Large Linguistic Corpus Reduction with SCP Algorithms

Nelly Barbot*
IRISA, University of Rennes 1

Olivier Boëffard*
IRISA, University of Rennes 1

Jonathan Chevelu*
IRISA, University of Rennes 1

Arnaud Delhay*
IRISA, University of Rennes 1

Linguistic corpus design is a critical concern for building rich annotated corpora useful in different domains of applications. For example, speech technologies such as ASR (Automatic Speech Recognition) or TTS (Text-to-Speech) need a huge amount of speech data to train data-driven models or to produce synthetic speech. Collecting data is always related to costs (recording speech, verifying annotations, etc.), and as a rule of thumb, the more data you gather, the more costly your application will be. Within this context, we present in this article solutions to reduce the amount of linguistic text content while maintaining a sufficient level of linguistic richness required by a model or an application. This problem can be formalized as a Set Covering Problem (SCP) and we evaluate two algorithmic heuristics applied to design large text corpora in English and French for covering phonological information or POS labels. The first considered algorithm is a standard greedy solution with an agglomerative/spitting strategy and we propose a second algorithm based on Lagrangian relaxation. The latter approach provides a lower bound to the cost of each covering solution. This lower bound can be used as a metric to evaluate the quality of a reduced corpus whatever the algorithm applied. Experiments show that a suboptimal algorithm like a greedy algorithm achieves good results; the cost of its solutions is not so far from the lower bound (about 4.35% for 3-phoneme coverings). Usually, constraints in SCP are binary; we proposed here a generalization where the constraints on each covering feature can be multi-valued.

* IRISA, University of Rennes 1, 6 rue de Kerampont, Lannion 22300, France.
E-mail: {nelly.barbot, olivier.boëffard, jonathan.chevelu, arnaud.delhay}@irisa.fr.
This work is partially supported by the French National Research Agency (ANR) in the framework of the project Phorevox.

Submission received: 7 May 2013; revised submission received: 20 November 2014; accepted for publication: 18 March 2015.

doi:10.1162/COLI_a_00225

1. Introduction

In automatic speech and language processing, many technologies make extensive use of written or read text sets. These linguistic corpora are a necessity to train models or to extract rules, and the quality of the results strongly depends on a corpus' content. Often, the reference corpus should provide a maximum diversity of content. For example, in Tian, Nurminen, and Kiss (2005), and Tian and Nurminen (2009), it turns out that maximizing the text coverage of the learning corpus improves an automatic syllabification based on a neural network. Similarly, a high quality speech synthesis system based on the selection of speech units requires a rich corpus in terms of diphones, diphones in context, triphones, and prosodic markers. In particular, Bunnell (2010) shows the importance of a good coverage of diphones and triphones for the intelligibility of a voice produced by a unit selection speech synthesis system.

To cover the best attributes needed for a task, several strategies are then possible. A first method—very simple—is to collect text randomly, but it soon becomes expensive because of the natural distribution of linguistic events following the Zipf's law. Very few events are extremely frequent and many events are very rare. This problem is often made difficult by the fact that many technologies require several variants of the same event (as in a Text-to-Speech [TTS] system using several acoustical versions of the same phonological unit). Usually, a large volume of data needs to be collected. However, and depending on the applications, building such corpora is often achieved under a constraint of parsimony. As an example, for a TTS system, a high-quality synthetic voice generally needs a huge number of speech recordings. But minimizing the duration of a recording is also a critical point to ensure uniform quality of the voice, to reduce the drudgery of the recording, to reduce the financial cost, or to follow a technical constraint on the amount of collected data for embedded systems. Moreover, a reduced set tends to limit the need of human implication for checking the data (transcription and annotation). Similarly, in the natural language processing field (NLP), the adaptation of a generic model to a specific domain often requires new annotated data that illustrate its specificities (as in Candito, Anguiano, and Seddah 2011). However, the creation cost of such data highly depends on the kind of labels used to adapt the model. In particular, the annotation in syntax trees is really more expensive than in Part-of-Speech (POS) tags. Then, it could be more efficient to annotate a compact corpus that reflects the phenomena variability than a corpus with a natural distribution of events, which implies many redundancies (see Neubig and Mori 2010).

In a machine learning framework, the active learning strategy can be used as an alternative that reduces the manual data annotation effort to design the training corpus without diminishing the quality of the model to train (see Settles 2010 or Schein, Sandler, and Ungar 2004). It consists of building the corpus iteratively by choosing an item according to an external source of information (a user or an experimental measure). This approach has been applied in NLP, speech recognition, and spoken language understanding (see for instance Tomanek and Olsson 2009 and Gotab, Béchet, and Damnati 2009).

A second alternative, when no direct quality measure is available, consists of covering a large set of attributes that may impact the final quality (after annotation or recording). This kind of approach might also be preferred when the final corpus is built in one batch (for instance, because of out-sourcing or annotator/performer consistency constraints). A method could be an automatic extraction from a huge text corpus of a minimal sized subset that covers the identified attributes. This

problem is a generalization of the **Set-Covering Problem (SCP)**, which is an NP-hard problem, as shown in Karp (1972). It is then necessary to use heuristics or sub-optimal algorithms for a reasonable computation time. Moreover, Raz and Safra (1997) and Alon, Moshkovitz, and Safra (2006) have shown that the SCP cannot be polynomially approximated with ratio $c \times \ln(n)$ unless $P = NP$, when c is a constant, and n refers to the size of the universe to cover. That means that one cannot be certain to obtain a result under this ratio with any polynomial algorithm. However, the latter complexity results are given for any kind of distribution in the mono-representation case. One can ask if good multi-represented coverages can be achieved efficiently on data following Zipf's law, which is usual in the domain of NLP.

Within the field of speech processing, the most frequently used strategy is a greedy method based on an agglomeration policy. This iterative algorithm selects the sentence with the highest score at each iteration. The score reflects the contribution of the sentence to the covering under construction. In Gauvain, Lamel, and Eskénazi (1990), this methodology has been applied to build a database of read speech from a text corpus for the evaluation of speech recognition systems using hierarchically organized covering attributes. Van Santen and Buchsbaum (1997) have tested different variants of greedy selection of texts by varying the units to cover (diphones, duration, etc.) and the "scores" for a sentence depending on the considered applications. In Tian, Nurminen, and Kiss (2005), the learning corpus for an automatic system of syllabification is designed using a greedy approach with the Levenshtein distance as a score function in order to maximize its text diversity. In François and Boëffard (2001), the methodology gives a priority to the rarest categories of allophones. The latter methodology has been implemented for the definition of the multi-speaker corpus Neologos in Krstulović et al. (2006). In the article of Krul et al. (2006), the authors constructed a corpus where the distribution of diphonemes/triphonemes matches a uniform distribution. A greedy algorithm is led by a score function based on the Kullback-Liebler divergence. A similar method is used in Krul et al. (2007) to design a reduced database in accordance with a specific domain distribution. Kawai et al. (2000) propose a pair exchange mechanism that Rojc and Kačič (2000) apply after a first reverse greedy algorithm—also called spitting greedy—deleting the useless sentences. In Cadic, Boidin, and d'Alessandro (2010), the covering of "sandwich" units (defined to be more adapted to corpus-based speech synthesis) is carried out by generating new sentences in a semi-automatic way. Candidates are generated using finite state transducers. The sentences are ordered according to a greedy criterion (their sandwiches richness) and presented to a human evaluator. This collection of artificial and rich sentences enables an effective reduction of the size of the covering but requires expensive human intervention to obtain semantically correct sentences that will be therefore easier to record.

The results of these previously cited studies are difficult to compare because of the different initial corpora and covering constraints (partial or full covering) and evaluation criteria (the number of gathered sentences, the Kullback divergence, etc.). In Zhang and Nakamura (2008), a priority policy for the rare units is added into an agglomerative greedy algorithm in order to get a covering of triphoneme classes from a large text corpus in Chinese language. The results show that this priority policy driven by the score function and the phonetic content of the sentences reduces the covering size compared with a standard agglomerative greedy algorithm.

Similarly, in François and Boëffard (2002), several combinations of greedy algorithms (agglomeration, spitting, pair exchange, or priority to rare units) were applied

to the construction of a corpus for speech synthesis in French containing at least three representatives of the most frequent diphones. Based on this work, the best strategy would be the application of an agglomerative greedy followed by a spitting greedy algorithm. During the agglomeration phase, the score of a sentence corresponds to the number of its unit instances that remain to be covered normalized by its length. During the spitting phase, at each iteration, the longest redundant sentence is removed from the covering. This algorithm is called the **Agglomeration and Spitting Algorithm (ASA)**.

As an alternative to a greedy algorithm, which is sub-optimal, solving the SCP using Lagrangian relaxation principles can provide an exact solution for problems of reasonable size. However, for speech processing, the SCP has several millions of sentences with tens of thousands of covering features. Considering these practical constraints, Chevelu et al. (2007) adapted a Lagrangian relaxation based algorithm proposed by Caprara, Fischetti, and Toth (1999). In the context of Italian railways, Caprara, Fischetti, and Toth proposed heuristics to solve scheduling problems and won a competition, called Faster, organized by the Italian Operational Research Society in 1994, ahead of other Lagrangian relaxation heuristics–based algorithms, like Ceria, Nobili, and Sassano (1998).

In Chevelu et al. (2007, 2008), the algorithm takes into account the constraints of multi-representation. A minimal number of representatives for the same unit may be required. The proposed algorithm, called **LamSCP — Lagrangian-based Algorithm for Multi-represented SCP** — is applied to extract coverings of diphonemes with a mono- or a 5-representation and coverings of triphonemes with mono-representation constraints. These results are compared with the greedy strategy ASA and are about 5% to 10% better. Besides, the LamSCP provides a lower bound for the cost of the optimal covering and allows for evaluating the quality of the results.

In Barbot, Boëffard, and Delhay (2012), phonological content of diphoneme coverings is studied regarding many parameters. These coverings are obtained by different algorithms (LamSCP, ASA, greedy based on the Kullback divergence) and some of the coverings are randomly completed to reach a given size (from 20,000 to 30,000 phones). It turns out that the coverings obtained using LamSCP and ASA provide a good representation of short units and the representation of long units mainly depends on the length of the corpus.

In this article, we present in more detail the LamSCP algorithm and its score functions and heuristics that take into account multi-representation constraints. We deepen the study about the performance of LamSCP for the construction of a phonologically rich corpus according to the size of the search space. We evaluate LamSCP and ASA algorithms on a corpus of sentences in English for a covering of multi-represented diphones, where the minimal number of required unit representatives varies from one to five. We also compare them in the case of very constrained triphoneme coverings in English and French, which represent about 12 times more units to cover. Additionally, both algorithms are tested to provide multi-represented coverings of POS tags in order to assess their ability to deal with different kinds of linguistic data. A particular effort has been made on methodology to obtain comparable measures, to study the stability of both algorithms, and to establish confidence intervals for each solution.

This article is organized as follows. In Section 2, the SCP framework and the associated notations are introduced. The ASA algorithm is described in Section 3 and the LamSCP is detailed in Section 4. The experimental methodology is presented in Section 5 and results are discussed in Section 6. Before concluding in Section 8, we present experiments in the context of TTS where we evaluate on that task the benefits of a reduction in section 7.

2. The Set-Covering Problem

Before describing the SCP-solving algorithms proposed in this article, we introduce in this section some notations and the Lagrangian properties used by LamSCP.

Let us consider a corpus $\mathcal{A}$ composed of n sentences $s_1, \ldots, s_n$. According to the target applications, these sentences are annotated with respect to phonological, acoustic, prosodic attributes, and so forth. Each sentence is then associated with a family of units of different types. The set of units present in $\mathcal{A}$ is denoted $\mathcal{U} = \{u_1, \ldots, u_m\}$ and $\mathcal{A}$ can be represented by a matrix $A = (a_{ij})$, where a_{ij} is the number of instances of unit u_i in the sentence s_j. Therefore, the jth column of A corresponds to sentence s_j in $\mathcal{A}$. To simplify the writing, we define the sets $M = \{1, \ldots, m\}$ and $N = \{1, \ldots, n\}$.

For a given vector of integers $B = (b_1, \ldots, b_m)^T$, a reduction $\mathcal{X}$ of $\mathcal{A}$, also called covering of $\mathcal{U}$, is defined as a subset of $\mathcal{A}$ which contains, for every $i \in M$, at least b_i instances of u_i. It can be described by a vector $X = (x_1, \ldots, x_n)^T$ where $x_j = 1$ if s_j belongs to $\mathcal{X}$ and $x_j = 0$ otherwise. In other words, a covering is a solution $X \in \{0,1\}^n$ of the following system

$$\forall i \in M, \sum_{j \in N} a_{ij} x_j \geq b_i \tag{1}$$

that is, $AX \geq B$ where B is called the constraint vector.

Our aim is to optimize a covering according to a cost function minimization criterion. The covering cost is given by summing the costs of the sentences that compose the covering. The optimization problem can be formulated as the following SCP:

$$X^* = \arg \min_{\substack{X \in \{0,1\}^n \\ AX \geq B}} CX \tag{2}$$

where $C = (c_1, \ldots, c_n)$ is the cost vector and c_j the cost of the sentence s_j. Because of the objective to minimize the total length of the covering, we have chosen to define the cost of a sentence as one of its length features. According to the considered application, the sentence cost can be defined as its number of phones (one of our objectives is to design a phonetically rich script with a minimal speech recording duration), or its number of words, part-of-speech tags, breath groups, and so on.

In Caprara, Fischetti, and Toth (1999), Caprara, Toth, and Fischetti (2000), and Ceria, Nobili, and Sassano (1998), the studied crew scheduling problem is a particular case of Equation (2) where A is a binary matrix and $B = \mathbf{1}_{\mathbb{R}^m}$ (i.e., with mono-representation constraints). In order to ensure that Equation (1) admits a solution, we assume that, for each $i \in M$, the minimal number b_i of u_i instances required in the covering is not greater than the number $(A\mathbf{1}_{\mathbb{R}^n})_i$ of u_i instances in $\mathcal{A}$, that is $A\mathbf{1}_{\mathbb{R}^n} \geq B$. Under this assumption, $\mathcal{A}$ is the maximal size solution of Equation (1), represented by $X = \mathbf{1}_{\mathbb{R}^n}$. In the case where b_i is greater than the number of u_i instances in $\mathcal{A}$, b_i is set to $(A\mathbf{1}_{\mathbb{R}^n})_i$.

To drive the SCP algorithms during the sentence selection phase, the covering capacity μ_j of sentence s_j is defined as the number of its unit instances required in the covering in view of the constraint vector:

$$\mu_j = \sum_{i \in M} \min \{a_{ij}, b_i\} \tag{3}$$

Let us notice that μ_j does not consider the excess unit instances: For example, if s_j contains $a_{ij} = 10$ instances of u_i and at least $b_i = 3$ instances of u_i are required, the contribution of u_i to μ_j derivation only takes into account three instances of u_i.

3. Greedy Algorithm ASA

In this section, the two main steps that compose the algorithm ASA are briefly described. First, an agglomerative greedy procedure is applied to $\mathcal{A}$ so as to derive a covering. Next, a spitting greedy procedure reduces this covering in order to approach the optimal solution of Equation (2).

3.1 Agglomerative Greedy Strategy

The greedy strategy builds a sub-optimal solution to the SCP Equation (2) in an iterative way. At each iteration, the lowest cost sentence is chosen from $\mathcal{A}$. If several sentences correspond to the lowest cost, the one coming first (i.e., the one with the lowest index) is chosen. Initially, the set of selected sentences $\mathcal{X}$ is empty, the matrix $\tilde{A}$ associated with the candidate sentences is assigned to A, the current covering capacity of s_j is given by $\tilde{\mu}_j = \mu_j$, and the current constraint vector $\tilde{B} = B$. The cost of sentence s_j is defined by

$$\sigma_j = \begin{cases} c_j/\tilde{\mu}_j \text{ if } \tilde{\mu}_j \neq 0 \\ \infty \quad \text{otherwise} \end{cases} \tag{4}$$

Indeed, if $\tilde{\mu}_j = 0$, it turns out that s_j does not cover any unit missing in the solution $\mathcal{X}$ under construction and its infinite cost σ_j avoids its selection.

At each iteration, the selected sentence s is added to $\mathcal{X}$. Taking into account the content of s, $\tilde{B}$ is updated to $\max\left\{\tilde{B} - \tilde{A}\Delta, \mathbf{0}_{\mathbb{R}^m}\right\}$ where the jth entry of Δ equals 1 if $s_j = s$ and 0 otherwise. Next, the associated column of s in $\tilde{A}$ is set to $\mathbf{0}_{\mathbb{R}^m}$. For each sentence s_j with a non-zero $\tilde{\mu}_j$ feature, $\tilde{\mu}_j$ is then updated using $\tilde{A}$ and $\tilde{B}$ in Equation (3).

At last, the agglomerative greedy algorithm is stopped as soon as all the constraints are satisfied, that is, $\tilde{B} = \mathbf{0}_{\mathbb{R}^m}$.

3.2 Spitting Greedy Strategy

The spitting greedy strategy also consists in building iteratively a sub-optimal solution $\mathcal{Y}$ to Equation (2) by reducing the size of a covering. The initial covering $\mathcal{Y}$ is set to the solution $\mathcal{X}$ derived by the agglomerative phase described earlier. At each iteration, the set of the redundant sentences of $\mathcal{Y}$ is calculated and the costliest one (according to the cost function C) is removed from $\mathcal{Y}$. An element s of $\mathcal{Y}$ is said to be redundant if for each $u_i \in \mathcal{U}$, its number of instances into $\mathcal{Y}$, denoted $m_i(\mathcal{Y})$, and into $\mathcal{Y} \setminus \{s\}$, denoted $m_i(\mathcal{Y} \setminus \{s\})$, check $\min\{m_i(\mathcal{Y}), b_i\} = \min\{m_i(\mathcal{Y} \setminus \{s\}), b_i\}$. In other words, s is a redundant element of the covering $\mathcal{Y}$ if $\mathcal{Y} \setminus \{s\}$ is also a covering solution of Equation (1). The spitting greedy algorithm stops when the redundant sentence set is empty.

4. Lagrangian Relaxation Based–Algorithm

This section describes the main phases of the algorithm called LamSCP. This algorithm takes advantage of the Lagrangian relaxation properties reviewed herein in order to approach the optimal solution of Equation (2) as close as possible. Strongly inspired by

Caprara, Fischetti, and Toth (1999), but generalized to the multi-representation problem, this algorithm provides a lower bound of the optimal solution cost. Having such information is very useful for assessing the achievements of the SCP algorithms.

4.1 Lagrangian Relaxation Principles

Let us briefly recall the main principles of Lagrangian relaxation on which LamSCP is based to solve Equation (2) (see Fisher [1981] for more details on Lagrangian relaxation).

First, the Lagrangian function associated with Equation (2) is defined by

$$\begin{aligned} L(X,\Lambda) &= CX + \Lambda^T(B - AX) \\ &= \Lambda^T B + C(\Lambda)X \end{aligned} \tag{5}$$

where $\Lambda \in \left(\mathbb{R}_+\right)^m$, $X \in \{0,1\}^n$, and $C(\Lambda) = C - \Lambda^T A$. The coordinates of $\Lambda = (\lambda_1, \ldots, \lambda_m)^T$ are called **Lagrangian multipliers** and can be interpreted as a weighting of constraints (1). The jth entry of $C(\Lambda)$, called **Lagrangian cost** $c_j(\Lambda)$ of sentence s_j, takes into account its cost c_j and the adequacy of its composition to address Equation (2). For every covering X and every $\Lambda \in \left(\mathbb{R}_+\right)^m$, the Lagrangian function satisfies $L(X,\Lambda) \leq CX$. Thus, the dual Lagrangian function defined by

$$L(\Lambda) = \min_{X \in \{0,1\}^n} L(X,\Lambda) \tag{6}$$

presents the following fundamental property: For every $\Lambda \in \mathbb{R}_+^m$ and every covering X, we have $L(\Lambda) \leq CX$. Hence, $L(\Lambda)$ is a lower bound of the minimal covering cost, CX^*, but does not necessarily correspond to the cost of a covering. In order to compute $L(\Lambda)$, an acceptable solution for the vector X minimizing $L(X,\Lambda)$ is $X(\Lambda) = (x_1(\Lambda), \ldots, x_n(\Lambda))^T$ where

$$\begin{aligned} x_j(\Lambda) = &\begin{cases} 1 \text{ if } c_j(\Lambda) < 0 \\ 0 \text{ if } c_j(\Lambda) > 0 \end{cases} \\ &\in \{0,1\} \text{ otherwise} \end{aligned} \tag{7}$$

Additionally, the dual Lagrangian function and the Lagrangian costs inform about the potential usefulness of sentences in the optimal covering. More precisely, for a given Λ and a known upper bound UB of minimal covering cost, the gap $g(\Lambda) = \text{UB} - L(\Lambda)$ measures the relaxation quality. If $c_j(\Lambda)$ is strictly greater than $g(\Lambda)$, we can check that any covering containing s_j has a cost value strictly greater than UB. Hence, sentence s_j is not selected and x_j can be fixed at zero. Similarly, if $c_j(\Lambda) < -g(\Lambda)$, any covering with a cost lower than UB contains s_j and one can fix x_j to 1. Therefore, an optimal covering is made up of sentences with a low Lagrangian cost, as done in Caprara, Toth, and Fischetti (2000) and Ceria, Nobili, and Sassano (1998), and the higher the relaxation quality (i.e., the lower $g(\Lambda)$) is, the cheaper the covering will be.

The resolution of the dual problem of Equation (2) consists in finding $\Lambda^* \in \mathbb{R}_+^m$ that maximizes the lower bound $L(\Lambda)$, that is

$$\Lambda^* = \arg\max_{\Lambda \in \mathbb{R}_+^m} L(\Lambda) \tag{8}$$

Because this real variable function L is concave and piecewise affine, a well-known approach for finding a near-optimal multiplier vector is the subgradient algorithm.

4.2 The Three Phases

The LamSCP is an iterative algorithm, composed of several procedures that aim to either improve the current best solution or reduce the combinatorial issue related to the considered problem. In order to derive a good solution, the algorithm calls on a great number of greedy procedures to solve sub-problems with the help of the Lagrangian costs. As for the combinatorial reduction, the most frequently used heuristic consists of downsizing the problem by mainly considering the sentences with low Lagrangian costs.

The algorithm is organized around a main procedure called **3-phases**. This procedure can single-handedly solve a multi-represented SCP. As its name suggests, the **3-phases** functioning consists in iterating a sequence of the three following sub-procedures as shown in Figure 1:

- The **subgradient phase** calculates an estimation $\tilde{\Lambda}$ of Λ^* that maximizes the dual Lagrangian function. This procedure requires an upper bound UB of the optimal covering cost. UB is initialized by a greedy algorithm (rather than the cost of the whole corpus $\mathcal{A}$). This phase is detailed in Section 4.2.1.
- The **heuristic phase** explores the neighborhood of $\tilde{\Lambda}$ by generating a great number of Lagrangian vectors $\tilde{\Lambda}^p$. A greedy-like procedure is associated with each $\tilde{\Lambda}^p$ so as to compute a covering using the Lagrangian cost vector $C(\tilde{\Lambda}^p)$. If, during this exploration, a less costly covering than the best known one (corresponding to the cost UB) is found, the upper bound UB is then updated to the cost of this less costly solution. Similarly, if a better estimation of Λ^* is obtained, $\tilde{\Lambda}$ is updated. This phase is described in Section 4.2.2.
- The **column fixing phase** selects a set $\mathcal{F}$ of sentences that are most likely to belong to the optimal covering. This phase is detailed in Section 4.2.3.

Following the column fixing phase, the constraint vector is updated and the unselected sentences define a set-covering sub-problem, called a **residual problem**. This sub-problem is processed similarly, via an additional iteration of the three phases. This iterative process is stopped when the residual problem is empty or when the associated dual Lagrangian function indicates a cost is too high. Indeed, because this function indicates a minimal cost for covering the sub-problem, its addition to the cost CF of the sentences already retained in $\mathcal{F}$ gives a lower bound of the total cost of the solution under construction, which should not rise beyond the cost UB of the best known solution so as to be potentially more advantageous.

4.2.1 Subgradient Phase. In order to reach the quality goal, the **subgradient phase** provides a near-optimal solution $\tilde{\Lambda}$ of the dual Lagrangian problem (8) using a subgradient

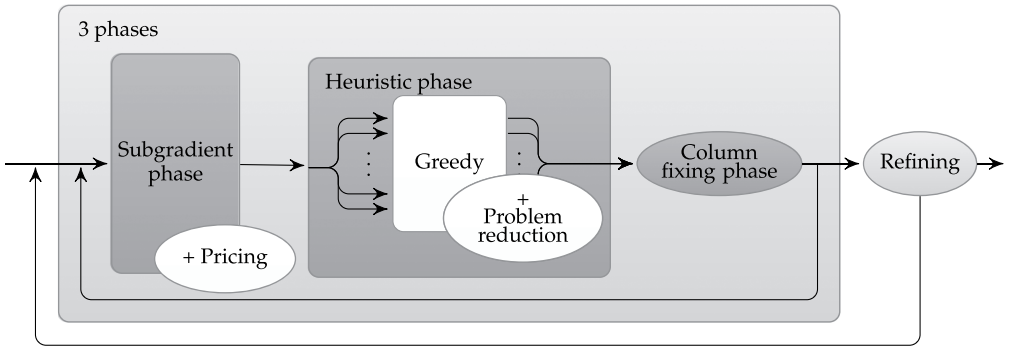


Figure 1
LamSCP algorithm. The rectangle blocks are the steps that aim at improving solution quality and the ellipses are those that try to reduce the size of the problem.

type algorithm. This iterative approach generates a sequence (Λ^p) using the following updating formula (see Caprara, Fischetti, and Toth 1999)

$$\Lambda^{p+1} = \max\left\{\Lambda^p + \mu\frac{g(\Lambda^p)}{||S(\Lambda^p)||^2}S(\Lambda^p), 0\right\} \tag{9}$$

where $S(\Lambda^p) = B - AX(\Lambda^p)$ so as to take into account the multi-representation constraints. Parameter μ is adjusted to fit the convergence fastness according to the method proposed by Caprara, Fischetti, and Toth (1999).

At the first call of 3-phases, Λ^0 is arbitrarily defined as follows: for each $i \in M$,

$$\lambda_i^0 = \min_{\substack{j\in N\\ a_{ij}\neq 0}} \frac{c_j}{\mu_j} \tag{10}$$

As for UB, its initial value is set to the cost of a covering previously calculated. In order to evaluate how much the covering cost derived by ASA can be improved, we have chosen to initialize UB by this value. At the following iterations of 3-phases, Λ^0 is given by a random perturbation (less than 10%) of the best known vector $\tilde{\Lambda}$ (of which the entries of the sentences fixed in the last column fixing phase are removed) and UB corresponds to the cost of the best covering found (after subtraction of the cost of the sentences fixed in the last column fixing phase). In another approach proposed by Ceria, Nobili, and Sassano (1998), UB corresponds to the upper bound of a dual Lagrangian problem, and the subgradient procedure simultaneously estimates the upper bound and the lower bound, generating two sequences of multipliers.

The subgradient phase also calls two procedures: **pricing** and **spitting**. Procedure pricing aims to reduce the size of the search space. For each unit u_i, the pricing selects the $5b_i$ smallest Lagrangian cost sentences covering u_i. If this selection contains less than $5m$ sentences, where m is the maximal entry of B, it is completed by low Lagrangian cost sentences (less than 0.1) to the limit of $5m$ sentences. The set of the chosen sentences is denoted $\mathcal{P}$ and its design guarantees a sufficient number of instances for each unit to cover and a small variety in its composition. Actually, the subgradient method is applied on $\mathcal{P}$ instead of $\mathcal{A}$, and $\mathcal{P}$ is updated every 10 subgradient iterations.

Finally, at each iteration, definition (7) of $X(\Lambda^p)$ and the large number of Lagrangian costs close to zero allow a considerable number of vectors $S(\Lambda^p)$. In order to get around the computational difficulty of finding the steepest accent direction, Caprara, Fischetti, and Toth (1999) propose a heuristic that, according to the experimental results, accelerates the convergence of the subgradient phase. This heuristic is implemented in the spitting procedure. Called at each iteration, this procedure extracts from $\mathcal{P}$ the subset $\mathcal{S}$ of sentences with a Lagrangian cost lower than 0.001. $\mathcal{S}$ is then reduced using a spitting greedy algorithm to remove its redundant elements in decreasing Lagrangian cost order. At last, for every $j \in M$, $x_j(\Lambda^p) = 1$ if $s_j \in \mathcal{S}$ and $x_j(\Lambda^p) = 0$ otherwise. Thus, $S(\Lambda^p)$ does not necessarily correspond to a subgradient vector.

4.2.2 Heuristic Phase. The heuristic phase calculates a large number of coverings before keeping the best one. To that end, a sequence of 150 multiplier vectors is generated by perturbing $\tilde{\Lambda}$ using the formula $\tilde{\Lambda}^{p+1} = \max\left\{\tilde{\Lambda}^p + \mu g(\tilde{\Lambda}^p)S(\tilde{\Lambda}^p), 0\right\}$ where $\tilde{\Lambda}^0 = \tilde{\Lambda}$ and μ is provided by the subgradient phase, so as to allow for a change in a large number of $\tilde{\Lambda}^p$. With each $\tilde{\Lambda}^p$, an agglomerative greedy algorithm followed by a spitting greedy one are associated in order to calculate a covering.

The agglomerative greedy chooses at each iteration the sentence s_j with the lowest cost $\sigma_j(\tilde{\Lambda}^p)$ where

$$\sigma_j(\tilde{\Lambda}^p) = \begin{cases} c_j(\tilde{\Lambda}^p) * \tilde{\mu}_j & \text{if } c_j(\tilde{\Lambda}^p) < 0 \text{ and } \tilde{\mu}_j > 0 \\ c_j(\tilde{\Lambda}^p)/\tilde{\mu}_j & \text{if } c_j(\tilde{\Lambda}^p) \geq 0 \text{ and } \tilde{\mu}_j > 0 \\ \infty & \text{if } \tilde{\mu}_j = 0 \end{cases} \tag{11}$$

This cost function provides an advantage to low Lagrangian cost sentences s_j containing $\tilde{\mu}_j$ unit instances that could be helpful to the covering under construction. The agglomerative step uses several heuristics so as to reduce the search space. It is run within a limited subset $\mathcal{P}_l$ of $\mathcal{P}$, composed of the sentences s_j with the lowest costs $\sigma_j(\tilde{\Lambda}_k)$. At each iteration, a sentence of $\mathcal{P}_l$ is selected and the cost of the sentences of $\mathcal{P}$ are updated. If the maximum sentence cost in $\mathcal{P}_l$ becomes greater than the minimal cost in $\mathcal{P} \setminus \mathcal{P}_l$, the working subset $\mathcal{P}_l$ is also updated. The definitions of $\mathcal{P}$ and $\mathcal{P}_l$ guarantee that the agglomeration step provides a nonpartial solution of the considered SCP. This solution is then reduced during the spitting step by iteratively removing its redundant sentences s_j with the highest costs c_j.

At the end of the heuristic phase, the best found covering $\mathcal{X}^*$ and its cost CX^* (stored in UB) are kept as well as the highest value of $L(\tilde{\Lambda}^p)$ (found during the subgradient or heuristic phases).

4.2.3 Column Fixing Phase. The column fixing phase aims to reduce the problem size by choosing "promising" sentences among the ones with very low Lagrangian cost or containing rare unit instances. The unselected sentences are less interesting for resolving the SCP and the residual problem associated with these sentences should be the subject of another call of the 3-phases.

More precisely, the column fixing phase calculates the subset $\mathcal{Q}$ composed of sentences s_j with a negative Lagrangian cost $c_j(\tilde{\Lambda})$. $\mathcal{Q}$ is represented by the binary vector $Q = (q_1, \ldots, q_n)^T$ where $q_j = 1$ if $s_j \in \mathcal{Q}$. For each $u_i \in \mathcal{U}$, the number of its instances covered by $\mathcal{Q}$ is given by $(AQ)_i$. If $(AQ)_i \leq b_i$, then u_i is considered as a rare unit and all the elements of $\mathcal{Q}$ containing some instances of u_i are fixed in a set $\mathcal{F}$. The covering constraints that are not satisfied by $\mathcal{F}$ constitute a residual SCP. In order to

complete $\mathcal{F}$ with a few sentences of the best known covering, a greedy-type algorithm is run on $\mathcal{X}^* \setminus \mathcal{F}$ to derive a solution of this residual SCP. From the obtained solution, the $\max\{\left(B^T \mathbf{1}_{\mathbb{R}^m}\right)/20, 1\}$ lowest Lagrangian cost sentences are also added in $\mathcal{F}$. The sentences that are "fixed" in $\mathcal{F}$ during the column fixing phase stay in $\mathcal{F}$ up to the end of the 3-phases. After the column fixing phase, the residual sub-problem is processed by iterating the three phases and the next fixed sentences are added to $\mathcal{F}$.

4.3 Refining Procedure

The 3-phases procedure is encapsulated in an outer loop that permits the partial reconsideration of the solution $\mathcal{X}^*$ provided by this procedure. To that end, the **refining** procedure, proposed in Caprara, Fischetti, and Toth (1999), is used in order to select elements of $\mathcal{X}^*$ that contribute at least to the gap $g(\tilde{\Lambda})$. In the case of the SCP with multi-representation constraints, the definition of the contribution of $s_j \in \mathcal{X}^*$ can be adapted as follows:

$$\delta_j \;=\; \max\{c_j(\tilde{\Lambda}), 0\} + \sum_{\substack{i \in M \\ a_{ij} > 0}} \tilde{\lambda}_i (AX^* - B)_i \frac{a_{ij}}{(AX^*)_i} \tag{12}$$

The second term of Equation (12) consists of sharing the contribution $\tilde{\lambda}_i(AX^* - B)_i$ of the excess instances of u_i in $\mathcal{X}^*$ according to the distribution of u_i instances in $\mathcal{X}^*$. Therefore, the refining procedure ranks in an increasing order the elements s_j of $\mathcal{X}^*$ according to their δ_j value, and fixes the first elements in a set $\mathcal{G}$ until its given covering rate $\tau_{\mathcal{G}}$ reaches π. $\tau_{\mathcal{G}}$ represents the rate of covering constraints satisfied by $\mathcal{G}$ and is defined by

$$\tau_{\mathcal{G}} \;=\; 1 - \frac{\sum_{i \in M} \max\{b_i - (AG)_i, 0\}}{B^T \mathbf{1}_{\mathbb{R}^m}} \tag{13}$$

where G denotes the binary vector corresponding to $\mathcal{G}$.

4.4 The Overall LamSCP

The LamSCP is made up of the main procedures introduced in the previous sections interlinked by the following steps. First, because of the adaptation of the algorithm to the SCP with multi-representation constraints, the entries of matrix A are clipped to the constraint vector in order to simplify the calculations such as the ones of $\mu_1, \ldots, \mu_n$. This threshold application implies that the excess instances of each unit u_i are taken into account in each sentence s_j beyond the number b_i, in the derivation of δ_j.

After the initialization of Λ^0 using Equation (10) and the upper bound UB, procedure 3-phases is called and provides a solution $\mathcal{X}$ to the complete SCP. The *refining* function fixes a sentence subset $\mathcal{G}$ such that $\mathcal{G}$ covers a given rate π of the covering constraints. Parameter π starts at a minimum value $\pi_{\min} = 0.3$. The residual SCP is then processed by 3-phases. The π value grows at a rate of 20 percent whenever 3-phases does not improve the solution to the complete SCP. If π is greater or equal to 1, the *refining* procedure fixes the whole best solution and the residual problem is then empty. On the other hand, π is set to $\pi_{\min}$ if a better solution is found in order to challenge $\mathcal{G}$ and improve this solution. This iterative sequence composed of the 3-phases and *refining* procedures is carried out until the residual problem is not empty, the gap $g(\tilde{\Lambda})$ is positive, and the number of iterations has not reached 20.

5. Experiments

We propose a twofold comparison of the ASA and LamSCP algorithms: One part is focused on the covering cost for a large SCP, and the other on the stability of the solutions. Moreover, in order to assess the ability and the behavior of both algorithms to process different linguistic data, a first set of experiments deals with phonological attributes (mainly covering co-occurrences of phonemes) and a second set with grammatical labels (mainly covering co-occurrences of POS labels).

Both attribute types, often involved in automatic linguistic processing, were chosen because their distribution consists of few highly frequent events and numerous rare events. On the one hand, for TTS tasks, the phonological type covering is a useful preliminary step of the text corpus design before the recording step. In order to produce the signal corresponding to a requested sentence, the unit selection engine requires at least one instance of each phone (or 2-phone, depending on the concatenation process). Because the recording and the post-recording annotation process are expensive tasks, the recording length of such a corpus has to be as short as possible. On the other hand, in order to train a domain-specific dependency parser, the covering of POS sequences may be useful for increasing the diversity of syntax patterns. Because the dependency annotation is a highly expensive task, the adaptation corpus to annotate needs to be as small as possible, containing characteristic examples of the specific lexical variation rather than following the natural distribution. One can expect that increasing its diversity of POS sequences may lead to more diversity in the syntax trees.

Experiments on covering co-occurrences of phonemes are carried out on two large phonologically annotated text corpora, and consist of covering at least k instances of each phoneme, diphoneme until n-phoneme (i.e., triphoneme if $n = 3$, diphoneme if $n = 2$). The cost c_j of the sentence s_j is given by its number of phones. From this point, this kind of SCP is called a "k-covering of n-phonemes."

A first corpus, *Gutenberg*, is composed of texts in English, mainly extracted from novels and short stories. This corpus is the production of the Gutenberg Project, presented by Hart (2003), and has been used by Kominek and Black (2003) to design the speech corpus *Arctic*. A second corpus, in French, named *Le-Monde*, is extracted from articles published in the newspaper *Le Monde* in 1997. Table 1 summarizes the main features of both corpora.

The phonological annotation of the *Gutenberg* corpus comes from the Arctic/Festvox database (see Kominek and Black 2003), and the annotation of the *Le-Monde* corpus is a by-product of the Neologos project, detailed by Krstulović et al. (2006).

Table 1
Statistics of the studied corpora.

	Le-Monde	*Gutenberg*
Number of sentences	172,168	53,996
Number of phonemes	35	57
Corpus size (number of phones)	16,668,609	1,539,735
Number of diphonemes	1,172	1,955
Number of diphones	16,496,441	1,485,739
Number of triphonemes	26,443	27,477
Number of triphones	16,324,273	1,431,743
Sentence length mean (phones) & Std. Dev.	96.81 (60.46)	28.51 (10.52)

Table 2
Statistics of 1-POS and 2-POS occurrences, corpus *Le-Monde*. On average, a sentence contains 27.64 POS with a standard deviation of 16.45.

	Number of distinct units	Number of occurrences
1-POS	141	4,587,320
2-POS	6,716	4,421,371

For each corpus, we have collected every phoneme, diphoneme, triphoneme, and their occurrences in each sentence so as to define the set $\mathcal{U}$ of units to cover and the matrix A. A is built by collecting one sentence after the other following the ordering inside the corpus, and one unit after the other inside the sentences. After this matrix translation, we obtain two description files and two index files. The first file describes the matrix A and the second one the cost vector C. Because of the low matrix density, we have chosen a sparse representation to save space and computation time: For instance, the 2-phoneme *Gutenberg* matrix is about 2.2% dense. We only store the cells of A that have a non-zero value so as to get a sparse matrix. The index files are made for the correspondence between the general covering problem and the application domain. The implementation is made in C. In terms of software engineering, our algorithms are working on an SCP that does not depend on the application data. For example, there is no information on what types of units are to be covered. The algorithms only have the matrix of occurrences A, the cost vector C, and the constraint vector B. A set of translation files (from application data to SCP and from SCP to application data) is built before each computation. As a consequence, there is no difficulty in addressing a different set of features to cover on the same or on a different corpus.

To study the achievements of ASA and LamSCP on different types of data, we have also chosen to address the "k-covering of n-POS" on the corpus *Le-Monde*. The grammatical and syntactical analyses are processed by the Synapse development analyzer presented in Synapse (2011). In order to consider a SCP with a substantial number of required units, a very detailed level of POS tagging has been selected, providing 141 distinct tags after analyzing *Le-Monde*. For example, this level provides tags like "Determiner_male_singular_Article" or "Noun_female_singular," whereas the simplest level gives "Determiner" or "Noun." This latter level of description would have given only nine different POS tags after analyzing *Le-Monde*. The main associated statistics are given in Table 2. For these experiments of POS covering, the cost of a sentence is defined as its number of POS occurrences.

We used a PC with 2 CPUs (E5320/1.86GHz/4 cores/64bits) and 32 GB RAM for the phonological coverings and the POS coverings were computed using a PC with 8 CPUs (Intel Xeon X7550/2.00Ghz/8 cores/64bits) and 128 GB RAM. Our implementations do not take advantage of any parallelism.

The following sections detail more precisely the different experiments conducted on French or English.

5.1 Performance of the Algorithms for 1-Covering of 2-Phonemes in French

The aim of Experiment 1 is the assessment of the achievements of both algorithms, ASA and LamSCP, and the robustness of the results when the sentence ordering is modified

in the corpus to reduce. Indeed, one of the difficulties of the greedy methodology is that the score function has discrete values and several sentences can yield the same score. In our implementation, among the sentences showing the best current score, the first one encountered is chosen. We would like to measure the influence of this random choice on the stability of the results. LamSCP uses greedy strategies based on Lagrangian costs. Because the Lagrangian costs $c_j(\Lambda)$ take the SCP in its entirety into account and are continuous real-value functions of Λ, they would be more selective than the sentence costs used by ASA. A simple solution for evaluating the stability consists of proceeding with an important amount of experiments on the same SCP by randomly modifying the sentence ranking in $\mathcal{A}$. Experiment 1 measures the impact of these permutations on the solutions computed by both algorithms. The considered SCP is the 1-covering of 2-phonemes on the corpus *Le-Monde*. Considering the computation time (more than 5 hours for LamSCP), only 47 instances of the SCP are considered, each instance corresponding to a random sentence ordering in *Le-Monde*. The 95% confidence intervals are derived using the bootstrap method, concerning the covering cost, the number and the length of sentences in coverings, the computation time, and the "distance" between the covering costs and the associated lower bound $L(\tilde{\Lambda})$.

5.2 Stability of the Algorithms for the *k*-Covering of 2-Phonemes in English

One of the goals of Experiment 2 is to compare the achievements of both algorithms on corpus *Gutenberg*, which has different features from the ones of *Le-Monde*. The sentences in *Gutenberg* are shorter on average and the associated variation of sentence length is lower. Furthermore, *Gutenberg* is 10 times smaller than *Le-Monde*.

In order to compare with the results of the previous experiment done on *Le-Monde*, we first observed a 1-covering of 2-phonemes on the *Gutenberg* corpus. The search space seems smaller than in Experiment 1. According to Table 1, *Le-Monde* is composed of more sentences than *Gutenberg*. *Le-Monde* contains 33,165,050 occurrences of 2-phonemes and *Gutenberg* only 3,025,474. Moreover, the number of attributes to cover is lower: 1,207 2-phonemes in *Le-Monde* and 2,012 in *Gutenberg*. We can also notice that five 2-phonemes have only one occurrence in *Le-Monde* and that the total cost of the five sentences covering these rare units is 751 phones, whereas this is the case for 109 2-phonemes in *Gutenberg* and the total cost of the 104 concerned sentences is 3,606 phones. Finally, if we consider the density of matrix A, 8.4% of the cells are non-empty for Experiment 1 and 2.2% for Experiment 2. As with Experiment 1, the sentence ordering in *Gutenberg* has been randomly modified to produce 60 instances of the SCP and similar solution statistics have been computed for both algorithms.

The second objective is to test and compare the ability of two algorithms to deal with the constraints of multi-representation. For this, we apply the same methodology to the *k*-covering of 2-phonemes in *Gutenberg*, for k from 2 to 5. We note that for the same original corpus to reduce, the size of the search space decreases when k increases. These different SCPs enable us to compare the performance of two algorithms depending on the size of the search space.

5.3 1-Covering of 3-Phonemes in English

In Experiment 3, the aim is to observe the behavior of both algorithms on very constrained problems. For this, we study their ability to treat a covering of 3-phonemes. We try to assess the impact on the solution features and on the stability of such an increase in the number of attributes to cover with many rare events. So as to compute statistics

on the 1-covering of 3-phonemes, an instance of *Gutenberg* has been proposed to ASA and to LamSCP. This instance counts 29,489 units to cover and the density of matrix A is 0.24%. The computation time is nearly 5 days for LamSCP and we then chose to carry out 35 instances of the 1-covering of 3-phonemes on *Gutenberg*. Additionally, it is interesting to compare these results with those of Experiment 2 concerning the 1-covering of 2-phonemes on the same corpus, which corresponds to a larger search space.

5.4 1-Covering of 3-Phonemes in French

In order to pursue the objective set out in the description of Experiment 3, that is, the ability of algorithms to treat with numerous constraints and a heavy-tail distribution of units, Experiment 4 consisted of testing both algorithms on the 1-covering of 3-phonemes on *Le-Monde*. The search space seems larger than in the previous experiment. Let us recall that *Le-Monde* contains 3.18 times more sentences than *Gutenberg* and it counts 27,650 units to cover. Furthermore, *Gutenberg* contains 5,000 3-phonemes with only one occurrence, which requires the selection of 4,180 sentences with a total length equal to 137,714 phones, whereas *Le-Monde* contains 2,274 rare 3-phonemes scattered in 2,107 sentences measuring a total of 283,208 phones. The associated matrix density is 0.69%. Because the computation of a first instance takes more than 8 days, we have limited the number of instances to 30 for this SCP.

5.5 k-Covering of 1-POS and 2-POS in French

The main goal of Experiment 5 is to study the behavior of both algorithms, ASA and LamSCP, dealing with another kind of linguistic attribute, and to compare this with the previous experiments. To achieve this goal, we consider POS attributes and the associated SCP: 1- and 5-coverings of POS, 1- and 5-coverings of 2-POS defined on *Le-Monde*. Indeed, we can observe in Table 2 that the global statistics of POS tags in *Le-Monde* are quite different from their phonological counterparts summarized in Table 1. In particular, the density of matrix A is 11.03% for a 1-POS covering and 0.57% for a 2-POS covering. Also, the search space size seems to decrease when considering successively the mono- and the multi-coverings of 1-POS, and the mono- and the multi-coverings of 2-POS, permitting us to compare them with the results coming from the experiments on phonological coverings. We evaluate the stability by computing 50 randomly mixed versions of the corpus *Le-Monde*.

6. Results and Discussion

In this section, the results of the experiments described in Section 5 are provided and discussed. As a consequence, the organization of this section and Section 5 are similar.

6.1 Performance of Algorithms for 1-Covering of 2-Phonemes in French

Table 3 shows the main results of Experiment 1, concerning the 1-covering of 2-phonemes from the corpus *Le-Monde*. Symbol $\pm$ indicates that the mentioned value corresponds to a 95% confidence interval, calculated using the bootstrap method from

Table 3
Statistics of the solutions of 1-covering of 2-phonemes computed by ASA and LamSCP from *Le-Monde*. The last column represents the best lower bounds found by LamSCP.

Experiment 1: 1-covering of 2-phonemes, corpus *Le-Monde*			
	ASA	LamSCP	$L(\tilde{\Lambda})$
Covering size (phones)	8,555 ± 22	7,786 ± 4	7,689 ± 5
Reduction rate relative to ASA (%)		**–9.00 ± 0.20**	**–10.13 ± 0.19**
Covering size [min; max]	[8,447; 8,669]	[7,767; 7,829]	[7,649; 7,715]
Covering size Std. Dev.	57.40	13.01	13.83
Sentence number	335.73 ± 1.69	268.76 ± 1.08	–
Sentence length	25.48 ± 0.10	28.97 ± 0.12	–
Time CPU (seconds)	51 ± 0	20,478 ± 508	–

the 47 instances of the SCP. In order to cover each of the 1,207 2-phonemes of *Le-Monde*, ASA drastically reduces the size of the initial corpus by 99.94% (±0.00). However, on average, LamSCP calculates a 9.00% shorter covering. The lower bound $L(\tilde{\Lambda})$ for the optimal covering cost is 7,689 ± 5 phones. $L(\tilde{\Lambda})$ is not a minimum value and may not correspond to the cost of a real covering. Because this lower bound is updated all along the execution of LamSCP, we do not mention a specific calculation time for this result.

For one instance of the SCP, let CX^*_{ASA} be the size of the solution given by ASA. The quantity $\tau_{\text{ASA}} = 1 - L(\tilde{\Lambda})/CX^*_{\text{ASA}}$ indicates that the optimum solution to SCP is at most τ_{ASA} times shorter than the covering calculated by ASA. It can be observed that the optimal solution is at most 10.13% (±0.19) shorter than the one yielded by ASA and at most 1.24% (±0.08) shorter than the solution yielded by LamSCP. The solutions obtained by LamSCP and the optimal solution to the SCP are therefore very close. Considered among the 47 instances of the SCP the best solutions yielded by ASA (8,447 phones) and LamSCP (7,767 phones), LamSCP is 8.75% better than ASA in terms of covering costs, while the best lower bound for the SCP is 7,715 phones, only 0.67% (respectively, 8.66%) shorter than the best covering by LamSCP (respectively, ASA).

The average length of the sentences selected by both algorithms is far below the average length of the sentences in the corpus (96.81 phones). LamSCP tends to choose sentences that are slightly longer than ASA, with an average 28.97 (±0.12) phones compared with 25.48 (±0.10) phones. Moreover, ASA selects on average 335.73 (±1.69) sentences per solution, about 24.91% more than LamSCP, which selects 268.76 (±1.08) sentences on average. This seems to indicate that LamSCP makes fewer local choices than ASA. This hypothesis can also be validated through the analysis of the variability of the results. The relative variation of the covering costs calculated by LamSCP is 13.01/7,786 = 0.16%, and 57.40/8,555 = 0.67% by ASA; that is to say a stability of the costs 4 times greater for the solutions yielded by LamSCP than for ASA. Moreover, the solutions are composed of a very stable number of sentences: The associated relative standard deviation is 5.31/335.73 = 1.58% for the 47 instances solved by ASA, and 3.85/268.76 = 1.43% for the instances solved by LamSCP. It turns out that the results of both algorithms are very stable when the order of the sentences is modified in the original corpus.

Finally, concerning computation time, the resolution of an instance of the SCP lasts on average 5 hr 41 min 18 sec (±8 min 28 sec) for LamSCP versus 51 sec

(±0 sec) for ASA. On average over the 47 instances, LamSCP takes 390 (±9) times as long as ASA.

6.2 Stability of the Algorithms for k-Covering of 2-Phonemes in English

The considered SCP consists of covering at least k times each of the 2,012 2-phonemes of the *Gutenberg* corpus, with k varying from 1 to 5. The results are summarized in Table 4.

For all instances of these SCPs, it has been observed that LamSCP computes shorter coverings than ASA. However, that advantage diminishes as k grows: The cost advantage offered by LamSCP compared with ASA decreases from 9.73% (±0.13) for $k = 1$ to 4.50% (±0.04) for $k = 5$. Also, the solutions obtained from ASA and LamSCP seem to get closer to the optimal solution as k rises. The corresponding figures are presented in Table 5: For instance, the optimal solution is at most 0.75% (±0.02) shorter than that obtained by LamSCP for $k = 1$, and 0.27% (±0.00) for $k = 5$.

Table 4
Experiment 2: Statistics based on 60 instances of a k-covering of 2-phonemes from *Gutenberg*.

		ASA	LamSCP	$L(\tilde{\Lambda})$
$k = 1$	Covering size (phones)	14,909 ± 23	13,458 ± 2	13,357 ± 2
	Reduction rate relative to ASA (%)		**–9.73 ± 0.13**	**–10.41 ± 0.12**
	Covering size [min; max]	[14,700; 15,107]	[13,441; 13,485]	[13,341; 13,378]
	Sentence number	623.75 ± 1.61	520.85 ± 0.57	–
	Sentence length	23.90 ± 0.04	25.83 ± 0.02	–
	Time CPU (seconds)	8 ± 0	2,711 ± 68	–
$k = 2$	Covering size (phones)	27,518 ± 25	25,604 ± 4	25,413 ± 1
	Reduction rate relative to ASA (%)		**–6.94 ± 0.09**	**–7.65 ± 0.08**
	Covering size [min; max]	[27,278; 27,727]	[25,576; 25,642]	[25,400; 25,430]
	Sentence number	1080.19 ± 1.60	948.30 ± 0.68	–
	Sentence length	25.48 ± 0.02	26.99 ± 0.01	–
	Time CPU (seconds)	14 ± 0	3,931 ± 68	–
$k = 3$	Covering size (phones)	39,319 ± 34	36,985 ± 4	36,746 ± 2
	Reduction rate relative to ASA (%)		**–5.92 ± 0.07**	**–6.53 ± 0.07**
	Covering size [min; max]	[39,091; 39,580]	[36,946; 37,028]	[36,724; 36,769]
	Sentence number	1,507.75 ± 1.75	1,359.28 ± 1.13	–
	Sentence length	26.07 ± 0.01	27.20 ± 0.02	–
	Time CPU (seconds)	20 ± 0	5,974 ± 114	–
$k = 4$	Covering size (phones)	50,491 ± 28	48,023 ± 4	47,820 ± 4
	Reduction rate relative to ASA (%)		**–4.89 ± 0.06**	**–5.29 ± 0.05**
	Covering size [min; max]	[50,300; 50,753]	[47,987; 48,075]	[47,780; 47,842]
	Sentence number	1,908.10 ± 1.73	1,744.40 ± 1.35	–
	Sentence length	26.46 ± 0.01	27.53 ± 0.01	–
	Time CPU (seconds)	28 ± 0	6,589 ± 239	–
$k = 5$	Covering size (phones)	61,375 ± 30	58,610 ± 4	58,447 ± 2
	Reduction rate relative to ASA (%)		**–4.50 ± 0.04**	**–4.76 ± 0.04**
	Covering size [min; max]	[61,137; 61,683]	[58,582; 58,645]	[58,420; 58,465]
	Sentence number	2,288.91 ± 1.88	2,101.62 ± 0.84	–
	Sentence length	26.81 ± 0.01	27.88 ± 0.01	–
	Time CPU (seconds)	39 ± 0	7,599 ± 344	–

Table 5
Ratios τ_{LamSCP} and τ_{ASA} for the k-covering of 2-phonemes from *Gutenberg*.

k	1	2	3	4	5
τ_{LamSCP}	0.75% ($\pm$ 0.02)	0.74% ($\pm$ 0.01)	0.64% ($\pm$ 0.01)	0.42% ($\pm$ 0.01)	0.27% ($\pm$ 0.00)
τ_{ASA}	10.41% ($\pm$ 0.12)	7.65% ($\pm$ 0.08)	6.53% ($\pm$ 0.07)	5.29% ($\pm$ 0.05)	4.76% ($\pm$ 0.04)

Because the search area diminishes as k increases, it may be observed that the algorithms tend to be more stable. This is true both for the size of the solutions, as well as for the number of sentences that define them. Table 6 represents the variation of the size of the solutions as a function of k. This variation is calculated as follows: For a given k number and a given algorithm, the standard deviation of the size of the k-covering computed by that algorithm is divided by the average size of these coverings. Thus, it can be noted that LamSCP offers a stability 4 to 8 times superior to ASA concerning the size of the coverings. As for the number of sentences, the relative standard deviation similarly decreases from 0.97% to 0.28% when k increases from 1 to 5 for ASA solutions, and from 0.42% to 0.15% for LamSCP ones.

One can note that the increase of the minimal number k of instances of each unit to cover leads to a selection, by LamSCP and ASA, of longer sentences on average. The average length of the sentences picked for a 1-covering was quite low. As the constraints increase along with k, it only seems natural that the algorithms tend to select longer sentences, as shorter sentences no longer contain enough occurrences of 2-phonemes. Moreover, as described in Section 2, when the minimal number b_i of a unit u_i demanded in the covering exceeds the number of instances of that unit in the initial corpus, all sentences containing instances of u_i in the initial corpus are selected, and b_i is set to $(A\mathbf{1}_{\mathbb{R}^n})_i$. Thus, as k increases, the algorithm tends to select more and more sentences, and their length tends towards the average value over the whole corpus, which is 28.51 phones for *Gutenberg*.

As for computation time, although it increases as k grows, because of the increasing number of constraints to update, the ratio between the computation time of LamSCP and ASA tends to diminish, as shown in Table 7. This tendency may find an explanation in the fact that the search space diminishes as k increases, which causes a lesser number of selected sentences to be questioned during the 3-phase iteration of LamSCP. Also, we notice that the average computation time of the two algorithms is greater in Experiment 1, owing to a greater number of sentences in corpus *Le-Monde* and a higher density of matrix A. Moreover, the ratio between the computation times of

Table 6
Relative standard deviation of solution cost for both algorithms of a k-covering of 2-phonemes from *Gutenberg*.

k	1	2	3	4	5
LamSCP	0.07%	0.05%	0.05%	0.04%	0.02%
ASA	0.57%	0.37%	0.29%	0.18%	0.17%

Table 7
Computation time ratio between LamSCP and ASA for a k-covering of 2-phonemes from *Gutenberg*.

k	1	2	3	4	5
Time LamSCP / Time ASA	333 ($\pm$ 7)	280 ($\pm$ 5)	292 ($\pm$ 6)	218 ($\pm$ 9)	194 ($\pm$ 8)

LamSCP and ASA decreases between Experiments 1 and 2, going from 390 ($\pm$9) to 333 ($\pm$7). Again, this can be explained by the diminishing of the search space.

For $k = 1$, the advantage offered by LamSCP on the covering costs compared with ASA is slightly higher than that observed in Experiment 1: 9.73% ($\pm$0.13) in this case, versus 9.00% ($\pm$0.20) in the previous experiment. This seems to contradict the idea that the performance of LamSCP improves as the search area becomes wider. However, the distributions of the units to cover in *Gutenberg* and *Le-Monde* are different, and the variation on the length of the sentences in *Le-Monde* is very high, which may account for this slight difference in terms of gain. Note that the size of the calculated coverings and the lower value $L(\tilde{\Lambda})$ are closer in the experiment carried out on *Gutenberg*. It is difficult, however, to perform further comparisons with Experiment 1 regarding the "distance" between the costs of the solutions computed by these algorithms, and the optimal covering cost, given that the quality of the lower bound cannot be evaluated. The gain in stability offered by LamSCP, both for the costs of the solutions or the number of sentences, is more important than that noticed during the previous experiment. We think that the increase is due to a more restricted search space, and less variability of the length of the sentences in corpus *Gutenberg*, which may be observed in Table 1.

6.3 1-Covering of 3-Phonemes in English

Table 8 sums up the main results of Experiment 3, where 35 instances of 1-covering of 3-phonemes from *Gutenberg* were processed. According to the $L(\tilde{\Lambda})$ values, covering all 3-phonemes requires a solution size greater than or equal to 226,635 phones. On average, the solution measures 227,360 $\pm$ 12 phones using LamSCP, and 236,828 $\pm$ 94

Table 8
Experiment 3: Statistics based on 35 experiments of a 1-covering of 3-phonemes from *Gutenberg*.

	ASA	LamSCP	$L(\tilde{\Lambda})$
Covering size (phones)	236,828 $\pm$ 94	227,360 $\pm$ 12	226,559 $\pm$ 8
Reduction rate relative to ASA (%)		**–3.99 $\pm$ 0.04**	**–4.33 $\pm$ 0.04**
Covering size [min; max]	[236,075; 237,615]	[227,317; 227,425]	[226,518; 226,635]
Covering size Std. Dev.	301.75	33.12	23.67
Sentence number	8,005.20 $\pm$ 5.02	7,606.77 $\pm$ 2.14	–
Sentence length	29.88 $\pm$ 0.00	29.58 $\pm$ 0.01	–
Time CPU (seconds)	2,834 $\pm$ 20	371,501 $\pm$ 10	–

phones using ASA. The optimal covering is at most 0.35% ($\pm$0.00) shorter than solutions derived by LamSCP and 4.33% ($\pm$0.04) shorter than the ones derived by ASA. We can then observe that both algorithms manage to compute solutions with close sizes when scaling up the required attribute set. The solutions are very stable, even more than in Experiment 2: The relative variation of their size is 0.12% for ASA and 0.01% for LamSCP; the relative variation of their sentence number is 0.17% for ASA and 0.07% for LamSCP. This increase of stability is due to a smaller search space and the increase of the number of rare units required, which also compels the algorithms to select a higher number of inevitable sentences for all the instances of the SCP. Furthermore, the decrease of the ratio between the computation time of LamSCP and ASA from 332 for the 1-covering of 2-phonemes to 130 for the one of 3-phonemes on *Gutenberg* may confirm this idea, which has also been put forward in Experiment 2.

Concerning the length of the selected sentences by both algorithms, it is greater than the one for the 5-covering of 2-phonemes, observed in Experiment 2, and slightly deviates from the average sentence length for the whole corpus. Consequently, it turns out that covering longer and generally rarer units involves a selection of longer sentences. This is confirmed by the fact that the sentences of *Gutenberg* covering units with a single occurrence in *Gutenberg* represent more than half the size of the solutions and are composed of 33 phones on average.

6.4 1-Covering of 3-Phonemes in French

In this section, we analyze the results of Experiment 4, the 30 instances of the 1-covering of 3-phonemes from *Le-Monde* carried out by ASA and LamSCP. The results are given in Table 9. First, although the main features of *Le-Monde* and *Gutenberg* are different, notice that the closeness between the size of coverings calculated by both algorithms and the lower bound $L(\tilde{\Lambda})$ is comparable to the one observed in Experiment 3. Indeed, the optimal covering size is at most 0.48% ($\pm$0.03) and 4.35% ($\pm$0.05) shorter than the solution size derived by LamSCP and ASA, respectively. Similarly, the size of solutions and the number of selected sentences are as stable as those observed in the previous experiment: The solution length varies from 0.01% for LamSCP to 0.10% for ASA and the number of sentences fluctuates about 0.16% for ASA and 0.10% for LamSCP.

As for the comparison with the results of Experiment 1 (1-covering of 2-phonemes from *Le-Monde*), the main trends are similar to the ones observed for the transition from the 1-covering of 2-phonemes to the 1-covering of 3-phonemes from *Gutenberg*.

Table 9
Experiment 4: Statistics based on 30 experiments of a 1-covering of 3-phonemes from *Le-Monde*.

	ASA	LamSCP	$L(\tilde{\Lambda})$
Covering size (phones)	620,434 $\pm$ 222	596,323 $\pm$ 27	593,417 $\pm$ 127
Reduction rate relative to ASA (%)		**–3.89 $\pm$ 0.03**	**–4.35 $\pm$ 0.04**
Covering size [min; max]	[619,319; 622,201]	[596,190; 596,486]	[592,640; 594,250]
Covering size Std. Dev.	633.03	88.59	391.82
Sentence number	6,969.10 $\pm$ 4.63	6,436.76 $\pm$ 2.56	–
Sentence length	89.02 $\pm$ 0.03	92.64 $\pm$ 0.03	–
Time CPU (seconds)	7,845 $\pm$ 78	660,928 $\pm$ 24,355	–

However, in Experiment 4, the average selected sentence length has markedly increased, approaching the mean value on the whole corpus: 89.02 ($\pm$0.03) for ASA and 92.64 ($\pm$0.03) for LamSCP, whereas in Experiment 1 these values are, respectively, 25.48 ($\pm$0.10) and 28.97 ($\pm$0.12). We have already observed in Experiment 3 that covering longer units increases the length of selected sentences but this high amplitude seems to be inherent to the design of corpus *Le-Monde*. Furthermore, notice that the 1-coverings of 2-phonemes from *Le-Monde* are almost half as small as the ones from *Gutenberg*, whereas the 1-coverings of 3-phonemes from *Le-Monde* are between twice and three times longer than the ones from *Gutenberg*. This is due to the fact that the 3-phonemes with a single instance in *Le-Monde* are very scattered in long sentences (their length mean is about 134 phones), and these indispensable sentences represent nearly half the size of the solutions. The other sentences of the solutions are around 70 phones long. Lastly, the ratio between the computation time of both algorithms is about 84, which is smaller than the ratios previously observed, but this SCP is the most time consuming: 2 hr 10 min for ASA and more than 7 days for LamSCP.

6.5 k-Covering of 1-POS and 2-POS in French

Table 10 sums up the main results of Experiment 5, dealing with the 1- and 5-coverings of 1-POS and 2-POS. For all these SCP, LamSCP produces smaller coverings, composed of longer sentences, than the coverings obtained with ASA. When the search space diminishes, the relative "distance" between the size of solutions provided by both algorithms decreases, as well as between the lower bound $L(\tilde{\Lambda})$ and the size of solutions obtained by ASA. These trends were also observed in the earlier experiments. In particular, as for the 1-covering of 1-POS, not only does LamSCP provide 10.06% ($\pm$ 0.00) shorter solutions than ASA, but its solutions are optimal for all 50 instances of this SCP. Indeed, the lower bound value varies from 482.51 to 482.87 occurrences of 1-POS while all the solutions given by LamSCP are made of exactly 483 occurrences of POS. For the other k-coverings of n-POS, the optimal solution is at worst 0.39% ($\pm$0.02) shorter than the covering given by LamSCP for $(k, n) = (5, 1)$, 0.11% ($\pm$0.00) for $(k, n) = (1, 2)$, and 0.22% ($\pm$0.00) for $(k, n) = (5, 2)$. The solutions obtained by ASA or by LamSCP are very stable. For example, the relative standard deviation of number of POS in a covering solution varies from 0.00% to 1.23% for ASA, and from 0.00% to 0.04% for LamSCP.

As previously observed for both algorithms, their computation times grow when the number of required covering features increases. However, the ratio between the computation time of LamSCP and ASA does not behave as in Experiment 2 (see Table 7): For the k-covering of 1-POS, this ratio increases from 290 ($\pm$11) to 657 ($\pm$ 43) when k goes from 1 to 5, and for the k-covering of 2-POS, it increases from 75 ($\pm$5) to 108 ($\pm$6).

7. Evaluation on a Text-to-Speech Synthesis System

In the previous sections, different algorithms dealing with corpus reduction were introduced and studied. The proposed experiments mainly evaluate the effects of these algorithms in terms of corpus reduction but not according to a practical task. This section proposes an experiment to assess the impact of the corpus reduction on a unit selection speech synthesis system.

As explained in Section 1, a corpus reduction for a TTS system is a trade-off between minimizing the recording and post-processing time to build the speech corpus and

Table 10
Statistics of 50 instances of a k-covering of n-POS from *Le-Monde*.

Experiment 5: k-covering of n-POS, corpus *Le-Monde*			
$n = 1$ *and* $k = 1$			
	ASA	LamSCP	$L(\tilde{\Lambda})$
Covering size (POS)	537.70 ± 1.78	483.00 ± 0.00	482.68 ± 0.02
Reduction rate relative to ASA (%)		**–10.06 ± 0.00**	**–10.17 ± 0.00**
Covering size [min; max]	[524; 552]	[483; 483]	[482.51; 482.87]
Covering size Std. Dev.	6.64	0	0.08
Sentence number	72.07 ± 0.51	60.65 ± 0.31	–
Sentence length	7.46 ± 0.04	7.97 ± 0.03	–
Time CPU (seconds)	2 ± 0	735 ± 25	–
$n = 1$ *and* $k = 5$			
Covering size (POS)	2,659 ± 4	2,502 ± 0	2,492 ± 0
Reduction rate relative to ASA (%)		**–5.90 ± 0.11**	**–6.27 ± 0.09**
Covering size [min; max]	[2,628; 2,682]	[2,501; 2,505]	[2,482; 2,493]
Covering size Std. Dev.	12.42	1.11	1.85
Sentence number	285.00 ± 0.82	238.91 ± 0.65	–
Sentence length	9.33 ± 0.02	10.47 ± 0.02	–
Time CPU (seconds)	5 ± 0	2,712 ± 162	–
$n = 2$ *and* $k = 1$			
Covering size (POS)	77,022 ± 18	75,281 ± 1	75,192 ± 2
Reduction rate relative to ASA (%)		**–2.25 ± 0.02**	**–2.37 ± 0.02**
Covering size [min; max]	[76,913; 77,187]	[75,273; 75,297]	[75,176; 75,220]
Covering size Std. Dev.	63.33	3.44	8.26
Sentence number	3,288.22 ± 1.48	3,120.89 ± 0.91	–
Sentence length	23.42 ± 0.00	24.12 ± 0.00	–
Time CPU (seconds)	175 ± 0	13,095 ± 881	–
$n = 2$ *and* $k = 5$			
Covering size (POS)	281,532 ± 17	278,114 ± 5	277,497 ± 4
Reduction rate relative to ASA (%)		**–1.21 ± 0.00**	**–1.43 ± 0.00**
Covering size [min; max]	[281,362; 281,652]	[278,084; 278,152]	[277,459; 277,524]
Covering size Std. Dev.	72.09	15.27	14.33
Sentence number	10,379.67 ± 1.67	10,034.96 ± 2.66	–
Sentence length	27.12 ± 0.00	27.71 ± 0.00	–
Time CPU (seconds)	2,863 ± 113	309,095 ± 901	–

keeping the highest phonological richness of the corpus to ensure the quality of the synthetic speech. The goal of this experiment is to measure this trade-off by evaluating the quality of the same TTS system fed with different speech corpora uttered by the same speaker. Note that the intrinsic quality of this system is not the purpose here.

Firstly, a brief presentation of a state-of-the-art unit selection–based TTS system is proposed in Section 7.1. The linguistic parameters used by the TTS system are detailed because they are linked to the required features in the reduction stage. In Section 7.2, corpora used in the experiment are introduced. The attributes to cover and the

methodology of evaluation are described in Section 7.3; the results are given and discussed in Section 7.4.

7.1 Text-to-Speech System

For this experiment, a state-of-the-art unit selection–based TTS system is used to produce an acoustic signal from an input text. A linguistic front end processes the text to extract features taken into account by the algorithm that selects segments in a speech corpus (see Boëffard and d'Alessandro 2012).

The input text is converted into a sequence of phonemes using a French phonetizer proposed by Béchet (2001). Non-speech sound labels can be added to this sequence (silences, breaths, para-verbal events, etc.). A vector of features is defined as follows:

1. The phone or non-speech sound label
2. Is the described segment a non-speech sound?
3. Is the phone in the onset of the syllable?
4. Is the phone in the coda of the syllable?
5. Is the phone in the last syllable of its syntagm?
6. Is the current syllable at the end of a word?
7. Is the current syllable at the beginning of a word?

Extraction of features is done using the ROOTS toolkit described in Boëffard et al. (2012). The unit selection process aims to associate a signal segment from the speech corpus to each vector of features computed from the input text. This is performed in two steps. In the first step, for each unit, a set of candidates that match the same features are extracted from the speech corpus. In the second step, given all candidates, the best path is searched using an optimization algorithm so as to produce the sequence of speech units. The algorithm tries to minimize three sub-costs, commonly used in unit selection based TTS systems, which are spectral discrepancies based on MFCC distance, amplitude, and f0 distances.

7.2 Corpora

Two corpora are used in this experiment. The first one, *Learning corpus*, is an annotated acoustic corpus used to provide speech data for the TTS engine. It is an expressive corpus in French, spoken by a male speaker reading *Albertine disparue*, an excerpt from *À la recherche du temps perdu* by Marcel Proust. The corpus is composed of 3,138 sentences automatically annotated using a process described in Boëffard et al. (2012). The overall length of the speech corpus is 9 hr 57 min. When creating a voice for a unit selection–based TTS system, long sentences are generally removed or split into syntagm groups in order to help the speaker.

A second corpus, named *Test corpus*, is a text corpus that is synthesized and used in the listening experiment. It is composed of 30 short sentences randomly extracted from a phonetically balanced corpus in French, proposed by Combescure (1981). The use of a corpus with a different linguistic style minimizes the bias introduced by the learning corpus.

Statistics are given in Table 11.

Table 11
Characteristics of the two corpora.

	Learning corpus	*Test corpus*
Number of syntagms	19,587	30
Number of labels	36	36
Corpus size (labels)	392,865	813
Number of 2-phoneme	1,033	317
Number of diphones	373,278	787
Syntagm length mean (phones) & Std. Dev.	20.1 (11.9)	27.1 (5.6)

7.3 Methodology

For this experiment, two reduced corpora are evaluated. They are built by reducing the full learning corpus using the two different algorithms presented in the previous sections: ASA and LamSCP.

As described in Section 7.1, the unit selection process of the speech synthesis system is based on a set of phonological attributes. It seems natural to try to cover features that reflect the variability of these attributes. For this experiment, algorithms must cover all the units at least once, where a unit is described by the following:

- Its label, that is, one of the 35 phonemes or a non-speech sound label
- The structure of the syllable that contains the phoneme, if it is a vowel
- The position of the associated syllable in the word (start, middle, or end)
- A Boolean indicated if the associated syllable is at the end of a syntagm

The feature extraction is performed by the same set of tools used by the speech synthesis engine. Given this set of features, the learning corpus contains 1,497 classes of units. The cost function to minimize by the reduction algorithms is the total length of the set of the selected syntagms, in phones.

Two speech synthesis systems are defined, extracting the speech units to concatenate from the coverings provided by LamSCP and ASA. Two other systems are added as baselines. First, a system named *Full*, built with the whole learning corpus, is used as an upper bound. Second, a system named *Random* uses a random reduction of the *Learning corpus* as a pool corpus of speech units. This reduction is done by randomly selecting sentences from the whole learning corpus until the size of the covering obtained by LamSCP is reached. *Random* is used as a lower bound.

Whereas the optimization efficiency is measured by statistics on the reduced corpora, the quality of the synthesized speech signals is evaluated by a listening test. The protocol is based on a MUSHRA test, presented in ITU-R (2003), where for every sentence of *Test corpus*, the signals synthesized by the four systems are presented to each tester in a random order. If a system is not able to produce a signal for a requested sentence (because of a missing 2-phone in the pool corpus), an empty signal is presented. Ten native French testers (four naives and six experts) are asked to evaluate the overall quality of the stimuli and to give a mark from 0 to 100 (by steps of 5 points).

Table 12
Statistics about the reduced corpora computed by ASA and LamSCP from *Learning* corpus. The last column concerns the best lower bound found by LamSCP.

Speech synthesis experiment: corpus reduction			
	ASA	LamSCP	$L(\tilde{\Lambda})$
Covering size (phones)	36,538	35,681	35,655
Reduction rate relative to the full corpus (%)	**–90.70**	**–90.92**	**–90.92**
Reduction rate relative to ASA (%)		**–2.35**	**–2.42**
Number of syntagms	2,191	2,119	–

7.4 Results and Discussion

The *Learning corpus* composed of 19,587 syntagms is first reduced using ASA and LamSCP. Statistics of the resulting solutions are summarized in Table 12. The covering of the 1,497 constraints divides the input corpus size by almost 10 and reduces the 10 hours of speech to around 1 hr 20 min. As for the previous experiments, even though different kinds of features are mixed (phonemes, syllable structures, position in a word or a syntagm) the ASA algorithm produces a solution close to the optimal one. However, LamSCP is again slightly better in terms of covering size.

For the measure of the acoustic impact of corpus reduction, the listening test results are presented in Table 13 for the average marks and in Table 14 for the average ranks. Note that without a natural speech reference during the test, the marks should not be seen as an absolute score. Even if the LamSCP corpus is slightly smaller than the one from ASA, the acoustic quality of both systems is comparable according to the testers (with a slight advantage for the LamSCP corpus). In comparison with the baseline, which uses the whole learning corpus, the acoustic degradation is significant. This illustrates well the trade-off between corpus size and speech quality: For a 90% corpus size reduction, the acoustic quality drops by 10 points. Further research should be focused on the set of attributes to cover and their number of occurrences in order to improve this compromise. As expected, the baseline built from random sentences is preferred significantly less than the other systems because of the lack of relatively rare acoustic units.

Table 13
MUSHRA average marks of 30 sentences from the *Test corpus* with 10 listeners per sentence (the higher the better).

System	MUSHRA marks	95% confidence interval
Full	55.52	2.37
LamSCP	45.72	2.22
ASA	44.45	2.20
Random	33.2	3.18

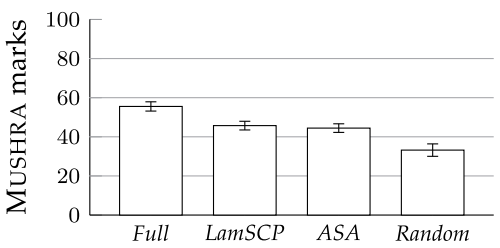
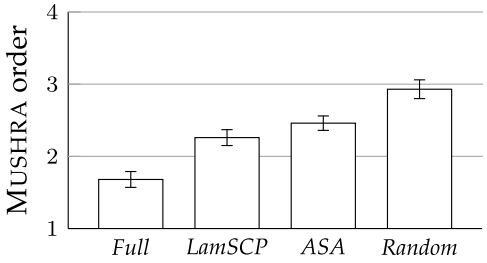


Table 14
MUSHRA average ranks for 30 sentences from the *Test corpus* with 10 listeners per sentence (the lower the better).

System	MUSHRA order	95% confidence interval
Full	1.68	0.11
LamSCP	2.26	0.11
ASA	2.46	0.10
Random	2.93	0.13



8. Conclusion

This article discussed the building of linguistic-rich corpora under an objective of parsimony. This task, a generalization of SCP, turns out to be an NP-hard problem that cannot be polynomially approximated. We studied the behavior of several algorithms in the particular domain of NLP, where the considered events follow a heavy-tailed type distribution.

The proposed algorithms have been compared through three kinds of experiments: The first one is the coverings of 2- and 3-phonemes from two text corpora, one in French, the other in English; the second one consists of the coverings of part-of-speech labels from a corpus in French; the third one evaluates the impact of both algorithms on the acoustic quality of a corpus-based TTS system. The first algorithm, ASA, is composed of an agglomerative greedy strategy followed by a spitting greedy stage. The second one, LamSCP, is based on Lagrangian relaxation principles combined with greedy strategies. LamSCP is our adaptation of an algorithm proposed in Caprara, Fischetti, and Toth (1999) to the multi-representation constraints. The comparison of SCP solutions is mainly about their size, their maximal distance with the optimal covering, and their robustness in case of perturbation of the initial corpus ordering.

Although ASA is much faster than LamSCP, it does not permit us to single-handedly assess the quality of its solution in terms of size. The main assets of LamSCP are the calculation of a lower bound to the optimal covering size and shorter solutions than the ones obtained by ASA. Indeed, in our experiments of phonological coverings, the optimal solution is at most 1.24% (10.13%, respectively) smaller than the solutions derived by LamSCP (ASA, respectively). As for the coverings of 1-POS, LamSCP provides the optimal solution in a case of a mono-representation constraint, whereas the ASA solution is 10.17% greater than the optimal one. These relative gaps between the lower bounds and solution sizes of both algorithms generally decrease when the size of the search space decreases. Thanks to the lower bound derived by LamSCP, we empirically show that it is possible to get almost optimal solutions in a linguistic framework following Zipf's law distribution, despite the theoretic complexity of the multi-represented SCP. Concerning the last experiment in the TTS framework, even if LamSCP provides a smaller corpus, the subjective test shows no significant difference between the TTS systems based on LamSCP and ASA corpora. Therefore, we think that ASA remains the most adequate strategy, in terms of performance, ease of development, and computation time to solve SCP in the NLP field. However, it would be interesting

to test a parallelized version of the heuristic phase that calls an important number of greedy sub-procedures.

Our future prospects for this work are in automatic language processing and speech synthesis. First, in the framework of the Phorevox project supported by the French National Research Agency, we are considering the automatic design of exercise contents for language learning by the selection of texts covering some phonological or linguistic difficulties. Secondly, this work is a preliminary step to building a phonetically rich script before its recording in order to produce a high quality speech synthesis. The covering choices, such as the attributes to cover, the number of required occurrences, or the "sentence" length (utterances, syntagms, etc.) need to be validated. Moreover, in this article, we have observed the great impact of the distribution of rare units in the corpus to reduce, and we believe it will be interesting to adapt the "sentence" granularity according to this distribution.

References

Alon, Noga, Dana Moshkovitz, and Shmuel Safra. 2006. Algorithmic construction of sets for k-restrictions. *ACM Transactions on Algorithms (TALG)*, 2(2):153–177.

Barbot, Nelly, Olivier Boëffard, and Arnaud Delhay. 2012. Comparing performance of different set-covering strategies for linguistic content optimization in speech corpora. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 969–974, Istanbul.

Béchet, Frédéric. 2001. Liaphon: un systeme complet de phonétisation de textes. *Traitement automatique des langues*, 42(1):47–67.

Boëffard, Olivier, Laure Charonnat, Sébastien Le Maguer, Damien Lolive, and Gaëlle Vidal. 2012. Towards fully automatic annotation of audiobooks for TTS. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 975–980, Istanbul.

Boëffard, Olivier and Christophe d'Alessandro, 2012. *Speech Synthesis*. Wiley.

Bunnell, H. Timothy. 2010. Crafting small databases for unit selection TTS: Effects on intelligibility. In *Proceedings of the ISCA Tutorial and Research Workshop on Speech Synthesis (SSW7)*, pages 40–44, Kyoto.

Cadic, Didier, Cédric Boidin, and Christophe d'Alessandro. 2010. Towards optimal TTS corpora. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 99–104, Malta.

Candito, Marie, Enrique Henestroza Anguiano, and Djamé Seddah. 2011. A word clustering approach to domain adaptation: Effective parsing of biomedical texts. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 37–42, Dublin.

Caprara, Alberto, Matteo Fischetti, and Paolo Toth. 1999. A heuristic method for the set covering problem. *Operations Research*, 47(5):730–743.

Caprara, Alberto, Paolo Toth, and Matteo Fischetti. 2000. Algorithms for the set covering problem. *Annals of Operations Research*, 98(1-4):353–371.

Ceria, Sebastián, Paolo Nobili, and Antonio Sassano. 1998. A Lagrangian-based heuristic for large-scale set covering problems. *Mathematical Programming*, 81(2):215–228.

Chevelu, Jonathan, Nelly Barbot, Olivier Boëffard, and Arnaud Delhay. 2007. Lagrangian relaxation for optimal corpus design. In *Proceedings of the ISCA Tutorial and Research Workshop on Speech Synthesis (SSW6)*, pages 211–216, Bonn.

Chevelu, Jonathan, Nelly Barbot, Olivier Boëffard, and Arnaud Delhay. 2008. Comparing set-covering strategies for optimal corpus design. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 2951–2956, Marrakech.

Combescure, Pierre. 1981. 20 listes de 10 phrases phonétiquement équilibrées. *Revue d'Acoustique*, 56:34–38.

Fisher, Marshall L. 1981. The Lagrangian relaxation method for solving integer programming problems. *Management Science*, 27(1):1–18.

François, Hélène and Olivier Boëffard. 2001. Design of an optimal continuous speech database for text-to-speech synthesis considered as a set covering problem. In *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*, pages 829–832, Aalborg.

François, Hélène and Olivier Boëffard. 2002. The greedy algorithm and its application

to the construction of a continuous speech database. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 1420–1426, Las Palmas, Canary Islands.

Gauvain, Jean-Luc, Lori Lamel, and Maxine Eskénazi. 1990. Design considerations and text selection for Bref, a large French readspeech corpus. In *Proceedings of the International Conference of Spoken Language Processing (ICSLP)*, pages 1097–1100, Kobe.

Gotab, Pierre, Frédéric Béchet, and Géraldine Damnati. 2009. Active learning for rule-based and corpus-based spoken language understanding models. In *Proceedings of the IEEE workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 444–449, Merano.

Hart, Michael. 2003. Project gutenberg. `http://www.gutenberg.org/` (Last consulted April 2015).

ITU-R. 2003. ITU-R recommendation bs.1534: Method for the subjective assessment of intermediate quality levels of coding systems. Radio communication Bureau, Geneva.

Karp, Richard M. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, the IBM Research Symposia Series. Springer, pages 85–103.

Kawai, Hisashi, Seiichi Yamamoto, Norio Higuchi, and Tohru Shimizu. 2000. A design method of speech corpus for text-to-speech synthesis taking account of prosody. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, pages 420–425, Beijing.

Kominek, John and Alan W. Black. 2003. The CMU Arctic speech databases for speech synthesis research. Technical Report CMU-LTI-03-177, Carnegie Mellon University Language Technologies Institute. Pittsburg, PA.

Krstulović, Sacha, Frédéric Bimbot, Olivier Boëffard, Delphine Charlet, Dominique Fohr, and Odile Mella. 2006. Optimizing the coverage of a speech database through a selection of representative speaker recordings. *Speech Communication*, 48(10):1319–1348.

Krul, Aleksandra, Géraldine Damnati, François Yvon, Cédric Boidin, and Thierry Moudenc. 2007. Adaptive database reduction for domain specific speech synthesis. In *Proceedings of the ISCA Tutorial and Research Workshop on Speech Synthesis (SSW6)*, pages 217–222, Bonn.

Krul, Aleksandra, Géraldine Damnati, François Yvon, and Thierry Moudenc. 2006. Corpus design based on the Kullback-Leibler divergence for Text-To-Speech synthesis application. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, pages 2030–2033, Pittsburgh, PA.

Neubig, Graham and Shinsuke Mori. 2010. Word-based partial annotation for efficient corpus construction. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 2723–2727, Malta.

Raz, Ran and Shmuel Safra. 1997. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, STOC '97, pages 475–484, El Paso, TX.

Rojc, Matej and Zdravko Kačič. 2000. Design of optimal Slovenian speech corpus for use in the concatenative speech synthesis system. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 321–326, Athens.

Schein, Andrew I., Ted S. Sandler, and Lyle H. Ungar. 2004. Bayesian example selection using BaBiES. Technical Report MS-CIS-04-08, Department of Computer and Information Science, University of Pennsylvania.

Settles, Burr. 2010. Active learning literature survey. Technical Report 1648, Department of Computer Sciences, University of Wisconsin, Madison.

Synapse. 2011. Documentation technique: Composant d'étiquetage et lemmatisation. `http://www.synapse-fr.com/`.

Tian, Jilei and Jani Nurminen. 2009. Optimization of text database using hierachical clustering. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 4269–4272, Taipei.

Tian, Jilei, Jani Nurminen, and Imre Kiss. 2005. Optimal subset selection from text databases. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 305–308, Philadelphia, PA.

Tomanek, Katrin and Fredrik Olsson. 2009. A Web survey on the use of active learning to support annotation of text data. In *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, pages 45–48, Boulder, CO.

Van Santen, Jan P. H. and Adam L. Buchsbaum. 1997. Methods for optimal text selection. In *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*, pages 553–556, Rhodes.

Zhang, Jin-Song and Satoshi Nakamura. 2008. An improved greedy search algorithm for the development of a phonetically rich speech corpus. *IEICE Transactions on Information and Systems*, E91-D(3):615–630.