ISR Planner
Post-Optimization Tools

Insert Missed • Swap Closer • Crossing Remove

Generated: December 13, 2025

Executive Summary

This document describes three post-optimization tools developed for the ISR (Intelligence, Surveillance, and Reconnaissance) Planner system. These tools enhance multi-drone mission planning by refining initial route solutions through target insertion, trajectory-aware reassignment, and route smoothing techniques.

Tool	Purpose	Performance Impact
Insert Missed	Add unvisited high-priority targets	Increases coverage
Swap Closer	Reassign targets to closer drones	Reduces distance by ~18%
Crossing Remove	Remove route crossings (2-opt)	Smooths routes

1. Insert Missed

Overview

Location: server/solver/post_optimizer.py (lines 1-531)
Purpose: Identifies and inserts unvisited high-priority targets into existing drone routes after initial optimization.
Type: Greedy insertion heuristic

How It Works

The Insert Missed tool operates in three phases:
• **Identification:** Analyzes the solution to find targets that were not included in any drone's route
• **Prioritization:** Ranks unvisited targets by priority level (higher priority targets are inserted first)
• **Insertion:** For each unvisited target, finds the optimal insertion point in the most suitable drone's route

Key Features

Constraint Awareness:
• Respects frozen route segments (segments that cannot be modified)
• Validates fuel capacity before insertion
• Checks drone capability requirements
• Maintains route feasibility

Coverage Statistics:
• Calculates and reports coverage percentages
• Tracks high-priority vs. low-priority target coverage
• Provides feedback on mission completeness

Insertion Strategy:
• Evaluates all possible insertion positions
• Minimizes additional distance cost
• Considers proximity to existing route points

Technical Details

Metric	Value
Lines of Code	531
Complexity	O(n × m × k) where n=unvisited, m=drones, k=route length
Processing Time	~50-100ms (typical mission)
Success Rate	Depends on fuel/capability constraints

Use Cases

• **Incomplete Coverage:** When initial optimization couldn't visit all high-priority targets due to fuel constraints
• **Priority Missions:** Maximizing coverage of critical targets
• **Dynamic Replanning:** Adding newly discovered targets to existing plans

2. Swap Closer (Trajectory Swap Optimizer)

Overview

Location: server/solver/post_optimizer.py (lines 532-1124)
Purpose: Reassigns targets between drones to minimize total mission distance by identifying targets that are closer to other drones' trajectories.
Type: Trajectory-aware optimization heuristic

How It Works

Swap Closer uses sophisticated geometric analysis to identify beneficial target swaps:
• **Trajectory Analysis:** For each drone-target pair, calculates two key metrics:
• - *SSD (Start-Segment Distance):* Distance from drone's starting position to the target
• - *OSD (Origin-Segment Distance):* Perpendicular distance from target to drone's trajectory line
• **Swap Candidate Identification:** Finds targets that are significantly closer to another drone's trajectory
• **Swap Execution:** Performs one swap per invocation to prevent cyclic behavior
• **Validation:** Ensures swaps maintain fuel feasibility and capability constraints

Key Algorithms

Distance Metrics:
The tool computes two critical distances for trajectory-aware optimization:

1. SSD (Start-Segment Distance)
Measures how far a target is from a drone's starting position. Lower SSD indicates the target is in the general direction of the drone's route.

2. OSD (Origin-Segment Distance)
Calculates the perpendicular distance from a target to a drone's trajectory line. Lower OSD indicates the target is close to the drone's flight path, making it a good candidate for reassignment.

Swap Decision Logic:
A target T assigned to drone D1 is swapped to drone D2 if:
• D2's trajectory passes closer to T than D1's trajectory
• The swap reduces total mission distance
• D2 has sufficient fuel to visit T
• D2 has the required capabilities for T

Recent Improvements

Trajectory Boundary Fix (Commits: 10db304, d060704, b1e1c20):
• Fixed handling of segments ending at airports (A1→Tx→A1 patterns)
• Improved evaluation of targets at trajectory boundaries
• Better handling of airport return segments

Cyclic Swap Prevention (Commit: dcb860e):
• Limited to one swap per invocation to prevent infinite loops
• Eliminates A→B→A→B cyclic swapping behavior
• Improved stability and predictability

Fuel Calculation Accuracy (Commit: a5ce53f):
• Now uses trajectory vertices instead of route waypoints
• More accurate fuel consumption estimates

• Better validation of swap feasibility

Technical Details

Metric	Value
Lines of Code	593 (lines 532-1124)
Complexity	O(n × m) where n=targets, m=drones
Processing Time	~100-150ms (typical mission)
Distance Reduction	Up to 18% improvement
Swaps Per Call	1 (prevents cycles)

Performance Impact

In typical missions, Swap Closer achieves:
• **18% reduction** in total flight distance
• **Better fuel utilization** across the drone fleet
• **More logical target assignments** based on actual flight paths
• **Reduced mission time** due to shorter routes

3. Crossing Remove (2-opt Optimizer)

Overview

Location: server/solver/post_optimizer.py (lines 1126-1379)
Purpose: Removes self-intersecting route segments using the classic 2-opt local search algorithm.
Type: Local search optimization

How It Works

The Crossing Remove tool applies the 2-opt algorithm, a well-established technique for route optimization:
• **Crossing Detection:** Examines all pairs of route segments to detect intersections
• **Segment Reversal:** When a crossing is found, reverses one segment to eliminate the intersection
• **Distance Evaluation:** Only accepts reversals that reduce total route distance
• **Iterative Refinement:** Repeats the process in multiple passes until no more improvements are found
• **Safety Limits:** Built-in iteration limits prevent infinite loops

The 2-opt Algorithm

Classic Algorithm:
2-opt is a simple local search algorithm for the Traveling Salesman Problem (TSP). It works by systematically removing two edges from a route and reconnecting them in a different way.

Example:
Given a route: A → B → C → D → A
If segments B→C and D→A cross each other, 2-opt reverses the segment between them:
Result: A → B → D → C → A (crossing eliminated)

Visual Representation:
Before: Route has a figure-8 or crossing pattern
After: Route is smoothed with no self-intersections

Complexity:
$O(n^2)$ per pass, where n is the number of waypoints in the route. Multiple passes may be performed until no further improvements are found.

Technical Details

Metric	Value
Lines of Code	254 (lines 1126-1379)
Complexity	$O(n^2)$ per pass
Processing Time	~50-100ms (typical mission)
Passes	Multiple until convergence
Safety Limit	Built-in iteration cap

Benefits

• **Route Smoothing:** Eliminates unnecessary backtracking and crossing patterns
• **Distance Reduction:** Shorter, more direct routes
• **Visual Clarity:** Routes are easier to understand and visualize

- **Fuel Efficiency:** Reduced distance means better fuel utilization
- **Fast Execution:** Completes quickly even for complex routes

Integration & Optimization Pipeline

These three tools work together as part of the post-optimization phase in the ISR Planner pipeline:

- **Step 1: Initial Solution** - Held-Karp algorithm generates optimal routes for allocated targets
- **Step 2: Insert Missed** - Adds unvisited high-priority targets to routes
- **Step 3: Swap Closer** - Reassigns targets to minimize trajectory distances (18% improvement)
- **Step 4: Crossing Remove** - Smooths routes by removing crossings using 2-opt
- **Step 5: Final Solution** - Optimized, feasible routes ready for execution

Execution Order

The tools are applied in a specific order to maximize effectiveness:

1. Insert Missed first because:
- Adds missing targets before trajectory optimization
- Ensures all high-priority targets are considered for swapping

2. Swap Closer second because:
- Works with complete target assignments
- May introduce new crossings that need to be removed

3. Crossing Remove last because:
- Cleans up any crossings introduced by swaps
- Provides final route smoothing
- Ensures optimal visual and distance characteristics

Combined Performance Summary

Tool	Processing Time	Impact	Complexity
Insert Missed	50-100ms	Increased coverage	$O(n \times m \times k)$
Swap Closer	100-150ms	~18% distance reduction	$O(n \times m)$
Crossing Remove	50-100ms	Route smoothing	$O(n^2)$
Total	**200-350ms**	**Complete optimization**	**Combined**

Note: Times based on typical mission with 19 targets and 5 drones. Performance scales with problem size.

Practical Use Cases

Scenario 1: Incomplete Initial Coverage
Problem: Initial Held-Karp optimization couldn't visit all high-priority targets due to fuel constraints.
Solution: Insert Missed adds the unvisited targets where feasible.
Result: Improved mission coverage without violating constraints.

Scenario 2: Suboptimal Target Assignments
Problem: Initial allocation assigned targets without considering actual flight trajectories.
Solution: Swap Closer reassigns targets based on trajectory proximity.
Result: 18% reduction in total flight distance, better fuel efficiency.

Scenario 3: Routes with Crossings
Problem: Routes have self-intersecting segments creating inefficient paths.
Solution: Crossing Remove applies 2-opt to eliminate crossings.
Result: Smooth, logical routes that are easier to execute and visualize.

Scenario 4: Dynamic Replanning
Problem: New high-priority targets discovered mid-mission.
Solution: Insert Missed + Swap Closer + Crossing Remove pipeline.
Result: Updated routes that incorporate new targets optimally.

Configuration & Best Practices

When to Enable Each Tool:

Insert Missed:
• Always enable for missions with high-priority targets
• Particularly useful when initial optimization has low coverage
• Disable if all targets are already visited (saves processing time)

Swap Closer:
• Recommended for all multi-drone missions
• Especially valuable when targets are geographically distributed
• Can be disabled for single-drone missions (no swaps possible)
• Run iteratively until no more swaps are beneficial

Crossing Remove:
• Recommended for all missions
• Fast execution makes it worthwhile even for simple routes
• Essential after swap operations that may introduce crossings
• Can be disabled for very simple, linear routes

Recommended Configuration:
For most missions, enable all three tools in sequence. The combined processing time (~200-350ms) is negligible compared to the quality improvements achieved.

Technical Architecture

Code Organization:
All three tools are implemented in a single file: *server/solver/post_optimizer.py*
Total: 1,379 lines of well-structured, modular code

Design Principles:
• **Modularity:** Each tool is self-contained and can be used independently
• **Constraint Awareness:** All tools respect fuel, capability, and frozen segment constraints
• **Safety First:** Built-in limits prevent infinite loops and invalid operations
• **Performance Optimized:** Efficient algorithms with appropriate complexity bounds
• **Maintainability:** Clear code structure with recent bug fixes and improvements

Dependencies:
• Distance calculations from SAM Distance Matrix Calculator
• Environment and constraint data from Solver Bridge
• Initial solutions from Held-Karp Orienteering Solver

Integration Points:
• Called by Solver Bridge as part of the optimization pipeline
• Compatible with LangGraph workflow orchestration
• Returns enhanced solutions with metadata (coverage stats, distance improvements)

Recent Development & Bug Fixes

The optimization tools have been actively maintained with recent improvements:

Date	Commit	Description
Recent	10db304	Fix Swap Closer to evaluate segments ending at airports
Recent	d060704	Fix Swap Closer to check segments ending at airports
Recent	b1e1c20	Fix Swap Closer trajectory boundary targets (A1→Tx→A1)
Recent	dcb860e	Fix Swap Closer cyclic swapping (one swap per call)
Recent	a5ce53f	Fix Swap Closer fuel calc (use trajectory vertices)

These fixes demonstrate ongoing refinement of the Swap Closer algorithm, particularly around:
• Edge cases with airport return segments
• Trajectory boundary handling
• Prevention of cyclic swapping behavior
• Accuracy of fuel calculations

Conclusion

The three post-optimization tools—**Insert Missed**, **Swap Closer**, and **Crossing Remove**—form a powerful suite for enhancing multi-drone mission planning solutions.

Key Strengths:
• **Fast Performance:** Complete post-optimization in 200-350ms
• **Significant Improvements:** Up to 18% distance reduction, increased coverage
• **Constraint-Aware:** Maintains fuel, capability, and frozen segment requirements
• **Well-Maintained:** Recent bug fixes and improvements ensure reliability
• **Production-Ready:** Proven algorithms with safety limits

Best Results:
Enable all three tools in the recommended order (Insert Missed → Swap Closer → Crossing Remove) for comprehensive route optimization. The minimal processing time makes this the recommended configuration for virtually all missions.

Future Development:
• Continue monitoring edge cases in Swap Closer
• Consider parallel execution for large drone fleets
• Potential integration of machine learning for swap decisions
• Enhanced visualization of optimization improvements

ISR Planner Post-Optimization Tools
Insert Missed • Swap Closer • Crossing Remove
Generated: December 13, 2025 at 04:58 PM
Code Location: server/solver/post_optimizer.py (1,379 lines)