10th International Command and Control Research and Technology Symposium
The Future of C2

1. Title: **Time-Sensitive Planning Using Point-Interval Logic**[•]

2. Topic: Assessment, Tools, and Metrics

3. Authors: **Mashhood Ishaque*** **Abbas K. Zaidi**

4. Point of Contact: **Mashhood Ishaque**

5. Organization: System Architectures Lab, George Mason University

6. Address: System Architectures Laboratory
MSN 4B5
George Mason University
Fairfax, VA - 22030
703.993.1725 (v)
703.993.1706 (f)

7. Email: mishaque@gmu.edu

Student Paper

[•] The work was carried out with support provided by the Air Force Office of Scientific Research under contract numbers FA9550-05-1-0106.

Time-Sensitive Planning Using Point-Interval Logic[•]

Mashhood Ishaque and Abbas K. Zaidi

System Architectures Lab
George Mason University
Fairfax, VA – 22030
{mishaque,szaidi2}@gmu.edu

Abstract

This paper presents an application of Point-Interval logic (PIL) for the problem of planning time-sensitive aspects of a mission. The logic incorporates both point and interval structures of time associated with mission activities and also provides for qualitative and quantitative descriptions of temporal requirements among mission activities. An algorithm is presented that extends the inference engine of PIL for mission planning application. The planning approach is demonstrated with the help of a small illustrative mission planning problem with non-trivial temporal constraints. The temporal formalism with its descriptive input language, an inference engine for reasoning about generated plans, and the planning algorithm has been implemented in the form of a software tool.

1 Introduction

The time-sensitive aspects of a mission require a planner to sequence time intervals (or points) associated with mission activities, or services required, without violating any of the system specification, given a priori as part of a doctrine and/or as an outcome of a mission assessment process. This makes mission planning a constraints satisfaction problem in terms of temporal constraints between the mission activities together with resource availability constraints. In most real world situations, the dynamic nature of a domain may require revising a produced plan in terms of new added constraints and/or modified old requirements. A temporal constraint satisfaction formalism, used for mission planning, must therefore be able to (a) generate a *feasible* plan satisfying all the (temporal) constraints, or report infeasibilities present in the specifications; (b) allow a plan to be revised with minimal perturbation; (c) compare the generated alternate plans on the basis of some pre-defined performance criteria; and (d) provide a graphical representation of system specifications, and the generated plans, preferably capable of modeling information at different levels of abstraction.

The paper employs a point-interval formalism PIL (Zaidi and Levis 2001), which is an extension of Allen's interval logic (Allen, 1983), for constructing such a planner. PIL is a tractable point and interval formalism which handles both qualitative and quantitative temporal constraints. It provides the capability of revising a *feasible* temporal system by making *minimum* change to the system without violating the existing constraints. It is a graph-based approach with a specialized graph representation, called Point Graph (PG), to model the temporal statements/constraints. The graphical properties of the point graph help decide the *feasibility* of the system in polynomial time and require linear searches to identify feasible relations between points/intervals. The contribution in this paper offers an enhancement that extends the capabilities of the PIL based planner beyond

---

[•] The work was carried out with support provided by the Air Force Office of Scientific Research under contract numbers FA9550-05-1-0106.

that of Zaidi and Wagenhals' [2005], in particular, and of existing graph-based operations research formalisms, e.g. CPM and PERT (Moder and Philips 1970), in general. The integration of PIL's language, knowledge representation mechanism, inference engine, and the proposed planning algorithm results in a planning tool with an expressive language for specification of temporal relations among points and intervals representing activities in a mission and an ability to generate alternate solutions to perform what-if analysis. It is also superior to mathematical programming approaches to planning for it does not require a plan to be regenerated from scratch in case of any changes (revisions) to mission specifications.

The paper concludes with a fictitious but real world example to illustrate how the approach presented could be applied to military planning and execution problems. The facts and constraints that apply to the mission at hand are input using language of PIL. The mission constraints are then converted into corresponding point graph representation and checked for any inconsistencies or temporal anomalies. Once a consistent point graph representation of the mission requirements has been built, it can be used to capture useful parameters associated with mission activities. Some of these parameters which provide useful insight for the mission planner are: earliest occurrence time of an activity, late occurrence time, latest occurrence time, whether or not an activity is critical, and the values of the various time slacks available to the planner, e.g., total float, free float, and stretch float. Since the approach provides the revision capability, the mission planner can change some of the constraints, before and/or during the plan execution, and perform what-if analyses. The illustration demonstrates the use of point-interval logic and point graphs for such an *interactive* time-sensitive mission planning exercise.

The paper is organized as follows: Section 2 presents Point Graphs (PG), Point Interval Logic (PIL) and a brief discussion on the verification mechanism. Section 3 presents a planning application and an analysis technique that identifies the *critical* activities and time slacks for the non-critical activities. An illustration of the approach is given in Section 4 with the help of an example. Section 5 gives a brief discussion on the contribution of this paper.

2 Point Graph and Point Interval Logic

Definition 1 Point Graph

A Point Graph PG (V, E_A, D, T) is a directed graph with:

V: Set of vertices with each node or vertex $v \in V$ representing a point on the real number line. Two points pX and pY are represented as a composite point [pX;pY] if both are mapped to a single point on the line.

E_A: Union of two sets of edges: $E_A = E \cup E_{\leq}$, where

E: Set of edges with each edge $e_{12} \in E$, between two vertices v1 and v2, also denoted as (v1, v2), representing a relation '<' between the two vertices - (v1 < v2). The edges in this set are called LT edges;

$E_{\leq}$: Set of edges with each edge $e_{12} \in E_{\leq}$, between two vertices v1 and v2, also denoted as (v1, v2), representing a relation '≤' between the two vertices - (v1 ≤ v2). The edges in this set are called LE edges.

D (Length) Edge-length function (possibly partial):

$E \rightarrow \Re^{+}$.

T (Stamp) Vertex-stamp function (possibly partial):

$V \rightarrow \Re$.

In a temporal situation the '$<$' edge between two nodes in a PG, corresponds to the temporal relation 'Before.' Similarly, the '$\leq$' edge represents the relation 'Precedes' which can also be represented as a disjunctive temporal relation 'Before or Equals' written as {Before, Equals}. It can be easily shown that the PG formalism captures all the temporal relations of Pointisable Algebra (Ladkin and Maddux 1988) with the exception of '$\neq$' (not-equal-to) relation. The graph formalism can be extended to include this relation; however, the issue is not discussed in this paper.

The Point Interval Logic (PIL), on the other hand can be defined with the help of its lexicon, which consists of the following primitive symbols:

Points (Event):

A point X is represented as [pX, pX] or simply [pX].

Interval:

An interval X is represented as [sX, eX], where 'sX' and 'eX' are the two end points of the interval, denoting the 'start' and 'end' of the interval, s.t. $sX < eX$.

Point Relations:

These are the relations that can exist between two points. The set of relations R_P is given as:

R_P = {Before, Equals, Precedes}

Interval Relations:

These are the *atomic* relations that can exist between two intervals. The set of relations R_I is given as:

R_I = {Before, Meets, Overlaps, Starts, During, Finishes, Equals}

Point-Interval Relations:

These are the *atomic* relations that can exist between a point and an interval. The set of relations $R_{\square}$ is given as:

R_{PI} = {Before, Starts, During, Finishes}

Functions:

Interval length function that assigns a non-zero positive real number to a system interval e.g.

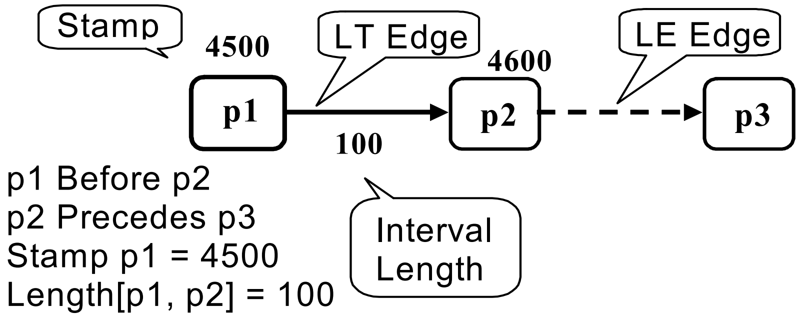
Length X = d, where $d \in \Re$.

The stamp function assigns a real number to a system point e.g.

Stamp p1 = t, where $t \in \Re$.

A system of PIL statements, also termed as a temporal system, is given by a conjunction of statements each describing a PIL relation between a *unique* pair of intervals/points.

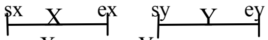
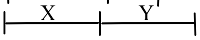
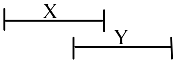
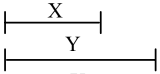
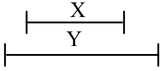
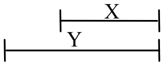
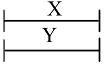
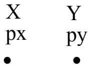
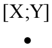
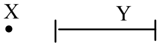
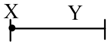
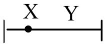
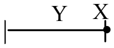
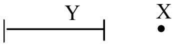
The syntactic and semantic structure of atomic relations in PIL is shown in Table 1. A qualitative relation between two intervals (or points) can be described with the help of algebraic inequalities, also shown in Table 1, among points representing the start and end of these intervals. Given the definition of a PG and the set of inequalities (in Table 1) representing temporal relations between intervals/points, it is straightforward to devise a mechanism to convert a PIL statement to its PG representation. Figure 1 presents a three-node Point Graph with vertex stamps and arc length and the corresponding PIL system represented by the PG.



p1 Before p2
p2 Precedes p3
Stamp p1 = 4500
Length[p1, p2] = 100

Figure 1. PG Representation of PIL Statements

Table 1. PIL Expressions and Their Semantics

<u>CASE I</u>— X and Y both intervals with non-zero lengths: X = [sx, ex], Y = [sy, ey] with sx < ex and sy < ey				
1. X Before Y	ex < sy			
2. X Meets Y	ex = sy			
3. X Overlaps Y	sx < sy;	sy < ex;	ex < ey	
4. X Starts Y	sx = sy;	ex < ey		
5. X During Y	sx > sy;	ex < ey		
6. X Finishes Y	sx > sy;	ey = ex		
7. X Equals Y	sx = sy;	ex = ey		
<u>CASE II</u>—X and Y both points: X = [px] and Y = [py]				
Before:	X < Y	px < py		
Equals:	X = Y	px = py		
<u>CASE III</u>— X is a point and Y is an interval: X = [px] and Y = [sy, ey]				
Before:	X < Y	px < sy		
Starts:	X s Y	px = sy		
During:	X d Y	sy < px < ey		
Finishes:	X f Y	px = ey		
Before:	Y < X	ey < px		

A set of PIL statements can now be represented as a set of PGs where each PG corresponds to a single statement in the temporal system. A consolidated PG for the entire temporal system can be constructed by *unifying* and *folding* the individual PGs (Zaidi and Wagenhals 2005). The unification looks at the nodes of a set of PGs and merges the nodes with identical node labels or the ones with equality relation between them. The folding process, on the other hand, looks at the quantitative information on nodes, and edges, of a PG and folds the edges based on the available information. Figure 2 illustrates the process of constructing a PG for a set of PIL statements with the help of an example.

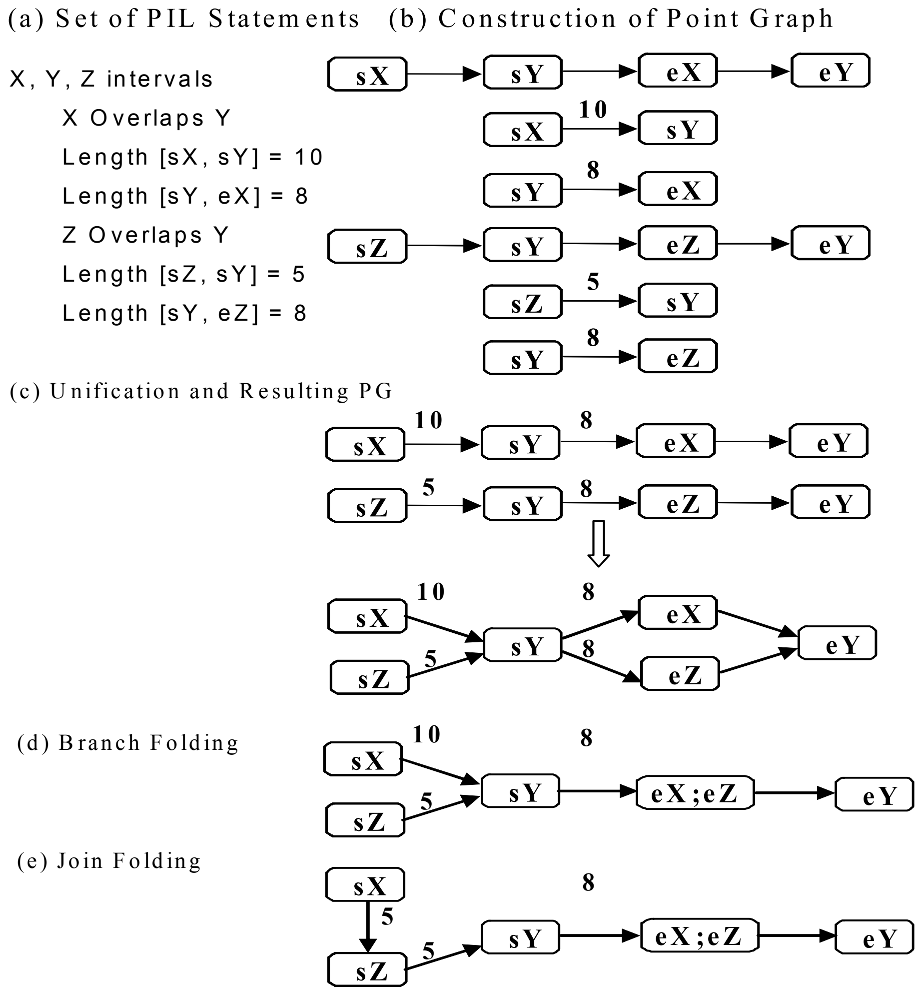


Figure 2. Steps in PG Construction

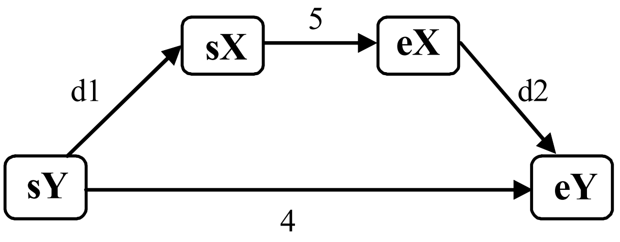
2.1 Verification of PIL Statements

The presence of inconsistent information in a temporal system results in an *erroneous* PG, which may result in erroneous inferences and/or analyses preformed on the PG. It is, therefore, imperative to identify and correct the inconsistent cases prior to any analysis. Theorem 1 characterizes the inconsistencies both with respect to relations in a PIL system and with respect to its PG representation.

Theorem 1 Inconsistency in PIL
A system's description in PIL contains inconsistent information iff

a) For some intervals/points X and Y, and atomic PIL relations Ri and Rj, both 'X Ri Y' and 'X Rj Y', $i \neq j$, or 'X Ri Y' and 'Y Ri X' (with the exception of 'Equals' relation) hold true; *or*
b) For a point p1, the system calculates two different stamps; *or*
c) For some points p1 and p2, 'p1 < p2', the system can determine two different lengths for the interval [p1, p2].

A path-consistency algorithm that employs techniques by Busacker (1965) and Warshall's algorithm (Warshall 1962), is presented in Zaidi and Wagenhals (2005) for identifying the erroneous cases in the PG representation. Figure 3 shows an inconsistent point graph



$$d1 + 5 + d2 = 4,\ \ d1, d2 > 0$$

Figure 3. An inconsistent PG

3 Application to Planning

This section presents an application of Point Graphs and Point Interval Logic, presented in Section 2, for modeling and planning temporal aspects of projects/missions. The approach presented in this section requires the temporal constraints of a mission to be converted to PIL statements. The temporal system is then converted to its PG representation. The PG, so obtained, is processed by applying unification. A necessary condition for the application of the algorithms presented in this section also requires that the length function of the PG, so constructed, be a total function, i.e., every 'Before' relationship be specified with a length of the interval involved. A temporal system that does not conform to this condition can be pre-processed, without violating any temporal requirements, by replacing every relation of the type '$X < Y$' between two points X and Y with relations '$X < Z$', 'Length[X, Z] = d', and '$Z \leq Y$', where Z is a dummy activity and d is a user-defined smallest time increment, e.g. for systems with only integer lengths and time stamps, $d = 1$. The unified PG is checked for inconsistency and temporal anomalies. Finally the folding process is applied to the PG.

In order to construct a model of the temporal system, the PG is added with a pair of s*ource* and *sink* nodes (Definition 2). The time stamps on individual nodes are not considered in the approach; the stamps can be ignored without any loss of generality. The time stamp can be easily incorporated either before or after the analysis that follows. Once a plan is constructed using the approach, the plan can be shifted on a timeline to match with the stamps provided in the input PIL statements.

Definition 2 Source and Sink Nodes to PG

A source node V_{in} and a sink V_{out} node are added to the PG representation of a system of PIL statements by applying the following:

a) $\forall vi,\ vi \in V$ such that $*v = \phi$ (i.e., null set), connect the source node V_{in} to all vi by LE type edges (V_{in}, vi);

b) $\forall vi,\ vi \in V$ such that $v* = \phi$, connect the sink node V_{out} to all vi by LE type edges (vi, V_{out}).

The graph-based approach assigns three parameters to each node in the PG representation. The parameter values are calculated by running the two algorithms, *Forward-Reverse** followed by *Reverse-Forward**, on the graph. The values of these parameters help determine the *critical activities* and time floats/slacks for intervals in the system, and *interval/point activities* defined for the PG under consideration. The three parameters are termed as *earliest occurrence (Ev), late occurrence (Lv)*, *and latest occurrence (Tv)* of a node 'v', and are formally defined in Definitions 3-5. The analysis applies two algorithms (Algorithms 1-2) on the PG using the Forward and Reverse passes in Definitions 3-5. The first calculates the value for the earliest occurrence time of a node; the other calculates the values for the late and latest occurrences of a node in the PG. Figures 4-5 illustrate the two passes with the help of example cases.

Definition 3 Earliest Occurrence, Ev – Forward Pass

The earliest occurrence Ev of a node v, $v \in V$, is defined to be the smallest time stamp on the node that satisfies the earliest occurrences of the preceding nodes, i.e.,

Let $*v = \{vi\}$

$$Ev = \begin{cases} Evi + D\,(vi, v), & \text{for } (vi, v) \in E \text{ and } |{*v}| = 1 \\ \max\limits_{i}\,[Evi], & \forall (vi, v) \in E_{\leq} \\ \max\limits_{i}\,[Evi, Evk + D(vk, v)], & \text{for } (vk, v) \in E \\ 0, & \text{otherwise} \end{cases}$$

For a *non-critical* interval/activity [v1, v2] (Definitions 6-8), Ev1 represents the *earliest start time* of the activity.

Definition 4 Late Occurrence, Lv – Reverse Pass I

The late occurrence Lv of a node v, $v \in V$, is defined to be the largest time stamp on the node that satisfies the earliest occurrences of the following nodes, i.e.,

Let $v* = \{vi\}$

$$Lv = \begin{cases} Lvi - D(v, vi), & \text{for } (v, vi) \in E \text{ and } |v*| = 1 \\ \min\limits_{i}\,[Evi], & \forall (v, vi) \in E_{\leq} \\ \min\limits_{i}\,[Evi, Lvk - D(v, vk)], & \text{for } (v, vk) \in E \\ Ev, & \text{otherwise} \end{cases}$$

Definition 5 Latest Occurrence, Tv – Reverse Pass II

The latest occurrence Tv of a node v, $v \in V$, is defined to be the largest time stamp on the node that satisfies the latest occurrences of the following nodes, i.e.,

Let $v* = \{vi\}$

$$Tv = \begin{cases} Tvi - D(v, vi), & \text{for } (v, vi) \in E \text{ and } |v*| = 1 \\ \min\limits_{i}\,[Tvi], & \forall (v, vi) \in E_{\leq} \\ \min\limits_{i}\,[Tvi, Tvk - D(v, vk)], & \text{for } (v, vk) \in E \\ Ev, & \text{otherwise} \end{cases}$$

For a *non-critical* interval/activity [v1, v2], Tv2 represents the *latest completion time* of the activity.

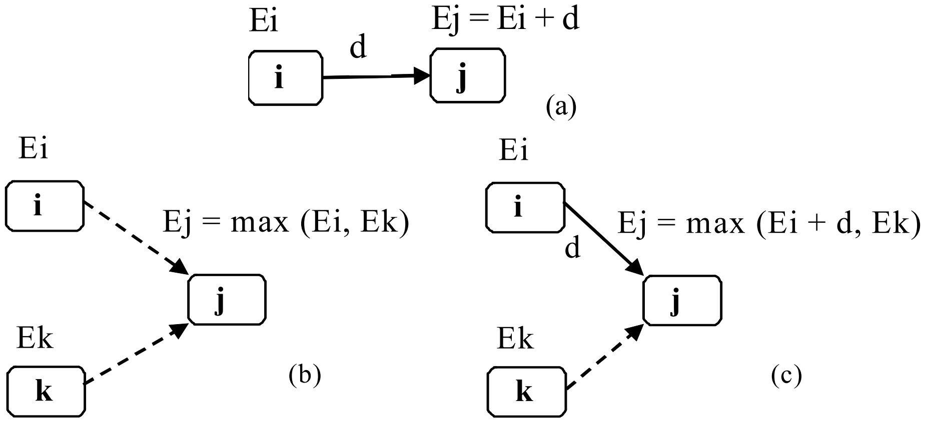


Figure 4. Illustration of Forward Pass

Algorithm 1 Forward-Reverse*

```
Apply Forward Pass to the entire PG starting from the source node Vin.
Loop |E| times.
    Set Flag = false.
    Loop for each edge (vi, vj) $\in E_A$.
        If (vi, vj) $\in E_\leq$ then
            If Ei > Ej then
                Set Ej = Ei.
                Set Flag = true.
        Else
            If Ej > Ei + D(vi, vj) then
                Set Ei = Ej - D(vi, vj).
                Set Flag = true.
            Else If Ej < Ei + D(vi, vj) then
                Set Ej = Ei + D(vi, vj).
                Set Flag = true.
        If (Flag = false) then exit Loop.
```

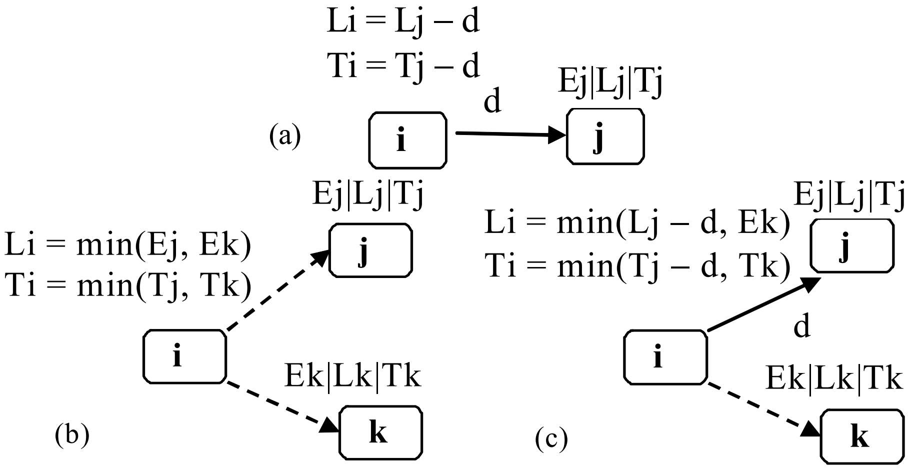


Figure 5. Illustration of Reverse Pass

Algorithm 2 Reverse-Forward*–I

```
Apply Reverse Pass I to the entire PG starting from the sink node Vout.
Loop |E| times.
    Set Flag = false.
    Loop for each edge (vi, vj) $\in E_A$.
        If (vi, vj) $\in$ E then
            If Lj > Li + D(vi, vj) then
                Set Lj = Li + D(vi, vj).
                Set Flag = true.
        Else If Lj < Li + D(vi, vj) then
            Set Li = Lj - D(vi, vj).
            Set Flag = true.
        If (Flag = false) then exit Loop.
```

A variant of Algorithm 2 is applied to calculate Tv for all v $\in$ V.

Definition 6 Point Activity

A node $v \in V$ is called a point activity. A point, start of an interval and end of an interval, are all point activities in the PG representation of a PIL system.

Definition 7 Interval Activity

An interval [v1, v2], where $v1, v2 \in V$, is called an interval activity if the two time points represented by the nodes v1 and v2 are the two end points of a path comprising of LT type edges only.

Definition 8 Critical Activity

An activity is defined to be critical if:

(a) A delay in its start will cause a delay in the completion time of the entire mission, i.e.,

i. For a point activity $v \in V$, Ev = Tv; for an interval activity [v1, v2], where $v1, v2 \in V$, $v \in \{v1, v2\}$, Ev = Tv, *or*

ii. For an interval activity, it 'Meets' or is met by another critical activity; for a point activity, it 'Starts' and/or 'Ends' another critical activity.

Definition 9 Total Float (TF) and Free Float (FF)

Total Float (TF) is the difference between the maximum time available to perform an activity and its duration. Free Float (FF) for an activity is defined by assuming that all the activities start as early as possible, it is the excess time available over its duration.

(a) Total float (TF) and free float (FF) for a non-critical point activity v are calculated as:

TFv = Tv – Ev

FFv = Lv – Ev

(b) Total float (TF) and free float (FF) for a non-critical interval activity [v1, v2] are calculated as:

$TF_{[v1, v2]}$ = Tv2 – Ev2 = Tv1 – Ev1

$FF_{[v1, v2]}$ = Lv2 – Ev2 = Lv1 – Ev1

(For all critical activities TF = FF = 0.)

Finally the PG corresponding to a mission's requirements with the values of the parameters calculated, can be used to construct a time chart, e.g. Gantt chart, showing the start and finish times for each activity as well as its relationship to other activities. For non-critical activities the plan also shows the amount of slacks or floats that can be used advantageously when such activities are delayed or when limited resources are to be used. The PG representation and the time chart can, therefore, be used for a real-time and periodic control of the plan. The PG may be updated and analyzed, and if necessary a new plan/schedule is determined for the remaining portion of the mission in a dynamic environment.

4 Illustrative Example : Mission Planning

This section provides a fictitious but real world example to illustrate how the approach presented in this paper could be applied to military planning and execution problems. The example scenario has been taken from Zaidi and Wagenhals (2005) for an illustration of the approach. The illustration is for a precision engagement against a Time Critical Target (TCT). To do this, a scenario is presented in which several assets must concurrently perform activities with implicit synchronization in order to attack a target of importance. The target is time critical in that it is difficult to locate and when it is located, it must be struck in a very short time, otherwise it will disappear.

Assume the following facts and constraints apply to the planning for precision engagement of TCTs. There is a list of high value TCTs that when located and identified need to be attacked quickly with precision engagement weapons. When such a target is found, a weapon platform such as an attack aircraft must ingress to a weapon launch point to release a precision-guided weapon (PGW). During the ingress, the on-board navigation and guidance processor of the PGW will be uploaded with the precise data it needs to fly to and hit the target. During the ingress and PGW update activities, a local, on site, aid to the navigation and guidance activity must participate in providing updates to the PGW. This local, on site activity must cease just prior to the weapon striking the target. Once the weapon is launched, the launch platform egresses the area.

Table 2. Mission Requirements

Interval ID	**Activity Description**	**Corresponding PIL Statement**
A	Weapon Platform ingresses to PGW launch point	Length A = 5
B	Weapon Platform egresses from PGW launch point	Length B = 5
C	Target parameters are uploaded into the PGW navigation processor	Length C = 5
D	PGW is launched and flies to the Target	Length D = 2
E	Local, on site activity provides navigation and guidance update to PGW	Length E = 10

Table 3. Additional Constraints

Natural Language Description	**Corresponding PIL Statement**
The platform will not loiter in the area due to threat considerations	A meets B
The PGW is launched immediately after the target parameters are uploaded	C meets D
The PGM launch precedes the egress	C Precedes B
Local, on site activity must cease just prior to weapon striking the target	eE Precedes eD

The plan for this scenario can be mapped to PIL statements presented in Table 2. The table shows the five activities together with the PIL statements representing the mission operational concept. The additional constraints are described in Table 3 with their corresponding PIL statements. The reader should note that the temporal requirements described by ‘eE Precedes eD’ (eE and eD represent the end points of intervals E and D, respectively) and the combination of ‘A meets B’ and ‘C Precedes B’ cannot be handled by existing critical path methods. These constraints represent partial-order relations between the intervals E and D, and between intervals A and C. A partial-order relation is a relation that might not *total* order the intervals of the temporal system. Instead, it specifies constructs ($<$ and/or $\leq$) among all combination of start and end points of the interval involved. The existing critical path methods can not handle partial-order relations.

These mission requirements are converted to the corresponding PG representation as shown in Figure 6. Note the source node Vin and sink node Vout representing the start and completion of the mission, respectively. Table 4 shows the attributes of the various activities involved in the mission. From this table, the minimum time required to execute the mission is 13 time units (perhaps 13 minutes). Furthermore, the start and end times of all activities are captured in the PG. All the activities are critical. The values in Fig. 6, therefore, show the *only* feasible schedule for the activities involved for the mission duration of 13 time units. Thus the local, on site activity starts at time 0, the Ingress and the PGW upload start at time 3. The PGM launch occurs at time 8 and commences the Egress activity. The PGW strikes the target at time 10 just after the local, on site activity ceases. This plan provides a total mission view that can be used to provide, to the individual resources that are carrying out the plan, the critical start and complete times for their activities to ensure that the implicit synchronization of the concurrent activities is accomplished.

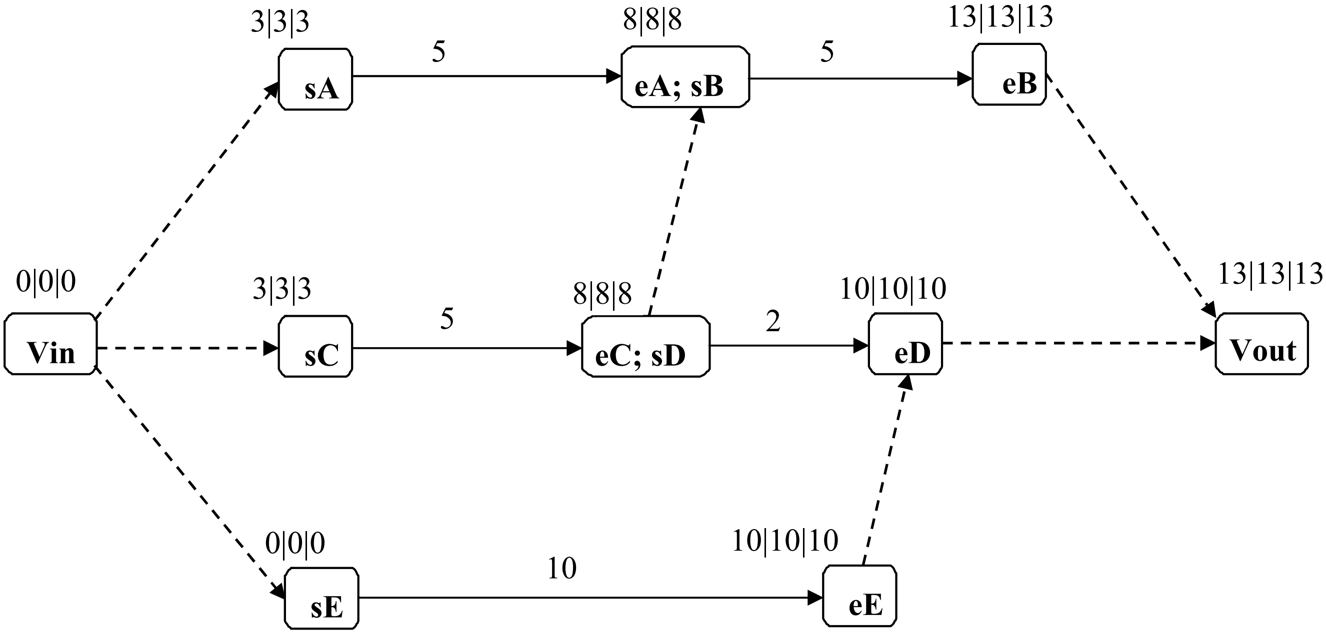


Figure 6. Point Graph for the mission

Table 4. Attributes of Mission Activities

Activity	Duration	Earliest Start Time	Latest End Time	Critical	Total Float	Free Float
A	5	3	8	yes	0	0
B	5	8	13	yes	0	0
C	5	3	8	yes	0	0
D	2	8	10	yes	0	0
E	10	0	10	yes	0	0

5 Conclusion

The approach presented in this paper extends the classical duration-based quantitative approaches for planning and project management by adding the provision for point (instantaneous) activities and specification of partially ordered relation between system activities. It also offers an expressive input language for planners to input their specifications. The approach is based on a logic that provides an added benefit for planners, especially when they need to analyze a generated plan and/or run a 'what-if' type analysis. A variant of the approach is presented by Zaidi and Wagenhals (2005) that introduces a notion of 'Stretch Slack' for critical activities. The two approaches, in our opinion, offer an effective toolkit for mission planners. The approach offers an enhanced formalism for planning in terms of its expressive language for specifications, provision for point and interval descriptions of temporal events, and a powerful inference engine.

References

Allen, J. F. 1983. Maintaining Knowledge About Temporal Intervals. *Communications of ACM* 26:832-843.

Allen, J. F. 1984. Towards a General Theory of Action and Time. *Artificial Intelligence* 23(2):123-154.

Busacker, R. G. 1965. *Finite Graphs and Networks: an introduction with application.* New York.: McGraw-Hill.

Ladkin, P. B., and Maddux, R. 1988. On binary constraint networks, Technical Report, KES.U.88.8, Kestrel Institute, Palo Alto, Calif.

Moder, J., and Philips, C. 1970. *Project Management with CPM and PERT*, 2nd Edition. New York.: Van Nostrand Reinhold.

Warshall, S. 1962. A theorem on boolean matrices. *Journal of the ACM* 9(1):11-12.

Zaidi, A. K. 1999. On Temporal Logic Programming Using Petri Nets. *IEEE Transactions on Systems, Man and Cybernetics*, Part A, 29(3): 245-254.

Zaidi, A. K., and Levis, A. H. 2001. TEMPER: A Temporal Programmer for Time-sensitive Control of Discrete-event Systems. *IEEE Transaction on Systems, Man, and Cybernetic,* 31(6):485-496.

Zaidi, A. K. 2002. A Temporal Programmer for Time-Sensitive Modeling of Discrete-Event Systems. In *Proceedings of IEEE - Systems, Man, and Cybernetics Society 2000 Meeting* . Nashville, TN.

Zaidi, A. K., and Wagenhals, L. W. 2005. Planning Temporal Events Using Point Interval Logic. Special Issue of *Mathematical and Computer Modeling*. Forthcoming.