The Moving-Target Traveling Salesman Problem*

C. S. Helvig[‡], Gabriel Robins, and Alex Zelikovsky[†]

[‡] Volition Inc., Champaign, IL 61820

Department of Computer Science, University of Virginia, Charlottesville, VA 22903

[†] Department of Computer Science, Georgia State University, GA 30303

Abstract

Previous literature on the Traveling Salesman Problem (TSP) assumed that the sites to be visited are stationary. Motivated by practical applications, we introduce a time-dependent generalization of TSP which we call Moving-Target TSP, where a pursuer must intercept in minimum time a set of targets which move with constant velocities. We propose approximate and exact algorithms for several natural variants of Moving-Target TSP.

1 Introduction

The classical Traveling Salesman Problem (TSP) has been studied extensively, and many TSP heuristics have been proposed over the years (see surveys such as [8] and [11]). Previous works on TSP have assumed that the cities/targets to be visited are stationary. However, several practical scenarios give rise to TSP instances where the targets to be visited are themselves in motion (e.g., when a supply ship resupplies patrolling boats, or when an aircraft must intercept a number of mobile ground units). In this paper, we introduce a generalization of the Traveling Salesman Problem where targets can move with constant velocities, formulated as follows:

> **The Moving-Target Traveling Salesman Problem**: Given a set $S = \{s_1, \ldots, s_n\}$ of *targets*, each s_i moving at constant velocity $\vec{v}_i$ from an initial position p_i, and given a *pursuer* starting at the origin and having maximum speed $v > |\vec{v}_i|$, find the fastest tour starting (and ending) at the origin, which intercepts all targets.

Related formulations consider *time-dependent* versions of the Vehicle Routing Problem, where one or more trucks whose speeds vary with time-of-day (due to traffic congestion) serve customers at fixed locations [9]. These works give exponential-time algorithms based on a mixed integer linear programming formulation and dynamic programming [9] [10] [12]. One major drawback of such general formulations is that they do not simultaneously yield both efficient and *provably* bounded-cost heuristics (e.g., the restricted dynamic programming heuristic of [10] is efficient, but is not provably bounded-cost). These

*This work was supported by NSF Young Investigator Award MIP-9457412, by NSF grant CCR-9988331, and by a Packard Foundation Fellowship. The corresponding author is Professor Gabriel Robins, (804) 982-2207, FAX (804) 982-2214, robins@cs.virginia.edu, www.cs.virginia.edu/robins. A preliminary version (extended abstract) of this paper has appeared in [5].

formulations generalize TSP without the triangle inequality, which admits no approximation bounds unless $P = NP$ [2]. Applicability of dynamic programming to related formulations was also considered in [7]. The approximation complexity of Moving-Target TSP was studied in [4], where it was shown that Moving-Target TSP cannot be approximated better than by a factor of $2^{\Omega(\sqrt{n})}$ times optimal within polynomial time unless $P = NP$.

In this paper, we propose and address several natural variants of Moving-Target TSP. In Section 2, we show that unlike the classical TSP, the restriction of Moving-Target TSP to one dimension (i.e., where all points move on a line) is not trivial, and we give an exact $O(n^2)$-time algorithm. For Moving-Target TSP instances where the number of moving targets is sufficiently small, we develop a $(1 + \alpha)$-approximation algorithm, where α denotes the approximation ratio of the best classical TSP heuristic. It was shown in [4] that the $(2-\epsilon)$-approximation is NP-hard even in the case when only two targets are moving. Thus, combining our approach with the polynomial-time approximation scheme[1] for Euclidean TSP [1] yields almost optimal $(2 + \epsilon)$-approximation algorithms for Moving-Target TSP when enough of the targets are stationary.

Next, in Section 3, we shift our attention to selected variants of Moving-Target TSP with Resupply[2], where the pursuer must return to the origin for resupply after intercepting each target, as shown in Figure 1(c). For these variants, we assume that all targets are moving on a line through the origin away from (or towards) the origin. We present a surprisingly simple exact algorithm for Moving-Target TSP when the targets approaching the origin are either far away or slow. We also consider the case when all targets are moving towards the origin, but with the additional requirement that the pursuer must intercept all of the targets before they reach the origin. We show that such a tour always exists and that no tour which satisfies the constraints is more than twice as long as the optimal tour.

Finally, in Section 4, we generalize Moving-Target TSP with Resupply to allow multiple pursuers which all move with the same maximum speed. This problem also can be viewed as a dynamic generalization of multiprocessor scheduling where the processing time depends on the starting time of processing a job. We show that the problem is NP-hard, which is a non-trivial result because our formulation has a total time objective which is different from the standard "makespan" and minimum latency objectives. We also develop an approximation algorithm for the case when, projecting back in time, all targets would have left the origin simultaneously. Finally, we present an exact algorithm for the case when all targets have the same speed, and conclude in Section 5 with future research directions. A preliminary version of this research was presented in [5].

[1] The existence of a polynomial-time approximation scheme implies that for any $\epsilon > 0$, there is a polynomial-time algorithm with an approximation ratio of at most $1 + \epsilon$.

[2] Our formulation corresponds to a dynamic version of the Vehicle Routing Problem, defined as follows. Given a set of n targets, each with *demand* c_i moving at a constant velocity $\vec{v}_i$ from an initial position p_i for $i = 1, \ldots, n$, and a set of k pursuers initially located at the origin and having maximum speed V_j and *supply* C_j for $j = 1, \ldots, k$, find a tour for each pursuer such that the demand of each target is satisfied and the total time of vehicles in operation is minimized.

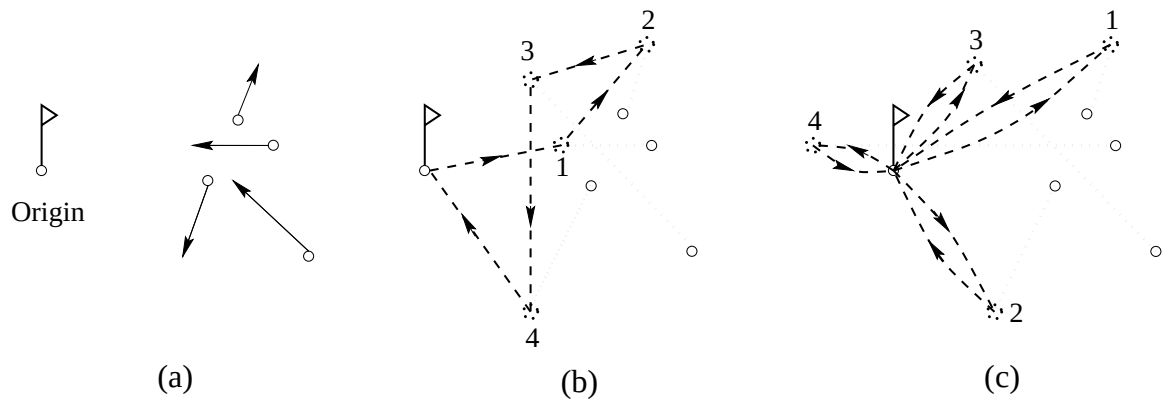

Figure 1: (a) An instance of Moving-Target TSP. (b) A shortest-time tour (dashed line) which begins at the origin (flag) and intercepts all of the targets. (c) In Moving-Target TSP with Resupply, the pursuer must return to the origin after intercepting each target (an optimal interception schedule is shown).

2 Special Instances of Moving-Target TSP

Since unrestricted Moving-Target TSP is NP-hard[3], and because non-optimal tours can have unbounded error, we consider special variants where Moving-Target TSP is solvable either exactly or to within a reasonable approximation ratio. In this section, we consider two variants: (1) when targets are confined to a single line, and (2) when the number of moving targets is small. The following lemma is central to our analyses of Moving-Target TSP and its variants:

Lemma 1 *(***The No-Waiting Lemma***): In any optimal Moving-Target TSP tour, the pursuer must always move at maximum speed.*

Proof: Assume towards contradiction that we have an optimal tour T that involves a pursuer moving slower than the maximum speed. This is equivalent to letting the pursuer wait in place for some time periods and then move at maximum speed during other periods. However, a tour that involves any waiting cannot be optimal, which can be seen as follows. Let s be the first target to be intercepted after the first waiting period, and let the pursuer intercept the target s at position p and time t. If instead, the pursuer first intercepts s and then proceeds to the point p, then it will reach p earlier than t, since the pursuer is faster than the target s. Afterwards, the pursuer could wait at p until time t and continue along the original tour T. Thus, the pursuer can postpone the waiting period until after intercepting s. Repeatedly applying this argument, the waiting periods can all be postponed until after the end of the tour. But waiting after the end of the tour is unnecessary, and thus we can obtain a tour which is faster than the supposedly optimal tour T, a contradiction. □

Lemma 1 allows us to define a tour purely by the sequence of intercepted targets in their order of interception by the pursuer. Furthermore, we may consider only tours in which the pursuer always moves with maximum speed towards intercepting the next target.

[3]Note that classical TSP is a special case of Moving-Target TSP where all the velocities equal zero.

2.1 Moving-Target TSP in One Dimension

In this subsection, we consider Moving-Target TSP where the pursuer and all targets are confined to a single line, and we develop an $O(n^2)$ exact algorithm for this variant based on dynamic programming. To see that the one-dimensional constraint does not trivialize the problem, consider the Moving-Target generalization of the exact algorithm for the one-dimensional classic TSP. First, compute the cost of the tour which intercepts all targets to the left of the origin and then intercepts all targets to the right of the origin. Next, compute the cost of the other "natural" tour which intercepts all targets to the right of the origin and then intercepts all targets to the left of the origin. Finally, from this pair of possible tours, choose the one with the least cost.

Unfortunately, this simple heuristic has unbounded error, as can be illustrated with the following example. Consider the case when there are four targets, two of them on either side of the origin, extremely close to the origin but moving away from the origin very quickly, while the other two targets are on either side of the origin, but much further from it, and are so slow as to be almost stationary. If we intercept the two fast targets immediately, then we spend almost no time in chasing them. However, if we first intercept all of the targets on one side of the origin, then we will later be forced to spend an arbitrarily long time chasing the remaining fast target on the other side of the origin. Thus, the pursuer may repeatedly change direction in an optimal tour. The following lemma bounds the number of turning points in an optimal tour.

Lemma 2 *In an optimal tour for one-dimensional Moving-Target TSP, the pursuer cannot change direction until it intercepts the fastest target ahead of it.*

Proof: Suppose towards contradiction that in an optimal tour T, the pursuer changes direction at time t before intercepting the fastest target s ahead of it. There exists some sufficiently small $\delta > 0$ such that, in the time period between $t - \delta$ and $t + \delta$, the pursuer changes direction only at time t. Consider an alternative tour where the pursuer stops at time $t - \delta$, waits without moving until time $t + \delta$, and then continues along the original tour.

Note that in both the original and the alternative tours, the pursuer is: (1) at the same point P by time $t + \delta$; and (2) has always been between the point P and the target s during the time period $(t - \delta, t + \delta)$. All the targets that the pursuer would intercept in the time period between $t - \delta$ and $t + \delta$ are between P and the target s and are moving slower than s, so therefore they cannot pass s. In essence, the turning of the pursuer could not have decreased the total tour time, even if it proceeds to intercept targets on the other side of the starting point, because during the time that it was *not* chasing the fastest target on one side, it was only intercepting targets that it must eventually intercept on the way to the fastest target s anyway. In fact, the time spent not chasing the fastest target is thus equivalent to time spent waiting in place. By the No-Waiting Lemma (Lemma 1), the original tour is therefore not optimal, because it is equivalent to a tour with a waiting period. □

To fully define a tour, we need to know only the points when the pursuer changes direction, and by Lemma 2, it may happen only when the pursuer intercepts the fastest target ahead of it. Thus, we may view a tour as a sequence of snapshots at the moments when the pursuer intercepts the fastest target which was then ahead of it. For any such snapshot, the information relevant to our algorithm

can be represented as a *state* (s_k, s_f), where s_k is the target just intercepted, and s_f is the fastest target on the other side. All tours have the same initial state A_0 and the same final state A_{final}. Note that neither A_0 nor A_{final} have corresponding targets s_k or s_f. States have a time function associated with them, denoted $t(A)$, representing the shortest time in which this state can be achieved. Naturally, we assign $t(A_0) = 0$ for the initial state.

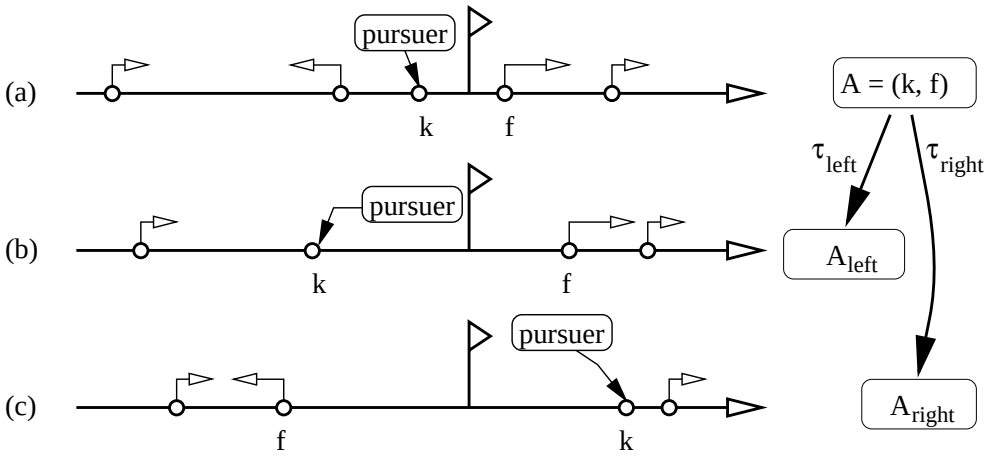

Figure 2: An instance of one-dimensional Moving-Target TSP (the flag represents the origin). On the right, we show the two possible transitions τ_{left} and τ_{right} from a state $A = (s_k, s_f)$ which is shown in (a). The transition τ_{left} corresponds to intercepting the target to the left of the pursuer. It transforms the state A into the state A_{left}, depicted in (b). Alternatively, the transition τ_{right} transforms A into the state A_{right} shown in (c).

Note that Lemma 2 implies that there are at most two possible transitions from any state $A = (s_k, s_f)$ at assigned time $t(A)$ (see Figure 2). These two transitions represent the two possible choices of the next target to be intercepted, either the fastest to the left of the pursuer or the fastest to the right of the pursuer. The time of each transition τ, denoted $t(\tau)$, is the time necessary for the pursuer to intercept the corresponding target (the fastest on the left or on the right) from the position of target s_k at time $t(A)$. Note that the state (s_k, s_f) does not have transitions into it when the corresponding target s_k or s_f would be intercepted (i.e., overtaken) by a faster target behind it before the pursuer can reach s_k.

Our algorithm works as follows (see Figure 3). In the preprocessing step of our algorithm, we partition the targets into two lists, *Left* and *Right* (according to whether targets are on the left or the right side of the origin, respectively). Then, we sort the lists according to non-increasing order of their speeds. We traverse both sorted lists and delete any target from the list which is closer to the origin than its predecessor. The targets which remain in these two lists are the only targets for which the pursuer may change direction after intercepting them.

Let $G = (V, E)$ be the graph with the nodes V corresponding to states, and the arcs E corresponding to transitions. Each tour in this graph corresponds to a path from the initial state to the final state and

vice versa. Note that the graph G is acyclic. Indeed, the pursuer cannot return to the state in which it has already been before because it cannot intercept the same target twice. Our algorithm can now be viewed as a simple generalization of the algorithm for finding the shortest path in the acyclic graph G where the weight of an arc τ (the time necessary to perform transition τ) depends on the weight of the shortest path from the initial state to the beginning state of τ. The runtime of our algorithm is $O(V + E)$ or $O(n^2)$.

In order to consider *all* possible paths, our algorithm traverses the states in a topological order, where no transition is allowed to go backwards with respect to this order. This is accomplished by sorting (and later traversing) the states in ascending order of the sum of the indices of the targets s_k and s_f (for each state $A = (s_k, s_f)$) in the *Left* and *Right* lists. When we traverse a given state A in the algorithm, we do one of three things: (1) if the state has no transitions into it, we proceed to the next state in the list; (2) if the pursuer has intercepted all of the targets on one side, we make a transition into the final state A_{final}; or (3) we make two transitions which correspond to sending the pursuer after either the fastest target on the left or the fastest target on the right. We define the time of a state B as $t(B) = min\{t(A) + t(\tau) \mid \tau : A \rightarrow B\}$, the time required for the shortest sequence of transitions from state A_0 to state B.

Once we have visited each of the states, we can traverse the transitions backwards from the final state A_{final} back to the initial state A_0. This gives us the list of states which describes the turning points for the pursuer in an optimal tour. For each pair of turning points, we find the subset of targets which are intercepted between them. Once we targets are partitioned into subsets, we sort the targets inside each subset by their interception times. Finally, we merge the sorted subsets into a combined interception order, yielding an optimal solution. This algorithm is summarized formally in Figure 3.

Theorem 3 *The above algorithm for One-Dimensional Moving-Target TSP finds an optimal tour.*

Proof: When the algorithm given in Figure 3 terminates, all transitions lead to states with defined times. By Lemmas 1 and 2, this will guarantee that the time will be defined for all states which may be reached by any optimal tour. Our strategy is to prove that no tour can reach state B earlier than $t(B)$, as defined in our algorithm.

Assume towards contradiction that B is the first state in an optimal tour which was reached in time $t(B) > OPT(B)$. The state B cannot be the initial state A_0, since $t(A_0) = 0$. Let the state A precede the state B in the optimal tour, and let τ be the transition from A to B (see Figure 4). Since B is the first state for which $t(B) < OPT(B)$, and A precedes B, then we have $t(A) \leq OPT(A)$. Our algorithm guarantees that there is a tour that reaches the state $A = (s_k, s_f)$ in time $t(A)$.

If $t(A) = OPT(A)$, then $t(B) \leq OPT(B)$ since our algorithm finds the fastest transition from A to B. Otherwise, if $t(A) < OPT(A)$, rather than proceed to state B at time $t(A)$, the pursuer can instead proceed to the position p of the target s_k at time $OPT(A)$, and arrive there earlier than time $OPT(A)$; this is true because the pursuer is faster than the target s_k. The pursuer can wait at position p until time $OPT(A)$, and then proceed along the optimal tour, reaching the state B at time $OPT(B)$ (see Figure 4).

Consider what would happen if, after intercepting s_k at time $t(A)$, the pursuer had proceeded

Exact Algorithm for One-Dimensional Moving-Target TSP

Input: The initial positions and velocities of n targets, and the maximum pursuer speed
Output: A time-optimal tour intercepting all targets, and returning back to the origin

Preprocessing
Partition the list of targets into the targets on the left side and the right side of the origin
Sort the targets on the left into list *Left* in order of non-increasing speeds
Sort the targets on the right into list *Right* in order of non-increasing speeds
Delete targets from *Left* which are closer to the origin than faster targets in this list
Delete targets from *Right* which are closer to the origin than faster targets in this same list
If *Left* or *Right* is empty **then**
    Calculate the time required to intercept all remaining targets; and
    Go to the postprocessing step

Main algorithm
Let A_0 be the start state
Let A_{final} be the final state
$STATE$ is the sorted list of states in order of non-decreasing sum of the indices
    of each state's targets in lists *Left* and *Right*
Place A_0 first in the list $STATE$
Place A_{final} last in the list $STATE$
$t(A) \leftarrow \infty$ for any state $A \neq A_0$
$t(A_0) \leftarrow 0$
$current \leftarrow 0$

While $current \leq$ the number of states in $STATE$ **do**
    $A = STATE[current]$
    If there are no transitions into A **then**
        Increment $current$ and jump back to the beginning of the while loop
    If for state A, all remaining targets are on one side of the origin **then**
        $t(\tau_{final}) \leftarrow$ time required to intercept the remaining targets (and return to the origin)
        If $t(A) + t(\tau_{final}) < t(A_{final})$ **then**
            $t(A_{final}) \leftarrow t(A) + t(\tau_{final})$
    Else
        Calculate the two transitions τ_{left} and τ_{right} from state A using lists *Left* and *Right*,
        (see Figure 2)
        If $t(A) + t(\tau_{left}) < t(A_{left})$ **then**
            $t(A_{left}) \leftarrow t(A) + t(\tau_{left})$
        If $t(A) + t(\tau_{right}) < t(A_{right})$ **then**
            $t(A_{right}) \leftarrow t(A) + t(\tau_{right})$
    Increment $current$

$OUTPUT \leftarrow$ the reverse list of states from A_{final} back to A_0

Postprocessing
For each pair of consecutive states in $OUTPUT$
    Calculate which targets are intercepted between the state pair
    Sort the intercepted targets by the interception order

Output the concatenated sorted lists of targets

Figure 3: An exact algorithm for Moving-Target TSP in one dimension.

directly to state B along an alternative transition τ'. If the pursuer had followed this alternative strategy (instead of going indirectly through position p as described above), then it would have arrived at state B at time no later than $OPT(B)$. Thus, $OPT(B)$ cannot be less than $t(A) + t(\tau')$, which in turn cannot be less than $t(B)$, a contradiction. □

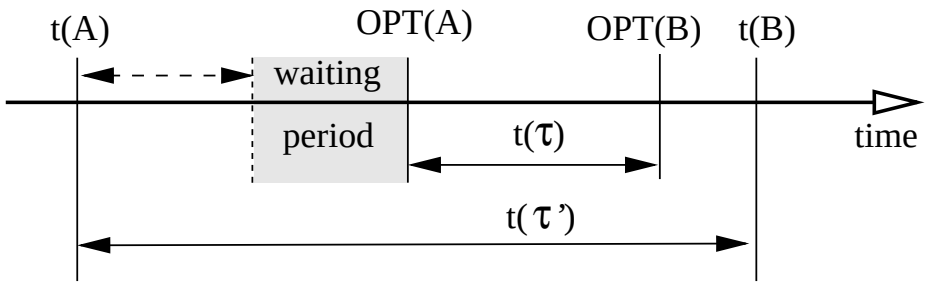

Figure 4: Rather than proceed to state B at time $t(A)$ along the transition τ' (lower solid line), the pursuer can instead proceed (along the dashed line) to position p where target k will be at time $OPT(A)$, arriving there earlier than time $OPT(A)$ (dotted line). Then it can wait (shaded region) and proceed along transition τ.

Theorem 4 *The above algorithm for One-Dimensional Moving-Target TSP runs in $O(n^2)$ time for n targets.*

Proof: In the preprocessing phase, the partitioning and sorting requires $O(n \log n)$ time. Deleting targets from lists *Left* and *Right* requires only $O(n)$ time. The entire preprocessing phase therefore runs within $O(n \log n)$ time.

The analysis of the main algorithm is as follows. First, referring to Figure 3, we observe that there are $O(n^2)$ states to traverse, and that we only traverse each state only once. Also, note that transitions can only go from earlier states in the list $STATE$ to later states. This is because any transition leads from a state whose targets have smaller indices in lists *Left* and *Right*, to states with larger indices. Thus, by the time we traverse a state in our algorithm, we have already considered all possible optimal paths to this state. Therefore, we need to traverse each of the $O(n^2)$ states only once. When traversing a state, we must traverse at most 2 transitions from that state. Lemma 5 below concludes the proof that the main algorithm can be completed in $O(n^2)$ time.

Finally, we must show that the postprocessing can also be accomplished within $O(n^2)$ time. First, note that finding a single subset of targets intercepted along a transition requires $O(n)$ time, because we must check each target to determine if it was intercepted. Since there may be $O(n)$ transitions in an optimal tour, we may need to find $O(n)$ subsets, and thus the partitioning of the targets into subsets may require $O(n^2)$ time. Sorting the individual subsets and combining them into an optimal target interception order takes at most $O(n \log n)$ time. Thus, the postprocessing step requires a total of $O(n^2)$ time. Since preprocessing, postprocessing, and the main algorithm can all be accomplished within $O(n^2)$ time, the runtime of our overall algorithm for the one-dimensional Moving-Target TSP is bounded by $O(n^2)$. □

Now we complete the proof of Theorem 4 by proving the following lemma.

Lemma 5 *All transitions can be traversed in constant amortized time per transition.*

Proof: In order to compute both of the possible transitions from a state $A = (s_k, s_f)$, we need to

determine the fastest remaining targets to the left and to the right of the current target s_k at time $t(s_k, s_f)$, the time of state (s_k, s_f). We already know the fastest target on the opposite side of the origin. Therefore, we only need to compute the fastest target on the same side of the origin as the just-intercepted target s_k.

Without loss of generality, let $s_k \in Left$. Let $next(s_k, s_f)$ be the fastest target to the left of the pursuer at state (s_k, s_f). To prove the lemma, it is sufficient to find the set $Next(s_k) = \{next(s_k, s_f) | s_f \in Right\}$ in time $O(n)$. Our algorithm is based on the following claim.

Claim *Let $s_{f'}$ be later than s_f in the list Right. Then $next(s_k, s_{f'})$ occurs later than $next(s_k, s_f)$ in the list $Left$.*

Proof: By time $t(s_k, s_{f'})$, the target s_f is intercepted, since otherwise s_f would be the fastest target on the right. Time $t(s_k, s_f)$ is the *minimum* time required to reach state (s_k, s_f), therefore $t(s_k, s_{f'}) \geq t(s_k, s_f)$. This implies that the target s_k intercepts at least as many targets from $Left$ by time $t(s_k, s_{f'})$ as it intercepts by time $t(s_k, s_f)$. Finally, $next(s_k, s_{f'})$ is no faster and no closer to the origin than $next(s_k, s_f)$. □

We can now finish the proof of Lemma 5. All the targets in $Next(s_k)$ occur later than s_k in the list $Left$. Thus, for the earliest state A_1 in $Next(s_k)$, we find $next(A_1)$ by traversing $Left$ starting from s_k. The above claim implies that for the next state A_2, we can traverse $Left$ starting from $next(A_1)$, and inductively, for state A_i we can traverse $Left$ from $next(A_{i-1})$. Thus, we need to traverse the list $Left$ only once. Traversing a target s_l in $Left$ for state A_i takes constant time: we need to check whether by time $t(A_i)$ the target s_k has already intercepted (i.e., overran) target s_l or not, by comparing the speeds and the original positions of s_k and s_l. □

We have implemented this algorithm using the C++ programming language. Computational benchmarks against an exhaustive search algorithm empirically confirmed its correctness, as well as its quadratic runtime.

2.2 Heuristics for Few Moving Targets

In this subsection, we consider Moving-Target TSP when only some of the targets are moving (while the majority are stationary). From among the many existing approximation algorithms for classical (stationary) TSP [8], choose one such heuristic, having performance bound α. Using this algorithm for stationary targets, we now show how to construct an efficient algorithm having a performance bound of $1 + \alpha$ when the number of moving targets is sufficiently small.

Theorem 6 *Moving-Target TSP where at most $O(\frac{\log n}{\log \log n})$ of the targets are moving can be approximated in polynomial time with performance bound $1+\alpha$, where α is a performance bound of an arbitrary classical TSP heuristic.*

Proof: Our approach consists of two parts. First, the pursuer intercepts all moving targets in optimal time and returns to the origin. Secondly, the pursuer intercepts the stationary targets using any chosen approximation algorithm.

Since the pursuer first intercepts all moving targets optimally, the time required to intercept them cannot be longer than the time of an optimal tour over both the stationary and the moving targets. The chosen stationary target heuristic retains its bound for intercepting the remaining (stationary) targets. Thus, we may sum the bounds for the two tours (over the the moving targets and the stationary targets, respectively) to obtain a total performance bound of $1 + \alpha$ for this combined algorithm.

An optimal tour which intercepts $k = O(1) \cdot \frac{\log n}{\log \log n}$ moving targets can be found efficiently using exhaustive search of all possible tours over the moving targets. In order to find the optimum, we need to check $k! \leq k^k$ possible tours. The logarithm of this quantity is at most:

$$\log k^k = O(1) \cdot \frac{\log n}{\log \log n} \cdot \log \left(O(1) \cdot \frac{\log n}{\log \log n} \right) \leq O(1) \cdot \log n$$

Thus $k^k = \exp(O(1) \cdot \log n) = n^{O(1)}$, and an exhaustive enumeration of all the k^k possible tours over the moving targets can therefore be achieved within polynomial time. □

Figure 5 illustrates this approach with an example.

3 Moving-Target TSP With Resupply After Intercepts

In this section we consider Moving-Target TSP where a single pursuer must return to the origin after intercepting each target. We call this problem *Moving-Target TSP With Resupply*. In order to simplify the analysis, we assume that the targets all move on radial lines passing through the origin, i.e., either directly towards or away from the origin. When targets are either (1) slow, (2) far from the origin, or (3) already moving along a line containing the origin, this model provides a good approximation since the projections of target velocities onto radial lines through the origin are approximately constant.

We define a *valid* tour to be a tour where no target passes near (or through) the origin without first being intercepted by the pursuer, and thus the velocities of all targets may be considered to be fixed with respect to the origin. Note that the No-Waiting Lemma (Lemma 1) enables us to still assume that the pursuer always moves with maximum speed towards intercepting the next target. In Subsection 3.1, we restrict the problem by considering the scenario when the shortest tour is valid. In other words, no target will pass the origin for a "sufficiently" long time (we will later elaborate on how long that should be). We show that there is a simple way to determine an optimal target interception order for this scenario. It is worthwhile to note that in this case, an optimal order is quite similar to the optimal schedule for minimizing a weighted completion time on a single machine.

We consider a similar scenario in Subsection 3.2, except that we introduce a new constraint, namely that the pursuer must intercept each target before it reaches the origin (i.e., the tour *must* be valid). We formulate the problem in terms of "defending" the origin from the (incoming) targets, and show that there always exists a valid tour regardless of the number of targets. Then, we prove that the longest tour can be at most only twice the length of the shortest valid tour.

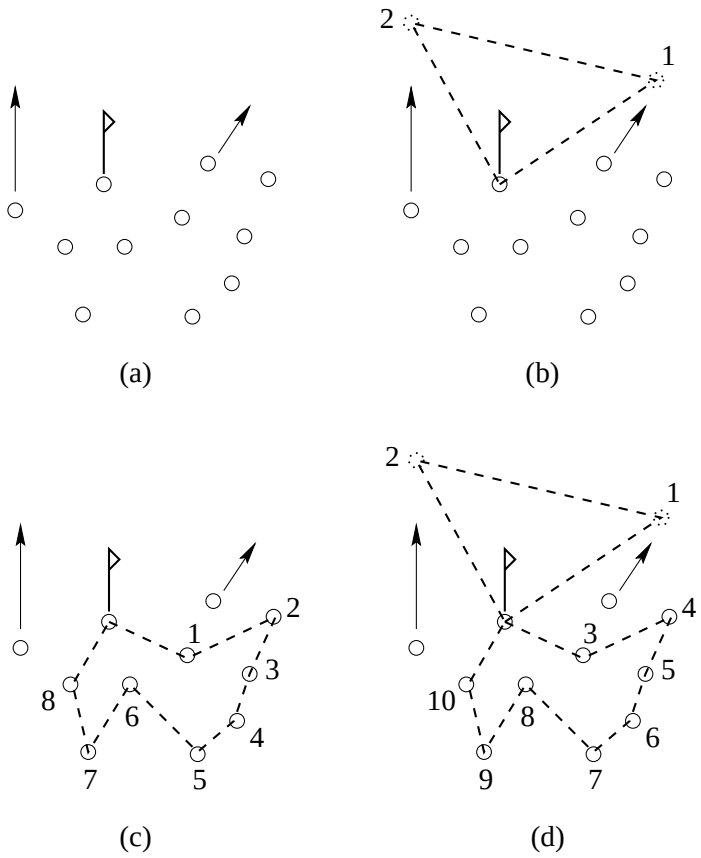

Figure 5: (a) In this example, we have two moving targets and the rest are stationary. (b) First, we find an optimal tour for the moving targets by trying all permutations. (c) Then, we use a classical TSP algorithm with performance bound α for intercepting the stationary targets. (d) Finally, we combine both tours into a single tour having a performance bound of $1 + \alpha$.

3.1 The Case When Targets Never Reach the Origin

Let d_i be the initial distance between the target s_i and the origin, i.e., $d_i = |p_i|$ where p_i is the position of s_i in the plane. If the target is moving towards the origin, then we define v_i to be negative, otherwise v_i is positive (see Figure 6).

First, we will determine an optimal intercept order for the receding targets, if all of the targets move away from the origin. Next, we will determine the order for intercepting targets when all targets move towards the origin. Finally, under certain conditions, we can combine the two orderings to obtain a single optimal ordering for a mix of approaching and receding targets.

Intuitively, when all targets move away from the origin, we would like to intercept the closest receding target early, because spending time chasing a more distant target will enable all of the other receding targets to move even further away. Also, we would like to intercept the fastest target earlier, because the longer we postpone this, the longer and further we would have to chase it later in order

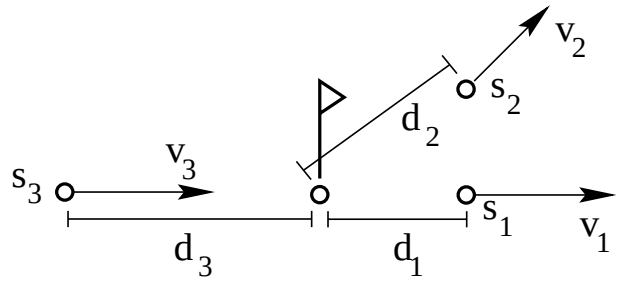

Figure 6: Two targets, s_1 and s_2, are moving directly away from the origin (i.e., on lines passing through the origin) at positive velocities v_1 and v_2, starting at distances d_1 and d_2. A third target s_3 approaches the origin, and thus its velocity v_3 is negative. The pursuer must return to the origin after intercepting each target.

to catch it. Thus, it would seem that we should strive to simultaneously minimize the distance-to-intercept while also maximizing the target's velocity when choosing the interception order for targets which move away from the origin.

Our solution for this variant of Moving-Target TSP is to intercept targets in increasing order of their ratios $\frac{d_i}{v_i}$. Note, that projecting backward in time, $\frac{d_i}{v_i}$ is the amount of time since the target left (or passed through) the origin. Thus, the ratio $\frac{d_i}{v_i}$ corresponds to what we refer to as the *age* of a target. The following theorem proves that an optimal algorithm must intercept the targets in non-decreasing order of their ages (i.e., younger targets should be intercepted first).

Theorem 7 *In Moving-Target TSP with Resupply where all targets move directly away from the origin, an optimal tour intercepts the targets in non-decreasing order of their respective ratios* $\frac{d_i}{v_i}$.

Proof: First, we show that the theorem is true when an instance of Moving-Target TSP with Resupply contains only two targets. Let $t_{1,2}$ be the time required to intercept s_1 and then intercept s_2. Similarly, let $t_{2,1}$ be the time required to intercept s_2 and then intercept s_1. We assume that s_1 is younger (the other case is symmetrical). We would like to show that $t_{1,2} \leq t_{2,1}$. The time required for the pursuer to intercept s_1 is $\frac{2 \cdot d_1}{v - v_1}$. Afterwards, intercepting s_2 will take time $2 \cdot \left(\frac{d_2 + \left(\frac{2 \cdot d_1}{v - v_1} \right) \cdot v_2}{v - v_2} \right)$. Thus, the total time to intercept s_1 first and then intercept s_2 is the sum:

$$t_{1,2} = 2 \cdot \left(\frac{d_1}{v - v_1} \right) + 2 \cdot \left(\frac{d_2 + \left(\frac{2 \cdot d_1}{v - v_1} \right) \cdot v_2}{v - v_2} \right)$$

Algebraic manipulation yields:

$$t_{1,2} - t_{2,1} = \frac{4 \cdot d_1 \cdot v_2 - 4 \cdot d_2 \cdot v_1}{(v - v_1) \cdot (v - v_2)}$$

If s_1 is younger than s_2, then $\frac{d_1}{v_1} \leq \frac{d_2}{v_2}$ and $d_1 \cdot v_2 \leq d_2 \cdot v_1$. This proves that $t_{1,2} \leq t_{2,1}$.

Given that Theorem 7 is true for two targets, we now show that it is also true for any number of targets. Assume towards contradiction that in an optimal tour, the pursuer intercepts two consecutive targets, first s_i and then s_j, in non-increasing order of their ages, i.e., $\frac{d_i}{v_i} \geq \frac{d_j}{v_j}$. First intercepting s_i and then intercepting s_j will require no less time than intercepting them in the reverse order, namely s_j first and then s_i. Thus, if the pursuer would alternatively intercept these two targets s_i and s_j in the reverse order, it would have time to wait at the origin right after intercepting the second target, and then continue the rest of the original tour using the original interception order. But by the No-Waiting Lemma (Lemma 1), this means that the original (presumably optimal) tour is not optimal, since it can be improved. Thus, all pairs of consecutive targets in an optimal tour must be intercepted in nondecreasing order of their $\frac{d_i}{v_i}$ ratios. □

Next, we analyze an analogous variant where all targets are approaching the origin. This variant is essentially the time reversal of the previous variant where all targets are receding away from the origin. The concept of the "age" of a target, however, is replaced with the analogous concept of the "dangerousness" of a target. The problem of intercepting targets moving towards the origin can thus be reformulated as requiring a pursuer to *defend* the origin (e.g., against incoming missiles or other threats).

The essential difference between the time reversal of the resupply variant where all targets move away from the origin and the case when all targets move towards the origin, is that in the latter scenario a target may pass through the origin and then move away from it. This possibility makes the problem quite complicated, because it causes an implicit change in direction which is absent in the first variant. We therefore consider only valid tours where no targets pass through the origin before the pursuer intercepts them.

Lemma 8 *Let all targets move towards the origin, and let T be the tour which intercepts targets in non-increasing order of their respective ratios $\frac{d_i}{-v_i}$. If T is a valid tour, then it is an optimal tour for Moving-Target TSP with Resupply.*

Proof: The proof is the same as for Lemma 7. Simply making the velocities of both targets negative does not change the inequalities. An optimal order still intercepts the targets in non-decreasing order of their ratios $\frac{d_i}{v_i}$, which means that we should intercept targets in the order of least dangerous to most dangerous if all targets are approaching the origin. □

Note that for a mixture of approaching and receding targets, we should intercept the receding targets first. The longer we wait to intercept these targets, the further away they will be able to travel before we eventually intercept them. Targets that move towards the origin should be allowed as much time as possible to come even closer to the origin. Therefore, if we assume that no targets will pass through the origin while we are pursuing the targets that move away from the origin, then we should first intercept targets that are moving away from the origin, and then intercept targets that are moving towards the origin. Further, if we can intercept the receding targets and still intercept the approaching targets in increasing order of their dangerousness before any target crosses the origin, then the optimal strategy for intercepting all of the targets is to first intercept the receding targets in order of increasing age and then to intercept the approaching targets in order of increasing dangerousness. This strategy

is formalized in the following theorem.

Theorem 9 *Let T be the tour which first intercepts all the targets which move away from the origin in non-decreasing order of their ratios $\frac{d_i}{v_i}$, and then intercepts the targets which move towards the origin in non-increasing order of their ratios $\frac{d_i}{-v_i}$. If T is a valid tour, then it is an optimal tour for Moving-Target TSP with Resupply.*

3.2 "Defending" the Origin Against Incoming Targets

In this subsection, we consider Moving-Target TSP when all the targets approach the origin. We first show that if we intercept targets in order of most dangerous to least dangerous, we will intercept all of the targets before any of them reach the origin. Next, we observe that from among all tours which intercept all targets before they reach the origin, the tour that intercepts targets in order of most dangerous to least dangerous is the longest. Finally, we will show that even this longest tour is never longer than twice an optimal tour which intercepts (in the best possible order) all the targets before they reach the origin.

Although we can prove that the strategy of intercepting targets in order of least dangerous to most dangerous is optimal when no targets intercept the origin, it is still open whether there is an efficient algorithm for determining an optimal intercept order when some targets may pass through the origin before being intercepted. However, we can prove that there always exists a tour that intercepts all the targets before they reach the origin.

Theorem 10 *A tour which intercepts the targets in non-decreasing order of $\frac{d_i}{-v_i}$ is the slowest (i.e., worst) valid tour.*

Proof: First, we will show that the slowest tour for this problem variant occurs when we intercept the targets in order of most dangerous to least dangerous. The proof is similar to that of Lemma 7. We have shown that the optimal order to intercept two targets is to first intercept the least dangerous target and then intercept the most dangerous target. Thus, if at any point in the tour we intercept a less dangerous target before intercepting a more dangerous one, then we can swap the intercept order of these two targets and induce a longer tour. When all targets are intercepted in order of most dangerous to least dangerous, no more such swaps to increase the time of the tour are then possible; therefore, such a tour is the slowest possible tour.

We now prove that this tour is valid. In such a tour, the pursuer intercepts the targets in order of most dangerous to least dangerous. After the pursuer intercepts the most dangerous target (i.e., the incoming target that will next reach the origin), the pursuer returns to to origin. Since the pursuer is faster than any target, it will return to the origin before the next most dangerous target can arrive. Then, the pursuer intercepts the next target before it reaches the origin, and so on. Given that the origin is a 0-size point, no target will reach it (though targets may come arbitrarily close). □

Lemma 8 and Theorem 10 lead to the obvious question: what is the shortest valid tour? A natural strategy would be to always intercept the least dangerous target unless intercepting that target would

allow the most dangerous target to actually reach the origin. In this case, we should intercept the least dangerous target that we can intercept and still obtain a valid tour. Unfortunately, this simple strategy does not always yield an optimal tour, as illustrated in the following example.

Consider the case when three targets, s_1, s_2, and s_3 are moving towards the origin. Without loss of generality, let the targets be numbered in increasing order of their dangerousness (i.e., the least dangerous is s_1 and the most dangerous is s_3). Suppose that after intercepting s_1, the tour that proceeds to intercept s_2 and then s_3 is invalid. On the other hand, intercepting s_2 and then s_3 yields a valid tour, if the pursuer does not start with intercepting s_1. According to the strategy above, the pursuer should intercept the targets in the order s_1, s_3, s_2. If s_1 is a stationary target which is very close to the origin, then the optimal tour instead starts with s_2 and s_3, and finishes with s_1 because tour (s_2, s_3) is faster than (s_3, s_2). Assume that s_2 is very slow, and that s_3 almost reaches the origin by the time the pursuer returns to the origin after intercepting s_2. Then the pursuer will waste considerable time chasing s_3 first. If s_3 is almost as fast as the pursuer, then the tour (s_3, s_2) will be about twice as long as the tour (s_2, s_3).

From the Moving-Target TSP instance described above, we see that there are instances of Moving-Target TSP for which the slowest tour may be up to twice as long as an optimal tour. Interestingly, we can also prove that this bound is tight, i.e., no valid tour requires more thant twice the time of an optimal valid tour, as follows.

Theorem 11 *For Moving-Target TSP with Resupply, when all targets move towards the origin, no valid tour is more than twice as long as an optimal valid tour.*

Proof: Enumerate the targets in order of least dangerous to most dangerous, and let T be an optimal valid tour. The slowest valid tour intercepts targets in order of most to least dangerous (i.e., $s_n, \ldots, s_1$). We will show that the slowest tour can be no more than twice the length of an optimal valid tour T, by iteratively transforming T into the slowest tour. Note that this transformation is equivalent to sorting, since an optimal tour can intercept targets in any order, and in the slowest order, the targets are sorted in decreasing order of their indices.

We denote the tour T as a list of targets where the left-most target will be intercepted first and the right-most target will be intercepted last. Our transformation starts with the right-most target in the original optimal tour T and gradually moves to the left, sorting all targets in decreasing order of their indices. In other words, at each step of our transformation, the current target s_i and all targets to the left of s_i occupy the same positions as in the tour T, while all of the targets to the right of s_i are already sorted in decreasing order of their indices. The step itself consists of removing target s_i from the current tour and inserting it into its proper position in the sorted list to the right.

Let t_i be the time required to intercept the target s_i in the original optimal tour T. We now show that each step of the transformation increases the total time of the tour by at most t_i. Note first that removing target s_i from the current tour cannot increase the total time of the tour. Indeed, the pursuer may wait for time t_i instead of intercepting the target s_i. Inserting the target s_i into its proper place in the sorted list decreases the time for intercepting targets to the right of its new location, because they will be intercepted later, i.e., when they will be closer to the origin. Similarly, the time to intercept

the target s_i in its new position is at most t_i. Thus, the insertion operation can increase the total time of the tour by at most t_i. Since each step of the transformation increases the cost of the tour by t_i for all s_i, the final tour may be at most twice the original cost. Note that this transformation results in a valid tour, since (i) the segment of the tour to the left of the new position of the target s_i is valid because this segment has been valid originally, and (ii) the segment of the tour to the right of s_i (including s_i) is valid by Theorem 10. □

Note that the Theorem above enables a broad class of $2 \cdot OPT$ approximation heuristics, which operate by simply producing valid tours, and/or modifying such tours while keeping them valid. Any such approach is guaranteed by Theorem 11 to yield solutions that are no worse than twice the optimal.

4 Multi-Pursuer Moving-Target TSP with Resupply

In this section, we address a generalization of Moving-Target TSP with Resupply when there are multiple pursuers. This generalization can also be considered as a dynamic version of the Vehicle Routing and Multiprocessor Scheduling problems. We focus on the case when targets move directly away from the origin on lines passing through the origin, and all k pursuers have the same top speed (normalized to 1). In the following two subsections we will consider two special cases: (1) where all the targets have the same age, and (2) where all the targets have the same speed.

In the presence of multiple pursuers, Moving-Target TSP may have different time objectives. In the Vehicle Routing Problem, the typical objective has been to minimize the *total tour time*, i.e., to minimize the sum over all pursuers for all time periods in which any pursuer is in operation[4]. Following the multiprocessor scheduling regime where a common objective has been to minimize the overall job *makespan*, we analogously seek to minimize the time when the last pursuer finally returns to the origin.

Note that achieving the makespan objective may be computationally more difficult than the total time objective, since for stationary targets, minimizing the makespan is equivalent to the NP-hard Multiprocessor Scheduling Problem, whereas the total time objective is invariant over all schedules. We will show that in the presence of moving targets, the problem of minimizing the total time also becomes NP-hard even in the case of two pursuers and where all targets are of the same age. On the other hand, we show that when all targets have the same speed, the total time can be minimized efficiently for any number of pursuers. If all targets have the same age, we also estimate the list scheduling error for multiple pursuers.

Before considering multiple pursuer problem variants, we prove the following useful lemma.

Lemma 12 *The total tour time for a single pursuer to intercept n targets s_i having speeds v_i is $T_n = T_{n-1} \cdot u_n + t_n$ where $u_i = \frac{1+v_i}{1-v_i}$ and $t_i = 2 \cdot \frac{d_i}{1-v_i}$ for any $i = 1, \ldots, n$ and $T_0 = 0$ (t_i is the time for a pursuer to intercept target s_i if it chases it first).*

Proof: Let T_i be the total time of a tour which intercepts targets $s_1, \ldots, s_i$. The total time of the tour which intercepts targets $s_1, \ldots, s_{i+1}$ is:

[4]We assume that each pursuer is in operation starting from time $t = 0$, until its final return to the origin.

$$T_{i+1} = T_i + 2 \cdot \frac{T_i \cdot v_{i+1} + d_{i+1}}{1 - v_{i+1}} = T_i \cdot \left(1 + 2 \cdot \frac{v_{i+1}}{1 - v_{i+1}}\right) + t_{i+1}$$

Since $1 + 2 \cdot \frac{v_{i+1}}{1-v_{i+1}} = u_{i+1}$, this yields:

$$T_{i+1} = T_i \cdot u_{i+1} + t_{i+1}$$

□

4.1 Targets with the Same Age

We start by applying Lemma 12 to the case of all targets having the same age.

Lemma 13 *Let $\frac{d_i}{v_i} = t$ be the same for all targets $s_1, \ldots, s_n$. For each target s_i, let $u_i = \frac{1+v_i}{1-v_i}$. The time required to intercept targets $s_1, \ldots, s_p$ with one pursuer is $t \cdot (\prod_{i=1}^{p} u_i - 1)$.*

Proof: Let t_i denote the time to intercept the target s_i if the pursuer chases s_i first, and express t_i in terms of the age $t = \frac{d_i}{v_i}$ and u_i.

$$t_i = 2\frac{d_i}{1 - v_i} = t\frac{2v_i}{1 - v_i} = t(u_i - 1)$$

By Lemma 12, $T_n = T_{n-1} \cdot u_n + t(u_n - 1)$, and therefore:

$$T_n + t = (T_{n-1} + t) \cdot u_n$$

Now repeatedly use the last equality to finish the proof:

$$T_n + t = (T_0 + t) \prod_{i=1}^{n} u_i = t \prod_{i=1}^{n} u_i$$

□

Lemma 13 implies that the problem of distributing n targets between two pursuers includes as a special case the well-known NP-hard problem of partitioning a set of numbers into two subsets, each having the same sum. Note that NP-hardness here does not directly follow from TSP (either classical or Moving-Target), because of the origin resupply requirement after each intercept.

Theorem 14 *Moving-Target TSP with Two Pursuers and Resupply is NP-hard when the objective is to minimize either the total time or the makespan, even when all targets have the same age.*

Proof: By Lemma 13, in order to minimize the total time, we need to partition the set of targets into two subsets, A and B, where the product of the u_i's of the targets in subset A plus the product of the u_i's of the targets in subset B is minimized. When we seek to minimize the makespan, we need to minimize the maximum of these two products. If it is possible to partition the targets into subsets A

and B with the same product of the u_i's, then this yields an optimal partition of the targets among the two pursuers for either objective. We refer to this problem as the Product Bipartition (PB) problem.

We will show that the PB problem is strongly NP-hard by a reduction from the Subset Product (SP) problem (see [3], page 224). The SP problem seeks a subset Y of numbers from a given set X where the product of the numbers in Y equals a given number z (both the multisets X and Y may contain duplicate numbers). We will construct an instance X' of the PB problem such that X' has a solution if and only if the given instance of the SP problem (X, z) has a solution. Let x denote the product of all the numbers in X, and let the corresponding instance of the PB problem consist of the set $X' = X \cup \{x\} \cup \{z^2\}$. Note that the product of all numbers in X' is x^2z^2. If the SP instance (X, z) has a solution Y, then $Y' = Y \cup \{x\}$ is the solution for the PB instance for the set of numbers X'. Let Y' be a solution for the PB instance for X'; then the product of all the numbers in Y' is xz. Since the product of *all* elements in X equals x, Y' must contain either $\{x\}$ or $\{z^2\}$. The SP instance (X, z) will have a solution Y in both cases, since if $x \in Y'$, then $Y = Y' - \{x\}$, and on the other hand if $z^2 \in Y'$, then $Y = X \setminus Y'$. □

Lemma 13 yields a reduction of Moving-Target TSP with k pursuers (in the case when all targets have the same age) to the Multiprocessor Scheduling Problem: given a set of n jobs with processing times t_i, and k equivalent processors, find a schedule having the minimum makespan. There exist many heuristics for the Multiprocessor Scheduling Problem, including list scheduling, longest processing, and polynomial-time approximation schemes [6]. Unfortunately, the error estimates for these heuristics cannot be transformed into bounds for Moving-Target TSP with Resupply and multiple pursuers, because a multiplicative factor corresponds to the exponent in such transformations. A tour in which the next available pursuer is assigned to the next target from the list (to which no pursuer has yet been assigned) is called a *list tour*.

Theorem 15 *Let $\frac{d_i}{v_i} = t$ be the same for all targets $s_i, \ldots, s_n$. Then the makespan and the total time of the list tour are at most $\max_{i=1,\ldots,n} \frac{1+v_i}{1-v_i}$ times the optimal makespan and total time, respectively.*

Proof: The list tour is obviously optimal if the number of pursuers k is at least as large as the number of targets n. We therefore assume that $k < n$, and normalize time so that $t = \frac{d_i}{v_i} = 1$.

Let $P = \prod_{i=1}^{n} u_i$. By Lemma 13, the total time and makespan of an optimal tour with k pursuers is at least $opt\_total \geq k(P^{1/k} - 1)$ and $opt\_make \geq P^{1/k} - 1$, respectively. Let $P_j = \prod_{i \in I_j} u_i$, $j = 1, \ldots, k$, where I_j are the indices of targets intercepted by j-th pursuer in the list tour. Let $P_l = \max_{j=1}^{k} P_j$ and let t_{last} be the time when the l-th pursuer starts chasing its last target s_{last}. We then have $\frac{P_l}{u_{last}} - 1 = t_{last}$.

Since in the list tour *all* pursuers are chasing targets before t_{last}, we have $P_j - 1 \geq t_{last}$ and $P_j \geq \frac{P_l}{u_{last}}$ for $j \neq l$. This yields:

$$\begin{aligned} t_{last} &\leq \min\left\{\min_{j \neq l} P_j, \frac{P_l}{u_{last}}\right\} - 1 \\ &\leq P^{1/k} - 1 \end{aligned}$$

Thus, the list tour makespan $list\_make$ is at most $list\_make = P_l - 1 \leq P^{1/k} \cdot u_{last} - 1$, and the makespan approximation ratio is at most:

$$\begin{aligned} \frac{list\_make}{opt\_make} &\leq \frac{P^{1/k} \cdot u_{last} - 1}{P^{1/k} - 1} \\ &\leq u_{last} \cdot \frac{P^{1/k} - \frac{1}{u_{last}}}{P^{1/k} - 1} \\ &\leq u_{last} \end{aligned}$$

Similarly, the list tour total time $list\_total$ is at most $list\_total \leq k(\max_{j=1}^{k} P_j - 1) = k \cdot P_l - k$ and the total time approximation ratio is at most:

$$\frac{list\_total}{opt\_total} \leq \frac{k \cdot (P^{1/k} \cdot u_{last} - 1)}{k(P^{1/k} - 1)} \leq u_{last} \qquad \square$$

This theorem implies, for example, that if the speed of any target is at most half the speed of pursuer, then the list tour interception order has an approximation ratio of 3.

4.2 Targets Moving with Equal Speed

In the multiple pursuer case where all targets have the same speed v, we can efficiently compute an optimal solution. Similarly to the method outlined above, we order the targets by increasing value of their $\frac{d_i}{v_i}$, but since $v_i = v$ is the same for all targets, this reduces to ordering the targets by their initial distance from the origin. Thus, the following natural strategy suffices: at the time when a pursuer resupplies at the origin, send that pursuer to intercept the next closest target to the origin. We call the resulting tour "CLOSEST", and we prove several lemmas which help establish the optimality of this tour.

While Lemma 12 is a general result which is applicable when targets may have different speeds, it has an important corollary when the speeds of all targets are the same, and thus u_i becomes constant for all i. Using this observation and reformulating the result of Lemma 12 using summation notation, we obtain the following corollary.

Lemma 16 *The tour cost for a single pursuer to intercept n targets s_i all having the same speed v, is $T_n = \sum_{i=1}^{n} t_i \cdot u^{n-i}$ where $u = \frac{1+v}{1-v}$ and $t_i = 2 \cdot \frac{d_i}{1-v}$ (t_i is the time for a pursuer to reach target s_i if it intercepted that target first).*

Proof: Since $T_0 = 0$ and $T_i = T_{i-1} \cdot u + t_i$, we obtain $T_1 = t_1$. We will prove the lemma inductively. Using Lemma 12, we obtain:

$$\begin{aligned} T_n &= T_{n-1} \cdot u + t_n \\ &= (\sum_{i=1}^{n-1} t_i \cdot u^{n-i}) \cdot u + t_n \end{aligned}$$

$$= \sum_{i=1}^{n} t_i \cdot u^{n-i}$$

□

In the rest of this section we derive properties of solutions yielded by this algorithm. We then use Lemma 16 to show that swapping some of the targets between pursuers will result in tours of equal total time. Finally, we show that any optimal tour can be transformed by such swaps into the tour produced by our algorithm, which implies that our algorithm always produces optimal solutions. We begin by proving the following lemma.

Lemma 17 *Given a pair of targets which are intercepted by different pursuers (all having the same speed) such that each pursuer has an equal number of targets left to pursue after intercepting them, these targets may be swapped between the two pursuers (i.e., each pursuer may chase its counterpart's target instead of its own), while keeping the total required time the same.*

Proof: For simplicity, we reorder the targets of each pursuer in their reverse interception order and denote them $s_1, s_2, \ldots, s_n$. As in the proof of Lemma 12, let t_i be the time required to intercept target s_i and return to base, if a pursuer were to intercept that target first.

The cost of the tour for each pursuer to intercept n targets was proven in Lemma 12 to be $t_1 + t_2 \cdot u + \ldots + t_n \cdot u^{n-1}$, where the constant u only depends on the velocity of the pursuer relative to the targets. Thus, we may swap the targets s_i of any two pursuers, and the total time for all pursuers will still remain the same. □

Finally, we prove the following theorem.

Theorem 18 *The tour CLOSEST is an optimal tour for Moving-Target TSP with Resupply and multiple pursuers (all having the same speed), where all targets have equal speeds.*

Proof: Let each of $G_1, \ldots, G_m$ (where $m = \lceil \frac{n}{k} \rceil$) be a group of k targets, each intercepted by one of the k pursuers, such that after its interception there are exactly $i - 1$ targets remaining to be pursued for that pursuer. In particular, the first group G_1 consists of targets which were intercepted first, and the last group G_m may be "underfilled" if the number of targets is not a multiple of k. By Lemma 17, swapping targets within each group G_i results in tours of the same total time. We need to show that one such tour is the CLOSEST tour.

Without loss of generality, we may assume that all targets begin with a different initial distance from the origin. It is sufficient to show that for any two targets $s_j \in G_i$ and $s_{j'} \in G_{i+1}$, we have $d_j < d_{j'}$. By Theorem 7, we know that in an optimal tour for a single pursuer, we should intercept all targets in non-decreasing order of their ratios $\frac{d_i}{v_i}$, but in this case, all of the v_i's are identical. Therefore, in any optimal tour, each pursuer must intercept the targets in non-decreasing order of their original distance d_i from the origin. In any optimal tour, where s_j and $s_{j'}$ are intercepted by the same pursuer, the target s_j is intercepted before the target $s_{j'}$, and therefore $d_j < d_{j'}$. □

5 Conclusion and Future Research

We formulated a Moving-Target version of the classical Traveling Salesman Problem, and provided the first heuristics and performance bounds for this problem and for other time-dependent variants. Topics for future research include providing approximation algorithms for more general variants of Moving-Target TSP (e.g., where targets are moving with non-zero accelerations, and/or along non-linear paths). Also, it would be interesting to generalize our results for Moving-Target TSP with Resupply to cases when each pursuer may intercept multiple targets before requiring resupply. Alternatively, any non-approximability results for such cases would be of interest as well.

Finally, while our formulations in this paper focused on minimizing the total *time* of a tour, it would also be of interest to explore the analogous problem variants where the goal is to instead minimize the total *distance* traveled by the pursuer(s). While for classical (stationary target) TSP these two goals of minimizing travel time vs. distance traveled are equivalent, for Moving-Target TSP these two objectives are very different, each leading to distinct properties, strategies, and results (e.g., when minimizing the total distance objective, the No-Waiting Lemma does not apply, as there are counter-example instances where waiting is actually beneficial).

6 Acknowledgments

We thank Piotr Berman for his help in developing the $O(n^2)$ dynamic programming algorithm for one-dimensional Moving-Target TSP. We also thank Doug Bateman, Doug Blair, Mike Nahas, John Karro, Cheryl Rembold, Bhupinder Sethi, Anish Singh, Tongtong Zhang for proofreading assistance and various helpful discussions. Additional related works may be found at www.cs.virginia.edu/robins.

References

[1] S. Arora, *Polynomial time approximation schemes for euclidean TSP and other geometric problems*, in Proc. IEEE Symp. Foundations of Computer Science, 1996, pp. 2–11.

[2] T. H. Cormen, C. E. Leiserson, and R. Rivest, *Introduction to Algorithms*, MIT Press, 1990.

[3] M. R. Garey and D. S. Johnson, *Computers and Intractability: a Guide to the Theory of NP Completeness*, W. H. Freeman, San Francisco, 1979.

[4] M. Hammar and B. J. Nilsson, *Approximation Results for Kinetic Variants of TSP*, in Proc. International Colloquium on Automata, Languages, and Programming, 1999, pp. 392–401. *Lecture Notes in computer Science* **1644**.

[5] C. H. Helvig, G. Robins, and A. Zelikovsky, *Moving target TSP*, in Proc. European Symposium on Algorithms, Venice, Italy, August 1998, pp. 453–464. In Algorithms-ESA'98, G. Bilardi, G. F. Italiano, A. Pietracaprina and G. Pucci (eds.) *Lecture Notes in Computer Science* **1461**.

[6] D. Hochbaum, ed., *Approximation Algorithms for NP-Hard Problems*, PWS Publishing Company, Boston, MA, 1995.

[7] A. V. Kryazhimskiy and V. B. Savinov, *The travelling-salesman problem with moving objects*, J. Comput. Syst. Sci. Int., (1993), pp. 144–148.

[8] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy, and D. B. Shmoys, *The Traveling Salesman Problem: a Guided Tour of Combinatorial Optimization*, John Wiley and Sons, Chichester, New York, 1985.

[9] C. Malandraki and M. S. Daskin, *Time Dependent Vehicle Routing Problems: Formulations, Properties and Heuristic Algorithms*, Journal of Transportation Science, 26 (1992), pp. 185–200.

[10] C. Malandraki and R. B. Dial, *A Restricted Dynamic Programming Heuristic Algorithm for the Time Dependent Traveling Salesman Problem*, European Journal of Operations Research, 90 (1996), pp. 45–55.

[11] G. Reinelt, *The Traveling Salesman: Computational Solutions for TSP Applications*, Springer Verlag, Berlin, Germany, 1994.

[12] R. J. V. Wiel and N. V. Sahinidis, *Heuristic Bounds and Test Problem Generation for the Time-Dependent Traveling Salesman Problem*, Journal of Transportation Science, 29 (1995), pp. 167–183.