The Structure of and Constraints on Strike Packages

Kestrel Institute Technical Report

Stephen Fitzpatrick
fitzpatrick@kestrel.edu

Marcel Becker
becker@kestrel.edu

Douglas R. Smith
smith@kestrel.edu

A strike package is a coordinated group of air missions that traverse enemy airspace together to engage targets. This report discusses the structure of and constraints on strike packages.[1]

1. Introduction

It is common practice for aircraft carrying out missions in enemy airspace to be grouped together in *strike packages* — they form up as a group outside enemy airspace, travel together within enemy airspace to various locations to achieve their objectives, exit enemy airspace together, and then disband.

The aircraft may come from different air units, located at geographically dispersed bases. The aircraft may be of different types, with different characteristics (e.g., speed, equipment loads, fuel capacity and fuel burn rates) and different capabilities (e.g., some may engage targets while others jam enemy anti-aircraft facilities).

This report presents a model of strike packages that is high-level and somewhat simplified, but not entirely unrealistic. In the following, the basic entities are introduced, then the structure and constraints on strike packages are detailed. Relaxed constraints (necessary conditions) are then discussed — these can be useful in tools for interactively constructing packages or for automatically constructing packages using search.

2. Basic Entities of the Air Domain

2.1. Air Units, Platforms and Missions

An *air unit* is a facility that provides *platforms* (i.e., aircraft) for use in *air missions*. A mission is a group of the unit's platforms operating simultaneously and coherently to achieve some task — they take off together, travel together, carry out activities together (e.g., refueling, or engaging targets), return to their unit together and land together. A unit provides platforms of a single type (e.g., F01 fighters or E01 electronic warfare aircraft). Some units are based on aircraft carriers and are, in principle, mobile, but in this report they are treated as immobile.

[1]The results described in this document were supported by DARPA Resilient Synchronized Planning and Assessment for the Contested Environment (RSPACE) Program under Air Force Research Laboratory (AFRL) contract HR0011-15-C-0138.

The platforms provided by a single unit may be used for multiple types of mission. For example, a fighter unit may provide platforms for:

- *strike missions* that directly engage targets,
- *defensive counter air* (DCA) missions that patrol outside enemy airspace near high-value assets (such as aerial refueling tankers or command buildings),
- *offensive counter air* (OCA) missions that defend friendly forces against enemy fighters in enemy airspace,
- *airborne alert missions* that patrol areas where mobile targets are expected to appear.

Other mission types include: *aerial refueling* (AR) in which tanker platforms maintain station on designated *orbits* waiting to provide fuel to other platforms); *command and control* (C2) missions in which airborne battle managers maintain control of friendly forces; and *close air support* (CAS) in which air platforms engage enemy ground forces in close proximity to friendly ground forces.

2.2. Goes

Each unit makes its platforms available for missions during certain operational periods, called *goes*. More specifically, each go provides a fixed number of *sorties* between some start time and some end time, where a sortie refers to a single platform taking off, performing some activities and then landing. (So a mission comprising n platforms can also be thought of as comprising n identical sorties.)

For a given unit, the time periods of the goes do not overlap.

For example, Table 1 shows three goes (per day) for two fighter units, FS1 and FS2, providing F01 and F02 platforms, and for one SEAD unit providing EA01 platforms. (The decrease in the number of sorties through a day is to accommodate maintenance and repair of the platforms.)

Unit	Platform	Go Start Time	Go End Time	# Sorties Available
FS1	F01	0600	1200	12
		1200	1800	10
		1800	0000	8
FS2	F02	0600	1200	8
		1200	1800	6
		1800	0000	6
SEAD1	EA01	0600	1200	6
		1200	1800	4
		1800	0000	4

Table 1: Goes for three units

In general, a sortie must take off and land within the time period of a single go.

2.3. Standard Configuration Loads

Individual platforms at a unit may be configured differently, with different combinations of munitions, sensors and external fuel tanks, for example. Such a combination is referred to as an *Standard Configuration Load* (SCL). Thus, each go can be more finely characterized in terms of the number of sorties for each platform/SCL combination made available.

A specific platform's SCL can be changed, but this requires time — the amount of time depends on the initial and final SCLs, the platform type, and the availability of ground crew — and thus may result in the platform not being available for one or more goes, which may require the number of sorties provided in those goes to be reduced.

This report assumes that the effects of any planned SCL changes are already reflected in the numbers of sorties advertised for each go.

In a given mission, all of the platforms are configured identically. For example, a mission might be 2 F01s, each with an SCL of 4 guided bombs, from the 0600 go at FS1, that are to engage some target.

2.4. Constraint on the Number of Sorties/Platforms Used

For each go, the total number of sorties committed to missions of all types must not exceed the number of sorties available in that go:

$$\forall g : \mathsf{Go} \cdot \sum_{m \in \mathsf{missions}(g)} \mathsf{sorties}(m) \quad \leq \mathsf{availableSorties}(g)$$

where $\mathsf{missions}(g)$ is the set of missions with sorties from go g, $\mathsf{sorties}(m)$ is the number of sorties in mission m, and $\mathsf{availableSorties}(g)$ is the number of sorties available in go g. (Recall that the sorties in a given mission all come from a single go.)

Likewise for each configuration in each go.

$$\forall g : \mathsf{Go}, L : \mathsf{SCL} \cdot \sum_{m \in \mathsf{missions}(g) \wedge \mathsf{scl}(m) = L} \mathsf{sorties}(m) \quad \leq \mathsf{availableSorties}(g, L)$$

where $\mathsf{scl}(m)$ is the SCL of mission m, and $\mathsf{availableSorties}(g, L)$ is the number of sorties with configuration L available from go g. (Recall that all platforms in a given mission have the same SCL.)

2.5. Targets, Strike Points, NETs and NLTs

A target is an enemy vehicle, structure, piece of equipment, etc, that is to be engaged by friendly forces. A target has one or more *strike points*, given by $\mathsf{strikePoints} : \mathsf{Target} \to \mathsf{Set}(\mathsf{StrikePoint})$, that indicate where a weapon should be delivered against the target. For example, an airfield may have as strike points the control tower, radars, hangers and runways.

Each strike point has its own location that precisely designates where the weapon should be delivered, but for some purposes the strike points are close enough that they can be treated as being co-located at the target's nominal location. For example, a mission's *route* may contain just a target (e.g., an airport) rather than each individual strike point (e.g., the airport's control tower).

Each strike point has a set of *munition options* that represent recommended ways to destroy the strike point. Each munition option designates:

- a recommended munition type, given by $\mathsf{munition} : \mathsf{MunitionOption} \to \mathsf{Munition}$;
- a recommended quantity of the munition, given by $\mathsf{quantity} : \mathsf{MunitionOption} \to \mathbb{N}^+$.

In addition, the strike point associates each munition option with a probability that use of the munition option will cause the strike point to be destroyed ('killed'):

$$\mathsf{p_k} : \mathsf{StrikePoint} \times \mathsf{MunitionOption} \to \mathbb{R}$$

where $\forall s, m : 0 \leq \mathsf{p_k}(s, m) \leq 1$.

Each target has a time window, outside of which it (i.e., its strike points) cannot be engaged:

$$\mathsf{timeWindow} : \mathsf{Target} \to (\mathsf{Time} \times \mathsf{Time}) \ .$$

The start of the time window is called the *no-earlier-than* time (NET) and the end is called the *no-later-than* time (NLT):

$$\begin{aligned}
&\mathsf{net}(T) : \mathsf{Target} \to \mathsf{Time} \\
&\mathsf{nlt}(T) : \mathsf{Target} \to \mathsf{Time} \\
&\forall T : \mathsf{Target} \cdot \mathsf{net}(T) < \mathsf{nlt}(T) \\
&\forall T : \mathsf{Target} \cdot \mathsf{timeWindow}(T) = [\mathsf{net}(T), \mathsf{nlt}(T)] \ .
\end{aligned}$$

A time window of $[-\infty, \infty]$ means that the target can be engaged at any time.

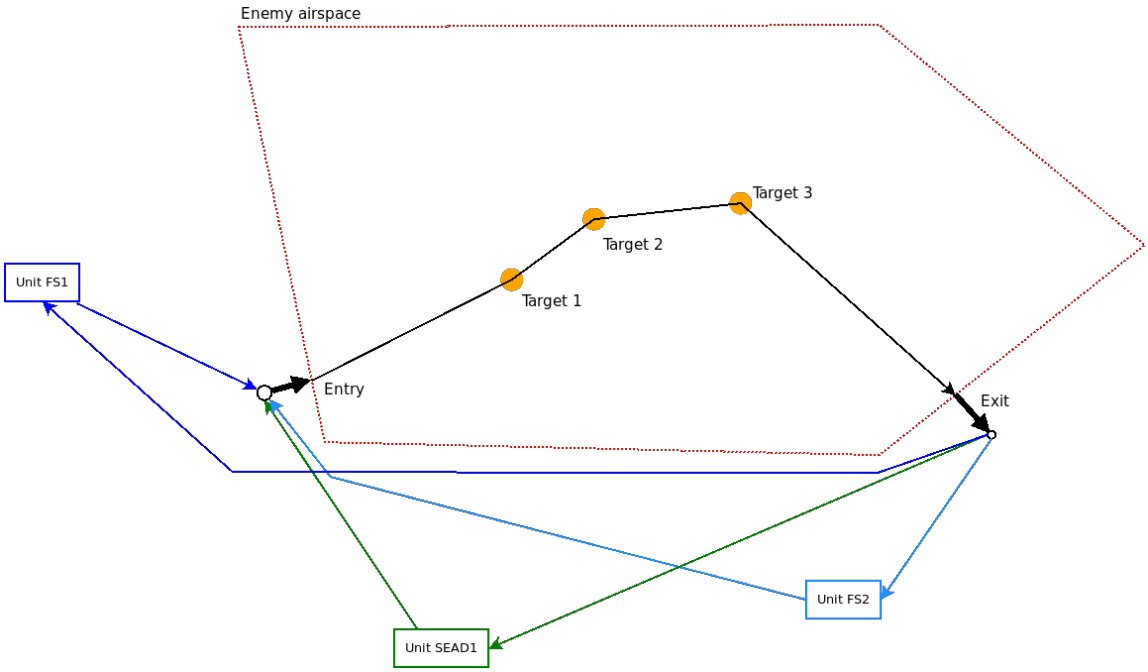

Figure 1: Missions from various units form up as a package to enter enemy airspace, engage targets, exit enemy airspace, and then return to their units

2.6. Threats

A threat is an enemy entity that poses a risk to friendly missions — ground-based threats such as SAM sites or AAA units, or air-based threats (i.e., airfields — note that this report models the airfield itself as the threat rather than the individual fighters that the airfield can deploy). A threat may also be a target (i.e., it may be planned for attack).

A threat is characterized in terms of its type (e.g., SAM02 or SAM03), its location (in this report, threats are immobile) and its range.

SEAD/EW missions are used to reduce the risk from ground-based threats. OCA missions are used to reduce the risk from air-based threats.

3. Strike Packages

A *strike package* is a group of air missions, from multiple units, that coordinate their passage through and activities in *enemy airspace*. A package contains one or more *strike missions* that are primarily responsible for engaging the package's designated targets. A package typically also contains *support missions*, including:

- *SEAD* (Suppression of Enemy Air Defenses) or *EW* (Electronic Warfare) support missions that provide defense against enemy anti-air defenses, such as *SAM* (Surface-to-Air Missiles) or *AAA* (Anti-Air Artillery);
- *OCA* (Offensive Counter-Air) support missions that provide defense against enemy fighter aircraft that might be encountered in enemy airspace.

Operating as a package, the missions can accomplish larger objectives while reducing the risk to the mission and to the constituent platforms and personnel.

Figure 1 illustrates the behavior of a package. The missions:

- travel from their respective units to come together outside of enemy airspace, at some designated *entry location*;

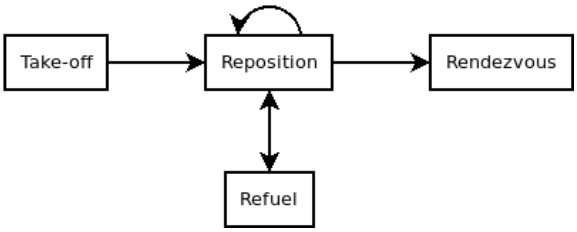

Figure 2: Structure of a mission's ingress COA

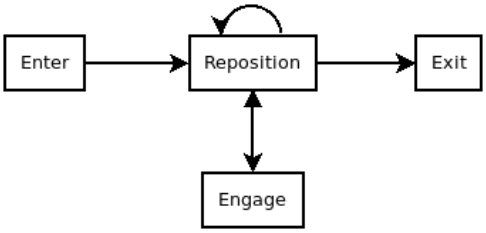

Figure 3: Structure of a package COA

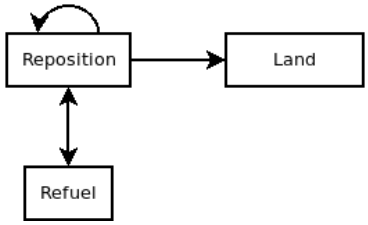

Figure 4: Structure of a mission's egress COA

- enter enemy airspace together;
- traverse enemy airspace together to engage assigned *targets*, one at a time;
- exit enemy airspace at some designated *exit location*;
- and disband, with the each mission returning to its own unit.

The activities of a mission from take off to rendezvous constitute the mission's *ingress Course of Action* (ingress COA); likewise, the activities of a mission from the enemy airspace exit location to landing constitute its *egress COA*. Each mission has its own ingress COA and egress COA.

In contrast, the activities of all of the missions between rendezvousing at the entry locaion and disbanding at the exit are modeled collectively as a single *package COA*. At any given moment, the package is performing some package-level activity, such as engaging a target — as part of that activity, some of the package's strike missions will be deploying munitions against the target (the remainder of the package's strike missions presumably were not assigned the target in question), while the package's EW missions might be jamming nearby SAM sites and the package's OCA missions might be patrolling for enemy aircraft. These mission-specific activities that occur while the package is formed up are not separately modeled in this report. Rather, this report models the package-level activities.

3.1. Activities and COAs

A COA provides details of what a mission or package does, in terms of a sequence of *contiguous*, *sequential activities*, each of which takes the mission or package from a designated start location at a designated start time, to a designated end location at a designated end time, possibly while carrying out some task (such as refueling or engaging a target). An activity's start and end location may be the same.

Specifically:

- A COA C is a non-empty sequence of activities $[A_i]$, $1 \leq i \leq n$.

- The origin and destination of a mission's take-off activity must be the location of the mission's unit.
- The origin and destination of a refueling activity must be the location of the activity's designated aerial refueling track.
- The origin and location of a target engagement must be the target's location.
- The origin and destination of a mission's landing activity must be the location of the mission's unit.

Figure 5: Constraints on activities' locations

- The duration of a take-off activity is taken to be some constant.
- The minimum duration of a mission repositioning activity is determined by the distance traveled and the speed. A single repositioning activity occurs along the great circle from the origin to the destination. The distance traveled is thus the great circle distance from origin to destination. The speed is determined by the mission's platform type.
- The start time of a refueling activity must be synchronized between the mission that is receiving the fuel and the tanker mission that is providing the fuel.
- The duration of a refueling activity is determined by the receiving platform type, the tanker type and the number of receiving sorties.
- The duration of a mission's rendezvous activity is not determined by the activity itself. Rather, the rendezvous's start time is determined by when the mission reaches the rendezvous location; and the rendezvous's end time is the soonest time at which all of the package's missions arrive at the rendezvous.
- The duration of a landing activity is taken to be some constant.

Figure 6: Constraints on the start/end time and duration of *mission* activities

- The duration of a package entry activity is determined by the distance traveled and the speed. The speed of a package is taken to be the speed of its slowest platform, since the package's missions are modeled as traveling together.
- The minimum duration of a package repositioning activity is determined by the distance traveled and the package speed.
- The duration of a target engagement is taken to be some constant, Δ_{E}.
- The start time of a target engagement activity must not be earlier than the target's no-earlier-than time (NET).
- The end time of a target engagement activity must not be later than the target's no-later-than time (NLT).
- The duration of a package exit activity is determined by the distance traveled and the package's speed.

Figure 7: Constraints on the start/end time and duration of *package* activities

- An Activity A is a discrete action executed in time interval $[\mathsf{st}(A), \mathsf{et}(A))$.
- Before joining a strike package, a mission's possible activities are: take off, reposition, refuel, and rendezvous (with other missions in a package at the package's entry location). Figure 2 shows the allowed transitions between activities. The activities in the ingress COA are executed in friendly airspace.
- While missions are formed up as a package, they are modeled as a single, coherent entity. The possible activities of a package are: enter into enemy airspace, reposition, engage a target, and exit from enemy airspace. See Figure 3. The activities in the package COA are executed in enemy airspace.
- After leaving a strike package, a mission's possible activities are: reposition, refuel and land. See Figure 4. The activities in the egress COA are executed in friendly airspace.
- Each activity A has a start time, end time and duration[2], given by $\mathsf{st}(A)$, $\mathsf{et}(A)$ and $\mathsf{dur}(A)$, where $\mathsf{dur}(A) = \mathsf{et}(A) - \mathsf{st}(A)$. The start time, end time and duration of an activity are constrained or determined by the type of activity — see Figures 6 and 7.
- Each activity A has an origin and a destination, given by $\mathsf{orig}(A)$ and $\mathsf{dest}(A)$. The origin or destination of some activities are constrained — see Figure 5.
- Consecutive activities in a COA C are required to be adjacent in time and location, with each starting where and when its predecessor, if any, ended: $\mathsf{et}(A_i) = \mathsf{st}(A_{i+1})$ and $\mathsf{dest}(A_i) = \mathsf{orig}(A_{i+1})$ for $i = 1, \ldots, n-1$, where n is the number of activities in C.
- The start time, duration, end time, start location and end location of a COA C are determined by the properties of the first activity A_1, and the last activity, A_n, in the COA:

$$\begin{array}{lcl} \mathsf{st}(C) & = & \mathsf{st}(A_1) \\ \mathsf{orig}(C) & = & \mathsf{orig}(A_1) \\ \mathsf{et}(C) & = & \mathsf{et}(A_n) \\ \mathsf{dest}(C) & = & \mathsf{dest}(A_n) \\ \mathsf{dur}(C) & = & \mathsf{et}(C) - \mathsf{st}(C) \quad = \quad \sum_{i=1}^{n} \mathsf{dur}(A_i) \ . \end{array}$$

In this report, routing is represented as a sequence of way-points. We assume the missions fly a straight line between two consecutive way-points. The route representation is embedded in the activity sequence representation. Additional routing details — such as from which direction, at what altitude and at what speed a mission should approach a target — are not modeled.

3.2. Fuel Feasibility

Each platform consumes fuel at a rate (per unit time) that depends on its speed, its altitude, its weight, its configuration (which affects wind resistance), wind speed and direction, etc. In this report, a simplified model is used — for a given platform, the rate of fuel consumption is determined only by a combination of speed and altitude. Fuel consumption is computed and maintained for each individual mission in the package, and it is based on the number and type of platforms assigned to the mission. Each activity (e.g., a flight segment between two consecutive way points) can have its own speed and altitude, but we assume they are constant for the duration of the activity. Moreover:

- Outside of enemy airspace, a mission is usually modeled as traveling at a constant speed and altitude, and thus having a constant rate of fuel consumption. Outside enemy airspace, the speed and altitude for a mission are based only on the selected platform capabilities.
- Within enemy airspace, we assume all missions in the package are flying exactly the same route. While the missions are positioning from one location to another, all missions fly at the same constant speed and altitude. This speed and altitude are usually defined by the less capable platform in the package. The fuel consumption is computed based on each mission's platform. (Note that different platforms flying at the same speed may have different rates of fuel consumption.) During target engagement, each mission may have a different set of parameters used to compute fuel consumption

[2] In this report, time is treated as an Integer or Long.

since speeds and altitudes will be different for different missions. In our models, we use the same speed and altitude for all activities within enemy airspace.

Thus, for a given platform in a given package, there are two main rates of fuel consumption: the rate outside of enemy airspace, and the rate inside enemy airspace.

Given these rates, and the rate at which the platform onloads fuel during aerial refueling, the platform's fuel level can be computed at any time during a COA:

- A platform takes off with some initial fuel load that depends on the unit. For example, a platform from a land unit may take off with a full load of fuel, while a platform from a carrier unit might take off with a minimal fuel load (to reduce its take-off weight) and immediately proceed to a tanker for aerial refueling.
- For all activities except refueling, the platform consumes fuel, per unit time, at a rate that depends on the platform type and location (i.e., outside/inside enemy airspace). Thus, for an activity with start time st and end time et, the fuel level at time t, where $\mathsf{st} \leq t \leq \mathsf{et}$, is

  $$\mathsf{fuel}(t) = \mathsf{fuel}(\mathsf{st}) - r(t - \mathsf{st})$$

  where r is the rate of fuel consumption.
- For refueling activities, the platform consumes fuel the same as other activities, at a rate r, but it also onloads fuel. In this report, onloading is modeled as occurring continuously throughout a mission's refueling activity, at a rate, r_o that depends on the platform type and the tanker type. Thus:

  $$\mathsf{fuel}(t) = \mathsf{fuel}(\mathsf{st}) + (r_o - r)(t - \mathsf{st}) \ .$$

Throughout a mission, a platform's fuel level is prohibited from falling below some reserve threshold.

3.3. Time Feasibility

The time constraints on a mission have been noted above, and are summarized here.

- Consecutive activities are adjacent in time.
- Each activity, except rendezvous, has a duration that is determined by the activity type and details.
- The duration of a mission's rendezvous is determined by when the specific mission arrives at the entry location (the start time of the rendezvous) and when the last of the package's missions arrives at the rendezvous location (the end time of the rendezvous).
- A target engagement must take place entirely within the target's NET-NLT time window.
- The start time of the mission and the end time of the mission must lie within the time window of the unit's go.

3.4. Risk

The risk to a strike mission m in a package p at time t is determined by:

- the threats for which the package is in range, $\mathsf{threats}(p, t)$;
- the type of platform in the mission, $\mathsf{platform}(m)$;
- the SEAD/EW/OCA support missions in the package, $\mathsf{support}(p)$.

For example, the risk to various platform types from various threat types may be defined in a matrix:

	SAM01	SAM02	SAM03
AC-1	MEDIUM	MEDIUM	HIGH
AC-2	LOW	LOW	MEDIUM

where the risk is quantified as NONE, LOW, MEDIUM, HIGH or EXTREME.

Multiple risks may be combined through promotion rules:

Individual Risks	Combined Risk
2+ HIGH	EXTREME
3+ MEDIUM	EXTREME
2+ MEDIUM	HIGH
3+ LOW	HIGH
2+ LOW	MEDIUM

That is, if a mission is within range of two or more threats, each of which is classified as HIGH risk for its platforms, then the risk to the mission is promoted to EXTREME; and so on.

Risk is reduced by the presence of EW/SEAD/OCA support missions in the package. For example, 4 or more SEAD platforms may reduce the risk by two levels (from EXTREME to MEDIUM, for example), while 2 or more EW platforms may reduce the risk by only one level (from EXTREME to HIGH, for example).[3]

The result of such analysis is a risk level estimate for the mission at each time t. The overall risk to the mission may be taken to be the highest of its instantaneous risks.

A package may be constrained in terms of the maximum risk that is acceptable for any of its missions:

$$\forall p : \mathsf{Package}, m : \mathsf{Mission} \in \mathsf{missions}(p) \cdot \mathsf{risk}(p, m) \leq R$$

where R is some risk level threshold.

3.5. Strike Point Assignment

A package p assigns a strike point s to the mission that will engage it, together with the munition option that will be used. For example, Table 2 shows strike points S0001a and S0001b (in the HQ target) assigned to mission FS1a, with each strike point to be struck with 2 GBU03s; etc.

Target	Strike Point	Mission	Munitions
HQ	S0001a	FS1a	2 GBU03
	S0001b	FS1a	2 GBU03
Airfield	S0002a	FS2a	8 GBU02
	S0002b	FS2a	8 GBU02

Table 2: Assignment of strike points to missions & munitions

The assignments can be formulated as follows:

- The assigned mission is given by:

$$\mathsf{assignedMission} : \mathsf{Package} \times \mathsf{StrikePoint} \rightarrow \mathsf{Mission} \ .$$

  The assigned mission must be one of the package's missions:

$$\forall p : \mathsf{Package}, s : \mathsf{StrikePoint} \in \mathsf{assignedStrikePoints}(p) \cdot \mathsf{assignedMission}(p, s) \in \mathsf{missions}(p)$$

  where $\mathsf{assignedStrikePoints}(p)$ is the set of assigned strike points in package p.

- The assigned munition option is given by:

$$\mathsf{assignedMunitionOption} : \mathsf{Package} \times \mathsf{StrikePoint} \rightarrow \mathsf{MunitionOption} \ .$$

  The munition option assigned must be one of the strike point's munition options:

$$\forall p : \mathsf{Package}, s : \mathsf{StrikePoint} \in \mathsf{assignedStrikePoints}(p) \cdot \mathsf{assignedMunitionOption}(p, s) \in \mathsf{munitionOptions}(s) \ .$$

[3] Risk analysis could presumably be cast in terms, say, of the probability of a platform being destroyed by a particular type of threat for each second that the platform is within range of the threat, although it is not clear that such data are available.

Also, the munition must be part of the assigned mission's SCL:

$$\forall p : \mathsf{Package}, s : \mathsf{StrikePoint} \in \mathsf{assignedStrikePoints}(p) \cdot \\ \mathsf{munition}(\mathsf{assignedMunitionOption}(p, s)) \in \mathsf{munitions}(\mathsf{scl}(\mathsf{assignedMission}(p, s)))$$

where $\mathsf{munitions}(\mathsf{scl})$ gives the set of munition types in the SCL.

Implicit in this formulation is that each strike point is engaged at most once *by a given package*. However, multiple packages can engage the same strike point.

An additional constraint is that for a given mission, the total munitions assigned to strike points is less than the total munitions carried by the mission. This can be determined for each munition type:

$$\forall p : \mathsf{Package}, m : \mathsf{Mission} \in \mathsf{missions}(p), M : \mathsf{Munition} \cdot \\ \sum_{s \in S(m,M)} \mathsf{quantity}(\mathsf{assignedMunitionOption}(s)) \leq \mathsf{quantity}(\mathsf{scl}(m), M)$$

where $\mathsf{quantity}(L, M)$ is the quantity of munitions of type M in SCL L, and $S(m, M)$ is the set of strike points engaged by mission m with munition M:

$$S(m, M) = \{s : \mathsf{StrikePoint} \mid \\ s \in \mathsf{assignedStrikePoints}(p) \\ \wedge\, \mathsf{assignedMission}(p, s) = m \\ \wedge\, \mathsf{munition}(\mathsf{assignedMunitionOption}(p, s)) = M\} \ .$$

As a matter of doctrine, a package may be constrained to engage a given target using only one mission (i.e., all strikes by the package against the target's strike points come from the same mission):

$$\forall p : \mathsf{Package}, s_1, s_2 : \mathsf{StrikePoint} \in \mathsf{assignedStrikePoints}(p) \cdot \\ \mathsf{target}(s_1) = \mathsf{target}(s_2) \implies \mathsf{assignedMission}(p, s_1) = \mathsf{assignedMission}(p, s_2) \ .$$

3.6. Metrics

Various cost metrics can be defined for a (feasible) strike package:

- the number of sorties in the package;
- the number of munitions assigned to strike points;
- the amount of fuel used;
- the total duration from earliest take-off to latest landing (the makespan).

Various performance metrics can also be defined:

- the number of strike points struck (perhaps weighted by the probability of destruction);
- the risk to each mission.

It is not useful to try to combine these metrics into simplistic trade-offs (e.g., combining all of the cost metrics into a single number).

Nonetheless, it may be useful to identify when a package is using more resources that is *apparently* necessary, such as:

- containing a strike mission that has no assigned strike points;
- containing a strike mission whose assigned strike points could feasibly all be assigned to other missions in the package;
- containing SEAD/EW missions when the package does not come within range of any ground threats;
- containing OCA missions when the package does not come within range of any aerial threats;
- refueling when there is sufficient fuel to remain above the required threshold without refueling.

However, such apparent wastefulness may be intentional, to allow for redundancy (e.g., a platform suffers a malfunction and must return to base) or changes to the environment (e.g., a new SAM site is discovered). Thus, over-provisioning a strike package is not considered an error, though it is something that should be noted.

3.7. Summary

A strike package is defined in terms of the following:

- An entry location and an exit location.
- The missions, where each mission designates:
  - The go providing the sorties for the mission, which implies the unit, the platform type, the per-sortie SCL, the go's start and end times, and the take-off and landing locations.
  - The number of sorties.
- An ingress COA for each mission, which includes the mission's take-off location, any aerial refueling to be done during ingress, the mission's arrival time at the package's entry location.
- The package COA, which includes:
  - The package's time of entry and exit.
  - The package's route through enemy airspace.
  - The targets assigned to the package.
  - A *time over target* (ToT) for each target.
- An egress COA for each mission, which includes any aerial refueling to be done on egress, and the mission's landing time.
- The strike point assignment (i.e., which mission will engage each strike point, and with what type and quantity of munition).

A *feasible* strike package must satisfy various constraints, including:

- Each mission's take-off and landing times must be within the go's time window.
- The total number of sorties assigned for a given go must not exceed the number of sorties available from that go.
- Each target strike must occur within the target's NET-NLT time window.
- Each strike must use one of the strike point's munition options.
- The total munitions required for all of the strikes assigned to a given mission must not exceed the munitions carried by the mission (on a per-munition-type basis).
- Sufficient time must be allowed for each activity (such as repositioning, refueling or engaging a target).
- Each mission's fuel level must remain above a certain threshold throughout the mission.
- The risk to a mission must remain below some threshold, throughout the mission.

4. Relaxations of Feasibility: Necessary Conditions

The preceding section detailed the information that must be provided to fully define a strike package, and constraints that characterize feasibility in platforms, munitions, time and fuel.

This section considers *necessary conditions* that follow logically from feasibility. In many cases, necessary conditions can be evaluated more efficiently than full feasibility, or indeed can be evaluated on partially defined packages — they may thus be used to optimize automated search processes by pruning off branches of the search space that cannot contain feasible packages.

For example, consideration of fuel feasibility may allow certain targets to be determined to be incompatible with certain pairs of entry and exit locations, in that no platform would have sufficient fuel to travel from the entry to the target and thence to the exit. (Recall that refueling is not possible inside enemy airspace.) Note that this applies regardless of what other targets might be in a putative package

using these entry and exit locations, what order the targets might be visited, what munitions might be used to strike the target, what time the package might enter enemy airspace, etc.

Thus, if the search process has already determined to use these particular entry/exit locations, then some subset of the targets can be excluded from the search space; or *vice versa*.

Moreover, necessary conditions, by being less dependent on details, often provide better explanations to a human operator during interactive package construction. For example, if the operator chooses an entry/exit pair, then incompatible targets can be de-emphasized in the operator's interface with a simple explanation provided, namely that they are out of fuel range for the chosen entry/exit pair. The operator knows that the excluded targets cannot be made feasible by tweaking such decisions as which goes provide the sorties, what route they follow, where they refuel on ingress, etc.

It is important to distinguish those constraint violations that can be repaired by *extending* a partially defined package — by adding more resources — versus those constraints that cannot be so repaired. Examples of constraint violations that might be repaired by extension include:

- A package having insufficient munitions to cover all strike points might be repaired by adding more strike mission to the package or increasing the number of sorties in the existing strike missions.
- The risk to a mission being too high might be repaired by adding more support, or by routing around threats.

Examples of constraint violations that cannot be repaired by extension include:

- Individual targets that are infeasibly far from the entry and exit locations, as discussed above.
- Pairs of targets that are too far apart to appear together in any package, due to time or fuel considerations.
- Pairs of target whose time windows (NETs and NLTs) are too far apart for them to appear together in any package.
- Goes that are incompatible because their time windows and distances from the entry and exit locations preclude their being in enemy airspace at the same time (see below).

For violations that cannot be repaired by extension, the only solution is to retract some decision that has already been made, such as which entry and exit locations are to be used, which goes are to provide sorties, or which targets are to be struck.

4.1. Compatibility between Munition Options and SCLs

Each strike point has a set of munition options that designate the munition types (and quantities) that can be used to destroy the strike point.

An SCL has one or more munition types. An SCL is incompatible with a strike point if it does not have munition types corresponding to one or more of the strike point's munition options.

Each go is able to provide certain SCLs. A go is incompatible with a strike point if all of the SCLs that it can provide are incompatible with the strike point.

By extension, a unit is incompatible with a strike point if all of its goes are incompatible with the strike point.

These constraints can be extended to a target by quantifying over the target's strike points: an SCL is incompatible with a target if it is incompatible with all of the target's strike points, and so on.

A package is incompatible with a target if all of the goes providing missions to the package are incompatible with the target. (Note that a package might be made compatible by adding additional missions with compatible SCLs.)

4.2. Compatibility between Goes/Units and Entry & Exit Locations

For a given unit and given entry & exit locations, the following distances can be statically determined (i.e., without the context of any particular package):

- the minimum distance required to navigate from the unit to the entry, navigating around enemy airspace as necessary;
- the minimum distance traveled in enemy airspace (i.e., the distance from the entry location to the exit location);
- the minimum distance required to navigate from the exit to the unit, navigating around enemy airspace as necessary.

The total minimum round-trip distance — from unit, to entry, to exit, and back to the unit — can also be determined from these distance.

For each of these distances, a minimum duration can also be determined based on the speed of the unit's platforms (outside of and within enemy airspace).

For a given go, the following upper bounds can be statically determined:

- The time that the go's platforms can spend in enemy airspace is bounded by the platform's maximum fuel capacity (possibly augmented with external fuel tanks, depending on the SCL) and rate of fuel consumption, since refueling inside enemy airspace is not possible.
- Moreover, as a matter of doctrine, a mission may be required to enter enemy airspace with some minimum fuel level (e.g., 90% of its capacity) and to exit enemy airspace with some minimum fuel level (e.g., 30%). In this case, a reasonable rule of thumb is that the mission cannot expend more fuel in enemy airspace than the difference between the two required levels (e.g., 60%). This reduces the time the mission can spend in enemy airspace and the distance it can travel.
- The duration of any mission from the go is bounded by the duration of the go's time window.

Thus, if any of the minima identified above exceed the corresponding upper bound, the go is incompatible with the entry & exit locations.

By extension, a unit is incompatible with given entry & exit locations if all of its goes are incompatible with the entry & exit locations.

4.2.1. Required Ingress and Egress Refueling

It may be possible to determine statically that refueling will be necessary for a mission on ingress, based solely on the mission's unit and the entry & exit locations, regardless of a package's targets.

Consider a mission that travels from its unit to the entry location and then to the exit location. The mission cannot refuel between the entry and the exit. The closest refueling location after the exit is not known exactly, so as a bound assume that the mission can refuel as soon as it exits enemy airspace.

Then ingress refueling is definitely required if the mission's fuel level would fall below the required reserve level if the mission were to travel from its unit to the entry and then to the exit without refueling.

More generally, if traveling from its unit to the entry and then to the exit requires a quantity of fuel that is f times the fuel capacity of a mission's platforms, then the mission will need to refuel $\llcorner f \lrcorner$ times during ingress (at least).

Alternatively, if a mission is doctrinally required to enter enemy airspace with some minimal fuel level (e.g., 90%), then ingress refueling is definitely required if traveling from unit to entry would reduce the mission's fuel level below this minimum.

Likewise, consider a mission traveling from the entry location to the exit location and then to its unit. The best case assumption is that the mission has 100% fuel at the entry. Under this assumption, if the mission cannot travel to the exit and then to its unit without its fuel level falling below the required reserve level, then the mission must refuel on egress.

If the minimum fuel required from entry to exit to unit is f times the platform's fuel capacity, then $\llcorner f \lrcorner$ egress refuelings are required (at least).

The increase in ingress duration and distance caused by refueling is not known precisely until all of the details of the refuelings are determined (e.g., it depends on the locations of tankers during the mission's ingress/egress and how many other missions they need to refuel during the same period). Nonetheless, some reasonable allowance can be made based on historical data.

This can be used to increase the minima in the previous section to strengthen the constraints.

4.3. Compatibility between Goes and Entry & Exit Locations

Since a mission is required to take off and land within its go's time period, the analysis of the preceding section can be recast in terms of the earliest time that a mission from a given go, taking off at the go's start time, can arrive at a given entry location, and the latest time that a mission from a given go can leave a given exit location if it is to land before the go's end time.

The take off/landing constraint is:

$$\forall g : \mathsf{Go}, m : \mathsf{Mission} \in \mathsf{missions}(g) \cdot \mathsf{st}(g) \leq \mathsf{st}(m) \leq \mathsf{et}(m) \leq \mathsf{et}(g)$$

where $\mathsf{st}(g)$ is the start time of go g, and $\mathsf{et}(g)$ is the go's end time.

Given entry location e and exit location x for a package, the *earliest* that a mission from a particular go can arrive at the entry, $\mathsf{earliestEntryTime}(e, x, g)$, depends on:

- the go's start time;
- the travel time from the unit to the entry, which depends on:
  - the distance, allowing for navigation around enemy airspace;
  - the speed of the go's platforms;
  - any refueling that *must* be done on ingress (see Section 4.2.1).

Similarly, the *latest* time that a mission can leave from the exit, $\mathsf{latestExitTime}(e, x, g)$, depends on:

- the go's end time;
- the travel time from the exit to the unit, which depends on:
  - the distance, allowing for navigation around enemy airspace;
  - the speed of the go's platforms;
  - any refueling that *must* be done on egress (see Section 4.2.1).

Consequently, for each pair of entry/exit locations, each go has a *maximum* time window

$$\mathsf{maxTimeWindow}(e, x, g) = [\mathsf{earliestArrivalTime}(e, x, g), \mathsf{latestDepartureTime}(e, x, g)]$$

during which its missions can be in enemy airspace. If a go's maximum time window is empty (i.e., the latest exit time precedes the earliest entry time), then the go is incompatible with the entry & exit locations.

If a package contains missions from multiple goes, then the maximum time window during which the package can be in enemy airspace is the intersection of the goes' maximum time windows:

$$\forall p : \mathsf{Package} \cdot \mathsf{maxTimeWindow}(p) = \bigcap_{m \in \mathsf{missions}(p)} \mathsf{maxTimeWindow}(\mathsf{entry}(p), \mathsf{exit}(p), \mathsf{go}(m))$$

where $\mathsf{missions}(p)$ is the set of missions in package p, and $\mathsf{go}(m)$ is the go that supplies mission m.

The start of the maximum time window, denoted $\mathsf{earliestEntryTime}(p)$, is the earliest time that the package can enter enemy airspace, and is the latest of the goes' earliest arrival times.

Likewise, the end of the maximum time window, denoted $\mathsf{latestExitTime}(p)$, is the latest time that the package can leave enemy airspace, and is the earliest of the goes' latest departure times.

There are several consequences to this:

- Goes g_1 and g_2 are incompatible, for given entry and exit locations, if their maximum time windows do not overlap.
- By extension, goes g_1 and g_2 are incompatible if they are incompatible for all entry/exit pairs. This is always the case for goes from the same unit, since a unit's goes cannot overlap.
- Adding more goes to a package can never increase the time that the package can spend in enemy airspace. Adding a go may move the earliest entry time later and may move the latest departure time earlier. In particular, if a package is infeasible due to incompatible goes, then adding more goes cannot make the package feasible.

4.4. Compatibility between Goes and Targets

Suppose that a mission from a go, g, is to visit some target, T via entry location, e, and exit location, x.

The earliest time that the mission can arrive at the target is the earliest time that the mission can arrive at the entry location, plus the time to travel directly to the target from the entry:

$$\begin{aligned}&\mathsf{earliestArrivalTime}(e, x, g, T) = \\ &\quad \mathsf{earliestEntryTime}(e, x, g) + \Delta_{\mathsf{F}}(\mathsf{speed}_E(\mathsf{platform}(g)), e, \mathsf{location}(T))\end{aligned}$$

where $\mathsf{speed}_E(\mathsf{platform})$ is the speed of the given platform in enemy airspace, and $\Delta_{\mathsf{F}}(s, l_1, l_2)$ is the duration of a flight at speed s from location l_1 to l_2.

Other factors, such as navigating to and engaging other targets before T, can only cause the mission to arrive later at T.

The latest time that the mission can depart from the target is the latest time that the mission can depart the exit, minus the direct flight time from the target to the exit:

$$\begin{aligned}&\mathsf{latestDepartureTime}(e, x, g, T) = \\ &\quad \mathsf{latestExitTime}(e, x, g) - \Delta_{\mathsf{F}}(\mathsf{speed}_E(\mathsf{platform}(g)), \mathsf{location}(T), x) \ .\end{aligned}$$

Other factors, such as navigating to and engaging other targets after T, can only require the mission to leave T earlier.

Thus a maximum time window can be determined on when a mission from the go can engage the target:

$$\begin{aligned}&\mathsf{maxTimeWindow}(e, x, g, T) = \\ &\quad [\mathsf{earliestArrivalTime}(e, x, g, T), \mathsf{latestDepartureTime}(e, x, g, T)] \ .\end{aligned}$$

If a go's maximum time window does not intersect the target's NET/NLT engagement window, then no mission from the go can engage the target (via the given entry and exit locations), regardless of which other targets are engaged or which other goes provide missions to the package. More precisely, the intersection of the maximum time window and the target's NET/NLT window must be long enough to allow the target to be engaged:

$$|\,\mathsf{maxTimeWindow}(e, x, g, T) \cap \mathsf{timeWindow}(T)| \geq \Delta_{\mathsf{E}} \ .$$

Moreover, if a go is incompatible with a given target's time window, then no package containing missions from that go can engage the target since adding missions from other goes cannot widen the go's window on the target.

More generally, for a package, p, engaging some target, T, the earliest time that the package can arrive at the target is:

$$\begin{aligned}&\mathsf{earliestArrivalTime}(p, T) = \\ &\quad \mathsf{earliestEntryTime}(p) + \Delta_{\mathsf{F}}(\mathsf{speed}(p), \mathsf{entry}(p), \mathsf{location}(T)) \ .\end{aligned}$$

If the package arrives at the target before the target's NET, then the package must wait until the NET before engaging the target. Let $\mathsf{est}(p, T)$ (*Earliest Start Time*) denote the earliest time that the package can begin the engagement:

$$\mathsf{est}(p, T) = \mathsf{max}(\mathsf{earliestArrivalTime}(p, T), \mathsf{net}(T)) \ .$$

The latest time that the package can depart the target is:

$$\begin{aligned}&\mathsf{latestDepartureTime}(p, T) = \\ &\quad \mathsf{latestExitTime}(p) - \Delta_{\mathsf{F}}(\mathsf{speed}(p), \mathsf{location}(T), \mathsf{exit}(p)) \ .\end{aligned}$$

A target's engagement cannot continue beyond the target's NLT. Let $\mathsf{lft}(p, t)$ (*Latest Finish Time*) denote the latest time that the package can complete engagement of the target:

$$\mathsf{lft}(p, T) = \mathsf{min}(\mathsf{latestDepartureTime}(p, T), \mathsf{nlt}(T)) \ .$$

The maximum time window for the package to engage the target is:

$$\mathsf{maxTimeWindow}(p, T) = [\mathsf{est}(p, T), \mathsf{lft}(p, T)] \ .$$

For the target to be compatible with the package, the maximum time window must be long enough:

$$|\,\mathsf{maxTimeWindow}(p, T)| \geq \Delta_{\mathsf{E}} \ .$$

Appendix A develops this reasoning further.

4.5. Compatibility between Two Targets

A pair of targets can be incompatible in several ways.

Consider some mission in a package that is to engage both targets. The distance between the targets determines a lower bound on the distance the mission must travel in enemy airspace. Together with the mission's speed, the distance also determines a lower bound on the time the mission must spend in enemy airspace.

However, the mission's distance and time in enemy airspace are bounded from above by the mission's maximum time window in enemy airspace (see Section 4.4) and the maximum fuel that the mission can use in enemy airspace (since refueling is not possible in enemy airspace).

Thus, if either lower bound exceeds either upper bound, then the mission cannot engage both targets.

If the two targets are incompatible for missions from all goes, then they are statically incompatible.

4.6. Constraints on Strike Point Assignment

A fully defined package assigns a mission and munitions to each strike point in a package's targets, resulting in a probability of destruction for each strike point — see Section 3.5.

For a given strike point, s, the minimum *number* of munitions required is the minimum quantity in any of the strike point's munition options:

$$\mathsf{minimumMunitionsRequired}(s : \mathsf{StrikePoint}) = \min_{m \in \mathsf{munitionOptions}(s)} \mathsf{quantity}(m) \ .$$

For a given set of strike points, S, the minimum number of munitions required is simply the sum of the minima of the individual strike points:

$$\mathsf{minimumMunitionsRequired}(S : \mathsf{Set\,StrikePoint}) = \sum_{s \in S} \mathsf{minimumMunitionsRequired}(s) \ .$$

For a strike package, p, the total (air-to-ground) munitions carried is simply the sum of the munitions carried by the package's missions:

$$\mathsf{numberOfMunitions}(p : \mathsf{Package}) = \sum_{m \in \mathsf{missions}(p)} \mathsf{sorties}(m) \times \mathsf{numberOfMunitions}(\mathsf{scl}(m))$$

where $\mathsf{numberOfMunitions}(L)$ is the number of air-to-ground munitions in the SCL L:

$$\mathsf{numberOfMunitions}(L : \mathsf{SCL}) = \sum_{M:\mathsf{Munition}} \mathsf{quantity}(L, M) \ .$$

A necessary condition for all of a package's strike points to be assigned is that the total munitions carried by the package be at least the minimum number of munitions required:

$$\mathsf{numberOfMunitions}(p) \geq \mathsf{minimumMunitionsRequired}(\mathsf{strikePoints}(p)) \ .$$

Note that violation of this constraint might be repaired by adding additional strike sorties.

Appendix B details the formulation of strike point assignment as an integer linear programming problem.

5. Conclusions

This report details the structure of and constraints on strike packages. It develops necessary conditions that might be useful in optimizing automated or interactive construction of packages.

The major assumption used in this report is that a package can be modeled as a single entity that forms up at its entry location, travels coherently from target to target through enemy airspace, and ultimately to its exit location, where it disbands. Minor deviations from this assumption should not be critical; e.g., if the package engages two nearby targets simultaneously.

Other limitations include:

- *Airspace control orders* (ACOs) are not considered. ACOs define allowed or forbidden areas of airspace, and thus affect the route that a package follows from one location to the next, e.g., between successive targets. Thus, ACOs affect the distances and times between locations. This report does not make any critical assumptions about the distances and times (e.g., it does not assume that they are Euclidean) so incorporating the effects of ACOs on routing should not require a significant extension.
- Routing is coarse, but this is a decision more than a limitation. Detailed routing can be determined at the tactical or unit level or by the package commander.
- The effects of topography on threat analysis are not considered. For example, a mission may be within range of a SAM site, but the SAM site may not be able to see the mission due to intervening mountains.
- Altitude is approximated and not precisely computed. Altitude affects fuel consumption rates and the range of ground-based threats.
- Bomber missions are not included as part of strike packages. Bomber missions differ structurally from fighter missions in that a bomber mission may travel to a *launch basket* (a designated location) from which it can launch cruise missiles against targets over significant distances. In addition, bomber missions may have much longer ingress/egress routes and carry many more munitions.

A. Feasible Ordering of Targets in a Package

Consider the following scenarios:

- One target, T_1, in a strike package has a no-later-than time of 1015. A second target, T_2, has a no-earlier than time of 1045. If the package were to visit and engage T_2 before T_1, there would be no way it could arrive at T_1 before T_1's deadline. In contrast, if the package were to visit T_1 before T_2, then it *may* be possible to complete the engagement of T_1, and travel to and complete the engagement of T_2. Thus, any feasible route must visit T_1 before T_2.
- One target, T_1 is close to the entry while a second target T_2 is close to the exit. A route that visits T_1 before T_2 is likely to be much shorter than a route that visits T_2 before T_1, since the latter must traverse the entire entry-exit distance twice. See Figures 8 and 9. Given that the routes are constrained by the package's earliest arrival time and latest departure time, it may be possible to show that *any* route that visits T_2 before T_1 will violate the constraints. Thus, any feasible route must visit T_1 before T_2.

Figure 8: Shorter route for targets near the entry and exit

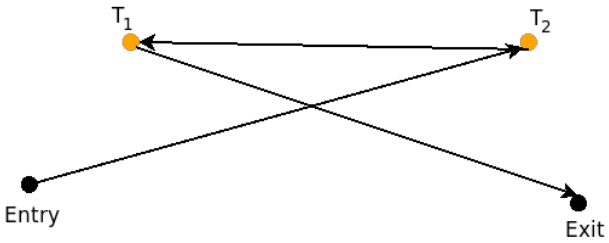

Figure 9: Longer route for targets near the entry and exit

A.1. Precedence

These observations can be formalized in terms of a (strict) partial order, $\prec$ on the targets and the entry and exit locations, where $a \prec b$ means that the package must visit a before b.

Since the partial order includes the entry and exit locations as well as targets, define a *node* to be either a target in some package, p, or the package's entry location, e, or exit location, x. See Figure 10.

The following can be observed about the partial order for a strike package:

- The entry precedes each target.
- The entry precedes the exit.
- Each target precedes the exit.
- The partial order is transitive: $N_1 \prec N_2 \wedge N_2 \prec N_3 \implies N_1 \prec N_3$.
- The partial order is anti-symmetric: it cannot be that $N_1 \prec N_2$ and $N_2 \prec N_1$. This also precludes cycles.

A.2. Earliest/Latest Start/Finish Times with Respect To Precedence

Let $\mathsf{est}_\prec(N)$ denote the earliest time that the package, p, can begin its activity at node N, consistent with the partial order, and let $\mathsf{eft}_\prec(N)$ denote the earliest time that the package can complete the activity.

- If the node is a target, then the activity is engagement of the target and $\mathsf{eft}_\prec(N) = \mathsf{est}_\prec(N) + \Delta_\mathsf{E}$.

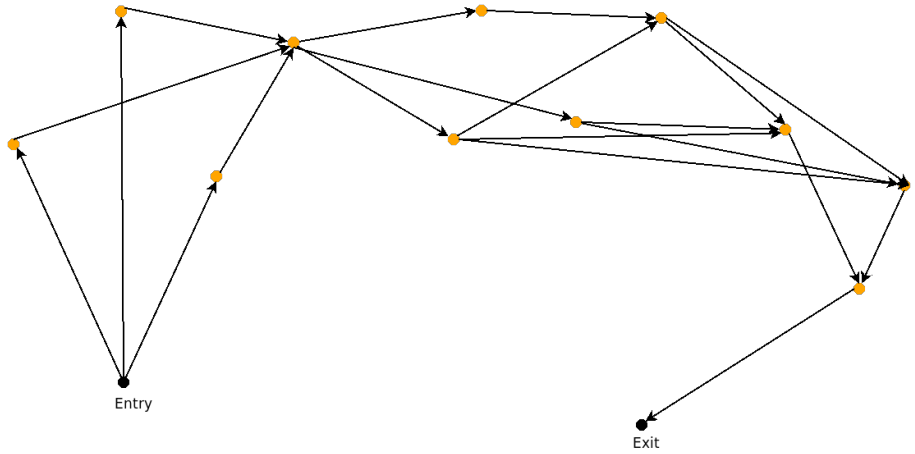

Figure 10: Node precedence for a package: $N_1 \prec N_2$ is indicated by an arrow from N_1 to N_2 (transitive constraints are not shown)

- If the node is the entry, then $\mathsf{est}_\prec(N) = \mathsf{eft}_\prec(N) = \mathsf{earliestEntryTime}(p)$. It is also convenient to define $\mathsf{net}(N) = -\infty$ and $\mathsf{nlt}(N) = \infty$.
- If the node is the exit, then $\mathsf{est}_\prec(N) = \mathsf{eft}_\prec(N)$. It is also convenient to define $\mathsf{net}(N) = -\infty$ and $\mathsf{nlt}(N) = \infty$.

Now if $N_1 \prec N_2$, the package must visit N_1 before N_2 and complete its activity at N_1 before beginning to travel to N_2. Thus, the package cannot begin to travel to N_2 until $\mathsf{eft}_\prec(N_1)$, and must complete the flight from N_1 to N_2 before it can start its activity at N_2. So:

$$N_1 \prec N_2 \implies \mathsf{eft}_\prec(N_1) + \Delta_\mathsf{F}(N_1, N_2) \leq \mathsf{est}_\prec(N_2)$$

where, for brevity, the speed parameter is omitted from $\Delta_\mathsf{F}(N_1, N_2)$ since the speed, in this section, is always the package speed.

In other words, each predecessor of a node defines a lower bound on the node's start time. Let $\mathsf{earliestArrivalTime}_\prec(N)$ denote the earliest time at which the package can arrive at the node:

$$\mathsf{earliestArrivalTime}_\prec(N) = \begin{cases} \mathsf{earliestEntryTime}(p) & \text{if } N \text{ is the entry} \\ \max_{N' \in \mathsf{pre}_\prec(N)} \mathsf{eft}_\prec(N') + \Delta_\mathsf{F}(N', N) & \text{otherwise} \end{cases}$$

where $\mathsf{pre}_\prec(N) = \{N' | N' \prec N\}$ is the set of nodes that precede N (i.e., the pre-image of N in $\prec$).

Then

$$\mathsf{est}_\prec(N) \geq \mathsf{earliestArrivalTime}_\prec(N) \ .$$

Moreover, by definition,

$$\mathsf{est}_\prec(N) \geq \mathsf{net}(N)$$

so

$$\mathsf{est}_\prec(N) = \max(\mathsf{net}(N), \mathsf{earliestArrivalTime}_\prec(N)) \ .$$

Dually, let $\mathsf{lft}_\prec(N)$ denote the latest time at which the package can finish its activity at node N, consistent with the partial order, and let $\mathsf{lst}_\prec(N)$ denote the latest time at which the package can start its activity at N.

- If N is a target, then $\mathsf{lst}_\prec(N) = \mathsf{lft}_\prec(N) - \Delta_\mathsf{E}$.
- If N is the exit, then $\mathsf{lst}_\prec(N) = \mathsf{lft}_\prec(N) = \mathsf{latestExitTime}(p)$.
- If N is the entry, then $\mathsf{lst}_\prec(N) = \mathsf{lft}_\prec(N)$.

Each successor of a node defines an upper bound on the node's latest finish time:

$$N_1 \prec N_2 \implies \mathsf{lft}_\prec(N_1) \leq \mathsf{lst}_\prec(N_2) - \Delta_\mathsf{F}(N_1, N_2) \ .$$

Let $\mathsf{latestDepartureTime}_{\prec}(N)$ denote the latest time at which the package can depart from node N:

$$\mathsf{latestDepartureTime}_{\prec}(N) = \begin{cases} \mathsf{latestDepartureTime}(p) & \text{if } N \text{ is the exit} \\ \min_{N' \in \mathsf{succ}_{\prec}(N)} \mathsf{lst}_{\prec}(N') - \Delta_{\mathsf{F}}(N, N') & \text{otherwise} \end{cases}$$

where $\mathsf{succ}_{\prec}(N) = \{N' | N \prec N'\}$ is the set of nodes that follow N (i.e., the image of N in $\prec$).

Then

$$\mathsf{lft}_{\prec}(N) \leq \min(\mathsf{nlt}(N), \mathsf{latestDepartureTime}_{\prec}(N)) \ .$$

A.3. Precedence with Respect To Earliest/Latest Start/Finish Times

Now, given a partial order on nodes, if the *earliest* time at which one node can finish comes *after* the *latest* time at which another node can start, then there is no way for the package to visit the former before the latter.

More precisely, if

$$\mathsf{eft}_{\prec}(N_2) + \Delta_{\mathsf{F}}(N_2, N_1) > \mathsf{lst}_{\prec}(N_1)$$

then there is no way for the package to visit N_2 before N_1; i.e., any route that has N_2 before N_1 is infeasible. Thus, a constraint requiring N_1 to precede N_2 can be added to $\prec$ without changing the set of routes that are compatible with $\prec$.

Now, adding constraints to the partial order can have two effects:

- for some nodes, the earliest start/finish times may move later;
- for some nodes, the latest finish/start times may move earlier.

Either of these changes may result in further constraints being added to the partial order.

And so on until either a fix point is reached, or infeasibility of the package is detected.

Formally:

- Let $\prec^{(0)}$ be the initial partial order over the targets plus the entry and exit locations, for package p. For each target, the entry precedes the target, and the target precedes the exit:

  $$\forall T : \mathsf{Target} \in \mathsf{targets}(p) \cdot \mathsf{entry}(p) \prec^{(0)} T \wedge T \prec^{(0)} \mathsf{exit}(p) \ .$$

  The entry also precedes the exit: $\mathsf{entry}(p) \prec^{(0)} \mathsf{exit}(p)$.
- Let $\mathsf{est}^{(i)}(N)$ be the earliest start time for node N with respect to $\prec^{(i)}$ for $i = 0, 1, \ldots$.
- Let $\mathsf{eft}^{(i)}(N) = \mathsf{est}^{(i)}(N) + \Delta_{\mathsf{node}}(N)$ for node N, where $\Delta_{\mathsf{node}}(N)$ is 0 if N is the entry or exit, and Δ_{E} is N is a target.
- Let $\mathsf{lft}^{(i)}(N)$ be the latest finish times for node N with respect to $\prec^{(i)}$ for $i = 0, 1, \ldots$.
- Let $\mathsf{lst}^{(i)}(N) = \mathsf{lft}^{(i)}(N) - \Delta_{\mathsf{node}}(N)$ for node N.
- Let $\prec^{(i+1)}$ be the partial order over the nodes induced by $\prec^{(i)}$, $\mathsf{eft}^{(i)}$ and $\mathsf{lst}^{(i)}$, for $i = 0, 1, \ldots$. That is:

  $$\begin{array}{l} \forall N_1, N_2 \neq N_1 \cdot \\ \quad N_1 \prec^{(i+1)} N_2 \iff \\ \quad N_1 \prec^{(i)} N_2 \vee \mathsf{eft}^{(i)}(N_2) + \Delta_{\mathsf{F}}(N_2, N_1) > \mathsf{lst}^{(i)}(N_1) \ . \end{array}$$

- Let i^* be the lowest i for which $\prec^{(i)} = \prec^{(i+1)}$.
- Then $\prec = \prec^{(i^*)}$.

Note that if $N_1 \prec N_2$ and $N_2 \prec N_1$ for any two different nodes N_1 and N_2, then $\prec$ is not truly a partial order and in fact p is infeasible.

Moreover, if for any node, N, $\mathsf{eft}_{\prec}(N) > \mathsf{lft}_{\prec}(N)$, then the node is infeasible.

A.4. Shortest Package Route

If a route has not already been established for the package, then a good candidate is a shortest route. Moreover, if a shortest route respecting the time bounds cannot be found, then no feasible route exists and the package overall is infeasible.

More specifically, a feasible route:

- begins at the entry location on or after the package's earliest entry time;
- visits each target once and engages it for a duration of Δ_E — for each target, the engagement begins no sooner than the target's NET and ends no later than the target's NLT;
- ends at the exit location on or before the package's latest exit time.

A shortest route satisfies these constraints, while minimizing the total duration of the route.

This is a variation of the Traveling Salesman Problem with Time Windows (also known as the Vehicle Routing Problem with Time Windows) [2].

In addition to the above constraints, a feasible route may be limited by fuel considerations — since refueling is not possible while the missions are formed up as a package (i.e., between entering and exiting enemy airspace), the package's duration cannot exceed the minimum fuel duration of any of its missions. If a mission's refueling activities are known, then its maximum in-package fuel consumption can be calculated. Alternatively, a mission might be restricted, per doctrine, to using no more than some fraction of its fuel within enemy airspace — for example, it might be required to enter with at least 90% and exit with at least 30%, meaning that it can use at most 60% of its fuel in enemy airspace. Since the package's speed is known (viz., the minimum of the in-enemy-airspace speeds of its missions' platforms), each mission's fuel consumption rate is known, and thus its maximum duration can be determined. Thence, the package's maximum duration can be determined.

Given these constraints, there are several algorithms that can compute shortest routes.

Smaller instances can be solved by iterating over all orderings of the targets, determining a route for each order and keeping those that are feasible and of minimal duration.

Larger instances can be solved using a branch and bound algorithm that also searches over permutations of the targets, but that uses upper and lower bounds on durations to quickly discard parts of the search space that cannot yield minimal routes:

- As the algorithm searches, it maintains an upper bound on the minimum duration of feasible routes. This can be initialized to ∞, using a greedy/randomized algorithm to find feasible routes, or to some bound determined, say, by fuel considerations. When a new feasible route is found by the search, the upper bound is lowered if the new route is shorter.
- The algorithm's search is based on a *target prefix*, i.e., a partial sequence of the targets. A prefix represents all complete routes that are *consistent* with the prefix, i.e., that start by visiting targets in the same order as the prefix.
- Given a prefix, the algorithm determines a lower bound on the duration of all feasible routes consistent with the prefix — see below for methods. If the lower bound exceeds the current upper bound, then none of the routes consistent with the prefix can be a shortest route, and the prefix can be discarded by the search.
- Otherwise, the search extends the prefix by adding a target not already in the prefix. Eventually, either the prefix is discarded, or it contains all of the targets, and thus represents a complete route whose duration is tested against the current upper bound.
- The partial order on targets, $\prec$, can be used to restrict the search to prefixes that are consistent with the order. For example, if $T \prec T'$ then T' cannot be added to the prefix unless the prefix already contains T.
- In addition, as the search proceeds and the route prefix is extended, the partial order can be updated to reflect the order implied by the prefix. For example, the prefix $[T_1, T_2, T_3]$ implies the constraints $T_1 \prec T_2$, $T_2 \prec T_3$ and $T_3 \prec T$ for all $T \neq T_1$, $T \neq T_2$ and $T \neq T_3$.

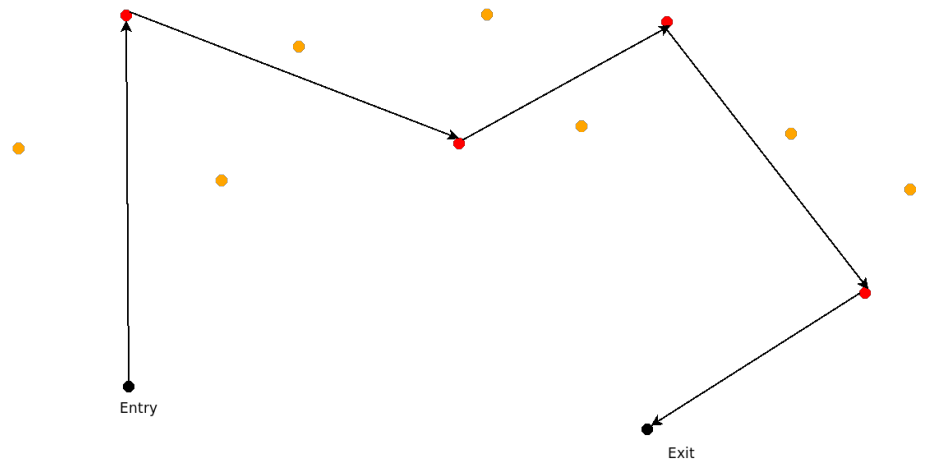

Figure 11: Computing a lower bound on the shortest route by sampling (indicated by red discs) the target set

A.5. Lower Bounds on Package Duration

Given an arbitrary target precedence for package p, such as shown in Figure 10, the entry e and exit x will always be connected by at least one path (through the graph corresponding to the partial order), since the initial construction of the precedence connects the entry to the exit via each target ($e \prec T \wedge T \prec x$ for each target T).

Consequently, the package's times at the entry and its times at the exit are related (i.e., mutually constrained). While they generally do not determine the package's duration exactly, they do provide a lower bound, as follows:

- The package must be in enemy airspace at time $\mathsf{est}_{\prec}(x)$, since, by definition, it cannot reach the exit before that time (for any feasible route).
- The package must be in enemy airspace at some time between $\mathsf{est}_{\prec}(e)$ and $\mathsf{lst}_{\prec}(e)$ — the former is the earliest that the package can assemble (prior to entry) and the latter is the latest that the package can enter and still reach the exit in time for its missions to return to their units before the ends of the their respective goes.
- Consequently, a lower bound on the duration for the package is $\mathsf{est}_{\prec}(x) - \mathsf{lst}_{\prec}(e)$. Note that this value may be negative — for example, if the goes are relatively long, compared with the flight durations, then $\mathsf{lst}_{\prec}(e)$ will be relatively late.

An alternative lower bound is $n\,\Delta_{\mathsf{E}} + \Delta_{\mathsf{F}}(\mathsf{speed}(p), e, x)$, where n is the number of targets. This quantity is simply the time it takes to engage n targets plus the time it takes to fly directly from the entry to the exit.

Another lower bound can be determined as follows. Given a set of targets, any shortest route on a *subset* of the targets provides a lower bound on the shortest route of the whole set. Thus, if the target set has more than, say, 7 targets, then a lower bound can be computed by selecting a subset of size, say, 4 and computing a shortest route for the subset, using enumeration of the routes. This can be repeated with different subsets and the maximum taken. The subsets should be chosen to (approximately) maximize the mean distance between the chosen targets and between the chosen targets and the entry/exit locations. See Figure 11.

Finally, discarding the time bounds from the Traveling Salesman Problem with Time Windows (TSPTW) formulation relaxes the problem into an instance of the standard Traveling Salesman Problem (TSP). A solution to the standard TSP thus provides a lower bound on the duration of solutions to the original TSPTW.

B. Strike Point Assignment Formulation as ILP Problem

Strike point assignment can be formulated as an *integer linear programming* problem [1], if the condition is dropped that only a single mission, per package, can engage the strike points in a given target.

The first task in formulating the ILP problem is to create a vector of all of the strike points in the package. The order is not important. The vector allows each strike point to be referenced using a unique positive integer. For example, Table 3 shows two targets to be engaged by a package. The vector of strike points is shown in Table 4a.

Target	Strike Point	Munition Option 1	Munition Option 2	Munition Option 3
HQ	S0001a	2 GBU03 → 90%	4 GBU02 → 80%	6 GBU01 → 70%
	S0001b	2 GBU03 → 90%	4 GBU02 → 80%	6 GBU01 → 70%
Airfield	S0002a	2 GBU04 → 90%	4 GBU03 → 80%	8 GBU02 → 70%
	S0002b	2 GBU02 → 90%	4 GBU01 → 80%	8 RKT01 → 70%

Table 3: Targets, strike points and munition options

Index	Strike Point
1	S0001a
2	S0001b
3	S0002a
4	S0002b

(a) StrikePoints[j]

Index	Munition Type
1	GBU04
2	GBU03
3	GBU02
4	GBU01
5	RKT01

(b) MunitionTypes[k]

Table 4: Vectors of strike points and munition types

The next task is to create a vector of munition types used in all of the strike points in all of the targets in the package. The order is not important. What is important is that each munition type appears exactly once in the vector. The vector of munition types are shown in Table 4b.

A cost matrix can then be constructed, with the rows corresponding to strike points (per the strike point vector) and the columns corresponding to the munition types (per the munition type vector). With reference to Table 12a, the number at row j, column k, is the number of munitions of type $\mathsf{MunitionTypes}[k]$ used by one of the munition options for strike point $\mathsf{StrikePoints}[j]$. (Recall that each munition option contains only one munition type, and no munition type is contained in more than one of the munition options of a given strike point.) If a given strike point has no munition option that contains a given munition type, then a cost of ∞ is assigned.

		Munition Type				
		1	2	3	4	5
Strike Point	1	∞	2	4	6	∞
	2	∞	2	4	6	∞
	3	2	4	8	∞	∞
	4	∞	∞	2	4	8

(a) Costs[j,k]

		Munition Type				
		1	2	3	4	5
Strike Point	1	0%	90%	80%	70%	0%
	2	0%	90%	80%	70%	0%
	3	90%	80%	70%	0%	0%
	4	0%	0%	90%	80%	70%

(b) Rewards[j,k]

Figure 12: Cost and reward matrices

Likewise, a reward matrix can be constructed with the rows corresponding to the strike points and the columns corresponding to the munitions types (as for the cost matrix), but with the entries being the probabilities of destruction. In other words, the number at row j, column k, is the probability of kill of the munition option that uses $\mathsf{MunitionTypes}[k]$ for $\mathsf{StrikePoints}[j]$. See Figure 12b.

Each target has a priority, a positive integer, with 1 being the highest priority. The priorities can be taken into account by dividing each element of the reward matrix by the priority of the appropriate target.

All of the strike points within a target have the same priority, which may result in rows of the reward matrix (for different strike points in the same target) being identical. To reduce this symmetry, which

may make finding a solution more difficult, the priorities of the strike points could be perturbed slightly, say by adding a random number in the range $[1 \times 10^{-5} \times r_{\mathsf{min}},\ 9 \times 10^{-5} \times r_{\mathsf{min}}]$, where r_{min} is the smallest value in the unperturbed reward matrix.

Next, a vector of the packages strike missions is constructed, again to allow missions to be referenced by a positive integer. See Table 5a.

Index	Strike Mission
1	FS1-001
2	FS2-005
3	FS3-003

(a) Missions[i]

		Munition Type				
		1	2	3	4	5
Mission	1	2	0	4	0	0
	2	0	4	2	2	0
	3	0	2	0	4	8

(b) Available[i,k]

Table 5: Vector of missions and matrix of available munitions

Then a matrix of munition availability can be constructed, with the rows corresponding to the missions (per the vector of missions) and the columns corresponding to the munition types (per the vector of munition types). In other words, $\mathsf{Available}[i, k]$ is the quantity of $\mathsf{MunitionTypes}[k]$ carried by $\mathsf{Missions}[i]$. See Table 5b.

Finally, a 3-dimensional 0/1 array, $x[i, j, k]$ is created that assigns missions to strike points and munition options: if $x[i, j, k]$ is 1, then Missions[i] engages StrikePoints[j] using MunitionOptions[k]. This incurs cost Costs[j,k] and receives reward Rewards[j,k].

The ILP problem then is to maximum the total reward

$$\sum_{i,j,k} x[i, j, k] \times \mathsf{Rewards}[j, k]$$

subject to the following constraints:

- For each strike point, at most one mission using at most one munition option engages the strike point:
$$\forall j \cdot \sum_{i,k} x[i, j, k] \leq 1 \ .$$
- For each mission and for each munition type, the total quantity of that munition type used by that mission cannot exceed the quantity of that munition carried by that mission:
$$\forall i, k \cdot \sum_{j} x[i, j, k] \times \mathsf{Costs}[j, k] \leq \mathsf{Available}[i, k] \ .$$

References

[1] *Integer Programming*
`https://en.wikipedia.org/wiki/Integer_linear_programming`

[2] *A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows*, Martin Desrochers, Jacques Desrosiers, Marius Solomon, 1992, Operations Research 40(2):342-354
`http://dx.doi.org/10.1287/opre.40.2.342`