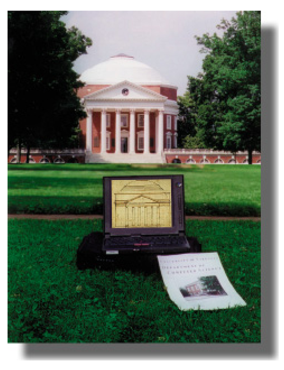
Moving-Target TSP and Related Problems



Department of Computer Science
University of Virginia
Charlottesville, Virginia 22904

C. S. Helvig
Gabriel Robins
Alexander Zelikovsky

Appeared in Proceedings of European Symposium on Algorithms (ESA'98), Venice, Italy, August, 1998

Introduction

- **Classical TSP:** sites to be visited are stationary
- **Related work:**
  - Time-dependent TSP (cost to travel between stationary sites changes with time)
- **Our contribution:**
  - Moving-Target TSP formulation
  - Algorithms for variants of Moving-Target TSP
- **Motivation:**
  - Supply ships resupply patrolling boats
  - Planes intercept mobile ground units

Problem Formulation

Given:

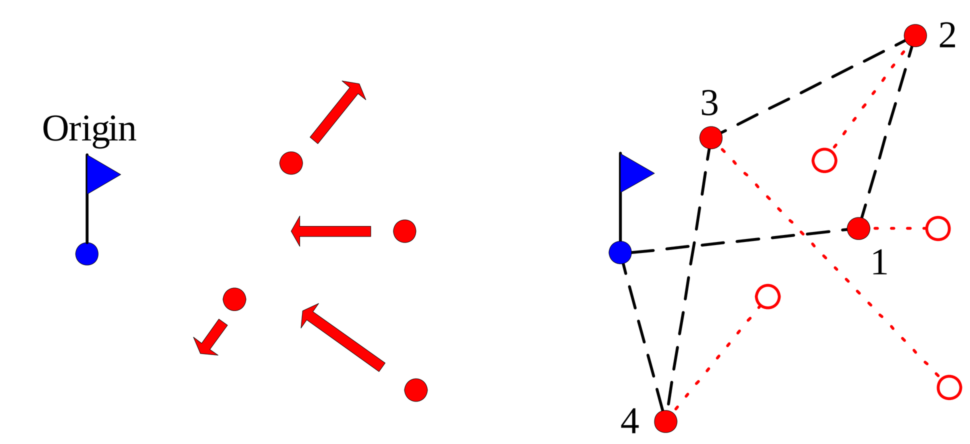
- A set $S = \{s_1, s_2,..., s_n\}$ of ***targets***
- Each target s_i has constant velocity $\mathbf{v_i}$
- Each target s_i starts moving from a position $\mathbf{p_i} \in \Re^n$
- A ***pursuer*** starting at origin with maximum speed $v > |\mathbf{v_i}|$

Find:

- A fastest tour which intercepts all targets

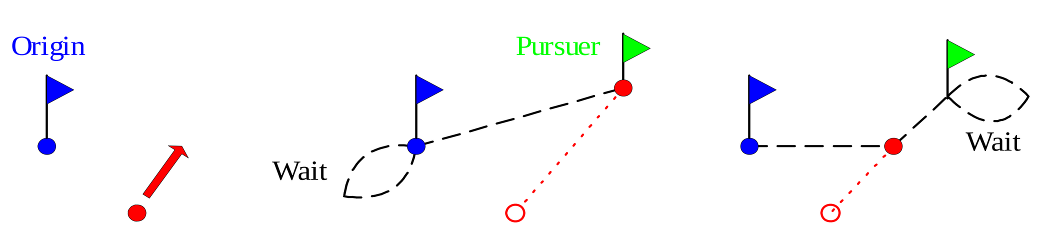
Variations (Moving-Target Vehicle Routing Problem):

- Multiple pursuers j = 1.. k
- Pursuers have a given capacity to fill target demand



Basic Observation

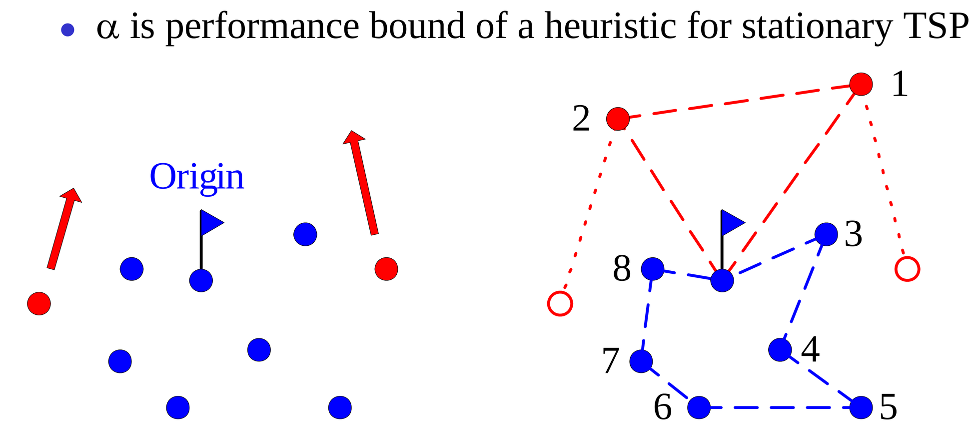
- **Lemma:** Optimal tours have no waiting periods



- **Corollary:** Pursuer always moves at max speed

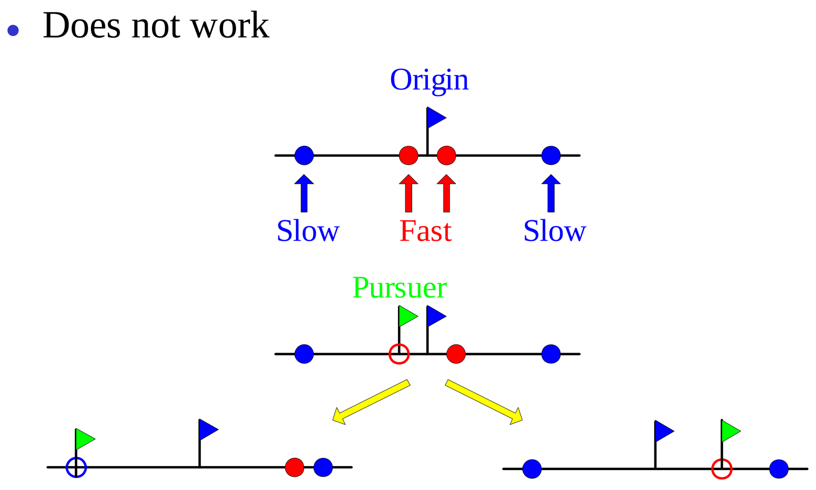
Few Moving Targets

- **Few (O(log n/log log n)) moving targets ⇒**
  - Efficient $(1+\alpha)$-approximation algorithm
  - α is performance bound of a heuristic for stationary TSP



One-Dimensional Variant

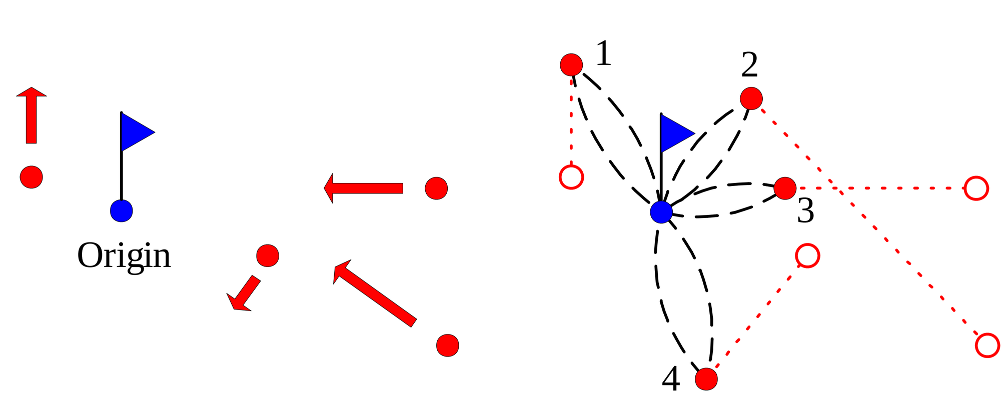
- **Trivial Algorithm:**
  - Intercept all targets on one side of the origin first
  - Then intercept targets on the other side
  - Does not work



- **Lemma:** Pursuer can change direction only after intercepting the fastest target
- Dynamic programming solution
- $O(n^2)$ algorithm
- Implemented and verified

Moving-Target TSP with Resupply

- Pursuer can intercept only one target before requiring resupply at the origin
  - Targets move directly away (or towards) the origin
- Corresponds to Moving-Target VRP except:
  - Single pursuer
  - Pursuer supply = target demand
- **Solution** (when targets move away from origin):
  - Intercept targets in order of increasing d_i/v_i
  - Algebraic proof for two targets
  - For n targets, swapping targets improves tour



- Targets intercept origin ⇒ Implicit change in target velocity
- **Valid tour**: No target passes through the origin
- **Theorem**: If the tour in which pursuer intercepts
  - Targets moving away in ascending order of d_i/v_i
  - Approaching targets in descending order of d_i/v_i

  is valid, then it is optimal
- **Problem:** Find the fastest *valid* tour
- **Theorem:** The tour where the pursuer intercepts targets in descending order of d_i/v_i is always valid and the slowest
- **Theorem:** Slowest tour ≤ **2**×(optimal valid tour)

Multi-Pursuer Moving-Target TSP with Resupply

- Moving-Target Vehicle Routing Problem with
  - Multiple pursuers
  - Pursuer supply = target demand
  - All targets are moving away
- Minimization Objectives:
  - **Makespan** = when the last pursuer at the base
    - Standard for multiprocessor scheduling
    - NP-hard even for stationary targets
  - **Total time** = total time while pursuers are moving
    - Standard for classical vehicle routing problem
    - Trivial for stationary case
- **Theorem**: Total time objective is NP-hard

- If all targets have the same d_i/v_i
  - Total time objective is still NP-hard
  - Approximation algorithm with

    Performance ratio = $\max_i \{(1 + v_i) / (1 - v_i)\}$
- If all targets have the same speed v_i
  - Nontrivial result
  - The following strategy is optimal:

    send next available pursuer after the closest target

Summary

- Formulation of Moving-Target TSP
- Exact algorithm for one-dimensional version
- Heuristic when few targets are moving
- Approximate and exact heuristics for selected variants of Moving-Target TSP with Resupply

Future Work

- 2-Dimensional Moving-Target TSP
- Moving-Target TSP with Resupply
  - When targets can pass through origin or when finding the fastest valid tour:
    - Prove/Disprove NP-Hardness
    - Find efficient algorithms/heuristics
- Multi-Pursuer Moving-Target TSP