2014/16

■

Efficient approximation algorithms for the Economic Lot-Sizing in continuous time

Claudio Telha and Mathieu Van Vyve

DISCUSSION PAPER

Center for Operations Research
and Econometrics

Voie du Roman Pays, 34
B-1348 Louvain-la-Neuve
Belgium
http://www.uclouvain.be/core

CORE DISCUSSION PAPER

2014/16

Efficient approximation algorithms for the Economic Lot-Sizing in continuous time.

Claudio Telha [1] and Mathieu Van Vyve[2]

March 2014

Abstract

We consider a continuous-time variant of the classical Economic Lot-Sizing (**ELS**) problem. In this model, the setup cost is a continuous function with lower bound $K_{\min} > 0$, the demand and holding costs are integrable functions of time and the replenishment decisions are not restricted to be multiples of a base period. Starting from the assumption that certain operations involving the setup and holding cost functions can be carried out efficiently, we argue that this variant admits a simple approximation scheme based on dynamic programming: if the optimal cost of an instance is **OPT**, we can find a solution with cost at most $(1+\epsilon)\mathbf{OPT}$ using no more than $O\left(\frac{1}{\epsilon^2}\frac{\mathbf{OPT}}{K_{\min}}\log\frac{\mathbf{OPT}}{K_{\min}}\right)$ of these operations. We argue, however, that this algorithm could be improved on instances where the setup costs are "generally" very large compared with $K_{\min}$. This leads us to introduce a notion of input-size σ that is significantly smaller than $\mathbf{OPT}/K_{\min}$ on instances of this type, and then to define an approximation scheme that executes $O\left(\frac{1}{\epsilon^2}\sigma^2\log^2\left(\frac{\mathbf{OPT}}{K_{\min}}\right)\right)$ operations. Besides dynamic programming, this second approximation scheme builds on a novel algorithmic approach for Economic Lot Sizing problems.

[1]Center for Operations Research and Econometrics (CORE), Catholic University of Louvain (UCL), 34 Voie du Roman Pays, 1348 Louvain-la-Neuve, Belgium; e-mail: claudio.telha@uclouvain.be. Supported by FSR Incoming Post-doctoral Fellowship of the Catholic University of Louvain (UCL), funded by the French Community of Belgium.

[2]Center for Operations Research and Econometrics (CORE), Catholic University of Louvain (UCL), 34 Voie du Roman Pays, 1348 Louvain-la-Neuve, Belgium; e-mail: mathieu.vanvyve@uclouvain.be

This text presents research results of the Belgian Program on Interuniversity Poles of Attraction initiated by the Belgian State, Prime Ministers Office, Science Policy Pro- gramming. The scientific responsibility is assumed by the authors.

1 Introduction

Economic Order Quantity (**EOQ**) and Economic Lot-Sizing (**ELS**) are two classical problems in inventory management [Harris(1990), Wilson(1934), Wagner and Whitin(1958)]. In both problems the cost decomposes into a fixed ordering/production cost that is charged each time there is a production order, and the holding costs that are charged proportional to the stock level. The main trade-off in **EOQ** and **ELS** is between low setup costs (few batches) and low holding costs (many batches or just-in-time production). This captures a key aspect of most production planning problems. The two problems differ by some key aspects: **EOQ** assumes constant cost functions and demand rate on a continuous-time infinite time horizon and admits an analytical solution while **ELS** assumes time-varying costs and demand in a finite discretized planning horizon and is solved by dynamic programming.

In this work, we consider the natural generalization of both **EOQ** and **ELS**: we handle time-varying cost and demand rates in continuous time in a finite planning horizon. In practice, considering decisions in continuous time is becoming more and more realistic as production processes become fully automated. Moreover, in industrial context, discetization of time in **ELS** is typically made before input data is known. As will appear below, our algorithm considers only a finite set of possible times for placing setups, implicitly building a discretization of the time horizon. Therefore, a good discretization of the time horizon is part of the decisions made by our algorithm.

Formally the input of the problem we consider is given by the following objects: an integrable *demand function* $d : [0, T] \to \mathbb{R}^+$, an integrable *holding cost function* $h : [0, T] \to \mathbb{R}^+$, and a continuous *setup cost function* $K : [0, T] \to [K_{\min}, K_{\max}]$ with a strictly positive lower bound $K_{\min} > 0$. The interval $[0, T]$ is called the *planning horizon*. A feasible solution for this problem is any finite sequence of strictly increasing setups $\{s_i\}_{i=1..n} \subseteq [0, T]$ with $s_1 = 0$. Because h is non-negative, it is always optimal to produce as late as possible given the setup sequence; therefore the setup times implicitly partition the time horizon into n *windows* $[s_i, s_{i+1})$ (for convenience, $s_{n+1} \equiv T$), where the inventory required to fulfill demands in $[s_i, s_{i+1})$ is ordered at s_i. The order at s_i incurs a total holding cost of

$$H_i \equiv \int_{s_i}^{s_{i+1}} h(t) D_s(t) dt, \quad \text{where} \quad D_s(t) = \int_t^{s_{i+1}} d(u) du.$$

and a setup cost of

$$K_i \equiv K(s_i).$$

The objective of the **ELS** problem is to find a feasible solution $\{s_i\}_{i=1..n}$ minimizing the total cost $\sum_{i=1}^{n} (K_i + H_i)$. The restriction to $K(\cdot) \geq K_{\min} > 0$ ensures that this minimum always exists. In the remaining of the paper, we will assume without loss of generality that $K_{min} = 1$.

Massonnet [Massonnet(2013)] is the only work we are aware of to give an approximation guarantee for the continuous-time **ELS** problem with general time-varying demand rate. Assuming a constant setup cost function K, he shows that balancing setup and holding cost in each window yields a 2-approximation algorithm. We dramatically improve this result by describing efficient approximation *schemes* while removing the constant setup cost assumption.

Summary. In Section 2, we define several basic notions that turn out to be non-trivial. More specifically, we discuss the complexity model we use, then the associated oracles and finally define the size of an instance σ that we will use. We also deal with some notational issues.

In Section 3, we give a simple approximation scheme based on dynamic programming. It finds an $(1 + \epsilon)$-approximated solution using $O\left(\frac{1}{\epsilon^2} \mathbf{OPT} \log \mathbf{OPT}\right)$ oracle calls. As discussed above, this

is efficient if $K_{max} = O(1)$, but not in general. In Section 4, we describe a constant approximation algorithm that performs $O(\sigma \log(\mathbf{OPT}))$ oracle calls. This algorithm significantly departs from the standard approaches to solve **ELS** in discrete time, and is built on new ideas that could be of independent interest.

In Section 5, we combine the ideas from the algorithms introduced in sections 3 and 4 in order to obtain an approximation scheme where the number of oracle calls is bounded by a function of σ. It performs $O\left(\frac{1}{\epsilon^2}\sigma^2 \log^2(\mathbf{OPT})\right)$ oracle calls to obtain a $(1+\epsilon)$ approximation. Finally, in Section 6, we briefly discuss how the measure σ can be strengthened, by defining yet a stronger parameter $\overline{\sigma} \leq \sigma$, and a constant approximation algorithm that executes $O(\overline{\sigma} \log \mathbf{OPT})$ oracle calls.

2 Preliminaries

2.1 Complexity of the problem

In the discrete setting (that is, when demand and holding cost functions are replaced by sequences) the theoretical notion of efficient algorithm is that of a polynomial time algorithm. Wagner and Whitin [Wagner and Whitin(1958)] solved the discrete **ELS** in time quadratic in the number of periods; more recently, Wagelmans et al. [Wagelmans et al.(1992)Wagelmans, van Hoesel, and Kolen, Federgruen and Tzur(1991)] reduced the running time from quadratic to linear. Because reading the input data already requires linear time, the algorithm of Wagelmans et al. is considered essentially best possible in terms of execution time.

In the continuous setting the situation is blurrier. For some problems involving continuous inputs, the theoretical computational time to solve them can still be bounded by a reasonable function of the input description. But often this is not the case, and ad-hoc approaches are used to justify, in theory, the efficiency of the proposed procedures to solve them. An example of these approaches is called information based complexity [Traub and Werschulz(1998), Sikorski(1985), Plaskota(1996)], where we impose conditions on the input so that an approximate solution can be found via some discretization of the input. This is used in numerical analysis, where sufficiently regular functions can, among several operations, be approximately integrated using finitely many sampled function values [Traub and Werschulz(1998)].

In this paper, we describe approximation algorithms for the continuous-time **ELS** problem that are efficient under a model of computation specifically tailored to the characteristics of this problem. It is based on the *oracle model* from computer science [Papadimitriou(1994)], that takes the computation of certain complex operations for granted; an algorithm executes these operations by calling an hypothetical procedure called the oracle, but in the total running time it is only the number of calls to the oracle that counts.

In the oracle model, the existence of efficient algorithms for a problem is subjugated to the existence of an efficient method for implementing the oracle. Because the operations encapsulated by the oracle are often deemed necessary for most thinkable algorithms for the problem, there is usually a weak, informal, correspondence between efficient algorithms for a problem and efficient implementations of the corresponding oracle. For example, in computational geometry, oracles are used to test membership to certain geometric sets (e.g. for volume estimation of convex bodies [Lovász and Vempala(2006)]); in combinatorial optimization, oracles are used to test membership to certain collection of subsets (e.g. for computing a maximum-weight independent set in a matroid [Schrijver(2003)]); in continuous optimization, oracles are used to determine any pointwise gradient or Hessian (e.g. for solving an optimization problem using any the gradient-based algorithm [Boyd and Vandenberghe(2004)]).

2.2 Oracles

Through the paper, we consider the two following oracles:

Oracle 1. For any $s \in [0,T]$ and $c > 0$: the last time x where $K(x) + \int_s^x h(t)D_s(t)dt \leq c$. For convenience, we will denote $x \equiv S(s,c)$.

Oracle 2. for any $s \in [0,T]$: the earliest time $x > s$ where $K(x) \leq K(s)/2$, or ∞ if no such x exists.

We will also need to evaluate $K(s)$ for any s, and the accumulated holding cost $H_{[s,s')} \equiv \int_s^{s'} h(t)D_s(t)dt$ in any interval $[s,s']$. We consider these as basic operations.

Oracles 1 and 2 are non-trivial one-dimensional optimization problems themselves of the form $\min_{x\in[a,b]}\{x : f(x)/leqc\}$ for $f(x)$ continuous and $f(a) > c$. Note that assuming we can solve $\min_{x\in[a,b]} f(x)$ for any $[a,b]$, then one can efficiently solve the former problem to arbitrary precision by binary search as follows. Solve $x^* = \min_{x\in[a,\frac{a+b}{2}]} f(x)$. If $x^* \leq c$, then the solution lies in $[a,x^*] \subseteq [a,\frac{a+b}{2}]$, otherwise the solution lies in $[\frac{a+b}{2},b]$. Therefore the oracles we assume are not really stronger than assuming that $K(x) + \int_s^x h(t)D_s(t)dt$ and $K(x)$ can be minimized on any interval. These in turn seem very reasonable and natural assumptions. We now briefly discuss a few specific cases where these oracles can be implemented efficiently.

(Piecewise) convex setup cost $K(t)$ and (piecewise) increasing demand rate $d(t)$. In this case, since $\int_s^x h(t)D_s(t)dt$ is (piecewise) convex in x, both $K(x)$ and $K(x)+\int_s^x h(t)D_s(t)dt$ are (piecewise) convex. Therefore both functions can be minimized on any interval in time linear in the number of pieces, and the oracles can be implemented efficiently.

Non-decreasing setup cost $K(t)$. Oracle 2 will always return ∞. Moreover $K(x)+\int_s^x h(t)D_s(t)dt$ is non-decreasing in x so that Oracle 2 can be implemented using Secant or Newton methods.

Discrete time and non-decreasing setup cost $K(t)$. Even if it is not the focus of our work, it is interesting to see what are the implications for the more classical discrete time setting. By discrete time, we mean here that setups can only be placed in a finite predefined set of time periods $\{s_1,\ldots,s_n\}$. Again non-decreasing setup cost implies that Oracle 2 is trivial. With n discrete time periods, Oracle 1 can be implemented by binary search in $O(\log n)$ time assuming that both the accumulated demand and the accumulated holding cost on $[1,t]$ can be queried in constant time for any period t. We need to be cautious here. If we assume that the input data of the discrete problem is accessed only by querying the demand and unit holding cost for any period t, then we cannot query the accumulated demand and cost in constant time. But the opposite is true: if we can directly query the accumulated holding cost and demand since the beginning of the horizon, we can compute the demand in any given period t in constant time, and the unit holding cost in any period as well.

2.3 Size of an instance

As discussed above, the input of an instance of continuous-time **ELS** are 3 continuous functions that we access by the oracles described below above. Therefore there is no meaningful notion of input size here. More interesting is to consider n, the size of the setup sequence $\{s_i\}_{i=1..n}$ output by an algorithm. One attractive definition of instance size would be n_{OPT}, the number of setups in an optimal solution. An algorithm would be efficient if the number of oracle calls is polynomial in n_{OPT} and $\frac{1}{\epsilon}$.

However, we want to briefly argue that this is too ambitious, unless one would assume unreasonably powerful oracles. Indeed, it is easy to construct a family of instances parametrized by an integer N with uniform unit setup cost and where the optimal solution is to place one single

setup at time 0, but with unbounded holding cost N. These instances have N small bursts of demand ever more spaced in time. It is hard to imagine how to solve these instances for growing N (and differentiate them from similar instances where the optimal solution places many setups) in a constant number of oracle calls.

One natural solution is to incorporate the holding costs in the instance size. When the setup function is constant and equal to 1, this suggests to use the value of the optimal solution **OPT** as the instance size. Note that the complexity of the natural 2-approximation of Massonnet [Massonnet(2013)] is linear in **OPT** (and not in n_{OPT}).

When the setup cost function $K(t)$ varies over time, the holding cost does not provide anymore a right measure for the instance size because it is expressed in different units than the number of setups. Let $\{s_i\}_{i=1..n}$ be an arbitrary solution for an **ELS** instance defined by functions K, h, d in $[0, T]$. Let us define the size of the solution as

$$\sigma(\{s_i\}_{i=1..n}) = \sum_{i=1}^{n} \left(1 + \frac{H_i}{\min_{s_i \le s \le s_{i+1}} K(s)}\right).$$

The function $\sigma(\{s_i\}_{i=1..n})$ counts the number of setups in the solution, while it weights the holding costs in a window with respect to the minimum setup cost there. Let us define the following instance-dependent parameter:

$$\sigma = \min_{n \in \mathbb{N}} \min_{\{s_i\}_{i=1..n}} \sigma\left(\{s_i\}_{i=1..n}\right),$$

that is clearly equal to **OPT** when $K(t) = 1$ and smaller in general. In fact, when the setup costs are large within large enough time intervals, σ can be (arbitrarily) smaller than **OPT**. This is the main input size measure that we will use throughout the paper. In Section 6 we will describe a slightly stronger input size measure and associated algorithms, but at the expense of slightly more powerful oracles.

2.4 Notational and end-of-horizon issues

All the algorithms we present in the paper build solutions $\{s_i\}_{i=1..n}$ in a serial fashion. However, they do it in such a way that the (unnatural) expression $K(s_{i+1}) + H_{[s_i, s_{i+1})}$ becomes more convenient to state the cost of a window $[s_i, s_{i+1})$. This causes two complications we prefer to address beforehand. First, the last window $[s_n, T)$ would need to be handled differently. To avoid this special case, we artificially enlarge the time horizon, by extending $K \equiv 0$ in $[T, \infty)$ (the discontinuity at T will not affect the analysis) and by attaching a fictitious setup $s_{n+1} \ge T$ that signals the end of the last window. We remark that this setup will never be indexed in $\{s_i\}_{i=1..n}$. We will also need to extend $h \equiv d \equiv 1$ in $[T, \infty)$. The need for this horizon extension artifact will be apparent once the algorithms are described. The second issue is that none of the cost terms $K(s_{i+1}) + H_{[s_i, s_{i+1})}$ accounts for the setup at time $s_1 = 0$. However, we can entirely remove the initial setup at time $s_1 = 0$ from the **ELS** model; the approximation factor of the algorithms under this modified model are also valid for the original one.

With the convention above, a solution $\{s_i\}_{i=1..n}$ (plus the implicit fictitious setup at time $s_{n+1} \ge T$) induces n windows $[s_i, s_{i+1}]$ with cost $K(s_{i+1}) + H_{[s_i, s_{i+1})}$.

3 A simple approximation scheme

Given input functions $h, d : [0, T] \to \mathbb{R}^+$ and $K : [0, T] \to [K_{\min}, K_{\max}]$ for the continuous-time **ELS** problem, let us define $T_c \in [0, T]$, for $c \in \mathbb{R}^+$, as the maximum time for which the truncated input

$K, h, d : [0, T_c] \to \mathbb{R}^+$ has a feasible solution with cost no larger than c. Clearly, the optimal cost **OPT** of the instance can be computed as $\mathbf{OPT} = \min\{c : T_c \geq T\}$; our aim is to approximately compute **OPT** by approximately computing the values T_c using dynamic programming.

Note that, since the total cost of a solution is additive with respect to the cost of their defining windows, we have the recursive formula

$$T_c = \max_{c' \in \mathbb{R}^+} S(T_{c-c'}, c'),$$

which, if applied to compute **OPT**, would need to be evaluated for $0 \leq c \leq \mathbf{OPT}$ only. In order to obtain an efficient dynamic programming algorithm from this recursion, we approximately solve it, by searching only through the set of feasible solutions where the cost of each defining window is a multiple of ϵ and an approximate power of $(1+\epsilon)$. The following lemma shows that this set contains a $(1+2\epsilon)$-approximation of the optimum.

Lemma 3.1. *For all instances of the continuous-time **ELS** problem and for every $\epsilon > 0$, there exists a solution with cost no larger than $(1+2\epsilon)\mathbf{OPT}$, where all its defining windows have costs that are powers of $(1+\epsilon)$ rounded up to the nearest multiple of ϵ.*

Proof. Let $\{\overline{s}_i\}_{i=1..n}$ be the setup times in an arbitrary, but fixed, optimal solution of an **ELS** instance. Let $\{\overline{c}_i\}_{i=1..n}$ be the costs of the corresponding windows defined by this solution.

Starting from this optimal solution, we build a (non-optimal) feasible solution $\{s_i\}_{i=1..n}$ as follows. For $i = 1, \ldots, n$, let $c_i = (1+\epsilon)^{k_i} + \epsilon_i$, where k_i is the smallest integer with $(1+\epsilon)^{k_i} \geq \overline{c}_i$, and $0 \leq \epsilon_i < \epsilon$ is the only real number that makes $(1+\epsilon)^{k_i} + \epsilon_i$ an integer multiple of ϵ. Beginning from $s_1 = 0$, define inductively $s_{i+1} = S(s_i, c_i)$ for $i = 1, \ldots, n$. Note that $\{c_i\}_{i=1..n}$, the costs of the corresponding windows induced by $\{s_i\}_{i=1..n}$, satisfy $\overline{c}_i \leq c_i \leq (1+\epsilon)\overline{c}_i + \epsilon\overline{c}_i$ for $i = 1, \ldots, n$.

By induction, we can easily show that $s_i \geq \overline{s}_i$ for all $i = 1, \ldots, n+1$. The induction statement is true for $i = 1$. If $s_i \geq \overline{s}_i$, then either $s_i \geq \overline{s}_{i+1}$, and the induction is complete, or $\overline{s}_i \leq s_i < \overline{s}_{i+1}$ and then $s_{i+1} = S(s_i, c_i) \geq S(s_i, \overline{c}_i) \geq S(\overline{s}_i, \overline{c}_i) = \overline{s}_{i+1}$, and the induction is complete. From here, we obtain that $s_{n+1} \geq \overline{s}_{n+1} = T$. Hence, the solution $\{s_i\}_{i=1..n}$ has cost at most

$$\sum_{i=1}^{n} c_i \leq \sum_{i=1}^{n} (1+2\epsilon)\overline{c}_i = (1+2\epsilon)\mathbf{OPT},$$

and satisfies the properties stated in the lemma. □

Let $c_{\epsilon,i} \equiv \lfloor (1+\epsilon)^i/\epsilon \rfloor \epsilon$, $i \in \mathbb{N}$. We are ready to describe the dynamic programming algorithm to approximately compute T_c, that we will call Algorithm **1**.

Let $\mathbf{T}_{j\epsilon} \in [0, T]$ be the maximum time for which there is a feasible solution to the truncated input $h, d : [0, \mathbf{T}_{j\epsilon}] \to \mathbb{R}^+$ of cost no larger than $j\epsilon$ and where each of its defining windows has cost $c_{\epsilon,i}$ for some $i \in \mathbb{N}$.

Similar to T_c, we can compute $\mathbf{T}_{j\epsilon}$ recursively:

$$\mathbf{T}_{j\epsilon} = \max_{i \in \mathbb{N}} S(\mathbf{T}_{j\epsilon - c_{i,\epsilon}}, c_{i,\epsilon})).$$

Lemma 3.1 implies that $\mathbf{T}_{j^*\epsilon} \geq T$ for $j^* = \lceil \frac{(1+2\epsilon)\mathbf{OPT}}{\epsilon} \rceil$, therefore, to obtain a $(1+2\epsilon)$ approximation we only need to evaluate the recursion for $j = 0, 1, \ldots, j^*$; in turn, each of these terms requires to compute the maximum over $\lceil \log_{1+\epsilon} \mathbf{OPT} \rceil + 1$ terms, hence the total time to compute the recursion is $O\left(\frac{\mathbf{OPT}}{\epsilon^2} \log \mathbf{OPT}\right)$. Putting all together, we obtain:

Theorem 3.1. *Algorithm **1** runs in $O\left(\frac{\boldsymbol{OPT}}{\epsilon^2}\log \boldsymbol{OPT}\right)$ and performs this same number of oracle calls.*

A major disadvantage of this algorithm is that it ignores the setup costs: the dynamic program advances by ϵ units of total cost even when the setup costs are very large. We address this issue in the following sections.

However, Algorithm 1 already improves the state-of-the-art in two particular cases. Firstly, for constant setup cost $K(t) = 1$, it improves on the 2-approximation of Massonet [Massonnet(2013)] by providing an approximation scheme at an additional cost of $O\left(\frac{\log \mathbf{OPT}}{\epsilon^2}\right)$ only. Secondly, in the discrete setting with increasing setup cost, Algorithm 1 runs in sub-linear time (e.g. when the optimal solution value and the number of setups are much smaller than the number of time periods). This shows that by allowing some approximation error ϵ, one can solve this specific discrete lot-sizing problem without querying all input data of the problem.

4 A constant approximation algorithm

We now describe a constant approximation algorithm whose number of oracle calls grows at most linearly in $\log(K_{\max})$. Because any approximate solution cannot place setups at times s where $K(s) = \Omega(\mathbf{OPT})$, it is possible to replace $\log(K_{\max})$ by $O(\log \mathbf{OPT})$.

The algorithm places setups iteratively: assuming that $s_1, s_2, \ldots, s_i$ have been placed, the algorithm places s_{i+1} at the latest time where $K(s_{i+1}) + H_{[s_i,s_{i+1})} = 2K(s_i)$ or the earliest time where $K(s_{i+1}) = K(s_i)/2$, whichever happens first. The algorithm stops once s_{i+1} exceeds T. We call this Algorithm **2**.

In the more compact notation that we will use from now on, Algorithm **2** determines windows $[s_i, s_{i+1})$ where the setup cost is at least $K_i/2$ and where either $K_{i+1} + H_i = 2K_i$ or $K_{i+1} = K_i/2$. Although the algorithm is simple, it is not immediate why it gives a good approximation. This is shown in the following result.

Theorem 4.1. *Algorithm **2** is a 8-approximation algorithm.*

Proof. Let $\{s_i\}_{i=1..n}$ and $\{H_i\}_{i=1..n}$ be the setup times and the holding costs of the corresponding windows of the solution returned by the algorithm; also, let $\{\overline{s}_i\}_{i=1..m}$ be the setup times in a fixed, but arbitrary optimal solution.

If the window $[s_i, s_{i+1})$ contains a setup from **OPT**, we say that s_{i+1} is an *extreme (ext.)* setup. We partition non-extreme setups s_{i+1} as follows: we call them *increasing (inc.)* if $K_{i+1} + H_i = 2K_i$ and $K_{i+1} \geq \frac{3K_i}{2}$; we call them *constant (cons.)* if $K_{i+1} + H_i = 2K_i$ and $H_i > K_i/2$; otherwise, they must satisfy $K_{i+1} = K_i/2$ and we call them *decreasing (dec.)*.

Note that if $[s_i, s_{i+1})$ contains a setup $\overline{s}_j$, then $K(\overline{s}_j) \geq K_i/2$. Therefore,

$$\sum_{s_{i+1}\text{ ext.}} (K_{i+1} + H_i) \leq \sum_{s_{i+1}\text{ ext.}} 2K_i \leq \sum_{s_{i+1}\text{ ext.}} \sum_{\overline{s}_j \in [s_i, s_{i+1})} 4K(\overline{s}_j) \leq 4\mathbf{OPT}$$

It is easy to see also that

$$\sum_{s_{i+1}\text{ cons.}} (K_{i+1} + H_i) = \sum_{s_{i+1}\text{ cons.}} 2K_i \leq \sum_{s_{i+1}\text{ cons.}} 4H_i \leq 4\mathbf{OPT},$$

where we used the fact that the window $[s_i, s_{i+1})$ has no setup from the optimum.

In the remaining of the proof, we provide a similar type of bound for increasing and decreasing setups. Let $\{s_i\}_{i=k..l}$ be a maximal sequence of increasing setups. Note that

$$\sum_{i+1=k}^{i+1=l} (K_{i+1} + H_i) = 2 \sum_{i+1=k}^{i+1=l} K_i \leq 2 \sum_{i=0}^{l-k+1} \left(\frac{2}{3}\right)^i K_{l-1} \leq 6 \cdot \frac{2}{3} K_l = 4K_l$$

We now give an upper bound on K_l as a function of **OPT**. This upper bound will depend on whether the setup s_{l+1} is extreme, constant or decreasing:

- If s_{l+1} is a constant setup, then $H_l > K_l/2$. The optimum does not place setups in $[s_l, s_{l+1}]$, so it "fully" pays H_l.
- If s_{l+1} is a decreasing setup, then $K_{l+1} = K_l/2$; since s_l is increasing, then $K_l + H_{l-1} = 2K_{l-1}$. Altogether, this gives $K_{l+1} = K_l/2 \leq K_{l-1} \leq K_l$. On the other hand, we have that $K(\cdot) \geq K_{l+1}$ in $[s_l, s_{l+1}]$; by continuity, there is $s \in [s_l, s_{l+1}]$ where $K(s) = K_{l-1}$. Note that $K(t) + H_{[s_{l-1},t)}$ is a continuous function of t, with a non-negative first term and a increasing unbounded (because of the horizon extension) second term. Therefore, if $K(s) + H_{[s_{l-1},s)} < 2K_{l-1}$ then s_l could not have been an increasing setup. Hence, $K(s) + H_{[s_{l-1},s)} \geq 2K_{l-1}$ and so $H_{[s_{l-1},s_{l+1})} \geq H_{[s_{l-1},s)} \geq K_{l-1} \geq K_l/2$. Again, the optimum places no setups in $[s_{l-1}, s_{l+1})$, so it "fully" pays $H_{[s_{l-1},s_{l+1})}$.
- If s_{l+1} is an extreme setup, then there is an optimal setup $\overline{s}_j$ in $[s_l, s_{l+1})$. It follows that $K(\overline{s}_j) \geq K_l/2$.

Adding all, we obtain

$$\begin{aligned}
\sum_{s_i \text{ inc.}} (K_{i+1} + H_i) &\leq \sum_{s_l \text{ inc.}, s_{l+1}\text{cons.}} 4K_l + \sum_{s_l \text{ inc.}, s_{l+1}\text{dec.}} 4K_l + \sum_{s_l \text{ inc.}, s_{l+1}\text{ext.}} 4K_l \\
&\leq \sum_{s_l \text{ inc.}, s_{l+1}\text{cons.}} 8H_l + \sum_{s_l \text{ inc.}, s_{l+1}\text{dec.}} 8H_{[s_{l-1},s_{l+1})} + \sum_{s_l \text{ inc.}, s_{l+1}\text{ext.}, \overline{s}_j \in [s_l, s_{l+1}]} 8K(\overline{s}_j) \\
&\leq 8\mathbf{OPT}.
\end{aligned}$$

With a similar argument, we can bound the number of decreasing setups. The main difference is the change from 2/3 to 1/2 in the geometric sum. This gives:

$$\sum_{s_i \text{ dec.}} (K_{i+1} + H_i) \leq 4\mathbf{OPT}$$

Finally, adding the bounds for each type of setup, we obtain $\mathbf{ALG} \leq 24\mathbf{OPT}$. It is not hard to see that if we aggregate all the different types of setups before establishing the comparison with **OPT**, this analysis gives $\mathbf{ALG} \leq 8\mathbf{OPT}$. □

By further elaborating on the proof of Theorem 4.1, it is easy to show that Algorithm **2** performs no more than $O(\mathbf{OPT} \log K_{\max})$ oracle calls. This bound is not better than the bound in number of oracle calls performed by Algorithm **1** when we aim for the same approximation guarantee. However, Algorithm **2** not only seems to advance faster on instances with larger setup costs; it is indeed faster than Algorithm **1** in these cases. This is formalized next.

Theorem 4.2. *Algorithm 2 places no more than $O(\sigma \log K_{\max})$ setups.*

Proof. Let $\{s_i\}_{i=1..n}$ and $\{H_i\}_{i=1..n}$ be the setup times and the holding costs of the corresponding windows in the solution returned by the algorithm; also, let $\{\overline{s}_i\}_{i=1..m}$ and $\{\overline{H}_i\}_{i=1..m}$ be the setup times and the holding costs of the corresponding windows in a fixed, but arbitrary solution **SOL**.

We partition the setups performed by the algorithm exactly as in the proof of Theorem 4.1, except that **OPT** is replaced by **SOL**: if a window $[s_i, s_{i+1})$ contains a setup from **SOL**, we say that s_{i+1} is an *extreme* setup. We partition non-extreme setups s_{i+1} in *increasing* setups if $K_{i+1} + H_i = 2K_i$ and $K_{i+1} \geq \frac{3K_i}{2}$; *constant* setups if $K_{i+1} + H_i = 2K_i$ and $H_i > K_i/2$; and *decreasing* setups if they are none of the above.

We now bound the number of setups of each type, as a function of σ. These bounds use similar ideas to those found in the proof of Theorem 4.1, so here we only sketch them:

- The number of extreme setups placed by the algorithm is not larger than the number of setups in **SOL**, and therefore no larger than σ.
- Constant setups satisfy $H_i \geq K_i/2$. In particular, the constant setups contained in a window $[\overline{s}_j, \overline{s}_{j+1})$ of **SOL** satisfy $H_i \geq \min_{\overline{s}_j \leq s \leq \overline{s}_{j+1}} K(s)/2$. It follows that no more than

$$\frac{2\overline{H}_j}{\min_{\overline{s}_j \leq s \leq \overline{s}_{j+1}} K(s)}$$

  constant setups can be contained in the window $[\overline{s}_j, \overline{s}_{j+1})$. Therefore, no more than 2σ of these setups can exist in $\{s_i\}_{i=1..n}$.
- To bound the number of increasing setups, let $\{s_i\}_{i=k..l}$ be a maximal sequence of increasing setups contained in a window $[\overline{s}_j, \overline{s}_{j+1})$ from **SOL**. The bound depends on what type of setup s_{l+1} is:
  - If s_{l+1} is an extreme (resp. constant) setup, then no more than σ (resp. 2σ) of these maximal sequences can exist. This follows from the bounds we already stated for the number of extreme (resp. constant) setups.
  - If s_{l+1} is a decreasing setup, then let us recall that, from the proof of Theorem 4.1, $H_{[s_{l-1}, s_{l+1})} \geq H_{[s_{l-1}, s)} \geq K_{l-1}$. Also, **SOL** places no setups in $[s_{l-1}, s_{l+1})$, so the algorithm fully pays $H_{[s_{l-1}, s_{l+1})}$. With an argument similar to the one used in the constant setup case, there cannot be more than σ of these maximal sequences.

  Since no more than $\log_{3/2} K_{\max}$ increasing setups can be consecutive, it follows that there are at most $4\sigma \log_{3/2} K_{\max}$ increasing setups.
- The number of decreasing setups can also be bounded by $4\sigma \log_{3/2} K_{\max}$, with the same argument.

Overall, this gives the desired bound $O(\sigma \log K_{\max})$ for the total number of setups. □

5 Another approximation scheme.

We now describe an approximation scheme for the continuous-time **ELS** problem with time-varying setup costs. It builds on the ideas developed for the algorithms we have seen so far; in particular, the solution **SOL** $= \{s_j\}_{j=1..m}$, found by Algorithm **2** plays a crucial role.

The approximation scheme, that we will call Algorithm **3**, builds an increasing sequence of cost values $\{c_i\}_{i=1..n}$ for which it computes the *grid points*

$$T_{c_i} \equiv \max_{l=1..i-1} S(T_{c_l}, c_i - c_l)$$

via dynamic programming (with $T_{c_1} = 0$). Implicitly, a grid point T_c corresponds to the maximum time for which the truncated input $h, d, K : [0, T_c] \to \mathbb{R}^+$ has a feasible solution of a certain type with cost no larger than c. While all of the above closely mimics the description of Algorithm **1**, the computation of $\{c_i\}_{i=1..n}$ is much more involved. Roughly speaking, when T_{c_i} is in a window $[s_j, s_{j+1})$ of **SOL**, the algorithm will try and make $c_{i+1} = c_i + \epsilon K(s_j)/2$, except when this puts $T_{c_{i+1}}$ out of the window $[s_j, s_{j+1})$; there, the step of $\epsilon K(s_j)/2$ has to be reduced carefully so that the sequence does not exit the window too quickly.

Starting from $c_1 = 0$, we build the sequence of cost values iteratively. Suppose that $c_1, c_2, \ldots, c_i$ have been computed and that $T_{c_i} \in [s_j, s_{j+1})$ for some j. Let k_i be the largest integer such that 2^{k_i+1} is no larger than the cost of each setup placed by **SOL** in $[s_j, T_{c_i+\epsilon 2^{k_i}}]$; equivalently, k_i is maximized subject to $2^{k_i} \le K(s_{j'})/2$ for every $s_{j'} \in [s_j, T_{c_i+\epsilon 2^{k_i}}]$). Note that $-1 \le k_i \le \log(K(s_j)/2)$. We then set $c_{i+1} = c_i + \epsilon 2^{k_i}$ as the next cost in the sequence. Algorithmically, determining k_i requires no more than $\log\log K_{\max}$ calls to the oracle, using binary search. For convenience, through all this section we denote by $\bar{k}_j \equiv \log(K(s_j))/2$ the maximum exponent k_i we can achieve inside the window $[s_j, s_{j+1})$.

We first show that Algorithm **3** is indeed an approximation scheme.

Theorem 5.1. *Algorithm 3 determines a solution with cost at most* $(1+\epsilon)\boldsymbol{OPT}$.

Proof. Let $\mathbf{OPT} = \{\overline{s}_l\}_{l=1..m}$ be an arbitrary optimal solution, and $\mathbf{SOL} = \{s_j\}_{j=1..m'}$ be the solution found by Algorithm **2**. Also, let $\{\bar{c}_l\}_{l=0..m}$ be the sequence defined by $\bar{c}_0 = 0$, and $\bar{c}_{l+1} = \bar{c}_l + K(\overline{s}_{l+1}) + H_{[\overline{s}_l, \overline{s}_{l+1})}$, for $1 \le l \le m$.

We now show, by induction, that for every setup $\overline{s}_l$ from the optimum, there exist a grid point $T_{c_{i(l)}} \ge s_l$ such that $c_{i(l)}$ is at most $(1+\epsilon)\bar{c}_l$. Suppose the induction statement is true for a certain l. If $\overline{s}_{l+1} \le T_{c_{i(l)}}$, then $i(l+1) = i(l)$ completes the induction. Otherwise, let $T_{c_{i'}}$ be the grid point satisfying $T_{c_i(l)} \le T_{c_{i'}} < \overline{s}_{l+1} \le T_{c_{i'+1}}$ and let $s_{j'}$ be the latest setup in **SOL** satisfying $s_{j'} \le T_{c_{i'}}$. Algorithm **2** chooses $k_{i'}$ so that $2^{k_{i'}} \le K(s_j)/2$ for all $s_j \in [s_{j'}, T_{c_{i'+1}}]$; the properties of **SOL** guarantee that $K(\overline{s}_{l+1}) \ge \min_{s_j \in [s_{j'}, T_{c_{i'+1}}]} K(s_j)/2 \ge 2^{k_{i'}}$. Hence, $c_{i'+1} - c_{i'} \le \epsilon K(\overline{s}_{l+1})$. We also have $c_{i'} - c_{i(l)} < K(\overline{s}_{l+1}) + H_{[\overline{s}_l, \overline{s}_{l+1})}$. Adding these two inequalities, we obtain $c_{i'+1} \le c_{i(l)} + (1+\epsilon)(K(\overline{s}_{l+1}) + H_{[\overline{s}_l, \overline{s}_{l+1})}) = (1+\epsilon)\bar{c}_{l+1}$, and therefore $i(l+1) = i'$ completes the induction. □

The following lemma gives a bound on the number of grid points in any window of **SOL**.

Lemma 5.1. *There are no more than* $O(1/\epsilon) + \log K_{\max}$ *grid points between any two consecutive setups of* ***SOL***.

Proof. Let T_{c_i} be a grid point contained in a window $[s_j, s_{j+1}]$ defined by **SOL**. Then, we have the property that $T_{c_{i'}} > s_{j+1}$ whenever $c_{i'} - c_i > K(s_{j+1}) + H_{[T_{c_i}, s_{j+1})}$; this follows from the fact that the dynamic program considers the possibility of adding an extra setup with cost $c_{i'} - c_i$, after placing a setup at T_{c_i}. The property implies, in particular, that there are no more than

$$O\left(\frac{K(s_{j+1}) + H_{[s_j, s_{j+1})}}{\epsilon 2^{\bar{k}_j}}\right) = O\left(\frac{K(s_j)}{\epsilon 2^{\bar{k}_j}}\right) = O\left(\frac{1}{\epsilon}\right)$$

grid points T_{c_i} in $[s_j, s_{j+1}]$ for which $k_i = \bar{k}_j$.

Now, suppose that T_{c_i} and $T_{c_{i+1}}$ are two consecutive grid points in the window $[s_j, s_{j+1}]$. If $k_i < \bar{k}_j$, then $T_{c_{i+1}+\epsilon 2^{k_i}} = T_{c_i+\epsilon 2^{k_i+1}} > s_{j+1}$. Note that this necessarily implies that either $k_{i+1} < k_i$ or $T_{c_{i+2}}$ leaves the window. In particular, the sequence of cost differences is always decreasing inside a window $[s_j, s_{j+1}]$; and it can be constant only when it equals $\epsilon 2^{\bar{k}_j}$. It follows that there are no more than $\log K_{\max}$ grid points T_{c_i} in $[s_j, s_{j+1}]$ for which $k_i < \bar{k}_j$.

Adding the two bounds for the number of grid points with $k_i = \bar{k}_j$ and $k_i < \bar{k}_j$, we obtain the desired result.

□

The result above can be strengthened for certain setups. We will need the notions of increasing/decreasing setups from Algorithm **2**.

Lemma 5.2. *Let $\{s_{j+1}\}_{j=l..l'}$ be any sequence of decreasing (resp. increasing) setups. Then, Algorithm **3** places no more than $O((l'-l)/\epsilon) + \log K_{\max}$ (resp. $O((l'-l)/\epsilon)$) grid points on $[s_l, s_{l'+1}]$.*

Proof. This proof builds on ideas already used in Theorem 5.1; here we only sketch them. Consider an arbitrary sequence $\{s_{j+1}\}_{j=l..l'}$ of increasing setups. Let T_{c_i} be any grid point in one of the windows $[s_{j'}, s_{j'+1}]$, $l \le j' \le l'$. Because $\{K(s_j)\}_{j=l..l'+1}$ is an increasing sequence, we have the equivalence

$$2^k \le K(s_j)/2 \text{ for every } s_j \in [s_{j'}, s_{l'+1}] \iff 2^k \le K(s_{j'})/2,$$

and therefore $k_i = \bar{k}_j$ if $T_{c_{i+1}} \le s_{l'+1}$. In the proof of Lemma 5.1, we showed that no more than $O(1/\epsilon)$ grid points $T_{c_i} \in [s_{j'}, s_{j'+1}]$, satisfy $k_i = \bar{k}_j$; the bound on the number of grid points in an increasing setup sequence follow directly from this.

On the other hand, if $\{s_{j+1}\}_{j=l..l'}$ is an arbitrary sequence of decreasing setups, then $\{K(s_j)\}_{j=l..l'+1}$ is a decreasing sequence. Therefore, for any $s_l \le s_{j'} < s_{j''} \le s_{l'+1}$

$$2^k \le K(s_j)/2 \text{ for every } s_j \in [s_{j'}, s_{j''}] \iff 2^k \le K(s_{j''})/2,$$

which implies that sequence of cost differences between grid points is decreasing in the whole interval $[s_{j'}, s_{l'+1}]$. Using similar ideas to those found in the proof of Theorem 5.1, we conclude that there are no more than $O((l'-l)/\epsilon) + \log K_{\max}$ grid points in $[s_l, s_{l'+1}]$. □

Aggregating the results of the previous two lemmas, we obtain our desired bound on the execution time and number of oracle calls performed by the algorithm:

Theorem 5.2. *Algorithm **3** places no more than $O(\sigma \log K_{\max}/\epsilon)$ grid points and executes no more than $O(\sigma^2 \log^2 K_{\max}/\epsilon^2)$ oracle calls.*

Proof. In the proof of Theorem 4.2 we showed that i) the number of setups in **SOL** that are neither increasing nor decreasing is $O(\sigma)$; and ii) the number of maximal sequences of either increasing setups or decreasing setups, is also $O(\sigma)$. Using Theorems 5.1 and 5.2, we obtain the desired bound on the total number of grid points.

The bound on the number of oracle calls follows from the fact that this amount is dominated by the number of oracle calls realized during the dynamic program, which is quadratic in the number of grid points. □

6 An improved input-size measure and algorithms

At expense of slightly stronger oracles than the ones we have used so far, we can obtain a better approximation algorithm. The algorithm (call it Algorithm **2A**) is similar to Algorithm **2**, but it uses

$$\widetilde{H}_{[s,s')} \equiv \int_s^{s'} h(t) \int_t^{s'} \frac{d(u)}{\min_{t\le v\le u} K(v)} du\ dt$$

as a "normalized" holding cost measure. It places setup s_{i+1} at the latest time where $K(s_{i+1}) + K(s_i)\widetilde{H}_{[s_i,s_{i+1})} = \frac{3}{2}K(s_i)$ or the earliest time where $K(s_{i+1}) = K(s_i)/2$, whichever happens first. Note that $\widetilde{H}_{[s_i,s_{i+1})} \le 1$.

Let $\{s_i\}_{i=1..n}$ be an arbitrary solution for an **ELS** instance defined by functions K, h, d in $[0,T]$. Let us define the size of the solution as $\bar{\sigma}(\{s_i\}_{i=1..n}) = \sum_{i=1}^{n}\left(1 + \widetilde{H}_{[s_i,s_{i+1})}\right)$. Let $\bar{\sigma}$ be the minimum of $\bar{\sigma}(\cdot)$, over all possible solutions for this instance. Note that $\widetilde{H}_i \le \frac{2H_i}{K_i}$, for all i, and therefore $\bar{\sigma} \le \sigma$.

We can prove a result analogous to Theorem 5.1, with $\bar{\sigma}$ playing the role of σ. We need however, a different lower bound on the optimal solution.

Lemma 6.1. *Let $\{s_i\}_{i=1..n}$ be the output of Algorithm **2A**. Then, $\sum_{i=1}^{n} H_{[s_i,s_{i+1})}$ is a lower bound on the cost of the optimal solution.*

Proof. Let $\mathbf{OPT} = \{\bar{s}_j\}_{j=1..m}$ be an arbitrary optimal solution. For $t \ge 0$, let $\text{next}^s(t)$ (resp. $\text{next}^{\bar{s}}(t)$ be the earliest setup in **ALG** (resp. **OPT**) that is placed after t. Clearly,

$$H_{[s_i,s_{i+1})} = \int_{s_i}^{\bar{s}_{i+1}} h(t) \int_t^{\text{next}^{\bar{s}}(t)} d(u)du\ dt + \int_{s_i}^{\bar{s}_{i+1}} h(t) \int_{\text{next}^{\bar{s}}(t)}^{\text{next}^s(t)} d(u)du\ dt.$$

Let $h(\mathbf{OPT})$ be the holding cost of the solution **OPT**. We have:

$$\begin{aligned}
\sum_{i=1}^{n} H_{[s_i,s_{i+1})} &= \int_0^T h(t) \int_t^{\text{next}^{\bar{s}}(t)} d(u)du\ dt + \int_0^T h(t) \int_{\text{next}^{\bar{s}}(t)}^{\text{next}^s(t)} d(u)du\ dt. \\
&\le h(\mathbf{OPT}) + \int_0^T h(t) \cdot 1_{\{\text{next}^{\bar{s}}(t) \le \text{next}^s(t)\}} \int_{\text{next}^{\bar{s}}(t)}^{\text{next}^s(t)} d(u)du\ dt. \\
&= h(\mathbf{OPT}) + \sum_{i=1}^{n} \sum_{\bar{s}_j \in (s_i,s_{i+1})} \int_{\max\{s_i,\bar{s}_{j-1}\}}^{\bar{s}_j} h(t) \int_{\bar{s}_j}^{s_{i+1}} d(u)du\ dt \\
&\le h(\mathbf{OPT}) + \sum_{i=1}^{n} \sum_{\bar{s}_j \in (s_i,s_{i+1})} \int_{s_i}^{\bar{s}_j} h(t) \int_{\bar{s}_j}^{s_{i+1}} d(u)du\ dt,
\end{aligned}$$

obtaining the lower bound

$$\mathbf{OPT} \ge \sum_{i=1}^{n} \left(H_{[s_i,s_{i+1})} + \sum_{\bar{s}_j \in (s_i,s_{i+1})} \left(K(\bar{s}_j) - \int_{s_i}^{\bar{s}_j} h(t) \int_{\bar{s}_j}^{s_{i+1}} d(u)du\ dt) \right) \right).$$

To conclude, we just need to argue that the inner sum in the expression above is non-negative. This follows from the fact that, if $\bar{s}_j \in [s_i, s_{i+1})$ is an arbitrary setup from **OPT**, then

$$\frac{1}{K(\overline{s}_j)}\int_{s_i}^{\overline{s}_j} h(t)\int_{\overline{s}_j}^{s_{i+1}} d(u)du\ dt \le \int_{s_i}^{\overline{s}_j} h(t)\int_{\overline{s}_j}^{s_{i+1}} \frac{d(u)}{\min_{\overline{s}_j\le v\le u} K(v)}du\ dt \le \widetilde{H}_{[s_i,s_{i+1})} \le 1.$$

□

Theorem 6.1. *Algorithm **2A** is an $O(1)$-approximation algorithm that places no more than $O(\widetilde{\sigma}\log K_{\max})$ setups.*

Proof. We only sketch the proof, as we use similar ideas than those found in the proof of Theorem 4.1. We utilize the simplified notation that appears in that theorem, extending it to $\widetilde{H}$ as well. If $\{s_i\}_{i=1..n}$ is the output of Algorithm **2A**, let us call a setup s_{i+1} constant if $K_{i+1} \le 8H_i$. Lemma 6.1 implies that the total cost of constant setups is at most 9**OPT**. We classify a non-constant setup s_{i+1} as increasing or decreasing according to whether $K_{i+1} + K_i\widetilde{H}_i = \frac{3}{2}K_i$ or $K_{i+1} = K_i/2$ holds.

We now bound the total cost of the increasing setups. Let $\{s_i\}_{i=k..l}$ be a maximal sequence of increasing setups. We have that $\frac{3}{2}K_i = K_{i+1} + K_i\widetilde{H}_i \le K_{i+1} + 2H_i \le \frac{5}{4}K_{i+1}$ that is, $K_{i+1} \ge \frac{6}{5}K_i$. As in Theorem 4.1, a geometric sum argument gives that the total cost of an increasing sequence of setups is at most $6K_l + \sum_{i=k}^{l} H_i$. By Lemma 6.1, the sum over all maximal increasing sequences, of the term $\sum_{i=k}^{l} H_i$, is at most **OPT**. We will now show that the sum, over all maximal increasing sequences, of the term $6K_l$, is at most 32**OPT**. We split the analysis into two cases:

- If s_{l+1} is a constant setup, then $\frac{K_l}{2} \le K_{l+1} \le 8H_l$.
- If s_{l+1} is a decreasing setup, then $K_{l+1} = K_l/2$; since s_l is an increasing setup, $K_l + K_{l-1}\widetilde{H}_{l-1} = \frac{3}{2}K_{l-1}$. Altogether, this gives $K_{l+1} = K_l/2 \le \frac{3K_{l-1}}{4}$. Because s_{l-1} is an increasing setup, $K_{l+1} + K_{l-1}\widetilde{H}_{[s_{l-1},s_{l+1})} \ge \frac{3}{2}K_{l-1}$. It follows that

$$\widetilde{H}_{[s_{l-1},s_{l+1})} \ge \frac{\frac{3}{2}K_{l-1} - \frac{3}{4}K_{l-1}}{K_{l-1}} \ge \frac{3}{4}.$$

  Finally, note that $H_{[s_{l-1},s_{l+1})} \ge \frac{K_{l+1}\widetilde{H}_{[s_{l-1},s_{l+1})}}{2} \ge \frac{3}{8}K_{l+1} = \frac{3}{16}K_l$. Either **OPT** places a setup in $[s_{l-1}, s_{l+1})$, paying at least $K_{l+1}/2$ in setup costs, or it does not, paying at least $\frac{3}{16}K_l$. In both cases, the optimal solution pays at least $\frac{3}{16}K_l$.

Aggregating, the total cost, of all possible maximal setups is at most 33**OPT**.

In order to prove that Algorithm **2A** places no more than $O(\bar{\sigma}\log K_{\max})$ setups, consider a fixed solution $\{\overline{s}_i\}_{i=1..n}$. We will prove that on each window $[\overline{s}_i, \overline{s}_{i+1})$, Algorithm **2A** places no more than $1 + \widetilde{H}_{[\overline{s}_i,\overline{s}_{i+1})}$ setups. We need to slightly redefine the notions of constant, increasing and decreasing setups. We call a setup s_{i+1} constant if $\widetilde{H}_i \ge \frac{3}{4}$. Increasing and decreasing setups are the non-constant setups s_{i+1} satisfying $K_{i+1} + K_i\widetilde{H}_i = \frac{3}{2}K_i$ and $K_{i+1} = K_i/2$, respectively. Clearly, the number of constant setups in $[\overline{s}_i, \overline{s}_{i+1})$ is at most $\widetilde{H}_{[\overline{s}_i,\overline{s}_{i+1})}$. Consider now a maximal sequence $\{s_i\}_{i=k..l}$ of increasing setups. We have two cases:

- If s_{l+1} is a constant setup, then we immediately have $\widetilde{H}_i \ge \frac{3}{4}$. It follows that there are no more than $\frac{4}{3}\widetilde{H}_{[\overline{s}_i,\overline{s}_{i+1})}$ maximal increasing sequences in the window.
- If s_{l+1} is a decreasing setup, we can show as before that $\widetilde{H}_{[s_{l-1},s_{l+1})} \ge \frac{\frac{3}{2}K_{l-1} - \frac{3}{4}K_{l-1}}{K_{l-1}} \ge \frac{3}{4}$. Again, no more than $\frac{4}{3}\widetilde{H}_{[\overline{s}_i,\overline{s}_{i+1})}$ of such setups can exist.

Since each of these increasing maximal sequences can have at the most, $O(\log K_{\max})$ setups, this gives an overall total of $\frac{4}{3}\bar{\sigma}\log K_{\max}$ of these setups.

□

7 Open Questions

We have presented the first efficient approximation schemes for Economic Lot-Sizing in Continuous Time, using reasonable oracles. There are several interesting questions and problems that we have left open. It would firstly be interesting to see if our algorithms can be adapted to rely on approximate oracles instead of exact ones. This makes sense as the assumed oracle are themselves continuous optimization problems. Secondly, there is a lack of lower bound on the complexity under our oracle models. Last, it would be interesting to study which (stronger) oracles are necessary to get an approximation scheme that runs in something polynomial in n_{OPT} (the number of setups in an optimal solution) instead of σ (our instance size measure).

References

[Boyd and Vandenberghe(2004)] Boyd, Stephen Poythress, Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

[Federgruen and Tzur(1991)] Federgruen, A., M. Tzur. 1991. A simple forward algorithm to solve general dynamic lot sizing models with n periods in $O(n \log n)$ or $O(n)$ time. *Management Science* **37**(8) 909–925. doi:10.1287/mnsc.37.8.909. URL `http://mansci.journal.informs.org/content/37/8/909.abstract`.

[Harris(1990)] Harris, Ford W. 1990. How many parts to make at once. *Operations Research* 947–950.

[Lovász and Vempala(2006)] Lovász, László, Santosh Vempala. 2006. Simulated annealing in convex bodies and an $O^*(n^4)$ volume algorithm. *Journal of Computer and System Sciences* **72**(2) 392–417.

[Massonnet(2013)] Massonnet, G. 2013. Algorithmes d'approximation pour la gestion de stocks. Ph.D. thesis, University of Grenoble, France.

[Papadimitriou(1994)] Papadimitriou, Christos H. 1994. *Computational complexity.*. Addison-Wesley.

[Plaskota(1996)] Plaskota, Leszek. 1996. *Noisy information and computational complexity*. Cambridge University Press.

[Schrijver(2003)] Schrijver, Alexander. 2003. *Combinatorial optimization: polyhedra and efficiency*, vol. 24. Springer Verlag.

[Sikorski(1985)] Sikorski, K. 1985. Optimal solution of nonlinear equations. *Journal of Complexity* **1**(2) 197–209.

[Traub and Werschulz(1998)] Traub, J Joseph Frederick, Arthur G Werschulz. 1998. *Complexity and information*. Cambridge University Press.

[Wagelmans et al.(1992)Wagelmans, van Hoesel, and Kolen] Wagelmans, A., S. van Hoesel, A. Kolen. 1992. Economic lot sizing: An $O(n \log n)$ algorithm that runs in linear time in the wagner-whitin case. *Operations Research* **40**(11) 145–156. doi:10.1287/opre.40.1.S145. URL `http://or.journal.informs.org/content/40/1-Supplement-1/S145.abstract`.

[Wagner and Whitin(1958)] Wagner, H. M., T. M. Whitin. 1958. Dynamic version of the economic lot size model. *Management Science* **5**(1) 89–96.

[Wilson(1934)] Wilson, RH. 1934. A scientific routine for stock control. *Harvard business review* **13**(1) 116–129.

Recent titles

CORE Discussion Papers

2013/51 Anthony PAPAVASILIOU, Yi HE and Alva SVOBODA. Self-commitment of combined cycle units under electricity price uncertainty.

2013/52 Ana MAULEON, Elena MOLIS, Vincent VANNETELBOSCH and Wouter VERGOTE. Dominance invariant one-to-one matching problems.

2013/53 Jean GABSZEWICZ and Skerdilajda ZANAJ. (Un)stable vertical collusive agreements.

2013/54 François MANIQUET and Massimo MORELLI. Approval quorums dominate participation quorums.

2013/55 Mélanie LEFÈVRE and Joe THARAKAN. Intermediaries, transport costs and interlinked transactions.

2013/56 Gautier M. KRINGS, Jean-François CARPANTIER and Jean-Charles DELVENNE. Trade integration and the trade imbalances in the European Union: a network perspective.

2013/57 Philip USHCHEV, Igor SLOEV and Jacques-François THISSE. Do we go shopping downtown or in the 'burbs'? Why not both?

2013/58 Mathieu PARENTI. Large and small firms in a global market: David vs. Goliath.

2013/59 Paul BELLEFLAMME and Francis BLOCH. Dynamic protection of innovations through patents and trade secrets.

2013/60 Christian HAEDO and Michel MOUCHART. Specialized agglomerations with areal data: model and detection.

2013/61 Julien MARTIN and Florian MAYNERIS. High-end variety exporters defying distance: micro facts and macroeconomic implications.

2013/62 Luca G. DEIDDA and Dimitri PAOLINI. Wage premia, education race, and supply of educated workers.

2013/63 Laurence A. WOLSEY and Hande YAMAN. Continuous knapsack sets with divisible capacities.

2013/64 Francesco DI COMITE, Jacques-François THISSE and Hylke VANDENBUSSCHE. Verti-zontal differentiation in export markets.

2013/65 Carl GAIGNÉ, Stéphane RIOU and Jacques-François THISSE. How to make the metropolitan area work? Neither big government, nor laissez-faire.

2013/66 Yu. NESTEROV and Vladimir SHIKHMAN. Algorithmic models of market equilibrium.

2013/67 Cristina PARDO-GARCIA and Jose J. SEMPERE-MONERRIS. Equilibrium mergers in a composite good industry with efficiencies.

2013/68 Federica RUSSO, Michel MOUCHART and Guillaume WUNSCH. Confounding and control in a multivariate system. An issue in causal attribution.

2013/69 Marco DI SUMMA. The convex hull of the all-different system with the inclusion property: a simple proof.

2013/70 Philippe DE DONDER and Pierre PESTIEAU. Lobbying, family concerns and the lack of political support for estate taxation.

2013/71 Alexander OSHARIN, Jacques-François THISSE, Philip USHCHEV and Valery VERBUS. Monopolistic competition and income dispersion.

2013/72 N. Baris VARDAR. Imperfect resource substitution and optimal transition to clean technologies.

2013/73 Alejandro LAMAS and Philippe CHEVALIER. Jumping the hurdles for collaboration: fairness in operations pooling in the absence of transfer payments.

2013/74 Mehdi MADANI and Mathieu VAN VYVE. A new formulation of the European day-ahead electricity market problem and its algorithmic consequences.

2014/1 Erik SCHOKKAERT and Tom TRUYTS. Preferences for redistribution and social structure.

2014/2 Maarten VAN DIJCK and Tom TRUYTS. The agricultural invasion and the political economy of agricultural trade policy in Belgium, 1875-1900.

2014/3 Ana MAULEON, Nils ROEHL and Vincent VANNETELBOSCH. Constitutions and social networks.

Recent titles

CORE Discussion Papers - continued

2014/4 Nicolas CARAYOL, Rémy DELILLE and Vincent VANNETELBOSCH. Allocating value among farsighted players in network formation.
2014/5 Yu. NESTEROV and Vladimir SHIKHMAN. Convergent subgradient methods for nonsmooth convex minimization.
2014/6 Yuri YATSENKO, Natali HRITONENKO and Thierry BRECHET. Modeling of enrironmental adaptation versus pollution mitigation.
2014/7 Sanjeeb DASH, Oktay GÜNLÜK and Laurence A. WOLSEY. The continuous knapsack set.
2014/8 Simon BUCKLE, Mirabelle MUÛLS, Joerg LEIB and Thierry BRECHET. Prospects for Paris 2015: do major emitters want the same climate.
2014/9 Lionel ARTIGE, Antoine DEDRY and Pierre PESTIEAU. Social security and economic integration.
2014/10 Mikhail ISKAKOV, Alexey ISKAKOV and Alexey ZAKHAROV. Equilibria in secure strategies in the Tullock contest.
2014/11 Helmuth CREMER and Pierre PESTIEAU. Means-tested long term care and family transfers.
2014/12 Luc BAUWENS, Lyudmila GRIGORYEVA and Juan-Pablo ORTEGA. Estimation and empirical performance of non-scalar dynamic conditional correlation models.
2014/13 Christian M. HAFNER and Arie PREMINGER. A note on the Tobit model in the presence of a duration variable.
2014/14 Jean-François CARPANTIER and Arnaud DUFAYS. Specific Markov-switching behaviour for ARMA parameters.
2014/15 Federico GRIGIS DE STEFANO. Strategic stability of equilibria: the missing paragraph.
2014/16 Claudio TELHA and Mathieu VAN VYVE. Efficient approximation algorithms for the economic lot-sizing in continuous time.

Books

V. GINSBURGH and S. WEBER (2011), *How many languages make sense? The economics of linguistic diversity*. Princeton University Press.
I. THOMAS, D. VANNESTE and X. QUERRIAU (2011), *Atlas de Belgique – Tome 4 Habitat*. Academia Press.
W. GAERTNER and E. SCHOKKAERT (2012), *Empirical social choice*. Cambridge University Press.
L. BAUWENS, Ch. HAFNER and S. LAURENT (2012), *Handbook of volatility models and their applications*. Wiley.
J-C. PRAGER and J. THISSE (2012), *Economic geography and the unequal development of regions*. Routledge.
M. FLEURBAEY and F. MANIQUET (2012), *Equality of opportunity: the economics of responsibility*. World Scientific.
J. HINDRIKS (2012), *Gestion publique*. De Boeck.
M. FUJITA and J.F. THISSE (2013), *Economics of agglomeration: cities, industrial location, and globalization.* (2nd edition). Cambridge University Press.
J. HINDRIKS and G.D. MYLES (2013). *Intermediate public economics.* (2nd edition). MIT Press.
J. HINDRIKS, G.D. MYLES and N. HASHIMZADE (2013). *Solutions manual to accompany intermediate public economics.* (2nd edition). MIT Press.

CORE Lecture Series

R. AMIR (2002), Supermodularity and complementarity in economics.
R. WEISMANTEL (2006), Lectures on mixed nonlinear programming.
A. SHAPIRO (2010), Stochastic programming: modeling and theory.