Routing Open Shop with unit processing times, few machines, and few locations∗

René van Bevern†,‡, Artem V. Pyatkin†,‡

†Novosibirsk State University, Novosibirsk, Russian Federation
‡Sobolev Institute of Mathematics, Siberian Branch of the Russian Academy of Sciences, Novosibirsk, Russian Federation

Abstract

Open Shop is a classical scheduling problem: given a set $\mathcal{J}$ of jobs and a set $\mathcal{M}$ of machines, find a minimum-makespan schedule to process each job $J_i \in \mathcal{J}$ on each machine $M_j \in \mathcal{M}$ for a given amount $p_{ij}$ of time such that each machine processes only one job at a time and each job is processed by only one machine at a time. In Routing Open Shop, the jobs are located in the vertices of an edge-weighted graph $G = (V, E)$, whose edge weights determine the time needed for the machines to travel between jobs. The travel times also have a natural interpretation as sequence-dependent family or batch setup times. Routing Open Shop is NP-hard for $|V| = |M| = 2$. For the special case with unit processing times $p_{ij} = 1$, we exploit a variant of Galvin’s theorem about list-coloring edges of bipartite graphs to prove a theorem that gives a sufficient condition for the completability of partial schedules. Exploiting this schedule completion theorem and integer linear programming, we show that Routing Open Shop with unit processing times is solvable in $2^{|V|}|E|\log |V||M|$ poly($|\mathcal{J}|$) time, that is, fixed-parameter tractable parameterized by $|V| + |\mathcal{M}|$. Various upper bounds shown using the schedule completion theorem suggest it to be likewise beneficial for the development of approximation algorithms.

Keywords: combinatorial optimization, routing, scheduling, graph theory, fixed-parameter tractability

1. Introduction

One of the most fundamental and classical scheduling problems is Open Shop (Gonzalez and Sahni, 1976), where the input is a set $\mathcal{J} := \{J_1, \ldots, J_n\}$ of jobs, a set $\mathcal{M} := \{M_1, \ldots, M_m\}$ of machines, and the processing time $p_{ij}$ that job $J_i$ needs on machine $M_j$; the task is to process all jobs on all machines in a minimum amount of time such that each machine processes at most one job at a time and each job is processed by at most one machine at a time.

Averbakh et al. (2006) introduced the variant Routing Open Shop, where the jobs are located in the vertices of an edge-weighted graph, whose edge weights determine the time needed for the machines to travel between jobs. Initially, the machines are located in a depot. The task is to minimize the time needed for processing all jobs by all machines and returning all machines to the depot. Routing Open Shop models, for example, tasks where machines have to perform maintenance work on stationary objects in a workshop (Averbakh et al., 2006). Routing Open Shop has also been interpreted as a variant of Open Shop with sequence-dependent family or batch setup times (Allahverdi et al., 2008; Zhu and Wilhelm, 2006). Formally, Routing Open Shop is defined as follows.

Definition 1.1 (Routing Open Shop). An instance of Routing Open Shop consists of a graph $G = (V, E)$ with a depot $v^* \in V$ and travel times $c : E \to \mathbb{N} \setminus \{0\}$, jobs $\mathcal{J} := \{J_1, \ldots, J_n\}$ with locations $L : \mathcal{J} \to V$, machines $\mathcal{M} := \{M_1, \ldots, M_m\}$, and, for each job $J_i$ and machine $M_q$, a processing time $p_{iq} \in \mathbb{N}$. A route with $s$ stays is a sequence $R := (R_i)_{i=1}^s$ of stays $R_i = (a_i, v_i, b_i) \in \mathbb{N} \times V \times \mathbb{N}$ from time $a_i$ to time $b_i$ in the vertex $v_i$ for $1 \leq i \leq s$ such that $v_1 = v^*$, $a_1 = 0$, and $b_i + c(v_i, v_{i+1}) \leq a_{i+1} \leq b_{i+1}$ for $1 \leq i \leq s - 1$. The length of $R$ is the end $b_s$ of the last stay.

A schedule $S : \mathcal{J} \times \mathcal{M} \to \mathbb{N}$ is a total function determining the start time $S(J_i, M_q)$ of each job $J_i$ on each machine $M_q$. That is, each job $J_i$ is processed by each machine $M_q$ in the half-open time interval $[S(J_i, M_q), S(J_i, M_q) + p_{iq})$. A schedule is feasible with respect to routes $(R_i)_{i=1}^s$ if

(i) no machine $M_q$ processes two jobs $J_i \neq J_j$ at the same time, that is, $S(J_i, M_q) + p_{iq} \leq S(J_j, M_q) \lor S(J_j, M_q) + p_{jq} \leq S(J_i, M_q)$ for all jobs $J_i \neq J_j$ and machines $M_q$,

(ii) no job $J_i$ is processed by two machines $M_q, M_r$ at the same time, that is, $S(J_i, M_q) + p_{iq} \leq S(J_i, M_r) \lor S(J_i, M_r) + p_{ir} \leq S(J_i, M_q)$ for all jobs $J_i$ and machines $M_q \neq M_r$,

(iii) machines stay in the location $L(J_i)$ while executing a job $J_i$, that is, for each job $J_i$ and machine $M_q$ with route $R_i = (R_k)_{k=1}^s$, there is a $k \in \{1, \ldots, s\}$ such that $R_k = (a_k, L(J_i), b_k)$ with $a_k \leq S(J_i, M_q) \leq S(J_i, M_q) + p_{iq} \leq b_k$.

A schedule $S$ is feasible and has length $L$ if there are machine routes $(R_i)_{i=1}^s$ of length $L$ such that $S$ is feasible with respect to $(R_i)_{i=1}^s$. An optimal solution to a Routing Open Shop instance is a feasible schedule of minimum length.
Preemption and unit processing times. Open Shop is NP-hard for $|M| = 3$ machines (Gonzalez and Sahni, 1976). Thus, so is ROUTING OPEN SHOP with $|V| = 1$ vertex and $|M| = 3$ machines. ROUTING OPEN SHOP remains (weakly) NP-hard even for $|V| = |M| = 2$ (Averbakh et al., 2006); there are approximation algorithms both for this special and the general case (Averbakh et al., 2005; Chernykh et al., 2013; Kononov, 2015; Yu et al., 2011). However, Open Shop is solvable in polynomial time if

(1) job preemption is allowed, or

(2) all jobs $j$ have unit processing time $p_{ij} = 1$ on all machines $M_j$.

It is natural to ask how these results transfer to ROUTING OPEN SHOP.

Regarding (1), Pyatkin and Chernykh (2012) have shown that ROUTING OPEN SHOP with allowed preemption is solvable in polynomial time if $|V| = |M| = 2$, yet NP-hard for $|V| = 2$ and an unbounded number $|M|$ of machines.

Regarding (2), ROUTING OPEN SHOP with unit processing times models tasks where machines process batches of equal-length jobs in several locations (or of different types) and where the transportation of machines between the locations (or the setup between jobs of different types) takes significantly longer than processing each individual job in a batch. Herein, there are conceivable situations where the number of machines and locations is small.

ROUTING OPEN SHOP with unit processing times clearly is NP-hard even for $|M| = 1$ machine since it generalizes the metric travelling salesperson problem. It is not obvious whether it is solvable in polynomial time even when both $|V|$ and $|M|$ are fixed. We show the even stronger result that ROUTING OPEN SHOP with unit processing times is solvable in $2^{O(|V|^2 \log |V|/|M|)} \cdot \text{poly}(|V|)$ time, that is, fixed-parameter tractable.

Fixed-parameter algorithms. Fixed-parameter algorithms are an approach towards efficiently and optimally solving NP-hard problems: the main idea is to accept the exponential running time for finding optimal solutions to NP-hard problems, yet to confine it to some smaller problem parameter $k$ (Cygan et al., 2015; Downey and Fellows, 2013; Flum and Grohe, 2006; Niedermeier, 2006). A problem with parameter $k$ is called fixed-parameter tractable (FPT) if there is an algorithm that solves any instance $I$ in $f(k) \cdot \text{poly}(|I|)$ time, where $f$ is an arbitrary computable function. The corresponding algorithm is called fixed-parameter algorithm. In contrast to algorithms that merely run in polynomial time for fixed $k$, fixed-parameter algorithms can potentially solve NP-hard problems optimally and efficiently if the parameter $k$ is small.

Recently, the field of fixed-parameter algorithmics has shown increased interest in scheduling (van Bevern et al., 2015a,c, 2016a,b; Bodlaender and Fellows, 1995; Fellows and Martin, 2003; Halldorsson and Karlsson, 2006; Hermelin et al., 2015; Mnich and Wiese, 2015) and routing (van Bevern et al., 2014, 2015b; Dorn et al., 2013; Gutin et al., 2013, 2014a,b, 2015; Klein and Marx, 2014; Sorge et al., 2011, 2012), whereas fixed-parameter algorithms for problems containing elements of both routing and scheduling are still rare (Böckenhauer et al., 2007).

Our results. Using a variant of Galvin’s theorem on list-coloring edges of bipartite graphs (Borodin et al., 1997; Galvin, 1995), in Section 3 we prove a sufficient condition for the polynomial-time completability of partial schedules, which do not necessarily assign start times to all jobs on all machines, into feasible schedules.

We use the schedule completion theorem to prove upper bounds on various parameters of optimal schedules, in particular on their lengths in Section 4.

Using these bounds and integer linear programming, in Section 5 we show that ROUTING OPEN SHOP with unit processing times is fixed-parameter tractable parameterized by $|V| + |M|$. Note that, for arbitrary processing times, this is impossible unless $P = NP$.

Since the schedule extension theorem is a useful tool for proving upper bounds on various parameters of optimal schedules, we expect the schedule completion theorem to be likewise beneficial for approximation algorithms.

Input encoding. In general, a ROUTING OPEN SHOP instance requires at least $\Omega(|J| \cdot |M| + |E|)$ bits in order to encode the processing time of each job on each machine and the travel time for each edge. We call this the standard encoding. In contrast, an instance of ROUTING OPEN SHOP with unit processing times can be encoded using $O(|V|^2 \log c_{\max} + |V| \cdot \log |J|)$ bits by simply associating with each vertex in $V$ the number of jobs it contains, where $c_{\max}$ is the maximum travel time. We call this the compact encoding.

All running times in this article are stated for computing and outputting a minimum-length schedule, whose encoding requires at least $\Omega(|J| \cdot |M|)$ bits for the start time of each job on each machine. Thus, outputting the schedule is impossible in time polynomial in the size of the compact encoding. We therefore assume to get the input instance in standard encoding, like for general ROUTING OPEN SHOP.

However, we point out that the decision version of ROUTING OPEN SHOP with unit processing times is fixed-parameter tractable parameterized by $|V| + |M|$ even when assuming the compact encoding: our algorithm is able to decide whether there exists a schedule of given length $L$ in $2^{O(|V| + |M| \log |V| + |M|)} \cdot \text{poly}(|I|)$ time, where $|I|$ is the size an instance $I$ given in compact encoding. To this end, the algorithm does not apply the schedule completion Theorem 3.4 to explicitly construct a schedule but merely to conclude its existence.

2. Preprocessing for metric travel times

In this section, we show how any instance can be transformed into an equivalent instance with travel times satisfying the triangle inequality. This will allow us to assume that, in an optimal schedule, a machine only stays in a vertex if it processes at least one job there: otherwise, it could take a “shortcut”, bypassing the vertex.

Lemma 2.1. Let $I$ be a ROUTING OPEN SHOP instance and $I'$ be obtained from $I$ by replacing the graph $G = (V, E)$ with travel times $c: E \to \mathbb{N}$ by a complete graph $G'$ on the vertex set $V$
with travel times \( c' : [v, w] \mapsto \text{dist}_G(v, w) \), where \( \text{dist}_G(v, w) \) is the length of a shortest path between \( v \) and \( w \) in \( G \) with respect to \( c \).

Then, any schedule for \( I \) is a schedule of the same length for \( I' \) and vice versa. Moreover, \( c' \) satisfies the triangle inequality \( c'(v, w) \leq c'(u, v) + c'(u, w) \) for all \( u, v, w \in V \) and can be computed in \( O(|V|^2) \) time.

**Proof.** It is obvious that \( c' \) satisfies the triangle inequality. It can be computed in \( O(|V|^2) \) time using the Floyd-Warshall algorithm (Floyd, 1962).

Any feasible schedule for \( I \) is also a feasible schedule for \( I' \) of the same length since any route \( R \) for \( I \) is also a route for \( I' \): for two consecutive stays \( (a_i, v_i, b_i) \) and \( (a_{i+1}, v_{i+1}, b_{i+1}) \) of \( R \), one has \( b_i + c'(v_i, v_{i+1}) \leq b_i + c(v_i, v_{i+1}) \leq a_{i+1} \).

Any feasible schedule for \( I' \) is a feasible schedule of the same length for \( I \) since any route \( R' \) with \( s \) stays for \( I' \) can be turned into a route of the same length with additional stays for \( I \): for each \( i \in \{1, \ldots, s - 1\} \), take two consecutive stays \( (a_i, v_i, b_i) \) and \( (a_{i+1}, v_{i+1}, b_{i+1}) \) on \( R' \) and a shortest path \( P = (v_1 = v_i, v_2, \ldots, v_{s-1} = v_{i+1}) \) between \( v_i \) and \( v_{i+1} \) in \( G \) with respect to \( c \). Between stay \( i \) and \( i + 1 \), add zero-length stays in the vertices of \( P \). That is, for each \( k \in \{1, \ldots, \ell - 2\} \), add stays

\[
\left( a_i + \sum_{j=1}^{k} c(w_j, w_{j+1}), \quad w_{k+1}, \quad a_i + \sum_{j=1}^{k} c(w_j, w_{j+1}) \right)
\]

to \( R' \). This yields a route \( R \) for \( I \) since

\[
a_i + \sum_{j=1}^{k} c(w_j, w_{j+1}) + c(w_{k+1}, w_{k+2}) \leq a_i + \sum_{j=1}^{k+1} c(w_j, w_{j+1})
\]

for all \( k \in \{1, \ldots, \ell - 1\} \). Moreover, \( R \) has the same length as \( R' \) since the end of the last stay has not changed. \( \square \)

The main advantage of working on instances satisfying the triangle inequality is that we may assume that machines in an optimal schedule do not stay in vertices without processing jobs in them, except for the depot, which is always the first and last stay of a machine.

**Lemma 2.2.** Let \( S \) be a feasible schedule of length \( L \) for a Routing Open Shop instance satisfying the triangle inequality.

Then, \( S \) is feasible with respect to machine routes \( (R_{M_k})_{M_k \in M} \) of length at most \( L \) such that, for each route \( R = ((a_k, v_k, b_k))_{k=1}^s \) and each stay \( (a_k, v_k, b_k) \) on \( R \), except, maybe, for \( k \in \{1, s\} \), there is a job \( J_i \in S \) with \( S(J_i, M_k) \in (a_k, b_k) \).

**Proof.** Since \( S \) is a feasible schedule of length \( L \), it is feasible with respect to machine routes \( (R_{M_k})_{M_k \in M} \) of length \( L \). Assume that each machine route \( R_{M_k} \) is minimal, that is, no stay can be removed without violating the feasibility of \( S \) with respect to \( R_{M_k} \).

For the sake of contradiction, assume that the route \( R_{M_k} = ((a_k, v_k, b_k))_{k=1}^s \) contains a stay \( (a_k, v_k, b_k) \), where \( k \notin \{1, s\} \), such that there is no job \( J_i \in S \) with \( S(J_i, M_k) \in (a_k, b_k) \). Then, removing \( (a_k, v_k, b_k) \) from \( R_{M_k} \) would yield a route \( R'_{M_k} \) with fewer stays since

\[
b_{k-1} + c(v_{k-1}, v_k) \leq b_{k-1} + c(v_k, v_{k+1}) \leq a_k + c(v_k, v_{k+1}) \leq b_k + c(v_k, v_{k+1}) \leq a_{k+1}.
\]

Since \( S \) is feasible with respect to \( R'_{M_k} \), this contradicts \( R_{M_k} \) being minimal. \( \square \)

Clearly, from Lemma 2.2, we get the following:

**Observation 2.3.** Vertices \( v \in V \setminus \{v'\} \) with \( F_v = \emptyset \) can be deleted from a Routing Open Shop instance satisfying the triangle inequality, where \( v' \) is the depot.

From now on, we assume that our input instances of Routing Open Shop satisfy the triangle inequality and exploit Lemma 2.2 and Observation 2.3.

### 3. Schedule completion theorem

In this section, we present a theorem that allows us to complete partial schedules, which do not necessarily assign a start point to each job on each machine, into feasible schedules.

In the following, we consider only Routing Open Shop with unit processing times and say that a machine \( M_k \) processes a job \( J_i \) at time \( S(J_i, M_k) \) if it processes job \( J_i \) in the time interval \([S(J_i, M_k), S(J_i, M_k) + 1)\). We use \( S(J_i, M_k) = \emptyset \) to denote that the processing time of job \( J_i \) on machine \( M_k \) is undefined.

**Definition 3.1 (Partial schedule).** A partial schedule with respect to given routes \( (R_{M_k})_{M_k \in M} \) of length at most \( L \) is a partial function \( S : J \times M \rightarrow \mathbb{N} \) satisfying Definition 1.1(iii) for those jobs \( J_i, J_j \in J \) and machines \( M_k, M_r \in M \) for which \( S(J_i, M_k) \neq \emptyset \) and \( S(J_j, M_r) \neq \emptyset \). For a partial schedule \( S : J \times M \rightarrow \mathbb{N} \), we introduce the following terminology:

\[
F_{J_i}^{S} := \{ J_j \in J : S(J_i, M_k) = \emptyset \}
\]

is the set of jobs that lack processing by machine \( M_k \),

\[
M_k^S := \{ M_j \in M : S(J_i, M_k) = \emptyset \}
\]

is the set of machines that job \( J_i \) lacks processing of (note that \( J_i \in F_{J_i}^{S} \) if and only if \( M_k \in M_k^S \)),

\[
T^{S}_{J_i} := \{ t \leq L \mid 3M_k \in M : S(J_i, M_k) = t \}
\]

is the set of time units where job \( J_i \) is being processed,

\[
T^{S}_{J_i} := \{ t \leq L \mid \exists J_k \in J : S(J_i, M_k) = t \}
\]

is the set of time units where machine \( M_k \) is processing,

\[
R^{S}_{J_i} := \{ t \leq L \mid R_{M_k} \text{ has a stay } (a_t, v, b_t) \text{ with } a_t \leq t < b_t \}
\]

are the time units where \( M_k \) stays in a vertex \( v \in V \), and

\[
F_{J_i} := \{ J_i \in J : L(J_i) = v \}
\]

is the set of jobs in vertex \( v \in V \) of \( G \).

The schedule completion theorem will allow us to turn any completable partial schedule into a feasible schedule. Intuitively, a schedule is completable if a machine has enough "free time" in each vertex to process all yet unprocessed jobs and to wait for other machines in the vertex to free the jobs to be processed.
Definition 3.2 (Completable schedule). Let \((R_M)_M \in M\) be a family of routes and, for each vertex \(v \in V\), let \(\bigcup_{M \in M} M^v \coloneqq M\) be a partition of machines such that, for any two machines \(M_i \in M^v_i\) and \(M_i \in M^v_i\) with \(s \neq t\), one has \(T^R_v \cap T^R_u = \emptyset\).

A partial schedule \(S : J \times M \rightarrow N\) with respect to \((R_M)_M \in M\) is comprehensible if, for each vertex \(v \in V\), each \(1 \leq s \leq g_i\), each machine \(M_i \in M^v\), and each job \(J_i \in J\times M\), it holds that
\[
|T^R_v \setminus (T^S_i \cup T^S_i)| \geq \max(|J\times M|, |M|, |M_i^v|, |M_i^v|).
\]

Example 3.3. Let \((R_M)_M \in M\) be routes such that all machines in are in the same vertex at the same time, that is, \(T^R_v = T^R_u\), for all vertices \(v \in V\) and machines \(M_i, M_i \in M\). Moreover, assume that each machine \(M_i \in M\) stays in each vertex \(v \in V\) at least \(\max(|J_i|, |M|)\), time, that is, \(|T^R_v| \geq \max(|J_i|, |M|)\). Then the empty schedule is comprehensible and, by the following schedule completion theorem, there is a feasible schedule with respect to the routes \((R_M)_M \in M\).

Theorem 3.4 (Schedule completion theorem). Given a partial schedule \(S : J \times M \rightarrow N\) that is comprehensible with respect to routes \((R_M)_M \in M\), one can compute a feasible schedule \(S' \supseteq S\) with respect to the routes \((R_M)_M \in M\) in time polynomial in \(|J| + |M| + |V| + \sum_{e \in M \cap |M_i^v|} |T^R_v|\).

We prove Theorem 3.4 using a stronger version of Galvin’s theorem about properly list-coloring the edges of bipartite graphs (Borodin et al., 1997; Galvin, 1995).

Definition 3.5 (Proper edge coloring, \(f\)-edge-choosable). By \(\deg(v)\), we denote the degree of a vertex \(v\), that is, the number of edges incident to \(v\).

A proper edge coloring of a graph \(G = (V, E)\) is a coloring \(C : E \rightarrow \mathbb{N}\) of the edges of \(G\) such that \(C(e_1) \neq C(e_2)\) if \(e_1 \cap e_2 \neq \emptyset\), that is, if \(e_1 \cap e_2\) share a vertex.

A graph \(G = (V, E)\) is \(f\)-edge-choosable for some function \(f : E \rightarrow \mathbb{N}\) if \(G\) allows for a proper edge coloring \(C : E \rightarrow \mathbb{N}\) with \(C(e) \in L_e\) for every family \(L_e \subseteq \mathbb{N} \mid e \in E\) with \(|L_e| \geq f(e)\).

Theorem 3.6 (Borodin et al. (1997)). Any bipartite graph \(G = (V, E)\) is \(f\)-edge-choosable for
\[
f : E \rightarrow \mathbb{N}, \{u, v\} \mapsto \max(\deg(u), \deg(v))\]

Remark 3.7. The proof given by Borodin et al. (1997) is constructive: a given bipartite graph \(G = (V, E)\) and a set \(L_e \subseteq \mathbb{N}\) with \(|L_e| \geq f(e)\) for each edge \(e \in E\), a proper edge coloring \(C : E \rightarrow \mathbb{N}\) with \(C(e) \in L_e\) is computable in time polynomial in the size of \(G\) and the color sets.

Before Borodin et al. (1997) proved Theorem 3.6, Galvin (1995) proved the special case for \(f : e \mapsto \Delta\), where \(\Delta\) is the maximum degree of \(G\). Before Galvin’s proof, its special case with \(G = K_{n,n}\) being a complete bipartite graph and \(f : e \mapsto n\) was known as Dinitz’ conjecture. We now proceed to prove Theorem 3.6 to prove Theorem 3.4.

Proof of Theorem 3.4. Let \(B = (J \cup M, X)\) be a bipartite graph with an edge \((J_i, M_j) \in X\) if and only if \(S(J_i, M_j) = \perp\) for \(J_i \in J\) and \(M_j \in M\). We compute a proper edge coloring \(C\) of \(B\) such that, for each edge \((J_i, M_j) \in X\), we have
\[
C((J_i, M_j)) \in T^R_v \setminus (T^S_i \cup T^S_i)
\]
and define \(S'((J_i, M_j)) :=\) \(C((J_i, M_j))\) if \((J_i, M_j) \in X\) and otherwise.

It remains to show that
\[
(1)\ \text{the edge coloring } C \text{ is computable in time polynomial in } |J| + |M| + |V| + \sum_{e \in M \cap |M_i^v|} |T^R_v|\text{ and that}
\]
\[
(2)\ S' \text{ is a feasible schedule.}
\]

(1) We obtain the proper edge coloring \(C\) by independently computing a proper edge coloring \(C_{e_i}\) satisfying (3.2) for each independent \(\text{subgraph } B_{e_i} := B[J_i \cup M_i]\) for all \(v \in V\) and \(1 \leq s \leq g_i\). To this end, observe that, in \(B_{e_i}\), a vertex \(J_i \in J_i\) has degree \(|M_i^v| \cap |M_i^v|\) and that a machine \(M_j \in M_i^v\) has degree \(|J_i\times |M_i^v|\). Thus, by Theorem 3.6, if, for each edge \(e_i := (J_i, M_j)\) of \(B_{e_i}\), we have a list \(L_{e_i}\) of colors with \(|L_{e_i}| \geq \max(|J_i\times |M_i^v|\), then \(B_{e_i}\) has a proper edge coloring \(C_{e_i}\) with \(C_{e_i}(e_i) \in L_{e_i}\) for each edge \(e_i\). Since \(C\) is computable (Definition 3.2), simply choosing \(L_{e_i} := T^R_v \setminus (T^S_i \cup T^S_i)\) for each edge \((J_i, M_j)\) of \(B_{e_i}\) yields a proper edge coloring \(C_{e_i}\) for \(C\) satisfying (3.2).

We now let \(C := \bigcup_{e_i \in E[B]} C_{e_i}\). This is a proper edge coloring for the bipartite graph \(B\) since, for edges \(e_{i_1}\) of \(B_{e_{i_1}}\) and \(e_{i_1}\) of \(B_{e_{i_1}}\) with \(v \neq w\) or \(s \neq t\), we have \(L_{e_{i_1}} \cap L_{e_{i_1}} = \emptyset\). For any vertex \(v \in V\) and machines \(M_i \in M_i\), \(M_j \in M_j\) with \(s \neq t\), one has \(T^R_v \cap T^R_u = \emptyset\), and for any machine \(M_i \in M_i\) and \(v \neq w\), one has \(T^R_v \cap T^R_u = \emptyset\). Moreover, \(C\) satisfies (3.2) since each \(C_{e_i}\) for \(v \in V\) and \(1 \leq s \leq g_i\) satisfies (3.2).

Regarding the running time, it is clear that, for each \(v \in V\) and \(1 \leq s \leq g_i\), the bipartite graph \(B_{e_i}\) and the sets \(L_{e_i} = L_{e_i}\) of allowed colors for each edge \((J_i, M_j)\) are computable in time polynomial in \(|J| + |M| + |T^R_v|\). Moreover, by Remark 3.7, the sought edge coloring \(C_{e_i}\) for each \(B_{e_i}\) is computable in time polynomial in \(|J| + \sum_{e_i \in E[B]} |L_{e_i}|\).

We first show that \(S'\) is a schedule. For each job \(J_i \in J\) and each machine \(M_j \in M\) we have \(S(J_i, M_j) \neq \perp \) or an edge \((J_i, M_j) \in X\) in \(B\). Thus, \(S'((J_i, M_j)) = S((J_i, M_j)) \neq \perp \) or \((J_i, M_j) \neq C((J_i, M_j)) \neq C((J_i, M_j))\). Finally, if \(S(J_i, M_j) = \perp \) and \((J_i, M_j) \neq \perp\), then \((J_i, M_j) \neq S'((J_i, M_j))\) since \((J_i, M_j) = C((J_i, M_j)) \in T^R_v \setminus (T^S_i \cup T^S_i)\) and \(S(J_i, M_j) \in T^S_i\).

3We abstain from a more detailed running time analysis since no such analysis is available for the forthcoming application of Theorem 3.6 (yet).
Now, let \( J_i, J_j \in J \) be two distinct jobs and \( M_q \in M \) be a machine. We show that \( S'(J_i, M_q) \neq S'(J_j, M_q) \). We distinguish three cases. If \( S(J_i, M_q) \neq \perp \) and \( S(J_j, M_q) \neq \perp \), then \( S'(J_i, M_q) = S(J_i, M_q) \neq S'(J_j, M_q) = S(J_j, M_q) \). If \( S(J_i, M_q) = \perp \) and \( S(J_j, M_q) \neq \perp \), then \( S'(J_i, M_q) = \perp \) since \( S'(J_i, M_q) = C(J_i, M_q) \neq C(J_j, M_q) \). Finally, if we have \( S(J_i, M_q) = \perp \) and \( S(J_j, M_q) = \perp \), then \( S'(J_i, M_q) = S'(J_j, M_q) \neq S'(J_i, M_q) = S'(J_j, M_q) \). \( \square \)

4. Upper and lower bounds

In this section, we show lower and upper bounds on the lengths of optimal solutions to Routing Open Shop with unit processing times. These will be exploited in our fixed-parameter algorithm and make first steps towards approximation algorithms.

We assume Routing Open Shop instances to be preprocessed to satisfy the triangle inequality. By Lemma 2.1, this does not change the length of optimal schedules. However, it ensures that the minimum cost of a cycle visiting each vertex of the graph \( G = (V, E) \) with travel times \( c: E \to \mathbb{N} \) at least once coincides with the minimum cost of a cycle doing so exactly once (Serdyukov, 1978), that is, of a Hamiltonian cycle.

A simple lower bound is given by the fact that, in view of Observation 2.3, all machines have to visit each vertex at least once and have to process \( |J| \) jobs.

**Observation 4.1.** Let \( H \) be a minimum-cost Hamiltonian cycle in the graph \( G = (V, E) \) with metric travel times \( c: E \to \mathbb{N} \). Then, any feasible schedule has length at least \( c(H) + |J| \).

A trivial upper bound can be given by letting the machines work

\[
\text{for jobs } v \in V \text{ and have to process } n - q + i \text{ if } i < q, \
i - q \text{ otherwise.}
\]

Call a cell \( s'_{iq} \) red if \( i < q \) and green otherwise. Note that if \( s'_{iq} \) and \( s'_{jq} \) are of the same color and \( i < j \) or \( r < q \), then \( s'_{iq} < s'_{jq} \).

Moreover, the number in a red cell is larger than the number in any green cell of the same row or column: if \( s'_{iq} \) is red and \( s'_{jq} \) is green, then from

\[
n + i > j \quad \text{follows} \quad s'_{iq} = n - q + i > j - q = s'_{jq}
\]

and if \( s'_{iq} \) is red and \( s'_{iq} \) is green, then from

\[
n - q > -r \quad \text{follows} \quad s'_{iq} = n - q + i > i - r = s'_{ir}.
\]

Let \( c_k = \sum_{v \in J} c(v_{i-1}, v_i) \) be the travel time from \( v_1 \) to \( v_k \) along \( H \). Clearly, the sequence \( (c_k)_{1 \leq k \leq n} \) is non-decreasing and \( c_{|V|} \leq c(H) \). Our schedule is now given by \( S = (s_{iq})_{1 \leq i \leq n, 1 \leq q \leq m} \), where

\[
s_{iq} := s'_{iq} + \left\{ \begin{array}{ll}
c_k & \text{if } L(J_i) = v_k \text{ and } s_{iq} \text{ is green}, \\
c(H) + c_k & \text{if } L(J_i) = v_k \text{ and } s_{iq} \text{ is red,}
\end{array} \right.
\]

and \( L(J_i) \) is the vertex where job \( J_i \) is located.

Let us prove that this schedule is feasible in terms of Definition 1.1. Indeed, by construction, for two elements \( s_{iq} \) and \( s_{ir} \) with \( i = j \) or \( q = r \) and \( s_{iq} > s_{ir} \), one has \( s_{iq} \geq \max_{v \in V} s_{ij} \) since the value added to \( s_{ij} \) is not smaller than the value added to \( s_{ir} \) due to our sorting of jobs by non-decreasing vertex indices and because the value added to any red cell is larger than any value added to a green cell. Therefore, conditions (i) and (ii) are satisfied.

It remains to determine the routes \( R_M = \{(a_{ij}, v_i, b_{ij})\}_{i,j=1}^n \) for each machine \( M_q \in M \). Machine \( M_q \) will follow \( H \) up to two times. During the first stay \((a_{ij}, v_i, b_{ij})\) in a vertex \( v_k \), it will process all jobs \( J_i \) such that \( s_{iq} \) is green. During the second stay \((a_{ij}^2, v_i, b_{ij}^2)\), it will process all jobs \( J_i \) such that \( s_{iq} \) is red. That is, the beginning and end times of the stays are

\[
a_{ij}^1 := \min\{S(J_i, M_q) | J_i \in J, L(J_i) = v_k \text{ and } s_{ij} \text{ is green},
\]
\[
a_{ij}^2 := \max\{S(J_i, M_q) | J_i \in J, L(J_i) = v_k \text{ and } s_{ij} \text{ is red},
\]

and the sequence \( (a_{ij}^1, v_i, b_{ij}^1) \), \( (a_{ij}^2, v_i, b_{ij}^2) \), it will process all jobs \( J_i \) such that \( s_{ij} \) is red. That is, the beginning and end times of the stays are

\[
as_{ij} := \min\{S(J_i, M_q) | J_i \in J, L(J_i) = v_k \text{ and } s_{ij} \text{ is green},
\]
\[
as_{ij} := \max\{S(J_i, M_q) | J_i \in J, L(J_i) = v_k \text{ and } s_{ij} \text{ is red},
\]

By the choice of \( s_{ij} \) for red cells, the machines have enough time to go around \( H \) a second time. It is thus easy to verify that the chosen routes satisfy the condition (iii) and that the length of the schedule is at most \( n + 2c(H) \). \( \square \)

We next study for which instances one gets an upper bound that matches the lower bound from Observation 4.1. In Example 3.3, we have already seen that arbitrary machine routes that stay in each vertex \( v \) at least \( \max\{|J|, |M|\} \) time can be completed into a feasible schedule. We therefore distinguish vertices \( v \) for which staying \( |J| \) time is both necessary and sufficient.

**Definition 4.4 (Criticality of vertices).** For a vertex \( v \in V \), we denote by

\[
k(v) := \max\{0, |M| - |J| \}
\]

the criticality of \( v \), and by

\[
K := \sum_{v \in V} k(v)
\]

the total criticality.

A vertex \( v \in V \) is critical if \( k(v) > 0 \), that is, if \( |J| < |M| \).
5. Fixed-parameter algorithm

In this section, we present a fixed-parameter algorithm for Routing Open Shop with unit processing times, which is our main algorithmic result:

**Theorem 5.1. Routing Open Shop with unit processing times is solvable in** $2^{O(|F||M|^2 \log |V||M|)} \cdot \text{poly}(|F|)$

The outline of the algorithm for Theorem 5.1 is as follows: in Section 5.1, we use the schedule completion Theorem 3.4 to show that the routes of a minimum-length schedule comply with one of $2^{|F||M|^2 \log |V||M|}$ pre-schedules, which determines the sequence of vertices that each machine stays in, the durations of stays in critical vertices, and the time offsets between stays in critical vertices.

In Section 5.2, we use integer linear programming to compute, for each pre-schedule, shortest complying routes so that each machine stays in each non-critical vertex $v$ for at least $|F_v|$ time. The schedule for non-critical vertices is then implied by the schedule completion Theorem 3.4, whereas we compute the schedule for critical vertices using brute force.

5.1. Enumerating pre-schedules

We will show that the routes of a minimum-length schedule comply with some pre-schedule, which is defined below and illustrated in Figure 2.

**Definition 5.2 (Pre-schedule).** A pre-stay is a triple $(M_q, v, \sigma_i) \in \mathcal{M} \times V \times \{1, \ldots, |V||M| + 2\}$, intuitively meaning that a machine $M_q \in \mathcal{M}$ has its $\sigma_i$-th stay in vertex $v \in V$. We call $T = (\{(M_q, v, \sigma_i)\}_{i=1}^{|I|})$ a pre-stay sequence if,

(i) for each $M_q \in \mathcal{M}$, the $\sigma_i$ with $q_i = q$ increase in steps of one for increasing $i$.

**Machine routes** $(R_{M_q})_{M_q \in \mathcal{M}}$, where $R_{M_q} = (\{a_q^0, w_q^0, b_q^0\})_{i=1}^{|I|}$, comply with a pre-stay sequence if

(ii) route $R_{M_q}$ has a stay $(a_q^i, w_q^i, b_q^i)$ if and only if $(M_q, v, \beta_q^i) \in T$ and,

(iii) for pre-stays $(M_q, v, \sigma_i)$ and $(M_{q'}, v, \sigma_j)$ with $i < j$, one has $a_q^i \leq a_{q'}^j$.

Let $\mathcal{K}$ be the indices of pre-stays in critical vertices of $T$. A length assignment is a map $A: \mathcal{K} \rightarrow [0, \ldots, 2|V||M| - 1]$. Machine routes $(R_{M_q})_{M_q \in \mathcal{M}}$ comply with a length assignment $A$ if,

(iv) for each pre-stay $(M_q, v, \sigma_i)$ in $T$ with $i \in \mathcal{K}$, one has $b_q^i - a_q^i = A(i)$.

A displacement is a map $D: \mathcal{K} \rightarrow [0, \ldots, 2|V||M|]$. The machine routes $(R_{M_q})_{M_q \in \mathcal{M}}$ comply with a displacement $D$ if
We show that an optimal solution for R

The proof of Proposition 5.4 is based on proving that there are

The set with some pre-schedule (the pre-schedule is not shown). Hatched squares correspond to jobs being processed. Illustrated are the lengths of stays and displacements between stays in critical vertices that are consecutive in the pre-stay sequence (stays in non-critical vertices are not shown). Herein, this displacement is either smaller than 2\(|M|\) in which case the stays cannot intersect in time for any set of complying machine routes since stays in critical vertices have length at most 2\(|M| - 1\) by Definition 5.2(iv). This will allow us, without knowing the absolute start and end time of stays, to check the feasibility of partial schedules for stays in critical vertices in any set of complying machine routes.

We call (\(T, A, D\)) a pre-schedule and say that machine routes comply with (\(T, A, D\)) if they comply with each of \(T, A, D\), that is, (i)–(v) hold.

We show that an optimal solution for Routing Open Shop with unit processing times can be found by solving instances of the following problem:

**Problem 5.3.**

**Input:** An instance \(I\) of Routing Open Shop with unit processing times, a pre-schedule \((T, A, D)\), and a natural number \(L\).

**Task:** Compute a schedule whose machine routes \((R_M)_{M \in \mathcal{M}}\) have length at most \(L\) and comply with \((T, A, D)\), if such a schedule exists.

**Proposition 5.4.** For a Routing Open Shop instance \(I\) with unit processing times there is a set \(\mathcal{I}\) of \(2^{O(|V||M| \log |V||M|)}\) instances of Problem 5.3 such that

(i) if some instance \((I, (T, A, D), L) \in \mathcal{I}\) has a solution \(S\), then \(S\) is a schedule of length at most \(L\) for \(I\) and

(ii) there is a minimum-length schedule \(S\) for \(I\) such that \(S\) is a solution for at least one instance \((I, (T, A, D), L) \in \mathcal{I}\), where \(L\) is the length of \(S\).

The set \(\mathcal{I}\) can be generated in \(2^{O(|V||M| \log |V||M|)} \cdot \text{poly}(|\mathcal{J}|)\) time.

The proof of Proposition 5.4 is based on proving that there are at most \(2^{O(|V||M| \log |V||M|)}\) pre-schedules and that the routes of an optimal schedule comply with at least one pre-schedule.

We will use the following lemma to show that there is a pre-stay sequence that the routes of an optimal schedule comply with:

**Lemma 5.5.** Each of the routes \((R_M)_{M \in \mathcal{M}}\) of an optimal schedule consists of at most \(|V||M| + 2\) stays.

**Proof.** Let \(H\) be a minimum-cost Hamiltonian cycle for the graph \(G\) with travel times \(c : E \rightarrow \mathbb{N}\). Let \(M_q \in \mathcal{M}\) be an arbitrary machine. It has to stay in all vertices and return to the depot, that is, its tour \(R_{M_q}\) has at least \(|V| + 1\) stays. Moreover, by Observation 4.1, its length is at least \(c(H) + |\mathcal{J}|\). Since \(c(e) \geq 1\) for each \(e \in E\) (see Definition 1.1), each additional stay increases the length of the tour by at least one.

Thus, if \(R_{M_q}\) had more than \(|V| + K + 1\) stays, where \(K\) is the total critically of vertices in the input instance (see Definition 4.4), then it would have length at least \(c(H) + |\mathcal{J}| + K + 1\), contradicting the optimality of the schedule by Proposition 4.5. Thus, the number of stays on \(R_{M_q}\) is at most

\[|V| + K + 1 = |V| + \sum_{v \in V} \max\{0, |M| - |\mathcal{J}_v|\} + 1\]

\[\leq |V| + (|V| - 1)(|M| - 1) + |M| + 1 = |V||M| + 2\]

since, by Observation 2.3, only for the depot \(v^*\) one might have \(\mathcal{J}_{v^*} = \emptyset\).

We will use the following lemma to show that there are also length assignments and displacements that the routes of an optimal schedule comply with. For the notation used in Lemma 5.6, recall Definition 3.1.

**Lemma 5.6.** For each feasible schedule \(S\) with respect to machine routes \((R_M)_{M \in \mathcal{M}}\), there is a feasible schedule \(S'\) of the same length with respect to machine routes \((R'_M)_{M \in \mathcal{M}}\) such that \(|T_v^{R'_M}| \leq \max\{|\mathcal{J}_v|, |M|\} + |M| - 1\) for each vertex \(v \in V\).

**Proof.** For each machine \(M_q \in \mathcal{M}\), construct the route \(R'_{M_q}\) from the route \(R_{M_q} = ((a_k, v_k, b_k))_{k=1}^{2}\) as follows:

1. If \(|T_v^{R_{M_q}}| \leq \max\{|\mathcal{J}_v|, |M|\} + |M| - 1\), then \(R'_{M_q} := R_{M_q}\).
2. Otherwise, let \(R'_{M_q} := ((a'_i, v_i, b'_i))_{i=1}^{s}\), where \(a_i \leq a'_i \leq b'_i \leq b_i\) for \(1 \leq i \leq s\) are chosen arbitrarily with \(|T_v^{R'_{M_q}}| = \max\{|\mathcal{J}_v|, |M|\} + |M| - 1\).

Denote by \(\overline{\mathcal{M}} := \{M_q \in \mathcal{M} | R_{M_q} \neq R'_{M_q}\}\) the set of machines whose tours have been altered. If \(\overline{\mathcal{M}} = \emptyset\), then there is nothing...
to prove. Henceforth, assume $M \neq \emptyset$. Then, $S$ might not be a feasible schedule for the routes $(R_M^*)_{M \in M}$ but

$$S'(J_i, M_q) := \begin{cases} \perp & \text{if } M_q \in \overline{M} \\ S(J_i, M_q) & \text{otherwise} \end{cases}$$

is a partial schedule for the routes $(R_M^*)_{M \in \overline{M}}$ since the machines in $\overline{M}$ do not process any jobs in $S'$. We show that $S'$ is complete with respect to $(R_M^*)_{M \in \overline{M}}$ in terms of Definition 3.2.

To this end, choose an arbitrary vertex $v \in V$ and an arbitrary machine $M_q \in \overline{M}$ with some unprocessed job $J_i \in T_M^v$. Then, $M_q \in \overline{M}$ since only machines in $\overline{M}$ have unprocessed jobs in $S'$. Moreover, $|T_M^v| \leq |M| - 1$, since at least the machine $M$ does not process job $i$. Finally $T_M^v = \emptyset$ since $M$ does not process any jobs in $S'$. Thus,

$$|T_M^v| \leq |M| - 1 \leq |M| - 1 - (|M| - 1) = \max(|J_i|, |M|)$$

and Theorem 3.4 shows how to complete a feasible schedule $S'$ for the routes $(R_M^*)_{M \in \overline{M}}$.

Remark 5.7. Lemma 5.6 gives an upper bound of $\max(|J_i|, |M|) + |M| - 1$ on the total amount of time that each machine stays in a vertex $v$ in an optimal schedule. Note that neither Example 3.3 nor Proposition 4.5 give such an upper bound: these show, in order to obtain a feasible schedule, it is sufficient that each machine stays in each vertex $v$ for at least $\max(|J_i|, |M|)$ time. They do not exclude that, in an optimal schedule, a machine might stay in a vertex significantly longer in order to enable other machines to process their jobs faster.

We are now ready to prove Proposition 5.4.

Proof of Proposition 5.4. (i) is trivially true by definition of Problem 5.3.

(ii) Let $S$ be a schedule for $I$ with respect to minimum-length machine routes $(R_M^*)_{M \in M}$. We choose $I := (I, (T, A, D), L)$ where $T$ is a pre-schedule and $c(H) + |J| \leq L \leq c(H) + |J| + K$, where $H$ is a minimum-cost Hamiltonian cycle for $G$ and $K$ is the total criticality (see Definition 4.4). We first show that the routes $(R_M^*)_{M \in M}$ comply with some pre-schedule $(T, A, D)$. Since, by Observation 4.1 and Proposition 4.5, the machine routes $(R_M^*)_{M \in M}$ have length $L \in [c(H) + |J|, c(H) + |J| + K]$, it follows that $S$ is a solution for some $(I, (T, A, D), L) \in I$. Thereafter, we analyze the cardinality of $I$. By Lemma 5.5, each route $R_M^* = ((a_i^0, w_i^0, b_i^0), v_i)_{v_i \in 1}$ of a machine $M_q \in M$ has $t_q \leq |M| + 2$ stays. Thus, by Definition 5.2(iii), they comply with the pre-schedule $T := ((I, (T, A, D), L), \sigma)_{\sigma}$ which has a pre-stay $(M_q, v_i, \sigma) = (M_q, w_i^0, b_i^0)$ if and only if $R_M^*$ has a stay $(a_i^0, w_i^0, b_i^0)$, where $T$ is sorted so that one has $a_i^0 < a_i^0$ for $i < j$ and so that one has $a_i^0 < a_i^0$ for $i < j$ with $q_i = q_j$. Such a sorting exists since $a_i^0 < a_i^0$ for $i < j$ and all machines $M_q \in M$. Now, let $K := I \setminus S$ has a pre-stay in a critical vertex $v_i$. By Lemma 5.6, we may assume that the routes $(R_M^*)_{M \in M}$ stay in a critical vertex at most $2|M| - 1$ units of time. Thus, by Definition 5.2(iv), the routes $(R_M^*)_{M \in M}$ comply with the length assignment $A : K_T \rightarrow [0, \ldots, 2|M| - 1], i \rightarrow b_i^0 - a_i^0$. It is trivial that the routes $(R_M^*)_{M \in M}$ comply with the displacement $D : K_T \rightarrow [0, \ldots, 2|M|], j \rightarrow \min(2|M|, a_i^0 - a_i^0)$, where $i \in K_T$ is the maximum number with $i < j$ for the smallest number $j \in K_T$, we define $D(j) := 0$, but any other choice would fit the purpose).

It remains to count the number of instances in $I$. We have $K + 1 \leq (|V| - 1)(|M| - 1) + |M| + 1$ choices for $L \in [c(H) + |J|, c(H) + |J| + |K| + K]$. For each pre-stay $(M_q, v_i, \sigma)$, there are $|M|$ choices for $M_q$, and $|V|$ choices for $v_i$. There is only one choice for $\sigma$; in the pre-stays for each machine $M_q$, $\sigma$ increases from 1 to at most $|V||M| + 2$ in steps of one by Definition 5.2(i). Thus, there are at most $|M| \cdot (|V||M| + 2)$ pre-stays in a pre-stay sequence $T$ and hence, at most $|M|^{|M|(|M|(|M| + 2)} \in 2^{O(|V||M| \log |V||M|)}$ pre-stay sequences. Moreover, this implies that, for each pre-stay sequence $T$, one has $|K_T| \leq |M| \cdot (|V||M| + 2)$. Thus, there are at most $2(|M|(|M|(|M|(|M| + 2) \in 2^{O(|V||M| \log |M|)}$ length assignments $A : K_T \rightarrow [0, \ldots, 2|M| - 1]$ and $(2|M| - 1)\log |M| \in 2^{O(|V||M| \log |M|)}$ displacements $D : K_T \rightarrow [0, \ldots, 2|M|]$.}

Having Proposition 5.4, for proving Theorem 5.1, it remains to solve Problem 5.3 in $2^{O(|V||M| \log |V||M|)} \cdot \poly(|V|)$ time since a shortest schedule for an instance $I$ of Routing Open Shop with unit processing times can be found by solving the instances $(I, (T, A, D), L) \in I$ for increasing $L$.

5.2. Computing routes and completing the schedule

In this section, we provide the last missing ingredient for our fixed-parameter algorithm for Routing Open Shop with unit processing times:

Proposition 5.8. Problem 5.3 can be solved in $2^{O(|V||M| \log |V||M|)} \cdot \poly(|V|)$ time.

By Proposition 5.4, this proves Theorem 5.1. The key to our algorithm for Proposition 5.8 is the following lemma.

Lemma 5.9. Let $(I, (T, A, D), L)$ be an instance of Problem 5.3 that has a solution. Then, for arbitrary routes $(R_M^*)_{M \in M}$ of length $L$ complying with $(T, A, D)$ and satisfying $|T_M^v| \geq |J_i|$, for each non-critical vertex $v$, in $V$,

(i) there is a partial schedule $S$ with respect to $(R_M^*)_{M \in M}$ such that $S(J_i, M_q) \neq \perp$ if and only if $L(J_i)$ is critical,

(ii) any such partial schedule is complete with respect to $(R_M^*)_{M \in M}$.

Proof. We first show (ii). Let $S$ be any partial schedule such that $S(J_i, M_q) \neq \perp$ if and only if $L(J_i)$ is critical. We show that $S$ is complete with respect to $(R_M^*)_{M \in M}$ in terms of Definition 3.2. For each critical vertex $v \in V$, each machine $M_q \in M$, and each job $J_i \in T_M^v \cap J_v$, (3.1) of Definition 3.2 is trivially satisfied since there is no such job: $J_M^v \cap J_v = \emptyset$. For each non-critical vertex $v \in V$, each machine $M_q \in M$, and each job $J_i \in T_M^v \cap J_v$, one has

$$|T_M^v \setminus (T_M^v \cup S_M) \geq |J_i| = \max(|J_i|, |M|).$$
We show how to construct a partial schedule with respect to the given routes \((R_M)_{M \in \mathcal{M}}\) of length \(L\), complying with the pre-stay sequence \(T = ((M_{q_i}, v_k, \sigma_i))_{i=1}^t\). Thus, for \(\mathcal{K}(p, \pi) := \{p < k \leq \pi \mid (M_{q_k}, v_k, \sigma_k)\) is a pre-stay of \(T\) in a critical vertex\), one has, by Definition 5.2(ii) and (vi),

\[
\alpha^*_r - \alpha^*_s = \sum_{k \in \mathcal{K}(p, \pi)} D(k) = \alpha^*_s - \alpha^*_r \tag{5.2}
\]

since both tours \((R_M)_{M \in \mathcal{M}}\) and \((R_M^*)_{M \in \mathcal{M}}\) comply with the displacement \(D\). By adding \(\alpha^*_r - \alpha^*_s\) to both sides of

\[
S(J_i, M_q) + \alpha^*_s - \alpha^*_r = S(J_i, M_q) + \alpha^*_s - \alpha^*_r,
\]

which is true by the definition of \(S\) from \(S^*\), one obtains

\[
S(J_i, M_q) + \alpha^*_r - \alpha^*_s \neq S(J_i, M_q) + \alpha^*_s - \alpha^*_r,
\]

and, therefore, \(S(J_i, M_q) \neq S(J_i, M_q)\) from (5.2).

Lemma 5.9 shows that, to solve Problem 5.3, it is sufficient to compute routes \((R_M)_{M \in \mathcal{M}}\) of length \(L\) that comply with a given pre-schedule \((T, A, D)\) and stay in each non-critical vertex \(v\) for at least \(|J_i|\) units of time. If no such routes are found, then the instance of Problem 5.3 has no schedule of length \(L\) since any feasible schedule has to spend at least \(|J_i|\) units of time in each vertex \(v\). If such routes are found, then, by Lemma 5.9(ii), a feasible schedule for the non-critical vertices can be computed using the schedule completion Theorem 3.4. For critical vertices, we use the following brute force approach:

**Lemma 5.10.** Let \((J, (T, A, D), L)\) be an instance of Problem 5.3 and \((R_M)_{M \in \mathcal{M}}\) be arbitrary routes complying with \((T, A, D)\). If there is a partial schedule \(S\) for \(I\) that satisfies Lemma 5.9(i), then we can find it in \(2^{O(|V||M|^2 \log |M|)} \cdot \text{poly}(|J|)\) time.

**Proof.** Observe that, in total, there are at most \(|V| \cdot |M|\) jobs in critical vertices. Thus, we determine \(S(J_i, M_q)\) for at most \(|V| \cdot |M|^2\) pairs \((J_i, M_q) \in \mathcal{J} \times \mathcal{M}\). By Lemma 5.6, each machine can process all of its jobs in a critical vertex staying there no longer than \(2|M| - 1\) units of time. Thus, for each of at most \(|V|\cdot |M|^2\) pairs \((J_i, M_q)\), we enumerate all possibilities of choosing \(S(J_i, M_q)\) among the smallest \(2|M| - 1\) numbers in \(T_{\Sigma_{i|J_i|}}\) and check each of them for feasibility. There are \((2|M| - 1)^{|V| \cdot |M|^2}\) possibilities to do so.

Finally, we compute the routes required by Lemma 5.9 by testing the feasibility of an integer linear program with \(|O(|M| \cdot (|V| \cdot |M| + 2))|\) variables and constraints, which, by Lenstra’s theorem, works in \(2^{O(|V| \cdot |M|^2 \log |M|)}\) time:

**Theorem 5.11 (Lenstra (1983); also Kannan (1987)).** A feasible solution to an integer linear program with \(p\) variables and \(m\) constraints is computable in \(p^{O(p)} \cdot \text{poly}(m)\) time, if such a feasible solution exists.

Any feasible schedule has to stay in each non-critical vertex \(v\) for at least \(|J_i|\) time. Thus, the following lemma together with Lemmas 5.9 and 5.10 and Theorem 3.4 completes the proof of Proposition 5.8 and, hence, of Theorem 5.1.
Lemma 5.12. Let \((I, (T, A, D), L)\) be an instance of Problem 5.3. In \(2^{O(|V| \log |V||M|)}\) time, one can compute routes \((R_m)_{m \in M}\) that have length \(L\), comply with \((T, A, D)\), and satisfy \(|T[R_m]| \geq |\mathcal{F}_v|\) for each non-critical vertex \(v \in V\), if such routes exist.

Proof. Denote the given pre-stay sequence as \(T = ((M_q, v_i, \sigma_j))_{i=1}^{\ell}\). For each machine \(M_q \in M\), let \(t_q = \max \{\sigma | (M_q, w, \sigma)\) is a pre-stay on \(T\}\). By Definition 5.2, \(t_q \leq |V||M| + 2\) for each machine \(M_q \in M\). We compute the routes \((R_m)_{m \in M}\) where \(R_{M_q} := ((a_i^q, b_i^q, d_i^q))_{i=1}^{\ell}\), as follows. For each pre-stay \((M_q, v_i, \sigma_j)\) on \(T\), we let \(a_i^q := v_i\). If \(w_i^q \neq v^{'}\) or \(v^{'} \neq w_i^q\) for some machine \(M_{q'}\), where \(v^{'}\) is the depot, then there is no solution and we answer “no” accordingly. Otherwise, the \(a_i^q\) and \(b_i^q\) for each machine \(M_q \in M\) and \(1 \leq k \leq t_q\) are at most \(2|V||M| + 2\) variables, which we determine using a feasible solution to an integer linear program. This, together with Theorem 5.11 directly yields the running time stated in Lemma 5.12.

Our linear program consists of the following constraints. We want each route to have length at most \(L\), that is, 

\[
b_i^q \leq L \quad \text{for each } M_q \in M.
\]

A route must have sufficient travel time between stays, that is, 

\[
b_i^q + c(v_i^q, v_{i+1}^q) \leq a_{i+1}^q \quad \text{for each } M_q \in M
\]

and \(1 \leq k \leq t_q - 1\).

Stays should have non-negative length, that is, 

\[
a_i^q \leq b_i^q \quad \text{for each } M_q \in M
\]

and \(1 \leq k \leq t_q\).

Each machine should stay in \(v \in V\) for at least \(|\mathcal{F}_v|\) time, that is 

\[
\sum_{1 \leq k \leq t_q, v_i^q = v} (b_i^q - a_i^q) \geq |\mathcal{F}_v| \quad \text{for each } M_q \in M \text{ and } v \in V.
\]

Stays must be ordered according to the pre-stay sequence \(T\), that is 

\[
a_i^q \leq a_j^q \quad \text{for pre-stays } (M_q, v_i, \sigma_i) \text{ and } (M_q, v_j, \sigma_j) \text{ with } i \leq j.
\]

Stays should adhere to the length assignment \(A\), that is 

\[
b_i^q - a_i^q = A(i) \quad \text{for each pre-stay } (M_q, v_i, \sigma_j) \text{ such that } v_i \text{ is critical.}
\]

Finally, the routes have to comply with the displacement \(D\). To formulate the constraint, let \(\mathcal{K} := \{i \leq s \mid v_i \text{ is critical}\}\) be the indices of pre-stays in critical vertices of \(T\). For any two pre-stays \((M_q, v_i, \sigma_i)\) and \((M_q, v_j, \sigma_j)\) such that \(i, j \in \mathcal{K}\) and \(k \notin \mathcal{K}\) for all \(k \in \{i + 1, \ldots, j - 1\}\), we want that 

\[
a_i^q \geq a_i^q + D(j) \quad \text{if } D(j) = 2|M|, \text{ and}
\]

\[
a_i^q = a_i^q + D(j) \quad \text{if } D(j) < 2|M|.
\]

\(\square\)

6. Conclusion

We have proved the schedule completion Theorem 3.4 and used it for a fixed-parameter algorithm for Routing Open Shop with unit processing times. Precisely, we used it to prove upper bounds on various parameters of optimal schedules. This suggests that Theorem 3.4 will be likewise beneficial for approximation algorithms. Indeed, our Section 4 makes first steps into this direction.

A natural direction for future research is determining the parameterized complexity of Routing Open Shop with unit processing times parameterized by the number \(|V|\) of vertices. Even the question whether the problem is polynomial-time solvable for constant \(|V|\) is open, yet we showed fixed-parameter tractability in the absence of critical vertices (Corollary 4.6). Finally, it would be desirable to find a fast polynomial-time algorithm for finding the coloring whose existence is witnessed by the theorem of Borodin et al. (1997) (Theorem 3.6).

References

A. Allahverdi, C. Ng, T. Cheng, and M. Y. Kovalyov. A survey of scheduling problems with setup times or costs. European Journal of Operational Research, 187(3):985–1032, 2008. doi: 10.1016/j.ejor.2006.06.060.

I. Averbakh, O. Berman, and I. Chernykh. A \(\mathcal{F}\)-approximation algorithm for the two-machine routing open-shop problem on a two-node network. European Journal of Operational Research, 166(1):3–24, 2005. doi: 10.1016/j.ejor.2003.06.050.

I. Averbakh, O. Berman, and I. Chernykh. The routing open-shop problem on a network: Complexity and approximation. European Journal of Operational Research, 173(2):531–539, 2006. doi: 10.1016/j.ejor.2005.01.034.

R. Bellman. Dynamic programming treatment of the Travelling Salesman Problem. Journal of the ACM, 9(1):61–63, 1962. doi: 10.1145/321105.321111.

R. van Bevern and A. V. Pyatkin. Completing partial schedules for open shop with unit processing times and routing. In Proceedings of the 11th International Computer Science Symposium in Russia (CSR’16), volume 9691 of Lecture Notes in Computer Science, pages 73–87. Springer, 2016. doi: 10.1007/978-3-319-34171-2_6.

R. van Bevern, R. Niedermeier, M. Sorge, and M. Weller. Complexity of arc routing problems. In Arc Routing: Problems, Methods, and Applications, MOS-SIAM Series on Optimization. SIAM, 2014. doi: 10.1137/111378161.

R. van Bevern, J. Chen, F. Hüffner, S. Kratsch, N. Talmon, and G. J. Woeginger. Approximability and parameterized complexity of multicolor by \(c\)-intervals. Information Processing Letters, 115(10):744–749, 2015a. doi: 10.1016/j.ipl.2015.03.004.

R. van Bevern, C. Komusiewicz, and M. Sorge. Approximation algorithms for mixed, windy, and capacitated arc routing problems. In Proceedings of the 15th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS’15), volume 48 of OpenAccess Series in Informatics (OASIcs), pages 130–143. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2015b. doi: 10.4230/OASIcs.ATMOS.2015.130.

R. van Bevern, M. Mnich, R. Niedermeier, and M. Weller. Interval scheduling and colorful independent sets. Journal of Scheduling, 18:449–469, 2015c. doi: 10.1007/s10951-014-0398-5.

R. van Bevern, R. Bredereck, L. Bulteau, C. Komusiewicz, N. Talmon, and G. J. Woeginger. Precedence-constrained scheduling problems parameterized by partial order width. In Proceedings of the 2016 International Conference on Discrete Optimization and Operations Research (DOOR’16), 2016a. To appear, arXiv:1605.00001.

R. van Bevern, R. Niedermeier, and O. Suchý. A parameterized complexity view on non-preemptively scheduling interval-constrained jobs: few machines, small looseness, and small slack. Journal of Scheduling, 16:610–629, 2016b. doi: 10.1007/s10951-016-0478-9.

H. L. Bodlaender and M. R. Fellows. W[2]-hardness of precedence constrained \(k\)-processor scheduling. Operations Research Letters, 18(2):93–97, 1995. doi: 10.1016/0167-6377(95)00031-9.
