Regularization Paths for Sparse Nonnegative Least Squares Problems with Applications to Life Cycle Assessment Tree Discovery

Jingu Kim, *Nokia Inc.*, *jingu.kim@nokia.com*
Naren Ramakrishnan, *Virginia Tech*, *naren@cs.vt.edu*
Manish Marwah, *HP Labs*, *manish.marwah@hp.com*
Amip Shah, *HP Labs*, *amip.shah@hp.com*
Haesun Park, *Georgia Tech*, *hpark@cc.gatech.edu*

***Abstract*—The nonnegative least squares problems are useful in applications where the physical nature of problem domain permits only additive linear combinations. We discuss the l_1-regularized nonnegative least squares (L1-NLS) problem, where l_1-regularization is used to induce sparsity. Although l_1-regularization has been successfully used in least squares regression, when combined with nonnegativity constraints, developments of algorithms and their understandings have been limited. We propose an algorithm that generates the entire regularization paths of the L1-NLS problem. We prove the correctness of the proposed algorithm and illustrate a novel application in environmental sustainability. The application relates to life cycle assessment (LCA), a technique used to estimate environmental impact during the entire lifetime of a product. We address an inverse problem in LCA. Given environmental impact factors of a target product and of a large library of constituents, the goal is to reverse engineer an inventory tree for the product. Using real-world data sets, we demonstrate how our L1-NLS approach controls the size of discovered trees, and how the full regularization paths effectively illustrate the spectrum of discovered trees with varying sparsity and compositions.**

I. Introduction

Due to increasing public consciousness about sustainability, eco-friendly products and services have recently been emphasized. In a 2010 article in the *New York Times* [1], Goleman and Norris investigated whether an e-reader or a printed book is more environmentally friendly. After considering the life cycle of both products (including materials, manufacturing, transportation, even the light bulb energy used for reading, and finally discard), they concluded that the environmental impact of one e-reader is somewhere between 50-100 paper books. This implies that e-readers are not necessarily more environmentally friendly than printed books. This type of analysis is known as life cycle assessment (LCA) because it requires analysis of environmental impacts of each component of a product from "cradle to grave."

LCA is typically conducted by companies in-house using extensive information about components (parts or processes) involved in the life cycle of a product. The idea is to track a fixed set of environmental impact factors as a product goes through its life cycle stages of construction, use, and discard. For instance, impact factors could denote the release of CO_2 into the atmosphere, wildlife species affected, industrial waste dissipated into rivers, and so on. Companies depend on the bill-of-materials of components to manually compute impact factors associated with a product by propagating the impact factor measurements of individual components. By conducting such elaborate and laborious computation, a company will be able to publish impact factors associated with their products.

Our focus in this paper is on the inverse problem: Given the published impact factors of a product, can we reverse-engineer its composition? This is the LCA tree discovery problem, originally introduced in [2]. More precisely, given an impact factor database [3], [4], [5] and a specific product whose impact factors are known (published by the manufacturer), the goal is to reconstruct the composition of the product in terms of other components in the database. Given environmental impacts disclosed by manufacturers, the LCA tree discovery estimates the underlying composition of their product and consequently determines (i) whether the disclosures are valid in light of prevailing databases of impact factors and (ii) the nature of components that have the biggest environmental impacts.

The contributions of this paper are three-fold. First, we formulate the problem of estimating LCA trees as a l_1-regularized nonnegative least squares (L1-NLS) problem. Since nonnegativity constraints naturally arise when dealing with the composition of real world materials, Sundaravaradan et al. [2] previously employed a nonnegative least squares (NLS) approach for LCA tree discovery. However, solutions from the NLS method tends to include too many nonzero elements, and a heuristic to reduce the number of nonzero elements had to be used [2]. We take a principled approach based on l_1-regularization, which has been well-known to promote a sparse solution in linear regression and other estimation problems [6].

Second, we present an algorithm that computes the *full regularization paths* of the L1-NLS problem. For l_1-regularized linear regression, also known as Lasso [6], homotopy methods for generating the full-regularization paths have been developed [7], [8]. Built upon these methods, our algorithm additionally handles nonnegativity constraints.

Morup et al. proposed a nonnegative least angle regression (NLARS) algorithm for the L1-NLS problem [9]; however, they provided only an incomplete sketch without proof of correctness. In this paper, we propose an algorithm that generates the full-regularization paths of the L1-NLS problem and also prove its correctness.

Finally, with our proposed algorithm, we demonstrate how LCA trees can be discovered. We illustrate successful examples of LCA tree reconstruction using an impact factor database of real world products. We show how the inference of full regularization paths provides insight into tree reconstruction performance, alleviating difficulties in the identification of potential children sets and estimation of their contributions.

Notations. A lowercase letter, such as x, denotes a scalar. A boldface lowercase and an uppercase letter, such as $\mathbf{x}$ and $\mathbf{X}$, denote a vector and a matrix, respectively. Elements of a sequence are denoted by superscripts within parentheses; e.g., $\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(k)}$. For matrix $\mathbf{X}$, $\mathbf{x}_i$ denotes its i^{th} column, x_{ij} denotes its $(i,j)^{th}$ element, and $\mathbf{X}^T$ denotes its transpose. For vector $\mathbf{x}$, x_i denotes its i^{th} element. For an index set $\mathcal{P}$, $\mathbf{x}_\mathcal{P}$ denotes a subvector of $\mathbf{x}$ containing only corresponding elements, and $\mathbf{X}_\mathcal{P}$ denotes a submatrix of $\mathbf{X}$ containing only columns with corresponding indices. Inequality $\mathbf{x} \geq 0$ indicates that the elements of $\mathbf{x}$ are nonnegative.

II. Background

In LCA tree discovery, we are given a target product of interest, for which we desire to discover parts and processes involved in its life cycle. The target product is considered as a parent node, and its children are to be discovered. Input provided to us is an *impact factor database* (or library) for a large number of products, parts, and processes. Typically one of these is for the parent node, and the children are intended to be picked from the rest. A fragment of an impact factor database is shown in Table I. This table shows only three factors, but the total number of impact factors in commercial databases [3], [4], [5] could run into a few hundred. Similarly, the number of components (rows) in Table I could run into hundreds or even thousands. Figure 1 shows an example LCA tree for which children are known. The LCA tree could involve multiple layers, as shown in Figure 1. In this paper, we focus on discovering the immediate children of a given product. The full LCA tree can be, e.g., estimated by recursively applying algorithms discussed here.

Each of the impact factors follows the tree semantics: i.e., the impact factor of a parent node is a linear combination of the impact factors of its children. The coefficients of the linear combination denote the amount of each component involved. More formally, assume that the impact factor database provides values for a total of p impact factors. Let $\mathcal{F} = \{1, \cdots, n\}$ denote the indices of all library components, and let $\mathbf{a}_1, \cdots, \mathbf{a}_n$ ($\mathbf{a}_i \in \mathbb{R}^p$) represent their

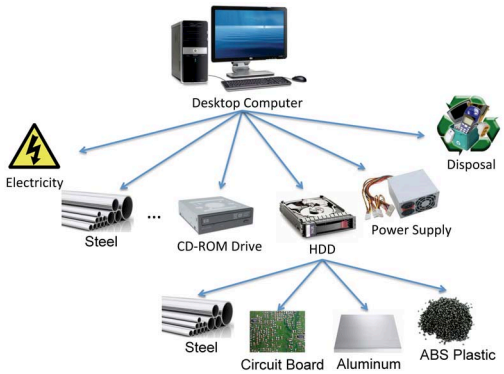


Figure 1. An example of a desktop LCA tree.

impact factor values. For a target product, let its impact factors be $\mathbf{b} \in \mathbb{R}^p$. We impose a linear reconstruction model

$$\mathbf{b} = \sum_{i=1}^{n} c_i \mathbf{a}_i, \ c_i \geq 0 \ \text{ for } \forall i. \tag{1}$$

That is, the impact factors of a parent node is equivalent to a nonnegative linear combination of the impact factors of children. Since physical components and processes can be combined only additively, it is natural to allow only nonnegative coefficients. Here, $c_i > 0$ represents that the i^{th} component is a children of the target product, and $\{i | c_i > 0\}$ represents the true children set.

A problem for finding unknown coefficients can be formulated as a least squares problem. Letting $\mathbf{A} = [\mathbf{a}_1, \cdots, \mathbf{a}_n] \in \mathbb{R}^{p \times n}$, the nonnegativity-constrained least squares (NLS) problem can be written as

$$\mathbf{x}^* \leftarrow \underset{\mathbf{x} \in \mathbb{R}^n}{\arg\min} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 \ \text{ s.t. } \mathbf{x} \geq 0, \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^n$. When $p < n$ (i.e., the size of library set is larger than the number of impact factors), matrix A is rank-deficient, and the solution of problem (2) tends to involve many nonzero coefficients. In order to find a sparse solution, an l_1-norm penalty term can be added as

$$\mathbf{x}^* \leftarrow \underset{\mathbf{x} \in \mathbb{R}^n}{\arg\min} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda \|\mathbf{x}\|_1 \ \text{ s.t. } \mathbf{x} \geq 0, \tag{3}$$

where $\lambda \geq 0$. The set of components corresponding to the nonzero elements of $\mathbf{x}^*$ are then used to estimate the children in the LCA tree. The l_1-norm regularization term in (3) is known to promote a sparse solution for $\mathbf{x}^*$, and the penalty parameter λ controls the degree of sparsity: The larger the λ, the lesser the number of nonzero elements in $\mathbf{x}^*$; conversely, the smaller the λ, the greater the number of nonzero elements in $\mathbf{x}^*$. Problem (3) is called the l_1-regularized nonnegativity-constrained least squares (L1-NLS) problem.

A difficulty in using the L1-NLS problem (3) is that often it is unclear how to choose λ. A common approach is to obtain solutions for various λ values and determine an appropriate value through qualitative or quantitive evalutions. Cross validation is typically used for supervised learning

Table I
ENVIRONMENTAL IMPACT FACTORS OF SOME NODES IN THE DESKTOP COMPUTER LCA TREE

	Sewage treatment (m^3 waste water) eco-toxicity	Radioactive waste (kg waste) land filling	Impacts on vegetation (RER m^2 ppm h) photochemical ozone formation
Electricity (kWh)	113.98	6.0638E-5	1.9741
Steel (kg)	1726.5	4.8377E-5	11.778
CD-ROM Drive (unit)	56106	1.0936E-3	109.59
HDD (unit)	27314	7.369E-4	66.808
Power Supply (unit)	40160	2.039E-3	224.44
Circuit Board (kg)	593750	1.0298E-2	983.24
Aluminium (kg)	2035.5	3.1756E-4	34.727
ABS Plastic (kg)	1838.7	7.2846E-7	25.36

problems, but it is not applicable to LCA tree discovery because an appropriate λ value for one parent node might be different from that for another. Our algorithm discovers the full regularization path of the L1-NLS problem. The main advantage of obtaining the full regularization path is that solutions for all possible λ's can be computed in a single execution of the algorithm. The solution path provides important insights regarding the different solutions of problem (3) with varying degrees of sparsity.

III. RELATED WORK

NLS. The NLS method has been used to address physical nonnegativity constraints in least squares estimation. In chemometrics, multivariate image analysis and multivariate curve resolution have been approached through the use of NLS [10]. In Raman spectroscopy, the NLS method has been used to identify the chemical composition of substances. NLS problems have also been used to compute nonnegative matrix factorizations [11], [12].

L_1-regularization has been a popular method for promoting sparsity. When nonnegativity constraints are dropped from problem (3), the formulation reduces to the well-known Lasso problem [6]. Our full-regularization path algorithm is built upon homotopy methods for Lasso [7], [8] but additionally deals with complication arising from nonnegativity constraints. Rosset and Zhu characterized conditions under which the regularization paths are piecewise linear [13], but their work did not address constrained minimization problems. For the L1-NLS problem, Morup et al. [9] proposed nonnegative least angle regression (NLARS) algorithm, but they provided only an incomplete sketch without proof of correctness. Our proposed algorithm generates the full regularization path of the L1-NLS problem, and we also prove its correctness.

It is worth contrasting formulation (3) with a previous formulation for promoting sparsity in the NLS problem. In literature on nonnegative matrix factorization [12], [14], a formulation for promoting a sparse solution was

$$\arg\min_{\mathbf{x}\in\mathbb{R}^n} \|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2 + \lambda\|\mathbf{x}\|_1^2 \text{ s.t. } \mathbf{x}\geq 0, \tag{4}$$

where the l_1-norm term is squared. Problem (4) can be recast into a simple NLS problem to which existing NLS algorithms can be applied without modification [12], [14]. However, its solution path with respect to λ is not piecewise-linear due to the squared l_1-norm term. Hence, it is difficult to discover the full-regularization path of problem (4). When it comes to problem (3), on the other hand, our method presented here effectively discovers its full-regularization path, and the computational complexity of the proposed algorithm is the same as that of NLS algorithms.

LCA tree discovery. There has been relatively little prior work in the space of LCA tree discovery. The only work to the best of our knowledge is by Sundaravaradan et al. [2]. Their method involves repeatedly solving the NLS problems with an additional strategy to reduce nonzero variables. Since their strategy for inducing sparsity is rather ad-hoc, it is difficult to understand the behaviour of the method. In contrast, designed based on l_1-norm regularization, our approach provides full regularization paths that reveal a solution spectrum from sparse to dense ones. Another drawback of the Sundaravaradan et al. method, which we point out in Section V, is that it requires a large amount of time when executed on a real-world problem.

IV. REGULARIZATION PATHS FOR L1-NLS

Our full-regularization path algorithm for problem (3) is presented in Algorithm 1. The Karush-Kuhn-Tucker (KKT) optimality conditions for problem (3) are written as

$$\begin{aligned} \mathbf{g} &= \mathbf{A}^T\mathbf{A}\mathbf{x}-\mathbf{A}^T\mathbf{b}, && (5a)\\ \mathbf{g} &\geq -\lambda, && (5b)\\ \mathbf{x} &\geq 0, && (5c)\\ x_i\,(g_i+\lambda) &= 0 \text{ for } \forall i. && (5d) \end{aligned}$$

The goal of the algorithm is, when $\mathbf{A}$ and $\mathbf{b}$ are given, to produce all possible triples of $\mathbf{x}$, $\mathbf{g}$, and λ that satisfy conditions (5).

An easy case is when λ is large enough. Let

$$\lambda_{max} = \arg\min\left\{\left(-\mathbf{A}^T\mathbf{b}\right)_i : i=1,\cdots,n\right\}.$$

If $\mathbf{x}=\mathbf{0}$ and $\mathbf{g}=-\mathbf{A}^T\mathbf{b}$, any $\lambda\geq\lambda_{max}$ satisfies (5). In other words, for $\lambda\in[\lambda_{max},+\infty]$, $\mathbf{x}=\mathbf{0}$ is the solution for (3). Therefore, the task of our algorithm is to find the solutions for $\lambda\in[0,\lambda_{max}]$.

Algorithm 1 Full Regularization Path for L_1-regularized Nonnegative Least Squares

Input: $\mathbf{A} \in \mathbb{R}^{p\times n}$, $\mathbf{b} \in \mathbb{R}^p$
Output: $\{\mathbf{x}^{(k)}, \lambda^{(k)}\}_{k=0}^{K}$

1: $\mathbf{x}^{(0)} \leftarrow 0, \mathbf{g}^{(0)} \leftarrow -\mathbf{A}^T\mathbf{b}$
2: $j^{(0)} \leftarrow \arg\min\left\{g_i^{(0)} : i = 1, \cdots, n\right\}$
3: $\mathcal{P}^{(0)} \leftarrow \left\{j^{(0)}\right\}, \mathcal{A}^{(0)} \leftarrow \{1, \cdots, n\} \setminus \left\{j^{(0)}\right\}$
4: $\lambda^{(0)} \leftarrow -g_{j^{(0)}}^{(0)}$
5: **for** $k = 0, 1, 2, \cdots$ while $\lambda > 0$ **do**
6: $\quad \mathbf{z}_{\mathcal{P}^{(k)}}^{(k)} \leftarrow \left(\left(\mathbf{A}_{\mathcal{P}^{(k)}}\right)^T \mathbf{A}_{\mathcal{P}^{(k)}}\right)^{-1} \left(\mathbf{A}^T\mathbf{b}\right)_{\mathcal{P}^{(k)}}, \mathbf{z}_{\mathcal{A}^{(k)}}^{(k)} \leftarrow \mathbf{0}$
7: $\quad \mathbf{h}^{(k)} \leftarrow \mathbf{A}^T\mathbf{A}\mathbf{z}^{(k)} - \mathbf{A}^T\mathbf{b}$
8: $\quad$ Let $\mathbf{q}^{(k)} \in \mathbb{R}^n$ be

$$q_i^{(k)} = \begin{cases} \frac{x_i^{(k)}}{x_i^{(k)} - z_i^{(k)}}, & i \in \mathcal{P}^{(k)} \text{ and } z_i^{(k)} < 0, \\ \frac{\lambda^{(k)} + g_i^{(k)}}{\lambda^{(k)} + g_i^{(k)} - h_i^{(k)}}, & i \in \mathcal{A}^{(k)} \text{ and } h_i^{(k)} < 0, \\ 1 & \text{otherwise.} \end{cases}$$

9: $\quad j^{(k)} \leftarrow \arg\min_i q_i^{(k)}$, $t^{(k)} \leftarrow q_{j^{(k)}}^{(k)}$
10: $\quad$ **if** $t^{(k)} < 1$ and $j^{(k)} \in \mathcal{P}^{(k)}$ **then**
11: $\quad\quad \mathcal{P}^{(k+1)} \leftarrow \mathcal{P}^{(k)} \setminus \left\{j^{(k)}\right\}$, $\mathcal{A}^{(k+1)} \leftarrow \mathcal{A}^{(k)} \cup \left\{j^{(k)}\right\}$
12: $\quad$ **else if** $t^{(k)} < 1$ and $j^{(k)} \in \mathcal{A}^{(k)}$ **then**
13: $\quad\quad \mathcal{P}^{(k+1)} \leftarrow \mathcal{P}^{(k)} \cup \left\{j^{(k)}\right\}$, $\mathcal{A}^{(k+1)} \leftarrow \mathcal{A}^{(k)} \setminus \left\{j^{(k)}\right\}$
14: $\quad$ **end if**
15: $\quad \mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} + t^{(k)}(\mathbf{z}^{(k)} - \mathbf{x}^{(k)})$
16: $\quad \mathbf{g}^{(k+1)} \leftarrow \mathbf{g}^{(k)} + t^{(k)}(\mathbf{h}^{(k)} - \mathbf{g}^{(k)})$
17: $\quad \lambda^{(k+1)} \leftarrow \left(1 - t^{(k)}\right)\lambda^{(k)}$
18: **end for**

The general strategy of our algorithm is to trace solutions with decreasing λ from $\lambda = \lambda_{max}$ to $\lambda = 0$. In Algorithm 1, $\mathcal{A}$ represents the index set of zero variables, and $\mathcal{P}$ represents the index set of variables that are allowed to be nonzero. The algorithm begins with $\lambda = \lambda_{max}$, where $\mathbf{x} = \mathbf{0}$ is the solution and hence $\mathcal{P}^{(0)} = \emptyset$, $\mathcal{A}^{(0)} = \{1, \cdots, n\}$. Then it follows the solution path with decreasing λ until λ becomes zero.

The k-th iteration of Algorithm 1 begins with $\mathbf{x}^{(k)}$, $\lambda^{(k)}$, $\mathcal{P}^{(k)}$, and $\mathcal{A}^{(k)}$. Next line segment of the solution path is searched assuming that variables associated with $\mathcal{A}^{(k)}$ are zero and variables associated with $\mathcal{P}^{(k)}$ are allowed to be nonzero. Subvector $\mathbf{z}_{\mathcal{P}^{(k)}}^{(k)}$ is computed in Step 6 as an unconstrained least squares solution for variables in $\mathcal{P}^{(k)}$, and subvector $\mathbf{z}_{\mathcal{A}^{(k)}}^{(k)}$ is set with zeros, together constituting $\mathbf{z}^{(k)} \in \mathbb{R}^n$. The line segment from $\mathbf{x}^{(k)}$ to $\mathbf{z}^{(k)}$ is the direction along which the solution paths are searched: While $\mathbf{x}^{(k)}$ is on the solution paths (i.e., satisfies conditions (5)), $\mathbf{z}^{(k)}$ is not necessarily on the solution paths. Along the line segment from $\mathbf{x}^{(k)}$ to $\mathbf{z}^{(k)}$, if one of more variables violate the KKT conditions before reaching $\mathbf{z}^{(k)}$, the maximum step-lengths for which each variable remains in the solution path are computed and stored in $\mathbf{q}^{(k)} \in \mathbb{R}^n$. The minimum of such step-lengths is chosen in Step 9, and it indicates where the solution paths should proceed to. An index that provided the minimum step length is exchanged between $\mathcal{P}^{(k)}$ and $\mathcal{A}^{(k)}$ as shown in Steps 10 and 12. Values for $\mathbf{x}^{(k+1)}$, $\mathbf{g}^{(k+1)}$, and $\lambda^{(k+1)}$ are computed accordingly.

In order to prove that Algorithm 1 correctly produces the full-regularization path, we first show the following lemmas.

Lemma 1: For $\forall k$, $\mathbf{g}^{(k)}$ computed by Algorithm 1 satisfies

$$\mathbf{g}^{(k)} = \mathbf{A}^T\mathbf{A}\mathbf{x}^{(k)} - \mathbf{A}^T\mathbf{b}. \tag{6}$$

Proof: We prove by induction. For $\mathbf{x}^{(0)} = \mathbf{0}$ and $\mathbf{g}^{(0)} = -\mathbf{A}^T\mathbf{b}$, equation (6) holds. Suppose $\mathbf{x}^{(k)}$ and $\mathbf{g}^{(k)}$ satisfy (6), then using Steps 15 and 16,

$$\begin{aligned} \mathbf{g}^{(k+1)} &= \left(1 - t^{(k)}\right)\mathbf{g}^{(k)} + t^{(k)}\mathbf{h}^{(k)} \\ &= \left(1 - t^{(k)}\right)\left(\mathbf{A}^T\mathbf{A}\mathbf{x}^{(k)} - \mathbf{A}^T\mathbf{b}\right) \\ &\quad + t^{(k)}\left(\mathbf{A}^T\mathbf{A}\mathbf{z}^{(k)} - \mathbf{A}^T\mathbf{b}\right) \\ &= \mathbf{A}^T\mathbf{A}\left(\mathbf{x}^{(k)} + t^{(k)}\left(\mathbf{z}^{(k)} - \mathbf{x}^{(k)}\right)\right) - \mathbf{A}^T\mathbf{b} \\ &= \mathbf{A}^T\mathbf{A}\mathbf{x}^{(k+1)} - \mathbf{A}^T\mathbf{b}. \end{aligned}$$

Hence, (6) holds for $k+1$, completing the proof. ■

The second lemma establishes optimality conditions that hold for $\mathcal{P}^{(k)}$ and $\mathcal{A}^{(k)}$.

Lemma 2: Suppose $\mathbf{x}^{(k)}, \mathbf{g}^{(k)}, \lambda^{(k)}, \mathcal{P}^{(k)}, \mathcal{A}^{(k)}$ are generated by Algorithm 1 for $\forall k$. For each $i \in \{1, \cdots, n\}$,

$$i \in \mathcal{P}^{(k)} \quad \rightarrow \quad x_i^{(k)} \geq 0,\ g_i^{(k)} = -\lambda^{(k)} \tag{7a}$$

$$i \in \mathcal{A}^{(k)} \quad \rightarrow \quad x_i^{(k)} = 0,\ g_i^{(k)} \geq -\lambda^{(k)}. \tag{7b}$$

Proof: We prove by induction. If $k = 0$, then $\mathbf{x}^{(0)} = 0$ and $\lambda^{(0)} = \min\left\{g_i^{(0)} : \mathbf{g}^{(0)} = -\mathbf{A}^T\mathbf{b}\right\}$ clearly satisfy conditions (7).

Suppose conditions (7) hold for $0, \cdots, k$: We will show that they hold for $k+1$. For $j^{(k)}$ selected in Step 9, $t^{(k)}$ ($0 \leq t^{(k)} \leq 1$) represents the step length taken along the line segment from $\mathbf{x}^{(k)}$ to $\mathbf{z}^{(k)}$. With this step length, $\mathbf{x}^{(k+1)}$ and $\mathbf{g}^{(k+1)}$ are computed by a convex combination as shown in Step 15 and Step 16, respectively. Since $t^{(k)}$ is a minimum of all $q_i^{(k)}$'s, one of the following properties holds for $\forall i \in \{1, \cdots, n\}$:

Case 1: $i \in \mathcal{P}^{(k)}$ and $z_i^{(k)} \geq 0$

Case 2: $i \in \mathcal{P}^{(k)}$ and $z_i^{(k)} < 0$ and $t^{(k)} \leq \dfrac{x_i^{(k)}}{x_i^{(k)} - z_i^{(k)}}$

Case 3: $i \in \mathcal{A}^{(k)}$ and $h_i^{(k)} \geq 0$

Case 4: $i \in \mathcal{A}^{(k)}$ and $h_i^{(k)} < 0$ and $t^{(k)} \leq \dfrac{\lambda^{(k)} + g_i^{(k)}}{\lambda^{(k)} + g_i^{(k)} - h_i^{(k)}}$.

We will consider each of these cases and show the following:

$$i \in \mathcal{P}^{(k)} \quad \rightarrow \quad x_i^{(k+1)} \geq 0,\ g_i^{(k+1)} = -\lambda^{(k+1)} \quad (8a)$$

$$i \in \mathcal{A}^{(k)} \quad \rightarrow \quad x_i^{(k+1)} = 0,\ g_i^{(k+1)} \geq -\lambda^{(k+1)}. \quad (8b)$$

Observe that for $\mathbf{h}^{(k)}$ in Step 7,

$$h_i^{(k)} = 0 \text{ for } \forall i \in \mathcal{P}^{(k)} \quad (9)$$

due the the way $\mathbf{z}^{(k)}$ is computed.

Case 1: $x_i^{(k)} \geq 0$ due to (7a) (induction hypothesis) and $z_i^{(k)} \geq 0$; hence, $x_i^{(k+1)} = x_i^{(k)} + t^{(k)}(z_i^{(k)} - x_i^{(k)}) \geq 0$. Using $h_i = 0$ (from (9)) and (7a), $g_i^{(k+1)} = g_i^{(k)} + t^{(k)}(h_i^{(k)} - g_i^{(k)}) = \left(1 - t^{(k)}\right) g^{(k)} = -\left(1 - t^{(k)}\right) \lambda^{(k)} = -\lambda^{(k+1)}$.

Case 2: Using $x_i^{(k)} \geq 0$ and $t^{(k)} \leq \frac{x_i^{(k)}}{x_i^{(k)} - z_i^{(k)}}$,

$$x_i^{(k+1)} = x_i^{(k)} + t^{(k)}\left(z_i^{(k)} - x_i^{(t)}\right) \geq x_i^{(k)} - x_i^{(k)} = 0.$$

Similarly to Case 1, $g_i^{(k+1)} = -\lambda^{(k+1)}$.

Case 3: From $x_i^{(k)} = z_i^{(k)} = 0$, we have $x_i^{(k+1)} = 0$. From $g_i^{(k)} \geq -\lambda^{(k)}$ due to (7b) (induction hypothesis) and $h_i^{(k)} \geq 0$, $g_i^{(k+1)} = \left(1 - t^{(k)}\right) g_i^{(k)} + t^{(k)} h_i^{(k)} \geq -\left(1 - t^{(k)}\right) \lambda^{(k)} = -\lambda^{(k+1)}$.

Case 4: From $x_i^{(k)} = z_i^{(k)} = 0$, we have $x_i^{(k+1)} = 0$. From $t^{(k)} \leq \frac{\lambda^{(k)} + g_i^{(k)}}{\lambda^{(k)} + g_i^{(k)} - h_i^{(k)}}$ and $h_i^{(k)} < 0 \leq \lambda^{(k)} + g_i^{(k)}$ (induction hypothesis),

$$g_i^{(k+1)} \quad (10a)$$

$$= g_i^{(k)} + t^{(k)}\left(h_i^{(k)} - g_i^{(k)}\right) \quad (10b)$$

$$\geq g_i^{(k)} + t^{(k)}\left(h_i^{(k)} - g_i^{(k)}\right) + \left(\frac{\lambda^{(k)} + g_i^{(k)}}{\lambda^{(k)} + g_i^{(k)} - h_i^{(k)}} - t^{(k)}\right)\left(h_i^{(k)} - g_i^{(k)} - \lambda^{(k)}\right) \quad (10c)$$

$$= -\lambda^{(k)} + t^{(k)}\lambda^{(k)} = -\lambda^{(k+1)}. \quad (10d)$$

From Cases 1-4, we have shown (8) holds. Now, we will show that statements (8) holds also with $\mathcal{P}^{(k+1)}$ and $\mathcal{A}^{(k+1)}$ instead of $\mathcal{P}^{(k)}$ and $\mathcal{A}^{(k)}$. Because one index is exchanged between $\mathcal{P}^{(k)}$ and $\mathcal{A}^{(k)}$ in each iteration, we consider $j^{(k)}$ is moved from $\mathcal{P}^{(k)}$ to $\mathcal{A}^{(k)}$ or from $\mathcal{A}^{(k)}$ to $\mathcal{P}^{(k)}$. If the condition in Step 10 is true, $j^{(k)} \in \mathcal{P}^{(k)}$ and $j^{(k)} \in \mathcal{A}^{(k+1)}$: Because $t^{(k)}$ is the minimum, we have $t^{(k)} = \frac{x_{j^{(k)}}^{(k)}}{x_{j^{(k)}}^{(k)} - z_{j^{(k)}}^{(k)}}$ and $x_{j^{(k)}}^{(k+1)} = x_{j^{(k)}}^{(k)} + t^{(k)}\left(z_{j^{(k)}}^{(k)} - x_{j^{(k)}}^{(t)}\right) = 0$. If the condition in Step 12 is true, $j^{(k)} \in \mathcal{A}^{(k)}$ and $j^{(k)} \in \mathcal{P}^{(k+1)}$: We have $t^{(k)} = \frac{\lambda^{(k)} + g_{j^{(k)}}^{(k)}}{\lambda^{(k)} + g_{j^{(k)}}^{(k)} - h_{j^{(k)}}^{(k)}}$ and therefore derivations in (10) holds with equality. Hence, (8) holds for $\mathcal{P}^{(k+1)}$ and $\mathcal{A}^{(k+1)}$, and this completes the proof. ■

Lemma 1 and 2 together establish that the end points of each line segment generated by Algorithm 1 is in the solution path. The rest of proof establishes that the midpoints of the line segments are also in the solution path.

Theorem 1: Suppose $\mathbf{x}^{(k)}$, $\mathbf{g}^{(k)}$, $\lambda^{(k)}$, $k = 0, 1, 2, \cdots$, are generated by Algorithm 1. Let

$$\mathbf{x}(s) = \mathbf{x}^{(k)} + s(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}),$$
$$\mathbf{g}(s) = \mathbf{g}^{(k)} + s(\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}),$$
$$\lambda(s) = \lambda^{(k)} + s(\lambda^{(k+1)} - \lambda^{(k)}).$$

For $\forall s$ such that $0 \leq s \leq 1$, $\mathbf{x}(s)$, $\mathbf{g}(s)$, and $\lambda(s)$ satisfy conditions (5) for $\forall k$.

Proof: Condition (5a) is readily satisfied by Lemma 1. If $i \in \mathcal{P}^{(k)}$, from (7) and (8),

$$g_i(s) = (1-s)g_i^{(k)} + sg_i^{(k+1)} = (s-1)\lambda^{(k)} - s\lambda^{(k+1)} = -\lambda(s),$$
$$x_i(s) = (1-s)x_i^{(k)} + sx_i^{(k+1)} \geq 0.$$

Therefore, conditions (5b), (5c), and (5d) holds.

If $i \in \mathcal{A}^{(k)}$, from (7) and (8),

$$g_i(s) = (1-s)g_i^{(k)} + sg_i^{(k+1)} \geq (1-s)\lambda^{(k)} + s\lambda^{(k+1)} = -\lambda(s),$$
$$x_i(s) = (1-s)x_i^{(k)} + sx_i^{(k+1)} = 0.$$

Therefore, conditions (5b), (5c), and (5d) holds. ■

Complexity. The time complexity of each iteration of Algorithm 1 is $O(n^3)$, dominated by the cost for solving a system of linear equations in Step 6. This is the same with the time complexity of an iteration of the standard active-set method by Lawson and Hanson [15] for the NLS problems. The number of iterations is equivalent to the number of line segments in the solution path, and it is in practice close to the number of iterations required for Lawson and Hanson's algorithm to terminate. With approximately the same computational cost, our algorithm produces entire regularization paths, whereas the NLS algorithm produces a single unregularized solution.

V. Discovery of Life Cycle Assesment Trees

Our goal in experiments is to apply our regularization path discovery approach to LCA data sets involving real products. We aim to answer the following questions:

1) Is L1-NLS an effective technique for LCA tree discovery? What is the impact of l_1-regularization? (see Section V-B)
2) How does the L1-NLS method compare against existing methods for LCA tree discovery? (See Section V-C)

A. Data Preparation

We applied our algorithm to a real environmental impact factor database. The database contains 272 environmental impact factors for 3831 different products or materials. Although the LCA tree discovery is an unsupervised procedure,

Table II
SPECTRUM OF DISCOVERED TREES FOR PSU. SEE TABLE III FOR DESCRIPTIONS OF COMPONENTS.

λ	Correct Child Nodes	False Positive Child Nodes	False Negative Child Nodes
0.984			1154,1174,7018,7116,10804,10806
0.759	10804		1154,1174,7018,7116,10806
0.377	10804	10802	1154,1174,7018,7116,10806
0.353	10804,10806	10802	1154,1174,7018,7116
0.340	10804,10806	10802,10874	1154,1174,7018,7116
0.297	10804,10806	10162,10802,10874	1154,1174,7018,7116
0.229	10804,10806	7020,10162,10802,10874	1154,1174,7018,7116
0.218	10804,10806	1974,7020,10162,10802,10874	1154,1174,7018,7116
0.186	10804,10806	1161,1974,7020,10162,10874	1154,1174,7018,7116
0.140	7116,10804,10806	1974,7020,10162,10874	1154,1174,7018
0.0968	7116,10804,10806	1974,10162,10803,10874	1154,1174,7018
0.0636	7116,10804,10806	1974,10162,10803,10874	1154,1174,7018
0.0636	7116,10804,10806	1602,1974,10162,10803,10874	1154,1174,7018
0.0636	7116,10804,10806	1974,10162,10803,10874	1154,1174,7018
0.0438	7116,10804,10806	1442,1651,10162,10803,10874	1154,1174,7018
0.0255	7116,10804,10806	1651,2442,10162,10803,10874	1154,1174,7018
0.0245	7116,10804,10806	1651,2442,7020,10162,10803,10874	1154,1174,7018
0.0240	7116,10804,10806	1651,2442,7020,10162,10803,10874	1154,1174,7018
0.0193	7116,10804,10806	1651,2442,5986,6991,7020,10162,10803,10874	1154,1174,7018
0.0170	7116,10804,10806	415,1651,2442,5986,6991,7020,10162,10803,10874	1154,1174,7018
0.0159	1174,7116,10804,10806	415,1651,2442,5986,6991,7020,10162,10803,10874	1154,7018
0.0138	1174,7116,10804,10806	1651,2442,5986,6991,7020,10162,10803,10847,10874	1154,7018
4.31e-03	1154,1174,7116,10804,10806	1651,2442,6991,7020,10162,10803,10847,10874	7018
2.27e-05	1154,1174,7018,7116,10804,10806	2442,6991,7020,10162,10803,10847,10874	
⋮	⋮	⋮	
2.66e-06	1154,1174,7018,7116,10804,10806	271,394,1150,1172,1177,1252,2198,6534,7020,7069,7219, 10802,10862,10901	
1.81e-06	1154,1174,7018,7116,10804,10806	271,394,395,995,1150,1172,1177,1252,2117,2198,6534,6569, 6578,7009,7020,7069,7079,7219,10100,10802,10862,10901	

in order to evaluate the performance of discovery methods, reference product trees are needed. Due to intellectual property reasons, however, it is difficult to obtain a large number of reference trees. We used reference trees of seven products shown in Table V. Among them, we show detailed analysis on results for Power Supply Unit (PSU) and an Electrode Negative LiC6. For each reconstruction task, all components except the target product are considered in the library set. That is, the reconstruction problem is to choose children from 3830 components using 272 impact factors. Hence, matrix $\mathbf{A}$ becomes a rank-deficient one of size 3830×272.

An issue in data preparation is the normalization of impact factor values. As can be seen from Table I, the units of impact factors and of library components are all different, and the magnitude of their values have different significance. In order to evenly consider contributions from all impact factors, we first rescaled each impact factor using its value in the target product. That is, for each $i \in \{1, \cdots, n\}$, the i^{th} row of $\mathbf{A}$ is divided by $\mathbf{b}_i$, while $\mathbf{b}_i$ is normalized to 1. Then, in order to normalize the units of library components, all columns of $\mathbf{A}$ and $\mathbf{b}$ are normalized to have unit l_2-norm.

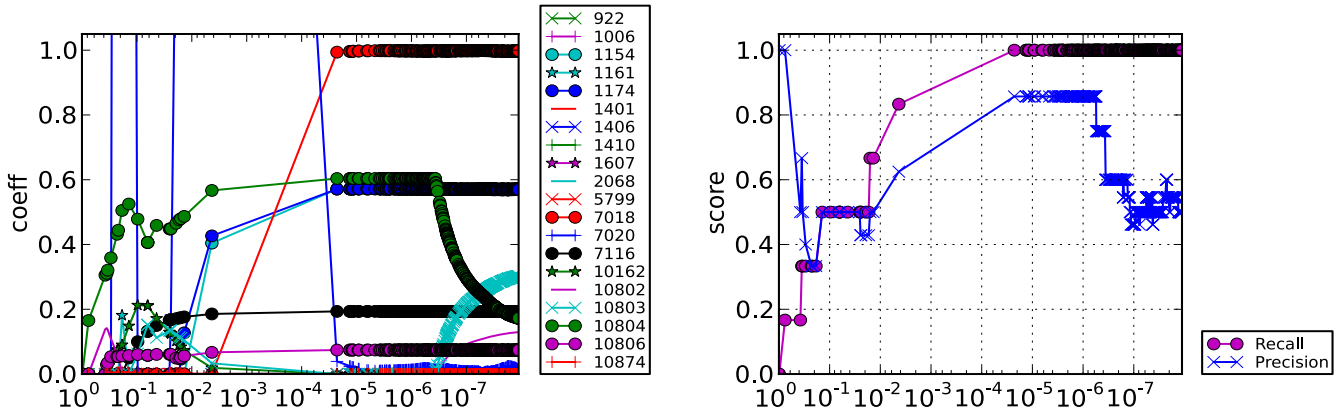
B. Regularization Path and Impact of L_1-Regularization

Coefficient paths discovered for PSU and Electrode Negative LiC6 are shown in Figure 2-(a) and Figure 3-(a), respectively. The three to five digit numbers shown in the legend represent unique codes for each library component. Descriptions of library components are shown in Table III and Table IV. In Figure 2-(a), since coefficients paths are within the range of $[0, 1]$ except that of 7020, the plot is magnified to show the relevant scale of y-axis. Observe that the λ values in x-axis are in log-scale. Although the coefficients paths are drawn as piecewise-linear for simplicity, they are piecewise-linear with respect to λ, not with respect to $\log(\lambda)$. Nonzero coefficients in the paths are interpreted as discovered children. The spectrum of discovered trees for PSU are shown in Table II.

It can be seen that l_1-regularization is crucial for accurate discovery. In Table II, when $\lambda \geq 0.984$, the solution was a zero vector. As λ decreased, the numbers of both correct and false positive nodes increased whereas the number of false negative nodes decreased. When $\lambda \approx 10^{-5}$, all true children were discovered: i.e., the set of false negative nodes is empty. When λ became smaller than 10^{-5}, the discovery quality became worse as the number of false positive nodes increased. That is, l_1 regularization with $\lambda \approx 10^{-5}$ provided the most accurate results.

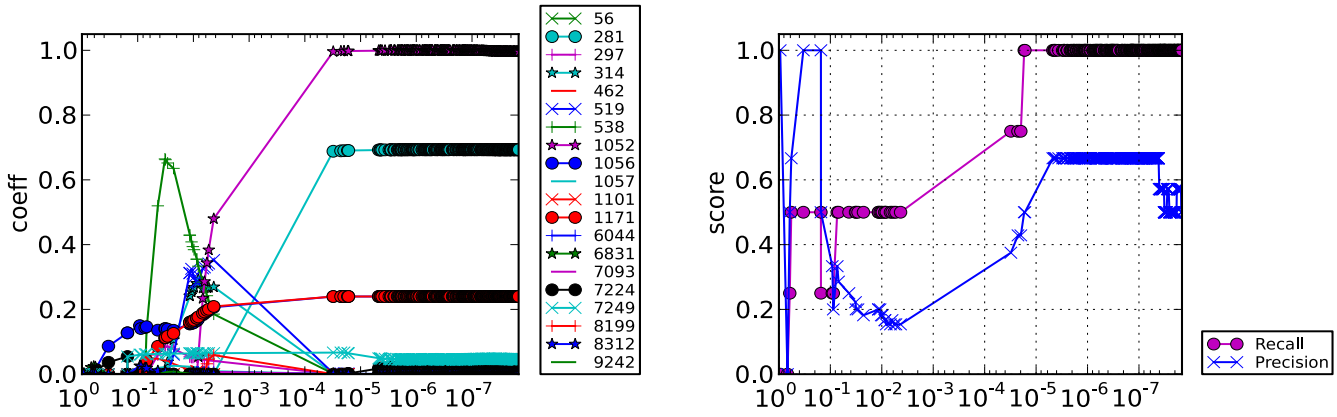
This observation can be confirmed from Figure 2 and Table III. In Figure 2-(a), the coefficient of 10804 stayed at values close to 0.6 for $\lambda \in [10^{-4}, 10^{-6}]$ but gradually decreased for λ's smaller than 10^{-6}. On the other hand, the coefficient of 10803 stayed at values close to zero for $\lambda \in [10^{-4}, 10^{-6}]$ but gradually increased for λ's smaller than 10^{-6}. Check the true and discovered coefficients from Table III, where the true coefficients of 10804 and 10803 are 0.604 and 0, respectively. The L1-NLS results in Table III were found with $\lambda = 10^{-5}$, and the coefficients at $\lambda = 10^{-5}$ are nearly the same as true coefficients in the reference tree.

For Electrode Negative LiC6, the coefficient paths shown



(a) (b)

Figure 2. Coefficient paths discovered by L1-NLS for PSU. The x-axis represents the regularization parameter, λ. (a) Coefficient values (b) Precision and recall values. See Table III for descriptions of components. Lines marked with filled circles represent the children in the reference tree. Best viewed in color.



(a) (b)

Figure 3. Coefficient paths discovered by L1-NLS for Electrode Negative LiC6. The x-axis represents the regularization parameter, λ. (a) Coefficient values (b) Precision and recall values. See Table IV for the description of components. Lines marked with filled circles represent the children in the reference tree. Best viewed in color.

in Figure 3-(a) do not immediately demonstrate the effect of l_1-regularization. Still, the effect can be observed through precision and recall scores. Figures 2-(b) and 3-(b) show precision and recall scores along the regularization paths. In Figure 2-(b), the precision score reached its maximum at $\lambda \in [10^{-5}, 10^{-6}]$ and then decreased thereafter. Similarly, in Figure 3-(b), the precision score decreased for small λ values. These results also show that l_1-regularization with $\lambda \approx 10^{-5}$ improves discovery performance.

C. Comparisons of Discovery Algorithms

We evaluated the effectiveness of L1-NLS approach through comparisons with two baseline methods. The first is the nonnegative least squares (NLS) algorithm by Lawson and Hanson [15] (no regularization), for which we used Scipy[1] with Fortran implementation of NLS.[2] The second baseline is the Sundaravaradan et al. [2] method.

Discovered coefficients from L1-NLS (with $\lambda = 10^{-5}$), NLS, and Sundaravaradan et al. method are shown in Table III and Table IV. For PSU (in Table III), L1-NLS discovered all true children and even recovered the coefficients very accurately. L1-NLS discovered a few false positive children, but their corresponding coefficients are small. On the other hand, NLS discovered none of the true children. Instead, NLS provided numerous false positive nodes that do not constitute PSU. For Electrode Negative LiC6 (in Table IV), L1-NLS discovered all true children and recovered accurate coefficients except that of 7224. NLS discovered 281, 1056, 7224 but failed to identify 1171. In addition, coefficients obtained by NLS were inaccurate for 1056 and 7224. Although both L1-NLS and NLS have found several false positive nodes, coefficients for false positive nodes from L1-NLS are generally quite small whereas that is not necessarily the case for NLS.

An interesting discovery in Table III is that the results

[1] http://www.scipy.org

[2] http://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.nnls.html

Table III

LIBRARY COMPONENTS INVOLVED IN LCA DISCOVERY FOR PSU. COEFFICIENTS FOR L1-NLS ARE OBTAINED FROM $\lambda = 10^{-5}$. COMPONENTS WITH NO COEFFICIENT VALUES ARE INCLUDED AS THEY PARTICIPATE IN THE SOLUTION PATH SHOWN IN FIGURE 2.

Id	Name	Coefficients				Unit
		True	L1-NLS	NLS	[2]	
317	portafer, at plant			0.387		kg
326	sodium borates, at plant			0.890		kg
415	liquid storage tank, chemicals, organics					unit
611	electricity, low voltage, production CENTREL, at grid			0.640		kWh
729	electricity, high voltage, at grid			1.486		kWh
844	anthracite, burned in stove 5-15kW			0.459		MJ
865	electricity, hard coal, at power plant			0.668		kWh
869	hard coal, burned in power plant			3.192		MJ
1006	lignite briquettes, at plant			3.326		MJ
1030	lignite, burned in power plant			0.738		MJ
1036	lignite, burned in power plant			2.386		MJ
1154	steel, low-alloyed, at plant	0.572	0.571			kg
1161	contour, brass					kg
1174	sheet rolling, steel	0.572	0.572			kg
1177	welding, gas, steel			0.277		m
1182	zinc coating, pieces, adjustment per um			4.716		m2
1307	natural gas, burned in cogen 1MWe lean burn			12.803		MJ
1392	natural gas, burned in gas motor, for storage			5.229		MJ
1442	plant offshore, natural gas, production					unit
1445	sweetening, natural gas			1.967		Nm3
1602	oil boiler 100kW					unit
1651	diesel-electric generating set production 10MW					unit
1969	transport, transoceanic tanker			4.767		tkm
1973	goods wagon				4.61e-6	unit
1974	locomotive					unit
1980	operation, freight train, electricity			4.985		tkm
2052	disposal, building, waste wood, untreated, to final disposal			0.281		kg
2085	process-specific burdens, inert material landfill			0.416		kg
2288	tap water, at user			15.587		kg
2442	chopper, mobile, diesel		1.17e-08			unit
5877	blast furnace gas, burned in power plant			3.239		MJ
5934	electricity, nuclear, at power plant pressure water reactor			0.335		kWh
5986	uranium open pit mine					unit
5987	uranium underground mine				1.55e-7	unit
6072	operation, regional train, SBB mix			1.563		pkm
6935	heat, biogas, at diffusion absorption heat pump 4kW, future			6.930		MJ
6991	desktop computer, without screen, at plant		1.40e-07			unit
7018	plugs, inlet and outlet, for computer cable, at plant	1.0	0.997			unit
7020	plugs, inlet and outlet, for network cable, at plant		0.0210			unit
7059	cable printer cable without plugs at plant				2.79	m
7092	zinc monosulphate				6.91e-2	kg
7116	cable, ribbon cable, 20-pin, with plugs, at plant	0.194	0.193			kg
7189	electricity hydropower at reservoir power plant				0.146	kWh
8323	deep drawing steel 38000 kN press single stroke operatio				0.207	kg
10152	silver from lead production at refinery				2.58e-3	kg
10157	production efforts inductor				4.85e-3	kg
10162	connector, computer, peripherical type, at plant		5.46e-05			kg
10802	printed wiring board, power supply unit desktop PC, Pb containing, at plant					kg
10803	printed wiring board, power supply unit desktop PC, Pb free, at plant					kg
10804	printed wiring board, power supply unit desktop PC, solder mix, at plant	0.604	0.604		0.615	kg
10806	fan, at plant	0.0742	0.0742			kg
10844	laminating foil with acrylic binder				32.32	m2
10847	exhaust air roof hood, steel, DN 400, at plant					unit
10862	silencer, steel, DN 315, 50 mm, at plant		2.33e-06			unit
10869	control and wiring, decentralized unit, at plant		2.95e-07			unit
10874	ventilation equipment, Storkair G 90, at plant					unit
10976	disposal fluorescent lamps				6.98	kg

from L1-NLS indicate that 6991 (desktop computer) is a child of PSU. In fact, the truth is the other way around since PSU is a part of a desktop computer. The coefficient for 6991 discovered by L1-NLS is very small: 1.40e-07. The small coefficient implies that 6991 (desktop computer) is discovered because PSU itself is part of a desktop computer. Observe that the unit of 6991 (desktop computer) is 'unit', indicating its coefficient takes only integral values. Although 6991 is labeled as a false positive discovery, it might be possible to remove this error if the units of library components are incorporated in a discovery method. We mention this as future work in Section VI.

We have computed the residual norms:

$$\begin{aligned} r_{L1-NLS} &= \|\mathbf{A}\mathbf{x}_{L1-NLS} - \mathbf{b}\|_2 / \|\mathbf{b}\|_2 \\ r_{NLS} &= \|\mathbf{A}\mathbf{x}_{NLS} - \mathbf{b}\|_2 / \|\mathbf{b}\|_2 , \end{aligned}$$

where $\mathbf{x}_{L1-NLS}$ and $\mathbf{x}_{NLS}$ are solutions from L1-NLS and NLS, respectively. For PSU, $r_{L1-NLS} = 2.62 \times 10^{-5}$ and $r_{NLS} = 5.83 \times 10^{-6}$. For Electrode Negative LiC6, $r_{L1-NLS} = 3.80 \times 10^{-5}$ and $r_{NLS} = 7.50 \times 10^{-6}$. This implies that the NLS solution, which is optimal in terms of the least squares residual norm, is not necessarily the best solution for LCA discovery. Although the L1-NLS solution provides a little higher residual norm due to l_1-regularization, it correctly discovered LCA tree children and

Table IV

LIBRARY COMPONENTS INVOLVED IN LCA DISCOVERY FOR ELECTRODE NEGATIVE LIC6. COEFFICIENTS FOR L1-NLS ARE OBTAINED FROM $\lambda = 10^{-5}$. COMPONENTS WITH NO COEFFICIENT VALUES ARE SHOWN AS THEY PARTICIPATE IN THE SOLUTION PATH SHOWN IN FIGURE 3.

Id	Name	Coefficients				Unit
		True	L1-NLS	NLS	[2]	
281	graphite, at plant	0.693	0.691	0.664		kg
297	magnesium sulphate, at plant		1.46e-04			kg
301	oxygen, liquid, at plant			0.162		kg
314	pigments, paper production, unspecified, at plant		1.02e-03			kg
461	clay, at mine			0.0305		kg
462	expanded vermiculite, at plant		3.32e-05			kg
473	pumice, at mine			0.0138		kg
519	asbestos, crysotile type, at plant		1.51e-03			kg
538	clay plaster, at plant		6.66e-04			kg
881	hard coal, burned in power plant			0.0212		MJ
1006	lignite briquettes, at plant			0.0300		MJ
1036	lignite, burned in power plant			0.0160		MJ
1052	aluminium scrap, new, at plant		0.998			kg
1054	aluminium, primary, at plant			0.0403		kg
1055	aluminium, primary, liquid, at plant			0.0306		kg
1056	aluminium, production mix, at plant	0.24	0.240	0.0389	0.233	kg
1057	aluminium, production mix, cast alloy, at plant		4.37e-05	0.0584		kg
1058	aluminium, production mix, wrought alloy, at plant			0.0634		kg
1145	rhodium, secondary, at refinery		2.98e-08			kg
1171	sheet rolling, aluminium	0.24	0.240		0.244	kg
1394	natural gas, burned in gas motor, for storage			0.131		MJ
1395	natural gas, burned in gas motor, for storage			0.176		MJ
1402	natural gas, burned in gas turbine, for compressor station			0.0441		MJ
1404	natural gas, burned in gas turbine, for compressor station			0.107		MJ
1410	sweet gas, burned in gas turbine, production			0.132		MJ
1447	transport, natural gas, offshore pipeline, long distance			0.0411		tkm
1593	heavy fuel oil, burned in refinery furnace			0.0371		MJ
1608	refinery gas, burned in furnace			0.0324		MJ
1775	facade construction, integrated, on roof		1.38e-06		6.45e-05	unit
1803	epoxy resin, liquid, disaggregated data, at plant		1.50e-06			kg
1980	operation, freight train, electricity			0.167		tkm
2291	water, decarbonised, at plant			0.0642		kg
6074	operation, long-distance train, SBB mix			0.0690		pkm
6831	facade construction, integrated, at building					m2
6833	flat roof construction, on roof		1.08e-07			m2
7093	hydrogen sulphide, H2S, at plant		2.65e-05		3.86e-02	kg
7224	lithium, at plant	0.0668	2.31e-03	0.0192		kg
7249	chlorine, gaseous, lithium chloride electrolysis, at plant		0.0645	0.0467	6.68e-02	kg
8199	compressed air, best installation, 30kW, 6 bar gauge, at supply network					m3
8312	aluminium product manufacturing, average metal working					kg
9272	facilities precious metal refinery	1.13e-07				unit

their coefficients whereas NLS failed to do so.

The method by Sundaravaradan et al. [2] did not perform well for the PSU tree, as shown in Table III. It failed to detect most of the true children except 10804 and instead discovered a number of false positives. On the other hand, it provided good results for Electrode Negative LiC6, as shown in Table IV. Although it did not detect 7224, it has found 1056 and 1171 with reasonably accurate coefficients. In addition to the failure on PSU, a major drawback of the Sundaravaradan et al. method was its significant time consumption. While L1-NLS and NLS methods were typically completed within 1-2 minutes, the Sundaravaradan et al. method had to run for between 5 to 10 hours on the same computer.

Care should be take when comparing the results in Tables III and IV and those reported in their paper [2]. In experiments in [2], the authors have used a limited set of library components that only contains the union of children in seven reference trees. As a result, the library set consisted of only 76 components, making the discovery problem significantly easy. In our experiments, we solved the discovery problem in a fully unsupervised manner using a library set that contains all available 3831 components. Although Sundaravaradan et al. [2] have reported meaningful recovery results using the limited library components, results in Tables III and IV reveal that their method have issues when attempted in a realistic setting.

Discovery results for all seven reference trees are shown in Table V. Execution results of Sundaravaradan et al. method are not included because it was not completed after 10 hours of execution. Two L1-NLS results with $\lambda = 10^{-5}$ and $\lambda = 10^{-4}$ are included. Overall, the results from NLS are inaccurate: the number of correct discoveries (TP) are small whereas the number of false positives are large. For example, for 10158 (HDD desktop), NLS made only 2 correct discoveries but provided 167 false positives. L1-NLS returned superior results than NLS. Discovery using L1-NLS is highly accurate for 7063, 7064, and 10805; moderately accurate for 10158 and 10160; and less accurate for 6991 and 7003. Between the $\lambda = 10^{-5}$ and $\lambda = 10^{-4}$ cases, the $\lambda = 10^{-4}$ case imposed stronger regularization and therefore provided a smaller number of false positives than the $\lambda = 10^{-5}$ case. As a trade off, $\lambda = 10^{-4}$ returned less correct discoveries in the case of 6991, 7063, 10160, and 10805. Generally, the number of false positives given by L1-NLS is much smaller than that of NLS except the

Table V
SUMMARY OF RESULTS OF APPLYING L1-NLS AND NLS METHODS FOR SEVEN TARGET PRODUCTS. TP, TN, AND FP REPRESENT THE NUMBERS OF TRUE POSITIVE, TRUE NEGATIVE, AND FALSE POSITIVE COMPONENTS, RESPECTIVELY.

	Children	L1-NLS with $\lambda = 10^{-5}$				L1-NLS with $\lambda = 10^{-4}$				NLS			
		TP	TN	FP	residual	TP	TN	FP	residual	TP	TN	FP	residual
Desktop (6991)	39	7	32	106	3.00×10^{-4}	6	33	88	1.41×10^{-3}	4	35	177	5.81×10^{-6}
LiC6 electrode (7063)	5	4	1	13	3.80×10^{-5}	3	2	16	9.25×10^{-3}	3	2	131	7.50×10^{-6}
LiMn2O4 electrode (7064)	11	8	3	79	1.04×10^{-3}	8	3	70	2.76×10^{-3}	6	5	148	6.96×10^{-6}
Battery (7003)	19	2	17	86	0.0650	2	17	86	0.0650	4	15	79	0.0645
HDD desktop (10158)	16	6	10	68	1.40×10^{-4}	6	10	44	6.82×10^{-4}	2	14	167	8.37×10^{-6}
CD-ROM (10160)	18	6	12	80	1.42×10^{-4}	5	13	59	7.35×10^{-4}	1	17	168	5.53×10^{-6}
PSU (10805)	6	6	0	6	2.61×10^{-5}	5	1	8	7.16×10^{-3}	0	6	167	5.83×10^{-6}

7003 case. The 7003 (Battery) case appears to be an outlier because the least squares residual norm is particularly higher than that of other reference trees.

VI. DISCUSSION

We have presented an approach to inferring LCA trees based on l_1-regularized nonnegative least squares (L1-NLS) formulation. We proposed an algorithm that computes the full regularization paths of the L1-NLS problem and theoretically proved the correctness of the algorithm. With the proposed method, we demonstrated how the LCA tree can be discovered in an unsupervised manner from a database of impact factors. The full regularization paths discovered by our algorithm enables us to track the spectrum of potential children sets. The effectiveness of the L1-NLS method was shown through demonstrations on real-world impact factor database and by comparisons against existing methods.

Future work on the LCA tree discovery problem revolves around several themes. First, there are still many false positive discoveries involved in the L1-NLS results, and additional constraints can be incorporated to mitigate this situation. For example, each library component is given an eco-category and an eco-subcategory assignments, which have not been utilized in existing methods. If domain knowledge is available indicating that a target product can have children from only certain categories of components, components belonging to irrelevant categories can be removed before a discovery method is applied. Second, transfer learning approaches can be considered so that the discovery of LCA tree for one product can aid the discovery of LCA tree for another similar product. Third, in addition to non-negativity constraints, strict constraints can be placed to improve accurate discovery. For example, in Table III, L1-NLS correctly discovered 7018 with an accurate coefficient 0.997, when the true coefficient is 1. However, the unit of this component is indicated as 'unit', implying that only integer values are allowed for coefficients. When integer constraints are imposed on 7018 and 7020, it might be possible to eliminate the false positive discovery of 7020. Mixed integer programming can be considered to handle such constraints.

ACKNOWLEDGEMENTS

This work is supported in part by US NSF grants CCF-0937133 and IIS-0905313.

REFERENCES

[1] D. Goleman and G. Norris, "How green is my ipad?" http://www.nytimes.com/interactive/2010/04/04/opinion/04opchart.html?hp, April 2010.

[2] N. Sundaravaradan, D. Patnaik, N. Ramakrishnan, M. Marwah, and A. Shah, "Discovering life cycle assessment trees from impact factor databases," in *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence (AAAI'11)*, 2011.

[3] R. Frischknecht *et al.*, "The Ecoinvent Database: Overview and Methodological Framework," *The Intl. Journ. of Life Cycle Assessment*, vol. 10, no. 1, pp. 3–9, 2005.

[4] PE International, "GaBi - Life Cycle Assessment (LCE/LCA) Software System," http://www.gabi-software.com, 2009.

[5] S. Spatari *et al.*, "Using GaBi 3 to perform Life Cycle Assessment and Life Cycle Engineering," *The Intl. Journ. of Life Cycle Assessment*, vol. 6, no. 2, pp. 81–84, 2001.

[6] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[7] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Annals of Statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[8] M. R. Osborne, B. Presnell, and B. A. Turlach, "On the lasso and its dual," *Journal of Computational and Graphical Statistics*, vol. 9, pp. 319–337, 2000.

[9] M. Morup, K. H. Madsen, and L. K. Hansen, "Approximate l0 constrained non-negative matrix and tensor factorization," in *Proceedings of 2008 IEEE International Symposium on Circuits and Systems*, 2008.

[10] M. H. Van Benthem and M. R. Keenan, "Fast algorithm for the solution of large-scale non-negativity-constrained least squares problems," *Journal of Chemometrics*, vol. 18, pp. 441–450, 2004.

[11] H. Kim and H. Park, "Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 2, pp. 713–730, 2008.

[12] J. Kim and H. Park, "Fast nonnegative matrix factorization: An active-set-like method and comparisons," *SIAM Journal on Scientific Computing*, vol. 33, no. 6, pp. 3261–3281, 2011.

[13] S. Rosset and J. Zhu, "Piecewise linear regularized solution paths," *Annals of Statistics*, vol. 35, no. 3, pp. 1012–1030, 2007.

[14] J. Kim and H. Park, "Sparse nonnegative matrix factorization for clustering," Georgia Institute of Technology GT-CSE-08-01, Tech. Rep., 2008.

[15] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*. Prentice Hall, 1974.