Machine Learning, 10, 341-363 (1993)

Wastewater Treatment Systems from Case-Based Reasoning

SRINIVAS KROVVIDY SKROVVID@UCENG.UC.EDU
WILLIAM G. WEE WWEE@UCENG.UC.EDU
Department of Electrical and Computer Engineering, University of Cincinnati, Cincinnati OH 45221-0030

Abstract. Case-Based Reasoning (CBR) is one of the emerging paradigms for designing intelligent systems. Preliminary studies indicate that the area is ripe for theoretical advances and innovative applications. Heuristic search is one of the most widely used techniques for obtaining optimal solutions to many real-world problems. We formulated the design of wastewater treatment systems as a heuristic search problem. In this article we identify some necessary properties of the heuristic search problems to be solved in the CBR paradigm. We designed a CBR system based on these observations and performed several experiments with the wastewater treatment problem. We compare the performance of the CBR system with the A* search algorithm.

Keywords: Heuristic search, case-based reasoning, learning, A* algorithm

1. Introduction

In recent years researchers have been investigating a new paradigm for problem solving and learning, by using specific solutions to specific problem situations. The basic idea is to make use of old solutions while solving a new problem. Such an approach is known as case-based reasoning (CBR). During the past few years researchers have been investigating a variety of tasks such as general problem solving (Kolodner, 1987), legal reasoning (Ashley & Rissland, 1988), medical diagnosis (Bareiss, 1989) and opportunistic learning (Hammond, 1989) in the context of CBR. In a related work, Stanfill and Waltz (1986) have studied an application for word pronunciation using memory-based reasoning. In all these cases the case-based reasoning approaches have been used to support learning techniques and improve problem-solving strategies.

Heuristic search is one of the most important techniques in the field of artificial intelligence. Several real-world problems involve searching for an optimal solution under several constraints. The computational complexity of these problems require the reduction of the solution search space. Therefore, CBR is expected to be a good methodology for the problems for which heuristic search has been used. Application of CBR for heuristic search has not been studied extensively. The only known previous work is an application for the eight-puzzle problem by Lehnert (1987; 1988). While this study showed some encouraging results, its implementation is influenced by the fact that eight-puzzle embodies complete knowledge. The case-based structure is built based on the availability of certain index functions specific to the problem. The indexing technique used in this study maps all legal eight-puzzle boards to 12 possible indices, represented in terms of 12 metric equivalent

classes. Any path in the search tree is considered as a sequence of integers between 0 and 12 (not including 1), while 0 representing the equivalence class of goal state. The search for any new problem is performed using these index functions. Bradtke and Lehnert (1988) suggest the use of a "perfect index" function that maps every input problem state onto a number that encodes the minimum number of moves required to transform the problem state into a goal state. This index assumes the availability of a maximally difficult problem. Such a perfect index function needs considerable knowledge about the problem space and the current goal state. In our present study we show that from an estimate of the cost for a maximally difficult problem we can search the case base for partial solutions and get optimal solutions efficiently. We also show that even if such an estimate from the maximally difficult problem is not available, we can generate bounds for the cost and use these bounds to search for a partial solution.

The primary problem for a CBR system is of determining those old situations that are "similar" to the current case. The relevant old solutions need to be organized in the memory so that the descriptions of the problem at hand can be used to retrieve the relevant cases. Relevance is often determined not by the obvious features of the input problem, but by some abstract relationships among features. In the heuristic search problems, the relevant old solutions are used to find the best place to "jump onto" an optimal path to a solution. In this article we identify some conditions under which two optimal solutions can share a partial solution for certain heuristic search algorithms. We design a case base that can exploit the above observation. We also provide an algorithm to solve any new problem using such a case base. The case base automatically updates itself with new knowledge as and when new problems are solved. In the next section we describe a real-world engineering problem, the wastewater treatment problem that motivated this research work. We show an example of using CBR to find an optimal treatment design for a given waste stream.

2. Wastewater treatment problem

The wastewater consists of several chemicals (compounds) that need to be removed during the treatement process. A variety of treatement processes and technologies exist that are capable of reducing the concentrations of one or more contaminants. In general, several compounds appear together in a mixture, and we may need to use two or more processes in series to achieve the desired level of treatment. Such series of treatment processes are called treatment trains. A treatment train is a sequence of individual unit processes where the effluent of one process becomes the influent to the next process. Therefore designing a treatment system involves selecting and sizing a set of these treatment processes (technologies) that will meet all the treatment objectives.

2.1. Problem description

A treatability database has been developed by the Water and Hazardous Waste Treatment Research Division of EPA, Cincinnati. Each record in the database consists of the name of a chemical, the type of a treatment process that was applied to the waste, and the observed

influent and effluent levels of chemical through the treatment process. Several such records are collected for each chemical. A concept learning system based on Quinlan's (1986) ID3 algorithm is used to extract knowledge rules from this database. This knowledge is represented in the form of unit process descriptions. One such description is generated for each treatment process. Each of these descriptions specify the effect of a particular treatment process on several compounds at different concentrations. A typical unit process description is given as follows:

```
General Format
   <technology : name
      { chemical1 --> infl_conc : effl_conc
        chemical2 --> infl_conc : effl_conc
                                        .
                                        .
      }>

An Example
   <technology : Activated Sludge
      { Benzene --> 0.1-1000 mg/L : 0.01 mg/L
        Phenol --> 10 mg/L : 0.01-0.1 mg/L
                                        .
                                        .
      }>
```

The above description for Activated Sludge describes its removal capabilities for chemicals, benzene, phenol, etc., at different influent concentrations. These unit process descriptions are supplemented with some external rules specifying the interactions between these technologies. A typical external rule is

Never(Activated Sludge, Aerobic Lagoons)

which means that the treatment train can contain only one of the two technologies, Activated Sludge and Aerobic Lagoons, but not both. Several such rules are included in the unit process descriptions.

Let there be n chemicals present in the water. Let the influent concentrations be I_1, I_2, ..., I_n and the target concentrations be O_1, O_2, ..., O_n. These two concentrations can be represented as two different points in an n-dimensional space. The objective of the problem is to determine the sequence of technologies that need to be applied to reach from the given influent concentrations to the target concentrations. Each unit process description determines the operation of travelling from one point (corresponding to the influent concentrations) to another point (corresponding to the effluent concentrations) in the n-dimensional space. A heuristic-based state-space search mechanism is developed to synthesize the processes for reducing the influent concentrations to the target concentrations. (Krovvidy et al., 1991).

2.2. CBR for wastewater treatment

The process of obtaining the optimal treatment trains using a heuristic search function involves a large search space. This search effort can be reduced if we can use some of the old treatment train already existing in the case base. We use a simple example to show the advantage of CBR in the heuristic search procedure. Assume that there are three contaminants in the water. We are required to design a treatment train for the wastewater to reduce the initial concentrations to the specified safe concentrations for the three contaminants: (All concentrations are given in μg/L)

Compound	Initial Concentrations	Safe Concentrations	
Contaminant1	200–100	50–1	
Contaminant2	3,000–2,000	50–1	(2.1)
Contaminant3	800–600	50–1	

We need to have an intelligent search mechanism to find out whether a solution exists in the case base for this problem. Assuming that there is no such solution present in the case base, we use A* algorithm to find an optimal treatment train for this node. Let technology E be used at state 1 with the effluent concentrations as

200–100	Technology E	200–100	
2,000–1,100	--------->	1,100–1,000	(2.2)
800–600		600–500	

We again search the case base to find a solution for the new levels of concentrations. Suppose there is a treatment train in the case base as shown in (2.3):

10,000	Technology A	2,000–1,000	Technology B	200–100	
5,000	--------->	4,000–3,000	--------->	1,200–1,000	
9,000		1,000–900		900–600	
				|	
				|	

|					
|	Technology C	100–10	Technology D	10–1	
	--------->	400–100	--------->	40–10	
		400–100		40–5	(2.3)

The effluent concentrations shown in (2.2) already exist in the treatment train (2.3) after the application of technology B. Therefore, the solution to be retrieved from the case base is C -> D. The overall treatement train for the new problem is E -> C -> D.

We now describe how to index a case base to obtain the most suitable case for a given problem using a formal analysis. In the next section we develop the necessary theoretical background. In section 4 we describe the structure of the case base and the procedure for

inserting/solving a new case into the existing case base. In section 5, we present the application of CBR for wastewater treatment problem. In section 6, we illustrate an example for the wastewater treatment design problem. In section 7, we provide the results from our experiments. In section 8, we conclude with some directions for further research.

3. Theory development

An excellent description of heuristic search methods and different algorithms is available in the works of Pearl (1984) and Nilsson (1972). In this section we introduce our notation for heuristic search functions and wastewater treatment problem and prove some properties that are useful in the CBR. A brief description of the A* algorithm is given in the appendix.

Notation

S	Start node
G	Goal node
N	Intermediate node
N′	A successor node to N
k(N, N′)	Cost incurred from going to N to N′
P	A new problem state
n	Number of compounds
$I_1, \ldots I_n$	Input concentrations of the compounds
$O_1, \ldots O_n$	Goal concentrations of the compounds
$g_S^*(N)$	The cost of the cheapest path going from S to N
$h^*(N)$	The cost of the cheapest path going from N to G
$f_S^*(N)$	$g_S^*(N) + h^*(N)$
	The optimal cost of all solution paths from S, constrained to go through N
$g_S(N)$	The cost of the current path from S to N
h(N)	An estimate of h*(N) and
$f_S(N)$	$g_S(N) + h(N)$
J	Total number of technologies
C_j $j = 1..J$	Unit cost of jth technology
R_{kyi}	Fraction of the kth compound removed by the yth technology at the ith stage
E_{ki}	Effluent concentratioin of kth compound after the ith stage
C_{min}	Minimum cost per unit removal among all technologies
C_{max}	Maximum cost per unit removal among all technologies
$\emptyset$	Pivotal node with maximum possible concentrations for all compounds
$f\varphi^S(N)$	$g_\varphi^*(N) + g_S(N) + h(N)$
τ	Threshold value for remembering partial solutions.

Monotonicity: If N′ is any descendent of N, then h(N) is said to be monotonic if

$$h(N) \leq k(N, N') + h(N') \text{ for all pairs of } (N, N')$$

where k(N, N′) = actual cost incurred in going from N to N′.

Pivotal f-values: For a given start node 'S', define any other node 'x' not on the optimal path as a pivot. Then define $f_x(N) = g^*_x(S) + f_S(N)$. These $f_x(N)$ values are called the pivotal f-values, with 'x' as a pivot. The term $f_x(N)$ denotes the cost incurred to find an optimal path from x to G passing through S and N.

Lemma 1: If h is monotonic, then for any pivotal node 'x', the pivotal f-values of the sequence of nodes expanded by A* during the search from a node 'S' is non-decreasing.

Proof: For all nodes on the path, we know that $f_S(N)$ values are non-decreasing (Nilsson, 1972). It can be seen from the definition of pivotal f-values that they are obtained by adding a constant amount, namely $g^*_x(S)$, to the corresponding f_S value. Hence the non-decreasing order is still maintained.

In the subsequent discussions, we assume that h is monotonic. Pearl (1984) argues that "monotonicity is not an exceptional property but rather a common occurrence among admissible heuristics."

Consider figure 1. Let φ be a pivotal node and S and P be two different start nodes with G as the goal node. Let N be the node present on the optimal paths from S and P.

$g^*_\varphi(P)$ = The cheapest cost for φ to P.

$g_P(N)$ = The cost incurred from P to N.

$h(N)$ = Estimated cost from N to G.

$g^*_\varphi(S)$ = The cheapest cost from φ to S.

$g_S(N)$ = The cost incurred from S to N.

Proper pivotal node: Let 'N' be a node on two different optimal paths originating from S and P. If for some node 'φ', $g^*_\varphi(P) + g_P(N) = g^*_\varphi(S) + g_S(N)$, then '$\varphi$' is called a proper pivotal node.

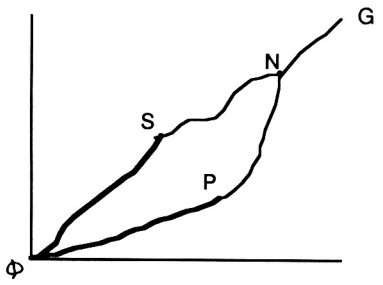

Figure 1. Pivotal f-values for a node on two optimal paths.

Lemma 2: If 'φ' is a proper pivotal node, then for any node 'N' present on two optimal paths originating from S and P the pivotal f-value for 'N' computed from either path is the same.

Proof: Let $f^S_\varphi(N)$ be the pivotal f-value computed from the path originating from S and $f^P_\varphi(N)$ be the pivotal f-value computed from the path originating from P.

$$f^S_\varphi(N) = g^*_\varphi(S) + f_S(N)$$

$$= g^*_\varphi(S) + g_S(N) + h(N)$$

$$f^P_\varphi(N) = g^*_\varphi(P) + f_P(N)$$

$$= g^*_\varphi(P) + g_P(N) + h(N)$$

Since 'φ' is the proper pivotal node, we have

$$g^*_\varphi(P) + g_P(N) = g^*_\varphi(S) + g_S(N)$$

$$\therefore f^S_\varphi(N) = f^P_\varphi(N)$$

Therefore, $f_\varphi(N)$ value is unique irrespective of the path chosen.

The interpretation for lemma 2 is that no matter which path we choose, the cost for the solution from the pivotal node to the goal node constrained to go through N is the same. However, we only need to know $g^*_\varphi(S)$ and $g^*_\varphi(P)$ values and the knowledge of actual paths from φ to S or P is not necessary.

Lemma 3: If 'N' is present on two different optimal paths form S and P, and if $f^S_\varphi(N) = f^P_\varphi(N)$ for some node 'φ', then 'φ' is a proper pivotal node.

Proof: Given

$$f^S_\varphi(N) = f^P_\varphi(N)$$

$$g^*_\varphi(S) + f_S(N) = g^*_\varphi(P) + f_P(N)$$

$$g^*_\varphi(S) + g_S(N) + h(N) = g^*_\varphi(P) + g_P(N) + h(N)$$

$$\therefore g^*_\varphi(S) + g_S(N) = g^*_\varphi(P) + g_P(N)$$

Therefore, 'φ' is a proper pivotal node.

Lemma 3 is useful to identify and establish the existence of a proper pivotal node for a given search space.

Indexing onto an optimal path: We use pivotal f-values as our index function. As defined earlier, this function has two parts: the cost from the most difficult problem to the given problem, and the estimated cost from the new problem state to the goal state.

Let $SS_1S_2 \ldots S_kG$ be an optimal path from S to G. Let P be a node from which we need to find an optimal path to G. Let 'φ' be a proper pivotal node. From lemma 1, we know that

$$f^S_\varphi(S_1) \leq f^S_\varphi(S_2) \leq \ldots \leq f^S_\varphi(S_i) \leq f^S_\varphi(S_{i+1}) \leq \ldots. \leq f^S_\varphi(S_k) \leq f^S_\varphi(G)$$

Let 'N' be an intermediate node generated as a part of the optimal path from P to G using the A* algorithm. We can search for the presence of 'N' in $SS_1S_2 \ldots S_kG$ by computing $f^P_\varphi(N)$ and locate S_i such that

$$f^S_\varphi(S_i) = f^P_\varphi(N).$$

Since f^S_φ values are in ascending order, this search can be performed efficiently.

The basic idea is to generate an optimal path from a given node P until it has some node 'N' on its expandable nodes list that also exists on the optimal path from some other node S. Therefore, the solution path for P consists of two parts; the path from P to 'N' (new generated) and the path from 'N' to G (already generated as part of solution for S).

In the following sections we describe the structure of the case base and procedures for inserting/solving any new problem using the case base. The existence of a proper pivotal node specifies that any two nodes P and S share a part of their optimal paths if both P and S themselves belong to two different optimal paths originating from the proper pivotal node to the goal node G. In the subsequent discussion, we simply use f-values instead of pivotal f-values wherever there is no ambiguity.

4. Design and searching the case base

In order to design a structure for the case base, we must describe a representation for each case. In a heuristic search problem, each case can be viewed as a sequence of states that takes a given problem state to a targeted goal state. Two or more cases may share part of their solution paths. Each state N in the solution path is represented as a D-tuple, $\langle N_1, N_2, \ldots, N_D \rangle$ where N_i determines the position of N in the ith dimension. In section 4.1 we describe the organization of the case base as a hierarchical structure. In section 4.2 we present an algorithm to index the case base and retrieve partial solutions to solve any new problem.

4.1. Structure of the case base

The case base is constructed as an hierarchical layered structure. At the primary level we have sections arranged in the ascending order of the f-values of the nodes. All nodes present in a given section have their f-values within a given range. Each section is further divided into several partitions. Each partition has a partial path generated in the optimal search. We consider these partitions as chunks of knowledge. The sections are connected to each other through the partitions. Figure 2 depicts a case base with m sections. Each section Si has Pi partitions, and each partition j from section Si has Nijk nodes.

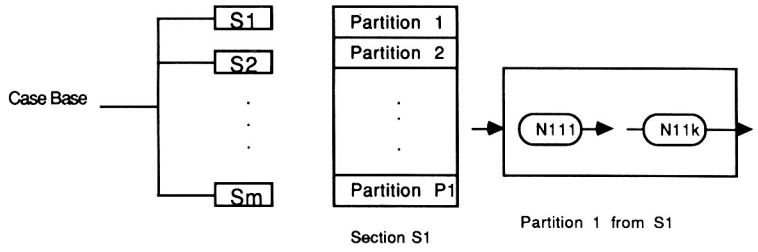

Figure 2. Structure of the case base.

4.2. Case-base construction/search

The case base is built in an incremental fashion. When a new problem state is given, the existing case base is used to find the closest case from its memory. An optimal path between the problem state and the old case is found. In the initial stages, when the case base has very few cases, the solution for a new problem requires more searching. However, after solving a sufficient number of problems, the case base improves its performance for solving any new problems, as it needs less search space. In this section we describe the procedures for constructing and searching the case base for solving a given problem.

Let P be a node representing the new problem state. Let N be an intermediate node generated while searching for a path from P to G. Given N, we can identify whether it is present in the existing case base in the following way:

```
get_case(N, case-base)          /*Get N from the case base */
        find the f-value for the node N
        obtain the section to which it can belong
        get_partition(N, section)
end get_case

get_partition(N, S)                    /* Get N from the section S */
        for each partition 'k' in S
           let p1k, p2k, ..., plk be the path stored in the partition k.
           find the minimum j such that N can be inserted between p(j-1)k and pjk.
           (Since the f-values for p1k, p2k, ..., plk are in the ascending order, the location
              of N can be easily found such that f-value of p(j-1)k < f_φ^P(N) ≤ f-value
              of pjk
           choose that partition with f_φ^P(N) = f-value of pjk associated with the existing
              path
           (Break any ties arbitrarily, always in favor of the goal node)
           Append the path from pjk to G, to the path from P to N and return
           If no such partition exists, then
              use A* algorithm to obtain the next node to be expanded
              while using A* algorithm, before expanding a new node
                 verify if it already exists in the case base using the above steps.
end get_partition
```

In the following sections we present a heuristic state-space search approach for wastewater treatment problem. We then present CBR approach for generating optimal treatment trains.

5. Heuristic state-space approach for wastewater treatment system

A heuristic-based state-space search mechanism is used to synthesize the processes for reducing the influent concentrations to the target concentrations. In the next section we describe the treatment train formally, and in section 5.2, we present a heuristic search algorithm for the wastewater treatment problem. In section 5.3 we present the application of CBR for the wastewater treatement problem.

5.1. Optimal treatment train

The treatment train can be considered as a chain of processes that are selected at different stages of the searching process. Let the unit process chain at the ith stage be as shown in figure 3.

The vectors $\mathbf{X}_i$ and $\mathbf{X}_{i+1}$ of size n, denote the influent and effluent concentrations of n compounds before and after stage 'i', where at stage 'i', we apply a technology to the water.

Let the removal efficiency, R, represent the fraction of a given compound removed. Then

$$R = \%\text{Removed}/100 \tag{5.1.1}$$

Let I_k represent the input influent concentration of the k^{th} compound. Let R_{kyi} represent the fraction of the k^{th} compound removed by the y^{th} unit process technology in the i^{th} step or order in the total treatment train.

Let E_{ki} represent the effluent or output concentration level of the k^{th} compound from the y^{th} unit process used in the i^{th} step. Then

$$E_{ki} = E_{k,i-1} * (1 - R_{kyi}) \tag{5.1.2}$$

The last two subscripts indicate that the y^{th} technology is used in the i^{th} step of the total treatment process. Also note that

$$E_{k0} = I_k \tag{5.1.3}$$

the original concentration.

For example, for technology 1 followed by technology 2 after the beginning of the treatment process, for the k^{th} compound we have

$$E_{k2} = I_k * (1 - R_{k22}) * (1 - R_{k11}) \tag{5.1.4}$$

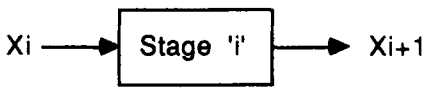

Figure 3. Treatment train at ith stage.

whereas, the corresponding equation when technology 2 follows technology 1 would be

$$E_{k2} = I_k * (1 - R_{k12}) * (1 - R_{k21}) \quad (5.1.5)$$

With each R_{kyi} there is an associated cost H_{yi}, the cost of treatment plant y at stage i. We need to find the minimum total cost associated with the train sequence of unit processes that bring all the final E_{km}'s below the acceptable level for all k compounds in the wastewater mixture.

Mathematically, we seek the particular sequence train of unit processes, y1, y2, ..., ym such that

$$\text{when all } E_{km}\text{'s} < L_k, \sum_{yi} H_{yi} \text{ is minimal,}$$

where y1 represents the first technology chosen, y2 represents the second technology chosen, etc., where each E_{km} represents the final effluent concentration for the k^{th} compound, and where each L_k represents the acceptable wastewater concentration level for the k^{th} compound.

5.2. Heuristic search for optimal treatment train

The heuristic function is based on the decrease in the expected amount of risk/toxicity after a technology is applied. Let there be n chemicals present in the water. Let the influent concentrations be $I_1, I_2, \ldots, I_n$ and the target concentrations be $O_1, O_2, \ldots, O_n$. Based on the A* search, the algorithm for reducing the influent concentrations to the target concentrations can be described as follows.

Start_state = **S** = $\{I_1, I_2, \ldots, I_n\}$; Goal_state = **G** = $\{O_1, O_2, \ldots, O_n\}$
Present_state = **P** = $\{P_1, P_2, \ldots, P_n\}$
Let g(**P**) = The cost incurred in reducing the concentrations from **S** to **P**
h(**P**) = Estimated cost to reduce the concentrations from **P** to **G**
f(**P**) = g(**P**) + h(**P**)
OPEN = Set of nodes to be examined
CLOSE = Set of nodes already examined
P = **S**, OPEN = { }, CLOSE = { }
OPEN = OPEN ∪ {**S**}
repeat
  select the node **P** with minimum f-value among all the nodes in OPEN
  OPEN = OPEN − {**P**}
  IF **P** ≠ **G** then
    CLOSE = CLOSE ∪ {**P**}
    Generate all the successors of **P** and place them in OPEN
    Establish a link between **P** and each of its successor.
    (The unit process descriptions are used to obtain these successors)

until **P** = **G**
Retrieve the treatment train by tracing the back path from **G** to **S**.

The choice of the least cost technology is based on a heuristic function developed in the next section.

5.3. Cost-based heuristic function

In this section we develop a heuristic function that will be used in the state space search process to generate the treatment train. The heuristic search is performed using the A* algorithm. Section 5.3.1 describes the concept of cost per unit removal. Section 5.3.2 presents a description of the application of this concept to obtain the optimal treatment train using A* algorithm.

5.3.1. Cost per unit removal of toxicity

Assume that we have the cost values for the treatment processes that are being considered. The unit process descriptions represent the vector valued function f_{ij} that describes how waste stream is transformed into an effluent stream. As described in section 2.1, the unit process descriptions determine the effect of a particular technology on several compounds at different influent concentrations. Consider figure 3, and let

V_{ij} = 1 if unit 'j' is chosen at stage 'i'

= 0 otherwise

Then

$$\mathbf{X}_{i+1} = \sum_{j=1}^{J} V_{ij} f_{ij}(\mathbf{X}_i) \tag{5.3.1}$$

where J is the total number of processes. Let there be n chemicals present in the water. Let the influent concentrations be $I_1, I_2, \ldots, I_n$ and target concentrations be $O_1, O_2, \ldots, O_n$.
Let

E_{ki} = concentration of compound k before stage i.

E_{ki+1} = concentration of compound k after treatment option j at stage i.

C_j = cost incurred due to option j.

The weight factor associated with the kth compound is given as

$$w_k = (E_{ki} - E_{ki+1}) \tag{5.3.2}$$

Then the cost per unit amount of target toxicity removed by process

$$j = C_j / \sum_{i=1}^{n} w_i \tag{5.3.3}$$

5.3.2. A algorithm applied for optimal treatment train*

The application of the A* algorithm as explained in section 5.2 needs the values for g(**P**) and h(**P**) for any given intermediate node. The g function is defined as the sum of the actual costs of the technologies in the treatment train. Let P be the concentrations at the ith stage in the treatment train;

$$T \dashrightarrow T_2 \dashrightarrow \ldots\ldots \dashrightarrow T_i$$

Then

$$g(\mathbf{P}) = \sum_{k=1}^{i} C_k \tag{5.3.4}$$

where C_k is the cost of technology T_k.

The h(**P**) function is defined as the estimate of the cost of the cheapest paths from **P** to the goal node **G**. This function is defined by finding the lower bounds on the cost of the treatment train for the given wastewater. The procedure to find these bounds is explained using an example.

Consider a wastewater stream with two contaminants, $cont_1$, $cont_2$. Let there be two technologies T_1 and T_2. The unit process descriptions for these technologies are defined as

<Technology : T_1
{ $cont_1$ --> 100 : 75
$cont_1$ --> 200 : 160
$cont_2$ --> 100 : 80
$cont_2$ --> 200 : 150
}>

<Technology : T_2
{ $cont_1$ --> 100 : 40
$cont_1$ --> 200 : 150
$cont_2$ --> 100 : 75
$cont_2$ --> 200 : 160
}>

We also assume that the cost of technology T_1 is 100 units and that of T_2 is 175 units. For T_1, the best removal rate is obtained when $cont_1 = 200$ and $cont_2 = 200$. From (5.3.3), we have the following:

The best cost per unit removal for T_1 = 100/((200 − 160) + (200 − 150)) = 100/90 ≈ 1.1
Similarly, the worst cost per unit removal for T_1 = 100/((100 − 75) + (100 − 80)) = 100/45 ≈ 2.2
The best cost per unit removal for T_2 = 175/((100 − 40) + (200 − 160)) = 175/100 = 1.5
Similarly, the worst cost per unit removal for T_2 = 175/((200 − 150) + (100 − 75)) = 175/75 ≈ 2.33
Therefore, the least unit cost of removal = min {1.1, 1.5} = 1.1
and the maximum unit cost of removal = max {2.2, 2.33} = 2.33

The next section presents a formal description of the above method followed by an expression for the heuristic function for estimating the cost of the treatment train. This heuristic estimate is shown to be monotonic.

Cost bounds for the solution: We compute the upper and lower bounds for the cost per unit amount of removal among all the processes. Let C_{min}, C_{max} denote the possible minimum and maximum costs per unit amount of removal among all the given processes. Let $C_1, C_2, \ldots, C_J$ be the costs for the J technologies, respectively.

For each technology 'j' $(1 \leq j \leq J)$
  For each compound 'k', $(1 \leq k \leq n)$
    Let I^j_{ka} be the influent concentration with maximum removal.
    O^j_{ka} the corresponding effluent concentration.
    I^j_{kb} the influent concentration with minimum removal.
    O^j_{kb} the corresponding effluent concentration.

$$C^j_{min} = C_j / \sum_{k=1}^{n} (I^j_{ka} - O^j_{ka}) \tag{5.3.5}$$

$$C^j_{max} = C_j / \sum_{k=1}^{n} (I^j_{kb} - O^j_{kb}) \tag{5.3.6}$$

Then,

$$C_{min} = \min_j \{C^j_{min}\} \qquad (1 \leq j \leq J)$$

$$C_{max} = \max_j \{C^j_{max}\} \tag{5.3.7}$$

Let G be the node (n-tuple) that has the allowable concentration for each compound. For any given set of concentrations Y (n-tuple), the heuristic estimate for the cost of the treatment train from Y to G is given by

$$h(Y) = |Y - G| * C_{min} \tag{5.3.8}$$

where $|Y - G|$ denotes the componentwise difference, i.e.,

$$|Y - G| = \sum_{i=1}^{n} Y_i - G_i \tag{5.3.9}$$

5.4. Analysis of the wastewater treatment heuristic function

In order to apply the CBR paradigm for the wastewater treatment problem, we need the following conditions to be satisfied:

i) the heuristic is monotonic.
ii) The existence of a proper pivotal node.

Lemma 4: The heuristic function described by equation (5.3.8) is monotonic.

Proof: While C_{min} denotes the minimum possible cost per unit amount of removal, the actual concentrations of Y may require us to use some other technology 'T' with a different cost. If Z is the descendant of Y, and if technology T is used, then

$$h(Z) = |Z - G| * C_{min}$$

$$h(Y) = |Y - Z| * C_{min} + |Z - G| * C_{min}$$

$$= |Y - Z| * C_{min} + h(Z)$$

Since C_{min} is the lower bound among all the technologies, for any technology T,

$$h(Y) \leq C_T + h(Z) \tag{5.4.1}$$

which implies that the heuristic is monotonic.

Let φ be the node with maximum possible concentrations for all compounds. Let S be a node from which there is a treatment train that needs to be stored in the case base. When the solution from S is stored in the case base, the index function $f_\varphi^S(N)$ values for each node 'N' in that solution need to be computed.

$$f_\varphi^S(N) = g_\varphi^*(S) + g_S(N) + h(N) \tag{5.4.2}$$

In this expression, we can compute the exact values of $g_S(N)$ and $h(N)$. However, $g_\varphi^*(S)$ cannot be computed directly, unless we generate an optimal path from φ to S. So, we use a lower bound $g_\varphi^l(S)$ instead of $g_\varphi^*(S)$, where

$$g_\varphi^l(S) = |\varphi - S| * C_{min}.$$

On a similar note, an upper bound on $g^*_\varphi(S)$ is given by $g^2_\varphi(S)$, where

$$g^2_\varphi(S) = |\varphi - S| * C_{max}.$$

Therefore,

$$g^1_\varphi(S) \leq g^*_\varphi(S) \leq g^2_\varphi(S)$$

All the nodes in the optimal path are stored using $g_\varphi{}^1(S)$ instead of $g^*_\varphi(S)$.

Lemma 5: Let φ be the node with maximum possible concentrations for all the compounds. Let S be a node from which there is a treatment train already available in the case base. If P is a new node, then there exists lower and upper bounds for the f-values for any intermediate node 'N' to be present on the optimal paths from S and P as well.

Proof: In the CBR paradigm, whenever, we are given a new problem state P, we need to search for the new node in the case base. All the solutions in the case base are stored using a lower bound on its f-value as an index.

Let

$$\{f^S_\varphi(N)\}_{min} = g^1_\varphi(S) + g_S(N) + h(N)$$

$$\{f^S_\varphi(N)\}_{max} = g^2_\varphi(S) + g_S(N) + h(N)$$

$$\{f^P_\varphi(N)\}_{min} = g^1_\varphi(P) + g_P(N) + h(N)$$

$$\{f^P_\varphi(N)\}_{max} = g^2_\varphi(P) + g_P(N) + h(N)$$

To find an intermediate node 'N' present on the optimal paths from S and P as well, we compute an upper bound f_{up} and a lower bound f_{lo} for f-values and confine our search in only those sections that are bounded by these f-values.

$$f_{up} = \min \{\{f^S_\varphi(N)\}max, \{f^P_\varphi(N)\}max\}$$

$$f_{lo} = \{\{f^S_\varphi(N)\}min, \{f^P_\varphi(N)\}min\}$$

Within these bounds, we use the *property of dominance* (explained in the following paragraph) to retrieve the closest node for a given problem state. For smaller case bases, we can search the entire case base without computing any bounds. For case bases of large size, computing these bounds and searching within these bounds is more efficient.

Property of dominance: Let $p_1, p_2, \ldots, p_m$ be points belonging to the n-dimensional space E^n, with coordinates $x_1, x_2, \ldots, x_n$. Point p_1 dominates point p_2 (denoted by $p_2 \prec p_1$) if $x_i(p_2) \leq x_i(p_1)$, for $i = 1, 2, \ldots, n$. Given a set of m points in E^n the relation dominance on set S is clearly a partial ordering on S for $n > 1$ (Preparata & Shamos, 1985).

In the wastewater treatment problem let $q_1, q_2, \ldots, q_m$ denote a treatment train. The nodes $q_1, q_2, \ldots, q_m$ are n-tuples representing the intermediate concentrations of n-compounds in the treatment train. It can be observed that $q_m \angle q_{m-1} \angle \ldots \angle q_1$, as we only remove/reduce the concentrations of any chemical and we never add/increase the existing concentrations. This property is used to search for the closest node in an existing solution path for a given problem.

6. An example of CBR for wastewater treatment design

Let the influent concentrations for seven compounds be as follows:

Compound	Concentration in μg/L
Phenol	10,000
Toluene	10,000
Benzene	10,000
Chloroform	10,000
Trichloroethylene	10,000
Methylene Chloride	10,000
Ethylbenzene	10,000

(6.1)

After generating unit process descriptions for 10 technologies, the minimum and maximum costs per unit removal are found to be 0.0318 and 1.3945, respectively. The goal concentrations are specified as 100 μg/L for all the compounds. Let A represent the node corresponding to the concentrations shown in (6.1). The f-value of A is found to be 14.864. Since no previous cases exist, the A* algorithm is used to find the treatment train. The treatment train along with the intermediate concentrations and their f-values are shown in (6.2).

f = 14.864		f = 14.948		f = 15.676		f = 16.975
10,000		10,000		100		100
10,000		100		100		100
10,000	Steam	100	Anaerobic	100	Activated	100
10,000	----->	100	------->	100	----->	100
10,000	Stripping	100	Fixed Film	100	Sludge	100
10,000		100		100		100
10,000		10,000		10,000		100
A		B		C		G

(6.2)

As described in section 4, the case base is arranged with sections having increasing f-values. The solution generated for the given problem is denoted as A --> B --> C --> G.

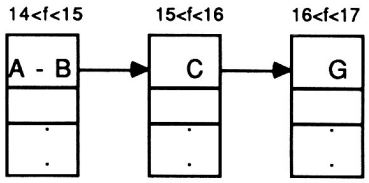

Figure 4. Case base after solving first problem.

The segment A --> B has f-values between 14 and 15. Therefore, they are placed in the section corresponding to that range. Similarly other nodes are stored in the case base according to their f-values. The complete solution is shown in figure 4.

Now suppose a new case is given as shown in (6.3).

Compound	Concentration in μg/L
Phenol	10,000
Toluene	250,000
Benzene	250,000
Chloroform	250,000
Trichloroethylene	250,000
Methylene chloride	250,000
Ethylbenezene	10,000

(6.3)

Let the node corresponding to the given concentrations be D. To check for the presence of D in the case base, we find the lower and upper bound for the f-value and confine search within those sections. This node has its f_{lo} value as 10.102 and its f_{up} value as 14.634. It can be easily seen that this node is not present in the case base. So, we use A* algorithm to generate a partial solution until we reach the goal concentrations or some node that is already present in the case base. After two stages, the resulting partial treatment train along with the intermediate concentrations is as shown:

f = 10.102		f = 10.186		f = 10.318
10,000		10,000		10,000
250,000		2,500		25
250,000	Steam	2,500	Steam	89
250,000	----->	2,500	----->	25
250,000	Stripping	2,500	Stripping	25
250,000		2,500		25
10,000		10,000		10,000
D		E		F

(6.4)

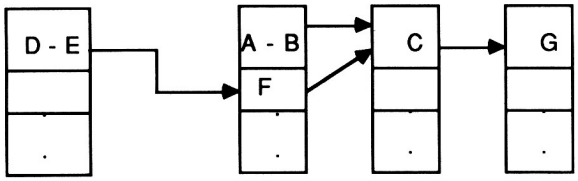

Figure 5. Case base after solving the example.

If the node F is present in the case base, the upper bound for f-value is found to be less than 15. Therefore, we only have to search the case base with f-values less than 15. We can see that node B matches with node F. Therefore, the complete treatment train is D --> E --> F --> C --> G. The complete case base is shown in figure 5.

7. Performance Study

We implemented a CBR system based on the ideas described in this article for the wastewater treatment problem. The unit process descriptions for 10 technologies are developed. We solved 10 different design problems with different concentrations for 7 compounds. Figures 6a and 6b show the improvement in the performance of CBR over A* algorithm for wastewater treatment design.

Remembering partial solutions: We can consider the case memory as a set of solutions. A path from the start node S to the goal node G is the sequence of states from S to G. However, if all the past cases are remembered in the memory, then we need a large amount of storage space. Therefore, it is necessary to remember the cases selectively. Most of the

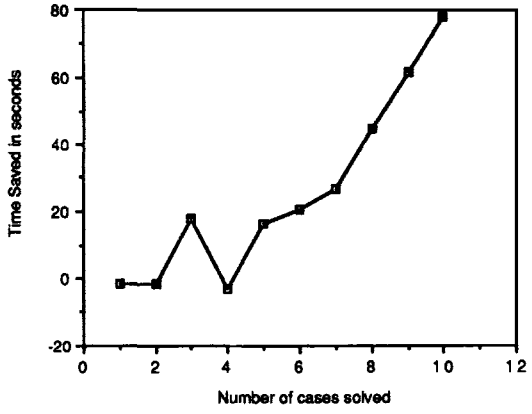

Figure 6a. Time saved by CBR compared with A* for wastewater treatment design.
X-axis The number of cases in the case base
Y-axis (Time taken by A* − Time taken by CBR) in seconds

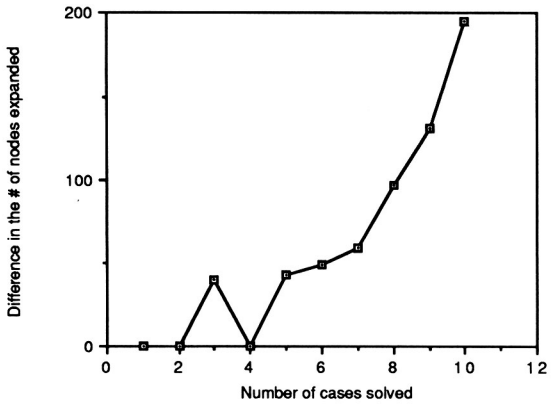

Figure 6b. Performance of CBR and A* search for wastewater treatment design.
X-axis The number of cases in the case base
Y-axis (# of nodes expanded by A* − # of nodes expanded by CBR)

case-based systems address the memory problem by remembering a selected set of cases. Ruby and Kibler (1988) conducted experiments with Lehnert's Case-Based Problem solver to determine if performance could be improved by selectively adding cases to memory. The performance is found to be slightly better when a new case is stored only when it cannot be correctly classified by the instances currently in memory. In our system, we present a different approach to store the cases. Instead of selectively choosing a case, we suggest to store partial solutions for a given case.

We store partial paths of the solutions rather than completely remembering or forgetting the cases. Let $SS_1S_2 \ldots S_kG$ be an optimal path from S to G (S_0 = S, S_{k+1} = G). We store the path S_j to G for some threshold τ, such that

$$j = \text{floor}(\tau^* (k + 1)) \; 0 \leq \tau \leq 1.0$$

In this scheme, we store the complete path if τ = 0.0 and no path for τ = 1.0. We have solved ten different wastewater design problems with different values of τ. In each case, the performance is studied by tabulating Dτ values (see table 1), where

Dτ = (# of nodes expanded by A*) − (# of nodes expanded by CBR with τ as threshold)

Table 1. Performance study with only partial solutions stored.

Threshold	0.0	0.25	0.5	0.75	1.0
Dτ	420	420	416	350	0

8. Conclusions and discussion

In this article, we have shown some properties for applying the CBR approach for heuristic search problems. We have shown how CBR can be used to obtain an optimal solution for a heuristic problem for an existing solution. We have proposed that from an estimate of the cost of a maximally difficult problem, we can efficiently search the case base for partial solutions. Even if such an estimate from the maximally difficult problem is not available, we can generate bounds for the cost and use these bounds to search for a partial solution. We have applied these studies to the problem of wastewater treatment design involving heuristic search. The CBR approach is used to obtain the treatment trains for treating the wastewater. We have observed that the search effort is significantly reduced through the CBR approach. Several experimental studies were performed, and their results have been reported. The empirical studies reported in section 7 demonstrate that CBR has a low overhead compared to normal heuristic search. In general, CBR approach has been found to perform very well after acquiring enough experience. Another advantage of the CBR approach is its incremental nature of solving new cases. The experimental results due to CBR approach are compared with A* algorithm for the number of nodes expanded and the total time spent for obtaining the solution.

The CBR approach and A* algorithm gave the same solutions due to the nature of the underlying search strategy. A* always searches from scratch, independent of the number of problems solved. The effort involved in obtaining the solutions depends only on the distance from the problem state to the goal state. However, CBR improves the performance when there are solutions in the case base that are similar to the given problem state. Therefore, the number of nodes expanded vis-a-vis the time taken to obtain the solution is improved with the number of cases in the case base.

The building of case base was found to be useful to record the old experiences for future use. A simple scheme to remember only partial solutions in the case base was implemented. Threshold values of 0.0, 0.25 and 0.5, and 1.0 were used and the performance of the case base was studied. It could be seen that the degradation in the performance was marginal from $\tau = 0.25$ to 0.5. The increment for τ was chosen based on the average size of any treatment train (4 or 5). For solutions of large size, this increment must be reduced. This scheme of remembering partial solutions is more relevant for problems in the heuristic search domain. The solutions for heuristic search domain can be considered as a sequence of states. For any new problem, when indexing into the case base, the fan-in is high for problem states closer to the goal node. This in turn implies that it is more appropriate to remember those states which have a high likelihood of being indexed, and forget those states which may not be indexed very often.

Further research work needs to be done to identify some other efficient methods of forgetting the old cases to control the growth of the case base. The case base can be periodically restructured such that those old cases that have not been referenced for a sufficiently long time can be deleted. Before deleting such cases, they can be stored in the secondary memory for future use. The A* algorithm can be used both for optimization problems and satisficing problems because the shortest path is the most natural choice for the small-is-quick principle. It should be noted that the breadth-first strategy is a special case of A* with $h = 0$ and $k(N, N') = 0$. Similarly, the uniform-cost strategy is also a special case of

A* with h = 0. Therefore, it is very important to compare the performance of CBR approach with A* algorithm. Another interesting problem for further investigation is to modify the case base for heuristic functions that are non-admissible.

Acknowledgments

The authors thank the reviewers for many useful suggestions. This work was partially supported by a grant from the United States Environmental Protection Agency through Work Assignment 2-16 of EPA contract 68-03-3379 to the University of Cincinnati. Although the research described in this article has been funded wholly or in part by the United States Environmental Protection Agency through Work Assignment 2-16 of EPA contract 68-03-3379 and WA #0031;0-11 to the University of Cincinnati, it has not been subjected to Agency review and therefore does not necessarily reflect the views of the Agency and no official endorsement should be inferred.

Appendix: A* Algorithm

Let S be the start node and G be the goal node. Following the notation from section 3, $f_S(N)$ is an estimate of the cost of a minimal cost path from S to G constrained to go through node N. Let OPEN be the list of nodes that are not yet examined and CLOSED be the list of nodes that are already examined. Then A* algorithm is given as follows:

1) Put node S in OPEN list.
2) If OPEN is empty, exit with failure.
3) Remove from OPEN that node N whose f-value is smallest and put it in CLOSED.
4) If N is a goal node, exit with the solution path obtained by tracing back through the pointers.
5) Expand node N, generating all its successors. For each successor N', compute its f-value.
6) Associate with all N's not already on either OPEN or CLOSED the f-values just computed. Put these nodes in OPEN and direct pointers from them back to N.
7) Associate with those successors that were already on OPEN or CLOSED the smaller of the f-values just computed and their previous f-values. Put on OPEN those successors on CLOSED whose f-values were thus lowered and redirect the pointers from N for those nodes.
8) Go to 2.

References

Ashley, K.D., & Rissland, E.L. (1988). Compare and contrast, a test of expertise. *Proceedings of Case-Based Reasoning Workshop* (pp. 31–36). Florida, CA: Morgan Kaufmann.

Bareiss, R. (1989). *Exemplar-based knowledge acquisition*. New York: Academic Press.

Bradtke, S., & Lehnert, W.G. (1988). Some experiments with case-based search. *Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 133–138), Minneapolis, MN.

Hammond, K.J. (1989). *Case-based planning, viewing planning as a memory task*. New York: Academic Press.
Kolodner, J.L. (1987). Extending problem solving capabilities through case based inference. *Proceedings of the Fourth Annual International Machine Learning Workshop* (p. 167–178). California, CA: Morgan Kaufmann.
Krovvidy, S., et al. (1991). An AI approach for wastewater treatment systems. *Applied Intelligence, 1(3)*, 247–261.
Lehnert, W.G. (1987). Case-based reasoning as a paradigm for heuristic search (COINS Technical Report 87-107). Department of Computer and Information Science, University of Massachusetts, Amherst.
Nilsson, J.N. (1972). *Problem-solving methods in Artificial Intelligence*. New York: McGraw-Hill.
Pearl, J. (1984). *Heuristics intelligent search strategies for computer problem solving*. Reading, MA: Addison-Wesley.
Preparata, F.P., & Shamos, M.I. (1985). *Computational geometry: An introduction*. New York: Springer-Verlag.
Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning*, Vol. 1. Morgan Kaufmann.
Ruby, D., & Kibler, D. (1988). Exploration of case-based problem solving. *Proceedings of Case-Based Reasoning Workshop* (pp. 345–356). Florida, CA: Morgan Kaufmann.
Stanfill, C., & Waltz, D. (1986). Toward memory-based reasoning. *CACM, 29(12)*, 1213–1228.

Received July 2, 1990
Accepted October 17, 1991
Final Manuscript May 7, 1992