A Mediated Communications Center

V.S. Subrahmanian, John R. Benton
University of Maryland
College Park, Maryland 20742

Sam Chamberlain
U.S. Army Research Laboratory
Adelphi, MD

Timothy J. Rogers
University of Maryland
College Park, Maryland 20742

Gautam Thaker
Lockheed Martin ATL
Camden, NJ 08102

Charlene Todd
Motorola
Scottsdale, AZ 8525

Abstract

This paper presents a theoretical framework for solving problems involving constraints on multiple paths. Examples of constraints considered in the paper include requirements that vehicles on separate paths (1) maintain line-of-sight (2) maintain ability to communicate between the vehicles or (3) maintain a minimum and/or maximum separation at all times. These constraints are all expressed within the theoretical framework. Three algorithms for computing multiple paths that satisfy constraints on the vehicles traversing the paths are presented.

1 Introduction

In this paper, we combine research results from several areas of computer science and database research. There have been recent advances in deductive database technology and mediator frameworks [8] for easily integrating both heterogeneous sources of data and software systems into a coherent whole. HERMES (Heterogeneous Reasoning and Mediator System) [1, 7, 2, 4] is a system that has been developed at the University of Maryland to facilitate the development and rapid deployment of mediators for different kinds of applications. It uses the Hybrid Knowledge Base paradigm, due to Lu, Nerode and Subrahmanian [5] to provide deductive database support for multiple modes of reasoning and multiple types of data.

One of the many applications of HERMES has been to link HERMES to route planners [3] developed by the U.S. Army Topographic Engineering Center (TEC). Initially, HERMES was linked to a grid-level planner which allowed complex queries such as: “Find the best path (shortest time) from the given location to a facility at a different location within the given radius.” This particular query illustrates the power of HERMES. It requires accessing a relational database to get a list of facilities, acquiring the geographic coordinates from a quadtree-based program and then calling the route planner for each of the retrieved facilities to determine which one has the smallest traversal time. The implementation of the system did not permit constraints to be specified between two separate paths. The need to specify constraints between vehicles moving on separate paths was one of the principal motivators for the work reported in this paper.

A second route planner developed by the U.S. Army Topographic Engineering has a hierarchic structure with planning done at both the grid and graph levels [3]. It first computes an optimum path using A* and then successively computes a second and third path subject to the constraint that there be no overlap with previously computed paths. A modified version of the route planner was developed at TEC to use the A* algorithm to optimize the sum of the paths subject to the same constraint that paths have no overlap. An improvement that could be made to the multiple route planner would be to relax the restriction that paths can not overlap to one that vehicles on different paths can not share the same path at the same time. Also, this route planner does not provide the capability to specify, for example, that vehicle locations on two separate routes should always have a separation between five and ten miles.

In this paper, we first establish a theoretical framework for solving problems involving constraints on multiple paths and then present several algorithms for solving such problems. Specific problem domains will likely determine which algorithms are optimum for a given problem. A future paper will go into much greater detail about the use of mediators in solving these problems.

2 Basic Definitions

Throughout this section, we will assume the existence of an image represented by an arbitrary, but fixed, 2-dimensional grid $G$ or matrix of size $(M \times N)$ and a corresponding 2-dimensional grid $M$ of size $(M \times N)$ that represents map coordinates.
<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute Set</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elevation</td>
<td>all non-negative reals</td>
<td>in some standard units (e.g., meters) above ground level</td>
</tr>
<tr>
<td>Jamming</td>
<td>all non-negative reals</td>
<td>0 denotes no jamming by enemy, 1 denotes full jamming by enemy</td>
</tr>
<tr>
<td>Slope</td>
<td>all non-negative</td>
<td>in percent slope</td>
</tr>
<tr>
<td>Soil</td>
<td>{ rocky, sandy, loamy, concrete, etc. }</td>
<td>partial list only</td>
</tr>
<tr>
<td>Trafficability for Humvee</td>
<td>all non-negative reals</td>
<td>in km per hour</td>
</tr>
<tr>
<td>Trafficability for Avenger</td>
<td>all non-negative reals</td>
<td>in km per hour</td>
</tr>
</tbody>
</table>

Figure 1: A table of attributes.

Definition 2.1 (Point) A point with respect to $G$ is any pair $(i, j)$ of integers such that $0 \leq i, j$ and $i < M$ and $j < N$. We will use the notation $PTS(G)$ to denote the set of all points associated with grid $G$.

Definition 2.2 (Scale Factor) Associated with any grid $G$, is a positive real number, $sc_G$, called the scale factor of $G$ that defines the ratio of distances between $G$ and $M$. This scale factor is valid when the size of $M$ is small enough that a flat earth can be assumed.

Definition 2.3 (Map Transformation Function) Associated with any grid, $G$, is a function, $MTF$, called the map transformation function that maps points in $G$ to points in $M$.

Definition 2.4 (Attribute) An attribute $A$ consists of a pair $(Attr\ Name, Attr\ Set)$ where Attr\ Name is a string called the name of the attribute, and Attr\ Set is a non-empty set, whose elements are called attribute values. If, in addition, Attr\ Set is partially ordered by some ordering $\leq_A$, then $A$ is said to be an ordered attribute.

When discussing attributes, we will often abuse notation, and use the attribute name to refer to the attribute itself. Furthermore, if $A$ is an attribute, we will often use the notation $Attr\ Set(A)$ to refer to the attribute set of $A$.

Example 2.1 Figure 2 shows some simple examples of attributes that may be used in the communications and terrain reasoning applications for the US Army.

Definition 2.5 (Map Schema) A map schema is triple $MS = (G, sc_G, A)$ where:

- $G$ is a grid of size $(M \times N)$ for some positive integers $M, N$;
- $sc_G$ is a scale factor;
- $A$ is a set of attributes.

Example 2.2 An example of a map schema consists of the triple:

1. $G$ is a grid of size $8 \times 8$, i.e., $M = 8 = N$;
2. $sc_G$ is the scale factor 100 (1 unit on the map denotes 100 units on the ground)
3. $A$ consists of the following three attributes elevation, jamming, and trafficability that we described earlier in Example 2.1.

Definition 2.6 (Map) A map $\mu$ over a map schema $MS$ is a mapping from $(PTS(G) \times A) \rightarrow \bigcup_{A \in A} Attr\ Set(A)$ such that:

$$(\forall (x, y) \in PTS(G))(\forall A \in A)\mu((x, y), A) \in Attr\ Set(A).$$

Definition 2.7 (Distance) Suppose $\mu$ is a map over map schema $MS = (G, sc_G, A)$ and $p_1 = (x_1, y_1), p_2 = (x_2, y_2)$ are two points w.r.t. this map.

- The map distance between $p_1, p_2$, denoted $md(p_1, p_2)$ is defined to be

$$md(p_1, p_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$  

- Assuming a flat earth, the horizontal distance between $p_1, p_2$, denoted $hd(p_1, p_2)$ is defined to be

$$hd(p_1, p_2) = sc_G \times \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$  

- Suppose $A \in A$ is an attribute. The $A$-distance weighted by factor $f$ (where $f$ is any non-negative real number) between $p_1$ and $p_2$, denoted $raw(A, f, p_1, p_2)$ is defined as:

$$raw(A, f, p_1, p_2) = \sqrt{f \times (\mu(p_1, A) - \mu(p_2, A))^2 + (x_1 - x_2)^2 + (y_1 - y_2)^2}.$$  

Figure 2: Table of attribute values
Example 2.3 Suppose we consider a map that shows the information of Table 2, which provides the formal description of some terrain map. Consider the points $p_1 = (0, 0)$ and $p_2 = (1, 1)$. Then:

$$md(p_1, p_2) = \sqrt{2},$$

$$hd(p_1, p_2) = 100\sqrt{2}. \quad (\text{Recall that } \text{csfg} = 100).$$

$$\text{raw(elevation)} \cdot (1, p_1, p_2) = \sqrt{1 \times (100 - 0)^2 + (0 - 0)^2 + (0 - 1)^2} = \sqrt{17}.$$ 

$$\text{raw(trafficability)} \cdot (1, p_1, p_2) = \sqrt{1 \times (50 - 55)^2 + (0 - 0)^2 + (0 - 1)^2} = \sqrt{26}.$$ 

Observation 2.1 The reader will immediately observe that Euclidean distance between two points in 3-space can be easily captured within the above framework by the expression:

$$\text{raw}(\text{elevation}, 1, p_1, p_2).$$

Throughout the rest of this paper, we will assume that $\mu$ is some arbitrary but fixed map schema over some arbitrary, but fixed map schema $\mathcal{M}_S$.

Definition 2.8 (Neighbor) Two points $p, q$ are said to be neighbors iff $md(p, q) \leq \sqrt{2}$.

For example, the point $(0, 3)$ has 5 neighbors viz. the points $(0, 4), (1, 4), (1, 3), (1, 2), (0, 2)$. Other points, such as $(4, 4)$ may have up to 8 neighbors. Similarly, the point $(0, 0)$ has 3 neighbors viz. $(0, 1), (1, 0), (1, 1)$.

Definition 2.9 (Contiguous Sequence of Points) $\sigma = (p_1, p_2, \ldots, p_n) \ (n \geq 1)$ is said to be a contiguous sequence of points (CSOP, for short) iff:

1. Each $p_i$ is a point w.r.t. map $\mu$ and
2. For all $1 \leq i \leq (n - 1)$, $p_i$ and $p_{i+1}$ are neighbors.

Example 2.4 For example, the following are continuous sequences of points from location $(3, 3)$ to $(6, 6)$.

$$\sigma_1 = (3, 3), (4, 4), (5, 5), (6, 6).$$

$$\sigma_2 = (3, 3), (3, 4), (3, 5), (3, 6), (3, 7), (4, 7), (5, 7), (6, 6)$$

$$\sigma_3 = (3, 3), (4, 3), (5, 3), (6, 4), (6, 5), (6, 6).$$

Definition 2.10 Single Point Attribute Requirement. A single point attribute requirement (SPAR for short) is a triple $(A, \text{op}, \delta)$ where:

1. $A$ is an attribute name and
2. $\delta$ is in AttrSet($A$) and
3. op is one of the operators in the set $\{=, \neq, \leq, \geq, >, <\}$

and
4. if $A$ is not an ordered attribute, then op $\in \{=, \neq\}$.

Example 2.5 Returning to our map $\mu$, we may write several SPARs such as:

- trafficability $\geq 35$;
- jamming $\leq 0.25$;

Intuitively, the only points $p$ that satisfy the first constraint are those for which $\mu(p, \text{trafficability}) \geq 35$. Likewise, the only points $p$ that satisfy the second constraint are those for which $\mu(p, \text{jamming}) \leq 0.25$.

The following definitions formalize what it means for a point and then a CSOP to satisfy a SPAR.

Definition 2.11 (Point Attribute Requirement) A point attribute requirement is a finite set of SPARs.

Definition 2.12 (Feasible CSOP) Suppose $\sigma = (p_1, p_2, \ldots, p_n) \ (n \geq 1)$ is a CSOP, and $(A, \text{op}, \delta)$ is a SPAR. We say that $\sigma$ is feasible with respect to $(A, \text{op}, \delta)$ iff for all $1 \leq i \leq n$.

$$\mu(p_i, A) \text{ op } \delta.$$ 

The CSOP $\sigma$ is said to be feasible with respect to a point attribute requirement $\mathcal{P}$ iff it is feasible with respect to each SPAR in $\mathcal{P}$.

Definition 2.13 (CSOP Problem) A CSOP-Problem $\mathcal{CP}$ is a triple, $\mathcal{CP} = (\text{orig}, \text{dest}, \text{cons})$ where:

- orig is a point called the origin;
- dest is a point called the destination;
- cons is a point attribute requirement (i.e. a finite set of SPARs).

Intuitively, the formal definition of a CSOP-problem given above captures, as special cases, many important problems of interest to the US Army. In other words, the concept of a CSOP-problem serves as an abstract, unifying model within which many different important Army problems may be addressed. Some of these include route planning, line of sight determination, and flight planning.

Definition 2.14 Suppose $\sigma = (p_1, p_2, \ldots, p_n) \ (n \geq 1)$ is a CSOP. Then $\tau = (t_1, t_2, \ldots, t_n) \ (n \geq 1)$ defines the time $t_i$ at which the point $p_i$ of $\sigma$ is occupied. There is a one-to-one mapping between the elements of $\sigma$ and $\tau$. The time between $t_i$ and $t_{i+1}$ is the interval during which the vehicle is in grid cell $p_i$.

Example 2.6 (Route Planning) In route planning, we are interested in finding a route from one point $p_1$ to another $p_2$. Suppose we have various terrain attributes called slope, vegetation, and soil. Suppose we are considering a vehicle $v$ that can only move when slope $\leq 10$ (degrees).
vegetation \neq forest, and soil is not loamy. We can capture the problem of planning a route for this vehicle through the triple:

\((\text{orig, dest, cons})\)

where \(\text{orig} = p_1, \text{dest} = p_2\), and \(\text{cons}\) contains the constraints:

\[
\begin{align*}
slope & \leq 10. \\
\text{vegetation} & \neq \text{forest.} \\
\text{soil} & \neq \text{loamy.}
\end{align*}
\]

In many applications, we want to solve not just one CSOP problem, but many CSOP problems simultaneously. For example:

1. We may wish to plan two routes \(r_1, r_2\) from point \(A\) to point \(B\), such that there are no more than 10 pairs of points along the two routes at which the vehicles (for whom the routes are being planned) will be less than one mile apart. If \(r_1 = p_{i_1}, \ldots, p_{i_m}, r_2 = q_{j_1}, \ldots, q_{j_n}\) and \(\tau_1 = t_1, \ldots, t_m\) then this requirement may be expressed as the constraint (in a language yet to be defined):

\[
10 \geq \text{card}\{i \mid 1 \leq i \leq \text{index}(\text{min}(\text{tm}, \text{tn}))\} \&
\text{interval}(\text{t}_i, \text{t}_j) \&(\text{dist}(\text{p}_i, \text{q}_j) \leq 5) \&
\text{can\_communicate}(\text{p}_i, \text{q}_j).
\]

where \text{interval} indicates there is some time \(t\) at which \(t_i\) is considered equal to \(t_j\) provided that \(t_i < t < t_{i+1}\) and \(t'_j < t' < t'_{j+1}\). Also, index \((\text{tm})\) returns the index \(m\). The predicate \text{can\_communicate}(\text{p}_i, \text{q}_j)\) may be expressed in terms of constraints involving jamming and elevation information.

2. We may likewise wish to plan two routes \(r_1, r_2\) from point \(A\) to point \(B\), such that there is never an interval of more than 5 miles at which the two vehicles can communicate. If \(r_1 = p_{i_1}, \ldots, p_{i_m}, r_2 = q_{j_1}, \ldots, q_{j_n}\) and \(\tau_1 = t_1, \ldots, t_m\) then this requirement may be expressed as a constraint (in a language yet to be defined):

\[
(\forall 1 \leq i \leq n)(\exists 1 \leq j \leq n)(\text{dist}(\text{p}_i, \text{q}_j) \leq 5) \&
\text{can\_communicate}(\text{p}_i, \text{q}_j) \&
\text{interval}(\text{t}_i, \text{t}_j).
\]

Definition 2.15 (Extended SPAR) An extended single point attribute requirement (eSPAR for short) is a triple \((A, op, \eta)\) where:

1. \(A\) is an attribute name and
2. \(\eta\) is either an attribute name or an element of the attribute value set of \(A\) and
3. If \(\eta\) is an attribute name, then \(\eta\) and \(A\) have the same attribute value sets and
4. \(op\) is one of the operators in the set \(\{, \neq, \leq, \geq, >, <\}\) and
5. If \(A\) is not an ordered attribute, then \(op \in \{, \neq\}.

Definition 2.16 (Binary CSOP Constraint) Suppose \(CP_1, CP_2\) are CSOP-problems, where \(CP_i = (\text{orig}, \text{dest}, \text{cons}_i)\). A binary-CSOP constraint is a quadruple of the form

\[
(CP_1, CP_2, \text{gap}, \delta, \text{econs})
\]

where (1) \(\text{gap}\) is a positive integer, (2) \(\delta\) is a positive integer and (3) \(\text{econs}\) is a finite set of extended SPARs.

Intuitively, suppose we have a binary-CSOP constraint of the form:

\[
(CP_1, CP_2, 50, 2, \text{econs})
\]

Suppose now that

\[
\sigma_1 = p_1, \ldots, p_m \& \sigma_2 = p'_1, \ldots, p'_m
\]

are solutions to the CSOP-problems \(CP_1, CP_2\), respectively. We say that the pair \(\sigma_1, \sigma_2\) satisfies the above binary-CSOP-constraint iff for all integers \(\text{gap} + \delta \leq i \leq \text{index}(\text{min}(\text{tm}, \text{tn}))\), there exists an integer \(j\) such that:

1. \(i = (\text{gap} + \delta) \leq j \leq i = (\text{gap} - \delta)\), and
2. there is a solution to the CSOP-problem \((p_j, p'_j, \text{econs})\).

More generally, if we ignore the instantiated binary-CSOP constraint \((CP_1, CP_2, 50, 2, CP)\), and consider instead, an arbitrary binary CSOP constraint \((\text{gap}, \delta, CP)\), then we say that the pair of solutions \(\sigma_1, \sigma_2\) satisfies \((\text{gap}, \delta, CP)\) iff for all integers \(\text{gap} + \delta \leq i \leq \text{min}(m, n)\), there exists an integer \(j\) such that:

1. \(i = (\text{gap} + \delta) \leq j \leq i = (\text{gap} - \delta)\), and
2. there is a solution to the CSOP-problem \((p_j, p'_j, \text{econs})\).

We need an additional definition before we consider the example below.

Definition 2.17 Time Synchronization CSOP Constraint. Let \(\text{Synch}(\sigma_1, \sigma_2)\) be the synchronization between \(\sigma_1, \sigma_2\) such that for all points \(p_i\) in \(\sigma_1\) at time \(t_i\), there is some corresponding \(q_j\) in \(\sigma_2\) at time \(t_j\) such that \(\text{interval}(t_i, t_j)\) is true.

Example 2.7 Suppose we consider the CSOPs \(\sigma_1, \sigma_2\) of Example 2.4. Suppose these two CSOPs are routes for two different Army columns moving on the ground, and we want to ensure that at any given point in time, at most 4 \pm 1 units of time have elapsed since the last two columns communicated with each other (“check in”). Let us further assume, for now, that communication merely means that the jamming value is less than 0.2 and the elevation value is less than 80. This can be captured as follows.
Consider now the binary-CSOP constraint:

\[(CP_1, CP_2, 4, 1, econs)\]

where `econs` consists of the two constraints: jamming $\geq 0.1$ and elevation $\leq 80$. Intuitively, the pair `synch(\sigma_1, \sigma_2)` is an acceptable pair of CSOP solutions iff at any given point $t$ in time, there is a time $t'$ such that $(4 - 1) \leq t' \leq (4 + 1)$ such that at time $(t - t')$, there is a CSOP solution $\sigma$ between points $t_1$ and $t'_1$ such that for each point in $\sigma$, the jamming value is less than 0.2.

**Definition 2.18 (Multi-Constraint CSOP)** A multi-constraint CSOP problem consists of a pair $(CP, B\!\!\!C)$ where:

- $CP$ is a set of CSOP problems and
- $B\!\!\!C$ is a set of binary CSOP constraints involving CSOP problems in CP.

**Definition 2.19 (Solution to Multi-Constraint CSOP)** A solution to a multi-CSOP problem $(CP, B\!\!\!C)$ is a mapping $\Lambda$ that assigns to each $CP \in CP$, a solution $\Lambda(CP)$ of CP such that for all binary-CSOP constraints

\[(CP_i, CP_j, gap, \delta, econs)\]

in $B\!\!\!C$, it is the case that the pair $\Lambda(CP_i), \Lambda(CP_j)$ of solutions associated with CSOP-problems $CP_i, CP_j$ satisfies the above binary CSOP constraint.

Below, we present a simple algorithm to find a CSOP connecting two points. Note that this algorithm does not necessarily find an optimal CSOP.

**Algorithm 1** `FindMultiCSOP(CP_1, CP_2, gap, \delta, econs);`

```
    done = false; i = 1
    while ~done do
        \{ \sigma_1 = solution to CP_1; \}
        while ~done do
            \{ \sigma_2 = solution to CP_2; \}
            \{ if (\sigma_1, \sigma_2) satisfies cons then done = true; \}
            \{ if done then return (\sigma_1, \sigma_2) and Halt \}
            \{ return "fail". \}
        end
    end
```

**Definition 2.20 (Optimal Multi-CSOP Solution)** Suppose $(CP, B\!\!\!C)$ is a Multi-constraint CSOP problem and $CP = \{CP_1, \ldots, CP_n\}$. A Multi-CSOP Objective function is an expression of the form:

\[
\min(\alpha_1 c_1 + \ldots + \alpha_n c_n)
\]

where $\alpha_1, \ldots, \alpha_n$ are non-negative integers, and $c_i$ is a variable. (Intuitively, the $c_i$'s are variables denoting the "cost" of a solution of constraint problem $CP_i$.)

It is easy to see that an algorithm such as the well known A* algorithm [6] can be used to enumerate solutions to a single CSOP. Furthermore, such algorithms can be used to enumerate such solutions in ascending order of cost. Let `FindCSOPSol` be any such algorithm and let `FindOptimalMultiCSOP()` be invoked with a number $n$ specifying how many solutions we would like. Then we may write an algorithm to compute optimal solutions of Multi-CSOP problems as follows:

**Algorithm 2**

```python
def FindOptimalMultiCSOP(CP_1, CP_2, gap, \delta, econs):
    done = false; i = 1
    while ~done do
        \{ \sigma_1 = FindCSOPSol(CP_1, i); \}
        while ~done && ~fail do
            \{ j = 1; \}
            \{ \sigma_2 = FindCSOPSol(CP_2, j); \}
            \{ if (\sigma_1, \sigma_2) satisfies cons then done = true \}
            \{ else j = j + 1; \}
        end
        \{ return "fail". \}
    end
```

Algorithm 2 assumes that the optimum solution to the multi-CSOP will always include the optimum CSOP. Reference [3] presents a counter-example. A different approach to the Multi-Constraint CSOP Problem is to generate the multi-path solution within the A* algorithm, thereby optimizing the sum of the paths subject to the constraints of the multi-constraint CSOP problem. In some problem domains, this approach may have an advantage over Algorithm 1 which finds solutions in ascending order of cost to a single CSOP problem and then searches to see if there is a set of solutions that meet the constraints of the multi-constraint CSOP problem.

Before we develop this algorithm, some definitions are required. Unlike Algorithm 2, each solution of the A* algorithm is for the multi-constraint CSOP problem and thus contains multiple CSOP’s. We allow an arbitrary number of solutions $CP_j \in CP$. An alternative solution will be explored in what we call an alternative world $k$. When an alternate world is created, all the data structures of the parent world are copied into the alternate world. Initialization of the parent world and conditions under which an alternate world is split off from the parent world are specified in the algorithm.
Any search node, $n_{ijk}$, is the $i$th node of the $j$th search path $P_j$ of the $k$th world. The number of search paths equals the number of paths specified in a given problem. The initial world is for $k$ equal to zero. We now define $M$ to be the number of $CP_j$s or Paths specified in the multi-constraint CSOP problem. $P_{jk} = (n_{0jk}, \ldots, n_{ijk}, \ldots)$, nodes of search tree in the ordered list, $SOS_j = [P_{0jk}, \ldots, P_{j-1k}, P_{jk}, \ldots, P_{M-1k}]$, $best(SOS_k) =$ lowest cost open node in any of the search trees of world $k$, $USOS_k =$ $[best(SOS_1), \ldots, best(SOS_k), \ldots]$. We can now write an algorithm to intelligently search for the optimum solution as follows:

Algorithm 3 is most likely to be useful within the context of software such as TEC's hierarchical route planner[3] in which the number of nodes in the search tree is much less than the number of nodes for a grid-based route planner covering the same physical area. The number of nodes that must be checked for constraint violation can be minimized by imposing a temporal constraint so that the node currently being expanded only has to be matched against other nodes within the same time interval.

In summary, we have developed a theoretical framework for multi-path route planning with arbitrary constraints on the computed paths. Algorithm 1 halts as soon as it finds two solutions that satisfy the constraints. There is no attempt at optimality. Algorithm 2 compares the optimal unconstrained solution to successively less optimal solutions until it finds a pair that satisfies the constraints. Algorithm 3 uses $A^*$ to get a simultaneously optimized solution to the multi-CSOP problem. The algorithm optimal for any given problem will likely depend on the problem domain.

References


[5] J. Lu, A. Nerode, and V.S. Subrahmanian, Hybrid knowledge bases, Univ. of Maryland, CS-TR-3037. Accepted for publication in IEEE Trans. on Knowledge and Data Engin.  


---

The views and documents contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied of the U.S. Army Research Laboratory or the U.S. Government.