Efficient Diagnosis of Multiple Disorders Based on a Symptom Clustering Approach

Thomas D. Wu
MIT Laboratory for Computer Science
545 Technology Square, Cambridge, Massachusetts 02139
tdwu@lcs.mit.edu

Abstract

Diagnosis of multiple disorders can be made efficient using a new representation and algorithm based on symptom clustering. The symptom clustering approach partitions symptoms into causal groups, in contrast to the existing candidate generation approach, which assembles disorders, or candidates. Symptom clustering achieves efficiency by generating aggregates of candidates rather than individual candidates and by representing them implicitly in a cartesian product form. Search criteria of parsimony, subsumption, and spanning narrow the symptom clustering search space, and a problem-reduction search algorithm explores this space efficiently. Experimental results on a large knowledge base indicate that symptom clustering yields a near-exponential increase in performance compared to candidate generation.

Introduction

Many challenges in artificial intelligence derive from the intractability of reasoning. One ubiquitous but computationally hard reasoning task is diagnosis, especially when it involves multiple disorders. The success of multidisorder diagnosis depends heavily on the efficiency of the algorithm employed. This paper augments the capabilities of diagnostic reasoning by developing an efficient representation and algorithm based on a symptom clustering approach.

The current approach to multidisorder diagnosis is based on candidate generation. A candidate is a set of disorders that explains a given set of symptoms. Set-covering approaches to diagnosis [5] explore a "candidate space" [1], also called a "hypothesis graph" [4] or "hitting set tree" [6]. In contrast, the approach introduced in this paper is based on symptom clustering. Whereas a candidate assembles disorders, a symptom clustering partitions symptoms into clusters. It structures symptoms into causal groups, with symptoms in the same cluster hypothetically caused by the same disorder. Experience with our implemented system SYNOPSIS indicates that the symptom clustering approach offers increased efficiency for multidisorder diagnosis.

In the rest of the paper, we develop the symptom clustering representation and show how it is related to candidate generation. We introduce search criteria to help narrow the symptom clustering search space and then devise a problem-reduction search algorithm that explores this space efficiently. Finally, we verify the performance gains of the symptom clustering approach empirically on a large, real-world knowledge base.

This research was supported in part by National Institutes of Health grant R01 LM04493 from the National Library of Medicine and by National Research Service Award T32 GM07753. I am grateful to Ramesh Patil, Peter Szolovits, and Randy Davis for their helpful discussions and to Randolph Miller for generously allowing use of the INTERNIST/QMR knowledge base for testing purposes.

The Candidate Generation Approach

In set-covering approaches to diagnosis, the task is to find minimal candidates that explain a case, where a case is a set of symptoms to be explained. Each symptom has a set of possible causes, which is called a conflict set. In this paper, we will usually refer to this conflict set as the *causes relation*, represented as Causes(s) for symptom s. A candidate explains a symptom if it contains a disorder in Causes(s). A candidate is *valid* for a case if it explains every symptom in the case. A candidate is *minimal* for a case if it is valid and none of its subsets is also valid for that case.

Minimal candidates can be generated recursively. Suppose we already have a minimal candidate for some symptoms S and consider a new symptom s. If the candidate explains s, no change is necessary since the candidate is already minimal for $(S \cup \{s\})$. Otherwise expand the candidate by creating, for each element of Causes(s), a new candidate that includes that element. Finally, prune any candidates that are nonminimal.

An example of the candidate generation algorithm, taken from [1], is shown in Figure 1. There are two features of the candidate generation process worth noting. First, candidates are generated individually, so that each node in the search tree has only one candidate. Second, candidates often share elements; for example, the candidates $[A_2M_2]$ and $[M_3M_2]$ share M_2. These two features are notable in contrast to the symptom clustering approach. In symptom clustering, candidates are generated aggregately rather than individually, and these aggregates are represented implicitly rather than explicitly. By *implicit representation*,

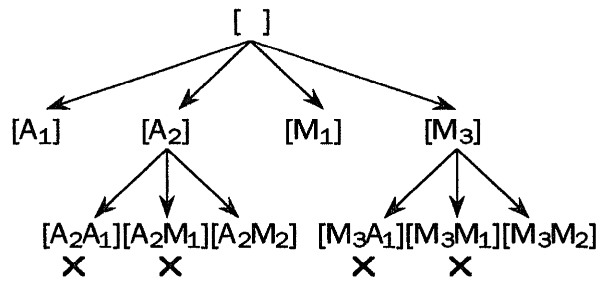

Figure 1: Candidate generation algorithm for a faulty circuit with two symptoms. One symptom can be explained by adders A_1 or A_2 or multipliers M_1 or M_3. The other symptom can be explained by A_1, M_1, or M_2. Pruned candidates, shown with ×'s, are nonminimal.

we mean that shared elements are factored completely into a cartesian product form. For instance, the explicit representation, $[A_2M_2]$ and $[M_3M_2]$, can be represented implicitly as a cartesian product: $\{A_2M_3\} \times \{M_2\}$. For this small example, the difference may seem minor, but it can become critical for large knowledge bases. The implicit representation is much more compact than the explicit representation and helps reduce one source of combinatorial explosion.

The Symptom Clustering Approach

The symptom clustering approach [8] changes the search space from subsets of disorders to clusterings of symptoms. A symptom clustering is a partition of symptoms into clusters. It denotes a possible interpretation of a case, hypothesizing which symptoms are caused by a common disorder and which by separate disorders. A disorder explains a cluster if it can cause every symptom in that cluster. For example, the symptom clustering (ABD)(C) indicates that two disorders are present: one that explains A, B, and D, and one that explains C.

The sets of disorders that meet this interpretation are called *differential diagnoses*, or simply differentials. Each differential contains disorders that can explain one cluster. Differential diagnoses are usually subject to parsimony criteria, which means that they may be shaped by the symptoms in other clusters as well as their own clusters. The cartesian product of the differential diagnoses implicitly represents an aggregate of candidates. A more precise definition of differentials will be presented later in this paper.

The symptom clustering approach offers a novel view of diagnosis. Whereas candidate generation views diagnosis as the *assignment of cause*, symptom clustering views it as the *assignment of structure*. It tries to determine the best way to decompose a problem, since a symptom clustering essentially represents a problem reduction. Its clusters represent the subproblems and its differentials contain solutions to each subproblem. Specifically, a symptom clustering may be expressed as a two-level AND/OR graph, since it contains a conjunction of clusters, and the differential associated with each cluster contains a disjunction of disorders.

Thus, diagnosis becomes primarily a matter of finding the correct structure for a problem and secondarily a matter of finding the correct causes. Focusing on structure in diagnosis opens a rich source of heuristics which might be called *structural heuristics*; these seem to be used commonly in everyday reasoning. Detectives often begin by piecing together evidence rather than trying to identify a culprit for each piece of evidence. In medical diagnosis, problem solving is facilitated by compiled knowledge about clusters of commonly co-occurring symptoms called *syndromes*. Another structural heuristic that ties symptoms together is temporal co-occurrence: symptoms that occur simultaneously are usually related. Structural heuristics such as these appear to be powerful and widely available. However, exploiting them requires viewing problem solving as the search for structure, a view that is inherent to the symptom clustering approach.

Relations between Representations

In a sense, candidates and symptom clusterings are duals, since candidates group disorders, while clusterings group symptoms. This duality can be expressed formally as a satisfaction relation from candidates to symptom clusterings. Formally, we say that candidate D *satisfies* clustering C if there exists a complete matching between clusters in C and disorders in D that can explain them.

For example, let us postulate a knowledge base containing four symptoms (A, B, C, and D) and seven disorders (d_1, d_2, …, d_7). The causes for each symptom are given below:

Symptom	Causes
A	$\langle d_1d_2d_3d_4\rangle$
B	$\langle d_1d_3d_5d_7\rangle$
C	$\langle d_1d_2d_5\rangle$
D	$\langle d_1d_3d_4d_6\rangle$

Then the candidate $[d_3d_5]$ satisfies the clustering (ABD)(C) because d_3 can explain A, B, and D, while d_5 can explain C.

The inverse of the satisfaction relation is the entailment relation: C *entails* D if and only if D satisfies C. We are interested in knowing the set of clusterings that a given candidate satisfies and the set of candidates that a given clustering entails. In the next two subsections, we see how to compute these sets.

Computing the Satisfaction Relation

The satisfaction relation for a given candidate and case can be computed as follows. List the possible effects for each disorder in the candidate. Then select each symptom in the case once, making sure to select at least one effect in each list. The selected symptoms from each list form a cluster in the clustering. For example, consider the candidate $[d_3 d_5 d_6]$:

$$\begin{array}{lllllll} d_3 & \rightarrow & \mathbf{A} & \mathbf{B} & & \mathrm{D} \\ d_5 & \rightarrow & & \mathrm{B} & \mathbf{C} & \\ d_6 & \rightarrow & & & & \mathbf{D} \end{array} \left(\text{or} \begin{array}{llll} \mathbf{A} & \mathrm{B} & & \mathrm{D} \\ & \mathbf{B} & \mathbf{C} & \\ & & & \mathbf{D} \end{array} \right)$$

There are two ways to select the symptoms A, B, C, and D, as shown in boldface in each of the above matrices. These selections correspond to the two clusterings satisfied by $[d_3 d_5 d_6]$, namely, (AB)(C)(D) and (A)(BC)(D).

In addition to the satisfaction relation, the validity or minimality of a particular case can be determined. Validity testing is simple: Each symptom in the case must appear in the effects matrix; otherwise, some symptom cannot be explained. Minimality testing is a bit trickier. Each disorder must have a *justification*, a symptom in the case that appears in exactly one effects list. When each disorder has a justification, the candidate is minimal. For instance, the candidate $[d_3 d_5 d_6]$ shown above is not minimal, because d_6 lacks a justification. (Its only effect, D, appears also as a possible effect of d_3.) However, removing d_6 yields a minimal candidate:

$$\begin{array}{llllll} d_3 & \rightarrow & \mathrm{A}^* & \mathrm{B} & & \mathrm{D}^* \\ d_5 & \rightarrow & & \mathrm{B} & \mathrm{C}^* & \end{array}$$

Each disorder has a justification (shown as starred), so $[d_3 d_5]$ is minimal.

Computing the Entailment Relation

The inverse of satisfaction is entailment: Given a symptom clustering, what are all candidates that satisfy it? Actually, we may be interested in finding either valid candidates or minimal candidates that satisfy the clustering. Thus, we need two entailment relations: V-entailment to obtain valid candidates and M-entailment to obtain minimal candidates.

The V-entailment relation can be computed as follows. Given a clustering, find the *causal intersection* for each cluster c, by intersecting the possible causes for each symptom s in that cluster:

$$\mathrm{Int}(c) = \bigcap_{s \in c} \mathrm{Causes}(s)$$

The valid candidates are then the cartesian product of the causal intersections, excluding sets with duplicate elements. This exclusion ensures that the number of disorders (cardinality) in the candidate equals the number of clusters (dimension) in the clustering. For example, consider the clustering (AC)(B)(D). The causal intersections of the three clusters are $\{d_1 d_2\}$, $\{d_1 d_3 d_5 d_7\}$, and $\{d_1 d_3 d_4 d_6\}$, respectively, yielding 22 valid candidates:

V-entails((AC)(B)(D)) =
$[d_1 d_3 d_4]$, $[d_1 d_3 d_6]$, $[d_1 d_5 d_3]$, $[d_1 d_5 d_4]$, $[d_1 d_5 d_6]$, $[d_1 d_7 d_3]$, $[d_1 d_7 d_4]$, $[d_1 d_7 d_6]$, $[d_2 d_1 d_3]$, $[d_2 d_1 d_4]$, $[d_2 d_1 d_6]$, $[d_2 d_3 d_1]$, $[d_2 d_3 d_4]$, $[d_2 d_3 d_6]$, $[d_2 d_5 d_1]$, $[d_2 d_5 d_3]$, $[d_2 d_5 d_4]$, $[d_2 d_5 d_6]$, $[d_2 d_7 d_1]$, $[d_2 d_7 d_3]$, $[d_2 d_7 d_4]$, $[d_2 d_7 d_6]$

To compute M-entailment, we use *exception filtering*. This process removes *exceptions* from causal intersections to yield differential diagnoses. An exception is a disorder that does not appear in any of the minimal candidates entailed. To find and remove exceptions from causal intersections, we use *filters*. A filter is essentially a cross-cluster constraint: A filter for each cluster identifies exceptions in other clusters. A filter is composed of subfilters, each subfilter deriving from a symptom in the associated cluster. The subfilter represents a situation where its corresponding symptom is a justification; it contains disorders inconsistent with this assumption. However, we may discover that some symptoms cannot be justifications; they impose such strong constraints that they would eliminate all disorders in some other cluster. The subfilters for these symptoms are called *infeasible*. Hence, an exception is a disorder appearing in every feasible subfilter of some filter. The algorithm for exception filtering follows:

1. Initialize: For each cluster c, initialize its differential to be its causal intersection. For each symptom s in c, initialize its subfilter to be the intersection of s with all disorders in other differentials:

$$\begin{aligned} \mathrm{Diff}(c) &\leftarrow \mathrm{Int}(c) \\ \mathrm{Subfilter}(s) &\leftarrow \mathrm{Causes}(s) \cap \bigcup_{c' \neq c} \mathrm{Diff}(c') \\ \mathrm{Filter}(c) &\leftarrow \{\mathrm{Subfilter}(s) \mid s \in c\} \end{aligned}$$

2. Remove infeasible subfilters: A subfilter f for a cluster c is infeasible if it subsumes another differential. Remove infeasible subfilters from each filter:

$$\begin{aligned} \mathrm{Infeasible}(f) &\Longleftrightarrow \exists c' \neq c.\ f \supseteq \mathrm{Diff}(c') \\ \mathrm{Filter}(c) &\leftarrow \mathrm{Filter}(c) - \{f \mid \mathrm{Infeasible}(f)\} \end{aligned}$$

   At this point, every filter must have at least one feasible subfilter. If not, then the clustering entails no minimal candidates.

3. Remove exceptions: Find exceptions for each filter by taking the intersection of its subfilters. Remove these exceptions from all differentials and subfilters:

$$\begin{aligned} \mathrm{Exc}(c) &\leftarrow \bigcap_{f \in \mathrm{Filter}(c)} f \\ \mathrm{Diff}(c) &\leftarrow \mathrm{Diff}(c) - \bigcup_{c} \mathrm{Exc}(c) \\ \mathrm{Subfilter}(s) &\leftarrow \mathrm{Subfilter}(s) - \bigcup_{c} \mathrm{Exc}(c) \\ \mathrm{Filter}(c) &\leftarrow \{\mathrm{Subfilter}(s) \mid s \in c\} \end{aligned}$$

4. Terminate: If no exceptions were found in the last step, halt. Else repeat step 2, since the smaller differentials may have made more subfilters infeasible.

   An example of this process for the cluster

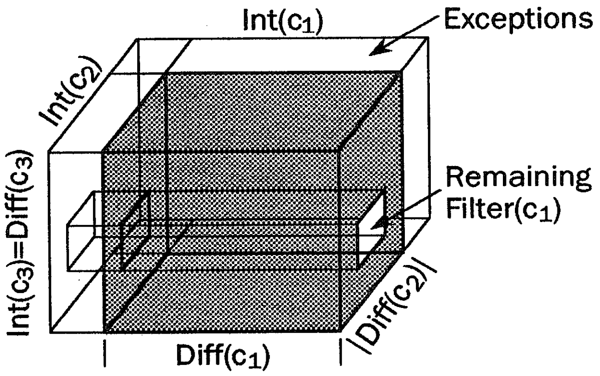

Figure 2: Geometric interpretation of symptom clusterings. A symptom clustering with n clusters bounds a region in n-dimensional candidate space. Minimal candidates entailed by a clustering are points within the shaded region. Causal intersections define the outer boundary; differentials define the inner boundary. Exception filtering removes exterior regions of nonminimality and represents interior regions implicitly.

(AC)(B)(D) is shown below:

	Clusters		
	(AC)	(B)	(D)
Diff:	$\{d_1 d_2\}$	$\{d_1 d_3 d_5 d_7\}$	$\{d_1 d_3 d_4 d_6\}$
Filter:	$((\{d_1 d_3 d_4\}\{d_1 d_5\}))$	$((\{d_1 d_3\}))$	$((\{d_1 d_3\}))$
Exc:	$\{d_1\}$	$\{d_1 d_3\}$	$\{d_1 d_3\}$
Diff:	$\{d_2\}$	$\{d_5 d_7\}$	$\{d_4 d_6\}$
Filter:	$((\{d_4 d_5\}))$	$((\{\}))$	$((\{\}))$
Exc:	$\{\}$	$\{\}$	$\{\}$

The minimal candidates are contained within the cartesian product of these differentials. Here, the differentials are $\{d_2\}$, $\{d_5 d_7\}$, and $\{d_4 d_6\}$, with a cartesian product of $[d_2 d_5 d_4]$, $[d_2 d_5 d_6]$, $[d_2 d_7 d_4]$, $[d_2 d_7 d_6]$. But the remaining exception filter $((\{d_4\}\{d_5\}))$ specifies that candidate $[d_2 d_5 d_4]$ is nonminimal because it contains an element from each subfilter. The minimal candidates are therefore

$$\text{M-entails}((\text{AC})(\text{B})(\text{D})) = [d_2 d_5 d_6], [d_2 d_7 d_4], [d_2 d_7 d_6]$$

Note that these three candidates cannot be factored into a single implicit representation. Exception filtering handles this by representing minimal candidates in two implicit parts: differentials and filters. Although differentials may contain some nonminimal candidates in their cartesian product, this enables them to represent the minimal candidates more economically. Thus, exception filtering achieves efficient representation by ignoring details about minimality that would fragment aggregates of candidates.

Geometric Interpretation

The relationship between symptom clusterings and candidates is shown geometrically in Figure 2. As illustrated, the efficiency of symptom clustering results from three sources. First, candidates are generated *aggregately* instead of individually. This allows entire sets of candidates to be pruned simultaneously. Second, these aggregates are represented *implicitly*. Analogously, we compute the *boundaries* of a candidate region, rather than every point in the volume. Finally, minimal candidates are computed *convexly*. This allows internal regions of nonminimality to remain in the differentials, thereby maintaining the n-dimensional cubic shape of the aggregates.

To a first approximation, we expect symptom clustering to achieve exponential time and space savings over candidate generation. If each symptom clustering has n clusters and each differential has a geometric mean of $\mathcal{D}$ disorders, each clustering entails approximately $\mathcal{D}^n$ candidates. However, the exact savings are difficult to determine, because some of the candidates are not minimal and because a candidate may satisfy more than one symptom clustering. Nevertheless, experimental results presented later lend support to a near-exponential increase in performance.

Search Criteria

Even with an more efficient representation, the symptom clustering search space can still grow rapidly. Thus, we impose additional criteria to help prune the search space, namely, parsimony, subsumption, and spanning. Parsimony is a preference for simplicity, subsumption minimizes unwarranted constraints, and spanning avoids redundant solutions.

Parsimony

Parsimony is a preference for simplicity in explanations. In candidate generation, the criterion of parsimony typically used is minimality. Likewise, the symptom clustering approach can adopt parsimony criteria, such as validity and minimality, to help reduce search. We define a symptom clustering to be valid (likewise minimal) if it V-entails (M-entails) at least one candidate:

$$\begin{aligned} \text{Valid}(C) &\iff \text{V-entails}(C) \neq \emptyset \\ \text{Minimal}(C) &\iff \text{M-entails}(C) \neq \emptyset \end{aligned}$$

Determining whether a clustering is valid or minimal can be accomplished by the entailment procedures given previously.

Subsumption

In addition to parsimony, symptom clustering can reduce search by exploiting subsumption. Subsumption occurs when a more general model or concept contains a more specific one. Subsumption applies in symptom clustering because the candidates entailed by one clustering may be a subset of those entailed by another clustering. Since both clusterings represent valid interpretations of the data, the more specific clustering is constrained unnecessarily by having its symptoms allocated to the "wrong" clusters. In the absence of any other information, we should therefore prefer the more general clustering, because it represents only constraints that are minimally necessary.

We say that clustering C_1 *subsumes* C_2 if there exists a complete matching between their causal intersections under the superset relation, that is, if each causal intersection in C_1 is a superset of some corresponding causal intersection in C_2. Clusterings C_1 and C_2 are *equigeneral* if there exists a complete matching under set equality. Finally, C_1 *properly subsumes* C_2 if C_1 subsumes C_2 but they are not equigeneral. These definitions lead to the subsumption criterion of generality: A clustering is *general* for a given case if no other valid clustering properly subsumes it. In the geometric interpretation given above, a clustering is general if its outer boundary is not enclosed by any other clustering.

This definition of generality can be computed readily using a process of *symptom rearrangement*. In symptom rearrangement, if a clustering is not already general, symptoms are moved between clusters until it is general. The conditions for moving a symptom between clusters can be expressed in terms of the following concepts. We say that a symptom s *constrains* a cluster c if it directly reduces its causal intersection, that is, if the possible causes for s do not subsume the causal intersection of $(c - \{s\})$:

$$\text{Constrains}(s,c) \iff \text{Causes}(s) \not\supseteq \text{Int}(c - \{s\})$$

A contrary notion is covering. We say that a symptom s *covers* a cluster c if its possible causes subsume the causal intersection of that cluster:

$$\text{Covers}(s,c) \iff \text{Causes}(s) \supseteq \text{Int}(c)$$

Finally, a symptom s in cluster c_1 is *movable* to another cluster c_2 if it constrains c_1 and covers c_2. (For definitional purposes, a symptom cannot move out of a singleton cluster.)

The key to symptom rearrangement is this theorem: A cluster is general if and only if all of its symptoms are not movable. This theorem yields a procedure to turn nongeneral clusterings into general ones: Simply move all movable symptoms until generality is obtained. This process must terminate because each movement of s from c_1 to c_2 has no effect on the causal intersection of c_2 but enlarges the causal intersection of c_1. At some point, the growth of causal intersections must halt.

For example, consider the clustering (AB)(CD) which has causal intersections $\{d_1\}$ and $\{d_1 d_3\}$. This is not general because A is movable from c_1 to c_2 and D is movable from c_2 to c_1. If we move A forward, we get (B)(ACD), which is general (but not minimal). If we move D backward, we get (ABD)(C), which is also general (and minimal). This example reveals that the rearrangement procedure is nondeterministic, a fact we will account for when we devise an algorithm to explore the search space.

Spanning

Spanning attempts to find a representative group of solutions by eliminating those that are redundant. Redundancy occurs in symptom clustering because some symptoms may be placed in several clusters without affecting the set of candidates entailed. Whenever the causes for a symptom are broad enough to covers more than one cluster, that symptom can be placed arbitrarily; each placement results in an equigeneral clustering. This redundancy is especially undesirable because broad symptoms have little discriminatory power. We seek to avoid this redundancy by generating only one clustering from each class of equigeneral clusterings. This yields a *spanning set* of general clusterings.

For example, suppose symptom E has possible causes $\langle d_1 d_3 d_4 d_5 \rangle$. Consider clustering (AD)(BC), which has causal intersections of $\{d_1 d_3 d_4\}$ and $\{d_1 d_5\}$. Symptom E covers either cluster, so that (ADE)(BC) and (AD)(BCE) are equigeneral. A spanning set would include either (ADE)(BC) or (AD)(BCE) but not both.

Problem-Reduction Search

The criteria given above define a set of clusterings that we would like to generate, namely a spanning set of general, minimal clusterings for a given case. We now present an algorithm that generates these clusterings. This algorithm is called problem-reduction search because it is a state-space search with each node of the search tree being one possible symptom clustering. As we mentioned before, a symptom clustering represents a problem-reduction, so this algorithm combines the features of both the problem-reduction and the state-space representations.

The algorithm processes one symptom in the case at a time. The first symptom can be clustered in only one way, a single cluster, yielding a frontier with a single clustering. The clusterings in each frontier are then expanded by another symptom from the case until all symptoms have been processed. Expansion consists of three steps: allocating a symptom to a cluster, rearranging the symptoms to achieve generality, and pruning nonminimal and equigeneral clusterings.

Allocating Symptoms

Symptom s may be allocated to a cluster in four ways:

Covering: Put s into a cluster c that it covers. Precondition: Covers(s,c).

Restricting: Put s into a cluster c that it would constrain. Precondition: Constrains(s,c).

Adding: Add a cluster consisting only of s. Precondition: Causes$(s) - \bigcup_c \text{Int}(c) \neq \emptyset$.

Extracting: Add a cluster consisting of s and a symptom s' already in some cluster c' that would constrain it. Precondition: Constrains$(s', c') \wedge$ Constrains$(s', \{s\})$.

Only some of these allocations need be performed for a given symptom and clustering. If some covering allocation is possible, the allocation process is finished. Else, all restrictions, additions, and extractions meeting the preconditions are invoked.

For example, Figure 3 shows problem-reduction search for symptoms A, B, C, and D in our running example. All four types of allocation, shown on each arc, are illustrated for symptom D. Node (ABCD) comes from covering (ABC); (AD)(BC) from restricting (A)(BC); (A)(BC)(D) from adding to (A)(BC); and (C)(B)(AD) from extracting A from (AC)(B).

Rearranging Symptoms

After allocating symptoms, they may need to be rearranged. However, as we mentioned previously, rearrangement can be nondeterministic. This presents problems in creating a spanning set of clusterings. Two sibling clusterings in the same frontier could rearrange to form the same clustering, thereby duplicating one clustering while failing to generate another one. We can eliminate this nondeterminism and create a spanning set by specifying exactly how symptoms should be rearranged. These techniques are called incremental and coordinated rearrangement.

Incremental rearrangement assumes that a clustering is general before allocating a new symptom and that nongenerality may be introduced during allocation. The specific source of nongenerality can then be repaired. Nongenerality can be introduced in three ways: (1) Reducing a causal intersection for cluster c enables symptoms in other clusters to move into c. (2) Adding a cluster c enables symptoms in other clusters to move into it. (3) Enlarging a causal intersection for c enables to symptoms in c to move to other clusters.

Thus, allocating a symptom can introduce nongenerality as follows: Covering has no effect; restricting or adding a cluster enables that cluster to import symptoms from other clusters; and extracting a symptom from cluster c' to create cluster c enables c' to export symptoms and c to import symptoms. We must also consider the secondary effects of moving symptoms. Moving a symptom from c_1 to c_2 enlarges the causal intersection of c_1, enabling c_1 to export symptoms, but has no effect on c_2.

Incremental rearrangement is performed in two stages: an importing stage and exporting stage. Suppose that cluster c has been restricted, added, or newly created by extraction. In the importing stage, c imports all symptoms that are eligible to move to it. Then in the exporting stage, any cluster that has exported a symptom exports any other symptom that can now move out. For extraction, the cluster with the extracted symptom originally also counts as an exporting cluster.

Figure 3 illustrates an example of rearrangement, indicated by the starred arc. Symptom D initially restricts clustering (AC)(B) to give (AC)(BD). Incremental rearrangement imports symptom A to the restricted cluster, giving (C)(ABD). In the exporting stage, cluster (C) has no symptoms to export, so rearrangement is complete.

Incremental rearrangement handles most cases of nondeterminism. Nevertheless, the exporting stage may still be nondeterministic, because exporting from c_1 to c_2 may preclude an export from c_3 to c_1. We say that these two movements *conflict*. Movement conflicts can be resolved by a coordinated rearrangement strategy. Coordinated rearrangement assumes that clusters are maintained in order of their creation. When two movements conflict, we favor a forward movements (from earlier clusters to later ones) over backward ones. Thus, in our example, if c_1 is before c_3, then movement $c_3 \rightarrow c_1$ is backward, and $c_1 \rightarrow c_2$ takes precedence. If c_3 is before c_1, then conflict can be avoided by performing $c_3 \rightarrow c_1$ before $c_1 \rightarrow c_2$.

This bias for forward movements depends on other parts of the search tree to explore missed opportunities that would have been generated by backward movements. To avoid foreclosing these opportunities prematurely, we also need a least-commitment strategy, whereby symptoms are moved to the earliest cluster possible, if there is more than one. Covering allocations must also adhere to the least-commitment strategy, so a symptom is allocated to the first cluster it covers, if any.

Coordinated rearrangement can be implemented easily. In the exporting stage, export from clusters in forward chronological order. For each constraining symptom in an exporting cluster, move it to the first cluster that it covers, if any. This may mean that a symptom moves backward, but this backward movement will not conflict with any forward movements, since earlier clusters will have already exported.

Pruning

After a clustering has been expanded and rearranged if necessary, its differentials are computed using the exception filter procedure discussed previously. If any filter lacks a feasible subfilter or any differential becomes null, that clustering is pruned.

In addition to nonminimal clusterings, redundant equigeneral clusterings must also be pruned. This can be done by comparing clusterings pairwise to check for equigeneral clusterings. This comparison will also uncover any duplicate clusterings that can arise. For example, in Figure 3, the clustering (ABD)(C) is generated twice, so one duplicate can be removed.

Experimental Results

Candidate generation and symptom clustering algorithms were implemented in Common Lisp on a Symbolics Lisp Machine and applied to the INTERNIST/QMR knowledge base [3]. This knowledge base contains 600 diseases and 4300 symptoms, covering 70% of the knowledge in internal medicine. To

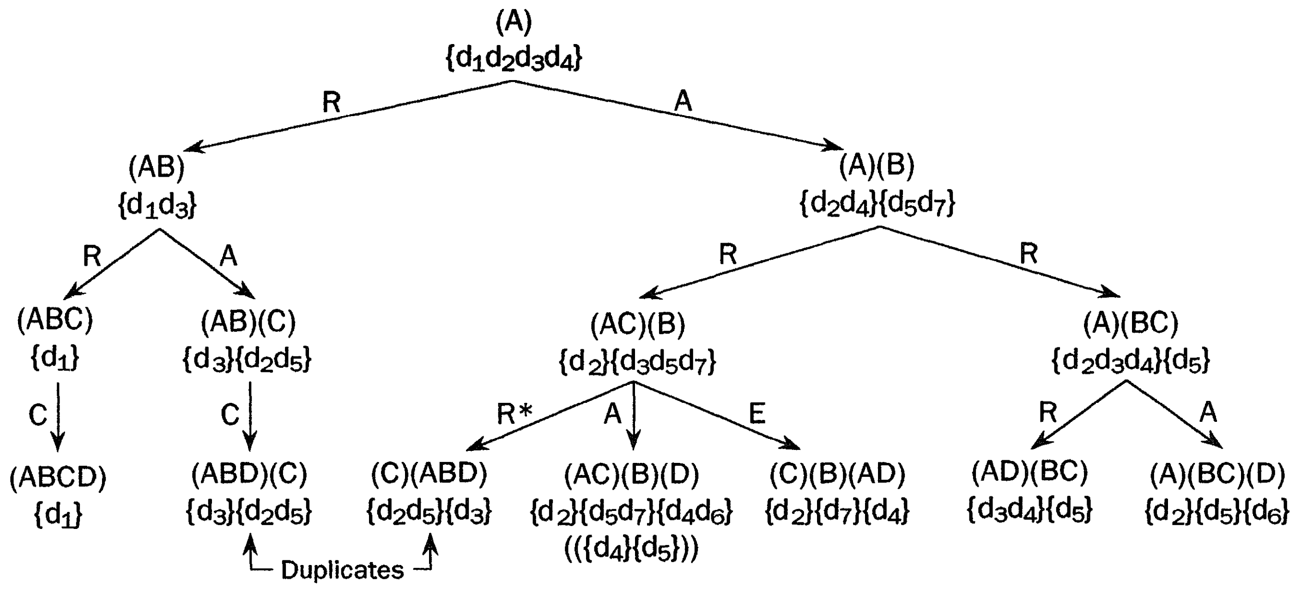

Figure 3: Problem-reduction search algorithm. The four symptoms, A, B, C, and D, correspond to the example given in the text. Each node shows a clustering and its differentials, along with any remaining exception filter. Arcs are labeled with the type of allocation: Covering, Restricting, Adding, or Extracting. A star indicates that rearrangement was performed.

control for effects of case variation and symptom ordering, we decided to analyze a single disease profile in depth. Prerenal azotemia was chosen because it generated relatively few symptoms (14, excluding age and sex), and these triggered various numbers of possible causes. The sets of possible causes ranged in size from 2 to 76, with a median of 29. The union of possible causes contained 147 diseases, meaning that a subgraph of 147 diseases and 14 symptoms was selected for this experiment.

Ten test cases were generated stochastically, by randomly selecting from the 14 symptoms for prerenal azotemia. These test cases contained 7–12 symptoms each. The probability of selecting a symptom depended on its frequency value: a frequency value of 1 resulted in a 3% chance of selection; 2, 20%; 3, 50%; 4, 80%; and 5, 98%. Contextual symptoms of age and sex were excluded, since the knowledge base lists all 600 disease as possible "causes", which would have severely penalized the candidate generation algorithm. However, we made no other checks for medical accuracy. In particular, some test cases contained two values for the same test, such as a blood urea nitrogen level of both 30–59 and 60–100 mg/dl, or a creatinine level of both 1.5–2.9 and 3–10 mg/dl. A reasonable interpretation of these readings is the common situation of testing a patient multiple times.

Solving each of the ten test cases for minimal candidates, we obtain six categories of solutions, ranging from 27 to 39101 minimal candidates. Although cases were generated from a single disease, minimal candidate solutions contained 1 to 5 disorders, with the following distributions:

	Candidates by Cardinality					
Category	1	2	3	4	5	Total
A	1	1	25			27
B	1	0	39	33		73
C	1	0	11	100		112
D	1	0	6	165	165	337
E	1	11	85	435	15	547
F	1	21	769	8985	29325	39101

The number of minimal clustering solutions ranged from 3 to 10, with the following size distributions:

	Clusterings by Dimensions					
Category	1	2	3	4	5	Total
A	1	1	1			3
B	1	0	3	1		5
C	1	0	2	1		4
D	1	0	1	3	1	6
E	1	2	2	1	1	7
F	1	1	3	4	1	10

We generated ten runs for each case by randomly permuting the order of the symptoms, and running times and search tree sizes were measured for each run. Any candidate generation run requiring more than 10^5 seconds (27.8 hours) was terminated. This eliminated 3 of the 40 runs in category A, 6 of the 20 runs in D, and all 10 runs in F. These cases are not shown on our graphs, but were solved by the symptom clustering algorithm in an average of 161 seconds. Figure 4 compares run times and search tree sizes.

We see that symptom clustering is substantially faster and more space-efficient than candidate generation. The roughly linear relationship on a log-linear scale indicates a near-exponential increase in performance, agreeing with the theoretically predicted re-

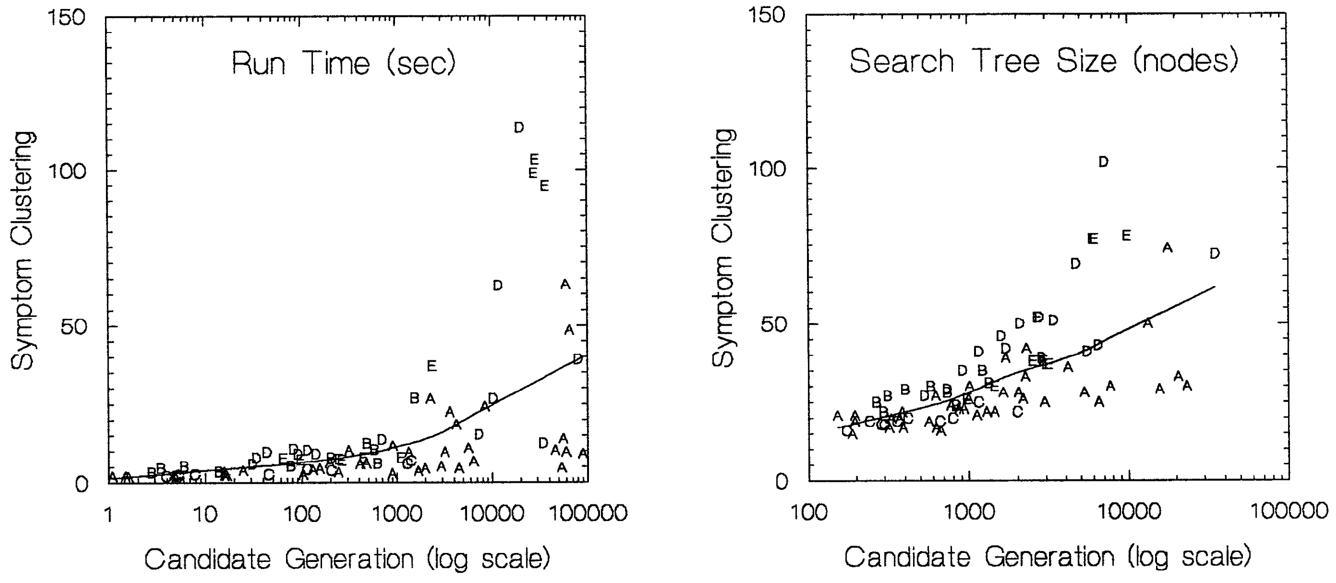

Figure 4: Comparison of candidate generation and symptom clustering for run times and search tree size. Search tree size is the total nodes over all frontiers after pruning. Letters correspond to the case category. Lines represent locally weighted scatterplot smoothing.

sults. As the problems become more difficult, however, these savings diminish somewhat, perhaps indicating the overhead of symptom clustering or other sources of computational complexity.

Conclusions and Further Research

This paper shifts the problem representation for multidisorder diagnosis from candidates to symptom clusterings. It establishes satisfaction and entailment relations between candidates and symptom clusterings, and defines criteria of minimality, generality, and spanning. The paper then devises an efficient algorithm to compute all symptom clusterings satisfying these criteria. Finally, the theoretical work is supplemented with empirical results that suggest a near-exponential increase in performance for symptom clustering.

This work presents several opportunities for further research. It offers a new representation for exploring novel types of search strategies, probabilistic theories, and heuristic sources. In particular, symptom clustering could potentially exploit "structural heuristics"—information about plausible groupings of evidence—which constitute a promising source of domain-specific knowledge.

Other fields of artificial intelligence besides diagnosis might benefit from the concepts here. For instance, previous research on learning and discovery systems [2] has also investigated clustering representations. Multidisorder diagnosis is closely related to the problems of multiple constraint satisfaction, conjunctive-goal planning and truth maintenance. Thus, some analogous notion of "constraint clustering" or "goal clustering" might be used effectively for these problems. Artificial intelligence deals primarily with problems that are ill-structured [7]. This paper suggests, at least for the domain of diagnostic problem solving, how the explicit assignment of structure might facilitate the solution of ill-structured problems.

References

[1] J. de Kleer and B. C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32:97–130, 1987.

[2] D. Fisher and P. Langley. Approaches to conceptual clustering. In *Proceedings, Ninth International Joint Conference on Artificial Intelligence*, pages 691–697, 1985.

[3] R. A. Miller, M. A. McNeil, et al. The Internist-1/Quick Medical Reference project—status report. *Western Journal of Medicine*, 145:816–822, 1986.

[4] Y. Peng and J. A. Reggia. A probabilistic causal model for diagnostic problem solving. *IEEE Transactions on Systems, Man, and Cybernetics*, 17:146–162 and 395–406, 1987.

[5] J. A. Reggia, D. S. Nau, and P. Y. Wang. Diagnostic expert systems based on a set covering model. *Intl. Journal of Man-Machine Studies*, 19:437–460, 1983.

[6] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–96, 1987.

[7] H. A. Simon. The structure of ill-structured problems. *Artificial Intelligence*, 4:181–201, 1973.

[8] T. D. Wu. Symptom clustering and syndromic knowledge in diagnostic problem solving. In *Proceedings, Thirteenth Symposium on Computer Applications in Medical Care*, pages 45–49, 1989.