Synchronized Traveling Salesman Problem

Gyula Pap and József Varnyú

September 2018

Synchronized Traveling Salesman Problem

Gyula Pap * and József Varnyú **

Abstract

We consider a variation of the well-known traveling salesman problem in which there are multiple agents who all have to tour the whole set of nodes of the same graph, while obeying node- and edge-capacity constraints that require no two agents may "crash". We consider the simplest model in which the input is an undirected graph with all capacities equal to one. A solution to the synchronized traveling salesman problem is called an "agency". Our model puts the synchronized traveling salesman problem in a similar relation with the traveling salesman problem as the so-called evacuation problem, or the well-known dynamic flow (flow-over-time) problem is in relation with the minimum cost flow problem.

We measure the strength of an agency in terms of number of agents which should be as large as possible, and the time horizon which should be as small as possible. Beside some elementary discussion of the notions introduced, we establish several upper and lower bounds for the strength of an agency under the assumption that the input graph is a tree, or a 3-connected 3-regular graph.

1 Introduction

The purpose of this paper is to explore and study some new problems that are defined from a natural merger of the traveling salesman problem, and the dynamic flow problem.

Our first starting point is the traveling salesman problem, which is one of the best known problems from combinatorial optimization, and there has been a wide range of variations to the original traveling salesman problem that have been investigated over the decades. In this paper we would consider a variation of the traveling salesman problem that has multiple salesman touring the same network. Of course earlier research has been done with multiple salesmen, including the so-called multiple traveling salesmen problem (Laporte, Norbert, [9]), vehicle routing problems (Christofides,

---

Mingozzi, Toth, [5]), price-collecting traveling salesman problem (Balas, [2]), and others. The problem variation considered in this paper will be completely different from those.

The second starting point is the dynamic flow problem first introduced by Ford, Fulkerson [6], also known as flows over time, which came as a variation to the flow problem in which we consider a flow as a dynamic notion that develops over time. See Skutella, [10] and Kotnyek [8] for a broad survey about this topic. The idea is to replace the static notion of a flow by a dynamic notion in which flow is described as the time-dependent motion of particles, either discrete, or continuous. The evacuation problem, quickest transshipment problem (see Hoppe, Tardos [7]), fastest flow problem, dynamic transportation [3], and many others are special cases or variations of the dynamic flow problem.

In this paper we would develop a notion of dynamic (or synchronized) traveling salesmen, that correspond to the original traveling salesman problem in a similar way as dynamic flows correspond to flows. The idea, basically, is to find a way to put several salesmen, "agents", in a network so that they all need to perform a traveling salesman tour, and the lot of them must obey network capacities. In this setting we may consider two different kind of objectives: try to add as many agents to the networks as possible, or try to find tours for the agents to finish in as short time as possible. Of course the tours of the agents must be coordinated between each other in order to let them obey the network capacities, thus our wording to call this problem the synchronized traveling salesmen problem. One case of this problem has been considered in [1]. A more precise definition follows below.

2 Problem Setting

Let $G = (V, E)$ be an undirected graph with $n = |V|$. A sequence $v(0), v(1),$ $v(2), \cdots, v(T)$ of nodes $v(t) \in V$ is called a **walk (with parking)** if for all $t = 0, 1, \cdots, T - 1$ we have either $v(t) = v(t + 1)$ or $v(t)v(t + 1) \in E$. A sequence $v(0), v(1), v(2), \cdots, v(T)$ of nodes $v(t) \in V$ is called a **walk (without parking)** if for all $t = 0, 1, \cdots, T - 1$ we have $v(t)v(t + 1) \in E$. Most of the time in this paper, we will consider walks with parking, so even if not expressly written, a **walk** means a walk with parking. For a walk, T is called the **time horizon**.

A sequence $v(0), v(1), v(2), \cdots, v(T)$ of nodes $v(t) \in V$ is called a **traveling salesman tour (with/ without parking)**, if it is a walk (with/without parking) in the graph such that every node appears at least once, and the tour returns to its initial node, that is, $v(0) = v(T)$. A traveling salesman tour with parking is called a **tour**, for short. As a tour returns to its initial node, we may consider a tour by time units modulo T (which is similar to picturing a tour as if it were to repeat all over after the time horizon).

Of course in the usual setting of the traveling salesman problem, there is no need for parking, because it is just a waste of time or cost; here in our setting, however, parking may be needed to avoid two salesmen of crashing into each other: one of them would wait until the other one passes a node or an edge, and move on afterwards.

Vaguely speaking: the point is that we introduce a setting in which there are multiple salesmen touring the same graph at the same time so that they are not allowed to crash into each other. In this setting it makes a lot of sense to allow parking, and this is what we do in this paper.

In the synchronized traveling salesman problem, we consider an **"agency"** of a number of salesmen that each one of them has to do a tour with the same time horizon, though they need to start from different initial nodes, and must not "crash" into each other. Essentially there is a unit capacity for each node or each edge. More precisely, we define an agency as follows.

Definition 2.1. Let $k, T \in \mathbb{Z}_+$ be positive integers. k denotes the number of salesmen, or **agents**, and T denotes the joint time horizon. Let $a_i(t) \in V$ be the node where agent i is supposed to be at time t, where $i = 1, 2, \cdots, k$, and $t = 0, 1, \cdots, T$. This is called an **agency** with time horizon T and k agents if for any fixed i, $v_i(t)$ is a tour (with parking). In practical terms, each i denotes an agent that moves along the unit-length edges of the graph, so that every agent makes a traveling salesman tour of time horizon T.

Definition 2.2. If i and j are agents from the same agency, $i \neq j$, then we say that these agents i and j **crash in a node** v at time t if $v = a_i(t) = a_j(t)$. In practical terms, this may be understood as the two agents walking around the graph, and at time t they both occupy in the same node v.

Definition 2.3. If i and j are agents from the same agency, $i \neq j$, then we say that these agents i and j **crash in an edge** $uv \in E$ at time t if $v = a_i(t) = a_j(t+1), u = a_i(t+1) = a_j(t)$ or $v = a_i(t+1) = a_j(t), u = a_i(t) = a_j(t+1)$. In practical terms, this may be understood as the two agents walking around the graph, thinking of edges as links of a unit length, and at time t they both use the same edge, entering from opposite ends, and thus meeting in (the middle of) the edge.

Definition 2.4. An agency is called a **feasible agency** if there is no crash between any pair of agents in neither an edge nor a node. In practical terms, this may be understood as a set of agents moving along the unit-length edges of the graph so that they avoid crashing into each other, but each of them manages to visit every node at least once, before finally arriving at their respective nodes of origin.

Given the definition of a feasible agency, we would like to set up a measure the "strength" of an agency. There are two options: we may try to maximize the number of agents, or we may try to minimize the time horizon. Anyway, for any G, k, T there is always the question: is there a feasible agency in graph G with k agents and time horizon T?

Problem 1. Given G, k, T, decide whether there is a feasible agency in G of k agents under time horizon T?

In Problem 1 if the answer is "yes" for a given G, k, T, then – because parking is allowed – the answer would be "yes" for any other instance G, k', T' when $k' \leq k$

and $T' \geq T$. For a given k we may want to determine the smallest T that admits a feasible agency, and for a given T we may want to determine the largest k that admits a feasible agency. Though these problems are hard to solve in general, we will look for some other ways how to measure the strength of an agency, thus we introduce the following parameters.

Definition 2.5. For an agency as above, let $\alpha_1 := n/k$, let $\alpha_2 := T/n$. The **strength** of the agency is given by $\alpha := \max\{\alpha_1, \alpha_2\}$.

Notice that for any connected graph there is an agency of $k \geq 1$ and $T \leq 2n - 2$. Also notice that $k \leq n$, and $T \geq n$, for any agency. A more interesting connection is with the Hamiltonian cycle problem.

Claim 2.6. *For a given graph there is an agency with $\alpha = 1$ if and only if there is a Hamiltonian cycle.*

Proof of Claim 2.6. $\alpha = 1$ implies $\alpha_2 = 1$, and thus $T = n$. The walk $a_1(0), a_1(0),$ $\cdots, a_1(T)$ is a Hamiltonian cycle. On the other hand, if there is a Hamiltonian cycle $v(i)$, then for the first agent we can set up a tour $a_1(t) := v(i)$, and all the other agents may follow this lead at a delay of $1, 2, \cdots n - 1$ time units. This makes a feasible agency.

Claim 2.7. *For any graph and any agency we have $\sqrt{T/k} \leq \alpha \leq T/k$.*

Proof of Claim 2.7. The claim follows from $\alpha_1 \alpha_2 = T/k$, and $1 \leq \alpha_1, \alpha_2 \leq \alpha$.

The main objective of this paper is to provide some nontrivial lower and upper bounds on the strength of a feasible agency.

3 Trees

Theorem 3.1. *If G is a tree, then for any feasible agency we have $T/k \geq 4$.*

Claim 3.2. *In any tree G with $n \geq 3$ there is an edge uv such that in $G - uv$ the component that contains v is a star that is centered at v.*

(A star is a $K_{1,m}$, where $m \geq 1$, and the node that corresponds to the color class of a single node is called its center. For ease of discussion, K_2 is called a star, and either of its nodes may be called its center.)

Proof of Claim 3.2. To prove this claim, consider a longest path P in the tree, and let u, v, z be the last three nodes in this path. Edge uv is a node as required: the component of $G - uv$ containing v does have another node, namely z, connected to v. If some other node z' in this component were to be not connected to v, then a path P' ending in z' would be longer than P, a contradiction. This proves the claim.

Proof of Theorem 3.1. We consider an edge uv obtained from Claim 3.2, and let us denote the star by $v, z_1, z_2, \cdots, z_m$, with v its center. Let $U := \{u, v, z_1, \cdots, z_m\}$. We distinguish three cases, depending on the value of m: $m = 1$, $m = 2$, and $m \geq 3$.

Figure 1: Case $m \geq 3$.
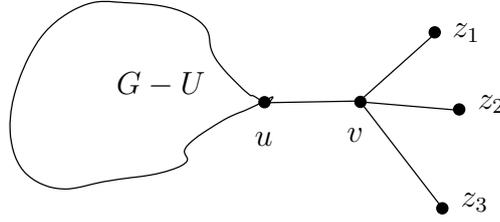
Figure 2: Case $m = 1$.
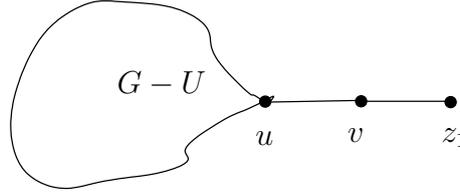
In case $m \geq 3$, the degree of node $deg_G(v) = m + 1 \geq 4$. Every agent needs to return at least $deg_G(v)$ number of times to v, and because they must not crash, $k \, deg_G(v) \geq T$. This implies $T/k \geq m + 1 \geq 4$.

In case $m = 1$, we are mainly concerned with node z_1. Every agent must visit z_1 at least once. Now two different agents visiting node z_1, say at times t, t', respectively, that is, $a_i(t) = a_j(t') = z_1$. We claim that $|t - t'| \geq 5$. To prove this, it is quite easy to see that if $|t - t'| \leq 4$, then agents i, j would crash at a time between t, t' in a node or edge in $G[\{u, v, z_1\}]$, a contradiction. Thus $|t - t'| \geq 5$, and actually, this still is true if we were to repeat the walks of the agents after time T, modulo T. This implies that $T \geq 5k$.

In case $m = 2$ we need to be somewhat more careful to prove $T/k \geq 4$. Every agent must visit z_1 and z_2 at least once, and for each i we consider the time interval(s) that agent i spends in the nodes $U := \{u, v, z_1, z_2\}$ while visiting at least one of the nodes z_1 or z_2. This means that for every agent i we pick one or two intervals so that they include one visit to v_1 and one visit to v_2. For an agent i there will be one or two corresponding intervals, say $I = [t^1, t^2]$. We consider time intervals modulo T, or "cyclically". Of course I will start and end in a time unit when agent i is occupying node u. Actually, we may assume that these time units satisfy the following:

- $a_i(t^1) = a_i(t^2) = u$, and

Figure 3: Case $m = 2$.
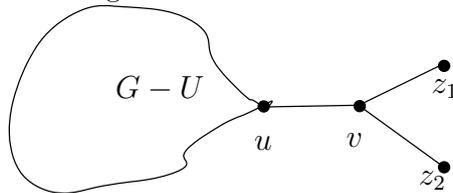
- $a_i(s) \in \{z_1, z_2\}$ for some $s \in [t^1, t^2]$

By removing parking in u from the beginning and end of the time interval (t^1, t^2), we may also assume that

- $a_i(t^1 + 1) = a_i(t^1 - 1) = v$

Definition 3.3. An interval that satisfies these 3 properties for agent i is called a U-interval for agent i.

Basically, a U-interval is the time interval that corresponds to a segment of that walk of agent i that looks like $u, v, \cdots, z_1, \cdots, v, u$ (or $u, v, \cdots, z_2, \cdots, v, u$) such that this segment of the walk stays within node subset U. The definition requires that during this time interval agent i starts with u, v, ends with v, u, and it touches z_1 or z_2 at least once, while staying within U. Of course there may be time intervals when an agent visits u, or even v, but without touching z_1, z_2 – we just don!t call such intervals U-intervals.

Every agent i must visit z_1 and z_2 at least once, and thus we can determine 1 or 2 U-intervals for agent i such that

- if there is one U-interval for agent i, than both z_1 and z_2 are visited by agent i during this U-interval I, or

- if there are two U-intervals for agent i, than z_1 is visited by agent i during only one of these U-intervals, and z_2 is visited by agent i during only the other U-interval. The two U-intervals for agent i are disjoint.

Let $\mathcal{I}_i$ denote the set of U-intervals for agent i, and let let $\mathcal{I} := \bigcup_i \mathcal{I}_i$ denote the set of all U-intervals for all agents. The most important property of U-intervals is that no 3 of them overlap at the same time unit, or, in other words, $\mathcal{I}$ is a 2-packing.

Claim 3.4. *For any $0 \le t \le T$, there are at most two intervals of $\mathcal{I}$ that contain t, that is, $|\{I : I \in \mathcal{I}, t \in I\}| \le 2$.*

To prove this, suppose there are three different agents for which the corresponding intervals contain t. This means that when the last of these agents enters $U := \{u, v, z_1, z_2\}$, the two others are already in U. Say the last to enter U is agent i, and the other two agents are i', i''. Say t^1 is the time when i enters U. Agent i at time $t^1 + 1$ moves to v, which implies that agent i', i'' occupy nodes z_1, z_2. By our assumption about these time intervals, at some time agent i while staying inside U, will to move to z_1 or z_2 – but the way is blocked by agents i', i''. A crash would be inevitable, and this contradiction proves the Claim.

The size of a time interval I is measured by how many time units it has, that is, $|I \cap \mathbb{Z}|$. For short we abbreviate this as $|I| := |I \cap \mathbb{Z}|$. We claim that for any agent i, if there is one U-interval for agent i, then that time interval has at least 7 time units; and if there are two intervals for agent i, then both of these time intervals contain at least 5 time units.

Formally,

Claim 3.5. *For any agent i,*

1. if $\mathcal{I}_i = \{I\}$, then $|I| \geq 7$, and

2. if $\mathcal{I}_i = \{I, J\}$, then $|I|, |I| \geq 5$, and

To prove this, not that the shortest walks to visit z_1 and satisfy the properties required for a U-interval is u, v, z_1, v, u, which requires 5 time units. And the shortest walks to visit z_1 and z_2 and satisfy the properties required for a U-interval is u, v, z_1, v, z_2, v, u, which requires 7 time units. This proves Claim 3.5.

By Claim 3.4 the set of intervals is a 2-packing, and this $2T \geq \sum_{I \in \mathcal{I}} |I|$. By Claim 3.5 we get that $\sum_{I \in \mathcal{I}} |I| \geq 7k$. This implies that $T/k \geq 3.5$. But our goal is to improve this bound to 4 from 3.5, so we need to look at these intervals more carefully. We have to look carefully especially at agents i for which there is only one interval, and that interval contains 7 time units.

The idea is to count tokens associated with time units. For every time unit $0 \leq t \leq T - 1$ we create 2 tokens, this is $2T$ tokens in total. We would

- award one of these tokens to an agent i such that $t \in \bigcup \mathcal{I}_i$

Because of Claim 3.4, we are not running out of tokens. By Claim 3.5, each agent has received at least 7 tokens, proving that $2T \geq 7k$. This comes just short of proving $T/k \geq 4$, so we need to be a bit more careful with the assignment of tokens. To improve on this bound, we adjust the placement of tokens as follows:

- If only one token is taken from time unit t by agent i, but the second token is up for grabs, then we award the second token to agent i, too. We do this for all time units t like this.

- If $I \in \mathcal{I}_i, J \in \mathcal{I}_j$, such that $I \subseteq J$ and $|I| = 7$, then agent j gives one of its tokens to agent i. We do this between all pairs I, J like this.

Claim 3.6. *All agents have at least 8 tokens.*

To prove this claim, we first show the following property of the intervals. Essentially, this claims that any U-interval I of 7 time units either has a time unit which overlaps with no other U-interval of any other agent, or there a U-interval of another agent contains I as a subset.

Claim 3.7. *If $\mathcal{I}_i = \{I\}$ with $|I| = 7$, then exactly one of the following assertions must hold:*

1. There is a time unit $t \in I - \bigcup_{J \in \mathcal{I}-I} J$.

2. There is another interval $J \in \mathcal{I} - I$ such that $I \subseteq J$.

In other words, Claim 3.7 states that for every agent with a single visit to U that takes the minimum 7 time units, there is either a time unit when i is the only agent on a visit to U, or there must be another agent that will on a visit in U through these whole 7 time units. Actually, this means that this other agent needs an interval of length at least 9 to visit U: at least one time unit before and at least one after i's visit.

Proof of Claim 3.7.   The only way for this claim to be false would be if there were (at least) two other intervals, say J_1 and J_2, which overlap with I, and so that there is no gap between J_1, J_2. Say J_1 corresponds to agent 1, J_2 corresponds to agent 2. Let us denote these U-intervals by $I = [a, a + 6]$, $J_1 = [b, c]$ and $J_2 = [c + 1, d]$, and thus by our assumption $a \le c < c + 1 \le a + 6$. This makes the choice of $c = a, a + 1, a + 2, a + 3, a + 4, a + 5$ possible. Note that by the definition of a U-interval, agents 1 and 2 occupy nodes v, u, u, v for time units $c - 1, c, c + 1, c + 2$. Also by the definition of a U-interval, agent i occupies nodes $u, v, \star, \star, \star, v, u$ at time units $a, a + 1, a + 2, a + 3, a + 4, a + 5, a + 6$. In all 6 possible cases of c, a crash happens in node u or v, or edge uv. This contradiction proves Claim 3.7.

Proof of Claim 3.6.   An agent i with two U-intervals has at least 10 tokens, because in both of its U-intervals it received at least 5 tokens. Say one of its U-intervals is J. If the agent has had to give one of them for another agent by our rule above for some $I \subseteq J$, then it has kept the 6 other tokens from the time units in I. In this case agent i has at least 6 tokens from U-interval J. Otherwise agent i has at least 5 tokens from U-interval J. So this makes at least 10 tokens for this agent, considering both of its U-intervals.

Consider an agent i with one U-interval J such that $|J| \ge 8$. If i has not had to give away any of its tokens then we are done. So now assume that agent i gave up one of its tokens because of another agent with a U-interval $I \subseteq J, |I| = 7$. In this case J must have at least one time unit before I, and at least one time unit after I, making it $|J| \ge 9$. So agent i has received at least 9 tokens, and gave away only one of them. If J overlaps with multiple other U-intervals $I, |I| = 7$ in this way, then it will keep at least 6 tokens from each of them, this way agent i will have at least $6 + 6 = 12$ tokens in the end. Considering all cases, agent i will have at least 8 tokens in the end.

Finally, consider an agent i with one U-interval I such that $|I| = 7$. This agent receives 7 tokens, and by Claim 3.7 it gets at least one more token for a time unit t where agent i is alone, or one token from another agent that is responsible for a U-interval J such that $I \subseteq I$. In both cases, this agent will hold at least 7 tokens in the end. This proves Claim 3.6.

Initially there are $2T$ tokens, and in the end we have accounted for at least $8k$ of them. This proves $T/k \ge 4$, and thus Theorem 3.1.

By Claim 2.7 this also implies a lower bound on the strength of a feasible agency that we might get in a tree.

Corollary 3.8. *If G is a tree, then for any agency we have $\alpha \ge 2$.*

The following example shows that the bound $T/k \ge 4$ given in Theorem 3.1 is tight,
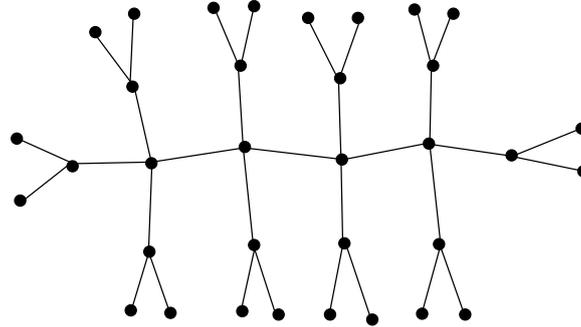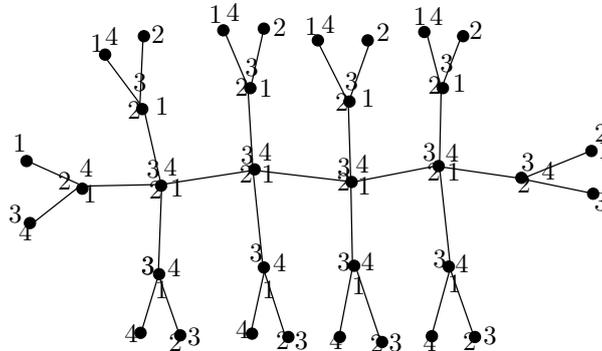
Figure 4: Example 1. Graph G.

Figure 5: Example 1. Annotated by time units modulo 4.

for certain trees. This example, though, is not tight for the bound given in Corollary 3.8 – we conjecture that that bound could be improved.

Example 1.

We define a tree $G = (V, E)$ as follows. Consider a tree that has $3r + 2$ (where r is some positive integer) nodes such that r of them have degree 4 and $2r + 2$ of them have degree 1. Let F denote this set of nodes of degree 4, and let L denote this set of $2r + 2$ nodes of degree 1. We add another $4r + 4$ nodes so that every node in L gets 2 new neighbors: say if $v \in L$, then we add nodes v^1 and v^2 joined by an edge each to node v. Let us say M denotes the set of new nodes. This defines a tree $G = (V, E)$ with $V = F \cup L \cup M$, having $7r + 6$ nodes and $7r + 5$ edges.

Consider an arbitrary closed walk of G that visits every nodes at least once, traverses every edge exactly twice, and returns to its origin in $14r + 10$ time units. We modify this walk so that for every node $v \in L$, we add parking in v_1 for one time unit – a change that provides a walk that spends 2 time units in v_1. This change creates a walk of time horizon $T = (14r + 10) + (2r + 2) = 16r + 12$. Let us denote this walk by $v(t)$. For agent 1, we define $a_1(t) := w(t)$. For all other agetns, we define their walks a_i by repeating w at a delay of $4(i - 1)$, repeated cyclically, that is $a_i(t) := w(t - 4(i - 1))$ where time is measured modulo T, i.e. cyclically. This defines a set of $T/4 = 4r + 3$ agents.

It is relatively straightforward to verify that this agency is feasible, thus proving the following claim. (For nodes $v \in L$ it can be verified manually that there is no crash in nodes or edges in $G[\{v, v^1, v^2\}]$. To see that there is no crash on any other node we need to observe that for any edge e adjacent with at least one node in F, any agent spends 3 modulo 4 time units in either component of $G - e$ – this implies that there is no crash in nodes of F or edges adjacent with a node in F.)

Claim 3.9. *There are trees of arbitrarily large size with a feasible agency such that* $T/k = 4$ *(with* $T = \frac{16}{7}n - \frac{12}{7}$ *and* $k = \frac{4}{7}n - \frac{3}{7}$*).*

4 Trees with $T = 2n - 2$

When graph G is a tree, then any traveling salesman tour has time horizon at least $2m - 2$, and thus of course, any agency has $T \geq 2n - 2$. A tour with $T = 2n - 2$ is shortest possible, meaning that every edge of the tree is traversed only twice. In this section we restrict ourselves to shortest tours, that is $T = 2n - 2$, and our main result shows an upper bound on the number of agents in this case.

Theorem 4.1. *If G is a tree, then for a feasible agency with $T = 2n - 2$ we have* $T/k \geq 5$.

Proof of Theorem 4.1.
Consider an edge uv as guaranteed by Claim 3.2, and let us denote the star by $v, z_1, z_2, \cdots, z_m$. We distinguish three cases, depending on the value of m, $m = 1$, $m = 2$, and $m \geq 3$. Let $U := \{v, z_1, z_2, \cdots, z_m\}$. (Let us remark that we consider the tours to be repeating cyclically modulo T, that is, we consider time units "modulo $T = 2n - 2$".) As in the proof of Theorem 3.1, every agent has a U-interval: for agent i let the shortest time interval that contains the times when agent i visits some z_j, and starts and ends with a time unit when agent i is occupying node u is called a U-interval. Because the tour of any agent i is shortest, there is only one U-interval, and it has $2m + 3$ time units. Say for agent i, the U-interval is equal to $[t(i), t(i) + 2m + 2]$. The walk of agent i during its U-interval looks like $u, v, \cdots, v, u$, and contains all leaves z_j in U.

Case $m = 1$. In this case it takes $2m + 3 = 5$ time units for any agent i to complete its U-interval, which is $[t(i), t(i) + 4]$. The segment of its tour during this U-interval must be (u, v, z_1, v, u). It is plain and easy to see that the U-intervals of no two agents may overlap, to avoid a crash in U. Thus at least 5 time units need to be spent before another agent may start its own U-interval.

Claim 4.2. *If $i \neq j$ then $[t(i), t(i) + 4] \cap [t(j), t(j) + 4] = \emptyset$.*

Now, for all time units between 1 and T we create one token. An agent takes all tokens corresponding to its U-interval. Because of Claim 4.2 $5k$ tokens are distributed among the agents. This proves the desired bound $T \geq 5k$ in this case.

Case $m = 3$. In this case it takes $2m + 3 = 9$ time units for agent i tocomplete its U-interval, which is $[t(i), t(i) + 8]$. The segment of its tour during this U-interval must be $(u, v, z_a, v, z_b, v, z_c, v, u)$, where a, b, c is a permutation of $1, 2, 3$. To avoid a crash in node v, if two agents i, j have overlapping U-intervals, then the intersection of their U-interval must be even.

Claim 4.3. *If* $i \neq j$, *then* $[t(i), t(i) + 4] \cap [t(j), t(j) + 4]$ *has even cardinality.*

This claim implies that any given time unit t may be part of at most two U-intervals corresponding to any agents, or we may say the U-intervals are a 2-packing. (Considering just this, we obtain a bound of $9k \leq 2T$.) Actually, this also implies the following:

Claim 4.4. *For any agent i there is a time unit t in its U-interval (i.e. $t \in [t(i), t(i) + 8]$) such that t is not part of any other agent's U-interval.*

Otherwise there would have to be two other agents, i' and i'' such that $[t(i), t(i) + 8] \subseteq [t(i'), t(i') + 8] \cup [t(i''), t(i'') + 8]$. Because a U-interval has an odd number of time units, $t(i')$ and $t(i'')$ need to have different parity. This contradicts Claim 4.3, and thus proves Claim 4.4.

We now use Claims 4.3 and 4.4 to finish the proof in this case. We create 2 tokens for every time unit $1 \leq t \leq T$. The token corresponding to time unit t is awarded to agent i if $t \in [t(i), t(i) + 8]$. Because the U-intervals are a 2-packing, the tokens created are enough for this step. Now if for t only one token is taken, say by agent i, then the other token of t is also awarded to agent i. Because a U-interval has 9 time units, every agent will have 9 tokens are taken initially. Because of Claim 4.4, every agent will have at least one extra token, thus every agent will receive at least 10 tokens in total. This implies $2T \geq 10k$, proving the bound in this case.

Case $m > 3$ is solved the same way.

Case $m = 2$. In this case it takes $2m + 3 = 7$ time units for agent i tocomplete its U-interval, which is $[t(i), t(i) + 6]$. The segment of its tour during this U-interval must be $(u, v, z_a, v, z_b, v, u)$, where a, b is a permutation of $1, 2$. To avoid a crash in node v, if two agents i, j have overlapping U-intervals, then the intersection of their U-interval must be even.

Claim 4.5. *If* $i \neq j$, *then* $[t(i), t(i) + 6] \cap [t(j), t(j) + 6]$ *has even cardinality.*

Moreover, to avoid a crash on edge uv, the overlap $[t(i), t(i) + 6] \cap [t(j), t(j) + 6]$ may not have cardinality 2. To avoid a crash in edges vz_1, vz_2, the overlap $[t(i), t(i) + 6] \cap [t(j), t(j) + 6]$ may not have cardinality 6. This implies that when there is an overlap, it must have cardinality exactly 4. Because the U-intervals are a 2-packing, this implies the following:

Claim 4.6. *For any agent i there is are 3 different time units t in its U-interval (i.e. $t \in [t(i), t(i) + 8]$) such that t is not part of any other agent's U-interval.*

Figure 6: Example 2.
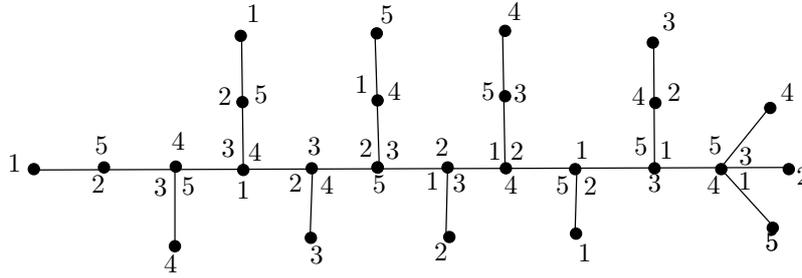
We now use Claims 4.5 and 4.6 to finish the proof in this case. We create 2 tokens for every time unit $1 \leq t \leq T$. The token corresponding to time unit t is awarded to agent i if $t \in [t(i), t(i) + 8]$. Because the U-intervals are a 2-packing, the tokens created are enough for this step. Now if for t only one token is taken, say by agent i, then the other token of t is also awarded to agent i. Because a U-interval has 7 time units, every agent will have 7 tokens are taken initially. Because of Claim 4.4, every agent will have at least three extra token, thus every agent will receive at least 10 tokens in total. This implies $2T \geq 10k$, proving the bound in this case.

This completes the proof of Theorem 4.1.

Corollary 4.7. *If G is a tree, then for any agency with $T = 2n - 2$ we have $\alpha \geq \sqrt{5}$.*

Example 2. The following example shows that the bound $T/k \geq 5$ given in Theorem 4.1 is tight, for certain trees. This example, though, is not tight for the bound given in Corollary 4.7 – we conjecture that that bound could be improved. We construct a graph as shown in the figure by using a path of even length $2q$, and adding a 1-path to its $2i+1$'th node, and adding a 2-path to its $2i+2$'th node. We add another 2-path to its first node, and a 3-star at its last node. This creates a graph of $n = 5q + 6$ nodes, and $5q + 5$ edges. We create a walk of length $10q + 10$ so that it first moves from left to right along the path, and also enters and leavse the 1-paths, then the walk enters and leavs the 3-star and all of its leaves, and then on the way back from right to left, the walk enters and leaves the 2-paths. This walk is indicated with the time units modulo 5 in the figure. Agent 1 would follow this tour, and agent i would follow this tour at a delay of $5(i - 1)$, cyclically. So that in time unit 1, the nodes indicated with "1" are occupied, and then after each step, the nodes indicated with "j" are occupied at time units congruent with j modulo 5. It is quite easy to verify that this creates a feasible agency without parking with time horizon $T = 10q + 10$ and number of agents $k = 2q + 2 = T/5$.

Theorem 4.8. *There are arbitrarily large trees for which there is a feasible agency without parking such that $T = 2n - 2$ and $k = \frac{1}{5}T$.*

5   3-edge-connected 3-regular graphs

Theorem 5.1. *If $G = (V, E)$ is a 3-regular 3-edge-connected graph, then there is an agency with $T/k = 4$, $T = 2n$, $k = n/2$ and $\alpha = \alpha_1 = \alpha_2 = 2$.*

By Petersen's theorem we know that graph G contains a perfect matching, or equivalently, by taking the complement of a perfect matching, we also know that G contains a 2-factor (a 2-regular subgraph spanning all the nodes). There are ways of formulating a result that is stronger than Petersen's Theorem, and one of them is a recent result of Boyd, Iwata and Takazawa [4], claiming the existence of a 2-factor of a special property.

Theorem 5.2 (Boyd, Iwata, Takazawa[4])**.** *If $G = (V, E)$ is a 3-regular bridgeless graph, then there is a 2-factor covering every 3-cut and every 4-cut.*

Proof of Theorem 5.1.   Because of Theorem 5.2, there is a perfect matching M in G such that $G - M = (V, E - M)$ consists of cycles of length at least 5. Let $\mathcal{C} = \{C_1, C_2, \cdots, C_s\}$ denote the set of these cycles. Let $N \subseteq M$ be a spanning tree in $G/\mathcal{C}$. Thus $F := N \cup \bigcup \mathcal{C}$ is connected, and the only cycles in F are those of $\mathcal{C}$. A tour $a(t)$ of subgraph (V, F) is obtained so that edges in N are traversed twice, and edges in $\bigcup \mathcal{C}$ are traversed once (an arbitrary orientation of the cycles is chosen). The time horizon for this tour is $T = |V - V(N)| + 4|N|$, because it uses both nodes of an edge in N twice.

We modify this tour $a(t)$ just by adding parking times to nodes in $V - V(N)$. We will come up with a "parking vector" $p : V - V(N) \to \{1, 2, 3, 4\}$, and we interpret this with the intention of making an agent stay in a node $v \in V - V(N)$ for $p(v)$ time units. This means the agent will park for $p(v) - 1$ time units. This defines a walk $a^p(t)$, which is obtained from $a(t)$ by repeating v by $p(v)$ copies. The time horizon of this walk will be

$$T = 4|N| + \sum_{v \in V - V(N)} p(v).$$

Though this defines the tour of an agent for an arbitrary parking vector p, we need p to satisfy a certain property in order to define a feasible agency.

Definition 5.3. We say that the parking vector p is a **4-cyclic parking vector** if for every $C \in \mathcal{C}$, the sum

$$\sum_{v \in V(C) - V(N)} p(v)$$

is a multiple of 4.

The point of this definition is that a 4-cyclic parking vector may be used to define an agency. Let $k := T/4$, and for $i = 1, 2, \cdots, k$, let a_i^p be the tour obtained from a^p at a delay of $4(i - 1)$. Thus all agents will have a traveling salesman tour, and it is also quite straightforward that this defines a feasible agency.

Claim 5.4. *For any 4-cyclic parking vector p, the set of tours a_i^p defined above is a feasible agency.*

To prove this claim, consider any edge $uv \in N$, which of course is a cut edge in (V, F), say it separates the node set into parts $u \in V_u$ and $v \in V_v$. Also let us suppose that t is the time unit when our agent comes to u to move to v in the following time unit, that is, $a^p(t) = u$ and $a^p(t + 1) = v$. Because the number of time units our agent spends in $V_v - v$ is a mulitple of 4, the time t' when it returns to v (i.e. $t' > t, a^p(t') = v$) is congruent with $t + 2$ modulo 4, that is, $t' \equiv t + 2 \, (mod \, 4)$. This implies that there is no crash in nodes u or v or on edge uv. Thus there is no crash on nodes or edges of matching N, but this also implies that there is no crash on nodes $V - V(N)$, because $p(v) \leq 4$, and thus there is no time for agent a^p_{i+1} to catch up with agent a^p_i. This proves Claim 5.4

Of course, $\overline{p}(v) := 4$ for all v is a 4-cyclic parking vector, and thus it defines a feasible agency. Because of 5.2, $|\mathcal{C}| \leq \lfloor n/5 \rfloor$, and thus $|N| \leq \lfloor n/5 \rfloor - 1$. Then for $\overline{P}$ we get a feasible agency with

$$T = 4|N| + 4|V - V(N)| = 2|V| + 2|V - V(N)| \geq 2n + 2(n - 2(\lfloor n/5 \rfloor - 1)) \geq \frac{16}{5}n.$$

Next we construct a different agency a^p from a different 4-cyclic parking vector p, trying to make p as small as possible. For a cycle $C_i \in \mathcal{C}$, the minimum number of time units spent in the nodes of $V(C) - V(N)$ just depends on $|V(C) - V(N)|$: we would want to have as many ones as possible, and then with the last remaining element, we fix the remainder mondulo 4. Here we need the function $r(s) := 4 \lceil \frac{1}{4}s \rceil$, thus the minimum number number of time units spent in the nodes of $V(C) - V(N)$ is equal to $r(|V(C) - V(N)|)$. We get the following claim.

Claim 5.5. *There is 4-cyclic parking vector p such that for agency a^p the time horizon is equal to $T = 4|N| + \sum_{C \in \mathcal{C}} r(|V(C) - V(N)|)$.*

We will be able to prove a bound on T by partitioning $\mathcal{C} = \mathcal{C}_0 \cup \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3$ such that

- $|V(C) - V(N)| = 0$ for cycles $C \in \mathcal{C}_0$,

- $|V(C) - V(N)| = 1$ for cycles $C \in \mathcal{C}_1$,

- $|V(C) - V(N)| = 2$ for cycles $C \in \mathcal{C}_2$,

- $|V(C) - V(N)| \geq 3$ for cycles $C \in \mathcal{C}_3$.

By the definition of function $r(\cdot)$ we get that

- $r(|V(C) - V(N)|) = 0 = 2|V(C) - V(N)|$ for $C \in \mathcal{C}_0$,

- $r(|V(C) - V(N)|) = 4 \leq 2|V(C) - V(N)| + 2$ for $C \in \mathcal{C}_1$,

- $r(|V(C) - V(N)|) = 4 = 2|V(C) - V(N)|$ for $C \in \mathcal{C}_2$,

- $r(|V(C) - V(N)|) \leq 2|V(C) - V(N)| - 2$ for $C \in \mathcal{C}_3$.

Consider the tree $F := (V, N)/\mathcal{C}$ that we get from edges in N by shrinking the cycles of $\mathcal{C}$. All cycles in $\mathcal{C}_1$ correspond to a node of F of degree at least 3. All leaves of F correspond to a cycle in $\mathcal{C}_3$. For any tree, the number of leaves is larger than the number of nodes of degree at least 3, which for tree F implies that $|\mathcal{C}_3| \geq |\mathcal{C}_1|$. We put all this together in the following calculation:

$$T = 4|N| + \sum_{C \in \mathcal{C}} r(|V(C) - V(N)|) =$$

$$= 2|V(N)| + \sum_{C \in \mathcal{C}_0} r(|V(C) - V(N)|) + \sum_{C \in \mathcal{C}_1} r(|V(C) - V(N)|) +$$

$$+ \sum_{C \in \mathcal{C}_2} r(|V(C) - V(N)|) + \sum_{C \in \mathcal{C}_3} r(|V(C) - V(N)|) \leq$$

$$\leq 2|V(N)| + \sum_{C \in \mathcal{C}_0} 2|V(C) - V(N)| + \sum_{C \in \mathcal{C}_1} (2|V(C) - V(N)| + 2) +$$

$$+ \sum_{C \in \mathcal{C}_2} 2|V(C) - V(N)| + \sum_{C \in \mathcal{C}_3} (2|V(C) - V(N)| - 2) =$$

$$= 2|V(N)| + \sum_{C \in \mathcal{C}} 2|V(C) - V(N)| + 2|\mathcal{C}_1| - 2|\mathcal{C}_3| \leq 2|V(N)| + \sum_{C \in \mathcal{C}} 2|V(C) - V(N)| = 2|V| = 2n.$$

Thus we have constructed two different 4-cyclic parking vectors: $\overline{p} \equiv 4$ with $T \geq \frac{16}{5}n$, and this latter 4-cyclic parking vector p with $T \leq 2n$. We can add any multiple of 4 time units to p while maintaining its 4-cyclic property, to obtain 4-cyclic parking vectors p' with $p \leq p' \leq \overline{p}$, and thus we can find one that provides a feasible agency with $T = 2n$. This proves Theorem 5.1.

6   Optimization problems

Optimization problems with regard to $k, T, \alpha, \alpha_1, \alpha_2$ seem to be rather hard to solve exactly – recall that by Theorem 2.6, minimizing α is NP-hard. There are a few problems, though, that may be solved exactly, namely the following:

Problem 2. For a given graph $G = (V, E)$, find a feasible agency (with parking) of $k = n = |V|$ agents and any time horizon T, or determine that there is no such agency.

Problem 3. For a given graph $G = (V, E)$, find a feasible agency without parking of $k = n = |V|$ agents and any time horizon T, or determine that there is no such agency.

We are going to prove that both of these problems can be solved in polynomial time.

Theorem 6.1. *There is a polynomial time algorithm to solve Problem 2.*

To prove this, note that all nodes are occupied by an agent at any time. Thus the step between time unit t and $t+1$ corresponds with a set of node-disjoint cycles in G. Agents in any of these cycles will move to their respective neighbors, cyclically, while agents in a node not covered by any of these cycles are just parking there during this time unit. This suggests that the following condition may describe the existence of a a feasible agency $k = n$ as required in Problem 2:

Claim 6.2. *There is a feasible agency with parking of $k = n$ agents if and only if G is 2-edge-connected.*

To show this claim, first note that if there is a cut edge e, then no agent may traverse e because e is not part of any cycle. Thus agent to one side of $G - e$ will never be able to get to the other side. To prove the other way around, suppose G is 2-edge-connected. It is well-known that in this case, every edge is contained in a cycle. Now consider a spanning tree of G, and agent 1: we make agent 1 move along edges of the spanning tree, while completing each step of agent 1 using a cycle to define steps for all other agents. We repeat this for edges of the spanning tree until agent 1 has visited all the nodes. Then we repeat this for all other agents, one by one. In the end we need to repeat all these steps backwards to get all the agents back to their original position. The time horizon for this agency will be $T \leq 2n(2n - 3)$. This proves Theorem 6.1.

Conjecture 6.3. *There is a polynomial time algorithm to solve Problem 3.*

To prove this, note that all nodes are occupied by an agent at any time. Thus the step between time unit t and $t + 1$ corresponds with a 2-factor in G. Recall that a 2-factor is a spanning subgraph such that all nodes have degree exactly 2. A 2-factor consists of cycles, and all the nodes are covered by exactly one cycle. This suggests that the following condition may describe the existence of a a feasible agency $k = n$ as required in Problem 3:

Claim 6.4. *There is a feasible agency without parking of $k = n$ agents if and only if the edges of G which are contained in a 2-factor form a connected spanning subgraph.*

To show this claim, first let us assume that there is a partition $V = V_1 \cup V_2$ into two disjoint nonempty subsets such that edges between V_1, V_2 are contained in no 2-factor. In this case, an agent that starts out in V_1 will never leave V_1, which implies that there is no agency with the required properties. To prove the other way around, suppose that there is a spanning tree $F \subseteq E$ such that all edges in F are contained in a 2-factor. Then it is straightforward to construct an agency: we make agent 1 move along edges of F, while completing each step of agent 1 using a 2-factor to define steps for all other agents. We repeat this for edges of the spanning tree until agent 1 has visited all the nodes. Then we repeat this for all other agents, one by one. After $n(2n - 3)$ time units every agent has finished with visiting all the nodes. To obtain a feasible agency, we need to make sure all the agents will get back to their original

position, all at the same time (this is required by the definition of a feasible agency). Let us say it took $T_1 \leq n(2n-3)$ time units to finish the moves up to this point, and note that the current positions of the agents define a permutation of V, where node $v \in V$ is mapped to node $\phi(v) \in V$ iff the agent starting in v has arrived at $\phi(v)$ after T_1 time units. We just have to repeat ϕ a few times until all agents get back to their original positions, that is, we are looking for a positive integer q such that $\phi^q = $ id. Here the least common multiple of cycle length's of cycles in ϕ will do – the trivial upper bound on this is $q \leq n!$. This proves Claim 6.4 with an exponential bound on the time hoziron, $T = qT_1 \leq n(2n-3)n!$. Unfortunately this is not polynomial, and for this reason we fall short of proving Conjecture 6.3 – though we are able to determine a yes/no answer to the existence of such an agency. At this point it is unclear if the existence of an agency with very long, maybe exponential time horizon would actually also imply the existence of an agency with polynomial time horizon, thus leaving the proof of Conjecture 6.3 open.

Acknowledgments

References

[1] *Problem N.174*, Középiskolai Matematikai és Fizikai Lapok (1998), no. 4.

[2] Egon Balas, *The prize collecting traveling salesman problem*, Networks **19** (1989), no. 6, 621–636.

[3] James H. Bookbinder and Suresh P. Sethi, *The dynamic transportation problem: A survey*, Naval Research Logistics Quarterly **27**, no. 1, 65–87.

[4] Sylvia C. Boyd, Satoru Iwata, and Kenjiro Takazawa, *Finding 2-factors closer to TSP tours in cubic graphs*, SIAM J. Discrete Math. **27** (2013), no. 2, 918–939.

[5] N. Christofides, A. Mingozzi, and P. Toth, *Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations*, Mathematical Programming **20** (1981), no. 1, 255–282.

[6] L. R. Ford and D. R. Fulkerson, *Constructing maximal dynamic flows from static flows*, Oper. Res. **6** (1958), no. 3, 419–433.

[7] Bruce Hoppe and Éva Tardos, *The quickest transshipment problem*, Mathematics of Operations Research **25** (2000), no. 1, 36–62.

[8] Balázs Kotnyek, *An annotated overview of dynamic network flows*, Tech. Report RR-4936, INRIA, Sep 2003.

[9] Gilbert Laporte and Yves Nobert, *A cutting planes algorithm for the m-salesmen problem*, Journal of the Operational Research Society **31** (1980), no. 11, 1017–1023.

[10] Martin Skutella, *An introduction to network flows over time*, pp. 451–482, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.