TABLE 3

The Optimal Solution of $T_1(\alpha)$ with $\alpha = 0.5$, 0.6, and 0.7

| $\alpha$ | 0.5 | 0.6 | 0.7 |
|---|---|---|---|
| $J(\bar{a}, c)$ | 18.26 | 25.81 | 38.73 |
| $\bar{a}_0$ | 3.201 | 3.367 | 3.508 |
| $c_0$ | 0.1703 | 0.4260 | 0.9242 |
| $\bar{a}_1$ | 0.5788 | 0.5595 | 0.5429 |
| $c_1$ | 0.0806 | 0.1108 | 0.1530 |

regression analysis, including the relationships to the usual regression analysis as well as the application techniques and how to decide the degree $\alpha$, we hope that such fuzzy regression analysis will become an efficient tool for analyzing real world systems where the ambiguity or fuzziness of human subjective judgment is influential.

## REFERENCES


1. D. Dubois, Linear programming with fuzzy data, in *Analysis of Fuzzy Information*, Vol. 3 (J.C. Bezdek, Ed.), CRC Press, 1987, pp. 241–263.
2. D. Dubois and H. Prade, *Fuzzy Sets and Systems: Theory and Application*, Academic, 1980.
3. D. Dubois and H. Prade, Ranking fuzzy numbers in the setting of possibility theory, *Inform. Sci.* 30:183–224 (1983).
4. H. T. Nguyen, A note on the extension principle for fuzzy sets, *J. Math. Anal. Appl.* 64:369–380 (1978).
5. M. Sakawa and H. Yano, Fuzzy linear regression and its application to the sales forecasting, in *Proceedings of International Computer Symposium 1988*, Tamkang Univ., Taiwan, 1988.
6. H. Tanaka, J. Watada, and I. Hayashi, On three formulations of fuzzy linear regression analysis (in Japanese), *SICE* 22:1051–1057 (1986).
7. H. Tanaka, Fuzzy data analysis by possibilistic linear models, *Fuzzy Sets and Systems* 24:363–375 (1987).
8. H. Tanaka, I. Hayashi, and J. Watada, Possibilistic linear regression analysis based on possibility measure, in *Proceedings of Second IFSA Congress*, Tokyo, 1987, pp. 317–320.
9. L.A. Zadeh, The concept of linguistic variable and its application to approximate reasoning, *Informat. Sci.* 8:199–249 (1975).


*Received 14 July 1989; revised 28 November 1989, 28 March 1990*

Marco Valtorta Dezembro 1992

*INFORMATION SCIENCES* **63**, 207–227 (1992) (September 15, 1992)

# Criticizing Solutions to Relaxed Models Yields Powerful Admissible Heuristics*

OTHAR HANSSON

and

ANDREW MAYER

*Computer Science Division, University of California, Berkeley, California 94720*

and

MOTI YUNG

*IBM Research Division, T. J. Watson Research Center, Yorktown Heights, New York 10598*

ABSTRACT

Branch-and-bound techniques allow intractable problems to be solved by using heuristics to bound the cost of partial solutions. The use of admissible heuristics can guarantee that the solutions found are optimal. This paper examines one paradigm—problem relaxation by constraint deletion—which has been used to develop many admissible heuristics. The paradigm suggests three steps: simplify (or relax) a problem, solve the simplified problem, and use that solution to guide the search for a solution to the original problem. We introduce the following extension to this methodology: by criticizing the feasibility of a relaxed solution, we arrive at a closer approximation of the solution to the original problem. We apply this methodology to two well-studied problems in operations research and artificial intelligence. For the traveling-salesman problem, iteration of our technique yields a series of novel heuristics, culminating in Held and Karp's minimum-spanning-tree heuristic. For the eight puzzle, it yields a heretofore undiscovered heuristic which is shown to perform significantly better than any previously known.

## 1. INTRODUCTION

Domain-specific heuristics enable us to solve many otherwise intractable problems by intelligently directing the search for solutions. Many have at-

*This paper describes research performed while the authors were at Columbia University, with the support of NSF Grant MCS-8303139 and an IBM Fellowship.


655 Avenue of the Americas, New York, NY 10010 0020-0255/92/$5.00

tempted to understand how human experts discover and employ heuristics [12–14], so as to formulate a general methodology for developing them.

In this paper, we concentrate on one paradigm [3, 4, 10, 13] for the systematic generation of admissible heuristics in the state-space model of problem solving [12] (we follow the notation and examples of Pearl [13]). An admissible heuristic is one which provides an underestimate of the cost to complete a partial solution. Such admissible heuristics can be used by branch-and-bound algorithms (such as $A^*$) to find optimal solutions.

This paradigm, problem relaxation by constraint deletion, offers a method for deriving heuristics by examining optimal solutions to simplified problem models. More specifically, the paradigm outlines a three-step process—first, simplify the problem; second, solve the simplified problem (preferably algorithmically [11, 16]); and third, use that solution to guide the search for an optimal solution to the original problem. We introduce an extension to this paradigm which enables us to recover some of the information lost in the relaxation—by more closely investigating those aspects of the original problem which are isolated by the simplification, and those which are overlooked, one can criticize the feasibility of the simplified solution and arrive at a better approximation to the actual solution.

We apply this solution-criticism method towards developing admissible heuristics for two problems—the traveling-salesman problem [2, 6, 7, 9] and the eight puzzle [1, 3, 13]. These problems have been used to study and develop heuristic problem-solving techniques for more than thirty years. We introduce the solution-criticism methodology on the traveling-salesman problem. Using the constraint-deletion paradigm to develop admissible heuristics, and applying solution criticism to refine them, we can mechanically derive Held and Karp's well-known minimum-spanning-tree heuristic [6, 7]. We then apply the methodology to the eight puzzle to develop a new admissible heuristic for the problem, which is more powerful then any previously known.

### 1.1. BACKGROUND

*Overview of Heuristic Search.* The state-space approach to problem solving considers a problem as a quadruple, $(S, O \in S \times S, I \in S, G \subset S)$. $S$ 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 in determining a sequence of operators, $O_1, O_2, \ldots, O_n$ which, 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*.



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 $A^*$ algorithm [13], orders the search by associating with each state $s$ two values—$g(s)$, the length of the shortest path from the initial state to $s$, and $h'(s)$, an estimate of the length of the shortest path from $s$ to any goal state [the actual length is $h(s)$]. In brief, $A^*$ is an ordered best-first search algorithm, which always examines the successors of the "most promising" state, based on the evaluation function, $f'(s) = g(s) + h'(s)$.

The following definitions will simplify the discussion:

(1) A heuristic function $h'(s)$ is said to be *admissible* if $\forall s[h'(s) \leqslant h(s)]$.

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

(3) A heuristic function $h'_1(s)$ is said to be *more informed* than another heuristic function $h'_2(s)$ if $\forall s[h'_2(s) \leqslant h'_1(s)]$ and $\exists s[h'_2(s) < h'_1(s)]$ and both are admissible.

Because the real-world cost of applying operators may be prohibitively expensive, it may be wise to search for optimal solutions, despite the possible extra time required to do so. If $A^*$ uses an admissible heuristic, it is guaranteed to find optimal solutions [13]. We will consider only admissible heuristics in this paper, and consequently, the word "solution" will henceforth imply "optimal solution."

The informedness of two heuristics determines their relative performance in search. If one has two heuristic functions, $h'_1(s)$ and $h'_2(s)$ (both of which are monotone), such that $h'_1(s)$ is more informed than $h'_2(s)$, then one is guaranteed that $A^*$ will examine an equal or smaller number of states if it uses $h'_1(s)$ instead of $h'_2(s)$ [13]. Therefore, if it is known that $h'_1(s)$ is never less than $h'_2(s)$, then the search time (measured by the number of states expanded) using $h'_1(s)$ is guaranteed not to exceed the search time using $h'_2(s)$. However, the actual computation time is equal to the product of the number of states examined and the computational effort needed to calculate the heuristic esti-

mate. Therefore, in attempting to improve heuristics, one must consider the complexity of the heuristic function as well as its informedness.

*The Traveling-Salesman Problem and the Eight Puzzle.* The development of lower bounds (i.e., admissible heuristics) for the NP-hard traveling-salesman problem [2] has been the subject of a great deal of research [6, 13]. The problem is that of finding a shortest Hamiltonian circuit through $n$ vertices. While recent work on this problem has concentrated on approximation methods for finding near-optimal solutions quickly, early work was concerned with branch-and-bound techniques for finding optimal solutions by using admissible heuristics.

The eight puzzle (Figure 1) is a classic example of small, well-defined, and conceptually simple problem which is sufficiently complex to exhibit interesting phenomena; therefore, it serves as a popular testing ground for heuristic search and problem-solving methods. In particular, it is used to demonstrate the development of heuristics in [3, 4, 13].

The eight puzzle consists of a $3 \times 3$ frame containing eight numbered, sliding tiles. One of the positions in the frame does not contain a tile; this space is called the *blank*. There is one legal operator in this state space—sliding any one of the tiles which are horizontally or vertically adjacent to the blank into the blank's position. A solution to a problem instance is a sequence of operators which transforms a given initial state into a particular goal state (Figure 1 shows the goal state used in this paper).

*Developing Heuristics through Problem Relaxation.* We focus our attention on the work of several researchers who propose that a natural method of developing good heuristics is to "consult simplified models of the problem domain" [3, 4, 13]. One method for developing such simplified models is constraint deletion, which involves ignoring selected constraints on the applicability of operators.

When constraints are removed from a problem, new edges and nodes are introduced into the state-space graph. The shortest path between any two given

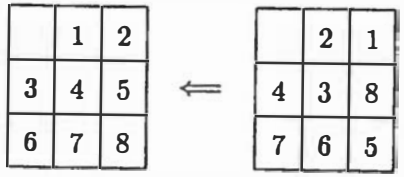



Fig. 1. Eight-puzzle goal state and sample initial state.



states in the relaxed graph cannot be longer than the shortest path between the same two states in the original graph (one can always choose to use the original path). Because it is a lower bound on the cost of an optimal solution to the original problem, one can use the solution length of the relaxed problem as an admissible heuristic for the original problem. Furthermore, because $h'(s)$ is derived from an actual path length in the relaxed state-space graph [it is equal to $h(s)$ in this graph], it is easily seen that the resulting evaluation function is monotone [16].

Our observation is that instead of using the relaxed solution directly to advise us in finding the solution to the original problem, we can first investigate characteristics of the relaxed solution (which can be thought of as a preliminary plan for solution) in comparison with those of the actual solution. If we can admissibly recover any information that was lost in the process of relaxation, but is easily identified when comparing the relaxed model and the original problem, we will have created a more informed admissible estimate than the heuristic from the relaxed model alone.

## 2. SOLUTION CRITICISM AND THE TRAVELING-SALESMAN PROBLEM

The NP-hard traveling-salesman problem [2] is one of the most famous problems in combinatorial optimization, and the development of heuristics for its solution has occupied many researchers.

Pearl [13] uses this problem as the very first example of the constraint-deletion method. However, he does not formalize it in terms of the state-space model of problem solving. Instead, he concentrates on a description of the goal and the constraints upon it. This formulation gives the flavor of the constraint-relaxation method.

However, to perform a heuristic search, what is required is a lower bound on the cost of completing a partial solution. To get such a lower bound for a problem, we must formulate it using a well-defined set of states and operators instead. We then delete constraints upon the operators to get relaxed models of the problem, find solutions to these relaxed models (and thus heuristics for the original problem), and analyze and criticize these solutions (to develop improved heuristics).

### 2.1. *FORMALIZATION OF THE PROBLEM*

Consider a search for solutions in a state space of partial tours. From any given state, the heuristic should be an estimate of the minimum-cost completion of the partial tour. In other words, if in a given state we have completed a

partial tour from $city_a$ to $city_k$, we want to estimate the length of a minimum partial tour from $city_k$ back to $city_a$ which visits all the remaining cities.

To calculate an exact heuristic, we must construct a tour from $city_k$ to $city_a$ [denote this by $TOUR(city_k, city_a)$]. A tour consists of "visiting" all of the remaining cities in the graph (with $city_a$ visited last), using the single operator $MOVE(city_i, city_j)$, defined as:

$MOVE(city_i, city_j)$

| | |
|---|---|
| precondition list: | $ON(salesman, city_i) \wedge \neg VISITED(city_j)$ |
| add list: | $ON(salesman, city_j) \wedge VISITED(city_j)$ |
| delete list: | $ON(salesman, city_i)$ |
| cost: | $DISTANCE(city_i, city_j)$ |

The goal has been reached when, for every $city_z$ that had to be visited, $VISITED(city_z)$ is true. The successful tour consists of the sequence of applications of the $MOVE$ operator by which the goal was reached.

### 2.2. HEURISTICS FROM CONSTRAINT DELETION

At any given state, the problem is to construct a $TOUR(city_k, city_a)$ which visits all the remaining cities. We may simplify the problem by deleting either of the preconditions of $MOVE$.

If we delete the $ON(salesman, city_i)$ requirement, we allow the salesman to jump (freely) to $city_i$, and then move to $city_j$. Because of the way that the add list was specified, the salesman does not actually visit $city_i$ in this move. We get the optimal solution to this problem by traveling the shortest edge incident with each city (if there are $n$ cities to visit, we use $n$ edges, one more than in the minimum-spanning tree). The solution to this *nearest-neighbor* problem can be found in polynomial time. We will return to this relaxed model later, and apply solution criticism to refine the heuristic estimate.

If we instead delete the $\neg VISITED(city_j)$ constraint, we get a simplified problem in which the optimal solution is given by the shortest tour which visits all cities, with multiple visits allowed. There seems to be no efficient algorithm for computing such a shortest relaxed tour. Thus, this particular relaxation did not produce a useful heuristic.

If we formulate the problem in a slightly different manner, we reach different relaxed models, and thus different heuristics. Notice that when we move from $city_i$ to $city_j$, we do not assume that $city_i$ has been visited in the process—this is an artifact of the definition of the operators. In the original description of the problem, we assume that $city_i$ has been visited previously; therefore, $VISITED(city_i)$ was not included in the original add list in the



operator's definition. However, if we do add $VISITED(city_i)$ to the add list, and then delete $ON(city_i)$ from the precondition list, we arrive at a problem where the optimal solution is given by the minimum-weight set of edges $E$ such that every city is incident with an edge in $E$. The set of edges $E$ is a solution to the minimum-weight edge cover problem (which can be reduced to the minimum-weighted-matching problem and solved in polynomial time).

Thus, we have seen the spectrum of constraint relaxation in these three different relaxations of the original traveling-salesman problem. The first yielded the nearest-neighbor heuristic, the second yielded no efficiently computable heuristic, and the third was the result of reformulating the problem. We will now reconsider the first relaxation, and apply the solution-criticism methodology to it.

### 2.3. ANALYZING THE NEAREST-NEIGHBOR HEURISTIC

The nearest-neighbor solution is always a subgraph consisting of connected components, where each component of $m$ nodes is a modified tree of $m$ edges, which has either a single cycle containing equal (shortest) length edges, or a duplicated shortest edge (creating a degenerate cycle). Notice that, for problem instances with very small numbers of evenly distributed cities, this easily computed heuristic estimate may be higher than the cost of the minimum spanning tree, because it contains one more edge. As problem size increases, however, this becomes less likely.

We see that the solutions to the nearest-neighbor relaxed model contain cycles and may be disconnected. We may admissibly increase the estimate by breaking these cycles (if there are $k$ components in the subgraph, there will be $k$ cycles; we break them by deleting $k$ edges), adding the minimum-weight set of $k-1$ edges which connect the components, and then adding another shortest edge (which creates a cycle). The total length of the new edges will not be less than the total length of the removed ones, so this new heuristic is more informed. The resulting graph is simply a minimum-spanning tree with an added shortest edge (which is computable in low-order polynomial time [15]).

We may, of course, iterate the process, and apply solution criticism to this solution in turn. The next criticism is that the resulting graph is not connected to $city_k$ (because it had been visited already in the partial tour from $city_a$ to $city_k$: $city_a$ was not visited in the partial tour). We may possibly increase the estimate by deleting the added shortest edge mentioned above, and adding instead the shortest edge which connects $city_k$ to the minimum-spanning tree. The resulting graph is merely a minimum-spanning tree which includes $city_k$, except in the case where $city_k$ equals $city_a$ (the initial state of the search), in

which we get Held and Karp's minimum-weight 1-tree instead [6, 7]. Interestingly, this heuristic, which we have derived by solution criticism, is one of the most famous admissible heuristics for the traveling-salesman problem. Its performance has been thoroughly studied in [7].

## 3. CONSTRAINT DELETION ON THE EIGHT PUZZLE

In this section, we examine Pearl's formulation of the eight-puzzle domain, in order to better understand the effects of the relaxation process. We discuss three known heuristics—Manhattan distance [1], relaxed adjacency [3], and misplaced tiles [1]—which Pearl uses to demonstrate the applicability of constraint deletions.

### 3.1. FORMALIZATION OF THE PROBLEM

Following Pearl, we use three predicates to describe the problem state of the eight puzzle:

| | |
|---|---|
| $ON(x, y)$: | tile $x$ is on cell $y$ |
| $CLEAR(y)$: | cell $y$ is clear of tiles |
| $ADJ(y, z)$: | cell $y$ is horizontally or vertically adjacent to cell $z$ |

The single operator on the state space is described as follows:

| $MOVE(x, y, z)$ | |
|---|---|
| precondition list: | $ON(x, y) \wedge CLEAR(z) \wedge ADJ(y, z)$ |
| add list: | $ON(x, z) \wedge CLEAR(y)$ |
| delete list: | $ON(x, y) \wedge CLEAR(z)$ |

By removing preconditions for this operator one is creating a relaxed model of the problem. This is, of course, only one possible description of the problem. As seen in the previous section, refining or changing the set of predicates used to describe the problem will lead to different relaxations [5].

### 3.2. HEURISTICS DEVELOPED THROUGH CONSTRAINT DELETION

*Misplaced Tiles.* The most severe relaxation is to delete both $ADJ(y, z)$ and $CLEAR(z)$. In this model of the puzzle, any tile in any position may be moved into any other position, with stacking allowed. The obvious algorithm



for solving this puzzle is simply to move each tile from its current position into its goal position. Thus, the length of the optimal solution is merely the number of tiles which are not currently in their goal positions--the *misplaced tiles*.

*Relaxed Adjacency.* If we delete only the $ADJ(y, z)$ precondition, we arrive at a new puzzle in which any tile, anywhere, may swap positions with the blank. In this *relaxed-adjacency* model, optimal solutions are given by the following algorithm, first introduced by Gaschnig [3]. (It can be proven optimal by viewing the puzzle in terms of cyclic permutations [5].)

*While* any tile is out of its goal position *do*
  *If* the blank is in its own goal position,
    *then* swap with any misplaced tile
  *else* swap with the tile that belongs in the blank's position

*Manhattan Distance*

If one chooses to delete only $CLEAR(z)$ from the list of preconditions, the optimal solution length is given by the familiar Manhattan-distance heuristic. In this new puzzle, a tile may be moved into any horizontally or vertically adjacent position, with stacking allowed. Obviously, the optimal solution to this puzzle is found by moving each tile along a *shortest path* between its initial and goal positions. For any one tile, the length of this shortest path is the grid distance (horizontal plus vertical distance) between its current and goal positions. Therefore, the total solution length is merely the summation of these grid distances for each tile.

## 4. REFINING RELAXED MODELS BY SOLUTION CRITICISM

We have seen how constraint deletion can generate a number of admissible heuristics for the eight puzzle. We may consider such a heuristic function as examining certain properties of a problem. But, because of the relaxation, some of the properties are overlooked, and others exaggerated or simplified; therefore the solution proposed is infeasible for the original problem. The different relaxations we have seen are weighted heavily towards certain properties—e.g., the shortest path of a tile to its goal position (Manhattan distance), and the role of the blank in moving the tiles (relaxed adjacency). We wish to study the properties which these relaxations stress, and those which they ignore or simplify, and thereby criticize the feasibility of the simplified solution.

Manhattan distance, on the average, is the best of the heuristics discussed

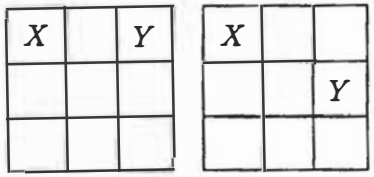



Fig. 2. Unique and nonunique shortest paths from $X$ to $Y$.

above. Applying the solution-criticism methodology, we will attempt to improve upon it by contrasting the solution plan which it suggests with that of the original problem.

### 4.1. *ANALYZING MANHATTAN-DISTANCE SOLUTIONS*

Manhattan distance can be thought of as proposing a solution for the problem. It proposes that the puzzle can be solved by moving each tile along a shortest path to its goal position. More specifically, the optimal solution in the Manhattan-distance model is a set of subgoal solutions, one for each tile. A subgoal solution is any shortest path for a given tile from its current to its goal position. In many cases, there is a single, unique shortest path—the tile is already in its correct row (column) and need only move within that row (column). In other cases, the path is not unique (see Figure 2).

We will explore only what happens to the unique shortest paths, because in these cases, the subgoal solution given by the relaxed model is uniquely determined. If it can be shown to be infeasible, we may improve the heuristic estimate. First, we present the following results about paths:

LEMMA 1. *If there exists one path from position X to position Y in the eight puzzle that is of even (odd) length, than all paths from X to Y are of even (odd) length.*

*Proof.* If the positions are colored like a checkerboard, the blank can move only from a "red" position to a "black" position—the tile positions clearly form a bipartite graph. In a bipartite graph, all paths of length 1 move between the two sides of the graph; therefore, all paths between two positions which are on opposite sides of the graph are of odd length, and all paths between two positions which are in the same side are of even length. ■

Consequently,

COROLLARY (to Lemma 1). *If there is a unique shortest path p between position X and position Y in the eight puzzle, then any alternate path will be at least 2 moves longer than p.*



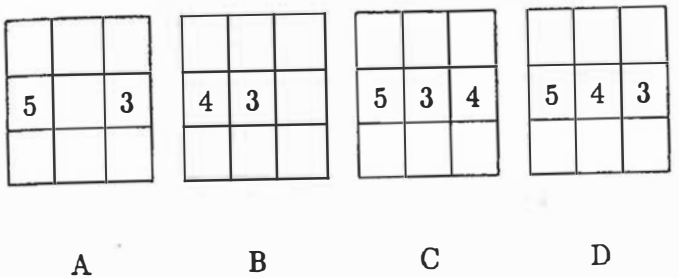



Fig. 3. Linear-conflict examples.

In comparing the subgoal solutions with the actual optimal solutions, one notices that the unique shortest paths of two tiles occasionally conflict. In these cases, one tile may be forced to take an alternate path, increasing the solution length by at least two (from the corollary to Lemma 1). In fact, there are several distinct cases in which this phenomenon occurs (the examples below indicate some of these). In general, these conflicts can only exist in a given line when two or more tiles have both their current and goal positions in that line—in that case, there are at least two unique shortest paths and the possibility of a local subgoal conflict.

The idea of shortest paths is only brought to our attention by studying the optimal solutions to this relaxed problem. We must then attempt to look critically at this concept, which influences the heuristic offered by the relaxed model, and attempt to gauge the degree to which it affects the solution to the relaxed model. Once one can recognize and characterize the cases where the unique shortest paths are necessarily violated in the optimal solution to the original problem, one may determine how to compensate for these oversimplifications. Before describing a precise method for doing so, we examine several examples of this phenomenon.

*Examples of Conflicting Shortest Paths.* Figure 3 illustrates four typical examples of conflicts between unique shortest paths.

In example A, either the 5 or the 3 must move outside of the middle row to make room for the other to pass. Therefore, two should be added to the estimate of Manhattan distance (one for the move out of the line and another for the move returning to the line).

B shows a conflict in which a tile which had previously been "solved" presents an obstacle to another tile. This conflict contradicts the intuition that solved subgoals will not be disturbed. To resolve this conflict, either the 4 or the 3 will have to follow a nonshortest path, adding at least two moves to the Manhattan-distance estimate.

In C, the 5 tile is in conflict with the 3 and the 4. Clearly, either the 5 has to move out of the way (two extra moves) and allow the others to pass to their goal positions, or the 3 and 4 have to move and allow the 5 to pass (four extra moves).

D illustrates the most complex case, where each tile is in conflict with the other two. This can only be resolved by moving two of the tiles off line. One should therefore add four to the Manhattan-distance estimate when this case is recognized.

If one can devise an algorithm for tabulating the additional moves forced by conflicting subgoals, one can add that total to the Manhattan-distance estimate and create a new admissible heuristic for this problem. Intuitively, one examines the puzzle state, row by row and column by column, and adds to the Manhattan distance the minimum number of additional moves necessary to resolve the conflicts within each row and column; therefore, this *linear-conflict* estimate is still a lower bound on the actual optimal solution length (a precise algorithm is given below). To give some idea of the relative informedness of this heuristic, we compare its estimates with those of Manhattan distance, relaxed adjacency, and misplaced tiles, for the puzzle instances shown in Figure 4.

While the linear-conflict heuristic is clearly more informed than the Manhattan-distance heuristic, one notes that, because the space of relaxations is a partial order [5], one cannot presuppose the relationship between the relative informedness of the linear-conflict and relaxed-adjacency heuristics. In fact, there are instances of the fifteen puzzle (a 4×4 frame with 15 tiles) for which

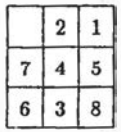
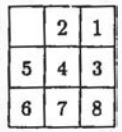
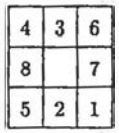
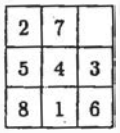
| Puzzle |  |  |  |  |
|---|---|---|---|---|
| Misplaced Tiles | 4 | 4 | 8 | 7 |
| Relaxed Adjacency | 6 | 6 | 10 | 10 |
| Manhattan Distance | 6 | 6 | 22 | 14 |
| Linear Conflict | 8 | 12 | 22 | 24 |
| Actual Distance | 22 | 20 | 26 | 26 |

Fig. 4. Comparison of heuristic estimates.



relaxed adjacency is more informed than linear conflict [e.g. the state (B 1 3 7 4 5 2 6 9 13 10 11 8 12 14 15)].

### 4.2. THE LINEAR-CONFLICT HEURISTIC FUNCTION

We present an algorithm, LC (Figure 5), which calculates the heuristic function described informally above by performing an analog of plan criticism upon the unique shortest paths which exist in any given line. We first define a linear conflict:

DEFINITION. Two tiles $t_j$ and $t_k$ are in a *linear conflict* if $t_j$ and $t_k$ are in the same line, the goal positions of $t_j$ and $t_k$ are both in that line, $t_j$ is to the right of $t_k$, and the goal position of $t_j$ is to the left of the goal position of $t_k$.

Theorems establishing the correctness of the heuristic and its monotonicity (and hence, admissibility) may be found in [5].

*Computational Complexity of the Linear-Conflict Heuristic.* The calculation of $MD(s)$ requires $O(N)$ operations. To calculate $LC(s)$ requires, for each line of tiles, $O(N)$ operations in the worst case. Since there are $2\sqrt{N+1}$ lines, it requires $N^{1.5}$ operations. However, during a search, one can calculate the heuristic estimate for a given state more efficiently, assuming that one has the estimate for its parent in the search space. Thus, even in a naive implementation, $MD(s)$ costs $O(1)$, and $LC(s)$ costs $O(N)$. In our implementation, we reduced both calculations to table lookup. To prepare the linear-conflict table, we precomputed the linear conflicts possible in a line. We stored with each tile in the state two numbers indicating whether it is in the goal row and column, and if so, where its goal position is in its current row and column. Thus, in our implementations, $LC(s)$ merely costs a small number of table-lookup operations. In fact, the calculation of $LC(s)$ caused the search program for the fifteen puzzle to be, on average, only 5% slower in examining each node, and this was more than compensated for by the dramatic decrease in the number of nodes that needed to be examined when the linear-conflict heuristic was used (cf. Section 4.3).

### 4.3. EMPIRICAL ANALYSIS OF THE LINEAR-CONFLICT HEURISTIC

We have developed a new heuristic for the eight puzzle (generalizable to the $N$ puzzle), which is more informed than the Manhattan-distance heuristic (which had been known to be, on average, the most informed heuristic for the

**Begin {Algorithm LC}**

{$s$ is the current state}

{$L$ is the size of a line (row or column) in the puzzle. $L = \sqrt{N+1}$}

{$C(t_j, r_i)$ is the number of tiles in row $r_i$ with which $t_j$ is in conflict}

{$C(t_j, c_i)$ similarly}

{$lc(s, r_j)$ is the number of tiles that must be removed from row $r_j$ to resolve the linear conflicts}

{$lc(s, c_j)$ similarly}

{$md(s, t_i)$ is the Manhattan Distance of tile $t_i$}

{$MD(s)$ is the sum of the Manhattan Distances of all the tiles in $s$}

{$LC(s)$ is the minimum number of additonal moves needed to resolve the linear conflicts in $s$}

*For each* row $r_i$ in the state $s$

$lc(s, r_i) = 0$

*For each* tile $t_j$ in $r_i$

determine $C(t_j, r_i)$

*While* there is a non-zero $C(t_j, r_i)$ value

Find $t_k$ such that there is no

$C(t_j, r_i) > C(t_k, r_i)$ { As $t_k$ is the tile with the most conflicts,

Fig. 5. Algorithm LC.



we choose to move it out of $r_i$ }

$C(t_k, r_i) = 0$

*For every* tile $t_j$ which had been in conflict with $t_k$

$C(t_j, r_i) = C(t_j, r_i) - 1$

$lc(s, r_i) = lc(s, r_i) + 1$

{ Check similarly for linear conflicts in each column $c_i$, computing $lc(s, c_j)$. }

$LC(s) = 2[lc(s, r_1) + \ldots + lc(s, r_L) + lc(s, c_1) + \ldots + lc(s, c_L)]$

*For each* tile $t_j$ in $s$

determine $md(s, t_i)$

$MD(s) = ms(s, t_i) + \ldots + md(s, t_n)$.

$h'(s) = MD(s) + LC(s)$

**End {Algorithm LC}**

Fig. 5. (Continued).

problem). The pruning power of the two heuristics was examined by solving the 100 random fifteen puzzles used in the tests of [8], and comparing the number of node expansions (this measure is proportional to search time).

Figure 6 in the Appendix (an extension of a similar table in [8]) shows that, for the fifteen puzzle, the average number of states examined using linear conflict is only one-eighth of the average number of states examined using Manhattan distance. For 61 out of 100 puzzle instances, linear conflict performed less than 20% of the search required by Manhattan distance; in only 7 cases did it exceed 30%. Manhattan distance caused the search to examine over 100 million states in 40 puzzle instances, and over 500 million states in 17 puzzle instances, while linear conflict caused the search to examine over 100 million states in only 11 puzzle instances, and no state required the examination of 500 million states. If the puzzles were sorted by problem difficulty (number of states examined), one would see that the linear-conflict heuristic demonstrates more pruning power on more difficult fifteen-puzzle problem instances. For the 20 most difficult problems, a search using linear conflicts would be over ten times as fast as one using Manhattan distance. We refer the interested reader to [5], where additional experiments are described.

## 5. DISCUSSION OF THE SOLUTION-CRITICISM METHOD

The process of relaxation allows us to concentrate on certain aspects of the problem while ignoring others. This often makes the problem easier to analyze, and perhaps easier to solve algorithmically. However, these solutions are poor approximations of a feasible solution to the original problem. For example, one of the relaxed traveling-salesman problems allows us to simplify the problem of visiting all the cities in a proper tour into the many subproblems of visiting each city, without regard to the connections between these subtours. By correcting this disconnected solution, we arrive at the well-known minimum-spanning-tree and minimum-weight 1-tree heuristics. Similarly, the Manhattan-distance relaxed model allows us to consider optimal solutions for each tile, without regard to the global conflicts which may result because of the interaction of these subgoal solutions. We may improve upon this relaxed model by taking account of those constraints which it overlooks.

The method of solution criticism is a general approach to restoring the global view to these myopic relaxed solutions, by comparing them with actual solutions to gain an understanding of those global considerations that they overlook. In many cases, we may be able to increase the estimate provided by the relaxed solution, either by refining it into a more feasible solution, or simply by adding some abstract lower-bound measure of what has been overlooked.



We note that several traveling-salesman heuristics, developed over years using sophisticated algorithmic methods, were shown to evolve naturally from solution criticism. Through they are not new for that problem, they can be derived systematically, suggesting the potential of the approach for less well-studied or newly posed problems. For example, although the eight puzzle has been used for many years as a research example to demonstrate the development of heuristics, the Manhattan-distance heuristic has not been improved upon. We believe that our improvement, the linear-conflict heuristic, is difficult to find directly without following the methodology suggested here.

Although implementing the constraint-relaxation with solution-criticism method is beyond the scope of this paper, this could be accomplished using the type of plan-debugging facilities common to classical AI planning programs. Consider that at any point during a heuristic search, there exists a partial plan which leads from the problem's initial state to the current state. To evaluate the partial plan, or equivalently, the cost of completing it, one consults the solution to a relaxed model. This will provide an abstract sketch of the plan which ultimately will become necessary to reach the goal state.

Solution criticism is merely a careful examination of the feasibility of this plan sketch. For example, solutions to the nearest-neighbor relaxed model will have unordered sets of operators on which no total ordering can be imposed (i.e., the cycles in the graph indicate cycles of temporal precedence among operators). Solutions in the Manhattan-distance relaxed model which contain linear conflicts will contain conflicting operators (the preconditions for one operator are destroyed by the postconditions of another). General conditions such as these could easily be detected by the plan critics used in nonlinear planners.

## 6. CONCLUSION

The process of solution criticism, i.e. criticizing the solutions to relaxed models, is suggested as a valuable addition to the method of problem relaxation by constraint deletion. The preceding analysis and empirical data show that one can develop very powerful heuristics easily, be attempting to understand the infeasibility of a proposed relaxed solution, and recovering some of the information that was lost in the relaxation. As evidence of this, we have demonstrated how such criticism can be used to derive powerful admissible heuristics for the traveling-salesman problem and the eight puzzle, which equal or surpass the product of years of human study.

The combination of constraint deletion and solution criticism is analogous to techniques used by human problem solvers. Relaxing a problem results in a preliminary plan for solution, which must then be modified dynamically by the

| NO | MD INIT | LC INIT | LEN | MD STATES | LC STATES | PCT |
|---|---|---|---|---|---|---|
| 01 | 41 | 43 | 57 | 276,361,933 | 12,205,623 | 4.4 |
| 02 | 43 | 43 | 55 | 15,300,442 | 4,556,067 | 29.8 |
| 03 | 41 | 41 | 59 | 565,994,203 | 156,590,306 | 27.6 |
| 04 | 42 | 42 | 56 | 62,643,179 | 9,052,179 | 14.5 |
| 05 | 42 | 44 | 56 | 11,020,325 | 2,677,666 | 24.5 |
| 06 | 36 | 40 | 52 | 32,201,660 | 4,151,682 | 12.9 |
| 07 | 30 | 30 | 52 | 387,138,094 | 97,264,710 | 25.1 |
| 08 | 32 | 36 | 50 | 39,118,937 | 3,769,804 | 9.6 |
| 09 | 32 | 36 | 46 | 1,650,696 | 88,588 | 5.4 |
| 10 | 43 | 45 | 59 | 198,758,703 | 48,531,591 | 24.4 |
| 11 | 43 | 45 | 57 | 150,346,072 | 25,537,948 | 17.0 |
| 12 | 35 | 35 | 45 | 546,344 | 179,628 | 32.9 |
| 13 | 36 | 38 | 46 | 11,861,705 | 1,051,213 | 8.9 |
| 14 | 41 | 43 | 59 | 1,369,596,778 | 53,050,799 | 3.9 |
| 15 | 44 | 46 | 62 | 543,598,067 | 130,071,656 | 24.0 |
| 16 | 24 | 26 | 44 | 17,984,051 | 2,421,878 | 13.5 |
| 17 | 46 | 46 | 66 | 609,399,560 | 100,843,886 | 16.6 |
| 18 | 43 | 43 | 56 | 23,711,067 | 5,224,645 | 22.0 |
| 19 | 36 | 38 | 46 | 1,280,495 | 385,369 | 30.1 |
| 20 | 38 | 38 | 62 | 17,954,470 | 3,642,638 | 20.2 |
| 21 | 34 | 36 | 54 | 257,084,815 | 49,290,448 | 19.1 |
| [illegible] | | | | | | |
| 28 | 38 | 40 | 54 | 117,076,111 | 10,085,238 | 8.6 |
| 30 | 35 | 35 | 47 | 2,196,593 | 680,254 | 31.0 |
| 31 | 38 | 40 | 50 | 2,351,811 | 538,886 | 22.9 |
| 32 | 43 | 45 | 59 | 661,041,936 | 183,341,087 | 27.7 |
| 33 | 42 | 42 | 60 | 480,637,867 | 28,644,837 | 6.0 |
| 34 | 36 | 42 | 52 | 20,671,552 | 1,174,414 | 5.7 |
| 35 | 39 | 39 | 55 | 47,506,056 | 9,214,047 | 19.4 |
| 36 | 36 | 38 | 52 | 59,802,602 | 4,657,636 | 7.8 |
| 37 | 40 | 42 | 58 | 280,078,791 | 21,274,607 | 7.6 |
| 38 | 41 | 43 | 53 | 24,492,852 | 4,946,981 | 20.1 |
| 39 | 35 | 35 | 49 | 19,355,806 | 3,911,623 | 20.2 |
| 40 | 36 | 38 | 54 | 63,276,188 | 13,107,557 | 20.7 |
| 41 | 36 | 40 | 54 | 51,501,544 | 12,388,516 | 24.1 |
| 42 | 30 | 32 | 42 | 877,823 | 217,288 | 24.8 |
| 43 | 48 | 54 | 64 | 41,124,767 | 7,034,879 | 17.1 |
| 44 | 32 | 38 | 50 | 95,733,125 | 3,819,541 | 4.0 |
| 45 | 39 | 39 | 51 | 6,158,733 | 764,473 | 12.4 |
| 46 | 35 | 41 | 49 | 22,119,320 | 1,510,387 | 6.8 |
| 47 | 35 | 35 | 47 | 1,411,294 | 221,531 | 15.7 |
| 48 | 39 | 39 | 49 | 1,905,023 | 255,047 | 13.4 |
| 49 | 33 | 37 | 59 | 1,809,933,698 | 203,873,877 | 11.3 |
| 50 | 39 | 41 | 53 | 63,036,422 | 6,225,180 | 9.9 |

| NO | MD INIT | LC INIT | LEN | MD STATES | LC STATES | PCT |
|---|---|---|---|---|---|---|
| 51 | 44 | 44 | 56 | 26,622,863 | 4,683,054 | 17.6 |
| 52 | 38 | 42 | 56 | 377,141,881 | 33,691,153 | 8.9 |
| 53 | 50 | 50 | 64 | 465,225,698 | 125,641,730 | 27.0 |
| 54 | 40 | 42 | 56 | 220,374,385 | 26,080,659 | 11.8 |
| 55 | 29 | 31 | 41 | 927,212 | 163,077 | 17.6 |
| 56 | 29 | 33 | 55 | 1,199,487,996 | 166,183,825 | 13.9 |
| 57 | 36 | 36 | 50 | 8,841,527 | 3,977,809 | 45.0 |
| 58 | 37 | 39 | 51 | 12,955,404 | 3,563,941 | 27.5 |
| 59 | 35 | 37 | 57 | 1,207,520,464 | 90,973,287 | 7.5 |
| 60 | 48 | 48 | 66 | 3,337,690,331 | 256,537,528 | 7.7 |
| 61 | 31 | 35 | 45 | 7,096,850 | 672,959 | 9.5 |
| 62 | 43 | 45 | 57 | 23,540,413 | 8,463,998 | 36.0 |
| 63 | 40 | 42 | 56 | 995,472,712 | 20,999,336 | 2.1 |
| 64 | 31 | 35 | 51 | 260,054,152 | 43,522,756 | 16.7 |
| 65 | 31 | 35 | 47 | 18,997,681 | 2,444,273 | 12.9 |
| 66 | 41 | 43 | 61 | 1,957,191,378 | 394,246,898 | 20.1 |
| 67 | 28 | 30 | 50 | 252,783,878 | 47,499,462 | 18.8 |
| 68 | 31 | 35 | 51 | 64,367,799 | 6,959,507 | 10.8 |
| 69 | 37 | 39 | 53 | 109,502,359 | 5,186,887 | 4.7 |
| 70 | 30 | 32 | 52 | 161,042,871 | 49,161,673 | 30.5 |
| 71 | 30 | 34 | 44 | 2,342,078 | [illegible] | |
| [illegible] | | | | | | |
| 78 | 38 | 40 | 54 | 540,660 | 107,085 | 19.8 |
| 80 | 43 | 43 | 57 | 132,945,856 | 39,691,475 | 29.9 |
| 81 | 39 | 41 | 53 | 9,982,569 | 1,088,123 | 10.9 |
| 82 | 40 | 44 | 62 | 5,506,801,123 | 203,606,265 | 3.7 |
| 83 | 31 | 37 | 49 | 65,533,432 | 2,155,880 | 3.3 |
| 84 | 37 | 41 | 55 | 106,074,303 | 17,323,672 | 16.3 |
| 85 | 32 | 32 | 44 | 2,725,456 | 933,953 | 34.3 |
| 86 | 35 | 37 | 45 | 2,304,426 | 237,466 | 10.3 |
| 87 | 34 | 36 | 52 | 64,926,494 | 7,928,514 | 12.2 |
| 88 | 43 | 45 | 65 | 6,009,130,748 | 422,768,851 | 7.0 |
| 89 | 36 | 40 | 54 | 166,571,097 | 29,171,607 | 17.5 |
| 90 | 36 | 40 | 50 | 7,171,137 | 649,591 | 9.1 |
| 91 | 41 | 41 | 57 | 602,886,858 | 91,220,187 | 15.1 |
| 92 | 37 | 39 | 57 | 1,101,072,541 | 68,307,452 | 6.2 |
| 93 | 34 | 34 | 46 | 1,599,909 | 350,208 | 21.9 |
| 94 | 45 | 45 | 53 | 1,337,340 | 390,368 | 29.2 |
| 95 | 34 | 36 | 50 | 7,115,967 | 1,517,920 | 21.3 |
| 96 | 35 | 37 | 49 | 12,808,564 | 1,157,734 | 9.0 |
| 97 | 32 | 34 | 44 | 1,002,927 | 166,566 | 16.6 |
| 98 | 34 | 34 | 54 | 183,526,883 | 41,564,669 | 22.6 |
| 99 | 39 | 39 | 57 | 83,477,694 | 18,038,550 | 21.6 |
| 100 | 38 | 40 | 54 | 67,880,056 | 17,778,222 | 26.2 |
| | | | TOTAL | 2.998476E10 | 3.759631E9 | 12.5 |

Fig. 6. Comparative performance: search on random fifteen-puzzle instances; cf. [8]. MD INIT: initial heuristic estimate for Manhattan distance; LC INIT: initial heuristic estimate for linear conflict; LEN: length of optimal solution; MD STATES: total number of states examined using Manhattan distance; LC STATES: total number of states examined using linear conflict; PCT: $100 \times [(\text{LC STATES})/(\text{MD STATES})]$.

problem solver. This, of course, is a generally effective approach (utilized in everyday life, as well as scientific analysis) to understanding complex situations—make simplifying assumptions, and then, as understanding increases (aided by that simplified model) or circumstances demand, reconsider factors ignored by the simplification. We believe that the process of relaxation and subsequent tightening captures and codifies one of the methods used by humans in coping with hard problems.

## APPENDIX

In Figure 6, the performance of various heuristics on the fifteen puzzle is compared.

## REFERENCES


1. J. Doran and D. Michie, Experiments with the graph traverser program, in *Proc. Roy. Soc. London Ser. A* 294:235–259 (1966).
2. M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, New York, 1979.
3. J. Gaschnig, A problem similarity approach to devising heuristics: First results, in *Proceedings of the International Joint Conference on Artificial Intelligence*, Tokyo,, 1979.
4. G. Guida and M. Somalvico, A method for computing heuristics in problem solving, *Inform. Sci.* 19:251–259 (1979).
5. O. Hansson, A. Mayer, and M. Yung, Generating Admissible Heuristics by Criticizing Solutions to Relaxed Models, Technical Report 219-85, Dept. of Computer Science, Columbia Univ., 1985.
6. M. Held and R. M. Karp, The traveling-salesman problem and minimum spanning trees, *Oper. Res.* 18:1138–1162 (1970).
7. M. Held and R. M. Karp, The traveling-salesman problem and minimum spanning trees: Part II, *Math. Programming* 1:6–25 (1971).
8. R. E. Korf, Depth-first iterative deepening: An optimal admissible tree search, *Artif. Intell.* 27:97–109 (1985).
9. E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys (Eds.), *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley, New York, 1985.
10. D. Mandrioli, A. Sangiovanni-Vincentelli, and M. Somalvico, Toward a theory of problem solving, in *Topics in Artificial Intelligence* (A. Marzollo, Ed.), Springer-Verlag, Vienna, 1976.
11. J. Mostow and A. E. Prieditis, Discovering admissible heuristics by abstracting and optimizing: A transformational approach, in *Proceedings of the International Joint Conference on Artificial Intelligence*, Detroit, 1989.





12. A. Newell and H. A. Simon, *Human Problem Solving*, Prentice-Hall, Englewood Cliffs, N.J., 1972.
13. J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Soving*, Addison-Wesley, Reading, Mass., 1984.
14. G. Polya, *How to Solve It*, 2nd ed., Princeton U.P., Princeton, 1973.
15. R. E. Tarjan, *Data Structures and Network Algorithms*, SIAM, Philadelphia, 1983.
16. M. Valtorta, A result on the computational complexity of heuristic estimates for the $A^*$ algorithm, *Inform. Sci.* 34:48–59 (1984).


*Received 10 November 1989; revised 25 January 1990, 6 April 1990, 25 May 1990*