Moving-Target TSP and Related Problems[*]

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

Department of Computer Science, University of Virginia, Charlottesville, VA 22903
[†] Department of Computer Science, UCLA, Los Angeles, CA 90095

Abstract. Previous literature on the Traveling Salesman Problem (TSP) implicitly 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 a set of targets which move with constant velocities in a minimum amount of time. We propose approximate and exact algorithms for several natural variants of Moving-Target TSP. Our implementation of the exact algorithm of One-Dimensional TSP is available on the Web.

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 [4] [7]). 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 $\mathbf{v}_i$ from an initial position $p_i \in \Re^n$, and given a *pursuer* starting at the origin and having maximum speed $v > |\mathbf{v}_i|$ for $i = 1, \ldots, n$, 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 [5]. These works give exponential-time algorithms based on a mixed integer linear programming formulation and dynamic programming [5] [6] [9]. One major drawback of such general formulations is that they do not yield efficient bounded-cost heuristics,

[*] Professor Robins is supported by a Packard Foundation Fellowship and by National Science Foundation Young Investigator Award MIP-9457412. The corresponding author is Gabriel Robins, (804) 982-2207, robins@cs.virginia.edu. Additional related papers and our code may be found at http://www.cs.virginia.edu/~robins/

since they generalize TSP without the triangle inequality, which admits no efficient approximation bounds unless $P = NP$ [1].

In this paper, we 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 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.

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 targets are moving strictly away from (or else 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 are constrained to 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 average completion time objectives. We also develop an approximation algorithm for the case when, if 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 some future research directions. Some proofs are condensed or altogether omitted due to page limitations. See [2] or our Web site http://www.cs.virginia.edu/~robins/ for a more complete version of this paper.

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

[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 $\mathbf{v}_i$ from an initial position $p_i \in \Re^n$ 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.

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

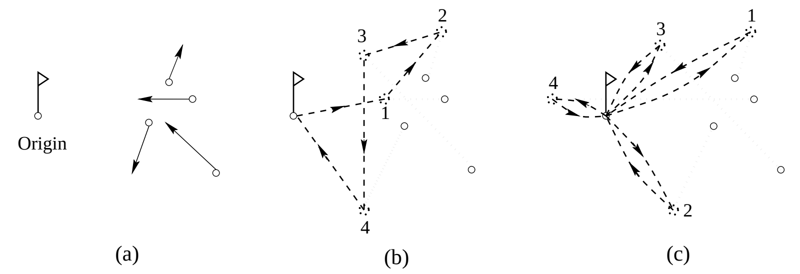

Fig. 1. (a) An instance of Moving-Target TSP. (b) A shortest-time tour (dashed line) which begins at the origin (flag) and intercepts all of the targets. (c) In Moving-Target TSP with Resupply, the pursuer must return to the origin after intercepting each target (the optimal interception schedule is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

ratio. In this section, we consider two variants: (1) when targets are confined to a single line[4], and (2) when the number of moving targets is small. The following lemma is essential for our analyses of Moving-Target TSP and its variants:

Lemma 1 *The pursuer must always move at maximum speed in any optimal Moving-Target TSP tour.*

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 based on dynamic programming. To see that this one-dimensional variant is not trivial, consider the following generalization of the exact algorithm when all targets are stationary. 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 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 can have unbounded error. Consider the case when there are four targets. Two of the targets are extremely close to the origin and moving away from the origin very fast, but they are on either side of the pursuer; the other two targets are again 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, but if we first intercept all of the targets on one side of the origin, then we will later need to spend much more time chasing the fast target on the other side.

[4] Historical note: after we submitted this paper to ESA'98, we received an unpublished technical report [8] that claims to solve the one-dimensional variant in $O(n^3)$ time; our algorithm for the same problem runs in time $O(n^2)$.

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. All the targets that the pursuer would intercept in the time period between $t - \delta$ and $t + \delta$ are moving slower than s, and therefore cannot pass s. These targets will remain "sandwiched" between the pursuer and target s, and thus will automatically be intercepted later by the pursuer on its way to the faster target s. By Lemma 1, the original tour is not optimal, because it is equivalent to a tour with a waiting period. □

Thus, we may view a tour as a sequence of snapshots at the moments when the pursuer intercepts the fastest target which was ahead of it. We will later show that all of the relevant information in any such snapshot 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.

Note that Lemma 2 implies that there are at most two possible transitions from any state $A = (s_k, s_f)$ at assigned time $t(A)$. 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 $t(A)$.

Our algorithm works as follows. 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.

We sort 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. Our algorithm iteratively traverses each state in this sorted list. Note that transitions cannot go backwards with respect to this order.

When we traverse a given state A in the algorithm, we do one of three things: (1) if the state has no transitions to it, then 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 to the final state A_{final}; or (3) 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 such that $t(B) = min\{t(A)+t(\tau)|\forall\tau : A \rightarrow B\}$. This is the shortest sequence of transitions from A_0 to B. We need constant time on average to traverse each of the $O(n)$ transitions.

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 the optimal tour. For each pair of turning points, we find the subset of targets which are intercepted between them. Once we have partitioned the targets into subsets, we sort the targets inside the subset by their interception times. Finally, we merge the sorted subsets into a combined interception order, yielding the solution.

Theorem 3 *The above algorithm for One-Dimensional Moving-Target TSP finds an optimal tour in $O(n^2)$ time.*

We have implemented this algorithm using the C++ programming language. Computational benchmarks against an optimal algorithm empirically confirm its correctness. Our implementation is available on the Web.

2.2 Heuristics when the Number of Moving Targets is Small

In this subsection, we consider Moving-Target TSP when most of the targets are stationary (while a few targets may be moving). From among the many existing approximation algorithms for classical (stationary) TSP [4], choose one such heuristic, having performance bound α. Using this algorithm for stationary targets, we can construct an efficient algorithm (see Figure 2) with performance bound of $1 + \alpha$ when the number of moving targets is sufficiently small.

Theorem 4 *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.*

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. We assume that targets all move either directly towards or away from the origin. These assumptions are natural because the projections of target velocities onto radial lines through the origin are approximately constant when the target is either (1) slow, (2) far from the origin, or (3) already moving along a radial line.

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. In Subsection 3.1, we restrict the problem by considering the scenario when the shortest

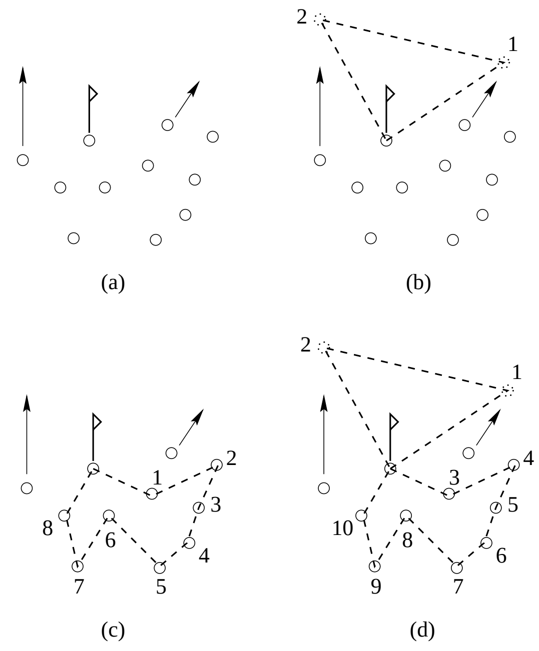

Fig. 2. (a) In this example, we have two moving targets and the rest are stationary. (b) First, we find the optimal tour for the moving points by trying all permutations. (c) Then, we use a classical TSP algorithm with performance bound α for intercepting stationary targets. (d) Finally, we combine both tours into a single tour with performance bound $1 + \alpha$.

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 is). We show that there is a simple way to determine the optimal target interception order for this scenario.

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 targets, and show that there is always a valid tour regardless of the number of targets. Then, we prove that the longest tour can be only twice the length of the shortest valid tour.

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 3).

First, we will determine the optimal order to intercept the receding targets, if all of the targets move away from the origin. Later, we will determine the

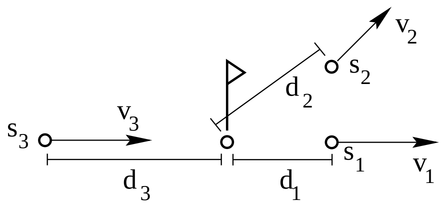

Fig. 3. Two targets, s_1 and s_2, are moving directly away from the origin at positive velocities v_1 and v_2, starting at distances d_1 and d_2. A third target s_3 approaches the origin, and thus its velocity v_3 is negative. The pursuer returns to the origin after intercepting each target.

order for intercepting targets when all targets move towards the origin. Finally, under the right conditions, we can combine the two orderings to obtain a single optimal ordering for a mix of approaching and receding targets.

Our solution for the case when targets move away from the origin 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 intercepted the origin. Thus, the ratio $\frac{d_i}{v_i}$ corresponds to what we denote as the *age* of a target. The following theorem proves that the optimal algorithm must intercept the targets in non-decreasing order of their ages (i.e. younger targets are intercepted first).

Theorem 5 *In Moving-Target TSP with Resupply when all targets move directly away from the origin, the optimal tour intercepts the targets in non-decreasing order of their respective ratios* $\frac{d_i}{v_i}$.

Proof sketch: 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 $2 \cdot (d_2 + \frac{2 \cdot d_1}{v - v_1} \cdot v_2)/(v - v_2)$. 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 5 is true for two targets, we show that it is also true for any number of targets. Assume towards the contradiction that in the optimal tour, the pursuer intercepts two consecutive targets, first s_i and then s_j, in non-increasing order of their ages, i.e. $d_i/v_i \geq d_j/v_j$. 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 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 d_i/v_i ratios. $\square$

Next, we analyze an analogous variant where all targets are approaching the origin. This variant is the time reversal of the previous variant where all targets are receding. The concept of the "age" of a target, however, is replaced with the 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 (against, e.g., incoming missiles).

The 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 a target may pass through the origin and then move away from it. This possibility makes the scenario quite complicated, because it causes an implicit change in direction which is absent in the first variant. Therefore, we consider only valid tours where no targets pass through the origin before the pursuer intercepts them.

Lemma 6 *Let all targets move towards the origin, and let T be the tour which intercepts targets in non-decreasing 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.*

Note that if we have a mixture of approaching and receding targets, then we should intercept the receding targets first. The longer we wait to intercept these targets, the further away they will be when we do intercept them. Targets that move towards the origin should be allowed as much time as possible to come close 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 tour 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 is formalized in the following theorem.

Theorem 7 *Let T be the tour which first intercepts targets moving away from the origin in non-decreasing order of their ratios $\frac{d_i}{v_i}$ and then intercepts the targets moving towards the origin in non-decreasing 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 Approaching Targets

In this subsection, we consider Moving-Target TSP when all 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 can show that even this longest tour is never longer than twice the optimal tour which intercepts all 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 the optimal intercept order when some targets may pass through the origin before being intercepted. We can prove, however, that there is always a tour that intercepts all targets before they reach the origin.

Theorem 8 *Let T be a tour that intercepts targets in non-increasing order of $\frac{d_i}{v_i}$. Then T is the slowest valid tour.*

Lemma 6 and Theorem 8 lead to a natural question: what is the shortest valid tour? (i.e. the tour which intercepts all targets, yet prevents any targets from passing the origin). A natural strategy would be to always intercept the least dangerous target unless intercepting that target would allow the most dangerous target to reach the origin. In this case, we can intercept the least dangerous target that we can intercept and still obtain a valid tour. Unfortunately, this simple algorithm does not always return an optimal tour.

There are instances of the Moving-Target TSP for which the slowest tour may be twice as long as optimal. We can also prove that this bound is tight, i.e. no valid tour takes time more than twice the optimal valid tour.

Theorem 9 *For Moving-Target TSP with Resupply, when all targets move towards the origin, no valid tour is more than twice the optimal valid tour.*

Proof: We enumerate the targets in order of least dangerous to most dangerous. Let T be an optimal valid tour. The slowest valid tour intercepts targets in order of most dangerous 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 the optimal valid tour T by iteratively transforming T into the slowest tour. Note that this transformation is equivalent to sorting, since the optimal tour can intercept targets in any order, and in the slowest order, the targets are sorted in decreasing order of their indices.

We write the tour T as a list of targets where the left-most targets 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, say s_i, and all targets to the left of s_i hold 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 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. Now, we 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, inserting 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. $\square$

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 special dynamic version of the well-known Vehicle Routing and Multiprocessor Scheduling Problems. We restrict ourselves to the case when targets move strictly away from the origin and all k pursuers have equal speed (normalized to 1).

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 of all periods of time in which a pursuer is in operation[5]. In multiprocessor scheduling, a common objective has been to minimize the *makespan*, or in other words, to minimize the time when the last pursuer returns to the origin.

Note that achieving the makespan objective may be more 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. In the presence of moving targets, the problem of minimizing the total time also becomes NP-hard. To show this, we need the following lemma.

Lemma 10 *Let $d_i/v_i = t$ be the same for all targets s_i, where $i = 1, ..., n$. For each target s_i, let $u_i = (1 + v_i)/(1 - v_i)$. The time required to intercept targets $s_1, ..., s_p$ with one pursuer is equal to $t \cdot (\prod_{i=1}^{p} u_i - 1)$.*

Lemma 10 implies that the problem of distributing n targets between two pursuers includes as a special case the well-known NP-hard problem of partitioning of a set of numbers into two subsets with each having the same sum.

Theorem 11 *Moving-Target TSP with Two Pursuers and Resupply is NP-hard when the objective is to minimize the total time.*

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

4.1 Targets with the Same Age

Lemma 10 yields a reduction of Moving-Target TSP with k pursuers (in the case when all targets have the same age) to the standard 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 are several different heuristics for the Multiprocessor Scheduling Problem: list scheduling, longest processing, and many others, including a polynomial-time approximation scheme [3]. 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 (to which no pursuer has yet been assigned) from the list will be called a *list tour.*

Theorem 12 *Let $\frac{d_i}{v_i} = t$ be the same for all targets s_i, for $i = 1, \ldots, n$. Then the list tour takes total time at most $\max_{i=1,\ldots,n} \frac{1+v_i}{1-v_i}$ times optimal.*

This theorem implies 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 case when there are multiple pursuers and 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 d_i/v_i, but since $v_i = v$ is the same for all targets, this reduces to ordering the targets by increasing initial distance from the origin. We have found that a very natural strategy suffices: send each pursuer after the next closest target to the origin at the time when pursuer resupplies at the origin. We call the resulting tour "CLOSEST", and we can prove that it is optimal.

Theorem 13 *The tour CLOSEST is the optimal tour for Moving-Target TSP with multiple pursuers when targets have equal speeds.*

Proof sketch: It can be shown that the cost of a tour for a single pursuer to intercept n targets having equal speeds $v_i = v$ is $T_n = \sum_{i=0}^{n} t_i \cdot c^{n-i+1}$ where $c = 1 + 2 \cdot \frac{v}{1-v}$ and $t_i = 2 \cdot \frac{d_i}{1-v}$ (in other words, t_i is the time for a pursuer to intercept a target if it intercepted that target first). In the tour CLOSEST, each of the k pursuers will intercept targets $s_i, s_{k+i}, \ldots, s_{k[n/k]+i}$, where $i = 1, \ldots, k$. Given a pair of targets which are intercepted by different pursuers such that each pursuer has an equal number of targets left to pursue after intercepting them, the targets may be swapped between the two pursuers (i.e. each pursuer may intercept its counterpart's target instead of its own), while keeping the total time required the same. This allows us to transform any optimal tour into the tour CLOSEST, by using a sequence of target swappings between different pursuers, without increasing the total tour time. Therefore, the tour CLOSEST requires the optimal total time. $\square$

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 constant accelerations), or else proving that that no polynomial-time approximation algorithms exist unless $P = NP$. Also, it may be possible to generalize our results for Moving-Target TSP with Resupply to cases when the pursuer may service multiple targets before requiring resupply.

6 Acknowledgements

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 Mike Nahas for implementing that algorithm and for finding areas where additional clarification was needed. Finally, thanks go to Bhupinder Sethi and Doug Bateman for numerous fruitful discussions.

References

1. T. H. Cormen, C. E. Leiserson, and R. Rivest. *Introduction to Algorithms.* MIT Press, 1990.
2. C. H. Helvig, G. Robins, and A. Zelikovsky. Moving target tsp and related problems. Technical Report CS-98-07, Department of Computer Science, University of Virginia, December 1997.
3. D. Hochbaum, editor. *Approximation Algorithms for NP-Hard Problems.* PWS Publishing Company, Boston, MA, 1995.
4. 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.
5. C. Malandraki and M. S. Daskin. Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Journal of Transportation Science*, 26:185–200, August 1992.
6. C. Malandraki and R. B. Dial. A restricted dynamic programming heuristic algorithm for the time dependent traveling salesman problem. *European Journal of Operations Research*, 90:45–55, 1996.
7. G. Reinelt. *The Traveling Salesman: Computational Solutions for TSP Applications.* Springer Verlag, Berlin, Germany, 1994.
8. G. Rote. The travelling salesman problem for moving points on a line. Technical Report 232, Technische Universitat Graz, 1992.
9. R. J. Vander Wiel and N. V. Sahinidis. Heuristic bounds and test problem generation for the time-dependent traveling salesman problem. *Journal of Transportation Science*, 29:167–183, May 1995.