Timing problems and algorithms: Time decisions for sequences of activities

Thibaut Vidal*
CIRRELT & Département d'informatique et de recherche opérationnelle,
Université de Montréal, Canada
& ICD-LOSI, Université de Technologie de Troyes, France
Thibaut.Vidal@cirrelt.ca

Teodor Gabriel Crainic
CIRRELT & Département de management et technologie, École des sciences de la gestion, UQAM,
Montréal, Canada

Michel Gendreau
CIRRELT & Département de mathématiques et génie industriel,
École Polytechnique, Montréal, Canada

Christian Prins
ICD-LOSI, Université de Technologie de Troyes, France

Abstract

Timing problems involve the choice of task execution dates within a predetermined processing sequence, and under various additional constraints or objectives such as time windows, time-dependent costs, or flexible processing times, among others. Their efficient resolution is critical in branch and bound and neighborhood search methods for vehicle routing, project and machine scheduling, as well as in various applications in network optimization, resource allocation and statistical inference. Timing related problems have been studied for years, yet research on this subject suffers from a lack of consensus, and most knowledge is scattered among operations research and applied mathematics domains. This article introduces a classification of timing problems and features, as well as a unifying multi-disciplinary analysis of timing algorithms. In relation to frequent application cases within branching schemes or neighborhood searches, the efficient resolution of series of similar timing subproblems is also analyzed. A dedicated formalism of re-optimization "by concatenation" is introduced to that extent. The knowledge developed through this analysis is valuable for modeling and algorithmic design, for a wide range of combinatorial optimization problems with time characteristics, including rich vehicle routing settings and emerging non-regular scheduling applications, among others.

Keywords: Optimization, Timing, Scheduling, Routing, Resource allocation, Statistical inference, Branch and bound, Neighborhood search.

* Corresponding author

1 Introduction

Time-related constraints and objectives appear in a variety of flavors within scheduling, project management, data transmission, routing, network optimization and numerous other fields. The related combinatorial optimization problems, e.g. vehicle routing or machine scheduling, often require the arrangement of *activities* under time requirements, such as tasks, visits to customers, production of objects, and so on. Several combined decisions are required: the *allocation* of resources to activities, activities *sequencing*, and finally the adjustment – *timing* – of activity execution dates, speed and idle time, within the chosen sequence. To solve these combinatorial optimization problems, most heuristic and exact approaches perform a search through a large number of sequence and resource allocation alternatives, and use repeatedly a timing algorithm to produce adequate execution dates, filter feasible solutions and evaluate costs. The timing solution method is thus called extensively, such that its complexity impacts dramatically the performance of the solution method, potentially making the difference between successful algorithmic approaches and failure.

Timing algorithms are the cornerstone of complex algorithms for difficult combinatorial optimization problems, but the literature dedicated to this subject remains is scarce and scattered. In fact, most developments on timing are inherent to a specific field, such as project planning, shortest path, routing, scheduling, and statistical inference, which bring into play, quite unexpectedly, the same formulations. Few relationships between domains have been actually exploited and, thus, close concepts and solution methods are independently developed within different formalisms, being rarely assessed in a more general context. In addition, real-life settings bring forth a large number of challenging timing variants with different constraints and objectives, such as target execution dates, penalized idle time, (possibly multiple) time windows on activities, penalized lateness and earliness, speed decisions, time-dependent activity durations and costs, congestion, learning issues, and so on. Even if efficient timing algorithms have been designed for some of these characteristics taken separately, problems involving combinations of characteristics become much more complex, and there is generally no systematic way to extend concepts developed for the separate problems into a methodology for the new ones.

To address these challenges, this paper contributes to the *timing field*, by means of a multidisciplinary review and analysis of timing features, problems, and algorithmic approaches. A large assortment of problems, often treated independently in the literature under various names, are identified and classified in relation to their structure. Successful solution methods and solving concepts are inventoried and analyzed. In the most noticeable cases, this analysis led to identify more than 26 algorithms from different research fields as similar implementations of three main general approaches. Not only does this review gather the keys for a stand-alone resolution of a large variety of timing problems, but it also analyzes the efficient resolution of timing problems within the context of global search methods, e.g. neighborhood-based heuristics and branch-and-bound-based approaches for rich vehicle routing and scheduling problems. For these applications, managing global information through the successive resolution of similar timing instances can lead to dramatic reductions of the overall computational effort. To this extent, a *re-optimization framework* is introduced in the second part of this paper. The body of knowledge developed in this paper is critical for both modeling work and algorithmic design, enabling to point out relationships between problems and their respective complexities. A portfolio of state-of-the-art timing algorithms is identified, which will prove useful to build more generalist solvers for many variants of difficult combinatorial optimization problems. To our knowledge, no such unifying review and methodological analysis of this rich body of issues has been performed in the past.

The remainder of this paper is organized as follows: Section 2 formally defines timing problems, while Section 3 presents examples of applications. Section 4 provides a detailed classification of the main timing features encountered in the literature as well as notations. Our methodological analysis of timing problems and their *independent resolution* is then organized in Sections 5 and 6 relatively to the previous classification. Section 7 finally introduces a *re-optimization framework* that encompasses

state-of-the-art approaches for solving *series* of related timing instances. Section 8 highlights a number of challenging avenues of research in the timing field and presents the conclusions of this paper.

2 Problem statement

In this paper, the term *activities* is used to represent, independently of the field of application, elementary operations that must be managed. The term *date* always stands for a point in time, whereas the words *duration* or *time* are employed for relative values (e.g., processing time). Without loss of generality, objective minimization is considered. The notation a^+ stands for $\max\{a, 0\}$.

Definition 1 (General timing problem)**.** *Let $A = (a_1, \ldots, a_n)$ be a sequence of n activities with processing times $p_1, \ldots, p_n$. The execution dates of these activities $\mathbf{t} = (t_1, \ldots, t_n)$ are required to follow a total order with respect to the subscripts, such that $t_i + p_i \leq t_{i+1}$ for $i \in \{1, \ldots, n\text{-}1\}$. Additional problem* features *F^x, for $x \in \{1, \ldots, m\}$, provide the means to address particular settings with either a role as* objective*, $F^x \in \mathcal{F}^{\text{OBJ}}$, or as* constraint *$F^x \in \mathcal{F}^{\text{CONS}}$. Any feature F^x is characterized by a set of m_x functions $f_y^x(\mathbf{t})$ for $y \in \{1, \ldots, m_x\}$. The general timing problem aims to find a feasible timing solution $\mathbf{t}$, respecting order constraints (Equation 2), features constraints (Equation 3), and minimizing the weighted sum of contributions from all feature objectives (Equation 1).*

$$\min_{\mathbf{t}=(t_1,\ldots,t_n)\in\Re^{n+}} \quad \sum_{F^x\in\mathcal{F}^{\text{OBJ}}} \alpha_x \sum_{1\leq y\leq m_x} f_y^x(\mathbf{t}) \tag{1}$$

$$s.t. \quad t_i + p_i \leq t_{i+1} \qquad 1 \leq i < n \tag{2}$$

$$f_y^x(\mathbf{t}) \leq 0 \qquad F^x \in \mathcal{F}^{\text{CONS}}\ ,\ 1 \leq y \leq m_x \tag{3}$$

The feature *deadlines* D, for example, involves a latest execution date d_i for each activity i, and characteristic functions $f_i^D(\mathbf{t}) = (t_i - d_i)^+$ for $i \in \{1, \ldots, n\}$. When D takes the role of a *constraint*, $f_i^D(\mathbf{t}) = (t_i - d_i)^+ \leq 0 \Leftrightarrow t_i \leq d_i$ yields the standard formulation of deadlines, while a role as objective leads to standard tardiness optimization criteria.

Timing problems can be viewed as shifting activity execution dates on a single resource, introducing idle time and optimizing activity speed, depending upon the features, without changing the processing order. Most basic versions of timing are simple to address, while various features arising from application cases can lead to dramatic increases in problem difficulty. It must also be noted that features have been defined independently from their role as constraint or objective for two main reasons. First, many algorithms are concerned with the effective calculation of some quantities, like total duration for example, which enables related constraints or objectives to be tackled in the same way. Secondly, since constraints can be transformed into objectives by Lagrangian relaxation, it is sometimes artificial to discriminate problems involving a given feature either as constraint or objective. Our study will thus be targeted on features, independently of their role, the latter being specified only when relevant to the method. Finally, in the scheduling domain, some constraints and objectives, such as due dates, are based on activity completion dates $C_i = t_i + p_i$. Without loss of generality, these problems are reformulated to involve only execution dates.

To emphasize the relations with practical problem settings, Section 3 details major problems in the fields of operations research and applied mathematics leading to underlying timing structures.

3 Timing issues and major application fields

Production and project scheduling. The development of just-in-time policies leads to challenging non-regular scheduling settings for which earliness or idle times are a major concern. In the earliness and tardiness (E/T) scheduling problem, a sequence of activities $(a_1, \ldots, a_n)$ is given with target

execution dates d_i and processing times p_i, as well as penalty factors for earliness ϵ_i and tardiness τ_i. The goal is to determine the sequence of activities and their execution dates on a single machine, such that linear penalties incurred for early or late processing are minimized. This scheduling problem is NP-hard in the strong sense [60] and most recent resolution methods consider branch and bound, neighborhood search or other metaheuristic frameworks working on the activity sequence [5]. For every sequence explored during the search, a timing algorithm is applied to compute the activity execution dates and thus the sequence cost. The related problem is formulated in Equations (4-5).

$$\min_{(t_1,\ldots,t_n)\in\Re^{n+}} \quad \sum_{i=1}^{n}\{\epsilon_i(d_i - t_i)^+ + \tau_i(t_i - d_i)^+\} \tag{4}$$

$$s.t. \quad t_i + p_i \leq t_{i+1} \qquad 1 \leq i < n \tag{5}$$

This timing problem is known to be solvable in $O(n \log n)$ (Sections 5.3 and 5.4). Yet, as the timing resolution is the main bottleneck for most (E/T) scheduling approaches, extensive research has been conducted to solve series of timing instances more efficiently within neighborhood searches. The use of global information through the search leads to timing "re-optimization" procedures working in amortized $O(\log n)$ complexity, and even $O(1)$ for some particular cases, as described in Section 7.

Network optimization and vehicle routing. Timing subproblems are also frequently encountered in network optimization settings, e.g., resource-constrained shortest paths, delivery-man and minimum latency, vehicle routing and scheduling [40, 42, 146]. Thus, for example, the vehicle routing problem with time windows (VRPTW) consists in designing vehicle itineraries to service a set of geographically scattered customers within allowed time intervals. This problem has been the focus of a significant research effort focused for a large part on heuristic methods [18, 19, 61]. In particular, it is common to consider solutions with penalized time-constraint violations to enhance the search [148]. Different relaxation schemes can be applied, with lateness (Taillard et al. [140]), lateness and earliness (Ibaraki et al. [86, 87]), "returns in time" (Nagata et al. [109]), or speed-increase (Vidal et al. [148]), leading to different timing sub-problems for producing route schedules and evaluating penalties. Yet, the efficient resolution of these sub-problems is critical, since it determines the complexity of the neighborhood-search procedures, which are the bottleneck of many current metaheuristics.

One also observes a recent important focus on "richer" VRPs [68, 146], which explicitly take into account various combined constraints and objectives issued from application cases. These complex combinatorial optimization problems frequently involve temporal considerations, time-dependent travel speed, crew costs, customer requirements in terms of visit times, employee breaks and duty times, learning or fatigue effects, fair repartition of working time among employees, and so on. Such characteristics must be directly managed within route evaluations in heuristics and exact methods, thus leading to a large variety of timing subproblems.

Energy optimization. Norstad et al. [111] introduce a ship routing problem with convex speed optimization, which presents two interlaced issues: the design of a ship itinerary, and the optimization of arrival dates and speed to reduce fuel consumption. For a fixed sequence of visits, the latter subproblem is formulated in Equations (6-9).

$$\min_{\mathbf{t},\mathbf{v}} \quad \sum_{i=1}^{n-1} d_{i,i+1}\ c(v_{i,i+1}) \tag{6}$$

$$s.t. \quad t_i + p_i + d_{i,i+1}/v_{i,i+1} \leq t_{i+1} \qquad 1 \leq i \leq n-1 \tag{7}$$

$$r_i \leq t_i \leq d_i \qquad 1 \leq i \leq n \tag{8}$$

$$v_{min} \leq v_{i,i+1} \leq v_{max} \qquad 1 \leq i \leq n-1 \tag{9}$$

The decision variables are the travel speeds $v_{i,i+1}$ for $i \in \{1, \ldots, n-1\}$ for each port-to-port leg, and the arrival dates at ports t_i for $i \in \{1, \ldots, n\}$. The objective is to minimize the fuel consumption on all trips, while respecting time-window constraints $[r_i, d_i]$ on arrival dates, and maintaining the speed in a feasible range $[v_{min}, v_{max}]$. The convex function $c(v)$ describes the energy consumption per mile as a function of speed. Let v_{opt} be the minimum of $c(v)$, let $d_{i,i+1}$ represent the leg distances and p_i stand for processing times at ports. Equations (7-9) ensure that port arrival and departure dates are consistent with leg speeds, that time windows at port arrivals are respected, and finally that speeds are within a feasible range.

This problem can be reformulated to rely exclusively on arrival dates by defining an extended cost/speed function $\hat{c}(v)$, which accounts for the fact that waiting times can be used in case of sub-optimal low speed values (Equations 10-13).

$$\min_{\mathbf{t}} \quad \sum_{i=1}^{n-1} d_{i,i+1} \hat{c}\left(\frac{d_{i,i+1}}{t_{i+1} - t_i}\right) \tag{10}$$

$$\text{s.t.} \quad t_i + p_i + \frac{d_{i,i+1}}{v_{max}} \leq t_{i+1} \qquad 1 \leq i \leq n-1 \tag{11}$$

$$r_i \leq t_i \leq d_i \qquad 1 \leq i \leq n \tag{12}$$

$$\text{with} \quad \hat{c}(v) = \begin{cases} c(v_{opt}) & \text{if } v \leq v_{opt} \\ c(v) & \text{otherwise} \end{cases} \tag{13}$$

The latter model falls into the category of timing problems. It involves time-window features characterized by functions $f_i(\mathbf{t}) = (t_i - d_i)^+ + (r_i - t_i)^+$ with a role as constraints, as well as flexible processing times characterized by convex functions $f_i(\mathbf{t}) = c_i(t_{i+1} - t_i)$ in the objective, such that $c_i(\Delta t_i) = d_{i,i+1}\hat{c}(d_{i,i+1}/\Delta t_i)$. Hvattum et al. [84] and Norstad et al. [111] introduced a strongly polynomial Recursive Smoothing Algorithm (RSA) to solve the previous timing problem with a worst case complexity of $O(n^2)$. A similar algorithm was then used in the context of a vehicle routing problem with CO_2 minimization (Demir et al. [36]). Other timing algorithms and re-optimization procedures are known for these settings (Sections 6.3 and 7.5.5).

Statistical Inference. The isotonic regression problem under a total order (IRC) constitutes an intensively studied particular case of our models. Given a vector $\mathbf{N} = (N_1, \ldots, N_n)$ of n real numbers, IRC seeks a vector of non-decreasing values $\mathbf{t} = (t_1, \ldots, t_n)$ *as close as possible* to $\mathbf{N}$ according to a distance metric $\| \ \|$ (generally the Euclidean distance), as in Equations (14-15).

$$\min_{\mathbf{t}=(t_1,\ldots,t_n)} \quad \|\mathbf{t} - \mathbf{N}\| \tag{14}$$

$$\text{s.t.} \quad t_i \leq t_{i+1} \qquad 1 \leq i < n \tag{15}$$

As underlined by the seminal books of Barlow et al. [8] and Robertson et al. [126], IRC is the key to performing many restricted maximum likelihood estimates in statistics, and is linked with various applications such as image processing and data analysis. It appears here as a timing problem with separable convex costs, similar to those encountered when solving vehicle routing problems with time windows or (E/T) scheduling settings.

Other applications. Timing formulations arise in various other contexts. For example, the nested resource allocation problem [47, 77, 149] is equivalent to a timing problem with flexible processing times and deadlines on activity completion dates. Different names are also used, e.g., *projection onto order simplexes* in Grotzinger and Witzgall [65]. Finally, some timing formulations constitute special cases of several convex optimization problems with underlying network structures [2, 76].

4 Features : classification and reductions

This section introduces a classification of the main timing features in the literature, and levers notations for the related problems. Reduction relationships between features are then investigated.

4.1 Classification and notations

The features are here classified relatively to the structure of their characteristic functions. We rely to that extent on a *feature dimension* measure ξ, which illustrates the links that a feature creates between decision variables.

Definition 2 (Feature dimension)**.** *The dimension $\xi(F^x)$ of a feature F^x is defined as the maximum number of variables involved together in any characteristic function $f^x_y(\mathbf{t})$ for $y \in \{1, \ldots, m_x\}$.*

Table 1 displays the most common features in the literature relatively to their dimension ξ. The first column provides an abbreviation for each feature. The next columns describe the parameters, characteristic functions and dimensions of these features. Finally, we report the most frequent roles of each feature in the literature.

Table 1: Classification of timing features and notations

Symbol	**Parameters**	**Char. functions**	ξ	**Most frequent roles**
C	Deadline t_{max} on last activity	$f(\mathbf{t}) = (t_n - t_{max})^+$	1	Deadline on last activity, lateness of last activity, makespan
W	Weights w_i	$f_i(\mathbf{t}) = w_i t_i$	1	Weighted execution dates
D	Deadlines d_i	$f_i(\mathbf{t}) = (t_i - d_i)^+$	1	Deadline constraints, tardiness
R	Release dates r_i	$f_i(\mathbf{t}) = (r_i - t_i)^+$	1	Release-date constraints, earliness.
TW	Time windows $TW_i = [r_i, d_i]$	$f_i(\mathbf{t}) = (t_i - d_i)^+ + (r_i - t_i)^+$	1	Time-window constraints, soft time windows.
MTW	Multiple TW $MTW_i = \cup[r_{ik}, d_{ik}]$	$f_i(\mathbf{t}) = \min_k [(t_i - d_{ik})^+ + (r_{ik} - t_i)^+]$	1	Multiple time-window constraints
$\Sigma c_i^{\text{CVX}}(t_i)$	Convex $c_i^{\text{CVX}}(t_i)$	$f_i(\mathbf{t}) = c_i^{\text{CVX}}(t_i)$	1	Separable convex objectives
$\Sigma c_i(t_i)$	General $c_i(t)$	$f_i(\mathbf{t}) = c_i(t_i)$	1	Separable objectives, time-dependent activity costs
DUR	Total dur. δ_{max}	$f(\mathbf{t}) = (t_n - \delta_{max} - t_1)^+$	2	Duration or overall idle time
NWT	No wait	$f_i(\mathbf{t}) = (t_{i+1} - p_i - t_i)^+$	2	No wait constraints, min idle time
IDL	Idle time ι_i	$f_i(\mathbf{t}) = (t_{i+1} - p_i - \iota_i - t_i)^+$	2	Limited idle time by activity, min idle time excess
$P(t)$	Time-dependent proc. times $p_i(t_i)$	$f_i(\mathbf{t}) = (t_i + p_i(t_i) - t_{i+1})^+$	2	Processing-time constraints, min activities overlap
TL	Time-lags δ_{ij}	$f_i(\mathbf{t}) = (t_j - \delta_{ij} - t_i)^+$	2	Min excess with respect to time-lags
$\Sigma c_i(\Delta t_i)$	General $c_i(t)$	$f_i(\mathbf{t}) = c_i(t_{i+1} - t_i)$	2	Separable functions of durations between successive activities, flex. processing times
$\Sigma c_i(t_i, t_{i+1})$	General $c_i(t, t')$	$f_i(\mathbf{t}) = c_i(t_i, t_{i+1})$	2	Separable objectives or constraints by successive pairs of variables
$\Sigma c_{ij}(t_i, t_j)$	General $c_{ij}(t, t')$	$f_{ij}(\mathbf{t}) = c_i(t_i, t_j)$	2	Separable objectives or constraints by any pairs of variables
$c(\mathbf{t})$	General $c(t)$	$f(\mathbf{t}) = c(\mathbf{t})$	–	Any feature

Most of the features presented in Table 1 are well-known in the scheduling or vehicle routing literature. Features D, C, and W, can be qualified as *regular*, as they involve non-decreasing characteristic

functions $f_y^x(\mathbf{t})$. The set of *active schedules "such that no operation can be made to start sooner by permissible left shifting"* (Giffler and Thompson [62]) is dominating for regular features. Solving the timing problem is then straightforward by means of a minimum idle time policy (Section 5.1). However, these regular features present notably different behaviors with regards to re-optimization, thus motivating a detailed study. Other features from Table 1 are *non-regular*. They lead to more complex timing problems for which the insertion of idle time can improve the objective or the satisfaction of constraints.

Single- and two-dimensional features are directly linked to physical quantities, execution dates and durations, respectively. As seen in Table 1, such features are frequently encountered in timing formulations. Higher-dimension features are more unusual in the literature, and can, by definition, lead to a wide range of difficult problems. Indeed, any mathematical program can be viewed as a combination of unary and binary mathematical operators of the form $x = f(y, z)$, and thus can be reformulated with constraints and objectives separable in groups of three variables. Hence, any problem on continuous totally ordered variables can be viewed as a timing problem with three-dimensional features. A reasonable delineation for timing problems is to consider, as in the present paper, only applications and problems presenting explicitly the aspect of an activity sequence.

We introduce a notation specifying for each problem the features considered, as well as information regarding their role. Each problem is tagged as a two-component string $\{O|C\}$, where O is a list of features included in the objective and C is a list of features included as constraints. Separating features in the field O with a comma indicates a weighted sum of objectives. The sign $\cup$ is used for multi-objective problems and the sign $>$ indicates an order of priority. Particular parameter characteristics are reported in parentheses after the feature symbol. For example, problems with common deadlines can be marked with $(d_i = d)$, null processing times as $(p_i = 0)$, and so on.

To illustrate, consider the problem of speed optimization of Section 3. This problem presents a separable and convex objective as a function of durations between successive activities, along with time-window constraints. It can thus be categorized as $\{\Sigma c^{\text{CVX}}(\Delta t)|TW\}$. The (E/T) timing problem presents linear penalties around a target execution date. These penalties can be assimilated to relaxed simultaneous release dates and deadlines, leading to the notation $\{R, D(r_i = d_i)|\emptyset\}$. Finally, the vehicle routing literature includes problem settings with a hierarchical objective aiming first to minimize the amount of time-window violations, then duration excess, and finally time-lag violations [12, 29]. Such a problem setting can be characterized as $\{TW > D > TL|\emptyset\}$.

4.2 Feature reductions

We use reduction relationships to illustrate the level of generality and complexity of timing features. A rich body of polynomial reductions has been developed in the scheduling literature. Most timing problems, however, are polynomially solvable, and the use of polynomial reduction relationships leads to consider most problems in the same class of equivalence. We thus seek stronger reduction properties to distinguish them. We also aim to build relationships between features instead of complete problems, leading to the following definition of feature reductions:

Definition 3 (Reducibility among timing features)**.** *A feature F is said to be reducible to feature F' if any timing problem $\mathcal{T}$ involving F and other features $\{F^1, \ldots, F^k\}$ admits a linear many-one reduction to a timing problem $\mathcal{T}'$ involving $F' \cup \{F^1, \ldots, F^k\}$.*

An overview of feature reductions is given in Figure 1, where an arrow from feature F^i to F^j indicates that feature F^i can be reduced to F^j. Four different categories of features are identified by different shades of gray. On the left, we present features involving at most one decision variable (the first part of Table 1) and separable costs. Progressing to the right, the next gray shade represents two-dimensional features that involve only pairs of consecutive activities, then features involving any

pair of activities, and finally other features. We also demarcate the area of "NP-hard" features, which alone are sufficient to lead to NP-hard timing problems.

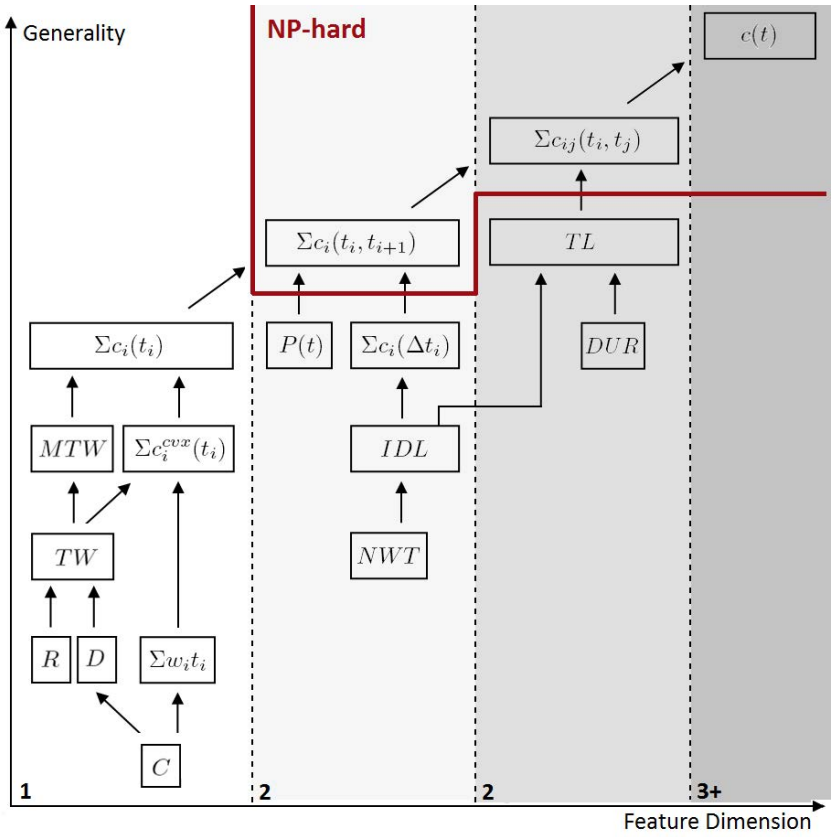


Figure 1: Hierarchy of timing features

The hierarchy of reductions presented in Figure 1 gives an indication on the level of generality of the features. Some features, such as $\Sigma c_i^{\text{CVX}}(t_i)$, $\Sigma c_i(t_i)$, $\Sigma c_i(\Delta t_i)$, and TL, generalize many other features while remaining polynomially solvable. An algorithm addressing such general features can tackle many problems, while specialized algorithms for simpler combinations of features may be more efficient. Both specialized and general algorithms are critical for practical applications. Thus, this methodological review is ordered by increasing generality, starting with the most simple cases of single-dimensional features in Section 5, and following with two-dimensional features in Section 6.

5 Single-dimensional features

Problems with single-dimensional features are analyzed according to their difficulty and generality, starting with simple regular features, following with time-window TW features, separable convex costs $\Sigma c_i^{\text{CVX}}(t_i)$ and, finally, general separable costs $\Sigma c_i(t_i)$. The latter feature encompasses multiple time windows MTW and generalizes all problems in this category.

5.1 Makespan, deadlines and weighted execution dates

Maximum execution dates C, deadlines D, and weighted execution dates W features lead to well-documented objectives in the scheduling literature, aiming to minimize makespan, tardiness, lateness or weighted completion time among others [64, 121]. W as an objective also arises in various routing

settings, such as the delivery-man problem [55], the minimum latency problem [17], and the cumulative TSP or VRP [16, 110], where the goal is to service a set of customer as early as possible. These features are *regular*, as any backward shift of execution date is beneficial for both feasibility and objective value. A very simple algorithm follows, which will be referred to as the "minimum idle time policy": *For each activity a_i of A in the sequence order, schedule a_i at its earliest possible execution date. If a_i cannot be scheduled, declare problem infeasibility and stop. If all activities have been successfully scheduled, declare problem feasibility.* An optimal solution is thus retrieved in n searches of the earliest feasible execution date, leading to a $O(n)$ complexity.

5.2 Release dates and time windows

Release-date and time-window features appear frequently in vehicle routing and scheduling applications. Time-window features generalize release dates R and deadlines D, as any release date r_i or deadline d_i can be transformed into a time window with an infinite value on the right $[r_i, +\infty]$ or the left $[-\infty, d_i]$. Two main issues are often considered regarding these features. The first involves stating on the feasibility of a sequence of activities under time-window constraints, whereas the second problem involves the minimization of infeasibility with respect to the time windows, and thus involves characteristic functions $f_i(\mathbf{t}) = (t_i - d_i)^+ + (r_i - t_i)^+$ in the objective.

Feasibility problem. Solving the feasibility problem $\{\emptyset|TW\}$ is straightforward, as the minimum idle time policy, presented in Section 5.1, is dominating in this respect. For a sequence of n activities $(a_1, \ldots, a_n)$, the algorithm starts with $t_1 = r_1$, then chooses each subsequent activity execution date to minimize idle time: $t_{i+1} = \max(t_i + p_i, r_{i+1})$. Hence, feasibility can be checked in $O(n)$ *from scratch*. Yet, more efficient feasibility checking procedures are available to solve *series* of timing instances in local-search context (Section 7).

Infeasibility minimization. Many real-case applications allow lateness or earliness, the so-called *soft* time-window settings, as a way to gain flexibility. Several contributions, such as Taillard et al. [140] and Cordeau et al. [30], focus on the problem $\{D|R\}$, where late activities are allowed with penalties, but not early activities. This case falls within the scope of regular features (Section 5.1), and choosing for each activity the earliest execution date is optimal. The problem can thus be solved with linear complexity $O(n)$.

However, when early activities are allowed, as in $\{TW|\emptyset\}$ [5, 6, 60, 86, 100], the objective function is no longer non-decreasing. Supposing that activity a_i is finished earlier than the beginning of the time-window of a_{i+1}, a choice must be made whether to insert idle time to reach a_{i+1}, or pay a penalty to better satisfy the time windows of remaining activities. The resulting timing problem becomes more complex. As an example, Appendix A shows that the problem of minimizing the number of time-window infeasibilities $\{TW(unit)|\emptyset\}$ generalizes the Longest Increasing Subsequence Problem (LISP). LISP admits a computational lower bound of $\Omega(n \log n)$ in the comparison tree model [59]. Sections 5.3 and 5.4 provide efficient algorithms to address these problems, leading to an $O(n \log n)$ algorithm for soft time-window relaxations.

5.3 Separable convex costs

Separable convex costs Σc_i^{CVX} include a wide range of problem settings as particular cases. The feature TW, and thus R, D, and C, can be reduced to $\Sigma c_i^{\text{CVX}}(t_i)$ as any time-window constraint can be formulated as a piecewise convex cost by associating arbitrary large costs to both sides of the feasibility interval. This feature also encompasses various other settings such as earliness-tardiness scheduling [5], isotonic regression problems with respect to a total order [8, 126], extensions of team

orienteering problems [52] in which the profit value can decrease with time [50], various convex penalty functions for time-window infeasibility [86, 89, 133, 134] and time-dependent convex processing costs [139], among others. The timing problem $\{\Sigma c_i^{\text{CVX}}(t_i)|\emptyset\}$ is formulated in Equations (16-17). Functions $c_i^{\text{CVX}}(t_i)$ are supposed to take infinite value for $t_i < 0$.

$$\min_{(t_1,\ldots,t_n)\in\Re^n} \quad \sum_{i=1}^{n} c_i^{\text{CVX}}(t_i) \tag{16}$$

$$s.t. \quad t_i + p_i \leq t_{i+1} \qquad 1 \leq i < n \tag{17}$$

Many methods have been proposed for this setting. We study, in this subsection, approaches specially designed for separable convex cost functions. Dynamic programming based algorithms, relying on fundamentally different concepts, are grouped in Section 5.4.

We base our analysis of algorithms for $\{\Sigma c_i^{\text{CVX}}(t_i)|\emptyset\}$ on a set of optimality conditions using the active set formalism of Best and Chakravarti [14], Chakravarti [23] and Best et al. [15]. The necessary and sufficient conditions provided below are more general than those previously developed in the literature, being applicable to any set of proper convex cost functions, including non-smooth cases.

Definition 4 (Blocks)**.** *A block B is defined as a sequence of activities $(a_{B(1)},\ldots,a_{B(|B|)})$ processed consecutively such that $t_i+p_i = t_{i+1}$ for all $i \in \{B(1),\ldots,B(|B|)-1\}$. For $k \in \{B(1),\ldots,B(|B|)-1\}$, we also define the prefix block $B^k = (a_{B(1)},\ldots,a_k)$. Let p_{ij} for $1 \leq i \leq j \leq n$ be the cumulative processing duration of activities $(a_i,\ldots,a_j)$. The execution cost C_B of a block B as a function of its first activity execution date $t_{B(1)}$ is given in Equation (18).*

$$C_B(t_{B(1)}) \quad = \quad c_{B(1)}(t_{B(1)}) + \sum_{i=B(1)+1}^{B(|B|)} c_i(t_{B(1)} + p_{B(1),i-1}) \tag{18}$$

When the costs are proper convex functions, the set of execution dates for the first activity with minimum block execution cost is an interval $[T_B^{-*}, T_B^{+*}]$.

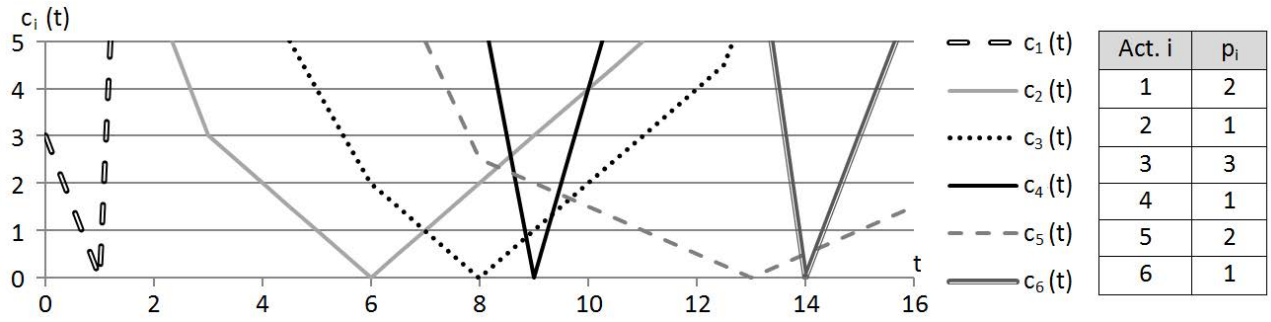
The following necessary and sufficient optimality conditions are obtained (c.f. Appendix B). Conditions 2 and 3 are direct consequences of the primal and the dual feasibility, respectively.

Theorem 1. *Let costs $c_i(t_i)$ for $i \in \{1,\ldots,n\}$ be proper convex, possibly non-smooth, functions. A solution $\mathbf{t}^* = (t_1^*,\ldots,t_n^*)$ of $\{\Sigma c_i^{\text{CVX}}(t_i)|\emptyset\}$ is optimal if and only if there exists blocks $(B_1,\ldots,B_m)$ such that the three following conditions are satisfied:*

1. *Blocks are optimally placed, $t^*_{B_i(1)} \in [T^{-*}_{B_i}, T^{+*}_{B_i}]$ for each block B_i;*
2. *Blocks are strictly spaced, $t^*_{B_i(1)} + p_{B_i(1),B_i(|B_i|)} < t^*_{B_{i+1}(1)}$ for each pair of blocks (B_i, B_{i+1});*
3. *Blocks are consistent, $T^{+*}_{B_i^k} \geq t^*_{B_i(1)}$ for each block B_i and prefix block B_i^k.*

Surveying the literature, we distinguish two main categories of methods: those who maintain primal feasibility, and those who maintain dual feasibility. These algorithms are issued from various domains. In the case of isotonic regression in particular, only precedence constraints among decision variables are considered ($p_i = 0$ for all i), yet the associated methods can be extended to solve problems with processing times with only minor modifications. We illustrate all algorithms on a simple problem, for which the cost functions and the processing times are given in Figure 2.

Primal methods. A first category of methods is based on respecting the primal feasibility conditions and iteratively restoring the dual conditions. The first method of this kind, called *Minimum Lower*



Act. i	p_i
1	2
2	1
3	3
4	1
5	2
6	1

Figure 2: Illustrative example with six activities: cost functions and durations

Set (*MLS*) algorithm, has been proposed by Brunk [21] for isotonic regression problems. The MLS algorithm starts with a single big block, then iteratively finds for each block B the biggest prefix block B^k violating dual conditions. If no such violation is found, this block is optimal, otherwise the current block is split in two and the procedure is recursively called on each sub-block until no dual conditions violation may be found. The algorithm can be implemented in $O(n^2)$ unimodal function minimizations.

Later on, Best and Chakravarti [14] introduced a primal feasible algorithm for IRC in $O(n)$ unimodal function minimizations. Again, activities are sequentially examined in each block to find the first violation of dual conditions (and not the most important violation). If such a violation exists, the block under consideration is split at this place. The leftmost block has an earlier optimal starting date, and thus can possibly be merged with one or several previously scheduled blocks to reach an optimal execution date. In the presence of quadratic costs, a closed form exists for the function minimums, and the complexity of this algorithm becomes $O(n)$ elementary operations.

The method of Garey et al. [60], originally designed for (E/T) scheduling, iterates on activities in the sequence order and yields at any step i an optimal solution to the subproblem containing only the first i activities. Each new activity is inserted at the end of the schedule, to be left-shifted and possibly merged with previous activity blocks until no improvement may be achieved. The algorithm can be implemented in $O(n \log n)$ for the case of (E/T) scheduling.

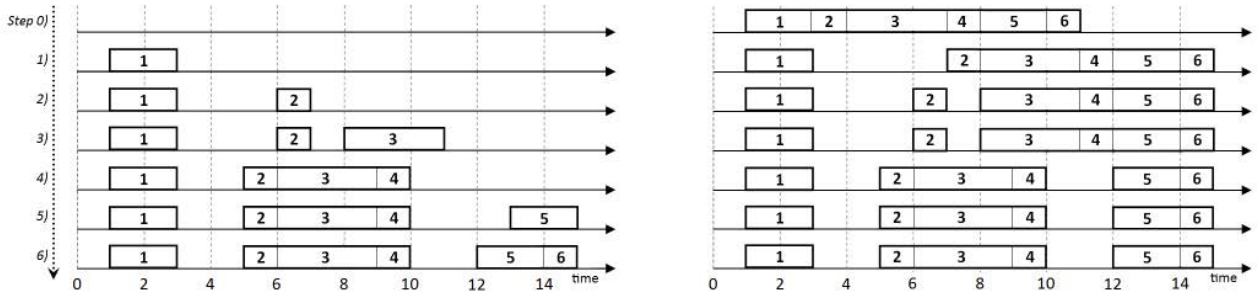


Figure 3: Comparison between Garey et al. [60] algorithm (left part of the figure), and Best and Chakravarti [14] algorithm (right part of the figure)

Figure 3 illustrates the algorithms of Best and Chakravarti [14] and Garey et al. [60] on the example of Figure 2. The problem is solved in six steps, illustrated from top to bottom along with the incumbent solutions, representing activities as rectangles with a length proportional to the processing time. The activity blocks in presence are very similar. These two algorithms can be viewed as two variations of the same underlying primal feasible method, with the exception that Garey et al. [60]

considers non-inserted activities as non-existing in the current solution, whereas Best and Chakravarti [14] maintains these non-scheduled activities in one final block which does not respect Condition 3.

The method of Garey et al. [60] was extended by Lee and Choi [101] and Pan and Shi [115] to address (E/T) scheduling problems with distinct penalty weights for earliness and tardiness, that is $\{D, R(d_i = r_i)|\emptyset\}$, in $O(n \log n)$ elementary operations. Szwarc and Mukhopadhyay [138] and Feng and Lau [53] also proposed to identify the tasks that are necessarily processed without idle time (in the same block) before solving. Chrétienne and Sourd [26] applied the algorithm to project scheduling with piecewise convex cost functions, and Hendel and Sourd [75] to timing problems with convex piecewise linear or quadratic costs. These algorithms work in a linear number of unimodal function minimizations, but differ in terms of the data structures used to represent the functions and thus on the complexity of the function minimizations. When the cost functions are Piecewise Linear (PiL), the method of Hendel and Sourd [75] attains a complexity of $O(\varphi_c \log n)$, where φ_c is the total number of pieces in the activity cost functions of the sequence. Finally, Davis and Kanet [34] proposed another primal method for (E/T) scheduling similar to Garey et al. [60], and generalized to PiL convex costs by Wan and Yen [151]. Activities are iteratively added to the solution in reverse sequence order. Each activity is scheduled at date 0, and then shifted onwards (while possibly merging blocks), until no improvement can be achieved.

Dual feasible methods. Simultaneously with the MLS algorithm, another seminal method for IRC was proposed by Ayer et al. [4] under the name of *Pool Adjacent Violators* (*PAV*). Starting with an initial solution consisting of n separate blocks, one for each activity, successive pairs of blocks (B_i, B_{i+1}) not satisfying primal conditions are iteratively identified. Such blocks are merged, and the next iteration is started. The order in which these block couples are identified does not affect the final result of the algorithm. An illustration of the method on the previous example is given in Figure 4. The algorithm iteratively merges the first pair of blocks that does not verify primal conditions. We notice that the optimal solution is reached after three merges (at Step 3).

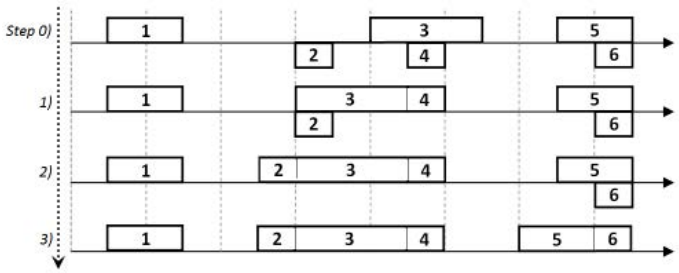


Figure 4: The PAV algorithm illustrated on timing problems

Chakravarti [23] proved that PAV is a dual feasible method for the linear problem when the distance considered is $\| \ \|_1$ ($c(\mathbf{t}) = \Sigma|t_i - N_i|$), while Grotzinger and Witzgall [65] and Best and Chakravarti [14] showed that PAV is a dual algorithm for IRC with quadratic costs (Euclidean distance).

The PAV algorithm was also generalized to convex functions by Best et al. [15] and Ahuja and Orlin [1], achieving a complexity of $O(n)$ unimodal minimizations. It is noteworthy that, under a totally different formalism, an equivalent algorithm was discovered by Dumas et al. [46] for a general vehicle routing setting with convex time-dependent service costs. For IRC with the $\| \ \|_1$ distance, the PAV algorithm can be implemented in $O(n \log^2 n)$ elementary operations using balanced search trees [117], or $O(n \log n)$ complexity using scaling techniques [1]. Finally, $O(n)$ algorithms are known for quadratic objectives [46, 65, 116].

5.4 Separable costs and multiple time windows

Without the previous convexity assumption, the timing problems $\{\Sigma c_i(t_i)|\emptyset\}$ become more complex, and many authors have focused on separable PiL costs. The feature MTW especially [27, 144] can also be linearly reduced to a $\{\Sigma c_i(t_i)|\emptyset\}$ problem when a closed form representation of the multiple time windows, such as a list of feasible intervals, is available.

In the presence of non-negative and Lower Semi-Continuous (LSC) costs ($c_i(t_i) \leq lim_{\epsilon \to 0} \min\{c_i(t_i + \epsilon), c(t_i - \epsilon)\}$ at every discontinuity point), the timing problem $\{\Sigma c_i(t_i)|\emptyset\}$ can be efficiently solved by dynamic programming. A large range of backward and forward approaches has thus been proposed in the routing and scheduling literature by Hendel and Sourd [74], Ibaraki et al. [86], Sourd [135], Yano and Kim [152] and Ibaraki et al. [87].

Solving $\{\Sigma c_i(t_i)|\emptyset\}$ by forward dynamic programming involves the *forward minimum cost* function $F_i(t)$, which evaluates the minimum cost to execute the sequence of activities $(a_1, \ldots, a_i)$ while starting the last activity before or at time t ($t_i \leq t$). $F_i(t)$ functions can be computed by means of Equation (19), starting from the case $i = 1$ with a single activity where $F_1(t) = \min_{x \leq t} c_1(x)$. The optimal solution value of the timing problem is $z^* = F_n(+\infty)$, and the optimal activity execution dates can be retrieved from the $F_i(t)$ functions.

$$F_i(t) = \min_{0 \leq x \leq t} \{c_i(x) + F_{i-1}(x - p_{i-1})\} \qquad 1 < i \leq n \tag{19}$$

The symmetric way to solve this problem by backward programming involves the *backward minimum cost* function $B_i(t)$, which evaluates the minimum cost to execute the sequence of activities $(a_i, \ldots, a_n)$, while executing the first activity a_i after or at time t ($t_i \geq t$). $B_i(t)$ functions are computed by backward recursion, starting with $B_n(t) = \min_{x \geq t} c_n(x)$ and using Equation (20). The optimal solution value of the timing problem is $z^* = B_1(-\infty)$.

$$B_i(t) = \min_{x \geq t} \{c_i(x) + B_{i+1}(x + p_i)\} \qquad 1 \leq i < n \tag{20}$$

These methods can be implemented in $O(n\varphi_c)$, where φ_c stands for the total number of pieces in the activity cost functions c_i. When the costs are also convex, the use of efficient tree data structures leads to a complexity of $O(\varphi_c \log \varphi_c)$ [87], matching the best available approaches in $O(n \log n)$ for the particular cases related to IRC, (E/T) scheduling, and soft time windows.

5.5 State-of-the-art: single-dimensional features

As illustrated by this section, single-dimensional features are related to many prominent problems such as LISP and IRC, which have been the subject of extensive research. Various algorithms were examined and state-of-the-art methods for each particular feature and problem were identified. In the particular case of $\{\Sigma c_i^{\text{CVX}}(t_i)|\emptyset\}$, 26 methods from various fields such as routing, scheduling and isotonic regression were classified into three main families: primal, dual, and dynamic programming methods. Efficient linearithmic methods are known for a very general problem of this category, $\{\Sigma c_i^{\text{CVX}}(t_i)|\emptyset\}$, in the presence of either convex or LSC and piecewise linear cost functions. Still, as illustrated in Section 7, the resolution of series of similar timing instances during a local search can be performed more efficiently by means of re-optimization procedures.

6 Two-dimensional features

We now focus on the problems with two-dimensional features. These features are also often considered in the presence of time-window TW constraints. The presentation is structured in relation to the level of problem complexity and generality. Starting with the duration feature DUR, which involves

exclusively the first and last activities together, we then examine two-dimensional features involving successive activities: no-wait NWT, idle time IDL, and flexible $c_i(\Delta t_i)$ or time-dependent $P(t)$ processing times. Finally, features involving any pair of activities, such as time-lags TL, and cost functions separable by pairs of variables $\Sigma c_{ij}(t_i, t_j)$ are analyzed.

6.1 Total duration and total idle time

Accounting for total duration or idle-time is meaningful when one has the possibility to delay the beginning of operations. Otherwise, considering the maximum execution date feature C is sufficient. Whereas delaying the start of production is generally not an option in scheduling problems, it becomes particularly relevant in routing, as real-life objectives and driver's wages are frequently based on duration. We mention Savelsbergh [132] for duration minimization under time-window and duration constraints in VRPs, Cordeau et al. [31] that generalizes the previous approach for soft time-windows and duration constraints, Desaulniers and Villeneuve [37] for shortest path settings with linear idle-time costs and time windows, and Desaulniers et al. [38] and Irnich [91] for a general framework which addresses duration and idle time features, among others, in various time-constrained routing and crew scheduling problems. It should be noted that computing the *total duration* or the *total idle time* is equivalent in the presence of fixed processing times. Therefore, without loss of generality we will focus on duration in this section.

To manage duration features in $\{DUR|TW\}$ and $\{\emptyset|DUR, TW\}$, Savelsbergh [132] proposed to first rely on a minimum idle time policy, and then shift activity execution dates forward to reduce the total duration. The related amount of shift was introduced many years ago in the project scheduling literature [105] as the latest processing date for an activity that does not cause a delay in the calendar. It is also known in the VRP literature under the name of *forward time slack* [131, 132]. The following quantities are computed for each activity a_i: the earliest feasible execution date T_i, the cumulative idle time W_i on the subsequence $(a_1, \ldots, a_i)$ according to these execution dates, and the partial forward time slack F_i on the subsequence $(a_1, \ldots, a_i)$. These values are computed recursively by means of Equations (21-23), starting with the case of a single activity where $T_1 = r_1$, $W_1 = 0$ and $F_1 = d_1 - r_1$.

$$T_i = \max(T_{i-1} + p_{i-1}, r_i) \qquad 1 < i \leq n \tag{21}$$

$$W_i = W_{i-1} + T_i - T_{i-1} - p_{i-1} \qquad 1 < i \leq n \tag{22}$$

$$F_i = \min(F_{i-1}, d_i - T_i + W_i) \qquad 1 < i \leq n \tag{23}$$

The problem admits a feasible solution if and only if $T_i \leq d_i$ for all i. The execution date of the first activity in an optimal solution is given by $t_1^* = r_1 + \min\{F_n, W_n\}$. The other dates are computed using the minimum idle time policy. Both feasibility checking and duration minimization problems are thus solved in $O(n)$. Desaulniers and Villeneuve [37], Kindervater and Savelsbergh [97] and Irnich [91] proposed different calculations of this optimal schedule. As pointed out in Parragh et al. [118], all these approaches are equivalent.

Tricoire et al. [144] recently considered a more complex timing problem aiming to minimize duration under MTW constraints $\{DUR|MTW\}$. Each activity a_i is associated with a set of k_i time windows, $MTW_i = \{[r_{i1}, d_{i1}], \ldots, [r_{ik_i}, d_{ik_i}]\}$. The authors proposed a procedure that first removes some unnecessary time-window segments, not suitable for any feasible solution, while detecting infeasible timing problems. In a second step, the procedure examines a subset of *dominant* schedules, such that "no better solution exists with the same last activity execution date". For a given execution date t_n of the last activity, a *dominant* schedule with minimum duration can be found using the backward recursion of Equation (24).

$$t_{i-1} = \max\{t \mid t \leq t_i - p_{i-1} \ \wedge \ t \in MTW_i\} \tag{24}$$

Starting from the dominant schedule $\bar{\mathbf{t}}$ with earliest completion time, the method iteratively identifies the last activity a_i followed by idle time: $\bar{t}_i + p_i < \bar{t}_{i+1}$. If activity a_i does not admit a later time window, the algorithm terminates. Otherwise, the execution date of activity a_i is set to the beginning of the next time window, and the execution dates of activities situated afterwards in the sequence are re-computed with a minimum idle time policy. This leads to a *dominant* schedule which becomes $\bar{\mathbf{t}}$ in the next iteration. Tricoire et al. [144] proved that at least one dominant schedule explored in the course of the algorithm is optimal. If each customer is associated to at least one time window, the overall method can be implemented in $O(n\varphi_{\text{MTW}})$, φ_{MTW} representing the number of time windows in the problem.

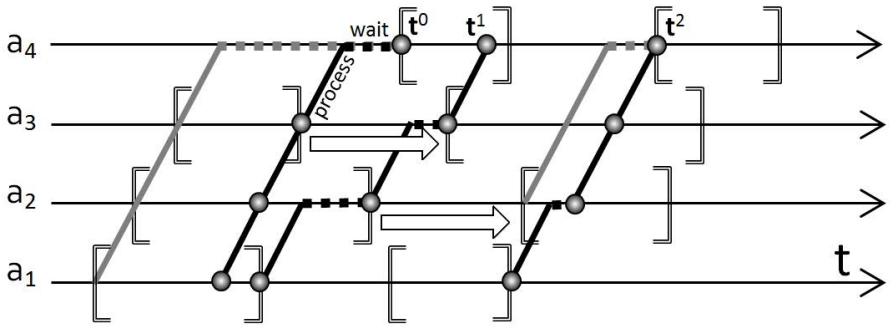


Figure 5: Duration minimization under multiple time-window constraints

This algorithm is illustrated in Figure 5 on a small example with four activities. Activities are represented from bottom to top with their time windows. The earliest completion date is computed with a minimum idle time policy, illustrated in gray lines. The initial dominant schedule $\bar{\mathbf{t}}^0$, in black, is then determined by backward recursion using Equation (24). This schedule presents waiting time after activity a_3, and thus the execution date of this activity is delayed to the next time window, leading to a dominant schedule $\bar{\mathbf{t}}^1$. Now the latest activity followed by waiting time is a_2. Its execution date is delayed, and leads to the dominant schedule $\bar{\mathbf{t}}^2$. The latest activity followed by waiting time is a_1. As there is no later time window for this activity, the algorithm terminates. Among the three dominant schedules explored by the method, the best solution with minimum duration has been reached by $\bar{\mathbf{t}}^2$, and is optimal.

6.2 No wait and idle time

No-wait NWT and idle-time IDL features appear in various settings involving, among others, deterioration of products, maximum waiting times in passenger transportation, fermentation processes in the food industry, and cooling in metal-casting processes. IDL reduces to NWT when the maximum idle time is set to $\iota_i = 0$. No-wait constraints $t_i = t_{i+1}$ can also be addressed by problem reformulation, merging unnecessary variables. When no waiting time is allowed on the entire activity sequence, the timing problem becomes a minimization problem of a sum of single-variable functions. Two main categories of problems have been considered in the literature for NWT and IDL: the feasibility problem under idle-time and time-window constraints, and the optimization problem when some of these features appear in the objective function, treated in Section 6.3 in a more general context.

Feasibility checking under maximum idle time and time-window constraints has been frequently studied in the routing literature. Hunsaker and Savelsbergh [82] designed an algorithm to check the feasibility of itineraries in dial-a-ride settings. This algorithm contains a $O(n)$ checking method for the special case of $\{\emptyset|IDL, TW\}$. The solution to $\{\emptyset|IDL, TW\}$ is found in two scans of the activity sequence. The first pass considers the relaxed subproblem $\{\emptyset|TW\}$, determining for each

activity i the earliest feasible execution dates T_i complying with time-windows and processing times. This calculation is performed by means of Equation (21). In a second pass, starting with $T'_n = T_n$, the algorithm proceeds backwards in the sequence to determine the earliest feasible execution dates $T'_i = \max(T'_{i+1} - p_i - \iota_i, T_i)$ and check idle-time constraints. The problem is declared infeasible if for any $i \in \{1, \ldots, n\}$, $T_i > d_i$ or $T'_i > d_i$.

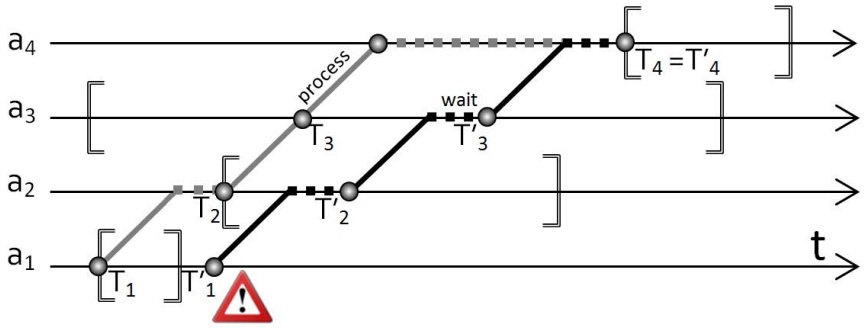


Figure 6: Feasibility checking under time-windows and idle-time constraints

The two passes are illustrated in Figure 6 for a small problem with four activities, represented from bottom to top with their time windows. The first pass (T_i values) has been represented in gray, while the backward scan, in black, provides the earliest feasible execution date for each activity T'_i. As shown in the figure, this value exceeds the time window of activity a_1 ($T'_1 > d_1$), and thus the timing problem illustrated is infeasible.

6.3 Flexible processing times

Time-resource trade-offs and project crashing problems have been thoroughly studied in the operations research literature, since they are critical in many applications, including resource allocation, energy optimization, project scheduling and crashing, optimization of search effort, portfolio selection, and advertising, among others [85, 95, 111, 119, 141]. These problems have all in common that the activity processing times can be increased or reduced at a cost. We refer to this feature as "flexible processing times" $\Sigma c_i(\Delta t_i)$. The characteristic function of this feature is a general separable function of successive execution time differences $t_{i+1} - t_i$. As such, it generalizes both *NWT* and *IDL*.

When only *NWT*, *IDL*, and $\Sigma c_i(\Delta t_i)$ features are encountered, the timing problem $\{\Sigma c_i(\Delta t_i)|\emptyset\}$ can be decomposed along $t_{i+1} - t_i$ values, and the independent minimization of every $c_i(\Delta t_i)$ leads to the optimal solution where $t_{i+1} - t_i = \arg\min_{\Delta t_i} \Sigma c_i(\Delta t_i)$.

When the c_i functions are convex and in the presence of an additional deadline t_{max} on the last activity, the problem $\{\Sigma c_i(\Delta t_i)|C\}$ can be reformulated (Equations 25-27) as a Resource Allocation Problem (RAP) (Equations 28-30) by setting $t_0 = 0$ and using the change of variables $x_i = t_{i+1} - t_i$.

$$\min_{\mathbf{t}} \quad \sum_{i=1}^{n-1} c_i(t_{i+1} - t_i) \qquad (25)$$

$$s.t. \quad t_n - t_1 \leq t_{max} \qquad (26)$$

$$t_i + p_i \leq t_{i+1} \qquad 1 \leq i < n \qquad (27)$$

$$\Leftrightarrow$$

$$\min_{\mathbf{t}} \quad \sum_{i=1}^{n-1} c_i(x_i) \qquad (28)$$

$$s.t. \quad \sum_{i=1}^{n-1} x_i \leq t_{max} \qquad (29)$$

$$x_i \geq p_i \qquad 1 \leq i < n \qquad (30)$$

Resource allocation problems with convex costs have been the subject of many articles. *Lagrangian methods* [24, 51, 137] seek to iteratively progress towards the optimal value of the single dual variable

associated to Equation (29) while respecting primal feasibility. *Pegging* algorithms [98, 102, 130] relax the constraints of Equations (29-30), iteratively solving the relaxed problems and fixing the values of variables that do not satisfy primal constraints. Dynamic programming can also be efficiently applied [93]. $\{\Sigma c_i^{\text{CVX}}(\Delta t_i)|C\}$ with integer variables can be addressed in $O(n \log \frac{t_{max}}{n})$ [58, 77], and an ϵ-approximate solution can be found in $O(n \log \frac{t_{max}}{\epsilon n})$ for the continuous version [77].

In the presence of polymatroidal constraints, convex resource allocation problems can be solved to optimality by a greedy algorithm iteratively incrementing the least-cost variable. Hochbaum [77] proposes a greedy algorithm with scaling for $\{\Sigma c_i^{\text{CVX}}(\Delta t_i)|C, D\}$, which produces an ϵ-approximate solution in $O(n \log n \log \frac{t_{max}}{\epsilon n})$. This problem appears in various application contexts, such as lot sizing [142], assortment with downward substitution [120], and telecommunications [113], among others. Let m be the number of deadline constraints. The problem $\{\Sigma c_i^{\text{CVX}}(\Delta t_i)|C, D\}$ with continuous or integer variables can be solved in $O(n \log m \log \frac{t_{max}}{\epsilon})$ and $O(n \log m \log \frac{t_{max}}{n})$, respectively, with the decomposition approach of Vidal et al. [149]. Finally, the quadratic $\{\Sigma c_i^{\text{CVX}}(\Delta t_i)|C, D\}$ with continuous variables can be addressed in $O(n \log n)$ [78] or $O(n \log m)$ [149].

Adding time-window constraints to the model of Equations (25-27) leads to other timing settings and resource allocation problems with non-polymatroidal constraints. In the special case of Norstad et al. [111], a timing problem $\{\Sigma c_i^{\text{CVX}}(\Delta t_i)|TW\}$ is addressed with the objective $z(t) = \sum_{i=1}^{n} d_{i,i+1}\bar{c}((t_{i+1} - t_i)/d_{i,i+1})$ and non-increasing and convex $\bar{c}(\Delta t)$ functions (independent of the activity). In the presence of such functions, relaxing time-window constraints leads to an optimal solution with constant ratio $(t_{i+1} - t_i)/d_{i,i+1}$ for all i, and thus constant speed on all legs. The *recursive smoothing algorithm* (RSA) exploits this property by maintaining this ratio constant on subsequences and progressively re-introducing violated time-window constraints. The overall method works in $O(n^2)$ elementary operations once the minimum of each function $\bar{c}(\Delta t)$ is given. This method is a *dual feasible* algorithm based on the relaxation and re-introduction of window constraints. It is closely related to the "string" methodology described in [33] and also discussed in [149].

Sourd [135] and Hashimoto et al. [70] have independently studied $\{\Sigma c_i(\Delta t_i), \Sigma c_i(t_i)|\emptyset\}$ (Equation 31) with piecewise linear functions in the context of (E/T) scheduling and vehicle routing, and report its NP-hardness.

$$\min_{(t_1,\ldots,t_n)\in\Re^{n+}} \quad \sum_{i=1}^{n} c_i(t_i) + \sum_{i=1}^{n-1} \tilde{c}_i(t_{i+1} - t_i) \tag{31}$$

When the functions c_i and $\tilde{c}_i$ are PiL with integer breakpoints, a dynamic programming algorithm is proposed to solve the problem in $O(T^2)$, where T represents an upper bound on the schedule durations. This dynamic programming algorithm can be viewed as an extension of the resource allocation algorithm of Karush [93]. The method can be implemented with a forward dynamic programming function $F_i(t)$ (Equations 32-33), which evaluates the minimum cost to process the subsequence of activities $(a_1, \ldots, a_i)$, starting the last activity exactly at time t $(t_i = t)$. The resulting optimal cost is given by $z^* = \min_t F_n(t)$.

$$F_1(t) = c_1(t) \tag{32}$$

$$F_i(t) = c_i(t) + \min_{0\le x\le t}\{F_{i-1}(x) + \tilde{c}_{i-1}(t - x)\} \qquad 1 < i \le n \tag{33}$$

A polynomial dynamic programming algorithm working in $O(n\varphi_c + n\widehat{\varphi_c} \times \widetilde{\varphi_c})$ exists for the case where the functions $\tilde{c}_i(\Delta t)$ are PiL and convex [70, 135]. φ_c and $\widetilde{\varphi_c}$ represent the total number of pieces in the cost functions c_i and $\tilde{c}_i$, and $\widehat{\varphi_c}$ stands for the number of convex pieces in the cost functions c_i. Efficient re-optimization procedures have also been proposed (Section 7).

Finally, DUR involved in the objective can be seen as a special case of $\Sigma c_i(\Delta t_i)$ where $c_i^{\text{DUR}}(\Delta t_i) = \Delta t_i$. MTW is also reducible to $\Sigma c_i(t_i)$, and thus the previous algorithm provides an alternative way to solve $\{DUR|MTW\}$ or $\{\emptyset|DUR, MTW\}$ in $O(n + n\varphi_{\text{MTW}})$, where φ_{MTW} represents the total number of time windows.

6.4 Time-dependent processing times

In several application settings, activity processing times may vary as a function of the execution dates. In machine and project scheduling for example, learning, deterioration effects and other time-dependencies can have a large impact [3, 25]. Network congestion is a major concern for vehicle routing and data transmission [99, 145] and, thus, the time-dependent processing-time feature $P(t)$ appears in various network optimization problems: shortest path [28, 44, 67], traveling salesman [104], and vehicle routing [11, 43, 71, 88, 103], among others.

The literature on the subject can generally be separated between discrete and continuous settings. Discrete optimization models generally involve time-space networks which are less likely to present the timing issues studied in this article, whereas several continuous models have led to explicit timing problems with $P(t)$ features, as in Donati et al. [43], Fleischmann et al. [56], Ichoua et al. [88], and Hashimoto et al. [71]. These models involve constraints of the type $t_i + p_i(t_i) \leq t_{i+1}$ within a timing formulation with other additional features.

The *FIFO* assumption on functions p_i is often valid. FIFO implies that any delay in an activity execution date results in a delay in its completion date. The assumption is meaningful in several settings, e.g. vehicle routing, as two vehicles that behave similarly on the same route are supposed to remain in the same arrival order, whatever congestion happens [88].

$$\text{FIFO assumption: } \forall i \; x \geq y \Longrightarrow x + p_i(x) \geq y + p_i(y) \tag{34}$$

Time-dependent processing-time features are generally assumed to result in more complex timing problems. However, one should clearly identify the source of the difficulty, which is frequently imputable to the computation and access to $p_{ij}(t)$ throughout the search, and not necessarily to the timing problem resolution. Assuming that $p_{ij}(t)$ can be evaluated in constant time and under FIFO, $\{D|R, P(t)\}$ is still solvable in $O(n)$ by means of a minimum idle time policy [56] and the time-slack approach of Savelsbergh [132] can still be applied [43] to $\{\emptyset|TW, P(t)\}$. Still, dedicated methodologies are necessary for other settings such as $\{DUR|TW, P(t)\}$.

The time-dependent timing problem $\{\Sigma c_i(t_i)|P(t)\}$, Equations (35-36), is addressed in Hashimoto et al. [71]. All functions considered are PiL, non-negative and lower semicontinuous.

$$\min_{(t_1,\ldots,t_n)\in\Re^{n+}} \quad \sum_{i=1}^{n} c_i(t_i) \tag{35}$$

$$s.t. \quad t_i + p_i(t_i) \leq t_{i+1} \qquad 1 \leq i < n \tag{36}$$

The authors propose a dynamic programming approach, which extends the method of Section 5.4. It involves the functions $F_i(t)$, which represent the minimum cost to process the subsequence of activities $(a_1, \ldots, a_i)$ while starting the last activity before t ($t_i \leq t$). Under the assumption of Equation (37), which is weaker than FIFO, the method can be implemented in $O(n\varphi_c + n\varphi_p)$, where φ_c and φ_p are the total number of pieces in cost and processing-time functions.

$$\text{(HYI) assumption: } \forall i \; x + p_i(x) = y + p_i(y) \Longrightarrow x + p_i(x) = z + p_i(z) \; \forall z \in [x, y] \tag{37}$$

This method for $\{\Sigma c_i(t_i)|P(t)\}$ thus presents the same quadratic complexity as in the case without time dependency (Section 5.4). When the previous assumption does not hold, the dynamic programming method of Hashimoto et al. [71] is not polynomial, and the question remains open whether $\{\Sigma c_i(t_i)|P(t)\}$ is polynomially solvable.

6.5 Time lags

The two-dimensional features surveyed in the previous sections involved linking constraints and objectives between the first and last tasks, in the case of DUR, or between pairs of successive variables

in the case of NWT and IDL. We now review the time-lag TL feature, which brings into play a time difference $t_j - t_i$ between any two activity execution dates t_i and t_j. This feature is thus a generalization of NWT, IDL and DUR.

To the best of our knowledge, early research on time lags has been conducted by Mitten [107] for flowshop scheduling problems. This feature has been used since to model many problem characteristics in various domains, such as the deterioration of food or chemical products, glue drying, customer requirements in some dial-a-ride problems, elevator dispatching, quarantine durations, and so on. Time-lag scheduling problems on a single machine have also been shown by Brucker et al. [20] to generalize all shop, multi-purpose machines, and multi-processor scheduling problems. Hence, timing problems with TL are likely to be difficult.

The most basic problem with TL feature relates to feasibility checking under time-lag constraints of the form $t_i + \delta_{ij} \leq t_j$. When $\delta_{ij} \geq 0$, the constraint is called positive time lag, and corresponds to a minimum delay between activities a_i and a_j, whereas $\delta_{ij} \leq 0$ corresponds to a negative time lag, and involves a maximum delay of $-\delta_{ij}$ between the activities a_j and a_i. Equality constraints $t_i + \delta_{ij} = t_j$ involve both positive and negative time lags. The resulting timing problem $\{\emptyset|TL\}$ can be seen as a special case of project scheduling on a chain of activities, and the METRA potential method (MPM) of Roy [128, 129] can be applied. In MPM, the time-lag constraints are represented on a graph $G = (V, A)$, where each activity a_i is associated with a node $v_i \in V$, and each arc (v_i, v_j), associated with a weight w_{ij}, represents a temporal constraint of the form $t_j - t_i \geq w_{ij}$. The feasibility of $\{\emptyset|TL\}$ is equivalent to the non-existence of a positive length cycle in this graph [9, 35]. The algorithm of Floyd-Warshall can be employed to solve this problem in $O(n^3)$, but the longest-path procedure of Hurink and Keuchel [83], also in $O(n^3)$, is shown to provide faster results in practice. Potts and Whitehead [122] also considered a coupled-operation scheduling problem with only $n/2$ time-lag constraints, and timing feasibility is checked in $O(n^2)$. The authors underlined the computational burden of such timing algorithms, which strongly degrades the performance of neighborhood searches or branch and bound procedures.

Hunsaker and Savelsbergh [82] studied a case of $\{\emptyset|TL, TW\}$ timing in the context of dial-a-ride problems. Activities represent customer requests on pick-up and deliveries services, which occur by pairs, such that any pick-up always precedes its corresponding delivery in the sequence. Each such pair of activities is linked by a single positive time-lag constraint. The total number of time-lag constraints is thus $n/2$. The problem also involves time windows and maximum idle times for each activity. The authors claim that the resulting feasibility problem can be solved in three passes on the sequence of activities with linear complexity. Yet, the algorithm presents a small flaw, which is straightforward to correct [143], but leads to a $O(n \log n)$ complexity [73]. A $O(n)$ complexity is finally achieved in Firat and Woeginger [54] by means of a reduction to a shortest path problem on a weighted interval graph.

The same setting is also addressed in Gschwind and Irnich [66]. The authors describe a labeling procedure based on $|M_i|$ resources for each pickup and each delivery activity a_i. M_i stands for the current number of open pickups at a_i. This labeling procedure provides another way to check the feasibility of a fixed activity sequence $\{\emptyset|TL, TW\}$ in $O(n^2)$. Furthermore, the feasibility of an extended sequence with one additional activity a_{n+1} can be evaluated in $O(M_{n+1})$ given the information on the original sequence. Such forward extension function is critical when solving shortest path problems with underlying timing features.

Cordeau and Laporte [29] and Berbeglia et al. [12] consider a dial-a-ride setting with an additional duration constraint on the entire trip duration. The authors solve heuristically a Lagrangian relaxation of the problem with a hierarchical objective. Total trip duration infeasibility is minimized, then time-window infeasibility and, finally, time-lag infeasibility, that is the timing problem $\{DUR > D > TL|R\}$. The algorithm first minimizes duration and time-window infeasibility as in Section 6.1, then iteratively delays some pick-up services to reduce time-lag infeasibility without increasing any other violation. A computational complexity of $O(n^2)$ is achieved. It was observed in a private

communication, however, that the previous approach only guarantees optimality under an additional assumption that we call *LIFO*, which requires that for any $1 \leq i < j < k < l \leq n$, no activities a_i, a_j, a_k, a_l present "entangled" time-lag constraints of the form $t_k - t_i \leq \delta_{ik}$ and $t_l - t_j \leq \delta_{jl}$. The LIFO assumption is frequently enforced in the vehicle routing literature, especially when transporting passengers, or in the presence of complex loading constraints. In this case, the last object or customer received in the vehicle is the first one to leave. Without this assumption, the difficulty of many problems with time lags strongly increases, and no specialized efficient algorithm is actually known for $\{DUR > D > TL|R\}$ and similar problems.

6.6 Separable costs by pairs of variables

Separable costs by pairs of variables $\Sigma c_{ij}(t_i, t_j)$ generalize all problems combining single or two-dimensional features. The timing problem with this feature alone is NP-hard in the presence of piecewise linear functions since it generalizes $\{\Sigma c_i(t_i), \Sigma c_i(\Delta t_i)|\emptyset\}$. When the objective function is convex, the problem $\{\Sigma c_{ij}^{\text{CVX}}(t_j - t_i), \Sigma c_i^{\text{CVX}}(t_i)|\emptyset\}$ is equivalent to the *convex cost dual network flow problem*, and a weakly polynomial algorithm is provided in Ahuja et al. [2].

6.7 State-of-the-art : "stand-alone" timing methods

In contrast to single-dimensional features, which appeared as fairly well addressed in Section 5 by means of a few algorithms and concepts, two-dimensional features lead to more diverse problem structures and algorithms. Several simple cases with duration minimization or time-dependent processing times can be solved in linear time, but other problems with time-lag features actually require $O(n^3)$ algorithms to be solved exactly. Although polynomial, the latter methods can be impracticable in the context of local searches or branch-and-bound approaches.

Many practical timing settings result in models with linear constraints and linear or separable convex objectives. For these problems, the linear and convex programming theory ensures weakly polynomial resolvability, and provides general solution methods [79, 92, 96]. Some more general problems, however, such as $\{\Sigma c_i(t_i), \Sigma c_i(\Delta t_i)|\emptyset\}$ with PiL functions, are NP-hard, while for other problems, such as $\{\Sigma c_i(t_i)|P(t)\}$ with general piecewise linear functions $P(t)$, the existence or non-existence of polynomial algorithm is still open. Timing settings thus lead to a rich variety of problem structures and complexities.

In all these cases, whether polynomial algorithms are available or not, research is still open to provide more efficient algorithms exploiting the particular structure of the features and problems at hand. The present paper contributed by building a formalism, a classification of features, timing problems and methods. We gathered the most efficient stand-alone timing approaches from various fields of research to tackle both specialized timing settings, and more general features. The focus can now be turned on filling the gaps that have been highlighted in this review, and which continue to appear, following the rich variety of application cases with time constraints emerging nowadays. Finally, important avenues of research target the efficient solving of *series* of timing problems, in the particular context of neighborhood searches and branch-and-bound. Such approaches are presented in the next section.

7 Timing re-optimization

In previous sections, we examined how to address timing problems as a stand-alone issue. Yet, most neighborhood-search-based heuristics, metaheuristics, and some exact methods require to solve iteratively a large number of closely related timing instances. In this case, solving each timing problem

"from scratch", without exploiting any knowledge on previous resolutions, can result in losses of information and redundant computations.

Most local searches for routing and scheduling problems (see [81] for a presentation of local search) rely on a neighborhood based on a limited number of sequence changes. One or several timing subproblems are solved for each neighbor to estimate its feasibility and cost. Figure 7 illustrates two classical neighborhoods to change sequences of activities, 2-OPT* which exchanges the tails of two sequences of activities [123], and OR-OPT to relocate a subsequence of consecutive activities [112]. It is noticeable that large subsequences of activities, SeqA, SeqB, SeqC and SeqD on the figure, are shared by successive timing subproblems. Branch-and-bound procedures for problems with sequencing decisions can similarly involve timing subproblems at nodes of the search tree when evaluating sequences of activities, during lower bound computation and branch pruning [80, 136]. The search for improving columns, in column generation approaches, also frequently involves elementary shortest paths with timing decisions and resource constraints [7, 41, 124].

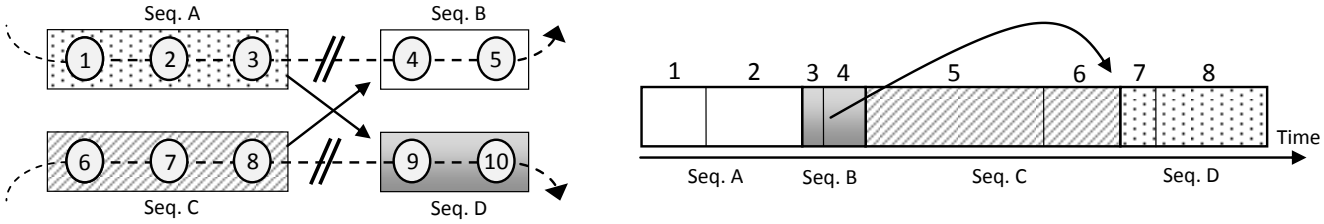


Figure 7: A 2-OPT* local search move (left) and an OR-OPT move (right)

In all these cases, numerous closely related timing problems must be solved, where long subsequences of consecutive activities remain unchanged, and only a minor proportion of problem parameters (reduced cost values for column generation) is impacted. Several authors thus propose to keep *meaningful data* on the global search process to save computations and solve more efficiently these series of similar timing problems. Neighborhood searches largely benefit from such techniques, as move evaluations (and thus the resolution of timing problems) take the largest part of the computational effort. These *re-optimization* methodologies therefore can lead to significant reductions in the computational burden of algorithms.

We now formally define *serial* timing problems in Section 7.1, and present a general framework for re-optimization methods based on sequence concatenations in Sections 7.2-7.3. Links with related re-optimization methodologies are analyzed in Section 7.4, before reviewing or introducing efficient concatenation-based re-optimization methods for each major timing feature and related problems in Section 7.5.

7.1 Problem statement: Serial timing

This section formally defines serial timing problems. Sequence-dependent processing times are also considered, in relation to a large range of vehicle routing applications which rely extensively on re-optimization methods.

Definition 5 (Serial timing)**.** *Let $\mathcal{T}$ be an* incumbent timing problem *with n activities $(a_1, \ldots, a_n)$, sequence-dependent processing-times p_{ij}, and additional features with characteristic functions, $f^x_y(\mathbf{t})$, separated into two sets $\mathcal{F}^{\text{OBJ}}$ and $\mathcal{F}^{\text{CONS}}$ following their role as objective or constraint (Section 2). N permutation functions $\sigma^k : \{1, \ldots, n\} \to \{1, \ldots, n\}$ for $k \in \{1, \ldots, N\}$, are also given. The* serial

timing *problem involves to solve the timing subproblems* $\mathcal{T}^k$ *of Equations (38-40), for* $k \in \{1, \ldots, N\}$.

$$(\mathcal{T}^k): \quad \min_{\mathbf{t}=(t_1,\ldots,t_n)\in\Re^{n+}} \quad \sum_{F^x\in\mathcal{F}^{\text{OBJ}}} \alpha_x \sum_{1\le y\le m_x} f_y^x(\mathbf{t}) \tag{38}$$

$$s.t. \quad t_{\sigma^k(i)} + p_{\sigma^k(i),\sigma^k(i+1)} \le t_{\sigma^k(i+1)} \qquad 1 \le i < n \tag{39}$$

$$f_y^x(\mathbf{t}) \le 0 \qquad F^x \in \mathcal{F}^{\text{CONS}}\ ,\ 1 \le y \le m_x \tag{40}$$

To efficiently solve the previous problem, several types of re-optimization approaches have been developed in the literature to take advantage of the information developed during the successive subproblem solving. One such approach involves re-arranging previously developed schedules in relation to the new settings. For network-flow or shortest-path formulations especially, re-optimization methods related to a change of arcs or costs in the network have been studied over a long period [57, 63, 106, 114]. Most timing problems can also be formulated as linear program. Sensitivity analysis and warm start following a problem modification may be done by means of a primal-dual simplex algorithm. Finally, a last methodology, on which we focus in the following, is based on the observation that a permutation of activities can be assimilated to a concatenation of some subsequences of consecutive activities. Hence, managing information on subsequences (e.g., dynamic programming labels) can lead to significant resolution speed ups [32, 97].

7.2 Breakpoints and concatenations

We first introduce some vocabulary, and then emphasize the links between operations on sequences of activities, such as changes of precedence and activity relocations, and the properties of the resulting permutation functions. These observations lead to efficient re-optimization approaches.

Definition 6 (Permutation breakpoints)**.** *Let* $\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$ *be a permutation. Any integer* b *such that* $\sigma(b) + 1 \neq \sigma(b + 1)$ *and* $1 \le i < n$ *is called a breakpoint of* σ*, and corresponds to non-consecutive values in the permutation representation.*

Let $b(\sigma)$ denote the number of breakpoints of σ, and $b_1^\sigma, \ldots, b_{b(\sigma)}^\sigma$ denote these breakpoints in increasing order. For instance, the permutation $\sigma_0 : \{1, 2, 3, 4, 5, 6\} \to \{4, \mathbf{5}, \mathbf{3}, 1, \mathbf{2}, 6\}$ has three breakpoints (indicated in boldface): $b_1^{\sigma_0} = 2$, $b_2^{\sigma_0} = 3$, and $b_3^{\sigma_0} = 5$. We now show the links between classical operations on activity sequences and the resulting permutation function properties in terms of breakpoints. Two main operations can be considered. The first operation is a change of precedence between two activities. For example, two precedences are changed on the left of Figure 7: activity 3 now precedes activity 9 instead of 4, and activity 8 precedes activity 4 instead of 9. The second operation is the relocation of a sequence of activities. For example, on the right of the figure, the sequence of activities $\{3, 4\}$ is relocated between activities 6 and 7.

Lemma 1 (Precedence changes)**.** *Let* A' *be an activity sequence obtained from* A *by changing* l *precedence relations and* $\sigma_{A\to A'}$ *the associated permutation function, then* $b(\sigma_{A\to A'}) = l$*.*

Lemma 2 (Activity relocations)**.** *Let* A' *be an activity sequence obtained from* A *by relocating* l *activities and* $\sigma_{A\to A'}$ *the associated permutation function, then* $b(\sigma_{A\to A'}) \le 3l$*.*

Any change of the precedence relation results in exactly one breakpoint while any relocation of activity can be assimilated to at most three changes of precedence relations, and thus yields three breakpoints. Situations where k precedence relations are changed from one timing problem T to another problem T' occur frequently in the context of neighborhood searches for combinatorial optimization problems with sequence optimization, e.g., vehicle routing or machine scheduling. A timing subproblem may need to be solved to evaluate the cost and feasibility of each new sequence in the

presence of complicating time constraints. The interest of breakpoints is highlighted in the following proposition. Although straightforward, it provides the basis of re-optimization methods working by concatenation.

Proposition 1. *Let $\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$ be a permutation with breakpoints $b^\sigma_1, \ldots, b^\sigma_{b(\sigma)}$. Let A be a sequence of n activities. Then, $A' = \sigma(A)$ corresponds to the concatenation of exactly $b(\sigma) + 1$ subsequences of consecutive activities in A, as presented in Equation 41. A dummy breakpoint $b^\sigma_0 = 1$ stands for the beginning of the sequence.*

$$A' = \bigoplus_{l=0,\ldots,b(\sigma)-1} (a_{\sigma(b^\sigma_l+1)}, \ldots, a_{\sigma(b^\sigma_{l+1})}) \tag{41}$$

Any bounded number of operations transforming an activity sequence A into A' (relocation of activities, or changes of precedence relations) thus involves a permutation function with a bounded number of breakpoints, such that A' can be seen as a concatenation of a bounded number of subsequences of A. As shown in the following, the data, which is pre-processed from a bounded number of subsequences, may be extended to their concatenation, thus enabling to solve the timing subproblems more efficiently by exploiting existing knowledge.

7.3 Re-optimization "by concatenation"

Re-optimization by concatenation can be formalized by means of a set of four basic re-optimization operators. Three of these operators, **initialization**, **forward extension** and **backward extension** are used to build re-optimization data on subsequences of activities from the incumbent timing problem, while the last operator, **evaluate concatenation**, is specifically tailored to solve a new problem, more efficiently, using the existing re-optimization data on its subsequences of activities.

Initialization – Initialize the data $\mathcal{D}(A)$ of a sequence containing a single activity.

Forward extension – Given an activity a_k and a sequence $A = (a_i, \ldots, a_j)$ with its data, determine the data $\mathcal{D}(A')$ for the sequence $A' = (a_k, a_i, \ldots, a_j)$.

Backward extension – Given an activity a_k and a sequence $A = (a_i, \ldots, a_j)$ with its data, determine the data $\mathcal{D}(A')$ for the sequence $A' = (a_i, \ldots, a_j, a_k)$.

Evaluate concatenation – Given L sequences of activities $\mathcal{D}(A_l), l = 1 \ldots l$ and their data, evaluate the feasibility and the optimal solution cost of the timing problem involving the concatenation of these sequences.

The re-optimization approach illustrated in Algorithm 1 is based on these operators. Data is first built on subsequences of the incumbent timing problem $\mathcal{T}$, by means of the forward and backward extension operators. Since there are $O(n^2)$ such subsequences of consecutive activities, preprocessing can be achieved appending iteratively $O(n^2)$ times one activity at the sequences' end (or beginning). This preprocessing can be either performed before or during the search. The resulting data is then exploited to solve the problems $\mathcal{T}^k$ for $k \in \{1, \ldots, N\}$ by means of the concatenation operator.

Consider a neighborhood search for routing problems with time constraints on routes. In this context, a local search improvement procedure based on sequence changes leads to a number of timing subproblems proportional to the number of neighborhood solutions to explore. The number of derived timing subproblems is usually $N = \Omega(n^2)$, and the derived activity sequences do not involve the concatenation of more than $k = 2, 3$ or 4 subsequences of the original problem.

The overall complexity of a neighborhood exploration, when solving each timing subproblem independently, is $Nc(T)$, $c(T)$ being the computational complexity of one stand-alone timing solution procedure. A straightforward re-optimization approach consists in exhaustively computing the data

Algorithm 1 Re-optimization

1: Build re-optimization data on subsequences of the *incumbent timing problem* $\mathcal{T}$, using *initialize*, and *forward extension* or *backward extension*.
2: For each timing subproblem $\mathcal{T}^k$, $k \in \{1, \ldots, N\}$;
3: Determine the breakpoints involved in the permutation function σ^k;
4: Evaluate the optimal cost of $\mathcal{T}^k$, as the concatenation of $b(\sigma)+1$ activity subsequences from $\mathcal{T}$ (Equation 41).

for each of the $n(n-1)$ subsequences of consecutive activities from $\mathcal{T}$, and then use it to evaluate all moves. Data computation is straightforward to perform in $O(n^2c(I) + n^2c(F/B))$. $c(I)$ and $c(F/B)$ stand for the computational complexity of initialization, and forward or backward extension, respectively. The overall complexity of the new neighborhood exploration procedure is thus $O(n^2c(I) + n^2c(F/B) + Nc(EC))$, $c(EC)$ being the complexity for evaluating the concatenation of less than 4 subsequences. Assuming that $N = \Omega(n^2)$, the computational complexity of neighborhood evaluation becomes $O(N[c(I) + c(F/B) + c(EC)])$ for re-optimization methods instead of $Nc(T)$ for independent solving. Re-optimization operators being less computationally complex than stand-alone methods, the resulting approach is likely to lead to reduced computational effort. The efficiency of Algorithm 1 thus directly relies on the potential to develop concatenation operations that are less computationally complex than stand-alone methods.

Concatenation operators involving more than two sequences are not actually available or not computationally suitable for efficient re-optimization for some settings such as $\{\emptyset|MTW\}$ and most problems with two-dimensional features. In this case, forward and backward propagation may be used along with concatenations of two subsequences only to perform the timing subproblem evaluations. Such an example is given by the lexicographic search of Savelsbergh [131], which can be used to evaluate timing subproblems associated to some well-known neighborhoods for vehicle routing problems exclusively by means of concatenation of two subsequence. Finally, if the concatenation of many subsequences can be operated efficiently, but data creation constitutes the bottleneck in terms of computational effort, the subset of subsequences involved can be restricted to $O(n^{4/3})$ or $O(n^{8/7})$, using the hierarchical approach of Irnich [90].

The relevant data to compute can also be tailored relatively to the neighborhoods at play. For example, the 2-OPT* move presented in Figure 7 involves only the concatenation of subsequences containing the first or last activity, which we call *prefix* or *suffix subsequences*. The number of such subsequences requiring data computation is thus reduced to $O(n)$.

7.4 General literature on the topic

A large range of vehicle routing and scheduling problems are addressed using local search on sequences. Serial timing issues thus frequently arise in these fields.

Following the seminal work of Savelsbergh [131, 132], Kindervater and Savelsbergh [97] proposed a framework to manage several constraints on vehicle routes, such as precedence constraints, time windows, collection and deliveries under capacity constraints. Several of these constraints are explicitly, or can be assimilated to, timing features. To perform efficient feasibility checking, the authors develop *global variables* on partial routes, which are used in concatenation operations to evaluate moves consisting of a constant number of edge exchanges. Move evaluations are performed in lexicographic order to allow calculation of the global variables through the search.

Cordone [32] and Duhamel [45] report similar concepts of global data management on subsequences and concatenation operators. Although these methodologies are essentially dedicated to the VRPTW, they explore different possibilities related to the concept of macro-nodes. Indeed, when the information

developed on subsequences has the same structure as the problem data on activities, subsequences of activities can be replaced by equivalent single activities during timing resolution. In this case, the size of the problem may be temporarily reduced by collapsing nodes into *macro-nodes*, leading to algorithms based on aggregation of activities and multi-level approaches [10, 48, 150].

Campbell and Savelsbergh [22] presented a compilation of efficient insertion heuristics for many vehicle routing problems with additional characteristics such as shift time limits, variable delivery quantities, fixed and variable delivery times, and multiple routes per vehicle. These methods iteratively create solutions by adding customers to the routes. The authors show that by managing global data on the routes, the cost of feasibility of customer insertions can be evaluated in amortized $O(1)$ for many of these settings.

A rich body of dynamic programming-based timing algorithms is also presented in Hashimoto [69], Hashimoto et al. [70, 71], Ibaraki et al. [86, 87], and Hashimoto et al. [72]. Forward and backward propagation is used, with an additional "connect" operator to manage concatenation of two subsequences, thus leading to efficient re-optimization approaches by concatenation for several timing problems involving Piecewise Linear (PiL) functions.

The framework of Desaulniers et al. [38] models many constraints and objectives on sequences of activities as resources that are subject to window constraints, and are extended from one activity to the next by means of resource extension functions (REFs). This framework proved extremely efficient to model many crew scheduling and routing problems and solve them by column generation [39]. It has also been recently extended by Irnich [90, 91] to perform efficient neighborhood search under various constraints on routes, such as load dependent costs, simultaneous pickup and deliveries, maximum waiting and duty times. To that extent, REFs "generalized to segments" are built on subsequences to characterize their resource consumption. Inverse REFs are defined to give an upper bound on the resource consumptions that allow the sequence to be processed. This data on subsequences can be used to evaluate efficiently the cost or feasibility of local search moves. This framework, however, requires rather restrictive conditions: the existence of REFs that can be generalized to subsequences and inversed. These conditions are satisfied only by a limited subset of the timing problems and features introduced previously, such as $\{\emptyset|TW\}$, $\{\emptyset|MTW\}$ or $\{DUR|TW\}$.

Several general methodologies thus exist in the literature to tackle timing problems within a neighborhood search context. However, these approaches are restricted by the types of concatenations allowed, the assumptions made on features, or the applicability of the models to a wide range of constraints. The timing formalism we propose and its generalization to re-optimization procedures by concatenation, following the concepts of Kindervater and Savelsbergh [97], is less specialized and thus can encompass a wider range of timing settings and methods. As shown in the next sections, this framework unifies previous successful concepts, and provides a line of thought for the development of efficient algorithms for various timing problems. The *forward*, *backward extension* and *concatenation* operations build upon dynamic programming and bi-directional search [125] concepts. Finally, this framework can be viewed as a generalization of the approach of Irnich [90, 91] to timing problems. Indeed, generalized REFs and their inverse provide, when they exist, the suitable re-optimization data.

7.5 Re-optimization algorithms

We now review and analyze re-optimization approaches for main timing features and problems. As in the first part of this paper, the analysis is organized by increasing order of complexity and feature dimensions. For each timing setting we describe the re-optimization data, as well as the operators that can be used for forward and backward data computation and for evaluating the concatenation of several subsequences. These operators can be used within Algorithm 1 to develop efficient re-optimization approaches by concatenation.

7.5.1 Constant activity costs and cumulative resources.

In the presence of constant activity costs or, more generally, with any cumulative resource such as distance, load or time bounded by a global constraint, evaluating a solution from scratch would involve to browse each activity and cumulate the resource, resulting in a $O(n)$ complexity. To perform more efficient evaluations, a well-known re-optimization approach involves preprocessing partial loads and resource consumption on subsequences. This is equivalent to applying Algorithm 1 with the following data and operators.

DATA. Partial cost $C(A_i)$ (or resource consumed) by each subsequence A_i.

DATA COMPUTATION. Cumulating the cost (or resource consumption) on the subsequence.

EVALUATE CONCATENATION. Cumulating the partial costs (or resource consumptions) on subsequences: $C(A_1 \oplus \cdots \oplus A_k) = \sum C(A_i)$.

The evaluation of the concatenation of a bounded number of subsequences can thus be performed in a bounded number of operations, leading to $O(1)$ complexity for move evaluation when the data is available. Data can be processed in amortized constant time for each move during a local search procedure for many classic neighborhoods, using a *lexicographic order* [97] for move evaluation, or developed in a preprocessing phase for a complexity of $O(n^2)$, which is usually dominated by the neighborhood size.

7.5.2 Weighted execution dates and non-decreasing linear costs.

The feature W and, in general, non-decreasing linear time-dependent costs of the form $c_i(t_i) = w_i t_i + c_i$ with $w_i \geq 0$ for $i \in \{1, \ldots, n\}$ can be addressed with the following re-optimization data and operators.

DATA. Total processing time $T(A_i)$ for all the activities of the sequence A_i but the last one. Waiting cost $W(A_i)$ related to a delay of one time unit in the sequence processing, and sequence cost $C(A_i)$ when started at time 0.

DATA COMPUTATION AND EVALUATE CONCATENATION. For a sequence A with a single activity, $T(A) = 0$, $W(A) = w_{A(1)}$ and $C(A) = c_{A(1)}$. Equations (42-44) can be used to evaluate the cost of concatenations and to compute the data for sequences with more activities by induction.

$$W(A_1 \oplus A_2) = W(A_1) + W(A_2) \tag{42}$$

$$C(A_1 \oplus A_2) = C(A_1) + W(A_2)(T(A_1) + p_{A_1(|A_1|),A_2(1)}) + C(A_2) \tag{43}$$

$$T(A_1 \oplus A_2) = T(A_1) + p_{A_1(|A_1|),A_2(1)} + T(A_2) \tag{44}$$

These equations can also manage sequence-dependent processing times. It is remarkable that, in this case, the re-optimization data is a simple generalization of single-activity characteristics to sequences of activities, similar to the *generalization to segments* concepts of Irnich [91].

7.5.3 Time-windows feasibility check.

Savelsbergh [131] opened the way to efficient feasibility checking with regards to time windows in the context of local search. In the subsequent work of Kindervater and Savelsbergh [97], the following re-optimization data and operators are introduced.

DATA. Total processing time $T(A_i)$ for all the activities of the sequence A_i but the last one. Earliest execution date $E(A_i)$ of the last activity in any feasible schedule for A_i. Latest execution date $L(A_i)$ of the first activity in any feasible schedule for A_i. A record $isFeas(A_i)$ valuated to true if and only if a feasible schedule for A_i exists.

DATA COMPUTATION AND EVALUATE CONCATENATION. For a sequence A with a single activity, $T(A) = 0$, $E(A) = r_{A(1)}$, $L(A) = d_{A(1)}$ and $isFeas(A) = true$. Equations (45-48) enable subsequence

concatenations to be evaluated in $O(1)$.

$$T(A_1 \oplus A_2) = T(A_1) + p_{A_1(|A_1|),A_2(1)} + T(A_2) \tag{45}$$

$$E(A_1 \oplus A_2) = \max\{E(A_1) + p_{A_1(|A_1|),A_2(1)} + T(A_2), E(A_2)\} \tag{46}$$

$$L(A_1 \oplus A_2) = \min\{L(A_1), L(A_2) - p_{A_1(|A_1|),A_2(1)} - T(A_1)\} \tag{47}$$

$$isFeas(A_1 \oplus A_2) \equiv isFeas(A_1) \ \wedge \ isFeas(A_2) \ \wedge \ (E(A_1) + p_{A_1(|A_1|),A_2(1)} \leq L(A_2)) \tag{48}$$

It should also be noted that the total duration to process the sequence is given by $DUR(A_i) = \max\{E(A_i) - L(A_i), T(A_i)\}$, and thus this approach enables to solve $\{\emptyset|DUR, TW\}$ and $\{DUR|TW\}$ serial timing problems in $O(1)$.

7.5.4 Earliness, tardiness, soft time-windows, and separable costs.

Timing problems with tardiness $\{D|\emptyset\}$, earliness and tardiness $\{R, D(r_i = d_i)|\emptyset\}$, or with soft time windows $\{TW|\emptyset\}$ can be solved in a stand-alone way by means of a minimum idle time policy in $O(n)$, or variants of the PAV algorithm (Section 5.3) in $O(n \log n)$, respectively. Nevertheless, a better complexity can be achieved by means of re-optimization approaches. Ibaraki et al. [87] considers the efficient resolution of the serial timing problem $\{\Sigma c^{\text{CVX}}(t)|\emptyset\}$, and attains an amortized logarithmic complexity per sub-problem when the activity costs are PiL non-negative, lower semicontinuous and convex. The re-optimization data corresponds to the dynamic programming functions described in Section 5.4. These functions are represented as segment pieces within a tree data structure, and computed only on *prefix* and *suffix* subsequences that contain the first or the last activity of the incumbent timing problem.

Data. Optimal cost $\bar{F}(A_i)(t)$ of a schedule for A_i, when the first activity is executed before t, and optimal cost $\bar{B}(A_i)(t)$ of a schedule for A_i, when the last activity is executed after t.

Data computation. $\bar{F}(A_i)(t)$ and $\bar{B}(A_i)(t)$ are computed by means of forward and backward dynamic programming (Equations 19-20), respectively. The use of tree structures for function representations allows for forward and backward extensions in $O(\varphi_c \log \varphi_c)$, where φ_c is the total number of pieces in the cost functions of the timing subproblem.

Evaluate concatenation. Equation (49) returns the optimal cost $Z^*(A_1 \oplus A_2)$ of the timing problem related to the concatenation of two sequences. This value can be computed in $O(\log \varphi_c)$. When the number of pieces of cost functions is linear in the number of activities, as in $\{D|\emptyset\}$, $\{R, D(r_i = d_i)|\emptyset\}$ or $\{TW|\emptyset\}$ settings, a $O(\log n)$ complexity is attained. The concatenation of a bounded number of subsequences can be also efficiently evaluated as in Ibaraki et al. [87].

$$Z^*(A_1 \oplus A_2) = \min_{t \geq 0}\{F(A_1)(t) + B(A_2)(t + p_{A_1(|A_1|),A_2(1)})\} \tag{49}$$

A similar approach can be used for the the more general case with separable PiL cost functions and multiple time-windows, as in Ibaraki et al. [86] and Hendel and Sourd [74]. Simple linked lists can be used in this case to store the functions, and concatenations are evaluated in $O(\varphi_c)$ operations.

Finally, Ergun and Orlin [49] and Kedad-Sidhoum and Sourd [94] have reported methods with an *evaluate concatenation* operator working in amortized $O(1)$ for some specific problems, $\{D|\emptyset\}$ and $\{D, R(d_i = r_i)|NWT\}$, in the presence of particular types of permutation functions (neighborhood search based on swap, insert as well as compound moves), and for sequence-independent processing times. The cornerstone of these two approaches is that they call the functions $B(A_2)(t)$ associated to subsequences by series of $O(n)$ increasing values of t. The methodology requires some ordering, in a pre-processing phase in $O(n \log n)$. This complexity is generally dominated by the number N of timing subproblems. Whether this type of approach can be extended to more general problems, and whether a $O(1)$ re-optimization algorithm exists for $\{D|R\}$ are two current open questions. Approximate procedures may also be used when computational time is critical, as in Taillard et al. [140].

7.5.5 Flexible processing times

The flexible processing time feature involves separable functions of successive activity execution dates. Complex problems are raised when this feature is combined with time-window constraints as in $\{\Sigma c_i(\Delta t_i)|TW\}$ (Equations 50-51), or with separable activity execution costs. The total order constraints can be directly taken into account in the objective when, for $\Delta t < p_{ij}$, $c_{ij}(\Delta t) = +\infty$.

$$\min_{(t_1,\ldots,t_n)\in\Re^{n+}} \quad \sum_{i=1}^{n} c_{\sigma_i\sigma_{i+1}}(t_{\sigma_{i+1}} - t_{\sigma_i}) \tag{50}$$

$$s.t. \quad r_i \leq t_i \leq d_i \qquad 1 \leq i \leq n \tag{51}$$

We first discuss a re-optimization approach for a particular shape of cost functions c_{ij}, introduced by Nagata [108] in the context of VRP with time-windows as an alternative time relaxation. Instead of allowing earliness or lateness with respect to time-window constraints, penalized processing time reductions are allowed. The resulting cost functions are given in Equation (52). In this case, no limit is fixed on the amount of processing time reduction, thus allowing negative processing times.

$$c_{ij}(\Delta t) = \begin{cases} 0 & \text{if } \Delta t \geq p_{ij} \\ \alpha(p_{ij} - \Delta t) & \text{otherwise} \end{cases} \tag{52}$$

Nagata et al. [109] and Hashimoto et al. [71] introduced forward and backward functions to efficiently evaluate the merge of two sequences as a result of some particular VRP neighborhoods. We describe here the re-optimization approach of Vidal et al. [147], which considers any number of concatenations and accounts for duration features. This approach is also related to the method of Kindervater and Savelsbergh [97] for $\{\emptyset|DUR\}$ and $\{\emptyset|DUR, TW\}$.

Data. Minimum duration $D(A_i)$ to perform all the activities of A_i but the last one. Earliest execution date $E(A_i)$ of the first activity in any feasible schedule with minimum idle time for A_i. Latest possible execution date $L(A_i)$ of the first activity in any feasible schedule with minimum processing time reduction for A_i. Minimum processing time reduction $TW(A_i)$ in any feasible schedule for A_i.

Data computation and evaluate concatenation. For a sequence A with a single activity, $D(A) = TW(A) = 0$, $E(A) = r_{A(1)}$ and $L(A) = d_{A(1)}$. Equations (53-58) can be used to obtain this data for concatenated subsequences.

$$E(A_1 \oplus A_2) = \max\{E(A_2) - D(A_1) + TW(A_1) - p_{A_1(|A_1|),A_2(1)}, E(A_1)\} - \delta_{WT} \tag{53}$$

$$L(A_1 \oplus A_2) = \min\{L(A_2) - D(A_1) + TW(A_1) - p_{A_1(|A_1|),A_2(1)}, L(A_1)\} + \delta_{TW} \tag{54}$$

$$D(A_1 \oplus A_2) = D(A_1) + D(A_2) + p_{A_1(|A_1|),A_2(1)} + \delta_{WT} \tag{55}$$

$$TW(A_1 \oplus A_2) = TW(A_1) + TW(A_2) + \delta_{TW} \tag{56}$$

$$\text{with } \delta_{WT} = \max\{E(A_2) - D(A_1) + TW(A_1) - p_{A_1(|A_1|),A_2(1)} - L(A_1), 0\} \tag{57}$$

$$\text{and } \delta_{TW} = \max\{E(A_1) + D(A_1) - TW(A_1) + p_{A_1(|A_1|),A_2(1)} - L(A_2), 0\} \tag{58}$$

Using this approach, the minimum necessary processing time reduction can be evaluated in $O(1)$, for a sequence resulting of a constant number of concatenations. From a computational complexity viewpoint, the flexible processing time relaxation is a good option for local search methods. In contrast, the best re-optimization methods for soft time-window relaxations $\{TW|P\}$ or $\{D|R, P\}$ attain a complexity of $O(\log n)$ per sub-problem (Section 7.5.4).

For the more general case of flexible and sequence-dependent processing-times $\{\Sigma c_i(\Delta t_i), \Sigma c_i(t_i)|\emptyset\}$, Sourd [135] and Hashimoto et al. [70] independently proposed a dynamic programming method similar to Section 7.5.4. The method of Hashimoto et al. [70] is applicable when functions $\tilde{c}_{\sigma_i\sigma_{i+1}}(\Delta t)$ (Equation 31) are convex, and all functions are PiL, lower semicontinuous, non-negative and take infinite

value for $t < 0$. For this setting, an amortized complexity of $O(\varphi_c + \widehat{\varphi_c} \times \widetilde{\varphi_c})$ for each concatenation is attained, where φ_c and $\widetilde{\varphi_c}$ are respectively the total number of pieces in cost functions c_{σ_i} and $\tilde{c}_{\sigma_i\sigma_{i+1}}$, and $\widehat{\varphi_c}$ represents the number of convex pieces in c_{σ_i}.

Finally, $\{DUR|MTW\}$ and $\{\emptyset|DUR, MTW\}$ are special cases of $\{\Sigma c_i(\Delta t_i), \Sigma c_i(t_i)|\emptyset\}$. The re-optimization approach of Hashimoto et al. [70] can thus be applied, leading to an amortized complexity of $O(1+\varphi_{\text{MTW}})$ per subproblem, φ_{MTW} being the total number of time windows. This is an improvement over the previous procedure in $O(n + n\varphi_{\text{MTW}})$ (Section 6.1).

7.5.6 Time-dependent processing times.

Donati et al. [43] address the feasibility problem $\{\emptyset|P(t), TW\}$ with an extension of the method of Savelsbergh [131]. Let $g_{ij}(t) = t + p_{ij}(t)$ be the completion date of an activity i started at t, and followed by activity j. Under the assumption that all $g_{ij}(t)$ are continuous and strictly increasing (any activity started strictly later will finish strictly later), the inverse function $g_{ij}^{-1}(t)$ can be defined and the following re-optimization data and operators enable efficient feasibility checks to be performed.

Data. Earliest possible execution date $E(A_i)$ of the last activity in any feasible schedule for A_i. Latest possible execution date $L(A_i)$ of the first activity in any feasible schedule for A_i.

Data computation. For a sequence A with a single activity, $E(A) = r_{A(1)}$, and $L(A) = d_{A(1)}$. Equations (59-62) return the re-optimization data on prefix and suffix subsequences by forward and backward dynamic programming.

$$E(A_1 \oplus A) = \max\{r_{A(1)}, g_{A_1(|A_1|),A(1)}(E(A_1))\} \tag{59}$$

$$isFeas(A_1 \oplus A) \equiv isFeas(A_1) \ \wedge \ \{E(A_1 \oplus A) \leq d_{A(1)}\} \tag{60}$$

$$L(A \oplus A_2) = \min\{d_{A(1)}, g^{-1}_{A(1),A_2(1)}(L(A_2))\} \tag{61}$$

$$isFeas(A \oplus A_2) \equiv isFeas(A_2) \ \wedge \ \{L(A \oplus A_2) \geq r_{A(1)}\} \tag{62}$$

Evaluate concatenation. Equation (63) can be used to state on the feasibility of any concatenation of a pair of prefix and suffix subsequences.

$$isFeas(A_1 \oplus A_2) \equiv isFeas(A_1) \ \wedge \ isFeas(A_2) \ \wedge \ \{E(A_1) + p_{A_1(|A_1|),A_2(1)}(E(A_1)) \leq L(A_2)\} \tag{63}$$

Assuming the existence of an oracle which evaluates function $g^{-1}(t)$ in $O(1)$, the previous re-optimization framework can be used to check the feasibility of a concatenation of two subsequences in $O(1)$. However, it does not allow to concatenate more than two subsequences, as opposed to the fixed processing time setting treated in Section 7.5.3, thus limiting the range of local search moves that can be evaluated efficiently without relying on a lexicographic search order.

Finally, Hashimoto et al. [71] proposed a dynamic programming approach to manage the general case of PiL time-dependent (and sequence-dependent) processing times with separable execution costs $\{\Sigma c_i(t_i)|P(t)\}$. Evaluations of concatenations of pairs of subsequences A_1 and A_2 are performed in $O(\varphi_c+\varphi_p)$, where φ_c and φ_p denote respectively the total number of pieces in the cost and processing-time functions.

8 Conclusions and Perspectives

In this paper, a rich body of problems with time characteristics and totally ordered variables has been identified and classified. Many algorithms from different research fields were analyzed to identify key problem features and main solution concepts. As timing subproblems frequently arise in the context of local search, both the stand-alone resolution of problems, and the efficient resolution of series of problems by means of re-optimization approaches were analyzed. A general re-optimization

framework based on decomposition and recombination of sequences was introduced, and links to other re-optimization approaches were highlighted.

Table 2 is one key achievement of this analysis. It summarizes the complexity of stand-alone resolution and re-optimization operations for the main timing settings in the literature. The leftmost column lists the problems, while the next block of columns present stand-alone approaches and their complexity. Following to the right, the next columns are dedicated to re-optimization approaches, displaying the complexity of forward and backward data construction, "F/B", the complexity of concatenation of two, "C2", and more than two subsequences, "C3+", if available. Column "Sd" finally indicates whether the mentioned re-optimization approach can address problems with sequence-dependent parameters. Additional assumptions on the problems are listed in the last column.

Thus, a subset of timing methods, originating from various research fields, and constituting the actual state-of-the-art for timing problems with different features, has been identified. These algorithms are a keystone for addressing many rich combinatorial optimization problems with time characteristics, for which the timing sub-problem represents the core of the originality and difficulty. Having a library of timing algorithms at hand opens the way to further developments on more generic solvers that relegate the problem difficulties to known subproblems and methods.

Many promising avenues of research arise as a result of this work. First of all, for several timing features studied in this article, more efficient stand-alone, re-optimization methods and complexity lower bounds should be investigated. In particular, re-optimization has clearly proven useful to address more efficiently several timing settings, reducing in many cases the computational complexity by a factor of n, yet no re-optimization algorithm is available, to our knowledge, for several problems such as $\{DUR|MTW\}$, $\{\emptyset|DUR, MTW\}$, $\{DUR|TW, P(t)\}$, $\{\emptyset|TL, TW\}$ and $\{DUR > D > TL|R\}$.

The impact of sequence dependency on re-optimization is another interesting concern, as sequence dependency constitutes a fundamental delimitation between routing related problems and scheduling settings. Identifying precisely its impact on re-optimization procedures would lead to better insights on local search-based methods for these two important classes of problems. Finally, even if the focus of this paper was on time characteristics, other cumulative resources such as load, stock, energy or workforce lead to similar features and models. The work performed on timing can thus prove useful for an even broader range of applications.

Acknowledgments

Partial funding for this project has been provided by the Champagne-Ardennes regional council, France, the Natural Sciences and Engineering Council of Canada (NSERC), through its Industrial Research Chair and Discovery Grant programs, by our partners CN, Rona, Alimentation Couche-Tard, la Fédération des producteurs de lait du Québec, the Ministry of Transportation of Québec, and by the Fonds québécois de la recherche sur la nature et les technologies (FQRNT) through its Team Research Project program.

References

[1] R.K. Ahuja and J.B. Orlin. A fast scaling algorithm for minimizing separable convex functions subject to chain constraints. *Operations Research*, 49(5):784–789, 2001.

[2] R.K. Ahuja, D.S. Hochbaum, and J.B. Orlin. Solving the convex cost integer dual network flow problem. *Management Science*, 49(7):950–964, 2003.

[3] B. Alidaee and N.K. Womer. Scheduling with time dependent processing times: review and extensions. *Journal of the Operational Research Society*, 50(7):711–720, 1999.

Table 2: Complexity of algorithms and re-optimization operators for common timing problems

Problem	From Scratch		Re-opt. by concat.	F/B	C2	C3+	Sd	Assumptions
Const. act. costs	—	$O(n)$	—	$O(1)$		$O(1)$	✓	
$\{W\|\emptyset\}$	Min idle time	$O(n)$	—	$O(1)$		$O(1)$	✓	
$\{\emptyset\|TW\}$	Min idle time	$O(n)$	Savelsbergh [131] & [97]	$O(1)$		$O(1)$	✓	
$\{D\|\emptyset\}$	Min idle time	$O(n)$	Ergun and Orlin [49]	$O(\log n)$ $O(1)$*		—		penalty coefficient depending upon act.
$\{D, R(d_i = r_i)\|NWT\}$	Min idle time	$O(n)$	Kedad-Sidhoum and Sourd [94]	$O(\log n)$ $O(1)$*		—		penalty coefficient depending upon act.
$\{D, R(d_i = r_i)\|\emptyset\}$	Garey et al. [60] & Ahuja and Orlin [1]	$O(n \log n)$	Ibaraki et al. [87]	$O(\log n)$		$O(\log n)$	✓	
$\{D\|R\}$	Min idle time	$O(n)$	Ibaraki et al. [87]	$O(\log n)$		$O(\log n)$	✓	
$\{\Sigma c_i^{\text{CVX}}(t_i)\|\emptyset\}$	Ibaraki et al. [87]	$O(n \log \varphi_c)$	Ibaraki et al. [87]	$O(\log \varphi_c)$		$O(\log \varphi_c)$	✓	cost f. ≥ 0, PiL & LSC
$\{\Sigma c_i(t_i)\|\emptyset\}$	Ibaraki et al. [86]	$O(n\varphi_c)$	Ibaraki et al. [86]	$O(\varphi_c)$		$O(\varphi_c)$	✓	cost f. ≥ 0, PiL & LSC
$\{\emptyset\|MTW\}$	Min idle time	$O(n \ + \ \varphi_{\text{MTW}})$	Ibaraki et al. [86]	$O(\log \varphi_{\text{MTW}})$		—	✓	
$\{DUR\|TW\}$, $\{\emptyset\|DUR, TW\}$	Malcolm et al. [105]	$O(n)$	Savelsbergh [132] & [97]	$O(1)$		$O(1)$	✓	
$\{DUR\|MTW\}$, $\{\emptyset\|DUR, MTW\}$	Tricoire et al. [144]	$O(n\varphi_{MTW})$	Hashimoto et al. [70]	$O(\varphi_{\text{MTW}})$		—	✓	
$\{\emptyset\|IDL, TW\}$	Hunsaker & Savelsbergh [82]	$O(n)$	—	—		—		
$\{\Sigma c_i^{\text{CVX}}(\Delta t_i)\|C\}$	Hochbaum [77]	$O(n \log(t_{max}/\epsilon n))$	—	—		—		cont. with ϵ precision
$\{\Sigma c_i^{\text{CVX}}(\Delta t_i)\|C, D\}$	Hochbaum [77]	$O(n \log n \log(t_{max}/\epsilon n))$	—	—		—		cont. with ϵ precision
$\{\Sigma c_i^{\text{CVX}}(\Delta t_i)\|C, D\}$	Vidal et al. [149]	$O(n \log m \log(t_{max}/\epsilon))$	—	—		—		cont. with ϵ precision
$\{\Sigma c_i^{\text{CVX}}(\Delta t_i), \Sigma c_i(t_i)\|\emptyset\}$	Sourd [135] & Hashimoto et al. [70]	$O(n(\varphi_c + \widehat{\varphi_c} \times \widehat{\varphi_c}))$	Sourd [135] & Hashimoto et al. [70]	$O(\varphi_c + \widehat{\varphi_c} \times \widehat{\varphi_c})$		—	✓	cost f. ≥ 0, PiL & LSC
$\{D\|R, P(t)\}$	Min idle time	$O(n)$	—	—		—		FIFO assumption
$\{\emptyset\|TW, P(t)\}$	Donati et al. [43]	$O(n)$	Donati et al. [43]	$O(1)$		—	✓	FIFO assumption
$\{\Sigma c_i(t_i)\|P(t)\}$	Hashimoto et al. [71]	$O(n(\varphi_c + \varphi_p))$	Hashimoto et al. [71]	$O(\varphi_c + \varphi_p)$		—	✓	cost f. ≥ 0, PiL & LSC & HYI assumption
$\{\emptyset\|TL, TW\}$	Hurink and Keuchel [83]	$O(n^3)$	—	—		—		
$\{\emptyset\|TL, TW\}$	Firat and Woeginger [54]	$O(n)$	Gschwind and Irnich [66]	O(M)	—	—	✓	$O(n)$ TL constraints, M $\Leftrightarrow$ nb. open deliv.
$\{DUR > D > TL\|R\}$	Cordeau and Laporte [29]	$O(n^2)$	—	—		—		$O(n)$ TL constraints & LIFO assumption

[4] M. Ayer, H.D. Brunk, G.M. Ewing, W.T. Reid, and E. Silverman. An empirical distribution function for sampling with incomplete information. *The Annals of Mathematical Statistics*, 26 (4):641–647, 1955.

[5] K.R. Baker and G.D. Scudder. Sequencing with earliness and tardiness penalties: a review. *Operations Research*, 38(1):22–36, 1990.

[6] N. Balakrishnan. Simple heuristics for the vehicle routeing problem with soft time windows. *Journal of the Operational Research Society*, 44(3):279–287, 1993.

[7] R. Baldacci, A. Mingozzi, and R. Roberti. New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, 59(5):1269–1283, 2011.

[8] R.E. Barlow, D.J. Bartholomew, J.M. Bremner, and H.D. Brunk. *Statistical inference under order restrictions: The theory and application of isotonic regression*. Wiley New York, 1972.

[9] M. Bartusch, R.H. Möhring, and F.J. Radermacher. Scheduling project networks with resource constraints and time windows. *Annals of Operations Research*, 16(1):199–240, 1988.

[10] J.C. Bean, J.R. Birge, and R.L. Smith. Aggregation in dynamic programming. *Operations Research*, 35(2):215–220, 1987.

[11] J.E. Beasley. Adapting the savings algorithm for varying inter-customer travel times. *Omega*, 9 (6):658–659, 1981.

[12] G. Berbeglia, J.-F. Cordeau, and G. Laporte. A hybrid tabu search and constraint programming algorithm for the dynamic dial-a-ride problem. *INFORMS Journal on Computing*, 24(3):343–355, 2012.

[13] D.P. Bertsekas, A. Nedi, and A.E. Ozdaglar. *Nonlinear Programming*. Athena Scientific, Nashua, NH, 2003.

[14] M.J. Best and N. Chakravarti. Active set algorithms for isotonic regression, a unifying framework. *Mathematical Programming*, 47(1-3):425–439, 1990.

[15] M.J Best, N. Chakravarti, and V.A. Ubhaya. Minimizing separable convex functions subject to simple chain constraints. *SIAM Journal on Optimization*, 10(3):658–672, 2000.

[16] L. Bianco, A. Mingozzi, and S. Ricciardelli. The traveling salesman problem with cumulative costs. *Networks*, 23(2):81–91, 1993.

[17] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, and M. Sudan. The minimum latency problem. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 163–171, New York, USA, 1994. ACM.

[18] O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, Part II : Metaheuristics. *Transportation Science*, 39(1):119–139, 2005.

[19] O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, Part I : Route construction and local search algorithms. *Transportation Science*, 39(1):104–118, 2005.

[20] P. Brucker, T. Hilbig, and J. Hurink. A branch and bound algorithm for a single-machine scheduling problem with positive and negative time-lags. *Discrete Applied Mathematics*, 94 (1-3):77–99, 1999.

[21] H.D. Brunk. Maximum likelihood estimates of monotone parameters. *The Annals of Mathematical Statistics*, 26(4):607–616, 1955.

[22] A.M. Campbell and M. Savelsbergh. Efficient insertion heuristics for vehicle routing and scheduling problems. *Transportation science*, 38(3):369–378, 2004.

[23] N. Chakravarti. Isotonic median regression: a linear programming approach. *Mathematics of*

Operations Research, 14(2):303–308, 1989.

[24] A. Charnes and W.W. Cooper. The theory of search: optimum distribution of search effort. *Management Science*, 5(1):44–50, 1958.

[25] T Cheng. A concise survey of scheduling with time-dependent processing times. *European Journal of Operational Research*, 152(1):1–13, 2004.

[26] P. Chrétienne and F. Sourd. PERT scheduling with convex cost functions. *Theoretical Computer Science*, 292(1):145–164, 2003.

[27] M. Christiansen and K. Fagerholt. Robust ship scheduling with multiple time windows. *Naval Research Logistics*, 49(6):611–625, 2002.

[28] K.L. Cooke and E. Halsey. The shortest route through a network with time-dependent internodal transit times. *Journal of Mathematical Analysis and Applications*, 14:493–498, 1966.

[29] J.-F. Cordeau and G. Laporte. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, 37(6):579–594, 2003.

[30] J.-F. Cordeau, G. Laporte, and A. Mercier. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52(8):928–936, 2001.

[31] J.-F. Cordeau, G. Laporte, and A. Mercier. Improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows. *Journal of the Operational Research Society*, 55(5):542– 546, 2004.

[32] R. Cordone. A note on time windows constraints in routing problems. Technical report, Department of Electronics and Information, Polytechnic of Milan, 2000.

[33] G.B. Dantzig. A control problem of Bellman. *Management Science*, 17(9):542–546, 1971.

[34] J.S. Davis and J.J. Kanet. Single-machine scheduling with early and tardy completion costs. *Naval Research Logistics*, 40:85–101, 1993.

[35] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49: 61–95, 1991.

[36] E. Demir, T. Bektas, and G. Laporte. An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research*, 223(2):346–359, 2012.

[37] G. Desaulniers and D. Villeneuve. The shortest path problem with time windows and linear waiting costs. *Transportation Science*, 34(3):312–319, 2000.

[38] G. Desaulniers, J. Desrosiers, I. Ioachim, M.M. Solomon, F. Soumis, and D. Villeneuve. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T.G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 129–154. Kluwer Academic Publishers, Boston, MA, 1998.

[39] G. Desaulniers, J. Desrosiers, and M.M. Solomon, editors. *Column generation*. Springer, 2005.

[40] M. Desrochers, J.K. Lenstra, and M.W.P. Savelsbergh. A classification scheme for vehicle routing and scheduling problems. *European Journal of Operational Research*, 46(3):322–332, 1990.

[41] M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354, 1992.

[42] J. Desrosiers, Y. Dumas, M.M. Solomon, and F. Soumis. Time constrained routing and scheduling. In M. Ball, T. L. Magnanti, C.L. Monma, and G.L. Nemhauser, editors, *Network Routing*, pages 35–139. North-Holland Amsterdam, 1995.

[43] A. Donati, R. Montemanni, N. Casagrande, A.E. Rizzoli, and L.M. Gambardella. Time dependent vehicle routing problem with a multi ant colony system. *European Journal of Operational*

Research, 185(3):1174–1191, 2008.

[44] S.E. Dreyfus. An appraisal of some shortest-path algorithms. *Operations Research*, 17(3):395–412, 1969.

[45] C. Duhamel. *Un Cadre Formel pour les Méthodes par Amélioration Itérative - Application à deux problèmes d'Optimisation dans les réseaux*. PhD thesis, Université Blaise Pascal, Clermont-Ferrand, 2001.

[46] Y. Dumas, F. Soumis, and J. Desrosiers. Optimizing the schedule for a fixed vehicle path with convex inconvenience costs. *Transportation Science*, 24(2):145–152, 1990.

[47] M.E. Dyer and J. Walker. An algorithm for a separable integer programming problem with cumulatively bounded variables. *Discrete applied mathematics*, 16:135–149, 1987.

[48] I. Elhallaoui, D. Villeneuve, F. Soumis, and G. Desaulniers. Dynamic aggregation of set-partitioning constraints in column generation. *Operations Research*, 53(4):632–645, 2005.

[49] O. Ergun and J. Orlin. Fast neighborhood search for the single machine total weighted tardiness problem. *Operations Research Letters*, 34(1):41–45, 2006.

[50] E. Erkut and J. Zhang. The maximum collection problem with time-dependent rewards. *Naval Research Logistics*, 43(5):749–763, 1996.

[51] H. Everett. Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, 11(3):399–417, 1963.

[52] D. Feillet, P. Dejax, and M. Gendreau. Traveling salesman problems with profits. *Transportation Science*, 39(2):188–205, 2005.

[53] G. Feng and H.C. Lau. Efficient algorithms for machine scheduling problems with earliness and tardiness penalties. *Annals of Operations Research*, 159(1):83–95, 2007.

[54] M. Firat and G.J. Woeginger. Analysis of the dial-a-ride problem of Hunsaker and Savelsbergh. *Operations Research Letters*, 39(1):32–35, 2011.

[55] M. Fischetti, G. Laporte, and S. Martello. The delivery man problem and cumulative matroids. *Operations Research*, 41(6):1055–1064, 1993.

[56] B. Fleischmann, M. Gietz, and S. Gnutzmann. Time-varying travel times in vehicle routing. *Transportation Science*, 38(2):160–173, 2004.

[57] A. Frangioni and A. Manca. A computational study of cost reoptimization for min-cost flow problems. *INFORMS Journal on Computing*, 18(1):61–70, 2006.

[58] G.N. Frederickson and D.B. Johnson. The complexity of selection and ranking in X + Y and matrices with sorted columns. *Journal of Computer and System Sciences*, 24(2):197–208, 1982.

[59] M.L. Fredman. On computing the length of longest increasing subsequences. *Discrete Mathematics*, 11(1):29–35, 1975.

[60] M.R. Garey, R.E. Tarjan, and G.T. Wilfong. One-processor scheduling with symmetric earliness and tardiness penalties. *Mathematics of Operations Research*, 13(2):330–348, 1988.

[61] M. Gendreau and C.D. Tarantilis. Solving large-scale vehicle routing problems with time windows: The state-of-the-art. Technical report, CIRRELT, 2010.

[62] B. Giffler and G.L. Thompson. Algorithms for solving production-scheduling problems. *Operations Research*, 8(4):487–503, 1960.

[63] S. Goto and A. Sangiovanni-Vincentelli. A new shortest path updating algorithm. *Networks*, 8 (4):341–372, 1978.

[64] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.

[65] S.J. Grotzinger and C. Witzgall. Projections onto order simplexes. *Applied Mathematics and Optimization*, 270(12):247–270, 1984.

[66] T. Gschwind and S. Irnich. Effective handling of dynamic time windows and synchronization with precedences for exact vehicle routing. Technical report, Johannes Gutenberg University, Mainz, Germany, 2012.

[67] J. Halpern. Shortest route with time dependent length of edges and limited delay possibilities in nodes. *Zeitschrift für Operations Research*, 21(3):117–124, 1977.

[68] R.F. Hartl, G. Hasle, and G.K. Janssens. Special issue on rich vehicle routing problems. *Central European Journal of Operations Research*, 14(2):103–104, 2006.

[69] H. Hashimoto. *Studies on local search-based approaches for vehicle routing and scheduling problems*. PhD thesis, Kyoto University, 2008.

[70] H. Hashimoto, T. Ibaraki, S. Imahori, and M. Yagiura. The vehicle routing problem with flexible time windows and traveling times. *Discrete Applied Mathematics*, 154(16):2271–2290, 2006.

[71] H. Hashimoto, M. Yagiura, and T. Ibaraki. An iterated local search algorithm for the time-dependent vehicle routing problem with time windows. *Discrete Optimization*, 5(2):434–456, 2008.

[72] H. Hashimoto, M. Yagiura, S. Imahori, and T. Ibaraki. Recent progress of local search in handling the time window constraints of the vehicle routing problem. *4OR*, 8(3):221–238, 2010.

[73] D. Haugland and S.C. Ho. Feasibility testing for dial-a-ride problems. In B. Chen, editor, *Algorithmic Aspects in Information and Management*, volume 6124 of *LNCS*, pages 170–179. Springer Berlin / Heidelberg, 2010.

[74] Y. Hendel and F. Sourd. Efficient neighborhood search for the one-machine earliness-tardiness scheduling problem. *European Journal of Operational Research*, 173(1):108–119, 2006.

[75] Y. Hendel and F. Sourd. An improved earliness-tardiness timing algorithm. *Computers & Operations Research*, 34(10):2931–2938, 2007.

[76] D. Hochbaum. Solving integer programs over monotone inequalities in three variables: A framework for half integrality and good approximations. *European Journal of Operational Research*, 140(2):291–321, 2002.

[77] D.S. Hochbaum. Lower and upper bounds for the allocation problem and other nonlinear optimization problems. *Mathematics of Operations Research*, 19(2):390–409, 1994.

[78] D.S. Hochbaum and S.-P. Hong. About strongly polynomial time algorithms for quadratic optimization over submodular constraints. *Mathematical Programming*, 69(1-3):269–309, 1995.

[79] D.S. Hochbaum and J.G. Shanthikumar. Convex separable optimization is not much harder than linear optimization. *Journal of the ACM (JACM)*, 37(4):843–862, 1990.

[80] J.A. Hoogeveen and S.L. Van De Velde. A branch-and-bound algorithm for single-machine earliness-tardiness scheduling with idle time. *INFORMS Journal on Computing*, 8(4):402–412, 1996.

[81] H.H. Hoos and T. Stützle. *Stochastic local search: Foundations and applications*. Morgan Kaufmann, 2005.

[82] B. Hunsaker and M.W.P. Savelsbergh. Efficient feasibility testing for dial-a-ride problems. *Op-*

erations Research Letters, 30(3):169–173, 2002.

[83] J. Hurink and J. Keuchel. Local search algorithms for a single-machine scheduling problem with positive and negative time-lags. *Discrete Applied Mathematics*, 112(1-3):179–197, 2001.

[84] L.M. Hvattum, I. Norstad, K. Fagerholt, and G. Laporte. Analysis of an exact algorithm for the vessel speed optimization problem. *Networks*, 62(2):132–135, 2013.

[85] T. Ibaraki and N. Katoh. *Resource allocation problems: algorithmic approaches*. MIT Press, Boston, MA, 1988.

[86] T. Ibaraki, S. Imahori, M. Kubo, T. Masuda, T. Uno, and M. Yagiura. Effective local search algorithms for routing and scheduling problems with general time-window constraints. *Transportation Science*, 39(2):206–232, 2005.

[87] T. Ibaraki, S. Imahori, K. Nonobe, K. Sobue, T. Uno, and M. Yagiura. An iterated local search algorithm for the vehicle routing problem with convex time penalty functions. *Discrete Applied Mathematics*, 156(11):2050–2069, 2008.

[88] S. Ichoua, M. Gendreau, and J.Y. Potvin. Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research*, 144(2):379–396, 2003.

[89] I. Ioachim, S. Gélinas, F. Soumis, and J. Desrosiers. A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. *Networks*, 31(3):193–204, 1998.

[90] S. Irnich. A unified modeling and solution framework for vehicle routing and local search-based metaheuristics. *INFORMS Journal on Computing*, 20(2):270–287, 2008.

[91] S. Irnich. Resource extension functions: properties, inversion, and generalization to segments. *OR Spectrum*, 30:113–148, 2008.

[92] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4): 373–395, 1984.

[93] W. Karush. A general algorithm for the optimal distribution of effort. *Management Science*, 9 (1):50–72, 1962.

[94] S. Kedad-Sidhoum and F. Sourd. Fast neighborhood search for the single machine earliness-tardiness scheduling problem. *Computers & Operations Research*, 37(8):1464–1471, 2010.

[95] J.E. Kelley and M.R. Walker. Critical-path planning and scheduling. In *Proceedings of Eastern joint Computer conference*, pages 160–173, New York, 1959. ACM Press.

[96] L.G. Khachiyan. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20:191–194, 1979.

[97] G.A.P. Kindervater and M.W.P. Savelsbergh. Vehicle routing: Handling edge exchanges. In E.H.L. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 337–360. Princeton Univ Pr, 1997.

[98] M.S. Kodialam and H. Luss. Algorithms for separable nonlinear resource allocation problems. *Operations Research*, 46(2):272–284, 1998.

[99] A.L. Kok, E.W. Hans, and J.M.J. Schutten. Vehicle routing under time-dependent travel times: the impact of congestion avoidance. Technical report, University of Twente, 2009.

[100] Y.A. Koskosidis, W.B. Powell, and M.M. Solomon. An optimization-based heuristic for vehicle routing and scheduling with soft time window constraints. *Transportation Science*, 26(2):69–85, 1992.

[101] C.Y. Lee and J.Y. Choi. A genetic algorithm for job sequencing problems with distinct due dates and general early-tardy penalty weights. *Computers & Operations Research*, 22(8):857–

869, 1995.

[102] H. Luss and S.K. Gupta. Allocation of effort resources among competing activities. *Operations Research*, 23(2):360–366, 1975.

[103] C. Malandraki and M.S. Daskin. Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation Science*, 26(3):185–200, 1992.

[104] C. Malandraki and R.B. Dial. A restricted dynamic programming heuristic algorithm for the time dependent traveling salesman problem. *European Journal of Operational Research*, 90(1): 45–55, 1996.

[105] D.G. Malcolm, J.H. Roseboom, C.E. Clark, and W. Fazar. Application of a technique for research and development program evaluation. *Operations Research*, 7(5):646–669, 1959.

[106] E. Miller-Hooks and B. Yang. Updating paths in time-varying networks given arc weight changes. *Transportation Science*, 39(4):451–464, 2005.

[107] L.G. Mitten. Sequencing n jobs on two machines with arbitrary time lags. *Management Science*, 5(3):293–298, 1959.

[108] Y. Nagata. Effective memetic algorithm for the vehicle routing problem with time windows: Edge assembly crossover for the VRPTW. In *Proceedings of the Seventh Metaheuristics International Conference, Montreal, Canada (on CD-ROM)*, 2007.

[109] Y. Nagata, O. Bräysy, and W. Dullaert. A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research*, 37(4): 724–737, 2010.

[110] S.U. Ngueveu, C. Prins, and R. Wolfler Calvo. An effective memetic algorithm for the cumulative capacitated vehicle routing problem. *Computers & Operations Research*, 37(11):1877–1885, 2010.

[111] I. Norstad, K. Fagerholt, and G. Laporte. Tramp ship routing and scheduling with speed optimization. *Transportation Research Part C: Emerging Technologies*, 19(5):853–865, 2011.

[112] I. Or. *Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking*. PhD thesis, Northwestern University, Evanston, IL, 1976.

[113] A. Padakandla and R. Sundaresan. Power minimization for CDMA under colored noise. *IEEE Transactions on Communications*, 57(10):3103–3112, 2009.

[114] S. Pallottino. A new algorithm for reoptimizing shortest paths when the arc costs change. *Operations Research Letters*, 31(2):149–160, 2003.

[115] Y. Pan and L. Shi. Dual constrained single machine sequencing to minimize total weighted completion time. *IEEE Transactions on Automation Science and Engineering*, 2(4):344–357, 2005.

[116] P.M. Pardalos and G. Xue. Algorithms for a class of isotonic regression problems. *Algorithmica*, 23(3):211–222, 1999.

[117] P.M. Pardalos, G.-L. Xue, and L. Yong. Efficient computation of an isotonic median regression. *Applied Mathematics Letters*, 8(2):67–70, 1995.

[118] S.N. Parragh, J.-F. Cordeau, K.F. Doerner, and R.F. Hartl. Models and algorithms for the heterogeneous dial-a-ride problem with driver-related constraints. *OR Spectrum*, 34(3):593–633, 2012.

[119] M. Patriksson. A survey on the continuous nonlinear resource allocation problem. *European Journal of Operational Research*, 185(1):1–46, 2008.

[120] D.W. Pentico. The assortment problem: A survey. *European Journal of Operational Research*,

190(2):295–309, 2008.

[121] M. Pinedo. *Scheduling: theory, algorithms, and systems*. Prentice Hall, 2008.

[122] C.N. Potts and J.D. Whitehead. Heuristics for a coupled-operation scheduling problem. *Journal of the Operational Research Society*, 58(10):1375–1388, 2007.

[123] J.-Y. Potvin and J.-M. Rousseau. An exchange heuristic for routeing problems with time windows. *Journal of the Operational Research Society*, 46(12):1433–1446, 1995.

[124] E. Prescott-Gagnon, G. Desaulniers, and L.M. Rousseau. A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. *Networks*, 54(4): 190–204, 2009.

[125] G. Righini and M. Salani. Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3): 255–273, 2006.

[126] T. Robertson, F.T. Wright, and R.L. Dykstra. *Order Restricted Statistical Inference*. Wiley Series in Probability and Statistics. John Wiley & Sons, New York, 1988.

[127] R.T. Rockafellar. *Convex analysis*. Princeton Univ Press, 1970.

[128] B. Roy. Contribution de la théorie des graphes à l'étude de certains problèmes linéaires. *Comptes rendus de l'académie des sciences de Paris*, 248:2437–2439, 1959.

[129] B. Roy. Graphes et ordonnancement. *Revue Française de Recherche Opérationnelle*, 25:323–333, 1962.

[130] L. Sanathanan. On an allocation problem with multistage constraints. *Operations Research*, 19 (7):1647–1663, 1971.

[131] M.W.P. Savelsbergh. Local search in routing problems with time windows. *Annals of Operations Research*, 4(1):285–305, 1985.

[132] M.W.P. Savelsbergh. The vehicle routing problem with time windows: Minimizing route duration. *ORSA Journal on Computing*, 4(2):146–154, 1992.

[133] T.R. Sexton and L.D. Bodin. Optimizing single vehicle many-to-many operations with desired delivery times: I. Scheduling. *Transportation Science*, 19(4):378–410, 1985.

[134] T.R. Sexton and L.D. Bodin. Optimizing single vehicle many-to-many operations with desired delivery times: II. Routing. *Transportation Science*, 19(4):411–435, 1985.

[135] F. Sourd. Optimal timing of a sequence of tasks with general completion costs. *European Journal of Operational Research*, 165(1):82–96, 2005.

[136] F. Sourd and S. Kedad-Sidhoum. The one-machine problem with earliness and tardiness penalties. *Journal of Scheduling*, 6(6):533–549, 2003.

[137] K.S. Srikantan. A problem in optimum allocation. *Operations Research*, 11(2):265–273, 1963.

[138] W. Szwarc and S.K. Mukhopadhyay. Optimal timing schedules in earliness-tardiness single machine sequencing. *Naval Research Logistics*, 42(7):1109–1114, 1995.

[139] M. Tagmouti, M. Gendreau, and J.-Y. Potvin. Arc routing problems with time-dependent service costs. *European Journal of Operational Research*, 181(1):30–39, 2007.

[140] E. Taillard, P. Badeau, M. Gendreau, F. Guertin, and J.-Y. Potvin. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31(2):170–186, 1997.

[141] F.B. Talbot. Resource-constrained project scheduling with time-resource tradeoffs: the nonpre-

emptive case. *Management Science*, 28(10):1197–1210, 1982.

[142] A. Tamir. Efficient algorithms for a selection problem with nested constraints and its application to a production-sales planning model. *SIAM Journal on Control and Optimization*, 18(3):282–287, 1980.

[143] J. Tang, Y. Kong, H. Lau, and A.W.H. Ip. A note on "Efficient feasibility testing for dial-a-ride problems". *Operations Research Letters*, 38(5):405–407, 2010.

[144] F. Tricoire, M. Romauch, K.F. Doerner, and R.F. Hartl. Heuristics for the multi-period orienteering problem with multiple time windows. *Computers & Operations Research*, 37(2):351–367, 2010.

[145] T. Van Woensel, L. Kerbache, H. Peremans, and N. Vandaele. Vehicle routing with dynamic travel times: a queueing approach. *European Journal of Operational Research*, 186(3):990–1007, 2008.

[146] T. Vidal, T.G. Crainic, M. Gendreau, and C. Prins. Heuristics for multi-attribute vehicle routing problems: a survey and synthesis. *European Journal of Operational Research*, 231(1):1–21, 2013.

[147] T. Vidal, T.G. Crainic, M. Gendreau, and C. Prins. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, 40(1):475–489, 2013.

[148] T. Vidal, T.G. Crainic, M. Gendreau, and C. Prins. Time-window relaxations in vehicle routing heuristics. Technical report, CIRRELT, Montréal, 2013.

[149] T. Vidal, P. Jaillet, and N. Maculan. A decomposition algorithm for nested resource allocation problems. Technical report, MIT, Cambridge, 2014.

[150] C. Walshaw. A multilevel approach to the travelling salesman problem. *Operations Research*, 50(5):862–877, 2002.

[151] G. Wan and B.P.-C. Yen. Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties. *European Journal of Operational Research*, 142(2): 271–281, 2002.

[152] C.A. Yano and Y.-D. Kim. Algorithms for a class of single-machine weighted tardiness and earliness problems. *European Journal of Operational Research*, 52(2):167–178, 1991.

Appendix

A Reduction of LISP to $\{TW(unit)|P\}$

Given a vector $\mathbf{N} = (N_1, \ldots, N_n)$ of n real numbers, LISP aims to find the maximum length L of a non-decreasing subsequence of numbers: $L = \max\{k|1 \leq i_1 < \cdots < i_k \leq n \text{ and } N_{i_1} \leq \cdots \leq N_{i_k}\}$. From a LISP instance, we construct the following instance $\mathcal{T}$ of $\{TW(unit)|\emptyset\}$, with n activities such that for $i \in \{1, \ldots, n\}$, $r_i = d_i = N_i$ and $p_i = 0$. This instance is created in n elementary algorithmic operations.

Let $z^*(\mathcal{T})$ be the optimal solution cost of $\mathcal{T}$. This solution naturally initiates as many activities as possible without penalties, within a non-decreasing order of execution dates. Hence, the activities achieved without penalty correspond to the LISP subsequence sought in the original problem, whose length is $L^* = n - z^*(\mathcal{T})$. Hence, LISP admits a many-one reduction to $\{TW(unit)|\emptyset\}$. Conversely, $\{TW(unit)|\emptyset\}$ generalizes LISP.

B Proof of Theorem 1: Block optimality conditions of $\{\Sigma c_i^{\mathbf{cvx}}(t_i)|\emptyset\}$

We first recall the corresponding timing problem, stated in Equations (64-65).

$$\min_{(t_1,\dots,t_n)\in\Re^n} \quad \sum_{i=1}^{n} c_i(t_i) \tag{64}$$

$$s.t. \quad t_i + p_i \leq t_{i+1} \qquad 1 \leq i < n \tag{65}$$

The functions $c_i(t) : \Re \to \Re$ are assumed to be convex, but not necessarily smooth. We denote by $\delta c_i(t)$ the subdifferential of c_i at t, which is necessarily a non-empty interval, as a byproduct of the convexity assumption and the space of definition. We first recall two useful properties on subdifferentials from Rockafellar [127], to follow with the proof of Theorem (1).

Proposition 2. *Let $f_1, \dots, f_m$ be subdifferentiable functions on $\Re^n$, then:*

$$\delta(f_1(x) + \cdots + f_m(x)) \supset \delta f_1(x) + \cdots + \delta f_m(x) \ \forall x \tag{66}$$

Theorem 2. *Let $f_1, \dots, f_m$ be a set of proper convex functions on $\Re^n$ having at least one common point in the relative interior of their domains $ri(dom(f_i))$, then:*

$$\delta(f_1(x) + \cdots + f_m(x)) = \delta f_1(x) + \cdots + \delta f_m(x) \ \forall x \in \bigcap ri(dom(f_i)) \tag{67}$$

Constraint qualifications hold as the constraints are linear, and any solution with idle time between each activity is feasible, thus taking place in the relative interior of the polytope. Hence, strong duality applies, leading to the following necessary and sufficient optimality conditions. A solution $\mathbf{t}^* = (t_1^*, \dots, t_n^*)$ of Problem (64-65) is optimal if and only if a set of Lagrangian multipliers $(\lambda_1^*, \dots, \lambda_{n-1}^*)$ exists, such that conditions (68) are satisfied (Bertsekas et al. 13, Propositions 5.7.1 and 6.4.2).

$$\begin{cases} t_{i-1}^* + p_{i-1} \leq t_i^* & i = 2, \dots, n \\ 0 \in \delta c_1(t_1^*) + \lambda_1^* & \\ 0 \in \delta c_i(t_i^*) + \lambda_i^* - \lambda_{i-1}^* & i = 2, \dots, n-1 \\ 0 \in \delta c_n(t_n^*) - \lambda_{n-1}^* & \\ \lambda_{i-1}^*(t_{i-1}^* + p_{i-1} - t_i^*) = 0 & i = 2, \dots, n \\ \lambda_i^* \geq 0 & i = 1, \dots, n \end{cases} \tag{68}$$

Solution $\mathbf{t}^*$ can be represented as a succession of blocks of activities $(B_1, \dots, B_m)$, such that activities within each block are processed without idle time, and the last activities of blocks are followed by non-zero idle time. The previous definition, combined with primal feasibility, yields $t_{B_j(|B_j|)}^* + p_{B_j(|B_j|)} < t_{B_j(|B_j|)+1}^*$, and thus $\lambda_{B_j(|B_j|)}^* = 0$ for $j \in \{1, \dots, m-1\}$. Conditions (68) are thus equivalent to primal feasibility (equivalent to the first condition of (1) combined with the definition of blocks) and the following independent systems of equations for each block:

$$\forall j \in \{1, \dots, m\} \begin{cases} \textbf{i)} & 0 \in \delta c_{B_j(1)}(t_{B_j(1)}^*) + \lambda_{B_j(1)}^* & \\ \textbf{ii)} & 0 \in \delta c_i(t_{B_j(1)}^* + p_{B_j(1)i-1}) + \lambda_i^* - \lambda_{i-1}^* & i = B_j(1)+1, \dots, B_j(|B_j|)-1 \\ \textbf{iii)} & 0 \in \delta c_{B_j(|B_j|)}(t_{B_j(1)}^* + p_{B_j(1)B_j(|B_j|-1)})) - \lambda_{B_j(|B_j|)-1}^* & \\ \textbf{iv)} & \lambda_i^* \geq 0 \qquad i = B_j(1), \dots, B_j(|B_j|) & \end{cases} \tag{69}$$

Necessary condition proof. For $1 \leq i \leq j \leq n$, let p_{ij} be the cumulative processing duration of activities a_i to a_j. Relying on Proposition (2), one can sum i) , ii) and iii) of (69), leading to:

$$0 \in \delta c_{B_j(1)}(t^*_{B_j(1)}) + \sum_{i=B_j(1)+1}^{B_j(|B_j|)} \delta c_i(t^*_{B_j(1)} + p_{B_j(1)i-1}) \Rightarrow 0 \in \delta C_{B_j}(t^*_{B_j(1)}) \ , \tag{70}$$

and thus, following the definition of the optimal block execution cost of Equation (18), $t^*_{B_i(1)} \in [T^{-*}_{B_i}, T^{+*}_{B_i}]$. Any optimal solution thus verifies the first statement of Theorem (1). Finally, for any block B_j and prefix block B^k_j, summing i) , ii) and iii) for $j \in \{B_j(1)+1, \ldots, k\}$ leads to:

$$-\lambda^*_k \in \delta c_{B_j(1)}(t^*_{B_j(1)}) + \sum_{i=B_j(1)+1}^{k} \delta c_i(t^*_{B_j(1)} + p_{B_j(1)i-1}) \Rightarrow -\lambda^*_k \in \delta C_{B_j}(t^*_{B^k_j(1)}) \ , \tag{71}$$

which can be reformulated as $T^{+*}_{B^k_j} \geq t^*_{B_j(1)}$ and implies the last statement of Theorem (1).

Sufficient condition proof. Consider a solution $\mathbf{t} = (t_1, \ldots, t_n)$ with its blocks $(B_1, \ldots, B_m)$, respecting conditions of Theorem (1). Following block definitions and primal feasibility, it only remains to prove that Conditions (69) are respected for each block. We choose the following Lagrangian multipliers, which are non-negative as $T^{+*}_{B^k_i} \geq t_{B_i(1)} \Rightarrow \exists x \ \leq 0 \in \delta C_{B^i_j}(t_{B_j(1)}))$:

$$\forall j \in \{1, \ldots, m\} \begin{cases} \lambda^*_i = -\min(x \in \delta C_{B^i_j}(t_{B_j(1)})) & i = B_j(1)+1, \ldots, B_j(|B_j|)-1 \\ \lambda^*_{B_j(|B_j|)} = 0 \end{cases} \tag{72}$$

Proposition (2) then involves that for $i \in \{1, \ldots, m\}$ and $j \in \{B_j(1)+1, \ldots, B_j(|B_j|)-1\}$;

$$-\lambda^*_i \in \delta C_{B^i_j}(t_{B_j(1)}) = \delta c_{B_j(1)}(t_{B_j(1)}) + \sum_{k=B_j(1)+1}^{i} \delta c_k(t_{B_j(1)} + p_{B_j(1)k-1}) \tag{73}$$

In the case where $i = B_j(1)$, Equation (73) proves statement i) of (69). Also, $\lambda^*_{B_j(1)} \in \delta(-c_{B_j(1)})(t_{B_j(1)})$, and we can combine this statement with Equation (73) for $i = B_j(1)+1$, using Proposition 2), leading to:

$$\begin{aligned} \lambda^*_{B_j(1)+1} - \lambda^*_{B_j(1)} &\in \delta(-c_{B_j(1)})(t_{B_j(1)}) + \delta c_{B_j(1)}(t_{B_j(1)}) + \delta c_{B_j(1)+1}(t_{B_j(1)} + p_{B_j(1)}) \\ &= \delta c_{B_j(1)+1}(t_{B_j(1)} + p_{B_j(1)}) \end{aligned} \tag{74}$$

The remaining statements of Equation ii) and Equation iii) in (69) are proven by recurrence. Assuming that for a given $i \in \{B_j(1)+1, \ldots, B_j(|B_j|)-1\}$, $\lambda^*_{i-1} - \lambda^*_i \in \delta c_i(t_{B_j(1)} + p_{B_j(1)i})$, then

$$\begin{aligned} \lambda^*_i - \lambda^*_{i+1} &= -\lambda^*_{i+1} + \lambda^*_{i-1} - (\lambda^*_{i-1} - \lambda^*_i) \\ &\in \delta c_{B_j(1)}(t_{B_j(1)}) + \sum_{k=B_j(1)}^{i+1} \delta c_k(t_{B_j(1)} + p_{B_j(1)k-1}) + \delta(-c_{B_j(1)})(t_{B_j(1)}) \\ &+ \sum_{k=B_j(1)}^{i-1} \delta(-c_k)(t_{B_j(1)} + p_{B_j(1)k-1}) + \delta(-c_i)(t_{B_j(1)} + p_{B_j(1)i-1}) \\ &\subset \delta c_{i+1}(t_{B_j(1)} + p_{B_j(1)i}) \end{aligned} \tag{75}$$

All the sufficient optimality conditions are thus satisfied by solution $\mathbf{t} = (t_1, \ldots, t_n)$. □