Generating Admissible Heuristics by Criticizing Solutions to Relaxed Models

Othar Hansson

Andrew E. Mayer

Mordechai M. Yung[1]

CUCS-219-85

Department of Computer Science

Columbia University

New York, N.Y. 10027

December, 1985

CUCS-219-85

Abstract

This paper examines one paradigm used to develop admissible heuristics: problem relaxation [10, 11, 32]. This consists of three steps: simplify (or relax) a problem, solve the simplified problem, and use that solution as advice to guide the search for a solution to the original problem. We introduce an extension to this methodology which exploits the simplicity of relaxed models. By criticizing the feasibility of a relaxed solution, we arrive at a closer approximation of the solution to the original problem. This solution-criticism process recovers some of the information lost by relaxation, and yields more powerful admissible heuristics than by relaxation alone. We apply our methodology to the Traveling-Salesman problem and the N Puzzle. For the Traveling-Salesman Problem, it yields the well known, admissible minimum spanning tree heuristic. For the Eight and Fifteen Puzzles (in general the N puzzle), it yields a new heuristic which performs significantly better than all previously known heuristics.

Keywords: *Problem Solving, State-Space, Heuristic Search, Admissible Heuristic Function, Problem Relaxation, Automatic Heuristic Generation, N Puzzle, Traveling-Salesman Problem*

---

[1] Supported in part by NSF Grant MCS-8303139 and an IBM Fellowship.

1. Introduction

Domain-specific heuristics enable us to solve certain previously intractable problems by intelligently directing the search for solutions. At this point, we do not fully understand how heuristics are generated, and the development of heuristics is the major bottleneck in constructing intelligent systems [23, 24]. Many have attempted to understand the nature of heuristics [13, 22, 23, 24, 27, 28, 29, 30, 32, 35], so as to formulate a general methodology for developing them. The goal is to systematically, and perhaps automatically develop heuristics for arbitrary problems by applying this methodology.

In this paper, we concentrate on the state-space model of problem-solving [29, 30], and examine one paradigm [4, 10, 11, 32] for the systematic generation of heuristics (we follow the notation and examples of Pearl [32]). This paradigm offers a method for deriving heuristics by examining optimal solutions to simplified models of a given problem. More specifically, the paradigm outlines a three-step methodology: first, simplify the problem; second, solve the simplified problem (preferably algorithmically); and third, use information gained by solving this simplified problem as advice 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 improved methodology towards developing admissible heuristics for two problems: the N Puzzle (e.g., the Eight and Fifteen Puzzles) [5, 10, 16, 17, 19, 26, 31, 32, 40, 42] and the Traveling-Salesman Problem [1, 2, 3, 9, 14, 15, 25, 33, 34]. These problems have been used to study and develop heuristic problem-solving techniques for more than twenty years. The N Puzzle has been used by Pearl and others as an example of the use of his aforementioned paradigm: we use his formulation, and apply our improved methodology to develop a new admissible heuristic for the problem. Our new heuristic is more powerful than any previously known admissible heuristic (Manhattan Distance had been the best known): in tests on 1000 random instances of the Eight Puzzle, search using the new heuristic examined fewer than half as many states, on average, as search using Manhattan Distance. In tests on 100 random Fifteen Puzzle instances, search using the new heuristic examined one eighth as many states. We also demonstrate a similar formulation for the Traveling-Salesman Problem. We use Pearl's paradigm to develop admissible heuristics, and then apply solution-criticism to refine them. Interestingly, this leads to Held and Karp's well known minimum spanning tree heuristic [14, 15, 21].

1.1. Background

1.1.1. Overview of Heuristic Search

The state-space approach to problem-solving considers a problem as a quadruple, $\{S, O, I \in S, G \subseteq S\}$. 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*. This problem 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, $O_1, O_2, \ldots O_n$ 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. One such method, the celebrated A* algorithm (originally in [12]; also [6, 31, 32]) 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, $f'(s) = g(s) + h'(s)$.

To simplify the following discussion, we give the following definitions:

Definition 1: A heuristic function, $h'(s)$, is said to be *admissible* if $\forall s\,(h'(s) \leq h(s))$

Definition 2: A heuristic function, $h'(s)$, is said to be *monotone* if $\forall s, s'$ such that s' is a successor of s, $(f'(s) \leq f'(s'))$ (recall that $f'(s)$ is determined by $h'(s)$). Monotonicity implies admissibility [32].

Definition 3: A heuristic function, $h_1'(s)$, is said to be *more informed* than another heuristic function, $h_2'(s)$, if $\forall s\,(h_2'(s) \leq h_1'(s))$, 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 to search for optimal solutions, despite possible extra time required to do so. If A* uses an admissible heuristic, it is guaranteed to find optimal solutions [32]. 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 a 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 fewer number of states if it uses $h_1'(s)$ instead of $h_2'(s)$ [32]: $h_1'(s)$ is said to have more *pruning power* than $h_2'(s)$. Therefore, if it is known that $h_1'(s)$ is never less than $h_2'(s)$, then the search time (measured in number of states examined) using $h_1'(s)$ is guaranteed not to exceed the search time using $h_2'(s)$. However, the actual computation time is only linearly related to the number of states examined, and is, in fact, equal to the number of states multiplied by the computational effort needed to calculate the heuristic estimate [30]. Therefore, in attempting to improve heuristics, one must consider the complexity of the heuristic function as well as its informedness.

Recently, Korf [20] has examined a depth-first variant of A*: Iterative Deepening A* (IDA*). If the state-space we wish to search is a tree, or a graph which closely approximates a tree, IDA* is asymptotically optimal in terms of both time and space requirements. In order to study heuristic performance in the large state-spaces of the N Puzzles, we use the IDA* algorithm in our experiments (IDA* requires only $O(\log n)$ space to search a tree with n states). To guarantee optimality of solutions, IDA* requires the use of a monotone heuristic.

1.1.2. The N Puzzle and the Traveling-Salesman Problem

The Eight (figure 1-1) and Fifteen Puzzles are classic examples of small, well defined, and conceptually simple problems which are sufficiently complex to exhibit interesting phenomena: therefore, they serve as popular testing grounds for heuristic search and problem-solving methods. In particular, these problems are used to demonstrate the development of heuristics in [10, 11, 32]. The Eight Puzzle consists of a 3x3 frame containing 8 numbered, sliding tiles (the Fifteen Puzzle is a 4x4 frame with 15 tiles). One of the positions in the frame does not contain a tile: this space is called the "blank," and is given the number '0' for notational purposes. A state can then be considered as an ordered 9-tuple ($T_0\ T_1\ T_2\ T_3\ T_4\ T_5\ T_6\ T_7\ T_8$), understood to correspond to Figure 1-2. 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 1-1 shows the goal state used in this paper). The state-space for the Eight Puzzle contains $\frac{9!}{2}$ states, and the state-space for the Fifteen Puzzle contains $\frac{16!}{2}$ states [40, 45].

	1	2
3	4	5
6	7	8

Figure 1-1: Eight Puzzle goal state:
(0 1 2 3 4 5 6 7 8)
where 0 is the blank

T_0	T_1	T_2
T_3	T_4	T_5
T_6	T_7	T_8

Figure 1-2: Standard tile positions
($T_0\ T_1\ T_2\ T_3\ T_4\ T_5\ T_6\ T_7\ T_8$)

The NP-Hard Traveling-Salesman Problem [9] has also been used to explore the development of lower bounds (i.e., admissible heuristics) [1, 14, 32]. The problem is that of planning a shortest route for a salesman who must visit a number of cities and then return to his home city. In more mathematical terms, we must find a shortest closed Hamiltonian tour through n vertices. While recent work on this problem has concentrated on methods for finding near-optimal solutions quickly, early work was concerned with branch-and-bound techniques (branch-and-bound may be considered a generalized form of A^*) for finding optimal solutions by using admissible heuristics.

1.1.3. Generating Heuristics through Simplified Problems

We focus our attention on the work of Pearl [32], who proposes that one natural method of developing good heuristics is to "consult simplified models of the problem domain" [10, 11, 32]. He observes four general methods for the generation of these simplified models. The first method does so by deleting constraints on the applicability of operators in the state-space. A second method simplifies a problem by adding constraints to it: this reduces the size of the search space and results in a more directed (therefore faster) search. A third technique transforms the representation of the original problem into an analogous one within the domain of an expert problem-solver. Finally, Pearl mentions a probabilistic model, in which insufficient knowledge about a problem allows us to make only a statistically-based appraisal of the cost of solving it.

In this paper, we examine the first method - constraint relaxation - because it is a systematic method which guarantees the generation of *admissible* heuristics. When constraints are removed from a problem, new edges and nodes are introduced into the state-space graph. Clearly, the shortest path between any two given 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.

One can identify two properties of the heuristic information provided by the relaxed models: *simplicity* refers to the computational effort required to solve the relaxed problem and calculate the heuristic, and *proximity* refers to how closely the relaxed solution approximates the actual solution. In general, the more relaxed a model is, the less proximity and the more simplicity it has. The challenge is to increase the proximity of the relaxed model without significantly decreasing the simplicity of the heuristic calculation. If we translate this to the vocabulary of theoretical computer science, the challenge is to discover tight lower bounds which are easy to derive and compute.

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 to those of the original one. If we can admissibly recover any information that was lost in the process of relaxation, we will have created a more informed admissible estimate than the heuristic from the relaxed model alone.

1.2. Outline of the Paper

In section 2 we discuss the constraint relaxation model proposed by Pearl and examine its effectiveness on the Eight Puzzle. In section 3 we introduce and discuss our refinement of Pearl's model. We use our method to generate a new heuristic for the N Puzzle (the Linear Conflict heuristic), and we prove its monotonicity. We then show how this process may be iterated: we apply the method again to develop a slightly more powerful monotone heuristic. Based on experimental data, we chart the effectiveness of the Linear Conflict heuristic, as compared to known heuristics for this problem. Section 4 we apply our method to the Traveling-Salesman Problem, and by criticizing solutions to relaxed models we derive a well known admissible heuristic for this classic problem. In Section 5, we review the solution-criticism method in light of these two examples. Section 6, the conclusion, summarizes our results.

2. Examples of Heuristics from Relaxation: Constraint-Deletion on the N Puzzle

In this section, we examine Pearl's formulation of the N puzzle domain, in order to better understand the effects of the relaxation process. We discuss three known heuristics - Manhattan Distance [5], Relaxed Adjacency [10], and Misplaced Tiles [5] - which Pearl uses to demonstrate the applicability of

constraint-deletion. Aside from these heuristics, we use the constraint-deletion method to generate two new relaxed models.

2.1. Formalization of the Problem

To use the constraint-deletion method, one must formally represent the problem in terms of the states, operators and constraints upon those operators. Pearl uses a set of STRIPS-like [8] 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) , CLEAR (z) , ADJ (y,z)
add list	:	ON (x,z) , CLEAR (y)
delete list	:	ON (x,y) , CLEAR (z)

The essence of Pearl's method is that 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. One may either refine the predicates used, or describe the problem using a different set of predicates, as we demonstrate in both of our new relaxed models.

2.2. Manhattan Distance

If one chooses to delete CLEAR(z) from the list of preconditions, one generates a new model of the puzzle, in which the optimal solution length is given by the 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.

2.3. Relaxed Adjacency

One may instead choose to delete the ADJ(y,z) precondition. This results in 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 [10] but never proven to be optimal (see Appendix I.1 for the proof of an upper bound for this estimate and a proof of its optimality):

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

2.4. Misplaced Tiles

Another obvious 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".

2.5. The Checkerboard Relaxed Adjacency Model

In the original problem, 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, the red squares form one side of the bipartite graph, and the black squares the other side (see Figure 2-1). In the original 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 graph.

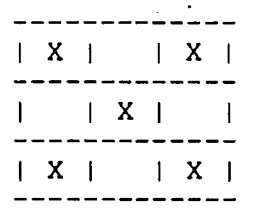

Figure 2-1: Checkerboard model

This model's proximity can be thought of as being somewhere between the original model and the Relaxed Adjacency model, because one has deleted only a part of the adjacency requirement. In this new model, one can think of any given tile position as being "adjacent" to half of the other positions. To formalize this, we define three new STRIPS-like predicates and change the preconditions for MOVE:

```
RED(y)                  :  y is a red position
BLACK(y)                :  y is a black position
DIFF-COLOR(x,y)         :  RED(x) ⊕ RED(y)

MOVE(x,y,z)  :

     precondition list  :  ON(x,y),CLEAR(z),DIFF-COLOR(y,z)
     add list           :  ON(x,z),CLEAR(y)
     delete list        :  ON(x,y),CLEAR(z)
```

In short, if a black position is blank (i.e., clear), only tiles in red positions may move into it, and *vice versa*. Unfortunately, this simplified model is not simple enough - we have yet to find an algorithm that solves it optimally. This points out a limitation of the constraint-deletion method, as well as the simplicity/proximity tradeoff. Of course, one may search to find solutions in any simplified model lacking an algorithm, in the unlikely hope that the search time will be short [44].

2.6. The Checkerboard Misplaced Model

One may generate a simplified version of the Checkerboard Relaxed Adjacency Model by deleting CLEAR(*z*) from the precondition list, producing the Checkerboard Misplaced Model. This can be solved optimally by the following algorithm:

```
While any tile is out of its goal position do
  If the tile is in the same half of the puzzle
     as its goal position
     then move the tile into any position
       in the opposite half of the puzzle
     else move the tile into its goal position
```

This heuristic can estimate as high as 16 moves, for the puzzle shown in Figure 2-2 (Relaxed Adjacency estimates only 10).

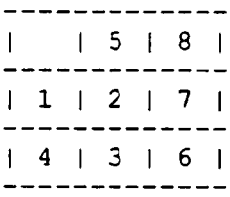

Figure 2-2: Checkerboard Misplaced = 16
(0 5 8 1 2 7 4 3 6)

2.7. Summary and Remarks on the Relaxed Models

Notice that some of the models presented are, in fact, relaxations of already relaxed models. For example, the Misplaced Tiles model is a relaxation of the Relaxed Adjacency model, since it can be generated by deleting a precondition in that model. Clearly, any relaxation of a given model will have solution lengths no longer than that model. Consequently, the Relaxed Adjacency heuristic is more informed than the Misplaced Tiles heuristic. Figure 2-3 shows the "relaxation space," which is a partial order (ordered by the informedness of the heuristics) of the models described above. This is analogous to hierarchical abstraction spaces (e.g., ABSTRIPS [38]).

The order is a partial one, because some models may be independent of one another, in that they are

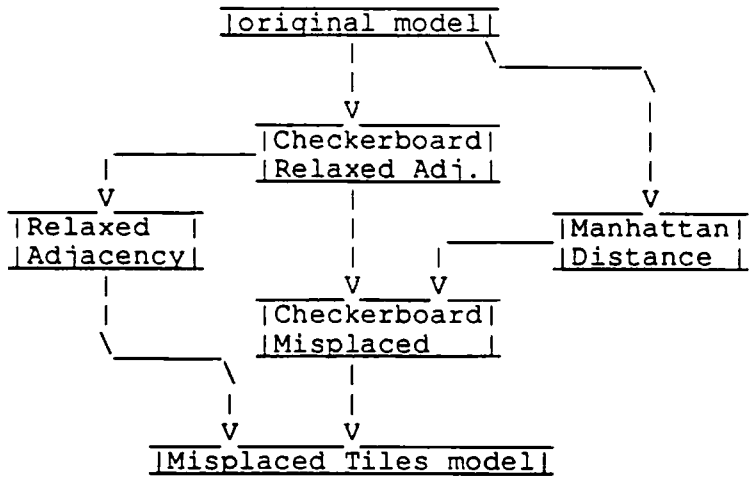

Figure 2-3: Partial order of the discussed N-Puzzle relaxation space

generated by relaxing different preconditions (e.g., Manhattan Distance and Relaxed Adjacency). Gaschnig investigated the Relaxed Adjacency heuristic and found that it evaluated only slightly higher than Misplaced Tiles in most, and lower than Manhattan Distance in all of the 875 puzzle instances which he studied [10]. We noticed, however, that there are exceptions which he overlooked: the Manhattan Distance estimate is lower in 0.2% of the possible Eight Puzzle instances (see Figure 2-4 for one such instance). Realizing that the evaluations made by these two heuristics are both guaranteed to be underestimates, one can employ the trivial improvement of using **MAX**(Manhattan Distance, Relaxed Adjacency) as an estimate (this is a general suggestion from Pearl [32], although neither he nor Gaschnig mention Relaxed Adjacency's occasional superiority).

	2	1
4	3	8
7	6	5

Figure 2-4: Relaxed Adjacency = 12
Manhattan Distance = 8
Optimal Solution = 24
(0 2 1 4 3 8 7 6 5)

3. Refining Relaxed Models by Solution-Criticism

We have seen how constraint relaxation 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., shortest path of a tile to its goal position (Manhattan Distance), the bipartite graph of tile positions (Checkerboard Relaxed Adjacency), 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 above. We will attempt to improve upon it by contrasting the solution plan which it suggests to that of the original problem.

3.1. Analyzing the Shortcomings of the Manhattan Distance Model

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) (see Figure 3-1). In other cases, the path is not unique (see Figure 3-2).

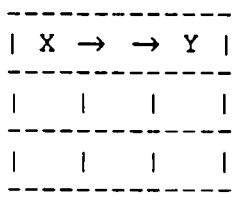

Figure 3-1: Unique, straight-line shortest path from X to Y

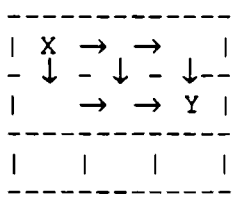

Figure 3-2: Non-unique shortest paths from X to Y

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 - hopefully it will be simple to analyze and improve the heuristic by recognizing when these suggested subgoal solutions are infeasible. First, we present the following results about paths:

Lemma 4: If there exists one path from position X to position Y in the N Puzzle that is of even (odd) length, then all paths from X to Y are of even (odd) length.

Proof: From the discussion in Section 2.5, one knows that the tile positions form a bipartite graph. Since, in a bipartite graph, all paths of length 1 move between the two sides of the graph, all paths between positions which are on opposite sides of the graph are of odd length, and all paths between positions which are in the same side are of even length. *Q.E.D.*

Consequently,

Corollary 5: If there is a unique shortest path, p, between position X and position Y in the N Puzzle, then any alternate path will be at least 2 moves longer than p.

One notices, when comparing the subgoal solutions to the actual optimal solutions, 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 Lemma 5). 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 a way to compensate for these oversimplifications. Before describing a precise method for doing so, we examine several examples of this phenomenon.

3.1.1. Examples of Conflicting Shortest Paths

Figures 3-3 through 3-6 illustrate four typical examples of conflicts between unique shortest paths.

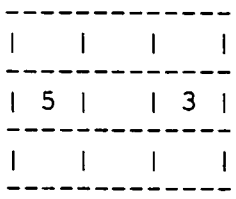

Figure 3-3: Shortest paths collide:
Add 2 to Manhattan Dist.

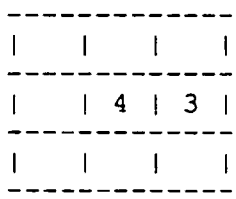

Figure 3-4: '4' acts as an obstacle, although its path length is 0:
Add 2 to Manhattan Dist.

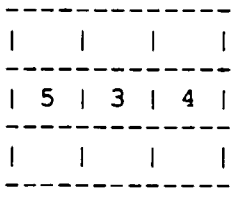

Figure 3-5: '5' must move off-line:
Add 2 to Manhattan Dist.

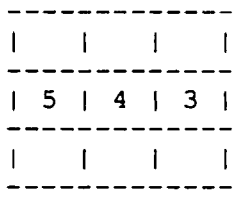

Figure 3-6: 2 tiles must move off-line:
Add 4 to Manhattan Dist.

In Figure 3-3, either the '5' or the '3' must move outside of the middle row to make room for the other to pass. Therefore, one should add two to the estimate of Manhattan Distance (one for the move out of the line and another for the move returning to the line).

Figure 3-4 shows a conflict in which a tile, which had previously been "solved", presents an obstacle to another tile. This conflict contradicts the Means-Ends analysis [28] intuition that solved subgoals will not be disturbed (similar failures occur in planning solutions to problems in the "Blocks World" [7, 46]). To resolve this conflict, either the '4' or the '3' will have to follow a non-shortest path, adding at least 2 moves to the Manhattan Distance estimate.

Figure 3-5 shows another typical conflict. In this state, the '5' tile is in conflict with the '3' and the '4'. Clearly, either the '5' has to move out of the way (2 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 (4 extra moves). To preserve admissibility, one must assume that the less costly resolution occurs, and add 2 to the Manhattan Distance estimate.

Lastly, Figure 3-6 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 4 to the Manhattan Distance estimate when this case is recognized.

If one can devise a method for tabulating the additional moves forced by conflicting subgoals, one can add that total to our Manhattan Distance estimate and create a new admissible heuristic for this problem: one which can easily be generalized to the N Puzzle. Intuitively, one examines the puzzle state, row by row and column by column, and adds to Manhattan Distance the minimum number of additional moves necessary to resolve the conflicts within each row and column: therefore, this 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 to those of Manhattan Distance, Relaxed Adjacency, and Misplaced Tiles, for the puzzle instances shown in Figures 3-7 to 3-10.

	2	1
7	4	5
6	3	8

Figure 3-7: Misplaced Tiles = 4
Relaxed Adjacency = 6
Manhattan Distance = 6
Linear Conflict = 8
Optimal Solution = 22
(0 2 1 7 4 5 6 3 8)

	2	1
5	4	3
6	7	8

Figure 3-8: Misplaced Tiles = 4
Relaxed Adjacency = 6
Manhattan Distance = 6
Linear Conflict = 12
Optimal Solution = 20
(0 2 1 5 4 3 6 7 8)

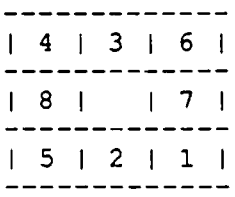

Figure 3-9: Misplaced Tiles = 8
Relaxed Adjacency = 10
Manhattan Distance = 22
Linear Conflict = 22
Optimal Solution = 26
(4 3 6 8 0 7 5 2 1)

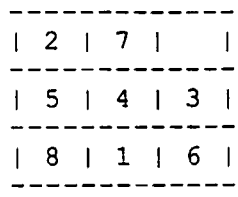

Figure 3-10: Misplaced Tiles = 7
Relaxed Adjacency = 10
Manhattan Distance = 14
Linear Conflict = 24
Optimal Solution = 26
(2 7 0 5 4 3 8 1 6)

While the Linear Conflict heuristic is clearly more informed than the Manhattan Distance heuristic, one notes that, because the relaxation space is a partial order, one cannot presuppose anything about the relative informedness of the Linear Conflict and Relaxed Adjacency heuristic. In fact, there are cases of the Fifteen Puzzle (e.g. (0 1 3 7 4 5 2 6 9 13 10 11 8 12 14 15)) for which the Relaxed Adjacency estimate is higher than the Linear Conflict heuristic.

We note that the idea of tile conflicts has been mentioned by others: Nilsson, for example, noted that Manhattan Distance "is too coarse, ... in that it does not accurately appraise the difficulty of exchanging the positions of two adjacent tiles" [31]. However, the heuristic he proposes to correct the situation, the Sequence Score (also investigated in [5, 36]), is inadmissible and is applicable only to special goal states of the Eight Puzzle. Furthermore, it is not generalizable to the N Puzzle.

3.1.2. An Iteration of the Process - an Even More Informed Heuristic

One may now consider the differences between the solutions proposed by the Linear Conflict model and those of the original problem. One can iterate the process of heuristic refinement by once again criticizing the differences between these proposed solution paths. We notice that the Linear Conflict model affords no consideration to the effects of diagonally adjacent correct tiles in the puzzle's corners. However, there are several simple cases where the solution length to the actual problem is clearly affected by this occurrence.

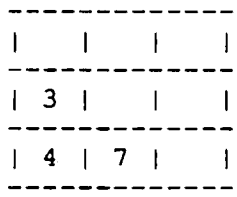

Figure 3-11: Corner tile is blocked:
Add 2 to Linear Conflict

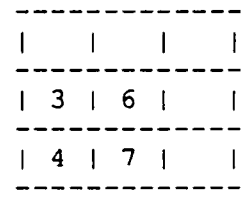

Figure 3-12: Two correct tiles prevent a swap:
Add 4 to Linear Conflict

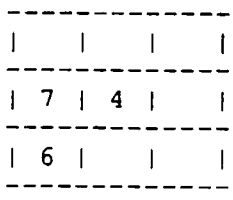

Figure 3-13: Corner tile blocks its neighbor: Add 2 to Linear Conflict

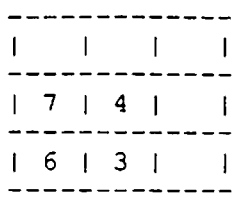

Figure 3-14: Corner tile blocks a swap: Add 4 to Linear Conflict

The Corner Conflicts are merely a special case of Diagonal Conflicts, which can be found throughout the puzzle. We could have chosen to admissibly add the effects of these Diagonal Conflicts to Manhattan Distance, however, one cannot admissibly add the effects of both Diagonal and Linear Conflicts (the movement of one tile might reduce both of these estimates, and so the heuristic estimate would not be monotone). The Corner Conflicts described above consider tiles which could not be involved in any Linear Conflicts, and therefore their contribution can be added to the Linear Conflict estimate. We note, however, that the relative contribution of the Corner Conflicts decreases as the size of the N Puzzle increases, and so, we will instead concentrate on the Linear Conflict Heuristic.

3.2. An Algorithm for Calculating the Linear Conflict Heuristic

The algorithm performs an analog of plan criticism upon the unique shortest paths which exist in any given line. We first define a *linear conflict*:

Definition 6: 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.

We now define some variables used by the algorithm:

s is the current state.

$C(t_j,r_i)$ is the number of tiles in row r_i with which t_j is in conflict. Similarly for $C(t_j,c_i)$

$lc(s,r_j)$ is the number of tiles that must be removed from row r_j in order to resolve the linear conflicts. Similarly, $lc(s,c_j)$ is the number of tiles that must be removed from column c_j in order to resolve the linear conflicts.

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

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

$LC(s)$ is the minimum number of additional moves necessary to resolve the linear conflicts in s.

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

Begin {Algorithm}

For each row r_i in the state s, one accounts for the conflicts local to that row $lc(s,r_i)$ as follows:

- $lc(s,r_i) = 0$.
- 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, do
  - Find t_k such that there is no $C(t_j,r_i)$ greater than $C(t_k,r_i)$. (As t_k is the tile with the most conflicts, 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 $lc(s,c_j)$. Then calculate the estimate of the Linear Conflict alone:

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

Determine, for each tile t_i in state s, its Manhattan Distance $md(s,t_i)$, and sum these to get the overall Manhattan Distance $MD(s) = md(s,t_1) + \ldots + md(s,t_N)$. Calculate the overall Linear Conflict heuristic estimate: $h'(s) = MD(s) + LC(s)$.

End {Algorithm}

We now prove that this algorithm calculates a lower bound on the minimum path length which needs to be added to the shortest paths in order to resolve conflicts in each line (other conflicts are still possible):

Theorem 7: The above algorithm calculates the minimum path length needed to resolve all linear conflicts.

Proof: First, we concentrate on one line and prove the following claim: for any line, the algorithm calculates the minimum number of tiles which must take non-shortest paths.

Notice that in the line, one may consider each tile as a node in a graph. Conflicts between tiles are arcs in the graph. In resolving conflicts, one wants to remove all arcs by removing a minimum number of nodes. Our algorithm follows a greedy strategy, removing a node with highest degree first, and continues recursively on the remaining graph. Therefore, it determines the minimal number of tiles which must be removed from the line. The algorithm counts 2 moves for each removal, which is the minimal number needed according to Corollary 5.

Since the algorithm calculates each line separately, what remains to be shown is that the number of conflicts in a line is independent of the conflicts in other lines. To do this, we demonstrate that removing a tile from one line (to resolve a conflict) will not affect conflicts in other lines. Consider a line in which one removes k tiles in order to resolve all conflicts. Among these tiles, one removes tile t_j. If t_j is not in its goal position, then it is only involved in conflicts in this line. If t_j is in its goal position, then moving it out of this line will have no effect on the conflicts which may exist in the perpendicular line, because it does not change position relative to the other tiles in that line (it merely moves into the blank position).

Therefore, the algorithm, which calculates the correct number of conflicts in each line independently (counting 2 moves for each removal), and sums these numbers, returns the minimum added path length to remove all linear conflicts. *Q.E.D.*

3.2.1. 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 $O(N^{1.5})$ 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 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 pre-computed 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 their goal positions are in their 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 per node (i.e., nodes examined per second) - this was more than made up for by the dramatic decrease in the number of nodes that needed to be examined when the Linear Conflict heuristic was used (cf. 3.3).

3.2.2. Proof of Monotonicity of the Linear Conflict Heuristic

Theorem 8: The Linear Conflict heuristic is monotone (and therefore admissible).

Proof: To establish monotonicity we must show that $\forall s, s'\ f(s') \geq f(s)$ (where s' is a successor of s). Recall that $f(s) = g(s) + h'(s)$, where $g(s') = g(s) + 1$ and $h'(s) = MD(s) + LC(s)$.

In the movement from a state to its successor, let us assume that tile x moves from row r_i to r_j, while remaining in column c_k. We now consider the effects of tile x's movement on $MD(s')$ and $LC(s')$, and consequently $h'(s)$:

1. The goal position of x is in neither r_i nor r_j.

   $md(s',x) = md(s,x) \pm 1$. LC does not change. Therefore, $h(s') = h(s) \pm 1$, and $f(s') = f(s) + 1 \pm 1 \geq f(s)$.

2. The goal position of x is in r_j.

   Since x moved into its goal row, $md(s',x) = md(s,x) - 1$. Because r_i is not x's goal row, it contributed nothing to $lc(s,r_i)$, and its absence has no effect: $lc(s',r_i) = lc(s,r_i)$. Because x is moving into its goal row, it may or may not contribute to the conflicts in that row, so either $lc(s',r_j) = lc(s,r_j)$ or $lc(s',r_j) = lc(s,r_j) + 2$. Therefore, $h(s') = h(s) \pm 1$, and $f(s') = f(s) + 1 \pm 1 \geq f(s)$.

3. The goal position of x is in r_i.

   Because x moved out of its goal row, $md(s',x) = md(s,x) + 1$. Because r_i is x's goal row, it may or may not contribute to the conflicts in that row, so either $lc(s',r_i) = lc(s,r_i)$ or $lc(s',r_i) = lc(s,r_i) - 2$. Because r_j is not x's goal row, its presence or absence contributes nothing to the conflicts there: $lc(s',r_j) = lc(s,r_j)$. Therefore, $h(s') = h(s) \pm 1$, and $f(s') = f(s) + 1 \pm 1 \geq f(s)$.

In all cases, $f(s') \geq f(s)$.

By the symmetry of the puzzle, one can construct a similar argument for movement within a row. *Q.E.D.*

3.3. Empirical Data and Analysis of the Linear Conflict Heuristic

We have developed a new heuristic for 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 problem). A study of its relative informedness and pruning power was conducted. Numerous search and analysis programs were implemented to measure the relative strengths of various heuristics on the Eight and Fifteen Puzzles. The search algorithm used was IDA* [20]. The relevant results are summarized in the tables in Appendix I.2.

To study the informedness of the different heuristic estimates, the entire state-space of the Eight Puzzle was evaluated using the Relaxed Adjacency, Manhattan Distance, and Linear Conflict heuristics. The heuristic evaluations are summarized in Tables 1 to 3. Table 1 shows the distributions of the three heuristic estimates, as well as the actual distribution of optimal solution lengths: in order to illustrate these distributions, the 50% of the states centered about the mean are marked with asterisks. Comparing the four distributions, one sees that the distribution of Linear Conflict estimates (mean at 15.11, with the middle 50% ranging from 14 to 17) most closely approximates the actual distribution of solution depths (mean at 21.97, with the middle 50% ranging from 19 to 24). The distribution of Manhattan Distance estimates has a mean at 14.00 (with the middle 50% ranging from 12 to 16), while the distribution of Relaxed Adjacency estimates has a mean at 8.05 (with the middle 50% ranging from 8 to 9). Table 2 shows how the three heuristics compare on individual puzzles: Relaxed Adjacency rarely estimates higher than Manhattan Distance on a given puzzle, but the Linear Conflict estimate is often much higher. Table 3 shows the best, worst, and average evaluations of the three heuristics for each depth of the search tree: once again, Linear Conflict's comparative power is evidenced. One notices that the Linear Conflict estimate increases as a function of Manhattan Distance: in other words, Linear Conflict's advantage over Manhattan Distance (i.e., the ratio of their estimates) grows faster as it is faced with harder problem instances.

Tables 4 to 6 summarize performance of the new heuristic compared with Manhattan Distance. The pruning power of the two heuristics was examined by solving 1000 randomly generated Eight puzzles and the 100 random Fifteen Puzzles used in the tests of [20], and comparing the number of states examined (this measure is proportional to search time: see 3.2.1). Table 4 shows the number of states examined for the 1000 Eight Puzzles, arranged by the depth of the solution found: overall, the number of states examined using Linear Conflicts is *less than half* of the number of states examined using Manhattan Distance. The table also suggests that the comparative pruning power of Linear Conflict increases with problem difficulty: this is expected from the discussion of Table 3 above. Moving onto the Fifteen Puzzle, Table 5 (an extension of a similar table in [20]), 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 perform more than 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. The 100 Fifteen Puzzles are sorted by problem difficulty (in number of states examined) into quintiles in Table 6: one sees that the Linear Conflicts heuristic demonstrates comparatively more pruning power on more difficult Fifteen Puzzle problem instances (just as it did for the Eight Puzzle).

One sees that the comparative power of the Linear Conflict heuristic increases with the difficulty of the problem instance. Furthermore, when moving from the Eight to the Fifteen Puzzle, the advantage of Linear Conflict increases (because there is a greater chance for conflicts to occur as the size of the puzzle increases).

4. Another Example - the Traveling-Salesman Problem

The Traveling-Salesman Problem [9, 34] is one of the most famous NP-Hard problems. This intractable problem is considered a "real-world" problem, and the development of heuristics for its solution has occupied many researchers.

4.1. Pearl's analysis of the Traveling-Salesman Problem

Pearl [32] 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. He suggests that we may consider a successful Traveling-Salesman tour as satisfying three constraints:

(1) being a graph (2) being connected (3) being of degree 2

Pearl outlines informally how we can develop simplified problems, and thence, admissible heuristics by deleting one or more of these constraints on the goal description. If we delete (2) we arrive at the "optimal assignment" heuristic, and if we delete (3) we arrive at the "minimum spanning tree" heuristic. Pohl [33, 34] describes some heuristics obtained by deleting (1).

However, to be consistent with our previous example and the state-space model that we are examining, we will formulate the problem differently, using a well-defined set of states and operators. We then delete constraints upon the operators to get relaxed models of the problem, find solutions to these relaxed models, and then analyze and criticize these solutions.

4.2. Formalization of the Problem

We will 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 $city_y$ to $city_x$, we want to estimate the length of a minimum partial tour from $city_x$ back to $city_y$ which visits all the remaining cities. We may simply forget

all of the cities we have already visited in the partial tour, and delete them (and the edges incident with them) from the graph.

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

MOVE($city_i$, $city_j$) :

precondition list	:	ON(salesman,$city_i$), NOT(VISITED($city_j$))
add list	:	ON(salesman,$city_j$), 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 (a symmetric tour is given by switching $city_i$ and $city_j$ in every MOVE). Notice that the movement of the salesman resembles the movement of the blank in the N Puzzle, except that in this problem, all positions are adjacent, the cost of the operator is variable, and (of course) the goal is entirely different. As far as we know, this is the first formulation of this problem in a state-space model using STRIPS-like predicates.

4.3. Relaxed Models and Solution-Criticism

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

If we delete the NOT(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. We note, however, that the cost of a minimum spanning tree is a lower bound on the cost of such a relaxed tour.

If we delete the ON(salesman, $city_i$) requirement, we allow the salesman to jump (for free) to $city_i$, and then move to $city_j$. Because of the add list, 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 the relaxed model is always a subgraph consisting of connected components, where each component of m nodes is a modified tree of m edges, which either has a cycle containing equal (shortest) length edges, or a duplicated shortest edge which creates a degenerate cycle. Such a solution can be calculated in polynomial time. 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 N increases, however, this becomes less likely.

Applying our method of solution-criticism, we see that the solutions to this relaxed model contain cycles and may be unconnected. 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). We replace *k* of the edges by *k* different edges (the shortest edge will be a member of both sets). The total length of the new edges will not be less than the total length of the removed ones. Note that the resulting graph is a minimum spanning tree with an added shortest edge. This estimate is computable in low-order polynomial time [43].

A further criticism may be added. The resulting graph is not connected to $city_x$ (because it had been visited already in the partial tour from $city_y$ to $city_x$: $city_y$ 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 $city_x$ to the minimum spanning tree. The resulting graph is merely a minimum spanning tree which includes $city_x$, except in the case where $city_x$ equals $city_y$ (the initial state of the search), in which we get Held and Karp's minimum-weight 1-tree instead [14, 15]. Interestingly, this heuristic, which we have derived by solution-criticism is one of the most famous admissible heuristics for the Traveling-Salesman problem [14, 15]. Its performance has been thoroughly studied in [15].

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. This problem can be reduced to the minimum weighted-matching problem, and solved in polynomial time.

5. Discussion of the Solution-Criticism Method

The process of relaxation makes a problem easy by allowing 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, 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 increase the proximity of this relaxed model by accounting for those constraints which it overlooks. 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 unconnected solution, we arrive at the celebrated minimum spanning tree heuristic.

The method of solution-criticism is a first attempt at restoring the global view to these myopic relaxed solutions, by comparing them to 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 them into a more feasible solution, or simply by adding some measure of what

they have overlooked. This may be considered as analogous to a planning process, in particular, the problem of planning in a hierarchy of abstraction spaces [38].

We note, that for more than two decades, the Eight Puzzle has been used as a research example to demonstrate the development of heuristics, and yet, the Manhattan Distance heuristic has not been improved upon in all these years. We believe that our improvement, the Linear Conflict heuristic, is difficult to find directly without following the methodology suggested here.

Pearl's method of constraint-deletion seems suitable for automation. Constraint-deletion is a straightforward procedure, given a representation of the problem. However, there is one major difficulty: choosing, from among the vast array of problem representations, one which yields useful relaxed models for which we can find algorithms. Therefore, a robust problem-solving system must be capable of changing the representation of a problem [18].

Adding our refinement will add another aspect to the automation, because discovering differences between the solutions to the relaxed model and the original problem demands the addition of learning components to the system. The system will have to learn, by examining problem instances, which characteristics of the relaxed model's solution are not found in the actual solution to the problem. Notice that this approach to learning heuristics is related to learning in the planning domain [38, 41], as opposed to approaches which attempt to learn heuristics by statistical analysis [37, 39].

6. Conclusion

The process of criticizing the solutions to relaxed models is suggested as a valuable addition to the constraint-deletion method. The preceding analysis and empirical data show that one can develop a very powerful heuristic by attempting to understand the infeasibility of a proposed relaxed solution, and recovering some of the information that was lost in the relaxation.

We have demonstrated how such criticism has been used to derive powerful admissible heuristics for the N Puzzle and the Traveling-Salesman Problem.

The process of generating admissible heuristics suggested here is analogous to the procedures used in developing lower bounds for problems. First, one simplifies the problem so that its solution is a lower-bound and is easily attainable, and then one attempts to tighten that bound by reconsidering factors ignored by the simplification. We believe that the process of relaxation and subsequent tightening captures one of the methods used by humans in coping with hard problems.

Acknowledgements

Aside from teaching us all we know about heuristic search, Rich Korf suggested the idea of tile conflicts. We thank him for many helpful comments and much support. We would also like to thank Bruce Abramson and Jens Christensen for many fruitful discussions, and Bill Schilit for his generous and friendly help in implementing some of the search programs.

I. Appendices

I.1. Proof of Upper Bound for Relaxed Adjacency Heuristic

Theorem 9: For any given N Puzzle, P, Relaxed-Adjacency(P) $\leq \frac{3}{2}$ Misplaced-Tiles(P).

Proof: One may represent the abstract solution plan to any N Puzzle as the permutation necessary to transform the given puzzle into the goal state. This permutation may be represented as:

$$\begin{pmatrix} t_0 & t_1 & \cdots & t_N \\ 0 & 1 & \cdots & N \end{pmatrix}$$

Alternately, this permutation can be given as a sequence of cycles (cyclic permutations):

$$(t_{0,1} \cdots t_{0,k_0}) \cdots (t_{j,1} \cdots t_{j,k_j}) \text{ where } \sum_{i=0}^{j} k_i = N+1 \text{ and } k_i \text{ is the length of the } i^{th} \text{ cycle.}$$

For example, the abstract solution to the puzzle shown in Figure 3-9 is

(1 3 8)(2 6 5 7)(4 0)

In other words, tile 1 moves to tile 3's position, tile 3 moves to tile 8's position, tile 8 moves to tile 1's position, tile 2 moves to tile 6's position, etc. Solving the entire permutation by solving each cycle individually is the optimal strategy for solving a puzzle in the Relaxed Adjacency model.

Of course, the individual operators in the Relaxed Adjacency model are not cycles, but permutations of two tiles (i.e., transpositions) of the form (0 t), where t is the tile to be swapped with the blank (call these operations "blank-swaps"). The problem of finding an optimal solution, then, is to show how the sequence of cycles which represents an abstract N Puzzle solution may be transformed into an optimal (i.e., shortest possible) sequence of blank-swaps. For example, the puzzle shown in Figure 3-9 may be solved by

(0 1)(0 3)(0 8)(0 1)(0 2)(0 6)(0 5)(0 7)(0 2)(0 4)

This sequence of blank-swaps corresponds directly to the sequence of moves in the optimal solution.

We present the following facts about such blank-swaps: they are elementary results of permutation group theory.

Fact A: A cycle of the form $(t_1 \cdots t_k)$, where $\forall t_i, 1 \leq i \leq k \quad (t_i \neq 0)$, may be optimally represented as a sequence (of length $k+1$) of blank-swaps as follows:

$$(0\ \ t_k)(0\ \ t_{k-1}) \cdots (0\ \ t_2)(0\ \ t_1)(0\ \ t_k)$$

Note that this can be thought of as an initial move which brings the blank into the cycle, creating a new cycle, and then a solution of the new cycle as in Fact B.

Fact B: A cycle of the form $(t_1 \cdots t_k)$, where $\exists t_i, 1 \leq i \leq k \quad (t_i = 0)$, may be optimally represented as a sequence (of length $k-1$) of blank-swaps as follows:

$$(0\ \ t_{i-1}) \cdots (0\ \ t_1)(0\ \ t_k) \cdots (0\ \ t_{i+1})$$

Fact C: A singleton cycle of the form (t) can be represented as a null sequence of blank-swaps.

Obviously, in any N Puzzle, one can only have one cycle which contains the 0 (there is only

one blank, after all). Call the length of that cyclic permutation LEN_b. To solve this cycle will require $LEN_b - 1$ blank-swaps.

One may have up to $\frac{N}{2}$ cycles of length > 1 which do not contain the blank. Call the number of such cycles NUM_{nb}, and the sum of their lengths LEN_{nb}. To solve such a cycle of length l will require $l+1$ blank-swaps. So, to solve all of them will require $LEN_{nb}+NUM_{nb}$ blank-swaps.

One may have up to N cycles of length = 1 excluding the cycle (0). These cycles represent single non-blank tiles which are in their goal positions. Call the number of such singleton cycles NUM_s. These cycles are already solved.

We will note that $(LEN_b - 1)+LEN_{nb}$ is equal to the number of misplaced tiles in the puzzle, MT.

Clearly, $LEN_{nb} + (LEN_b - 1) + NUM_s = N$ and by the above discussion, the collective length of the optimal blank-swap sequences for all the cyclic permutations is $(LEN_b - 1)+(LEN_{nb}+NUM_{nb})$, or $MT+NUM_{nb}$. Since there are at least two misplaced tiles in each non-singleton cycle which does not contain the blank, we can see that NUM_{nb} can never exceed $\frac{1}{2}MT$, and so $MT+NUM_{nb} \leq \frac{3}{2}MT$.

From the discussion in the above proof, the following is evident:

Corollary 10: Gaschnig's algorithm for calculating the Relaxed Adjacency heuristic divides the puzzle into these cyclic permutations and solves them optimally, by applying the following step until the puzzle is solved:

```
If the blank is in its own goal,
  then swap with any misplaced tile
  else swap with the tile that
            belongs in the blank's position
```

It is easily seen that the iteration of this step will produce a solution in which the l-length cycle which contains the blank is solved first (if such a cycle exists), in $l-1$ moves (cf. Fact A, above). Then cycles which do not contain the blank are solved, one at a time. For a cycle of length k, this is done by first moving any misplaced tile T into the blank's position, creating a new cycle which contains the blank (the new cycle has length $k+1$) - this is solved in $(k+1 - 1)$ moves. Together with the first move of T, such a cycle takes $k+1$ moves to solve (cf. Fact B), and leaves the blank in its goal position.

Thus, all cycles are seen to be solved optimally (cf. Facts A and B above) by Gaschnig's algorithm.

I.2. Tables

Table 1

Heuristic Estimates of the Entire State-Space of the Eight Puzzle arranged by Estimated Depth

ESTIMATE	RA	MD	LC	ACTUAL DEPTH
0	1	1	1	1
1	4	2	2	2
2	28	4	4	4
3	168	10	10	8
4	1036	115	53	16
5	4060	246	94	20
6	15274	695	237	39
7	33516	1134	494	62
8	* 61732	3655	1656	116
9	* 41632	5084	2344	152
10	22624	10999	5383	286
11	1260	11862	6620	396
12	105	* 21707	15178	748
13	0	* 20040	15662	1024
14	0	* 27625	* 26072	1893
15	0	* 20954	* 23150	2512
16	0	* 22180	* 27996	4485
17	0	14226	* 19946	5638
18	0	10825	17463	9529
19	0	5896	9752	* 10878
20	0	2790	5708	* 16993
21	0	1186	2274	* 17110
22	0	204	941	* 23952
23	0	0	280	* 20224
24	0	0	103	* 24047
25	0	0	10	15578
26	0	0	4	14560
27	0	0	2	6274
28	0	0	1	3910
29	0	0	0	760
30	0	0	0	221
31	0	0	0	2

RA = Relaxed Adjacency MD = Manhattan Distance LC = Linear Conflict

asterisks mark the 50% of the states centered around the mean

Table 2

Number of Deviations of Relaxed Adjacency and Linear Conflict Estimates from Manhattan Distance Estimate

ESTIMATE	RA	MD	LC
MD - 16	0	0	0
MD - 14	130	0	0
MD - 12	3058	0	0
MD - 10	16525	0	0
MD - 8	41349	0	0
MD - 6	56661	0	0
MD - 4	42674	0	0
MD - 2	17194	0	0
MD	3545	181440	101400
MD + 2	300	0	61260
MD + 4	4	0	16282
MD + 6	0	0	2304
MD + 8	0	0	186
MD + 10	0	0	8
MD + 12	0	0	0

RA = Relaxed Adjacency MD = Manhattan Distance LC = Linear Conflict

Table 3

Minimum, Average, and Maximum Values of each Heuristic Estimate for the Entire Eight Puzzle - arranged by Actual Depth

Depth	Minimum			Average			Maximum		
	RA	MD	LC	RA	MD	LC	RA	MD	LC
0	0	0	0	0.0	0.0	0.0	0	0	0
1	1	1	1	1.0	1.0	1.0	1	1	1
2	2	2	2	2.0	2.0	2.0	2	2	2
3	3	3	3	3.0	3.0	3.0	3	3	3
4	4	4	4	4.0	4.0	4.0	4	4	4
5	3	5	5	4.80	5.0	5.0	5	5	5
6	4	4	4	5.43	5.79	5.79	6	6	6
7	5	5	5	5.90	6.61	6.61	7	7	7
8	4	4	4	6.51	7.37	7.37	8	8	8
9	3	5	5	6.60	8.23	8.31	9	9	9
10	2	4	4	6.86	8.72	8.86	8	10	10
11	1	3	3	6.75	9.30	9.50	9	11	11
12	2	4	4	6.99	9.66	9.97	10	12	12
13	3	5	5	6.95	10.09	10.62	9	13	13
14	2	4	4	7.15	10.35	10.98	10	14	14
15	3	5	5	7.18	10.85	11.49	9	15	15
16	2	4	4	7.45	11.13	11.84	10	16	16
17	3	5	5	7.51	11.68	12.51	11	17	17
18	2	4	6	7.72	11.96	12.85	10	18	18
19	3	5	5	7.76	12.62	13.55	11	19	19
20	2	4	6	7.95	12.88	13.87	10	20	20
21	3	5	7	7.94	13.60	14.67	11	21	21
22	4	4	8	8.11	13.84	14.97	10	22	22
23	3	7	7	8.11	14.65	15.82	11	21	23
24	4	6	8	8.31	14.83	16.09	12	22	24
25	3	9	9	8.25	15.72	16.97	11	21	25
26	4	8	8	8.46	15.82	17.21	12	22	26
27	5	9	11	8.41	16.76	18.00	11	21	27
28	4	10	12	8.61	16.66	18.35	12	22	28
29	5	11	13	8.42	17.50	19.43	11	21	25
30	6	12	12	8.79	16.72	19.45	12	22	24
31	9	21	23	9.00	21.00	23.00	9	21	23
AVG				8.05	14.00	15.11			

Avg actual depth: 21.97

RA = Relaxed Adjacency MD = Manhattan Distance LC = Linear Conflict

Table 4

Comparative Performance: IDA* Search on 1000 Random Eight Puzzle Instances

Average number of states examined for each of the three heuristics - arranged by Depth of Optimal Solution

NUMBER OF PUZZLES	DEPTH	RA	MD	LC
0	0	0.0	0.0	0.0
0	1	0.0	0.0	0.0
0	2	0.0	0.0	0.0
0	3	0.0	0.0	0.0
0	4	0.0	0.0	0.0
0	5	0.0	0.0	0.0
0	6	0.0	0.0	0.0
0	7	0.0	0.0	0.0
0	8	0.0	0.0	0.0
2	9	21.0	15.0	15.0
2	10	77.0	47.0	40.50
0	11	0.0	0.0	0.0
2	12	196.0	58.0	55.50
6	13	232.66	54.33	41.33
11	14	445.0	106.72	66.90
17	15	796.76	173.0	123.70
28	16	1272.31	292.71	178.32
37	17	2045.59	335.89	205.43
45	18	2901.57	493.13	284.15
73	19	5108.36	654.12	357.87
92	20	7831.1	902.76	474.68
90	21	13751.71	1497.24	785.7
121	22	20581.09	1804.64	889.15
119	23	36971.29	2888.22	1382.27
113	24	56904.56	3543.43	1675.99
87	25	102872.56	5653.60	2699.85
96	26	159644.41	7260.80	3434.69
33	27	287195.12	12827.24	5597.21
21	28	352690.91	14815.38	6977.90
5	29	603842.8	20146.60	8917.19
0	30	0.0	0.0	0.0
0	31	0.0	0.0	0.0
TOTAL AVG			3299.53	1571.5

RA = Relaxed Adjacency MD = Manhattan Distance LC = Linear Conflict

Table 5

Comparative Performance: IDA* Search on 100 Random Fifteen Puzzle Instances (cf. [20]) page 1 (1 - 50)

NO	INITIAL STATE	MD INIT	LC INIT	LEN	MD STATES	LC STATES	PCT
01	14 13 15 7 11 12 9 5 6 0 2 1 4 8 10 3	41	43	57	276,361,933	12,205,623	4.4
02	13 5 4 10 9 12 8 14 2 3 7 1 0 15 11 6	43	43	55	15,300,442	4,556,067	29.8
03	14 7 8 2 13 11 10 4 9 12 5 0 3 6 1 15	41	41	59	565,994,203	156,590,306	27.6
04	5 12 10 7 15 11 14 0 8 2 1 13 3 4 9 6	42	42	56	62,643,179	9,052,179	14.5
05	4 7 14 13 10 3 9 12 11 5 6 15 1 2 8 0	42	44	56	11,020,325	2,677,666	24.5
06	14 7 1 9 12 3 6 15 8 11 2 5 10 0 4 13	36	40	52	32,201,660	4,151,682	12.9
07	2 11 15 5 13 4 6 7 12 8 10 1 9 3 14 0	30	30	52	387,138,094	97,264,710	25.1
08	12 11 15 3 8 0 4 2 6 13 9 5 14 1 10 7	32	36	50	39,118,937	3,769,804	9.6
09	3 14 9 11 5 4 8 2 13 12 6 7 10 1 15 0	32	36	46	1,650,696	88,588	5.4
10	13 11 8 9 0 15 7 10 4 3 6 14 5 12 2 1	43	45	59	198,758,703	48,531,591	24.4
11	5 9 13 14 6 3 7 12 10 8 4 0 15 2 11 1	43	45	57	150,346,072	25,537,948	17.0
12	14 1 9 6 4 8 12 5 7 2 3 0 10 11 13 15	35	35	45	546,344	179,628	32.9
13	3 6 5 2 10 0 15 14 1 4 13 12 9 8 11 7	36	38	46	11,861,705	1,051,213	8.9
14	7 6 8 1 11 5 14 10 3 4 9 13 15 2 0 12	41	43	59	1,369,596,778	53,050,799	3.9
15	13 11 4 12 1 8 9 15 6 5 14 2 7 3 10 0	44	46	62	543,598,067	130,071,656	24.0
16	1 3 2 5 10 9 15 6 8 14 13 11 12 4 7 0	24	26	44	17,984,051	2,421,878	13.5
17	15 14 0 4 11 1 6 13 7 5 8 9 3 2 10 12	46	46	66	609,399,560	100,843,886	16.6
18	6 0 14 12 1 15 9 10 11 4 7 2 8 3 5 13	43	43	55	23,711,067	5,224,645	22.0
19	7 11 8 3 14 0 6 15 1 4 13 9 5 12 2 10	36	38	46	1,280,495	385,369	30.1
20	6 12 11 3 13 7 9 15 2 14 8 10 4 1 5 0	36	36	52	17,954,870	3,642,638	20.2
21	12 8 14 6 11 4 7 0 5 1 10 15 3 13 9 2	34	36	54	257,064,810	43,980,448	17.1
22	14 3 9 1 15 8 4 5 11 7 10 13 0 2 12 6	41	45	59	750,746,755	79,549,136	10.6
23	10 9 3 11 0 13 2 14 5 6 4 7 8 15 1 12	33	37	49	15,971,319	770,088	4.8
24	7 3 14 13 4 1 10 8 5 12 9 11 2 15 6 0	34	38	54	42,693,209	15,062,608	35.2
25	11 4 2 7 1 0 10 15 6 9 14 8 3 13 5 12	32	36	52	100,734,844	13,453,743	13.4
26	5 7 3 12 15 13 14 8 0 10 9 6 1 4 2 11	40	44	58	226,668,645	50,000,803	22.1
27	14 1 8 15 2 6 0 3 9 12 10 13 4 7 5 11	33	35	53	306,123,421	31,152,542	10.2
28	13 14 6 12 4 5 1 0 9 3 10 2 15 11 8 7	36	36	52	5,934,442	1,584,197	26.7
29	9 8 0 2 15 1 4 14 3 10 7 5 11 13 6 12	38	40	54	117,076,111	10,085,238	8.6
30	12 15 2 6 1 14 4 8 5 3 7 0 10 13 9 11	35	35	47	2,196,593	680,254	31.0
31	12 8 15 13 1 0 5 4 6 3 2 11 9 7 14 10	38	40	50	2,351,811	538,886	22.9
32	14 10 9 4 13 6 5 8 2 12 7 0 1 3 11 15	43	45	59	661,041,936	183,341,087	27.7
33	14 3 5 15 11 6 13 9 0 10 2 12 4 1 7 8	42	42	60	480,637,867	28,644,837	6.0
34	6 11 7 8 13 2 5 4 1 10 3 9 14 0 12 15	36	42	52	20,671,552	1,174,414	5.7
35	1 6 12 14 3 2 15 8 4 5 13 9 0 7 11 10	39	39	55	47,506,056	9,214,047	19.4
36	12 6 0 4 7 3 15 1 13 9 8 11 2 14 5 10	36	38	52	59,802,602	4,657,636	7.8
37	8 1 7 12 11 0 10 5 9 15 6 13 14 2 3 4	40	42	58	280,078,791	21,274,607	7.6
38	7 15 8 2 13 6 3 12 11 0 4 10 9 5 1 14	41	43	53	24,492,852	4,946,981	20.1
39	9 0 4 10 1 14 15 3 12 6 5 7 11 13 8 2	35	35	49	19,355,806	3,911,623	20.2
40	11 5 1 14 4 12 10 0 2 7 13 3 9 15 6 8	36	38	54	63,276,188	13,107,557	20.7
41	8 13 10 9 11 3 15 6 0 1 2 14 12 5 4 7	36	40	54	51,501,544	12,388,516	24.1
42	4 5 7 2 9 14 12 13 0 3 6 11 8 1 15 10	30	32	42	877,823	217,288	24.8
43	11 15 14 13 1 9 10 4 3 6 2 12 7 5 8 0	48	54	64	41,124,767	7,034,879	17.1
44	12 9 0 6 8 3 5 14 2 4 11 7 10 1 15 13	32	38	50	95,733,125	3,819,541	4.0
45	3 14 9 7 12 15 0 4 1 8 5 6 11 10 2 13	39	39	51	6,158,733	764,473	12.4
46	8 4 6 1 14 12 2 15 13 10 9 5 3 7 0 11	35	41	49	22,119,320	1,510,387	6.8
47	6 10 1 14 15 8 3 5 13 0 2 7 4 9 11 12	35	35	47	1,411,294	221,531	15.7
48	8 11 4 6 7 3 10 9 2 12 15 13 0 1 5 14	39	39	49	1,905,023	255,047	13.4
49	10 0 2 4 5 1 6 12 11 13 9 7 15 3 14 8	33	37	59	1,809,933,698	203,873,877	11.3
50	12 5 13 11 2 10 0 9 7 8 4 3 14 6 15 1	39	41	53	63,036,422	6,225,180	9.9

LEGEND

GOAL STATE

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

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 * ((LC STATES) / (MD STATES))

Comparative Performance:
IDA* Search on 100 Random Fifteen Puzzle Instances
page 2 (51 - 100)

NO	INITIAL STATE	MD INIT	LC INIT	LEN	MD STATES	LC STATES	PCT
51	10 2 8 4 15 0 1 14 11 13 3 6 9 7 5 12	44	44	56	26,622,863	4,683,054	17.6
52	10 8 0 12 3 7 6 2 1 14 4 11 15 13 9 5	38	42	56	377,141,881	33,691,153	08.9
53	14 9 12 13 15 4 8 10 0 2 1 7 3 11 5 6	50	50	64	465,225,698	125,641,730	27.0
54	12 11 0 8 10 2 13 15 5 4 7 3 6 9 14 1	40	42	56	220,374,385	26,080,659	11.8
55	13 8 14 3 9 1 0 7 15 5 4 10 12 2 6 11	29	31	41	927,212	163,077	17.6
56	3 15 2 5 11 6 4 7 12 9 1 0 13 14 10 8	29	33	55	1,199,487,996	166,183,825	13.9
57	5 11 6 9 4 13 12 0 8 2 15 10 1 7 3 14	36	36	50	8,841,527	3,977,809	45.0
58	5 0 15 8 4 6 1 14 10 11 3 9 7 12 2 13	37	39	51	12,955,404	3,563,941	27.5
59	15 14 6 7 10 1 0 11 12 8 4 9 2 5 13 3	35	37	57	1,207,520,464	90,973,287	07.5
60	11 14 13 1 2 3 12 4 15 7 9 5 10 6 8 0	48	48	66	3,337,690,331	256,537,528	07.7
61	6 13 3 2 11 9 5 10 1 7 12 14 8 4 0 15	31	35	45	7,096,850	672,959	09.5
62	4 6 12 0 14 2 9 13 11 8 3 15 7 10 1 5	43	45	57	23,540,413	8,463,998	36.0
63	8 10 9 11 14 1 7 15 13 4 0 12 6 2 5 3	40	42	56	995,472,712	20,999,336	02.1
64	5 2 14 0 7 8 6 3 11 12 13 15 4 10 9 1	31	35	51	260,054,152	43,522,756	16.7
65	7 8 3 2 10 12 4 6 11 13 5 15 0 1 9 14	31	35	47	18,997,681	2,444,273	12.9
66	11 6 14 12 3 5 1 15 8 0 10 13 9 7 4 2	41	43	61	1,957,191,378	394,246,898	20.1
67	7 1 2 4 8 3 6 11 10 15 0 5 14 12 13 9	28	30	50	252,783,878	47,499,462	18.8
68	7 3 1 13 12 10 5 2 8 0 6 11 14 15 4 9	31	35	51	64,367,799	6,959,507	10.8
69	6 0 5 15 1 14 4 9 2 13 8 10 11 12 7 3	37	39	53	109,562,359	5,186,587	04.7
70	15 1 3 12 4 0 6 5 2 8 14 9 13 10 7 11	30	32	52	151,042,571	40,161,673	26.6
71	5 7 0 11 12 1 9 10 15 6 2 3 8 4 13 14	30	34	44	8,885,972	539,387	06.1
72	12 15 11 10 4 5 14 0 13 7 1 2 9 8 3 6	38	40	56	1,031,641,140	55,514,360	05.4
73	6 14 10 5 15 8 7 1 3 4 2 0 12 9 11 13	37	39	49	3,222,276	1,130,807	35.1
74	14 13 4 11 15 8 6 9 0 7 3 1 2 10 12 5	46	46	56	1,897,728	310,312	16.4
75	14 4 0 10 6 5 1 3 9 2 13 15 12 7 8 11	30	34	48	42,772,589	5,796,660	13.6
76	15 10 8 3 0 6 9 5 1 14 13 11 7 2 12 4	41	43	57	126,638,417	25,481,596	20.1
77	0 13 2 4 12 14 6 9 15 1 10 3 11 5 8 7	34	36	54	18,918,269	5,479,397	29.0
78	3 14 13 6 4 15 8 9 5 12 10 0 2 7 1 11	41	41	53	10,907,150	2,722,095	25.0
79	0 1 9 7 11 13 5 3 14 12 4 2 8 6 10 15	28	30	42	540,860	107,088	19.8
80	11 0 15 8 13 12 3 5 10 1 4 6 14 9 7 2	43	43	57	132,945,856	39,801,475	29.9
81	13 0 9 12 11 6 3 5 15 8 1 10 4 14 2 7	39	41	53	9,982,569	1,088,123	10.9
82	14 10 2 1 13 9 8 11 7 3 6 12 15 5 4 0	40	44	62	5,506,801,123	203,606,265	03.7
83	12 3 9 1 4 5 10 2 6 11 15 0 14 7 13 8	31	37	49	65,533,432	2,155,880	03.3
84	15 8 10 7 0 12 14 1 5 9 6 3 13 11 4 2	37	41	55	106,074,303	17,323,672	16.3
85	4 7 13 10 1 2 9 6 12 8 14 5 3 0 11 15	32	32	44	2,725,456	933,953	34.3
86	6 0 5 10 11 12 9 2 1 7 4 3 14 8 13 15	35	37	45	2,304,426	237,466	10.3
87	9 5 11 10 13 0 2 1 8 6 14 12 4 7 3 15	34	36	52	64,926,494	7,928,514	12.2
88	15 2 12 11 14 13 9 5 1 3 8 7 0 10 6 4	43	45	65	6,009,130,748	422,768,851	07.0
89	11 1 7 4 10 13 3 8 9 14 0 15 6 5 2 12	36	40	54	166,571,097	29,171,607	17.5
90	5 4 7 1 11 12 14 15 10 13 8 6 2 0 9 3	36	40	50	7,171,137	649,591	09.1
91	9 7 5 2 14 15 12 10 11 3 6 1 8 13 0 4	41	41	57	602,886,858	91,220,187	15.1
92	3 2 7 9 0 15 12 4 6 11 5 14 8 13 10 1	37	39	57	1,101,072,541	68,307,452	06.2
93	13 9 14 6 12 8 1 2 3 4 0 7 5 10 11 15	34	34	46	1,599,909	350,208	21.9
94	5 7 11 8 0 14 9 13 10 12 3 15 6 1 4 2	45	45	53	1,337,340	390,368	29.2
95	4 3 6 13 7 15 9 0 10 5 8 11 2 12 1 14	34	36	50	7,115,967	1,517,920	21.3
96	1 7 15 14 2 6 4 9 12 11 13 3 0 8 5 10	35	37	49	12,808,564	1,157,734	09.0
97	9 14 5 7 8 15 1 2 10 4 13 6 12 0 11 3	32	34	44	1,002,927	166,566	16.6
98	0 11 3 12 5 2 1 9 8 10 14 15 7 4 13 6	34	34	54	183,526,883	41,564,669	22.6
99	7 15 4 0 10 9 2 5 12 11 13 6 1 3 14 8	39	39	57	83,477,694	18,038,550	21.6
100	11 4 0 8 6 10 5 13 12 7 14 3 1 2 9 15	38	40	54	67,880,056	17,778,222	26.2
				TOTAL	2.998476E10	3.759631E9	12.5

LEGEND

GOAL STATE

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

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 * ((LC STATES) / (MD STATES))

Table 6

Comparative Performance: The 100 Random Fifteen Puzzle Instances Sorted into Quintiles by Number of States Examined

QUINTILES	STATE NUMBERS	STATES MD	STATES LC	PCT.
1st	79 12 42 55 97 19 94 47 93 09 74 48 30 86 31 85 73 28 45 61	$4.7 * 10^7$	$9.4 * 10^6$	20.01
2nd	95 90 57 71 81 78 05 13 96 58 02 23 20 16 77 65 39 34 46 62	$2.9 * 10^8$	$5.3 * 10^7$	18.21
3rd	18 38 51 06 08 43 24 75 35 41 36 04 50 40 68 87 83 100 99 44	$1.1 * 10^9$	$1.6 * 10^8$	15.21
4th	25 84 69 29 76 80 11 70 89 98 10 54 26 67 21 64 01 37 27 52	$3.9 * 10^9$	$6.0 * 10^8$	15.11
5th	07 53 33 15 03 91 17 32 22 63 72 92 56 59 14 49 66 60 82 88	$3.1 * 10^{10}$	$2.9 * 10^9$	9.61

LEGEND

MD STATES	Total number of states examined using Manhattan Distance
LC STATES	Total number of states examined using Linear Conflict
PCT	100 * ((LC STATES) / (MD STATES))

References

1. Bellmore, M. and Nemhauser, G.L. "The Traveling Salesman Problem: A Survey". *Operations Research 16* (1968), 538-558.

2. Croes, G.A. "A Method for Solving Traveling-Salesman Problems". *Operations Research 6* (1958), 791-812.

3. Dantzig, G., Fulkerson, R., and Johnson, S. "Solution of a Large-Scale Traveling-Salesman Problem". *Operations Research 2* (1954), 393-410.

4. Dechter, Rina and Pearl, Judea. The Anatomy of Easy Problems: A Constraint-Satisfaction Formulation. IJCAI-9, International Joint Conference on Artificial Intelligence, Los Angeles, California, August, 1985, pp. 1066-1072.

5. Doran, J. and Michie, D. Experiments with the Graph-Traverser Algorithm. Proceedings of the Royal Society, 294 (A), 1966, pp. 235-259.

6. Feigenbaum, Edward A. and Barr, Avram (Ed.). *Handbook of Artficial Intelligence, Vol. I.* Wm. Kaufmann, Los Altos, California, 1981.

7. Feigenbaum, Edward A. and Cohen, Paul R. (Ed.). *Handbook of Artficial Intelligence, Vol. III.* Wm. Kaufmann, Los Altos, California, 1982.

8. Fikes, R.E. and Nilsson, N.J. "STRIPS: A new approach to the application of theorem proving to problem solving". *Artificial Intelligence 2* (1971), 189-208.

9. Garey, M.R. and Johnson, D.S.. *Computers and Intractibility.* W.H. Freeman, San Francisco, 1979.

10. Gaschnig, John. A Problem Similarity Approach to Devising Heuristics: First Results. IJCAI-6, International Joint Conference on Artificial Intelligence, Tokyo, August, 1979, pp. 301-307.

11. Guida, Giovanni and Somalvico, Marco. "A Method for Computing Heuristics in Problem Solving". *Information Sciences 19* (1979), 251-259.

12. Hart, P.E., Nilsson, N.J., and Raphael, B. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". *IEEE Transactions on Systems Science and Cybernetics SSC-4*, 2 (1968), 100-107.

13. Hayes, J.E., Michie, D., Pole, K.E. and Schofield, P.D.A. A Quantitative Study of Problem-Solving Using Sliding Block Puzzles: The 'Eight-Puzzle' and a Modified Version of the Alexander Passalong Test. Experimental Programming Report No. 7, Experimental Programming Unit, University of Edinburgh, 1965.

14. Held, Michael and Karp, Richard M. "The Traveling-Salesman Problem and Minimum Spanning Trees". *Operations Research 18* (1970), 1138-1162.

15. Held, Michael and Karp, Richard M. "The Traveling-Salesman Problem and Minimum Spanning Trees: Part II". *Mathematical Programming 1* (1971), 6-25.

16. Horowitz, Ellis and Sahni, Sartaj. *Fundamentals of Computer Algorithms.* Computer Science Press, Rockville, Maryland, 1978.

17. Johnson, W.A. and Storey, W.E. "Notes on the '15' puzzle". *Am. J. Math 2* (1879), 397-404.

18. Korf, Richard E. "Towards a Model of Representation Changes". *Artificial Intelligence 14* (1980), 41-78.

19. Korf, Richard E.. *Learning to Solve Problems by Searching for Macro-Operators*. Pittman, London, 1985.

20. Korf, Richard E. "Depth-First Iterative-Deepening: An Optimal Admissable Tree Search". *Artificial Intelligence 27* (1985), 97-109.

21. Kruskal, J. B. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. Proceedings of the American Mathematical Society, 1956, pp. 48-50.

22. Lenat, Douglas B. "The Nature of Heuristics". *Artificial Intelligence 19* (1982), 189-249.

23. Lenat, Douglas B. "Theory Formation by Heuristic Search". *Artificial Intelligence 21* (1983), 31-59.

24. Lenat, Douglas B. "EURISKO: A Program That Learns New Heuristics and Domain Concepts". *Artificial Intelligence 21* (1983), 61-98.

25. Lin, S., and Kernighan, B.W. "An Effective Heuristic Algorithm for the Traveling-Salesman Problem". *Operations Research 21* (1973), 498-516.

26. Loyd, Sam. *Mathematical Puzzles of Sam Loyd*. Dover, New York, 1959.

27. Michalski, Ryszard, Carbonell, Jaime G., and Mitchell, Tom M.. *Machine Learning: An Artificial Intelligence Approach*. Tioga, Palo Alto, California, 1983.

28. Newell, Allen and Simon, Herbert A. GPS: A Program that Simulates Human Thought. In *Computers and Thought*, E. A. Feigenbaum and J. Feldman, Eds., McGraw-Hill, New York, 1963, pp. 279-293.

29. Newell, Allen, and Simon, Herbert A.. *Human Problem Solving*. Prentice-Hall, Englewood Cliffs, NJ, 1972.

30. Nilsson, Nils J.. *Problem-Solving Methods in Artifical Intelligence*. McGraw-Hill, New York, 1971.

31. Nilsson, Nils J.. *Principles of Artificial Intelligence*. Tioga, Palo Alto, 1980.

32. Pearl, Judea. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, Reading, Mass., 1984.

33. Pohl, Ira. The Avoidance of (Relative) Catastrophe, Heuristic Competence, Genuine Dynamic Weighting and Computational Issues in Heuristic Problem Solving. IJCAI-3, International Joint Conference on Artificial Intelligence, Stanford, California, August, 1973, pp. 12-17.

34. Pohl, Ira. "Practical and Theoretical Considerations in Heuristic Search Algorithms". *Machine Intelligence 8* (1977), 55-72.

35. Polya, George. *How to Solve It*. Princeton University Press, Princeton, NJ, 1945.

36. Purcell, Edward T. *Machine Learning of Heuristics for Ordered-Search Algorithms*. Ph.D. Th., University of California at Los Angeles, 1978.

37. Rendell, Larry. "A New Basis for State-Space Learning Systems and a Successful Implementation". *Artificial Intelligence 20* (1983), 369-392.

38. Sacerdoti, Earl D. "Planning in a Hierarchy of Abstraction Spaces". *Artificial Intelligence 5* (1974), 115-135.

39. Samuel, A.L. Some studies in machine learning using the game of checkers. In *Computers and Thought*, McGraw-Hill, New York, 1963, pp. 71-105.

40. Schofield, P. D. A. "Complete Solution of the 'Eight-Puzzle'". *Machine Intelligence 1* (1967), 125-133.

41. Sussman, G. J.. *A Computer Model of Skill Acquisition.* American Elsevier, New York, 1975.

42. Tait, P. G. Note on the Theory of the '15' Puzzle. Proceedings of the Royal Society, Edinburgh, 10, 1880, pp. 664-665.

43. Tarjan, R.E.. *Data Structures and Network Algorithms.* Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1983.

44. Valtorta, Marco. A Result on the Computational Complexity of Heuristic Estimates for the A^* Algorithm. IJCAI-8, International Joint Conference on Artificial Intelligence, Karlsruhe, West Germany, August, 1983, pp. 777-779.

45. Wilson, Richard M. "Graph Puzzles, Homotopy, and the Alternating Group". *Journal of Combinatorial Theory (B) 16* (1974), 86-96.

46. Winston, Patrick Henry. *Artificial Intelligence.* Addison-Wesley, Reading, Massachusetts, 1977.