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 facilites 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 heirarchic 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 to the restriction that paths can notoverlap 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.

*Prepared through collaborative participation in the Advanced Telecommunications & Information Distribution Research Program (ATIRP) Consortium sponsored by the U.S. Army Research Laboratories under the Federated Laboratory Program, Cooperative Research Agreement DAAL01-96-2-0002. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

2 Basic Definitions

Throughout this section, we will assume the existence of an image represented by an arbitrary, but fixed, 2-dimensional grid $\mathcal{G}$ or matrix of size $(M \times N)$ and a corresponding 2-dimensional grid $\mathcal{M}$ of size $(M \times N)$ that represents map coordinates.

Attribute Name	Attribute Set	Comments
Elevation	all non-negative reals	in some standard units (e.g. meters) above sea level
Jamming	all non-negative reals	0 denotes no jamming by enemy 1 denotes full jamming (by enemy)
Slope Soil	all non-negative { rocky, sandy, loamy, concrete, etc. }	in percent slope partial list only
Trafficability for Humvee	all non-negative reals	in km per hour
Trafficability for Avenger	all non-negative reals	in km per hour
...	...	...

Figure 1: A table of attributes.

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

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

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

Definition 2.4 (Attribute) *An* attribute *A consists of a pair $(AttrName, AttrSet)$ where $AttrName$ is a string called the name of the attribute, and $AttrSet$ is a non-empty set, whose elements are called attribute values. If, in addition, $AttrSet$ 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 $AttrSet(A)$ 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 $\mathcal{MS} = (\mathcal{G}, \mathsf{sc}_{\mathcal{G}}, \mathcal{A})$ where:*

Xcoord	Ycoord	Attribute	Value
0	0	elevation	100
0	0	jamming	0
0	0	trafficability	55
0	1	elevation	96
0	1	jamming	0
0	1	trafficability	50
...	...	...	...
7	7	elevation	86
7	7	jamming	0.20
7	7	trafficability	50

Figure 2: Table of attribute values

- *$\mathcal{G}$ is a grid of size $(M \times N)$ for some positive integers M, N;*
- *$\mathsf{sc}_{\mathcal{G}}$ is a scale factor;*
- *$\mathcal{A}$ is a set of attributes.*

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

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

Definition 2.6 (Map) *A* map *μ over a map schema $\mathcal{MS}$ is a mapping from $(\mathsf{PTS}(\mathcal{G}) \times \mathcal{A}) \to \bigcup_{A \in \mathcal{A}} AttrSet(A)$ such that:*

$$(\forall (x, y) \in \mathsf{PTS}(\mathcal{G}))(\forall A \in \mathcal{A})\mu((x, y), A) \in AttrSet(A).$$

Definition 2.7 (Distance) *Suppose μ is a map over map schema $\mathcal{MS} = (\mathcal{G}, \mathsf{sc}_{\mathcal{G}}, \mathcal{A})$ 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*
$$hd(p_1, p_2) = \mathsf{sc}_{\mathcal{G}} \times \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$
- *Suppose $A \in \mathcal{A}$ 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 $\mathsf{raw}(A, f, p_1, p_2)$ is defined as:*
$$\mathsf{raw}(A, f, p_1, p_2) = \sqrt{f \times (\mu(p_1, A) - \mu(p_2, A))^2 + (x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

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:*

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

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

$$\mathsf{raw}(elevation, 1, p_1, p_2) =$$

$$\sqrt{1 \times (100 - 96)^2 + (0 - 0)^2 + (0 - 1)^2} = \sqrt{17}.$$

$$\mathsf{raw}(trafficability, 1, p_1, p_2) =$$

$$\sqrt{1 \times (50 - 55)^2 + (0 - 0)^2 + (0 - 1)^2} = \sqrt{26}.$$

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:*

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

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

Definition 2.8 (Neighbor) *Two points* p, q *are said to be* neighbors *iff* $md(p_1, p_2) \leq \sqrt{2}$*.*

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 upto 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* μ *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, op, δ) *where:*

1. *A is an attribute name and*
2. δ *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* μ*, we may write several SPARs such as:*

- *trafficability* ≥ 35*;*
- *jamming* ≤ 0.25*;*

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

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, op, δ) *is a SPAR. We say that* σ *is* feasible *with respect to* (A, op, δ) *iff for all* $1 \leq i \leq n$*,*

$$\mu(p_i, A) \quad op \quad \delta.$$

The CSOP σ *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,* $\mathcal{CP} = (\mathsf{orig}, \mathsf{dest}, \mathsf{cons})$ *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* σ *is occupied. There is a one-to-one mapping between the elements of* σ *and* τ*. 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` ≤ 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:*

$$(\mathsf{orig}, \mathsf{dest}, \mathsf{cons})$$

where $\mathsf{orig} = p_1$, $\mathsf{dest} = p_2$, *and* cons *contains the constraints:*

$$\begin{array}{rcl} \texttt{slope} & \leq & 10. \\ \texttt{vegetation} & \neq & \mathit{forest}. \\ \texttt{soil} & \neq & \mathit{loamy}. \end{array}$$

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 $r_1 = p_1, \ldots, p_m$, $\tau_1 = t_1, \ldots, t'_m$, $r_2 = q_1, \ldots, q_n$ and $\tau_2 = t_1, \ldots, t_n$ then this requirement may be expressed as the constraint (in a language yet to be defined):

   $$10 \geq \mathrm{card}\{i \mid 1 \leq i \leq \mathrm{index}(min(t_m, t'_n))\} \;\&$$

   $$(interval_=(t_i, t'_j)) \,\&(dist(p_i, q_j) \leq 5) \;\&$$

   $$\mathrm{can\_communicate}(p_i, q_j).$$

   where $interval_=$ indicates there is some time t at which t_i is considered equal to t'_j provided that $t_i \leq t < t_{i+1}$ and $t'_i \leq t' < t'_{i+1}$. Also, $\mathrm{index}(t_m)$ returns the index m. The predicate $\texttt{can\_communicate}(\mathtt{p_i}, \mathtt{q_j})$ 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 $r_1 = p_1, \ldots, p_m$, $\tau_1 = t_1, \ldots, t_m$, $r_2 = q_1, \ldots, q_n$ and $\tau_2 = t'_1, \ldots, t'_n$ then this requirement may be expressed as a constraint (in a language yet to be defined):

   $$(\forall 1 \leq i \leq n)(\exists 1 \leq j \leq n)\,(\mathrm{dist}(p_i - q_j) \leq 5)\;\&$$

   $$\mathrm{can\_communicate}(p_i, q_j)\&$$

   $$interval_=(t_i, t'_j).$$

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

1. *A is an attribute name and*
2. *η is either an attribute name or an element of the attribute value set of A and*
3. *If η is an attribute name, then η 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* $\mathcal{CP}_1, \mathcal{CP}_2$ *are CSOP-problems, where* $\mathcal{CP}_i = (\mathsf{orig}_i, \mathsf{dest}_i, \mathsf{cons}_i)$. *A* binary-CSOP constraint *is a quadruple of the form*

$$(\mathcal{CP}_1, \mathcal{CP}_2, gap, \delta, \mathsf{econs})$$

where (1) gap is a positive integer, (2) δ is a positive integer and (3) econs *is a finite set of extended SPARs.*

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

$$(\mathcal{CP}_1, \mathcal{CP}_2, 50, 2, \mathsf{econs}).$$

Suppose now that

$$\sigma_1 = p_1, \ldots, p_n. \;\;\&\;\; \sigma_2 = p'_1, \ldots, p'_m$$

are solutions to the CSOP-problems $\mathcal{CP}_1, \mathcal{CP}_2$, respectively. We say that the pair σ_1, σ_2 *satisfies* the above binary CSOP-constraint iff for all integers $gap + \delta \leq i \leq \mathrm{index}(\min(t_m, t'_n))$, there exists an integer j such that:

1. $i - (50 + 2) \leq j \leq i - (50 - 2)$, and
2. there is a solution to the CSOP-problem $(p_j, p'_j, \mathsf{econs})$.

More generally, if we ignore the instantiated binary-CSOP constraint $(\mathcal{CP}_1, \mathcal{CP}_2, 50, 2, \mathcal{CP})$, and consider instead, an arbitrary binary CSOP constraint $(gap, \delta, \mathcal{CP})$, then we say that the pair of solutions σ_1, σ_2 satisfies $(gap, \delta, \mathcal{CP})$ iff for all integers $gap + \delta \leq i \leq \min(m, n)$, there exists an integer j such that:

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

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

Definition 2.17 Time Synchronization CSOP Constraint. *Let $Synch(\sigma_1, \sigma_2)$ be the synchronization between σ_1 and σ_2 such that for all points p_i in σ_1 at time t_i, there is some corresponding q_j in σ_2 at time t_j such that $interval_=(t_i, t'_j)$ is true.*

Example 2.7 *Suppose we consider the CSOPs σ_1 and σ_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 ± 1 units of time have elapsed since the two last communicated with each other ("check in"). Let us further asssume, for now, that communicability 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.*

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

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

Recall that these CSOPs were used in Example 2.4. Consider now the binary-CSOP constraint:

$$(\mathcal{CP}_1, \mathcal{CP}_2, 4, 1, \mathsf{econs}_1)$$

where econs_1 consists of the two constraints: `jamming` ≥ 0.1. and `elevation` ≤ 80. *Intuitively, the pair synch*(σ_1, σ_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* σ *between points* t_1 *and* t'_1 *such that for each point in* σ*, the jamming value is less than* 0.2.

Definition 2.18 (Multi-Constraint CSOP) *A* multi-constraint CSOP problem *consists of a pair* $(\mathbf{CP}, \mathcal{BC})$ *where:*

- $\mathbf{CP}$ *is a set of CSOP problems and*
- $\mathcal{BC}$ *is a set of binary CSOP constraints involving CSOP problems in* $\mathbf{CP}$.

Definition 2.19 (Solution to Multi-Constraint CSOP) *A* solution to a multi-CSOP problem $(\mathbf{CP}, \mathcal{BC})$ *is a mapping* Λ *that assigns to each* $\mathcal{CP} \in \mathbf{CP}$*, a solution* $\Lambda(\mathcal{CP})$ *of* $\mathcal{CP}$ *such that for all binary-CSOP constraints*

$$(\mathcal{CP}_i, \mathcal{CP}_j, gap, \delta, \mathsf{econs})$$

in $\mathcal{BC}$*, it is the case that the pair* $\Lambda(\mathcal{CP}_i), \Lambda(\mathcal{CP}_j)$ *of solutions associated with CSOP-problems* $\mathcal{CP}_i, \mathcal{CP}_j$ *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*$(\mathcal{CP}_1, \mathcal{CP}_2, gap, \delta, \mathsf{econs})$*;*

```
done = false;
while ¬done do
{
  σ1 = solution to CP1;
  while ¬done do
  {
    σ2 = solution to CP2;
    if (σ1, σ2) satisfies cons then done= true;
  }; (⋆ end inner while ⋆)
  if done then return (σ1, σ2) and Halt
} (⋆ end outer while ⋆)
Return "fail".
end
```

Definition 2.20 (Optimal Multi-CSOP Solution) *Suppose* $(\mathbf{CP}, \mathcal{BC})$ *is a Multi-constraint CSOP problem and* $\mathbf{CP} = \{\mathcal{CP}_1, \ldots, \mathcal{CP}_n\}$*. A* Multi-CSOP Objective function *is an expression of the form:*

$$\mathbf{min}(\alpha_1 c_1 + \ldots \alpha_n c_n)$$

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* $\mathcal{CP}_i$*).*

It is easy to see that an algorithm such as the well known $A^\star$ 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 `FindCSOPSol` be invocable with a number n specifying how many solutions we would like. Then we may write an algorithm to compute optimal solutions of MultiCSOP problems as follows:

Algorithm 2
FindOptimalMultiCSOP$(\mathcal{CP}_1, \mathcal{CP}_2, gap, \delta, \mathsf{econs})$*;*

```
done = false; i = 1
while ¬done do
{
  σ1 = FindCSOPSol(CP1, i);
  while ¬done & ¬fail do
  {
    j = 1;
    σ2 = FindCSOPSol(CP1, j);
    if (σ1, σ2) satisfies cons then done= true
    else j = j + 1;
  }; (⋆ end inner while ⋆)
  if done then return (σ1, σ2) and  bf Halt
} (⋆ end outer while ⋆)
Return "fail".
end
```

Algorithm 2 assumes that the optimum solution to the mult-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 $A^\star$ 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 2 which finds solutions in ascending cost of solution to a single CSOP problem and then searches to see if there is a set of solutution 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 $\mathcal{CP}_j \in \mathbf{CP}$. 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.

Algorithm 3
FindMultiOptimalCSOP($\mathcal{CP}_1, \mathcal{CP}_2, gap, \delta,$ econs);

($\star$ *INITIALIZATION* $\star$)
current_nbr_worlds = *1*;
for $j = 0$ *to* $M-1$, $P_{0j} = (n_{0j0})$;
$\mathcal{SOS}_0 = f[P_{00}, \ldots P_{j0}, \ldots P_{(M-1)0}]$;
$\mathcal{USOS} = ordered\_list(n_{000})$;
$answerlist_0 = ()$;
done = *false*;
($\star$ *Main loop of program* $\star$)
while $\neg$*done* **do** {
pull lowest cost node, n_{ijk}, *from* $\mathcal{USOS}$;
if $\mathcal{USOS} = null$; **return** *done* = *false*;
if *node* n_{ijk} *is marked for Alternate World exploration*
incr (current_nbr_worlds); $k' = current\_nbr\_worlds$;
copy $\mathcal{SOS}_k$ *to* $\mathcal{SOS}_{k'}$
$answerlist_{k'} = answerlist_k$; $k = k'$;
loop *over i' for daughter nodes of* n_{ijk} *in world k* {
add nodes $n_{i'jk}$ *to* $\mathcal{SOS}_k$ *in sorted order by* $A^\star$ *cost*;
mark $n_{i'jk}$ *for alternate world exploration*
if constraint violation;
mark $n_{i'jk}$ *pruned if ordinary* $A^\star$ *pruning*
otherwise *mark* $n_{i'jk}$ *open*;
if $n_{i'jk} = destination\_node$ {
add $n_{i'jk}$ *to* $answerlist_k$;
remove P_{jk}*from* $\mathcal{SOS}_k$;
($\star$ *the jth path or CP has been found* $\star$)
if $length(answerlist_k) = M$
then *done* = *true*;
return $answerlist_k$};
else *insert lowest-cost open node of* $\mathcal{SOS}_k$
onto $\mathcal{USOS}$ *in sorted order*;}}
loop *over j for* CP_j
loop *over i for nodes* n_{ijk} *in* $answerlist_k$
generate $\mathcal{CP}_i$ *by tracing* n_{ijk} *back to start node*
using parent pointers
end

Any seach node, n_{ijk}, is the *ith* node of the *jth* search path P_j of the *kth* 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 $\mathcal{CP}_j$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, $\mathcal{SOS}_k$=$[P_{0k}, \ldots P_{jk}, \ldots P_{(M-1)k}]$, $best(\mathcal{SOS}_k)$=*lowest* cost open node in any of the search trees of world k, $\mathcal{USOS}$=$[best(\mathcal{SOS}_1), \ldots best(\mathcal{SOS}_k), \ldots]$. 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 heirarchical 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 $A^\star$ 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

[1] S. Adalı R. Emery, J. Lu, A. Rajput, T.J. Rogers, R. Ross, and V.S. Subrahmanian, *HERMES: A heterogeneous reasoning and mediator system*, draft manuscript.

[2] S. Adalı and V.S. Subrahmanian. *Amalgamating knowledge bases, II: algorithms, data structures and query processing*, Univ. of Maryland, **CS-TR-3124** (1993). Accepted for publication in: Intl. J. of Intell. Coop. Info. Sys.

[3] J. Benton, S.S. Iyengar, Weian Deng, N. Brener and V.S. Subrahmanian, *Tactical Route Planning: New Algorithms for Decomposing the Map*, Int'l J. for Artificial Intelligence Tools, **5** 1996 199–218.

[4] M. Kifer and V. S. Subrahmanian. (1992) *Theory of Generalized Annotated Logic Programming and its Applications*, Journal of Logic Programming, Vol. 12, 4, pps 335–368, 1992.

[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 Engrng.

[6] Nil Nilsson, *Searching problem-solving and game-playing trees for minimal cost solutions,* Information Processing, ed. A. J.H. Morrell, 1556–62.

[7] V.S. Subrahmanian, *Amalgamating knowledge bases*, ACM Trans. on Database Sys. **19**, 1994 291–331.

[8] G. Wiederhold. *Intelligent integration of information,* Proc. 1993 ACM SIGMOD Conf. on Management of Data, 434–437.

[1] 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.