes n that satisfy $g(n) + h(n) < h^*(I)$, and *possibly expands* some of the nodes that satisfy $g(n) + h(n) = h^*(I)$, but expands none of the nodes that satisfy $g(n) + h(n) > h^*(I)$. The surely expanded nodes form a connected component in the state-space graph (containing the initial state), and those nodes, together with the possibly expanded nodes, form a larger connected component (containing both the initial and goal states).

2.3 Relaxed Models

As the accuracy of the heuristic function determines the time complexity (measured in node expansions) of $A^\star$, many researchers have attempted to automate the learning of heuristic functions. One proposed method of developing good heuristics is to "consult simplified models of the problem domain" [4, 5, 11, 13]. These simplified models are generated via *constraint-deletion*, i.e., ignoring selected constraints on the applicability of operators. Recently, there has been renewed interest in this and other uses of abstraction in problem-solving [9, 10, 12, 15], including a generalization of the notion of simplification [14] that will be described below.

Pearl [13] discusses the use of constraint-deletion to generate three known heuristics for the familiar Eight Puzzle problem. He formalizes the problem in terms of domain predicates, and then describes the single operator in the state-space – moving a tile x from position y to position z – as follows:

$Move(x, y, z)$

$$\begin{aligned} \text{preconditions}: &\quad On(x,y) \wedge Clear(z) \wedge Adj(y,z) \\ \text{add list}: &\quad On(x,z) \wedge Clear(y) \\ \text{delete list}: &\quad On(x,y) \wedge Clear(z) \end{aligned}$$

Removing preconditions or constraints for this operator creates a relaxed model of the problem. Deleting different sets of constraints yields different relaxed models.

Specifically, when constraints are removed from a problem, new edges and nodes are introduced into the state-space graph, yielding a "relaxed" state-space graph (a supergraph [4]). If the states and operators of the original problem, P, are denoted by the set S and the relation $O \subset S \times S$, then the relaxed problem, P', consists of $S' \supseteq S$ and $O' \supseteq O$ as well as I, and G. Note that relaxed models may permit certain "speedup transformations," such as factoring into independent subproblems [12]: our main result underscores the importance of discovering such optimizations, as discussed in Section 5.

An optimal (i.e., shortest) path between any two given states in P' cannot be longer than the shortest-path between the same two states in P, since all paths in P are also paths in P'. Thus the length of such a relaxed solution is a lower bound on the length of an optimal solution to the original problem, and this information can be used as an admissible heuristic to speed up a branch-and-bound search algorithm (e.g., $A^\star$). Note that, while our results apply to "traditional" [4, 5, 11, 13, 16, 17] relaxed problems obtained by constraint-deletion, other transformations, more general than constraint-deletion, may be used to produce admissible heuristics. Prieditis [14] gives conditions for transformations that guarantee the generation of problems whose solution can be used to compute an admissible heuristic. He calls these transformations *abstracting* and the resulting problems *abstracted problems.* Our results do not necessarily apply to abstracted problems.

Of course, the length of a solution to a further relaxed model P'' provides a lower bound on the length of a solution to P'. Thus, if no efficient algorithm[2] is known

[2]To be precise, we describe as *efficient* algorithms any techniques that perform no search in a

for solving P', we may use P'' to provide an admissible heuristic for an $A^\star$ search of P'. We may conceive of a search algorithm (Figure 2) that performs search at many levels of a relaxed model hierarchy, each level providing heuristic estimates for the level above.

```
findpath(State I, State G, Problem P) {
    if Efficient_algorithm_for(P) then
        return(Efficient_algorithm_for(P, I, G))
    else
        return(Astar(I, G, P))
}

Astar(State I, State G, Problem P) {
/* a standard A* algorithm, using, for each state s */
/* h(s) = heur(s,G,P)                                 */
}

heur(State s, State G, Problem P) {
    Problem P' = relax(P);
    return(length(findpath(s,G,P')));
}
```

Figure 2: Search in Relaxed Model Hierarchy

relaxed model.

3 A Motivating Example

The following example of the application of our result concerns a heuristic for the Eight Puzzle. Please refer to the previous section for a formalization of the puzzle in terms of domain predicates and operators. The example is adapted from [6].

In the Eight Puzzle, the tile positions form a bipartite graph of positions – each move shifts the blank from one side of the bipartite graph to the other. If one colors the puzzle like a checkerboard and connects adjacent squares by edges, the red squares will form one side of the bipartite graph and the black squares the other. In the Eight Puzzle problem, the blank is constrained to move to only a small subset of the other side of the graph (i.e., the adjacent positions). One may relax this constraint by allowing the blank to move to any of the positions in the other side of the bipartite graph.

This Checkerboard Relaxed Adjacency (Check-RA) relaxed model may be thought of as being somewhere between the original problem and the Relaxed Adjacency (RA) model [4], because one has deleted only a part of the adjacency requirement (the RA model results from deleting the adjacency requirement in moving tiles into the blank position). In the Check-RA model, one can think of any given tile position as being "adjacent" to half of the other positions. To formalize this, we define two new STRIPS-like predicates (where $\oplus$ denotes exclusive-or):

$$\begin{array}{lcl} RED(y) & : & y \text{ is a red position} \\ DIFFCOLOR(x,y) & : & RED(x) \oplus RED(y) \end{array}$$

and change the preconditions for MOVE:

$$\begin{array}{rl} \multicolumn{2}{l}{Move(x,y,z)} \\ \text{preconditions :} & On(x,y) \wedge Clear(z) \wedge DIFFCOLOR(y,z) \\ \text{add list :} & On(x,z) \wedge Clear(y) \\ \text{delete list :} & On(x,y) \wedge Clear(z) \end{array}$$

In short, if a black position is blank (i.e., clear), only tiles in red positions may move into it, and *vice versa*. There is no known algorithm to solve this relaxed model, except for one that searches the corresponding state space. However, other relaxed problems for the Eight Puzzle are more relaxed than Check-RA, e.g., Relaxed Adjacency (RA), for which an efficient solution algorithm is known [4, 6]. We can therefore solve the Check-RA problem by using $A^\star$ with a heuristic computed by solving the RA problem. A natural question to ask is whether it is more efficient to use the heuristic from RA indirectly (to solve Check-RA as just described) rather than to use it directly to solve the Eight Puzzle. Clearly, the Check-RA heuristic is never smaller than the RA heuristic and can therefore prevent $A^\star$ from expanding some nodes that it would expand using the RA heuristic. On the other hand, a secondary search procedure (using $A^\star$) must be carried out to compute the Check-RA heuristic. Is it possible to predict which one of the two uses of the RA heuristic is the most efficient? We will answer this question in the affirmative.

4 The Limitations of Abstraction

We prove a result that allows us to collapse arbitrary relaxed model hierarchies to two levels.

Consider that some constraints of a base-level problem P have been deleted to yield a relaxed model P', whose solutions provide a heuristic estimate h_1 for the base-level problem. Unfortunately, there may be no efficient algorithmic solution to this relaxed model P', and to compute h_1 one may have to perform a search in P' using h_2. We will denote this algorithm by $A^\star(P \leftarrow P' \leftarrow P'')$. Consider also the algorithm $A^\star(P \leftarrow P'')$, that uses h_2 directly as a heuristic for searching P. The two algorithms are depicted in Figure 1.

Note that $A^\star(P \leftarrow P' \leftarrow P'')$ performs two distinct heuristic searches, one in P, and one in P'. $A^\star(P \leftarrow P'')$ performs only the search in P. We will prove that $A^\star(P \leftarrow P'')$ *largely dominates* $A^\star(P \leftarrow P' \leftarrow P'')$, i.e., all nodes surely expanded by $A^\star(P \leftarrow P'')$ are also surely expanded by $A^\star(P \leftarrow P' \leftarrow P'')$, and all nodes possibly expanded by $A^\star(P \leftarrow P'')$ are also possibly expanded by $A^\star(P \leftarrow P' \leftarrow P'')$.

Theorem 1 *If a node n is surely expanded by algorithm $A^\star(P \leftarrow P'')$, then node n is surely expanded by algorithm $A^\star(P \leftarrow P' \leftarrow P'')$.*

Assume not. Consider a node n surely expanded by $A^\star(P \leftarrow P'')$, but not surely expanded by algorithm $A^\star(P \leftarrow P' \leftarrow P'')$. Consider an ancestor p of n that is the first node on a cheapest path from I to n that is not surely expanded by $A^\star(P \leftarrow P' \leftarrow P'')$:

$$g(n) = g(p) + g(p, n) \tag{4}$$

$$g(p) + h_1(p) \geq h^*(i) \tag{5}$$

$g(p, n)$ denotes the cost of the cheapest path from p to n. By assumption, computation of $h_1(p)$ does not surely expand n:

$$g^{P'}(p, n) + h_2(n) \geq h_1(p) \tag{6}$$

$g^{P'}(p, n)$ denotes the cost of the cheapest path from p to n in the relaxed model P'. Clearly,

$$g(p, n) \geq g^{P'}(p, n) \tag{7}$$

Substituting inequality 6 into inequality 5:

$$g(p) + g^{P'}(p, n) + h_2(n) \geq h^*(i) \tag{8}$$

and noting from equations 4 and 7 that:

$$g(n) \geq g(p) + g^{P'}(p,n) \tag{9}$$

we derive:

$$g(n) + h_2(n) \geq h^*(i) \tag{10}$$

But by assumption, n is surely expanded by $A^\star(P \leftarrow P'')$:

$$g(n) + h_2(n) < h^*(i) \tag{11}$$

which contradicts inequality 10. $\square$

Theorem 2 *If a node n is possibly expanded by algorithm $A^\star(P \leftarrow P'')$, then node n is possibly expanded by algorithm $A^\star(P \leftarrow P' \leftarrow P'')$.*

Assume not. Consider a node n possibly expanded by $A^\star(P \leftarrow P'')$, but not possibly expanded by algorithm $A^\star(P \leftarrow P' \leftarrow P'')$. Consider an ancestor p of n that is the first node on a cheapest path from I to n that is not possibly expanded by $A^\star(P \leftarrow P' \leftarrow P'')$:

$$g(n) = g(p) + g(p,n) \tag{12}$$

$$g(p,n) \geq g^{P'}(p,n) \tag{13}$$

$$g(p) + h_1(p) > h^*(i) \tag{14}$$

By assumption, computation of $h_1(p)$ does not possibly expand n:

$$g^{P'}(p,n) + h_2(n) > h_1(p) \tag{15}$$

Substituting inequality 15 into inequality 14:

$$g(p) + g^{P'}(p,n) + h_2(n) > h^*(i) \tag{16}$$

and noting from equations 12 and 13 that:

$$g(n) \geq g(p) + g^{P'}(p,n) \tag{17}$$

we derive:

$$g(n) + h_2(n) > h^*(i) \tag{18}$$

But by assumption, n is possibly expanded by $A^\star(P \leftarrow P'')$:

$$g(n) + h_2(n) \leq h^*(i) \tag{19}$$

which contradicts inequality 18. □

5 Interpretation of Results

As anticipated at the beginning of the previous section, Theorems 1 and 2 imply that $A^\star(P \leftarrow P'')$ largely dominates $A^\star(P \leftarrow P' \leftarrow P'')$. This is precisely the sense in which $A^\star$ is said to be *optimal* [13, p. 85]: given a monotone heuristic h,

$A^\star$ *largely dominates* every admissible algorithm that uses h. The relation misses some important distinctions. For example, greedy algorithms, that may operate in polynomial time, can be viewed as heuristic searches with optimal tie-breaking rules to choose among the possibly expanded nodes. Large domination does not imply that all nodes expanded by $A^\star(P \leftarrow P'')$ will necessarily be expanded by $A^\star(P \leftarrow P' \leftarrow P'')$. In fact there are some tie-breaking rules (for ordering expansion of nodes with equal f values) for which $A^\star(P \leftarrow P' \leftarrow P'')$ will expand fewer nodes than $A^\star(P \leftarrow P'')$. Traditionally, however, large domination is equated with superior efficiency, as the number of nodes n for which $g(n)+h(n) = h^*(i)$ is assumed to be small: for example, if either $g(n)$ or $h(n)$ are continuous valued functions [13, p. 85] or h is non-pathological (as defined in [2, p. 522]), such ties are rare or non-existent.

The theorems extend results of Valtorta [17], who proved that brute-force, uniform-cost search (Dijkstra's algorithm) is never less efficient than a heuristic search that relies on uniform-cost search of relaxed models for the computation of heuristics. This follows from our results by considering the special case in which h_2 is uniformly zero, yielding uniform-cost search. However, Valtorta's original result is for a sharper notion of "more efficient than": the number of node expansions, rather than large domination. The reason for this discrepancy lies in the fact that Dijkstra's algorithm expands all and only the nodes for which $g^*(n) < h^*(i)$, while $A^\star$ with monotone heuristics may expand some nodes for which $g^*(n)+h(n) = h^*(i)$: for Dijkstra's algorithm, the set of possibly expanded nodes coincides with the set of surely expanded nodes. Analogously, while Dijkstra's algorithm is optimal with respect to the measure "number of node expansions" within the class of blind forward unidirectional algorithms [17], $A^\star$ is optimal in that it largely dominates all informed algorithms that use consistent heuristics. "Large domination" is a weaker form of optimality than "fewer node expansions," as is explained by Dechter and Pearl in [1], and especially in [3].

We observe that $A^\star$ with monotone heuristics, just like Dijkstra's algorithm, never expands the same node twice. Since all heuristics computed by solving a relaxed problem are monotone [13, 17], we can use "the number of expanded nodes" and "the number of node expansions" interchangeably. Therefore our results hold for the cost measure "number of node expansions," as well as for the cost measure "number of expanded nodes."

While Valtorta [17] showed that there is no advantage to using $A^\star$ in certain situations, we show that, in certain situations, there is no advantage to using a relaxed-model hierarchy within an $A^\star$ algorithm. Theorems 1 and 2 indicate that in reducing the cost of finding optimal solutions, a heuristic is effective only when it is computable by an efficient algorithm – the cost of computing a heuristic function by a secondary search exceeds the savings offered by the heuristic. This possibility was recognized in the design of ABSOLVER [12], where "speedup transformations" are used, as well as in other recent work [8]. Our result shows (in a precise and general way) that speedup transformations are not only useful, but necessary.

Some readers may object to our use of the word "search" in the preceding paragraph, since, after all, a search in a much reduced state space is sufficient to turn an inefficient procedure into a practical, efficient one. In the spirit of [5, 17] we use the word search to signify that a relaxed problem is solved by search in *its* state space (called *underlying graph* in [5] and *skeleton* in [17]) and view the construction of a reduced state space as an instance of a "speedup transformation." A way of reducing the state space for a (relaxed) problem is to factor (or decompose) it into independent subproblems. Decomposability is possible when the goal state of a state-space search problem can be achieved by solving its subgoals independently. As discussed in [12, 13], the state-space for each independent subproblem can be so much smaller than the original state-space that computation of the heuristic by searching the factored problem is computationally advantageous over blind search. We illustrate these considerations with an example adapted from [12].

Mostow and Prieditis describe a heuristic (called X-Y) that is never smaller than the well-known Manhattan Distance (MD) heuristic. It corresponds to a relaxed problem in which "a horizontal move is allowed only into a column containing the blank, and a vertical move is allowed only into the row containing the blank. X-Y is therefore more accurate than Manhattan Distance, which ignores the blank completely." This relaxed problem can be obtained from a representation of the tiles in the Cartesian plane[3], by dropping all information about the position of the blank in the Y coordinate (YLOCB) from moves in the X coordinate (XMove) and all information about the position of the blank in the X coordinate (XLOCB) from moves in the Y coordinate (YMove). The X-Y relaxed problem can be decomposed into two subproblems (or *factors*), corresponding to each coordinate. The goal of the first subproblem is to place each tile in its correct column, while the goal of the second subproblem is to place each tile in its correct row. Since we dropped from the list of preconditions of the move operator in a coordinate all information about the position of the blank in the other coordinate, we obtain two independent subproblems, as described in Figure 3.

No algorithm is known to solve either of the independent subproblems without searching the corresponding state-space. However, since MD is a relaxed problem of X-Y for which a very efficient solution algorithm is known, it is possible to solve X-Y by using $A^\star$ with a heuristic computed by solving MD. Now consider the cost of computing X-Y as just outlined. This computation involves search. However, the search is in the reduced state-spaces of two independent subproblems of the X-Y relaxed problem, and therefore the results described in this paper are not applicable. In other words, the theorems we have proved in this paper are not strong enough to predict when a heuristic derived from relaxation and factoring is

[3]We use the traditional goal state with the blank in the middle and tiles arranged clockwise around the border, starting with tile 1 in the top left corner. Mandrioli et al. [11] used the Cartesian representation for the Eight Puzzle in the first published examples of relaxed problems, but they did not provide the X-Y heuristic as an example.

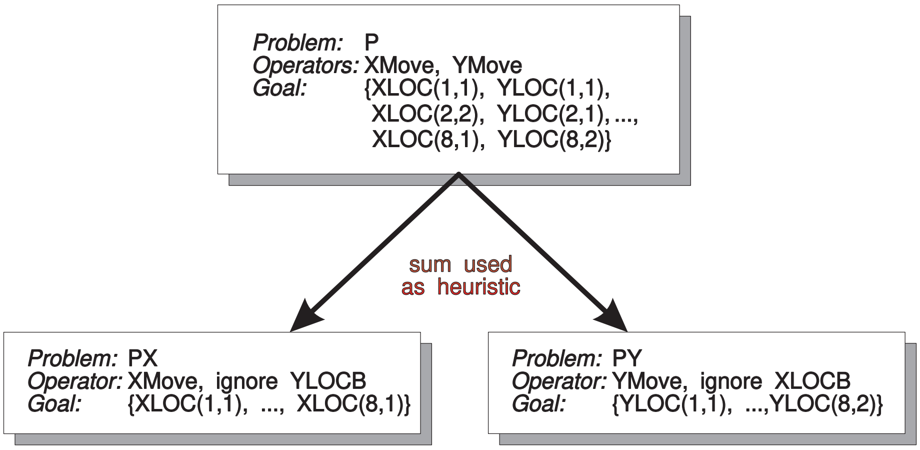

Figure 3: Two independent subproblems to compute the X-Y heuristic

cost effective, and it is *not* possible to conclude whether the computation of the X-Y heuristic is cost effective simply on the basis of our results. In fact, it has been determined empirically that computation of the X-Y heuristic as described is not cost effective. Mostow and Prieditis report that "unfortunately, even with such guidance [i.e., the heuristic computed by solving MD] the overall search time turns out to be about six times slower than using Manhattan distance alone." In other words, direct use of MD is more efficient than indirect use of MD to compute X-Y. This shows that heuristics computed by searching factors of relaxed problems need not be cost effective. On the other hand, the Box Distance heuristic of the Rooms World problem, when computed by search of factors of a relaxed problem, is cost effective [14]. This shows that heuristics computed by searching factors of relaxed problems can be cost effective. Additional work is needed to find conditions under which heuristics computed by searching factors of relaxed problems are cost effective, and conditions under which they are not.

In conclusion, we have shown that relaxed model hierarchies (chains of successively more abstract problems) collapse to only two levels – the base-level, and the least-relaxed model for which algorithmic solutions are known. Exploiting the power of abstraction will require either the development of techniques for synthesizing efficient algorithms from relaxed-model problem-descriptions, or the development of new problem relaxation transformations that are not subject to the fundamental limitations of the theorems presented here [8, 12, 14]. The most promising example of these approaches is the ABSOLVER system [12, 14], which exploits subproblem factoring transformations to reduce the cost of search in relaxation hierarchies.

6 Acknowledgements

We would like to thank two anonymous referees for their insightful and detailed comments on earlier drafts of this paper and Armand Prieditis for several interesting conversations. Marco Valtorta thanks Marco Somalvico and Giovanni Guida for introducing him to research on auxiliary problems and Jack Mostow for a conversation in which he shared some (at the time) unpublished details of the ABSOLVER system.

References

[1] R. Dechter and J. Pearl. The Optimality of $A^\star$ Revisited. In *Proceedings of the Third National Conference on Artificial Intelligence*, Washington, 1983, 95–99.

[2] R. Dechter and J. Pearl. Generalized Best-First Search Strategies and the Optimality of $A^\star$. *Journal of the Association for Computing Machinery*, 32, 3 (July 1985), 506–536.

[3] R. Dechter and J. Pearl. The Optimality of $A^\star$. In *Search in Artificial Intelligence*, eds. L. Kanal and V. Kumar, Springer-Verlag, New York, 1988, 166–199.

[4] J. Gaschnig. A Problem Similarity Approach to Devising Heuristics: First Results. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, Tokyo, 1979, 301–307.

[5] G. Guida and M. Somalvico. A Method for Computing Heuristics in Problem Solving. *Information Sciences*, 19:251–259, 1979.

[6] O. Hansson, A. Mayer, and M. Yung. Relaxed Models Yield Powerful Admissible Heuristics. *Information Sciences*, to appear.

[7] P. Hart, N. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 2:100–107, 1968.

[8] D. Kibler. Generation of Heuristics by Transforming the Problem Representation. Technical Report TR-85-20, Information and Computer Science Department, University of California at Irvine, July 1985.

[9] C. A. Knoblock. Automatically Generating Abstractions for Planning. In *Proceedings of the International Workshop on Change of Representation and Inductive Bias*, Philips Laboratories, Briarcliff Manor, New York, 1988.

[10] R. E. Korf. Planning as Search: a Quantitative Approach. *Artificial Intelligence*, 33:65–88, 1987.

[11] D. Mandrioli, A. Sangiovanni-Vincentelli, and M. Somalvico. Toward a Theory of Problem Solving. In *Topics in Artificial Intelligence*, ed. A. Marzollo, Springer-Verlag, Vienna, 1976, 48–167.

[12] J. Mostow and A. E. Prieditis. Discovering Admissible Heuristics by Abstracting and Optimizing: a Transformational Approach. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, 1989, 701–707.

[13] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley, Reading, MA, 1984.

[14] A. Prieditis. *Machine Discovery of Effective Admissible Heuristics.* In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, Sidney, 1991, 720–725.

[15] J. Tenenberg. Abstraction in Planning. Ph.D. Thesis, University of Rochester, 1988.

[16] M. Valtorta. Un Contributo alla Teoria della Risoluzione dei Problemi: Rappresentazione Semantica, Proprietà Algebriche ed Algoritmi di Ricerca (in Italian). Tesi di Laurea, Istituto di Ingegneria Elettrotecnica ed Elettronica, Politecnico di Milano, Milan, 1980.

[17] M. Valtorta. A Result on the Computational Complexity of Heuristic Estimates for the $A^\star$ Algorithm. *Information Sciences*, 34:48–59, 1984.