The School Bus Problem on Trees

Adrian Bock[1], Elyot Grant[2], Jochen Könemann[2], and Laura Sanità[1]

[1] EPFL, Lausanne, Switzerland
[2] University of Waterloo, Canada

Abstract. The School Bus Problem is an NP-hard vehicle routing problem in which the goal is to route buses that transport children to a school such that for each child, the distance travelled on the bus does not exceed the shortest distance from the child's home to the school by more than a given *regret* threshold. Subject to this constraint and bus capacity limit, the goal is to minimize the number of buses required.
In this paper, we give a polynomial time 4-approximation algorithm when the children and school are located at vertices of a fixed *tree*. As a byproduct of our analysis, we show that the integrality gap of the natural set-cover formulation for this problem is also bounded by 4. We also present a constant approximation for the variant where we have a fixed number of buses to use, and the goal is to minimize the maximum regret.

1 Introduction

Vehicle routing is an important and active topic in computer science and operations research. In the literature, the objective is typically to find a minimum-cost set of routes in a network that achieve a certain objective subject to a set of constraints. The constraints and cost are often related to the distance travelled, number of routes or vehicles used, coverage of the network by the routes, and so on. Problems of this kind are frequent and crucial in areas such as logistics, distribution systems, and public transportation (see, e.g., the survey by [13]).

In vehicle routing problems relevant to public transportation, a secondary objective often must be taken into account beyond minimizing operation cost: namely, it is crucial to design routes so as to optimize *customer satisfaction* in order to motivate customers to use the service. This requirement is essential in the so-called *School Bus Problem* (SBP)—the focus of this paper.

In the SBP, we must route buses that pick up children and bring them from their homes to a school. However, parents do not want their children to spend too much time on the bus relative to the time required to transport them to school by car along a shortest path. In fact, if the additional distance travelled by the bus exceeds a certain *regret* threshold, the parents would rather drive their children to school by themselves, which is unacceptable. Subject to this, the goal is to cover all of the children using a minimum number of buses.

Formally, we are given an undirected network $G(V, E)$ with distances on the edges $d : E \to \mathbb{Z}_+$, a node $s \in V$ representing the school, and a set $W \subseteq V$ representing the houses of children. Additionally, we are given a bus capacity

$C \in \mathbb{Z}_+$, and a regret bound $R \in \mathbb{Z}_+$. The aim is to construct a minimum cardinality set $\mathcal{P}$ of walks ending at s (bus routes) and assign each child $w \in W$ to be the responsibility of some bus $p(w) \in \mathcal{P}$ such that (i) for each walk $P \in \mathcal{P}$, the total number of children w with $p(w) = P$ is at most the capacity C; (ii) for every child the regret bound is respected, that is: $d^P(w, s) \leq d(w, s) + R$, where $d^P(w, s)$ is the distance from the child w to the school s on the walk $p(w)$, and $d(w, s)$ is the shortest distance from w to s in the graph G.

An additional variation of the problem is the "symmetric" version, in which we have a fixed number N of buses we can use to cover the children, and the goal is to minimize the maximum regret R. We call this variant the *School Bus Problem with Regret minimization* (SBP-R).

Like many vehicle routing problems, both SBP and SBP-R are strongly NP-hard, even on the simplest of graphs. To see this, consider a star centered at the school s with children located at all other vertices. If k of the edges are very long, then determining if k buses are sufficient with a regret bound R is precisely equivalent to solving the bin-packing decision problem with k bins and objects sized according to the distances from the remaining children to the school. It follows that bin-packing can be reduced to both SBP and SBP-R on stars.

Many variants of vehicle routing have been studied in the context of exact, approximate, and heuristic algorithms. For SBP and SBP-R, heuristical methods for practical applications have been examined (see the survey of [11]), but there is no literature concerning formal approximability and inapproximability results. Our goal is to advance the state of the art in this perspective.

To begin, it is easy to see that the SBP can be formulated as a set covering problem. With this observation, one can easily derive a logarithmic approximation as well as a logarithmic upper bound on the integrality gap of the natural set-cover formulation applied to the SBP (see Section 2 for more details). The SBP is closely related (but not equal) to the Distance Constrained Vehicle Routing Problem (DVRP), which is well known and widely studied in terms of approximation. The DVRP can also be approximated within a logarithmic factor in general graphs, but there is a better 2-approximation on *trees*, as shown by [10]. They also show that the set-cover formulation for the DVRP on trees has integrality gap of at most 20. A natural question is then whether or not the SBP also admits a constant approximation/integrality gap on such graphs. Unfortunately, a straightforward adaptation of their methods to the SBP does not work. Therefore, in order to develop improved approximation results for the SBP, we need to introduce some new ideas.

1.1 Our results

We first give a simple combinatorial 4-approximation for the SBP on *trees*. The algorithm splits the instance into pieces, each of which can each be well-approximated by greedily cutting an Euler tour into bus routes. Formally:

Theorem 1. *There exists a polynomial time 4-approximation for the School Bus Problem on trees. The approximation factor can be improved to 3 in the case of unlimited capacity.*

In contrast to the results given in [10] for DVRP, our algorithm for SBP immediately yields an integrality gap bound matching the approximation factor:

Theorem 2. *The integrality gap of the natural set-cover formulation of the SBP on trees is at most 4. In case of unlimited capacity, the gap is at most 3.*

Finally, we give a combinatorial 12.5-approximation for the SBP-R on trees. The algorithm is more involved, relying upon bipartite matching as a subroutine:

Theorem 3. *There exists a polynomial time* 12.5*-approximation algorithm for the SBP-R on trees in the case of unlimited capacity.*

1.2 Related work

There are an enormous number of results concerning vehicle routing problems; see the survey [13]. We briefly discuss work on the SBP and related problems.

The Capacitated Vehicle Routing Problem (CVRP) enforces a limit C on the number of visited locations in each route, and the goal is the minimization of the total length of all the routes. The paper [6] established a strong link to the underlying Travelling Salesman Problem (TSP) by giving an approximation algorithm that relies on the approximation algorithms for TSP. Depending on the capacity bound C, it is possible to obtain a PTAS in the euclidean plane for some special cases (if the capacity is either small [1], or very large [2]). If we restrict the input to trees, there is a 2-approximation [7]. Another constraint considered in literature is a bound D on the length of a vehicle tour, under the objective of minimizing the number of routes. This is the Distance Constrained Vehicle Routing Problem (DVRP). It was raised and studied for applications in [8] and [9]. Routing problems like the DVRP can be directly encoded as instances of Minimum Set Cover, and thus often admit logarithmic approximations. The authors of [10] give a careful analysis of the set cover integer programming formulation of the DVRP and bound its integrality gap by $\mathcal{O}(\log D)$ on general graphs and by $\mathcal{O}(1)$ on a tree. They also obtain a constant approximation for the DVRP on a tree and a $\mathcal{O}(\log D)$ approximation in general.

Many practical problems involving school buses have been studied, but primarily within the context of *heuristical methods* for real-life instances. We refer to [11] for a thorough survey of possible formulations and heuristical solution methods. Our notion of regret was first introduced as a vehicle routing objective in [12]. They considered a more general problem involving timing windows for customers and applied metaheuristics to produce solutions to real-life instances.

2 Preliminaries

We first observe that the capacity bound can be neglected for a slight loss in the approximation factor for the SBP. The proof, given in the appendix, is essentially identical to that of a similar result proven in [10] for the DVRP.

Lemma 1. *Given an α-approximation to the SBP with unlimited capacity for each bus, there is an $\alpha + 1$-approximation to the SBP that respects a capacity bound C on each bus.*

If P is a walk starting at some vertex v and ending at s, covering a subset S of nodes, then we say that P has regret R if a regret bound of R is respected for all children in S. The following useful fact (proved in the appendix) holds for both the SBP and the SBP-R:

Proposition 1. *For all nodes in S the regret bound R is respected if and only if it is respected for v.*

We next give a covering integer programming formulation of the SBP. Let $\mathcal{S}$ be the family of all feasible sets of C or fewer children that can be covered by a single walk ending at s having regret at most R. We introduce a varaible x_S for each $S \in \mathcal{S}$ and give the following formulation:

$$\begin{array}{lll} \min & \sum_{S \in \mathcal{S}} x_P & \textit{(IP)} \\ & \sum_{S: w \in S} x_S \geq 1 & \forall w \in W \\ & x_S \in \{0,1\} & \forall S \in \mathcal{S}. \end{array}$$

An $\mathcal{O}(\log |W|)$-approximation algorithm easily follows from adapting the greedy strategy for set cover. Such a greedy algorithm, applied to an SBP instance, repeatedly searches for a feasible walk ending at s that picks up the maximum number of uncovered children, doing do until every child is picked up. At each iteration, we could guess the starting point v^* (by trying all $|V|-1$ possibilities). Using Proposition 1, the resulting problem we are left with is to find a $v^* - s$ walk in G of length at most $d(v^*, s) + R$ visiting the maximum number of uncovered nodes in W. Such a problem is well known in the literature as the *Orienteering Problem*, and can be approximated within a constant [3, 4]. Following the method of [14], we may then obtain an $\mathcal{O}(\log C)$-approximation algorithm for the SBP and show that the integrality gap of (IP) is at most $\mathcal{O}(\log C)$ (see the appendix for a rigorous argument.) However, these logarithmic results may not be tight in general, and as we will show, are not tight for the SBP on *trees*.

In the remainder of this paper, we will focus on the infinite capacity version of the SBP on a tree T with root s. We denote by $P(u, v)$ and $d(u, v)$ the unique path from u to v in T, and its corresponding length. For a subset of edges F, we let $d(F) := \sum_{e \in F} d(e)$. We note that subtrees of T that contain no vertices in W will never be visited by a bus in any optimal solution, and thus we can assume without loss of generality that all leaves of T contain children. In such an instance, a feasible solution will simply cover all of T with bus routes and thus is still feasible if every node of T contains a child (assuming infinite capacities). We thus will assume, without loss of generality, that $W = V$.

3 A 4-approximation to the SBP on trees

We prove Theorem 1 by first giving a combinatorial 3-approximation for the SBP with unlimited capacity on graphs that are trees, and subsequently applying Lemma 1. Our algorithm is based on the following intuitive observations:

– When the input tree is very short (say, of height at most $\frac{R}{2}$ on an instance with regret R), then it is relatively easy to obtain a 2-approximation for the SBP by simply cutting an Euler tour of the tree into short pieces and assigning each piece to a bus.
– General trees can be partitioned into smaller pieces (subtrees) such that at least one bus is required for each piece, but each piece can be solved almost optimally via a similar Euler tour method.

We begin with some definitions. We call a set of vertices $\{a_1, \ldots, a_m\} \subseteq V$ *R-independent* if for all $a_i \neq a_j$, we have $d(a_i, \mathrm{lca}(a_i, a_j)) > \frac{R}{2}$, where $\mathrm{lca}(a_i, a_j)$ is the lowest common ancestor of the vertices a_i and a_j in T.

By iteratively marking the lowest leaf in T such that R-independence is maintained among marked leaves, we can obtain, in polynomial time, an inclusion-wise maximal R-independent set of leaves A such that all vertices in T are within a distance of $\frac{R}{2}$ from a path $P(s, a)$ for some $a \in A$. We shall call A a set of *anchors*. By construction, no two distinct anchors a_i and a_j can both be covered by a walk of regret at most R, immediately yielding the following lower bound:

Proposition 2. *The size $|A|$ of the set of anchors is a lower bound on the number of buses that is needed in any feasible solution.*

We now give a second useful lower bound. Let $Q := \bigcup_{a \in A} P(s, a)$. We call Q the *skeleton* of T, noting that Q is a subtree of T whose leaves are the anchors. Observe that all edges in the skeleton Q will automatically be covered if a bus visits each anchor. Since each anchor must be visited at least once, it suffices to only consider covering the anchors and the *non-skeletal edges* of T, i.e. the edges in $T \setminus Q$. The edges in $T \setminus Q$ form a collection of disjoint subtrees, each of which has height at most $\frac{R}{2}$. We call these *short subtrees*.

Suppose that a feasible walk starts at a vertex v in a short subtree $\mathcal{T}$. It will cover all the edges in $P(s, v)$, and may possibly cover some additional detour edges having total length at most $\frac{R}{2}$. Since $\mathcal{T}$ is a short subtree, the non-skeletal edges in $P(s, v)$ have total length at most $\frac{R}{2}$. It follows that:

Observation 1 *The set of non-skeletal edges covered by any feasible walk P must have total length at most R: at most $\frac{R}{2}$ in length along the path from its starting vertex to the root, and at most $\frac{R}{2}$ length in edges covered by detours.*

From this, we can observe the following lower bound on the number of buses:

Proposition 3. *The number $\frac{1}{R} \sum_{e \in T \setminus Q} d(e)$ is a lower bound on the number of buses that is needed in any feasible solution.*

We build our 3-approximation from these two lower bounds by partitioning the edges of T into a family of subtrees each containing a single anchor, and approximating the optimal solution well on each of these subtrees. For anchors $A = \{a_1, \ldots, a_m\}$, we define associated paths of edges $\{P_1, \ldots, P_m\}$ as follows:

– $P_1 = P(s, a_1)$, and

– $P_i = P(s, a_i) \setminus \left(\bigcup_{j=1}^{i-1} \{P_j\} \right)$ for $2 \leq i \leq m$.

The edges in $\{P_1, \ldots, P_m\}$ form a partition of the skeleton Q into paths, each of which starts at a different anchor.

We then let T_i be the set of all edges in both the path P_i and the set of all short subtrees attached to P_i. If a short subtree is attached to a junction point where two paths P_i and P_j meet, we arbitrarily assign it to either P_i or P_j so that the sets $\{T_1, \ldots, T_m\}$ form a partition of all of the edges of T into a collection of subtrees, each containing a single anchor.

For each $1 \leq i \leq m$ we define a directed walk W_i that starts at the anchor a_i, proceeds along P_i in the direction toward the root s, and collects every edge in T_i by tracing out an Euler tour around each of the short subtrees in T_i that are attached to P_i. One may easily verify that it is always possible to quickly find such a walk such that the following properties are satisfied:

– W_i contains each edge in P_i exactly once and always proceeds in the direction toward s when collecting each edge in P_i.
– W_i contains each edge in $T_i \setminus P_i$ exactly twice: once proceeding in the direction away from s, and once in the direction toward s.

We now greedily assign the edges in the short subtrees in T_i to buses by simply adding edges to buses in the order in which they are visited by W_i. We first initialize a bus β_1 at the anchor a_i and have it travel along W_i until the total length of all of the edges it has traversed in the *downward* direction (away from the root s) is exactly $\frac{R}{2}$. At this point, we assume it lies on some vertex v_1 (if not, we may imagine adding v_1 to the middle of an existing edge in T_i, although this will not be relevant to our solution as there are then no children at v_1). We send bus β_1 from v_1 immediately back to the root s and create a new bus β_2 that starts at v_1 and continues to follow W_i until it too has traversed exactly $\frac{R}{2}$ length in edges of T_i in the downward direction. We assume it then lies at a vertex v_2, create a new bus β_3 that starts at v_2 and continues to follow W_i, and so on. Eventually, some bus β_k will pick up the last remaining children and proceed to the root s, possibly with leftover detour to spare. We observe that the number of buses used is exactly $\left\lceil \frac{2 \sum_{e \in T_i \setminus P_i} d(e)}{R} \right\rceil$ since each bus other than the last one consumes exactly $\frac{R}{2}$ of the downward directed edges in W_i, and W_i proceeds downward along each edge in $T_i \setminus P_i$ exactly once. We also note that this is a feasible solution since a bus travelling a total downward direction of $\frac{R}{2}$ must make a detour no greater than R.

Doing this for each edge set T_i yields a feasible solution to the original instance using exactly

$$\sum_{i=1}^{m} \left\lceil \frac{2 \sum_{e \in T_i \setminus P_i} d(e)}{R} \right\rceil$$

buses. This is at most

$$m + \frac{2}{R} \sum_{i=1}^{m} \sum_{e \in T_i \setminus P_i} d(e) = m + 2 \frac{\sum_{e \in T \setminus Q} d(e)}{R} \leq 3OPT$$

by Proposition 2 and Proposition 3, where OPT is the optimal number of buses required in any feasible solution. Together with Lemma 1, this proves Theorem 1.

One may notice that the bounds given in Propositions 2 and 3 are necessarily also respected by fractional solutions to the LP relaxation of (IP). Together with the argument above, this immediately implies that (IP) has an integrality gap of at most 4 (and 3 in the case of infinite capacities), proving Theorem 2. In the appendix, we formally prove Theorem 2 by constructing feasible solutions of the dual of the LP relaxation of (IP). We also provide an example yielding a integrality gap lower bound of 2.

4 A 12.5-approximation to the uncapacitated SBP-R on trees

In this section, the School Bus Problem with Regret Minimization (SBP-R) is considered. SBP-R differs from SBP because of the exchanged roles of maximum regret and number of bus routes. In case of SBP-R, the number of routes is bounded by a given parameter $N \in \mathbb{N}$ while the maximum regret is to be minimized. We present here a proof of Theorem 3 by giving a polynomial time 12.5-approximation algorithm for SBP-R.

Without loss of generality, we may assume the tree T to be binary. Suppose we can fix a value R for the regret. We will develop an algorithm that, given an instance and the value R, either outputs a set of at most N bus routes, with a maximum regret of $12.5R$, or asserts that every solution with at most N buses must have a regret value $> R$. Then, we can do binary search on the regret values and output the best solution found.

Suppose to have guessed a value for R. First, we find a set of anchors A with respect to R as described in section 3. Now, we look for a set of routes which only start at the anchors. Using the notion introduced for SBP in section 3, our strategy is to cluster and cut the short subtrees into suitable pieces (called *tickets*) that can be collected efficiently by buses. In order to explain the cutting technique for the short subtrees, we need to introduce some more definitions.

Every vertex $v \in Q$ of the skeleton is called a *junction point* if either it is the root s or v has degree more than 2 in Q. Let J be the set of junction points. The skeleton Q can be split at its junction points into a set of edge-disjoint paths, which we will call *core segments*. Formally, a path in Q is a core segment if and only if its endpoints are anchors or junction points and it contains no junction points in its interior. The next lemma (whose proof is in the appendix) explains how we will cut a collection of short subtrees to produce suitable tickets.

Lemma 2. *There exists a polynomial-time algorithm that, given a path P and a collection $\mathcal{C}$ of short subtrees whose roots lie on P, produces a partition of the edges of $\mathcal{C}$ into tickets $E_0^{\mathcal{C}}$, $E_1^{\mathcal{C}}$, ..., $E_k^{\mathcal{C}}$ with $k \leq \left\lfloor \frac{\sum_{\mathcal{T} \in \mathcal{C}} d(\mathcal{T})}{R} \right\rfloor$ such that:*

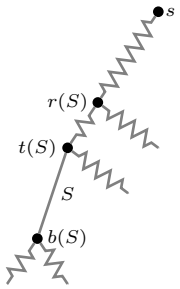
(P1) All of the edges in $E_0^{\mathcal{C}}$ can be collected with an additional regret $\leq 2.5R$ by a single bus whose route contains P.

(P2) For all $1 \leq i \leq k$, all the edges in $E_i^{\mathcal{C}}$ can be collected with an additional regret at most $3R$ by a single bus whose route contains P.

The next lemma is quite easy and will be useful later:

Lemma 3. *One can find a mapping $\phi : J \longrightarrow A$ from junction points to anchors in polynomial time with the following properties:*
(i) For all $j \in J$, the junction point j lies on the path $P(s, \phi(j))$;
(ii) For all $a \in A$, there is at most one junction point $j \in J$ with $\phi(j) = a$.

Proof. We construct ϕ by iteratively considering anchors $A = \{a_1, \ldots, a_m\}$ and building the skeleton Q using paths from anchors to junction points. We begin with the path $P(s, a_1)$ and set $\phi(s) = a_1$. When each new path $P(s, a_i)$ is added, a new junction point j_i is formed, namely the lowest intersection point of $P(s, a_i)$ with the previous paths. We set $\phi(j_i) = a_i$ for the remaining i, and we easily see that the resulting mapping ϕ satisfies properties (i) and (ii).



For every core segment S, let $t(S)$ and $b(S)$ be the top and the bottom junction points in S. Let further $r(S)$ be the highest junction point at distance at most $R/2$ from $t(S)$ (see Fig. at side). Our algorithm works as follows.

Algorithm 1

1. Find a maximal R-independent set of anchors A.
2. Initialize a default bus at each anchor $a \in A$.
3. For each junction point $j \in J$ in Bottom-Up order do:
   Assign an arbitrary left-to-right ordering of the two segments S_l, S_r with $t(S_l) = j = t(S_r)$.
   (a) Let $\mathcal{C}_1$ be the collection of short subtrees whose root node lies in the left core segment S_l at distance $\leq R/2$ from j. Let $\mathcal{C}_2$ be the collection of short subtrees whose root node lies in the core segment S_j with $b(S_j) = j$ at distance $> R/2$ from $t(S_j)$.
   (b) Apply Lemma 2 to $\mathcal{C}_1 \cup \mathcal{C}_2$ on $P = S_l \cup S_j$, and obtain tickets $E_0, E_1, \ldots, E_y$.
   (c) Let $\mathcal{C}_3$ be the collection of short subtrees whose root node lies in the right core segment S_r at distance $\leq R/2$ from j.
   (d) Apply Lemma 2 to $\mathcal{C}_3$ on path $P = S_r$, and obtain tickets $F_0, F_1, \ldots, F_z$.
   (e) Assign the tickets E_0 and F_0 to the default bus at $\phi(j)$, and remove these tickets.
   (f) Place the y tickets $E_1, \ldots, E_y$ and z tickets $F_1, \ldots, F_z$ at $r(S_l) = r(S_r)$.
4. For each anchor $a \in A$ do:
   Let $\mathcal{C}_a$ be the collection of short subtrees whose root node lies in the core segment S_a with $b(S_a) = a$ at distance $> R/2$ from $t(S_a)$.
   (a) Apply Lemma 2 to $\mathcal{C}_a$ on $P = S_a$, and obtain tickets $K_0, K_1, \ldots, K_w$.
   (b) Assign K_0 to a and remove this ticket.
   (c) Place the w tickets $K_1, \ldots, K_w$ at a.
5. Compute a maximum matching between tickets and anchors where a ticket can be matched to an anchor if and only if that anchor is a descendent of the node where the ticket is placed.

6. Add a new bus for each unmatched ticket.

Let B be the number of bus routes output by the algorithm. We show:
(i) the regret of each route output by the algorithm is at most $12.5 \cdot R$ and
(ii) B is a lower bound on the number of buses with regret at most R that are needed to cover all nodes.

Lemma 4. *Algorithm 1 outputs a set of buses with maximum regret* $12.5R$.

Proof. Any bus starting at an anchor a collects at most 4 different regret amounts:

- at most one ticket from the matching in step 5.
  In the worst case, the ticket is placed at a junction point $r(S)$ where S denotes the segment where the subtrees contributing to the ticket are rooted. Due to the definition of $r(S)$, the top junction point $t(S)$ is at distance at most $\frac{R}{2}$ from $r(S)$. Since the subtrees considered for the ticket are rooted on S at distance at most $\frac{R}{2}$ from $t(S)$, every loop of a ticket is rooted at the skeleton at distance $\leq R$ from $P(s, a)$. Together with $(P2)$, we can cover a ticket with a walk of regret at most $\leq 5R$.
- at most two remaining pieces of the Euler tour in step 3(e).
  There is at most one junction point j with $\phi(j) = a$ by (ii) of lemma 3. For a junction point j, each part E_0 and F_0 can be covered with regret $\leq 2.5R$ by (P2).
- at most one remaining piece of the Euler tour at a assigned in step 4(b).
  This ticket K_0 can be covered with regret $\leq 2.5R$ by (P2).

In total, the bus from anchor a collects regret of at most $5R+5R+2.5R = 12.5R$.

Lemma 5. *B is a lower bound on the number of buses with regret at most R needed to cover all points.*

Proof. Observe that if A is a set of anchors in T, then $A \cap F$ is a set of anchors if we restrict ourselves to any subtree $F \subseteq T$. In particular, proposition 3 can be strengthened in the following way:

Observation 2 *For any subtree $F \subseteq T$ rooted at some node f with skeleton Q_F, the number $\frac{\sum_{e \in F \setminus Q_F} d(e)}{R}$ is a lower bound on the number of buses that is needed to cover all points in F with routes of regret R that end in f.*

We can interpret B as the number of anchors plus the number of tickets that are not assigned to an anchor by the matching step. In order to show that this number is a lower bound, we show that every unmatched ticket forces also the optimal solution to use an additional bus. Consider the lowest unmatched ticket $\mathcal{T}$ at a junction point j. By construction we have that j is the highest junction point such that a bus starting from an anchor can pick up this ticket $\mathcal{T}$ on its shortest path to the root. Therefore, consider all anchors in the subtree of Q rooted at j. If there is an anchor that is not yet assigned a ticket, we can match it to F and obtain a contradiction to the maximality of the matching found in step 5. If there is an anchor a that is mapped to a ticket $\mathcal{T}'$ located at a junction

point above j on the path $P(s, j)$, then we can assign $\mathcal{T}$ to a and the unmatched ticket is moved strictly upwards. With this procedure, after at most $|J|$ many steps we find an unmatched ticket at a junction point j^* (that might be the root s) such that each anchor in the subtree F rooted at j^* is assigned to a ticket at some junction point within the subtree F. However, we can conclude at this point from the previous observation that a further bus is needed also for the optimum solution to cover all points with regret R.

References

[1] Adamaszek, A., Czumaj, A., and Lingas, A. (2009) PTAS for k-Tour Cover Problem on the Plane for Moderately Large Values of k. Algorithms and Computation, LNCS 5878, pp. 994–1003, Springer, 2009.

[2] Asano, Tetsuo and Katoh, Naoki and Tamaki, Hisao and Tokuyama, Takeshi (1997): Covering points in the plane by k-tours: towards a polynomial time approximation scheme for general k. STOC 1997.

[3] Blum, A., Chawla, S., Karger, D. R., Lane, T., Meyerson, A., and Minkoff, M. (2007): Approximation Algorithms for Orienteering and Discounted-Reward TSP. SIAM Journal on Computing, vol. 37, no. 2, pp. 653–670, 2007.

[4] Chekuri, C., Korula, N., and Pal, M. (2008) Improved Algorithms for Orienteering and Related Problems. in SODA, pp. 661–670, 2008.

[5] Grötschel, M., and Lovász, L. and Schrijver, A. (1981) The ellipsoid method and its consequences in combinatorial optimization. Combinatorica, 1(2), pp. 169–197, 1981.

[6] Haimovich, M., and Rinnoy Kan, A.H.G. (1985) Bounds and heuristic for capacitated routing problems. Mathematics of Operations Research, 10(4), pp. 527–542, 1985.

[7] Labbe, M., Laporte, G., and Mercure, H. (1991) Capacitated Vehicle Routing on Trees. Operations Research, vol. 39(4), pp. 616–622, 1991

[8] Laporte, G., Desrochers, M., and Norbert, Y. (1984): Two exact algorithms for the Distance Constrained Vehicle Routing Problem. Networks, vol. 14, pp. 47–61, 1984.

[9] Li, C.-L., Simchi-Levi, S., and Desrochers, M. (1992): On the distance constrained vehicle routing problem. Operations Research, vol. 40, pp. 790–799, 1992.

[10] Nagarajan, V., and Ravi, R. (2008): Approximation Algorithms for Distance Constrained Vehicle Routing Problems. Tepper School of Business, Carnegie Mellon University, Pittsburgh 2008.

[11] Park, J., and Kim, B.-I. (2010): The school bus routing problem: A review. European Journal of Operational Research, vol. 202, pp. 311–319, 2010.

[12] Spada, M., Bierlaire, M., and Liebling, Th. M. (2005): Decision-Aiding Methodology for the School Bus Routing and Scheduling Problem. Transportation Science, vol. 39(4), pp. 477 – 490, 2005.

[13] Toth, P., and Vigo, D. (2001): The Vehicle Routing Problem. 2001.

[14] Vazirani, V. V. (2001): Approximation Algorithms. Springer, 2001.

Appendix

Proof of Lemma 1. The idea is to cut the output of the approximation algorithm into parts of capacity C and connect them directly to the school. Let APX_C denote the number of buses output in case of capacity C and OPT_C, OPT_∞ the optimum solutions of the capacitated and uncapacitated case, respectively. We obtain

$$APX_C \leq \sum_{i=1}^{\alpha OPT_\infty} \frac{|\{w \in W : p(w) = P_i\}|}{C} + 1 \leq \frac{|W|}{C} + \alpha OPT_\infty \leq (1+\alpha)OPT_C$$

where P_i are the $\leq \alpha OPT_\infty$ walks output by the given approximation algorithm for the uncapacitated case.

Proof of Proposition 1. Assume for contradiction that there is a node $w \in S$ with $d^P(w,s) > d(w,s) + R$ while $d^P(v,s) \leq d(v,s) + R$. The triangle inequality then implies

$$d^P(v,s) \geq d^P(w,s) + d(v,w) > R + d(w,s) + d(v,w) \geq R + d(v,s),$$

a contradiction.

Lemma 6. *The greedy strategy for Set Cover implies a $\mathcal{O}(\log C)$-approximation to SBP. Using Dual-Fitting, the integrality gap of the LP relaxation of SBP's covering integer programming formulation can also be bounded by $\mathcal{O}(\log C)$.*

Proof. The greedy algorithm for SBP picks in every iteration a feasible route that maximizes the number of newly covered points. This means that we solve for each node $v \in V$ an Orienteering problem between s and v with length bound $d(s,v)+R$. Among these routes, we choose the one that visits the most uncovered nodes. An α-approximation to Orienteering with $\alpha \in \mathcal{O}(1)$ (e.g. $\alpha = 2+\varepsilon$, see [4]) implies that we can find a route that covers at least a $\frac{1}{\alpha}$ fraction of the optimum number of newly covered nodes at each iteration.
Consider now a route P in the optimum solution for SBP and order its nodes according to the iteration when they are covered by the greedy algorithm such that the first node was covered first and so on. By the time when $v_k \in P$ $(1 \leq k \leq |P|)$ is covered, at least $|P|-k+1$ many nodes on P are still uncovered. Thus the greedy algorithm covers in this iteration at least $\frac{1}{\alpha}(|P|-k+1)$ many nodes and the prize assigned to the node v_k during the greedy procedure (i.e. the average cost of a newly covered node) is at most $\frac{\alpha}{|P|-k+1}$. We obtain a prize of at most $\alpha \cdot \log C$ for route P, since the number of nodes on a walk is bounded by the capacity C. Summing up over all routes in the optimum solution for SBP, we get that the sum of the prizes (i.e. the number of routes selected by the greedy strategy) is at most $\mathcal{O}(\log C) \cdot OPT$, where OPT is the optimum value for the SBP.

In order to obtain the claimed result on the ratio of the optimal integer solution OPT and the fractional solution OPT_f, consider the dual of the LP

relaxation of SBP's covering integer programming formulation

$$\begin{aligned} \max \textstyle\sum_{v \in V \setminus \{s\}} y_v & \\ \textstyle\sum_{v \in P} y_v & \leq 1 \ \forall \text{ feasible walk } P \\ y_v & \geq 0 \ \forall v \in V \setminus \{s\}. \end{aligned}$$

We construct a feasible dual solution to prove a lower bound on the optimum value of the LP relaxation. Let

$$y_v := \frac{price(v_k)}{\alpha \log C}$$

for all $v \in V$. To show the feasibility, consider a walk P of i nodes that are numbered as above. Note that $i \leq C$. In the iteration when the node $v_k \in P$ is covered, there are still at least $i - k + 1$ nodes on P to cover. Since we pick approximately the best set, we pay at most $price(v_k) \leq \frac{\alpha}{i-k+1}$ for covering v_k. It follows that $y_{v_k} \leq \frac{1}{\log C(i-k+1)}$. If we sum over all nodes of P, we obtain

$$\sum_{k=1}^{i} y_{v_k} = \frac{1}{\log C} \sum_{k=1}^{i} \frac{1}{k} \leq 1.$$

Thus y is a feasible dual solution and we have

$$OPT \leq \sum_{v \in V} price(v) = \alpha \log C \sum_{v \in V} y_v \leq \mathcal{O}(\log C) \cdot OPT_f.$$

Proof of Theorem 2. The dual LP is a packing problem where we have a exponential number of constraints bounding the profits that are collected on each feasible set of children that can be picked up in a single walk.

$$\begin{aligned} \max \textstyle\sum_{v \in V \setminus \{s\}} y_v & \qquad (D) \\ \textstyle\sum_{v \in S} y_v & \leq 1 \ \forall \text{ feasible sets of children } S \\ y_v & \geq 0 \ \forall v \in V \setminus \{s\}. \end{aligned}$$

To prove our bound, we need to state the following lemma proven by [10]:

Lemma 7. [Distribution Lemma] *For any tree H with root r and distance function d on the edges, it is possible to distribute a total profit of 1 among the leaves of H such that the profit contained in any rooted (at r) subtree F is at most $\frac{d(F)}{d(H)}$.*

There are three things to prove for a feasible fractional solution to the LP relaxation of (IP):

i) The number $\frac{|V|}{C}$ is a lower bound on the value of any fractional solution. This lower bound is trivial. We assign a profit of $\frac{1}{C}$ to every node $v \in V \setminus \{s\}$. Any walk that collects profit > 1 has to visit more than C nodes.

ii) The size $|A|$ of the set of anchors is a lower bound on the value of any fractional solution.
The profit function that assigns a profit 1 to every anchor and 0 to all other vertices is a feasible solution to (D), since a feasible walk can never visit more than one anchor and thus collects profit at most 1.

iii) The lower bound of Proposition 3 holds for all fractional solutions.
In order to show this, we apply the distribution lemma to every short subtree H from the collection $\mathcal{H}$ of short subtrees that form $T \setminus Q$. On each subtree H, an amount of $d(H)/R$ is distributed among its leaves. Every other vertex gets profit 0. Therefore we distribute in total a profit of $\sum_{e \in T \setminus Q} d(e)/R$ over the vertices of T.
It remains to prove the feasiblity of this dual solution. Consider a feasible walk P in T and assume that it collects a profit > 1. This means by the distribution lemma that P visits the following length among edges of short subtrees.

$$\sum_{e \in (T \setminus Q) \cap P} d(e) = \sum_{H \in \mathcal{H}} \frac{\sum_{e \in H \cap P} d(e)}{d(H)} d(H) \geq \text{ profit}(P) \cdot R > R$$

This is a contradiction to the Observation 1.

Now we bring the three lower bounds together to obtain the claimed result. Let APX denote the feasible integer solution that we obtain from combining theorem 1 with lemma 1. Combining their proofs, we have:

$$APX \leq \frac{|V|}{C} + |A| + 2\frac{\sum_{e \in T \setminus Q} d(e)}{R} \leq 4 \cdot OPT_f.$$

Note that in case of unlimited capacity, the term $\frac{|V|}{C}$ is omitted and we obtain an integrality gap of at most 3.

The worst example that we are aware of has integrality gap 2. It consists in a star with $n+1$ nodes and edges of unit distance from the center. Set $R := 2(n-2)$ and the capacity C unlimited. The best integer solution uses exactly two routes while the fractional solution considers n routes (each possible route that skips exactly one leaf) with $\frac{1}{n-1}$ fraction. This yields a fractional solution close to 1 for n big enough and thus the claimed result.

Proof of Lemma 2. The idea is to cut the Euler tour of all short subtrees in $\mathcal{C}$ into suitable pieces. To do so, we first shift all short subtrees to the lowest node v of path P. We take then the Euler tour of all subtrees. Starting from v, cut it at the first node such that the current piece has length $> 2R$. Continue like this to obtain $k+1$ tickets.
Denote by $E_1^{\mathcal{C}}, \ldots, E_k^{\mathcal{C}}$ all but the last piece. The last part of the Euler tour defines $E_0^{\mathcal{C}}$. Note that $k \leq \left\lfloor \frac{\sum_{\mathcal{T} \in \mathcal{C}} d(\mathcal{T})}{R} \right\rfloor$, and that every cutting point is covered by exactly two tickets. Since every node needs to be covered only once in a solution, we can remove the last edge from each set $E_i^{\mathcal{C}}$ $(1 \leq i \leq k)$. By construction,

the resulting length is at most $2R$. Both the starting and the end point of each ticket $E_i^{\mathcal{C}}$ $(1 \leq i \leq k)$ are at distance at most $\frac{R}{2}$ from v (cf. the definition of a short subtree). Since the Euler tour goes back to v after one subtree is finished, we obtain from a ticket a set of loops of total length at most $3R$. A bus whose root walk contains P can cover all edges corresponding to a ticket with regret at most $3R$.
The last piece $E_0^{\mathcal{C}}$ of the Euler tour remaining from the cutting procedure certainly has length $\leq 2R$. Since the Euler tour goes back to v, only the distance to the starting point of the last piece has to be connected to v in order to obtain a set of loops. As in the previous case, one bus can cover these loops with regret $2.5R$, since the height of each subtree is at most $\frac{R}{2}$.
As one can easily see, this strategy fulfills the properties (P1) and (P2) and runs in polynomial time.