Study on Temporal Consistency Detecting in Coordinated Model Navy Aviation Operation Based on STN

Haiyang Ma*, Huayuan Zhu, Yunfei Ji, Yang Liu and Rong Chen
Qingdao Brunch of Naval Aeronautics University, Qingdao, China

*Corresponding author email: mahy0628@126.com

Abstract. The coordinated operation of navy aviation is complex and there may be hidden time conflicts. On the basis of the previous research work, this paper detected the conflicts of the time coordinated model of a sea-borne strike. Referring to the theory of temporal constraint network (STN), the paper converted the time conflicts detection problem from consistency detecting to negative cycles detecting in STN model and designs consistency detecting function based on BFCT algorithm and FIFO queue. Then, the conflict detection of coordinated actions was realized under specific combat cases and assumptions.

Keywords: STN; coordinated operations; Negative cycles; Consistency detecting.

1. Introduction
In the coordinated operation, the coordinated plan is an important basis for the combat troops to carry out command activities and coordinate the combat operations. Therefore, collaborative planning must be thorough, detailed, and highly enforceable. However, due to the complexity of the battlefield situation and the objective limitations of the planners themselves, the collaborative action between the participating forces may lead to potential conflicts in the implementation of the plan. Explicit conflicts can be dissolved by simple computational inference, but hidden conflicts in collaborative plans are not easily detected. At this point, if the coordination plan is not adjusted, there will be mutual influence between the participating forces and even mistaken injury, reducing the effectiveness of the coordinated operations.

In order to find the hidden conflict in the collaborative plan, it is necessary to detect the conflict between the possible collaborative actions in the process of collaborative operation from a quantitative point of view. It is found that the time constraint which cannot be satisfied provides the basis for the commander to adjust the coordination plan, so that the coordinated operation plan is carried out safely and smoothly.

Synergy planning in collaborative operations can be considered a business flow with time-bound information. For similar business flows, simple Temporal Network (STN) can be applied to establish a time constraint model to turn problems into detection and validation of constraint networks[1]-[6]. The Time Constraint Network (STN) is the 1991 R. An effective means of describing and reasoning the time constraint system proposed by Dechter and others. In the theoretical model of STN, sometime relationships implied in events can be derived from the explicit time relationship between certain events in the event set, which can provide an intuitive and convenient theoretical model for time constraint reasoning. Based on this, the author has established the STN model of the air force cooperation plan in the literature[4]and[5]. On the basis of the previous research, based on the theory of time-bound network, this paper applies the conflict detection function to carry out conflict detection...
on the STN model of the established air force's coordinated combat against the sea. It provides an effective way to find out the invisible time conflict in the naval air force's coordinated operation plan.

2. STN Related Theories

2.1. Basic Definitions and Theorems

Definition 1.1 Cycle. In the distance graph STN, the sequence of nodes and arcs \((t_0, e_1, t_1, e_2, \cdots, t_n, e_n, t_0) \quad (n \geq 1)\) is called STN cycle.

In Definition 1.1, If \(\forall i \neq j \Rightarrow t_i \neq t_j\) means the number of times the cycle mentioned in the definition passes through the same node is not more than 1, that is, the basic cycle.

Definition 1.2 Negative Cycle. For a cycle of the distance map STN, when the path length of the cycle is negative, it is called the negative cycle of STN.

When \(n = 2\), it is a negative cycle with a path length of -1, as shown in Figure 1.

![Figure 1. A negative cycle.](image)

Theorem 1.1 Consistency Judgment of STN. STN is consistent if and only if there is no negative cycle in the distance graph \([7]\).

The analysis shows that Theorem 1.1 is clearly established. Because if there is a negative cycle in the STN distance graph, then the conclusion drawn after simply superimposing the inequality is contradictory with \(t_i - t_j < 0\).

2.2. Problem Transformation

According to Theorem 1.1, if there are negative cycles in the STN distance graph, each negative cycle corresponds to contradictory set of coordinated action constraints (hereinafter referred to as "conflict"). Thus, the problem of time conflict detection in the cooperative plan of aviation cooperative operations can be transformed into the problem of detecting the negative circle in the distance map.

If there are only a few nodes in the STN distance graph, a more intuitive enumeration method can be directly used to find the negative circle. However, in a weighted directed graph, the number of circles will increase exponentially with the number of nodes.

Therefore, when the number of nodes is too many, it is not feasible to traverse all the circles to detect all negative circles. In view of this, the conflict detection algorithm of the STN model is designed, based on the BFCT negative cycle detection algorithm, in this paper.

3. Conflict Detection Algorithm

The conflict detection algorithm designed in this paper is implemented based on the conflict detection function \(Detect\_Consistency(Q,G)\).

Among the multiple negative circle detection algorithms for weighted directed graphs, the BFCT algorithm has the advantages of high efficiency and stability \([8][9]\), and uses the first-in-first-out (FIFO) queue form as an assistant data structure, which is convenient for the design and realization of the function. Therefore, the conflict detection function based on the BFCT algorithm is a better choice.

3.1. Principle of BFCT Algorithm

The BFCT algorithm detects the existence of a negative cycle in the process of calculating the shortest path of a single source.
When there is no negative cycle in the directed graph $G = (V, E)$, the shortest path length $d(v)$ from the source point $s$ to the rest of the nodes $v \in V$ can be obtained by applying the BFCT algorithm. The basic idea of the algorithm is: for arc $u \rightarrow v$, if $d(v) > d(u) + w(u,v)$, then let $d(v) = d(u) + w(u,v)$, and at the same time let the precursor pointer $pre(v) = u$. $w(u,v)$ is the weight of arc $u \rightarrow v$. If there is no negative cycle in the directed graph $G$, the shortest path from the source point $s$ to the other nodes forms a shortest path tree $G_s$, which is a spanning tree with the root node $s$ as the root node, which can be constructed by backward inference of the precursor pointer $pre(\cdot)$. Otherwise, if there is a negative cycle in, there will be a cycle in.

The basic idea of the algorithm is: The algorithm uses FIFO queue $Q$ as an assistant data structure to store the node $u$ to be scanned. The specific process is as follows:

**Step 1**: Initialization, for $G = (V, E), \forall v \in V$, let $d(v) = +\infty$, $pre(v) = Null$, add the source point $s$ to the queue $Q$ and set $d(s) = 0$;

**Step 2**: Suppose $u$ that represents the node $G$ that has been added to the queue and is about to leave the queue, and then let $u$ out of the queue. For the arc $\forall (u, v) \in E$, if $d(v) > d(u) + w(u,v)$, then let $pre(v) = u$, if $v \notin Q$, $v$ will join in $Q$;

**Step 3**: Judge $u$ that in $G_s$ whether it is on the subtree $G_s[v]$ that is the root node $v$. If it is, there is a cycle in the $G_s$, go to Step 4, otherwise judge $Q$ whether it is empty, if it is empty, go to Step 5, otherwise the subtree $G_s[v]$ will be decomposed, that is, the node in $G_s[v]$ except $v$, $\forall v' \in G_s[v]$, let $d(v') = +\infty$, $pre(v') = Null$, go to Step 2;

**Step 4**: The algorithm detects that there is a negative cycle in $G_s$ and constructs the negative cycle by backward inference of the precursor pointer $pre(\cdot)$, and the algorithm ends;

**Step 5**: There is no cycle in $G_s$, and the algorithm obtains the shortest path tree from the source point $s$ to the remaining nodes, indicating that there is no negative cycle in $G_s$, and the algorithm ends.

The above algorithm is easier to understand. In fact, if it is set $u'$ as the immediately preceding node of the source point $s$ in a certain cycle of the coordinated action STN distance graph, when a cycle appears in $G_s$, it means that it appears $d(s) > d(u') + w(u',s)$ that is $d(u') + w(u',s) < 0$ in the last step of constructing the shortest path, the unequal sign On the left is the path length (weight) of the cycle, that is, there is a negative cycle.

The process of the BFCT algorithm is represented by the flowchart shown in Figure 2.

### 3.2. Conflict Detection Function

Based on the conflict detection function $Detect\ _{Consistency}(Q,G)$ detected by the BFCT algorithm, when detecting whether there is a negative cycle in the STN model, there are two input parameters $Q,G$ among which $Q$ is the FIFO queue and $G$ is the established STN model.

The pseudo code of the design function is as follows:

```plaintext
Detect\ _{Consistency}(Q,G) // Q: FIFOQueue; G: STN distance graph
if (BFCT(Q,G) == True)
    construct NC using pre(\cdot);
end if
```

Function $Detect\ _{Consistency}(Q,G)$, the first step of detection is to call the return value of the BFCT algorithm. If the return value is True, it means that a negative cycle is detected. Then the negative cycle is constructed through the precursor pointer $pre(\cdot)$, which also means that a conflict is found in the coordination in the collaborative plan of the STN action model.
Initialization: for $\forall v \in V$, let $d(v) = +\infty$, $pre(v) = Null$

Let $u$ out of $Q$, if $d(v) > d(u) + w(u,v)$, Let $d(v) = d(u) + w(u,v)$, $pre(v) = u$, put $v$ into $Q$

$u$ in $G_p[v]$? $N$ $Y$

$Q$ is empty? $N$ $Y$

Resolve $G_p[v]$

Negative cycles in $G$, construct the cycle by $pre(\cdot)$

No negative cycles in $G$

End

Figure 2. The process of BFCT algorithm.

4. Simulation Example

Based on the basic assumptions, the conflict detection algorithm designed in this paper is used to detect the time conflict in the STN model of coordinated action.

4.1. Basic Assumptions

Operational scenario: The formation of red carrier-based aircraft performs coordinated assault operations against the sea. The task points are detection, suppression, interference, cover, penetration, and observation, and time nodes are set.
Let the reference time be 0 and the number be 1. According to the red side collaborative plan, the coordinated actions are decomposed, and the time constraint relationship (arc length) between the coordinated actions is set, and the task STN model is established according to the paper [4] and [5], as shown in Figure 3.

Figure 3. STN model of coordinated action.
4.2. Conflict Detection

The conflict detection algorithm designed in this paper is used to detect the conflict between coordinated actions in Figure 3, that is, whether there is a negative cycle in the coordinated action STN model shown in Figure 3.

First, calculate the single source shortest path. For simplicity, the node number in the STN model is used in the narrative process to replace each time node in carrier-based aircraft cooperative operations. Set the shortest path value of the node in the STN model to \( +\infty \), put node 1 into the queue \( Q \), and let \( d(1) = 0, \ pre(1) = \text{null} \).

Take node 1 out of the queue because:
\[
d(2) = +\infty \times d(1) + w(1,2) = 0 + 95 = 95, \quad d(4) = +\infty \times d(1) + w(1,4) = 0 + 70 = 70
\]
\[
d(6) = +\infty \times d(1) + w(1,6) = 0 + 70 = 70, \quad d(8) = +\infty \times d(1) + w(1,8) = 0 + 64 = 64
\]
\[
d(10) = +\infty \times d(1) + w(1,10) = 0 + 66 = 66, \quad d(12) = +\infty \times d(1) + w(1,12) = 0 + 69 = 69
\]
\[
d(14) = +\infty \times d(1) + w(1,14) = 0 + 75 = 75, \quad d(16) = +\infty \times d(1) + w(1,16) = 0 + 77 = 77
\]

Let \( d(1) + w(1,2) = 95, \ pre(2) = 1; d(4) = d(1) + w(1,4) = 70, \ pre(4) = 1; d(6) = d(1) + w(1,6) = 70, \ pre(6) = 1; d(8) = d(1) + w(1,8) = 64, \ pre(8) = 1; d(10) = d(1) + w(1,10) = 66, \ pre(10) = 1; d(12) = (1) + w(1,12) = 69, \ pre(12) = 1; d(14) = d(1) + w(1,14) = 75, \ pre(14) = 1; d(16) = d(1) + w(1,16) = 77, \ pre(16) = 1, \) put the node 2, 4, 6, 8, 10, 12, 14, 16 into the queue \( Q \).

Take node 2 out of the queue because:
\[
d(3) = +\infty \times d(2) + w(2,3) = 95 - 50 = 45, \quad \text{let} \ d(3) = 45, \ pre(3) = 2, \ \text{put node 3 into} \ Q .
\]

Take node 4 out of the queue because:
\[
d(5) = +\infty \times d(4) + w(4,5) = 70 - 11 = 59, \quad d(14) = 75 \times d(4) + w(4,14) = 70 + 0 = 70
\]
Let \( d(5) = 59, \ pre(5) = 4; d(14) = 70, \ pre(14) = 4, \) put node 5, 14 to the queue \( Q \).

Take node 6 out of the queue because:
\[
d(7) = +\infty \times d(6) + w(6,7) = 70 - 1 = 69, \quad d(9) = +\infty \times d(6) + w(6,9) = 70 + 0 = 70
\]
Let \( d(7) = 69, \ pre(7) = 6; d(9) = 70, \ pre(9) = 6, \) put node 7, 9 into the queue \( Q \).

Take node 8 out of the queue because:
\[
d(9) = 70 \times d(8) + w(8,9) = 64 - 1 = 63
\]
Let \( d(9) = 63, \ pre(9) = 8, \) put node 9 into the queue \( Q \).

Take node 10 out of the queue because:
\[
d(11) = +\infty \times d(10) + w(10,11) = 66 - 2 = 64
\]
Let \( d(11) = 64, \ pre(11) = 10, \) put node 10 into the queue \( Q \).

Take node 12 out of the queue because:
\[
d(13) = +\infty \times d(12) + w(12,13) = 69 - 3 = 66, \quad \text{let} \ d(13) = 66, \ pre(13) = 12, \ \text{put node 13 into} \ Q .
\]

Take node 16 out of the queue because:
\[
d(17) = +\infty \times d(16) + w(16,17) = 77 - 1 = 76
\]
Let \( d(17) = 76, \ pre(17) = 16, \) put node 17 to the queue \( Q \).

Take node 3 out of the queue, there is no node that meets the shortest path length \( d(v) > d(3) + w(3,v) \), so the nodes of the queue \( Q \) remain unchanged.

Take node 5 out of the queue, there is no node that meets the shortest path length \( d(v) > d(5) + w(5,v) \), so the nodes in the queue \( Q \) remain unchanged.

Take node 14 out of the queue because:
\[
d(15) = +\infty \times d(14) + w(14,15) = 70 - 6 = 64
\]
Let \( d(15) = 64, \ pre(15) = 14, \) put node 15 into the queue \( Q \).

Take node 7 out of the queue, there is no node that meets \( d(v) > d(7) + w(7,v) \), so the nodes in the queue \( Q \) remain unchanged.
Take node 9 out of the queue because: \( d(7) = 69 > d(9) + w(9,7) = 63 + 0 = 63 \)
let \( d(7) = 63 , pre(7) = 9 \), put node 7 into the queue \( Q \).
Take node 11 out of the queue because: \( d(5) = 59 > d(11) + w(11,5) = 64 - 10 = 54 \)
Let \( d(5) = 54 , pre(5) = 11 \), put node 5 into the queue \( Q \).
Take node 13 out of the queue, there is no node that meets \( d(v) > d(13) + w(13,v) \), so the nodes in the queue remain unchanged.
Take node 17 out of the queue, there is no node that meets \( d(v) > d(17) + w(17,v) \), so the nodes in the queue remain unchanged.
Take node 15 out of the queue because:
\[ d(12) = 69 > d(15) + w(15,12) = 64 + 0 = 64 \]
\[ d(1) = 0 > d(15) + w(15,1) = 64 - 69 = -5 \]
let \( d(12) = 64 , pre(12) = 15 \); \( d(1) = -5 , pre(1) = 15 \), put node 12, 1 into the queue \( Q \).
At this time, node 15 is on the subtree \( G_p[1] \) with node 1 as the root node, that is, a cycle appears in \( G_p \). That is, the algorithm detects that there is a negative cycle in the STN model. Then, the negative cycle is constructed through the inverse of the precursor pointer \( pre(\cdot) \), and the node sequence is \( 1 \rightarrow 4 \rightarrow 14 \rightarrow 15 \rightarrow 1 \), and the path length is -5, as shown by the bold line in Figure 4.

![Figure 4. The negative cycle in the STN model.](image)

Analysing the negative cycle in Figure 4, the time conflicts corresponding to it are obvious, that is, if the coordinated operations are implemented according to the coordinated plan, the coordinated actions 1, 4, 14, 15 corresponding to the negative cycle will not be implemented smoothly. According to the analysis of the detection results, the conflict can be eliminated by adjusting \( d(4) = 75 \).

5. Conclusion
In this paper, conflict detection is carried out on the STN model of cooperative planning of aviation cooperative operations, and the results prove the effectiveness of conflict detection. Conflict detection is the basis and prerequisite to ensure the time consistency of the collaboration and the feasibility of the collaboration plan. However, the established model still has shortcomings and only considers the time constraint relationship in the action plan. At the same time, as a special mission planning process, collaborative operations actually include not only time constraints, but also other constraints such as resources and space. The modelling and conflict detection of collaborative plans for multiple constraints will be our focus.
In addition, time conflicts include static conflicts and dynamic conflicts. The conflict detection and resolution algorithm designed in this paper is only suitable for static conflict processing. Dynamic and muti-constrained conflict detection for coordinated combat plans will be our future research.
References

[1] XIE Bin, LIN Hua, XING Chang-feng. Automatic Resource Conflict Resolution in Process of Task Execution Based on STN[J], Fire Control & Command Control, 2015,40(6):984-988.

[2] Groesbrink S, Oberthur S, Baldin D. Architecture for Adaptive Resource Assignment to Virtualized Mixed-Criticality Real-Time Systems [J]. ACM SIGBED Review, 2013, 10(1):18-23.

[3] TANG Luo-hao. Research on temporal representation and conflict resolution of action Plan based on STN [D]. National University of Defense Technology.

[4] MA Hai-yang, LI Dong, SUN Huai-liang. Research on the Time constraint Model of Aviation Air to Sea Cooperative Operation [J], Ship Science and Technology. 2016, 38(3):147-150.

[5] CHEN Rong, YAN Jian-gang, ZHOU Xiao-cheng. Research on Conflict Resolution of Time Coordination in Attack Against Ground Based on STN[J], Fire Control & Command Control,2013,38(8):21-24

[6] Yu Wenshu, Zhou Yong, Yan Xuefeng. Time Consistency Resolution for Business Flow Based On STN[J], Computer Applications and Software, 2016,33(4):239-241.

[7] Dechter R, Meiri I, Pearl J. Temporal Constraint Networks [J]. Artificial Intelligence, 1991, 49(5): 61-95

[8] Cherkassky B V, Georgiadis L, Goldberg A, et al. Shortest-Path Feasibility Algorithms: An Experimental Evaluation[J]. ACM Journal of Experimental Algorithmics, 2009, (14):118-132.

[9] TANG Luo-hao, ZHU Cheng, ZHANG Wei-ming. Temporal conflict detection and resolution in process of plan execution based on STN. Application Research of Computers, 2011, 28(5):1780-1782.