AFRL-RY-WP-TR-2016-0168

CONTINUALLY PLASTIC MODELING OF NON-STATIONARY SYSTEMS

Josh Bongard and Chris Danforth

University of Vermont

SEPTEMBER 2016
Final Report

AIR FORCE RESEARCH LABORATORY
SENSORS DIRECTORATE
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7320
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

NOTICE AND SIGNATURE PAGE

*//Signature//

STEVEN SCARBOROUGH, Program Manager
Radio Frequency Exploitation Branch

//Signature//

GREGORY CAZZELL, BranchChief
Radio Frequency Exploitation Branch

*//Signature//

DOUG HAGER, Deputy
Layered Sensing Exploitation Division
Sensors Directorate

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE ***(DD-MM-YY)*** September 2016	**2. REPORT TYPE** Final	**3. DATES COVERED** ***(From - To)*** 27 September 2011 – 27 June 2016

4. TITLE AND SUBTITLE CONTINUALLY PLASTIC MODELING OF NON-STATIONARY SYSTEMS	**5a. CONTRACT NUMBER** FA8650-11-1-7155
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 61101E
6. AUTHOR(S) Josh Bongard and Chris Danforth	**5d. PROJECT NUMBER** 1000
	5e. TASK NUMBER 11
	5f. WORK UNIT NUMBER Y0GH
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Vermont 82 University Place Burlington, VT 05405	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory Sensors Directorate Wright-Patterson Air Force Base, OH 45433-7320 Air Force Materiel Command United States Air Force — Defense Advanced Research Projects Agency (DARPA/DSO) 3701 North Fairfax Drive Arlington, VA 22203	**10. SPONSORING/MONITORING AGENCY ACRONYM(S)** AFRL/RYAP
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S) AFRL-RY-WP-TR-2016-0168

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This material is based on research sponsored by Air Force Research laboratory (AFRL) and the Defense Advanced Research Agency (DARPA) under agreement number FA8650-11-1-7155. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation herein. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies of endorsements, either expressed or implied, of Air Force Research Laboratory (AFRL) and the Defense Advanced Research Agency (DARPA) or the U.S. Government. Report contains color.

14. ABSTRACT
This award supported extensions to, and novel applications of, an increasingly useful machine learning methodology known as symbolic regression. The PI of this award was involved in earlier work that established this approach as a powerful method for discovering previously unknown relationships within and among arbitrary data sets.

15. SUBJECT TERMS
Machine learning methodology, symbolic regression

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT:**	**8. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** (Monitor)
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	SAR	120	Steven Scarborough **19b. TELEPHONE NUMBER** *(Include Area Code)* N/A

Continually Plastic Modeling of Non-Stationary Systems.

DARPA/AFRL Grant FA8650-11-1-7155, **Final Report**

Principal Investigator: Josh Bongard
Department of Computer Science
University of Vermont

Co-Principal Investigator: Chris Danforth
Department of Mathematics & Statistics
University of Vermont

Friday September 9, 2016

Contents

1 Overview.

NOTE: The most important deliverable resulting from this award is the Shadow Networks project. It is described in more detail in Sect. 2.

This award supported extensions to, and novel applications of, an increasingly useful machine learning methodology known as **symbolic regression**. The PI of this award was involved in earlier work that established this approach as a powerful method for discovering previously unknown relationships within and among arbitrary data sets[1].

Unlike more traditional linear and nonlinear regression methods, which attempt to find coefficients (α) for equations with terms selected by the investigator, symbolic regression attempts to make little to no assumptions about the form of the right-hand side of a set of equations. In other words, the investigator does not need to select linear or nonlinear terms *a priori*. Symbolic regression is also preferred over other state-of-the-art machine learning methods such as deep learning because symbolic regression is a white box modeling method: often, the models it produces may be highly nonlinear yet compact, readable by any member from the domain of interest who is mathematically literate.

For example, one result from this award was a model that can successfully predict the time it takes for information to flow from individual i to individual j (T_{ij}) as a function of structural properties of the social network in which those individuals are embedded. One such property is L_{ij}, the shortest path in the network from i to j. Trained against social network 'chatter' (people tweeting and retweeting information), symbolic regression constructed this model

$$T_{ij} \quad = \quad L_{ij}(1 + \ln(L_{ij} + k_i + k_j - c_j + \frac{N - k_i^2 - k_j^2}{L_{ij}^N + k_i k_j - \rho})) \qquad (1)$$

which, even to a casual observer, can see that it takes longer for information to flow from individual i to j if they are more distant from one another in the social network. However, the additional mathematical structure in this model indicates that there are more subtle influences between individuals' location in a social network and how long it takes for information to flow between them.

Symbolic regression is useful in that it can often find relationships within a data set that are missed by other regression methods because, in the latter case, the investigator may make the wrong assumptions about what kinds of relationships may exist and thus include inappropriate terms. Symbolic regression avoids this by allowing the investigator to make little or no assumptions about what relationships *may* exist in a data set.

Over the course of this award, four projects were pursued: the first three involve applying symbolic regression to novel domains such as social networks, brain imaging, and satellite imagery. The fourth project involved theoretical work to improve symbolic regression itself. These four projects are summarized as follows:

1. **The Shadow Networks project.** We have successfully adapted symbolic regression for addressing the node prediction problem in social network data [2]: if a person, along with

[1]Bongard J. and Lipson H.(2007). Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 104(24): 9943-9948.

all their messages, is deliberately ‘scrubbed’ from chatter collected from a social network, how could one not only identify that such tampering had occurred, but where that missing node may lie in the network (i.e. identify the friends of the erased person). More details are provided in Sect. 2.

2. **Symbolically regressing brain imaging data.** Using functional Magnetic Resonance Imaging (fMRI) data collected prior to this award, we have shown that applying symbolic regression can find heretofore unknown relationships between brain regions, and that those relationships can be used to predict behavioral tendencies of the participants. As example, we found that a model produced by symbolic regression could be used to predict whether an adolescent regularly consumed alcohol or not [1]. The model made successful predictions by finding previously unknown relationships between regions of the brain implicated in reward, emotion, and thirst. More details are provided in Sect. 3.

3. **Symbolically regressing satellite imagery.** In addition to large-scale data sets produced by medical scans such as fMRI, environmental modeling from satellite data is another promising domain for symbolic regression. This award has enabled us to demonstrate a modeling approach that intelligently balances requests for data for modeling against the differing costs of data produced by less- or more-expensive sensors [9]. We have also adapted symbolic regression for use with actual satellite data for predicting the amount of water contained in the snows of the Hindu Kush [3, 6]. More details are provided in Sect. 4.

4. **Improving symbolic regression.** Much theoretical work has been accomplished as a result of this award in order to improve symbolic regression. One pair of publications demonstrated that symbolic regression can be hybridized with other gradient-descent regression methods to produce a combined algorithm that outperforms either approach working alone [4, 5]. The heart of symbolic regression relies on stochastic modifications to existing models to sometimes discover more accurate new models. In more recent work [7, 8] we have shown that these modifications can be de-randomized somewhat to improve symbolic regression’s ability to discover more accurate and more parsimonious models. More details about these advances are provided in Sect. 5.

1.1 Human capital return on investment.

This award supported two postdoctoral associates: Ilknur Icke and Nicholas Allgaier.

Dr. Icke is now a senior engineer of scientific computing for Merck. There, she is developing high throughput implementations for vaccine development and medical image registration. She is also applying deep learning methods for automatically identifying the location of cardiac left ventricles in medical scans.

Dr. Allgaier is now a postdoctoral associate in the University of Vermont Medical Center, working under the supervision of Hugh Garavan, one of the Principal Investigators of the Adolescent Brain Cognitive Development (ABCD) Study, the largest long-term study of brain development and child health in the United States, and the ENIGMA Study, an attempt to construct one of the largest multi-site, data pooled, genetic and neuroimaging data sets. Dr. Allgaier is presently applying some of the methods developed as part of this award to data from both of these studies.

1.2 Other impacts from this award.

- Symbolic regression has become a common tool among the faculty, postdoctoral associates, and graduate students who comprise the Vermont Complex Systems Center. We graduate about a half dozen graduate students and postdoctoral associates a year who go on to take up prominent positions in academia and industry. Most members are involved in analyzing and synthesizing complex neural, biological, technological, and social networks. Common application domains involve social network analysis, the smart grid, cyberinfrastructure, and sociotechnical systems.

- PI Bongard has delivered a number of presentations on work drawn from this award:

May, 2016	Trusted autonomous systems. (ACFR, University of Sydney, Australia; Invited)
May, 2016	Trusted autonomous systems. (Intl. Symp. on Trusted Autonomous Systems, Australia; **Keynote**)
Mar, 2016	Philosophical implications of robotics. (UPitt HPS Annual Lecture Series; Invited)
Feb, 2016	Evo devo robo. (University of Toronto Cognitive Science Symposium; Invited)
Dec, 2015	ShanghAI lecture (simulcast to classrooms in Europe and Asia; Invited)
Dec, 2015	New Jersey Institute of Technology (host: Gal Haspel, biology; Invited)
May, 2015	Factory of Imagination lecture, Denmark (500 attendees; **Keynote**)
Feb, 2015	ShanghAI lecture (simulcast to classrooms in Europe and Asia; Invited)
Nov, 2014	Cornell Univeristy (host: Robert Shepherd, engineering; Invited)
Sept, 2014	University of Maryland workshop on soft robotics (Invited)
Aug, 2014	Scifoo (hosts: Nature, Google, O'Reilly Media, Digital Science; Invited)
July, 2014	Workshop on Artificial Life and the Web at ALife conference (Invited)
July, 2014	International Society for Artificial Life (ISAL) Summer School (Invited)
June, 2014	DARPA Biological Technologies Office (Invited)
June, 2014	Neural Systems & Behavior Summer School, Woods Hole Marine Biology Lab (Invited)
May, 2014	EPFL, Lausanne, Switzerland (host: Auke Ispeert; Invited)
Mar, 2014	National STEM Conference (Concept Schools), Cleveland, OH (**Keynote**)
Mar, 2014	Air Force Research Laboratories (AFRL), Rome, NY (Invited)
Dec, 2013	ShanghAI lecture (simulcast to 15 classrooms in Europe and Asia; Invited)
Nov, 2013	National Autonomous University of Mexico (host: Carlos Gershenson; Invited)
Oct, 2013	University of Iowa Delta Center (host: Mark Blumberg, psychology; Invited)
Sept, 2013	eSMC neuroscience/robotics graduate summer school (host: Andreas Engel; Invited)
Sept, 2013	Evolutionary Biology lecture, University of Zurich (host: Andreas Wagner; Invited)
Aug, 2013	Gordon Research Conference on Neuroethology (host: Heather Eisten, biology; Invited)
July, 2013	Soft Robotics Workshop at ETH, Zurich (host: Fumiya Iida, robotics; **Keynote**)
June, 2013	Evolution Meeting, SSE Presidential Symposium (host: Richard Lenski, biology; Invited)
June, 2013	Evolution Meeting, Education Symposium (host: George Gilchrist, NSF; Invited)
Mar, 2013	University of Texas at Austin (host: Dana Ballard, Computer Science; Invited)
Nov, 2012	Vassar College (host: John Long, biology; Invited)

Nov, 2012	Harvard University (host: Radhika Nagpal, engineering; Invited)
June, 2012	Tufts University (host: Michael Levin, biology; Invited)
Apr, 2012	Tufts University (host: Barry Trimmer, biology; Invited)
Jan, 2012	University of Southern California (host: Francisco Valero-Cuevas, bioengineering; Invited)

1.3 Software deliverables.

- The **source code** for two versions of the enhanced symbolic regression method developed throughout this award are available publicly. Either of these methods can be adapted to novel data sets by anyone proficient in Python and machine learning methods:
  - The github repository for forward semantic propagation in symbolic regression.
  - The github repository for behavioral diversity in symbolic regression.

2 The shadow networks project.

The most important product produced by this award is the 'Shadow Networks' method. It is summarized below, and a manuscript describing its technical details follows.

An important problem in analyzing data generated by people communicating over a social network is identifying whether the communications have been deliberately tampered with. Such challenges can be broken down into two classes of problems: link prediction and node prediction. In the link prediction problem, it is assumed that edges have either been removed from a social network (i.e., information about relationships between pairs of individuals have been erased) or fictitious links have been added (i.e., fictitious relationships have been embedded in the network).

Several methods now exist for tackling the link prediction problem. However, before our work in this award, there were no methods in existence for tackling the much harder node prediction problem: a node and all of its edges are either deliberately erased (someone, along with all their relationship information, is removed) or added (a fictitious actor is added to the network).

In a preliminary publication [2] we introduced a method for successfully addressing the node prediction problem, albeit only for simulated social networks. (We are currently seeking relevant real-world social network data for this project.) We have termed our particular approach to the node prediction problem, which employs symbolic regression as a part, the Shadow Networks approach.

A summary of the approach is as follows. Imagine one has access to several social networks. In addition, one can observe not only the structural properties of that network—who is connected to whom, and how—but also dynamic properties of that network—how information flows from one person to another, and at what rate. Armed with this data, it is possible to train a model, using data from these social networks, to successfully predict how long it generally takes for information to flow from one person to another, given structural properties of the network. In essence the model has the form

$$T_{ij} = f(S_i, S_j, S_{ij}) \qquad (2)$$

where T_{ij} represents the time (on average) it takes for information to flow from individual i to individual j, S_i is a set containing structural properties of i (i.e. how many friends he has in the network, how close to a hub he is, etc.), S_j is a set containing structural properties of j, and S_{ij} is a set containing structural properties of the relationship between i and j (e.g. what is shortest path linking i and j).

Once the model learns to make predictions of flow from structure, one can apply the model to a new social network. The model then acts in a diagnostic fashion: it makes predictions for flow between each pair of individuals in the network, and if its predictions systematically fail, it is likely that that network has been tampered with in some way.

To develop an intuition for this idea, consider exposing a model to a series of pipes, each of which is composed of k concrete segments. The model then observes water poured into one end of the pipe, and measures how long it takes for the water to emerge from the other end of the pipe. This model may learn, in this simple case, that the time for water to flow through the pipe is proportional to the number of concrete segments making up the pipe. If the model is then exposed to another pipe made up of three segments, but it takes water four units of time to traverse the pipe, the model may predict that there is a fourth segment in the pipe that it was forbidden to see.

In the manuscript that follows, we demonstrate that our method can be used to detect whether nodes have been removed (omission) or added (commission) to the network. Furthermore, in the case of node removal, the model's error tends to spike for individuals who are close to the hidden node. This provides not only a signal that someone may be scrubbed from the network, but who know about the scrubbing—the hidden actor's colleagues, as evidenced by the network itself.

There are several limitations that currently exist with the method. To date it has only been validated on synthetic data. We are currently seeking data from real social networks usable for this method. Further, it assumes that the data on which the models are trained come from a fully observable network, and that these training networks have not yet been tampered with. Future work will address these limitations.

2.1 Relevance for U.S. defense and security.

The rapid rise of big data is posing novel challenges for security and defense, especially data arising from social networks. It would be of great use to be able to automatically identify whether data from social networks has been tampered with, and specifically whether information generated by one or a few individuals have been deliberately erased.

The current method also does not make assumptions about what kind of information is flowing across the network: it could be tweets flowing across a social network, packets flowing across a computer network, or text messages flowing across a cellphone network.

Given this, it is possible that this method could be adapted for discovering trojan horses in software and/or hardware systems, or cyberinfrastructure in general.

It is possible that the method could be adapted for other domains in which it is imperative to find hidden individuals in a social network. One likely future domain of application is disease modeling. If individuals suffering from the outbreak of a disease will not or cannot report to a local clinic, data about those infected individuals is lost. However, if their friends and relatives do report to the clinic, it may be possible to indentify a group of individuals who share social ties with an individual *who is not present award in the collected data.* These peripheral individuals could then be contacted to verify the existence of these missing individuals and how to contact them.

2.2 Bagrow *et al.* “Shadow Networks...” (2015).

A technical manuscript describing the shadow networks method in detail follows.

Shadow networks: Discovering hidden nodes with models of information flow

James P. Bagrow * [1,2,3,4], Suma Desu [1,2,3,4], Morgan R. Frank [1,2,3,4], Narine Manukyan [5,2,3], Lewis Mitchell [1,2,3,4], Andrew Reagan [1,2,3,4], Eric E. Bloedorn [6], Lashon B. Booker [6], Luther K. Branting [6], Michael J. Smith [6], Brian F. Tivnan [6,2,3,4], Christopher M. Danforth [1,2,3,4], Peter S. Dodds [1,2,3,4], and Joshua C. Bongard [5,2,3]

[1]Department of Mathematics & Statistics, The University of Vermont, Burlington, VT 05401, USA

[2]Vermont Complex Systems Center, The University of Vermont, Burlington, VT 05401, USA

[3]Vermont Advanced Computing Core, The University of Vermont, Burlington, VT 05401, USA

[4]Computational Story Lab, The University of Vermont, Burlington, VT 05401, USA

[5]Department of Computer Science, The University of Vermont, Burlington, VT 05401, USA

[6]The MITRE Corporation, McLean, VA 22102, USA

December 20, 2013

Abstract

Complex, dynamic networks underlie many systems, and understanding these networks is the concern of a great span of important scientific and engineering problems. Quantitative description is crucial for this understanding yet, due to a range of measurement problems, many real network datasets are incomplete. Here we explore how accidentally missing or deliberately hidden nodes may be detected in networks by the effect of their absence on predictions of the speed with which information flows through the network. We use Symbolic Regression (SR) to learn models relating information flow to network topology. These models show localized, systematic, and non-random discrepancies when applied to test networks with intentionally masked nodes, demonstrating the ability to detect the presence of missing nodes and where in the network those nodes are likely to reside.

*james.bagrow@uvm.edu

1 Introduction

The field of complex networks has emerged and matured over the last 15 years, heralded by small-world [1] and scale-free networks [2], and principally enabled by the advent of readily available large-scale datasets. Much work has been focused on simple descriptions of complex networks, leading to an evolving collection of structures, network statistics [3, 4], and generative mechanisms [5, 6, 2].

All along, the problem of missing data has been both obvious and ubiquitous—few network datasets are complete or nearly so—and yet this issue has largely been ignored. The body of work that does exist on missing data has mostly focused on the problem of unrecorded edges or interactions [7, 8, 9, 10], while only some have explored the harder problems of node and context omission [11, 12, 13] using various approaches such as inference based on maximum likelihood estimation [14, 15].

While missing data is certainly understood to affect—sometimes dramatically—different kinds of static network statistics in different ways [11], the effects of measurement error on dynamic, real social networks [16, 17, 18, 19] remain largely unknown. This problem is especially challenging when the amount of data omission is not known and can only be estimated from the observed data set. The implications for how to contend with a given network, suspected to be corrupted in some fashion, are substantial. In the case of public health policy, for example, positive evidence for the role of social contagion in the spreading of such disparate attributes as happiness [20], obesity [21], and loneliness [22], have been challenged due to their reliance on under-sampled reconstructed social networks [23].

A systematic framework to accommodate missing data for static and dynamic networks remains elusive, and provides a great challenge to the network science community. Much success in the study of complex, dynamic networks has come from approaches born out of statistical mechanics and dynamical systems, with the great example arguably being Simon's rich-get-richer model underlying scale-free networks [5, 6, 2]. Yet it is clear that many adaptive complex systems are strongly algorithmic in nature, and are not well or completely described by integrodifferential equations.

Briefly, our approach to studying missing or hidden node detection is as follows. First, we construct a set of network topologies (Sec. 2.1). We then use an idealized transaction model to simulate the flow of information "packets" across these networks. These packets could represent IP packets flowing across a computer network, citations within a scientific collaboration network, or messages passed among members of a social network such as Twitter (Sec. 2.2). Next, the resulting transaction data is collected and fed to a stochastic optimization method. This goal of this step is to generate a mathematical model that predicts the speed of information flow between pairs of nodes in the network, given structural information about those nodes and the network they were drawn from (Sec. 2.3). Finally, the evolved transaction model is presented with rates of information flow between nodes from a different network. If there are sys-

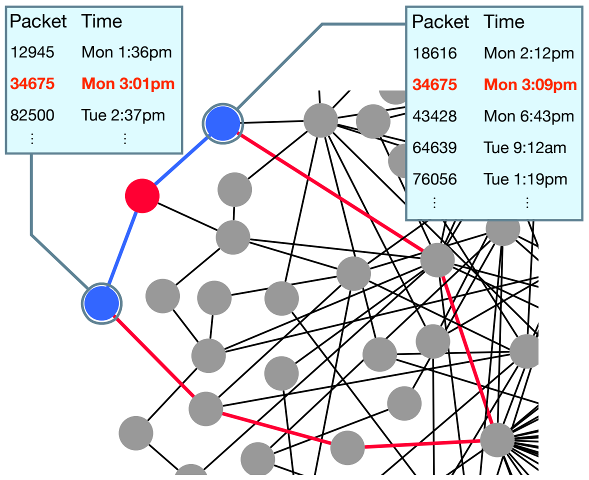


Figure 1: Illustration motivating the method. Nodes in this network pass information around (packets), and we monitor the arrival times of these packets. The two blue nodes appear much farther apart topologically when the red node is hidden. Given the observed information flows, the highlighted packet would appear to be arriving unusually quickly given the apparent long distance path it likely took (red links). This unexpectedly rapid flow may be a clue that unseen network elements are present.

tematic errors or biases in the model's prediction of information flow, this indicates that nodes may have been added or removed from the network.

The intuition underlying our approach to node prediction may be clarified by considering the cartoon example in Fig. 1. Two nodes are connected by a third node, making them two steps apart on the network topology. Due to their close proximity, information should flow between them relatively quickly, on average. However, if the bridge node is hidden from us, we may erroneously conclude these two nodes are actually quite far apart (illustrated in the figure by the red path). We would then expect information flow should be slow between them, even for information originating from other parties, and we would be surprised by the speed of flow we actually observe. If we consistently overestimate the time it takes for information to appear at one node after it appears at the other, then this provides evidence that a hidden presence in the network is facilitating the flow of information.

2 Methods

Here we describe the network topologies we will employ in this study, the details of how we simulate information flow on these topologies, the predictive models we generate for the flow times, and the test procedure and measurements we use to explore how well hidden nodes can be detected.

2.1 Network model

To gauge the potential of our approach, we first developed it against simulated transactional networks. We used scale-free networks generated according to the common preferential attachment model [5, 6, 2]. Each scale-free network was grown to a size of $N = 250$ nodes by adding new nodes one at a time, and each new node attached to two existing nodes preferentially according to their degree [2]. These undirected networks have a power-law degree distribution $\Pr(k) \sim k^{-3}$. The earlier a node is added to the network, the higher the degree it will tend to have. So *hubs*, highly connected nodes, tend to be among the early nodes of the network.

2.2 Transaction dynamics

For each network that was constructed, we simulated transactions, the creation and movement of **packets** of content, occurring between pairs of connected nodes. Each packet carries a unique identifier so that it can be tracked when it appears at different nodes in the network, and each node maintains a growing, time-ordered list of the content packets it has received. We simulated transactions as follows. At each time step, each node is activated with probability $p = 1$. This may represent a member of an online social network logging into their account, or a node in a computer network being turned on. For each node that is activated, it **creates** a new piece of content with probability $p_{\text{create}} = 1/9$ or **imports** a piece of content from a neighbor with probability $p_{\text{import}} = 2p_{\text{create}}$. In the former case, this may correspond to a member of the social network Twitter creating a new tweet; in the latter case it may correspond to them "retweeting" a tweet from someone they follow. The above probabilities were chosen to plausibly model the relative frequencies of creating versus importing content; an experimenter may equally estimate their values from a real dataset.

If a node i chooses to create a new packet, a new ID is generated and that packet is added to i's list of content. Neighbors of i may later choose to import this new packet into their own content lists, letting it spread throughout the network. Importing works as follows. If node i chooses to import content, one of i's neighboring nodes j is selected at random (assuming it has neighbors). Once j is selected, the information packets in j's list are scanned from most recently generated (or imported) to earliest generated (or imported). The scan stops when an information packet is found that is not contained in i's list. If no such packet can be found, no action for node i is taken and the next activated node is considered. If such a packet is found, it is copied from node j to node i.

This process is repeated for the next node that has been activated during the current time step. The simulation of transactions halts when 3000 time steps elapse. With the chosen values of p_{create} and p_{import}, each node will on average participate in the transaction model 1000 times. To avoid any pathological effects the nodes are activated in randomized order for each time step.

Once the transactions have been simulated and we have a timeline of packets for each node, we then compute the average time it takes for packets to flow between nodes. For every pair of nodes in a graph, we computed the intersection between their respective sets of information packets. We thus obtained each packet that both nodes in a pair either imported or created. We then computed the average time T_{ij} required for packets to travel between nodes i and j

$$T_{ij} = \sum_{k=1}^{n_{ij}} \left| t_i^{(k)} - t_j^{(k)} \right|, \tag{1}$$

where n_{ij} packets are shared by nodes i and j, and $t_i^{(k)}$ indicates the time step at which packet k was created (or arrived) at node i. In order to remove noise resulting from small sample sizes, all T_{ij} for which $n_{ij} < 100$ were discarded. Note that we are not measuring a causal or directional relationship between the node pair; a shared packet could easily have been created by a third node and then eventually reached both i and j through the importing process. The delay time T_{ij} is a dynamical measure of closeness between the nodes.

2.3 Symbolic Regression

Given a network topology and the information flow times T_{ij} (Eq. (1)), we then constructed a matrix D to serve as the dataset for training models to predict T_{ij} as a function of the structural properties of nodes i and j. Each pair of nodes is allocated its own row. One column in D contains the T_{ij} values, while the remaining columns correspond to structural network properties of node i, node j, or some metric relating them. An experimenter is free to choose which metrics to use. The individual node properties we used were node degrees k_i, k_j; clustering coefficients c_i, c_j; eccentricities e_i, e_j; node betweennesses B_i, B_j; eigenvector centralities x_i, x_j, where x_i is the i-th element of the leading eigenvector of the network's adjacency matrix; and closeness centralities C_i, C_j. For node-pair properties we used the length L_{ij} of the shortest topological path between i and j. Finally, we included global network properties N, the number of nodes; M, the number of edges; r, the degree-mixing assortativity coefficient [24]; and the graph's diameter Δ and radius ρ. These global quantities were the same for all rows of D, but providing them gives the optimization method a set of plausible constants to choose from.[1]

We then perform symbolic regression (SR) on this dataset to find functions f that predict T_{ij} as a function of the

[1]These can also become variables if one chooses to apply SR to a dataset containing multiple networks of different sizes.

node pair's structural properties:

$$
\begin{aligned}
T_{ij} = f(&k_i, c_i, e_i, B_i, x_i, C_i,\ k_j, c_j, e_j, B_j, x_j, C_j, \\
&L_{ij}, N, M, r, \Delta, \rho).
\end{aligned} \tag{2}
$$

Symbolic regression performs model selection and parameter estimation simultaneously to determine the functional form of Eq. (2). A commonly-employed method for instantiating symbolic regression is genetic programming [25], a stochastic optimization method that simultaneously optimizes a population of equations to increasingly fit the supplied data matrix D. As the name implies, this method is loosely based on Darwinian evolution. An initial population of random equations are assessed against D: models with high error are discarded, while models with lower error are retained. The now-vacant slots in the population are filled by repeatedly copying and mutating a single equation, or producing two new equations by performing sexual recombination with a pair of surviving equations. Mutations involve adding, removing, or altering a term in the equation.

The SR implementation we used in this study incorporates multiobjective optimization to perform search [26, 27]. The errors and sizes of the models in the population are computed. Size is defined as the total number of operators and operands in the equation. The Pareto front of models with least error and smallest size is determined, and models off this front are discarded. New models are generated by randomly choosing surviving models on the front. When run against a dataset generated by a single scale-free network composed of $N = 250$ nodes, the best equation found[2], in terms of balancing complexity and accuracy, was

$$
T_{ij} = L_{ij}\left[1 + \ln\left(L_{ij} + k_i + k_j - c_j + \frac{N - k_i^2 - k_j^2}{L_{ij}^N + k_i k_j - \rho}\right)\right]. \tag{3}
$$

This equation achieved a high correlation coefficient of $R = 0.88$ when compared with the simulated T_{ij}. We remark that Eq. (3) seems plausible in nature: the dominant variable is the distance L_{ij} between i and j, which is intuitive for the transaction model. The degrees of i and j, the clustering of j and global network properties N and the network radius then comprise a small, logarithmic correction to L_{ij}. Other variables did not factor into this function.

2.4 Tampered networks

To test the ability of the SR model to indicate the presence of a hidden node, we need access to a ground truth test bed. To create such a test using our model networks (Sec. 2.1), we generate a new scale-free network, simulate transactions on it (Sec. 2.2), then choose one or more nodes to **hide**; they are removed before computing the network structural

[2]Note that SR was prevented from using numerical prefactors to enforce greater structural diversity in models along the Pareto front.

metrics and the information flow times (Eq. (1)). In this way hidden nodes fully participate in the flow of packets, but otherwise they are unknown to the symbolically regressed model. Comparing the SR model's delay predictions (Eq. (3)) to the simulated delay times, we can *a posteriori* search for systematic errors among the neighbors of the hidden node or nodes.

To measure the effects of the hidden node we study three quantities. The first is the coefficient of determination R^2 between the T_{ij}'s measured for the non-hidden node-pairs from the transaction simulations and the predicted T_{ij}'s from the SR model, where R is the Pearson correlation coefficient. If the value of R^2 drops significantly compared to R^2 for the untampered network, then that supports the ability for us to detect missing or hidden nodes.

Beyond this global measure we also use two local measurements to assess the effect a hidden node has on a single non-hidden node i:

$$\mathrm{RMSE}_i = \sqrt{\mathbb{E}_j\left[\left(T_{ij}^{\mathrm{pred}} - T_{ij}^{\mathrm{obs}}\right)^2\right]}, \tag{4}$$

$$\mathrm{Bias}_i = \mathbb{E}_j\left[T_{ij}^{\mathrm{pred}} - T_{ij}^{\mathrm{obs}}\right], \tag{5}$$

where the expectation $\mathbb{E}_j\left[\cdot\right]$ runs over all (non-hidden) nodes $j \neq i$ that are connected to i ($L_{ij} < \infty$), and T_{ij}^{pred} and T_{ij}^{obs} denote the flow time predicted by the SR model and the actual flow time observed from the simulations, respectively. The RMSE_i captures the magnitude of the SR model's error for node i, while Bias_i measures whether it consistently over- or under-estimated T_{ij}. A positive bias indicates that information is traveling faster than expected by the SR model.

3 Results

Our first experiment consisted of measuring the change in the coefficient of determination R^2 for tampered scale-free networks (Sec. 2.4). To do this we first generated an ensemble of 100 untampered scale-free networks (Sec. 2.1) and simulated transactions on each (Sec. 2.2). We applied the SR model of T_{ij} to these networks (Eq. (3)) and computed R^2 for each. As shown in Fig. 2A, the distribution of R^2 was sharply peaked around $R^2 \approx 0.77$, the value that the SR model achieved on its training data (Sec. 2.3). The narrowness of this distribution indicates that the SR model has useful predictive power.

Next we generated another ensemble of scale-free networks and simulated transactions, but now we tampered with each network by hiding one random hub[3]. We see a significant drop in accuracy (lower R^2) for the SR model on these

[3]We take a hub to be a randomly chosen node that was introduced in the first 20% of the network growth process, taking advantage of preferential attachment's early-mover-advantage.

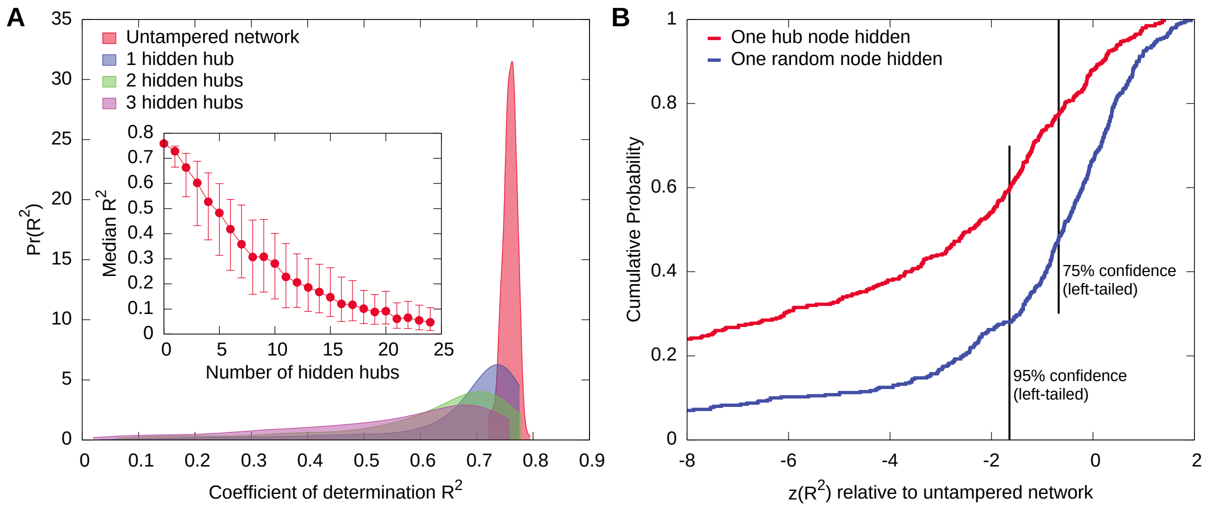


Figure 2: Detecting hidden nodes in scale-free networks. (**A**) The distribution of correlation coefficient comparing the simulated transaction times and those predicted by the symbolically regressed model. We see for networks drawn from the same ensemble as the training network (untampered) that R^2 is sharply peaked around 0.78. The distribution of R^2 changes significantly after a single node removal (Mann-Whitney U test $p < 10^{-10}$, Cohen's $d = -0.898$). (inset) The median R^2 decreases as more high-degree nodes are removed. (**B**) The likelihood of detecting a single hidden node by the change in R^2. Using the distribution of R^2 for the untampered network as the null model, we standardize the R^2 distribution for networks with one hidden node. Looking at this distribution we see that nearly 60% of the time we can successfully detect that a relatively high degree node is absent with 95% confidence. If we consider lower degree hidden nodes, which are more challenging to discover as they tend to participate less in information flow, this drops to approximately 25%. Distributions shown in panel A were computed using kernel density estimation; each curve was truncated at the largest value observed in the ensemble data to indicate the empirical ranges of R^2.

tampered networks (Mann-Whitney U test $p < 10^{-10}$, Cohen's $d = -0.898$), indicating that we are likely to see the effect of a hidden node by a drop in the accuracy of the model. Hiding multiple hubs leads to even greater losses in accuracy (Fig. 2A and inset).

However, the comparisons shown in Fig. 2A are for an ensemble of networks, while practically we seek to detect the presence of a missing node in a *single* network. To determine if this is feasible we standardized the distribution of R^2 for the ensemble of networks with a single hidden hub relative to the untampered ensemble, giving a z-score $z(R^2)$ for each tampered network. Large negative values of z indicate a statistically significant drop in R^2. The cumulative probability distribution shown in Fig. 2B tells us that nearly 60% of the tampered ensemble has $z(R^2) < -1.6449$, meaning that nearly 60% of the time we can determine with 95% confidence that a single network is missing a hub. The 50% confidence limit, $z < 0$, corresponding to how well we can beat a coin-flip, is nearly 90%.

These results indicate that the presence of a single hidden node can often be detected. An important question, however, is whether or not we can identify the *location* of this hidden node. To study this, we computed the errors and biases (Eqs. (4) and (5)) of each node in a tampered network. If the neighbors of the hidden node show significant error or bias, then that means we can determine the location of the hidden node. We show a network diagram of one tampered scale-free network in Fig. 3A. Node color and size is proportional to RMSE and the hidden node is indicated with a diamond ($\diamond$). We observe that many neighbors of the hidden node have far greater RMSE than other nodes in the network. This is exactly the evidence needed to estimate the hidden node's location within the network topology.

To determine if these results are significant, we computed, for the ensemble of scale-free networks with one hidden hub, distributions of RMSE and Bias separately for neighbors of the missing hub, next-nearest neighbors, and other nodes. The RMSE was significantly larger (Mann-Whitney U test $p \ll 10^{-10}$, Cohen's $d = 4.69$) for neighbors of the hidden node across the entire ensemble (the median error for neighbors was ≈ 4.33 timesteps compared with 1.1 timesteps for other nodes). Next-nearest neighbors, those nodes two steps away from the hidden node in the original topology, did not show a significant change in error relative to other nodes in the network ($p = 0.052$). However, a number of outliers do overlap with the RMSE values for the nearest neighbors, indicating that longer-range network effects are rare but do occur.

At the same time, the Bias was also positively skewed for neighbors of the hidden node (median Bias ≈ 2.1 timesteps), indicating that our intuition from Fig. 1 was correct. Next-nearest neighbors have no discernible bias (median Bias ≈ 0.03), while other nodes actually have a slightly negative bias (median Bias ≈ -0.18), indicating the information in the rest of the network actually travels slightly slower than expected due to the hidden node (however, a zero bias cannot be ruled out for this group).

4 Discussion

We have shown that the presence of hidden nodes can be inferred by modeling how network topology influences a dynamical process overlaying that network. We focused on an idealized information flow dynamics but there is great potential for applying this to other model dynamics. For future work, we intend to use our methodology alongside real world data on information cascades and other dynamical processes and to further study how different classes of network topologies help or hinder the node discovery process. We also plan to better incorporate the *directionality* of information flow, which was neglected here by the absolute value used in the equation for T_{ij}.

It is not particularly surprising that perturbing a network, which then leads to perturbed metrics such as those used in Eq. 2, will lead to a reduction in the accuracy of an SR model (e.g., Eq. 3). This was shown in Fig. 2. However, we have shown (Fig. 3) that the loss in accuracy is **localized** and correlates with the position of the defect, indicating that we are extracting useful information and not merely randomizing the terms within the SR model's functional form.

More generally, looking for discrepancies in the speed of information flow (or other quantities) can be used to study not just missing nodes but other defects and errors, such as missing links or false links that incorrectly appear

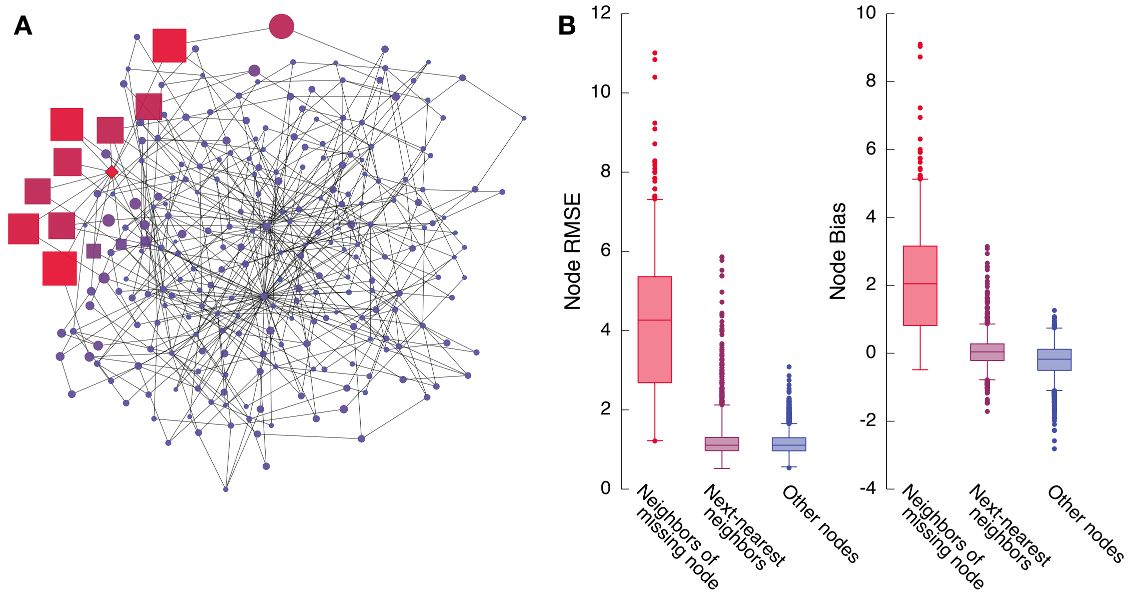


Figure 3: Identifying the location of a missing node. (**A**) A scale-free network of 250 nodes with a single node hidden (⋄). The neighbors of the hidden node are indicated with □ while other nodes are ∘. The size and color of each node is proportional to the rms error of the information transfer time from that node to every other node in the network. We see that the neighbors of the missing node consistently have higher errors than the rest of the network. (**B**) The distributions of error and bias across the ensemble of tampered networks for the hidden node's neighbors, next-nearest neighbors, and non-neighbors. The median error for neighbors is approximately 4.33 timesteps while for non-neighbors it is approximately 1.11 timesteps. The distributions are significantly different (Mann-Whitney U test $p \ll 10^{-10}$, Cohen's $d = 4.69$). The next-nearest neighbors have errors comparable to non-neighbors ($p = 0.052$) but we see a greater number of outliers skewing upward. This indicates that there are some network effects in how errors propagate, but they are relatively rare. Likewise, we see positive bias for neighbor nodes, significantly higher than for non-neighbors (Mann-Whitney U test $p \ll 10^{-10}$, Cohen's $d = 3.37$). This positive bias indicates that information spreads faster from (or to) neighbors of the hidden node than the SR model expects, supporting the intuition behind Fig. 1 To control for the centrality of the hidden node, in each realization the hidden node was the node with the fifth highest degree.

in the network, false nodes that do not actually exist, the splitting of a true node into multiple false nodes, or the merging of multiple true nodes into a single false node. Some of these errors will likely prove more challenging to detect than others, but the benchmarking procedure we have introduced here may offer some hope towards tackling these problems.

Acknowledgments

The authors are grateful for the computational resources provided by the Vermont Advanced Computing Core which is supported by NASA (NNX 08A096G). JCB was supported in part by NSF grant #PECASE–0953837 and DARPA grant #W911NF–11–1–0076. JCB and CMD were supported in part by DARPA grant #FA8650–11–1–7155. PSD was supported by NSF CAREER Award #0846668.

References

[1] Duncan J. Watts and S. J. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

[2] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–511, 1999.

[3] M. E. J. Newman. The structure and function of complex networks. *SIAM Rev.*, 45(2):167–256, 2003.

[4] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang. Complex networks: Structure and dynamics. *Physics Reports*, 424:175–308, 2006.

[5] H. A. Simon. On a class of skew distribution functions. *Biometrika*, 42:425–440, 1955.

[6] D. J. de Solla Price. Networks of scientific papers. *Science*, 149:510–515, 1965.

[7] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58:1019–1031, 2007.

[8] A. Clauset, C. Moore, and M. E. J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, May 2008.

[9] D. Wang, D. Pedreschi, C. Song, F. Giannotti, and A.-L. Barabási. Human mobility, social ties, and link prediction. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, page 11001108, New York, NY, USA, 2011. ACM.

[10] C. A. Bliss, M. R. Frank, Danforth C. M., and P. S. Dodds. An evolutionary algorithm approach to link prediction in dynamic social networks. `http://arxiv.org/abs/1304.6257`, 2013.

[11] G. Kossinets. Effects of missing data in social networks. *Social Networks*, 28(3):247–268, 2006.

[12] S.P. Borgatti, K.M. Carley, and D. Krackhardt. On the robustness of centrality measures under conditions of imperfect data. *Social Networks*, 28:124–136, 2006.

[13] E. de Silva, T. Thorne, P. Ingram, I. Agrafioti, J. Swire, C. Wiuf, and M. Stumpf. The effects of incomplete protein interaction data on structural and evolutionary inferences. *BMC Biology*, 4:39, 2006.

[14] Y. Maeno. Node discovery problem for a social network. *Connections*, 9:62–76, 2009.

[15] Y. Maeno. Discovering covert node in networked organization. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 2009.

[16] J. P. Bagrow, D. Wang, and A.-L. Barabási. Collective response of human populations to large-scale emergencies. *PLoS ONE*, 6(3):e17680, March 2011.

[17] Peter Sheridan Dodds, Kameron Decker Harris, Isabel M. Kloumann, Catherine A. Bliss, and Christopher M. Danforth. Temporal patterns of happiness and information in a global social network: Hedonometrics and twitter. *PLoS ONE*, 6(12):e26752, December 2011.

[18] O. Woolley-Meza, D. Grady, C. Thiemann, J. P. Bagrow, and D. Brockmann. Eyjafjallajkull and 9/11: The impact of large-scale disasters on worldwide mobility. *PLoS ONE*, 8(8):e69829, August 2013.

[19] L. Mitchell, M. R. Frank, K. D. Harris, P. S. Dodds, and C. M. Danforth. The geography of happiness: Connecting twitter sentiment and expression, demographics, and objective characteristics of place. *PLoS ONE*, 8(5):e64417, May 2013.

[20] J. H. Fowler and N. A. Christakis. Dynamic spread of happiness in a large social network: longitudinal analysis over 20 years in the Framingham Heart Study. *BMJ*, 337:article #2338, 2008.

[21] N. A. Christakis and J. H. Fowler. The spread of obesity in a large social network over 32 years. *New England Journal of Medicine*, 357:370–379, 2007.

[22] J. T. Cacioppo, J. H. Fowler, and N. A. Christakis. Alone in the crowd: The structure and spread of loneliness in a large social network. *Journal of Personality and Social Psychology*, 97:977–991, 2009.

[23] C. R. Shalizi and Andrew C. Thomas. Homophily and contagion are generically confounded in observational social network studies. *Sociological Methods & Research*, 40(2):211–239, May 2011.

[24] M. E. J. Newman. Mixing patterns in networks. *Physical Review E*, 67(2):026126, February 2003.

[25] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[26] J. Bongard and H. Lipson. Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 104(24):9943–9948, June 2007. PMID: 17553966.

[27] M. Schmidt and H. Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, April 2009. PMID: 19342586.

3 Symbolically regressing brain imaging data.

Note: All of the work on neuroimaging data conducted as part of this award used data collected by a separate group—the IMAGEN consortium—before the award commenced. None of the members of our group were involved in this data collection.

Brain dynamics are immensely complex across space and time and physiological subsystem. Thus, it is extremely unlikely that any assumptions can be made *a priori* about the structure of these dynamics. We thus have adapted symbolic regression for modeling data generated from functional MRI data sets: we induced models that predict behavior as a function of brain dynamics. In the manuscript that follows [1] we show that there are many previously unknown nonlinear relationships between brain regions that can be predictive of behavior.

Furthermore, we show that such relationships can, at least in the specific conditions investigated here, consistently predict behavioral tendencies. The method, in brief, was a hybrid method that employs symbolic regression for feature construction (it finds nonlinear terms that are weakly predictive of the outcome of interest) and more traditional regression methods for feature selection (optimizing coefficients for those terms).

Many of the models found in this way concord with those discovered using orthogonal methods, providing some confidence in the other models found by our method. Most notably, symbolic regression found that differences in the nonlinear relationships between brain regions implicated in thist, reward, and emotion accurately predict whether the adolescent participants from whom the data was drawn regularly consume alcohol or not.

Two additional manuscripts are in preparation. The first is an attempt to predict a different behavioral tendency (smoking or non-smoking) using the same method. The second is an attempt to predict how an individual's competence degrades with adversity. Predictions about both behavioral traits is made directly from models trained against brain imaging data, again collected by groups not associated with this award.

In future, such methods could be used to rapidly predict other behavioral traits. More importantly, it may be able to predict behavioral traits that have yet to manifest in the participants, thus providing an opportunity for proactive treatment and/or counseling.

3.1 Relevance for U.S. defense and security.

Predicting the behavior of warfighters in the field is of extreme interest, especially before they have been delivered into a theater of war. Similarly, it would be of extreme utility to assess the mental state of warfighters during deployment without having to distract them with explicit requests for updates. The method outlined in the manuscript below could be adapted for inferring current behavior and/or future degradation in behavior directly from neuroimaging. Current MRI technologies rule out real-time assessment, but advances in mobile MRI, tensor diffusion imaging, and/or EEG may make such assessment tractable in the near future.

3.2 Allgaier et al. "Nonlinear functional mapping..." (2015).

A technical manuscript describing the symbolic regression of neuroimaging and behavioral data sets follows.

Nonlinear functional mapping of the human brain

Allgaier, N.A.[a,b,c,d,*], Banaschewski, T.[e], Barker, G.J.[f], Bokde, A.L.W.[g], Bongard, J.C.[b,c], Bromberg, U[h], Büchel, C.[h], Cattrell, A.[i], Conrod, P.[j,k], Danforth, C.M.[a,b], Desrivières, S.[i], Dodds, P.S.[a,b], Flor, H.[l], Frouin, V.[m], Gallinat, J.[n], Gowland, P.[o], Heinz, A.[p], Ittermann, B.[q], Mackey, S.[d], Martinot, J-L.[s], Murphy, K.[r], Nees, F.[l], Papadopoulos-Orfanos, D.[m], Poustka, L.[e,t], Smolka, M.N.[u], Walter, H.[p], Whelan, R.[v], Schumann, G.[i], Garavan, H.[d], IMAGEN Consortium

[a]*Department of Mathematics and Statistics, University of Vermont, Burlington, Vermont, USA*
[b]*Vermont Complex Systems Center, Vermont Advanced Computing Core, Burlington, Vermont, USA*
[c]*Department of Computer Science, University of Vermont, Burlington, Vermont, USA*
[d]*Departments of Psychiatry and Psychology, University of Vermont, 05405 Burlington, Vermont, USA*
[e]*Department of Child and Adolescent Psychiatry and Psychotherapy, Central Institute of Mental Health, Medical Faculty Mannheim, Heidelberg University, Square J5, 68159 Mannheim, Germany*
[f]*Centre for Neuroimaging Sciences, Institute of Psychiatry, Psychology & Neuroscience, Kings College London, United Kingdom*
[g]*Discipline of Psychiatry, School of Medicine and Trinity College Institute of Neurosciences, Trinity College Dublin, Ireland*
[h]*University Medical Centre Hamburg-Eppendorf, House W34, 3.OG, Martinistr. 52, 20246, Hamburg, Germany*
[i]*Medical Research Council - Social, Genetic and Developmental Psychiatry Centre, Institute of Psychiatry, Psychology & Neuroscience, Kings College London, United Kingdom*
[j]*Department of Psychiatry, Université de Montreal, CHU Ste Justine Hospital, Canada*
[k]*Department of Psychological Medicine and Psychiatry, Institute of Psychiatry, Psychology & Neuroscience, King's College London*
[l]*Department of Cognitive and Clinical Neuroscience, Central Institute of Mental Health, Medical Faculty Mannheim, Heidelberg University, Square J5, Mannheim, Germany*
[m]*Neurospin, Commissariat à l'Energie Atomique, CEA-Saclay Center, Paris, France*
[n]*Department of Psychiatry and Psychotherapy, University Medical Center Hamburg-Eppendorf (UKE), Martinistrasse 52, 20246 Hamburg*
[o]*Sir Peter Mansfield Imaging Centre School of Physics & Astronomy, University of Nottingham, University Park, United Kingdom*
[p]*Department of Psychiatry and Psychotherapy, Campus Charité Mitte, Charité, Universitätsmedizin Berlin, Germany*
[q]*Physikalisch-Technische Bundesanstalt (PTB), Braunschweig and Berlin, Germany*
[r]*Cardiff University Brain Research Imaging Centre (CUBRIC), School of Psychology, Cardiff University, Cardiff, United Kingdom*
[s]*Institut National de la Santé et de la Recherche Médicale, INSERM Unit 1000 Neuroimaging & Psychiatry, University Paris Sud, University Paris Descartes - Sorbonne Paris Cité; and Maison de Solenn, Paris, France*
[t]*Department of Child and Adolescent Psychiatry and Psychotherapy, Medical University of Vienna, Austria*
[u]*Department of Psychiatry and Neuroimaging Center, Technische Universität Dresden, Dresden, Germany*
[v]*Department of Psychology, University College Dublin*

Abstract

The field of neuroimaging has truly become data rich, and novel analytical methods capable of gleaning meaningful information from large stores of imaging data are in high demand. Those methods that might also be applicable on the level of individual subjects, and thus potentially useful clinically, are of special interest. In the present study, we introduce just such a method, called *nonlinear functional mapping* (NFM), and demonstrate its application in the analysis of resting state fMRI (functional Magnetic Resonance Imaging) from a 242-subject subset of the IMAGEN project, a European study of adolescents that includes longitudinal phenotypic, behavioral, genetic, and neuroimaging data. NFM employs a computational technique inspired by biological evolution to discover and mathematically characterize interactions among ROI (regions of interest), without making linear or univariate assumptions. We show that statistics of the resulting interaction relationships comport with recent independent work, constituting a preliminary cross-validation. Furthermore, nonlinear terms are ubiquitous in the models generated by NFM, suggesting that some of the interactions characterized here are not discoverable by standard linear methods of analysis. We discuss one such nonlinear interaction in the context of a direct comparison with a procedure involving pairwise correlation, designed to be an analogous linear version of functional mapping. We find another such interaction that suggests a novel distinction in brain function between drinking and non-drinking adolescents: a tighter coupling of ROI associated with emotion, reward, and interoceptive processes such as thirst, among drinkers. Finally, we outline many improvements and extensions of the methodology to reduce computational expense, complement other analytical tools like graph-theoretic analysis, and allow for voxel level NFM to eliminate the necessity of ROI selection.

Keywords:
resting state fMRI, modeling, nonlinear, machine learning, genetic programming, symbolic regression

1. Introduction

Many advances in our understanding of brain function have been achieved through analysis of fMRI data. Though the BOLD (blood oxygen level dependent) signal obtained from fMRI is a proxy, physiological confounds such as breathing and heart rate are separable from neuronal-induced signal, as demonstrated in Birn et al. (2009). Inter-subject differences in vascular reactivity can be modeled as shown in Murphy et al. (2011), and BOLD has been directly shown to provide a reliable measure of neuronal activity in specific circumstances, as in Mukamel et al. (2005). The many years of successful research before and since support that assessment. Accomplishments include localization of regions responsible for particular tasks, such as episodic memory in Nolde et al. (1998) and human face recognition in Kanwisher et al. (1999), assessment of the risk of postoperative motor defect in patients with tumors in Mueller et al. (1996), analysis of the effects of acupuncture in Hui et al. (2000), and recently, identification of neural markers for both current *and future* alcohol use among adolescents in Whelan et al. (2012) and Whelan et al. (2014).

These examples, and indeed the majority of fMRI studies, make use of the GLM (general linear model) to determine neural correlates for various tasks and stimulus responses. Though typical analyses have been performed at the group level with a univariate approach, other recent work reported in Rio et al. (2013) has extended the capabilities of the GLM to analyze multivariate signal in the Fourier domain to reduce confounds from time-correlated noise, thus improving the suitability of the GLM for subject level analysis. Despite these advances, however, the GLM can only *confirm* hypothesized nonlinear models of function, not *discover* them.

Group-level inferences from fMRI have also been performed using linear ICA (independent component analysis), as described in Calhoun et al. (2001). Though ICA and the GLM can be used in conjunction, for example in Liu et al. (2010) to investigate the neural effects of stimulation of a particular acupoint, ICA is particularly useful in circumstances that preclude the use of the GLM, such as the analysis of resting-state data, for which there is no task or stimulus regressor. Covarying networks have been suggested by ICA of resting-state fMRI in Smith et al. (2009), and functional, hierarchical classification of these networks has been automated through HCA (hierarchical cluster analysis) of aggregated experimental metadata in Laird et al. (2011). However, it was determined early on, for example in McKeown and Sejnowski (1998), that nonlinear interactions within the brain need to be addressed in order to properly determine functional architecture.

Although ICA algorithms that employ nonlinear mixing functions exist, severe restrictions on those functions are required to avoid non-uniqueness of solutions, as explained in Hyvärinen and Pajunen (1999). Due to this failing, other methodologies have been employed in the attempt to account for nonlinearity. Examples include various forms of nonlinear regression, as in Kruggel et al. (2000), and dynamic causal modelling, as described in Friston et al. (2003). In each of these, a particular nonlinear form must be posited *a priori*, and thus the capability to *discover* previously unknown nonlinear interactions within the brain is diminished. As a result, a fuller picture of the nature of intra- and inter-network functional connectivity within the brain is missing from the literature.

Here we introduce a methodology designed to accomplish such a mathematical characterization, provide insight at the group, subject, and ROI levels, and to avoid linear and univariate assumptions. With some modification, analysis of higher dimensional data is likely attainable, allowing for eventual application at the voxel scale and eliminating the necessity of ROI selection. After standard preprocessing (slice-timing and motion correction, normalization, smoothing, etc.), our procedure consists of ROI selection, inter-ROI *symbolic regression* (a model-free form of nonlinear regression), accomplished by an evolutionary algorithm called *genetic programming* (GP; a form of stochastic optimization), and statistical analysis of the resulting models. We demonstrate our technique on a 242-subject collection of resting-state data from the IMAGEN project, though analysis of task or stimulus experiments can be accomplished with little or no modification. The IMAGEN project is described in detail in Schumann et al. (2010).

We organize the paper as follows. In Section 2, we discuss the data and selection of ROI, provide some background on GP, and describe the procedural details of NFM by symbolic regression. In Section 3, we report results of applying the technique to the IMAGEN data, including statistical and hierarchical visualizations, comparison with previous results for cross-validation, effects of nonlinearity, and an example of group-level variation. We discuss the results and potential applications of the technique in Section 4, and conclude the paper in Section 5.

2. Materials and methods

In this section, we first briefly describe the source of the data for our study, and then provide the details of ROI selection that allow for comparison with recent work. We then provide some background on the GP algorithm in general and the specific implementation employed here, along with the method by which it is applied to BOLD signal time series extracted from the selected ROI. Finally, we describe the statistical technique used to interpret the roughly quarter of a million mathematical models that result from the application of GP to all 52 ROI time series extracted from each of the 242 subjects.

*Corresponding author.
Email address: nicholas.allgaier@uvm.edu (Allgaier, N.A.)
URL: www.imagen-europe.com (IMAGEN Consortium)

2.1. Data

The data investigated here are a subset of the fMRI scans from the IMAGEN study, a European research project with the goal of better understanding teenage psychological and neurobiological development. The project is longitudinal, and utilizes several forms of high and low-tech experimental protocols including self-report questionnaires, behavioral assessment, interviews, neuroimaging, and blood sampling for genetic analyses. Each of the 2000 participating adolescents was 14 when entering the study, which itself commenced in late 2007, and data collection continues today.

More specifically, the data for the present study are 6-minute resting-state fMRI time series of 242 of the adolescent subjects who were asked to keep their eyes open while in the scanner, but were presented with no other task or stimulus. To allow for comparison with previous work, locations of the ROI were chosen based on results from Laird et al. (2011), in which statistical analysis across thousands of previous imaging studies (both stimulus/task-based and resting-state) was used to identify networks of brain regions that tend to activate together, termed ICN (intrinsic connectivity networks). The ICN were determined by ICA, from which z-statistic maps were derived. To select ROI for this study, a z-statistic threshold was set for each ICN to determine the number of regions in the network, and ROI were defined as rough spheres with radii of 3 voxels (9mm) and centered at the location of peak z-statistic in each region.

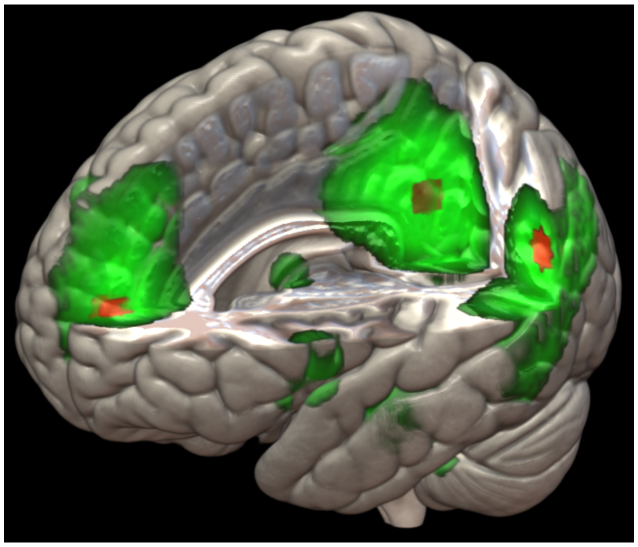


Figure 1: ROI Selection. (Red) ROI from within the default mode network, with radii of 3 voxels and centers corresponding to the highest z-statistics (green) in each region as determined in Laird et al. (2011).

We provide a cut-out illustrating ROI selection for the default mode network (ICN 13) in Figure 1, and Figure 2 contains axial cross sections showing many of the ROI derived from the 18 non-artifactual ICN in Laird et al. (2011). In Appendix A, Table A.1 we list all 52 ROI by number, give their anatomical names, indicate the ICN from within which they were defined, and provide visual representations of their locations within the brain.

Subsequent to ROI definition, a gray matter mask was applied to assure that only appropriate voxels were contained within each ROI. In some cases this resulted in a considerable reduction of ROI voxels, but the majority maintained the full complement of about 100 voxels. For each of the 242 subjects, time series were extracted from each of the 52 resulting ROI by averaging the BOLD signal over all voxels within the ROI. These time series then form the input to the GP algorithm.

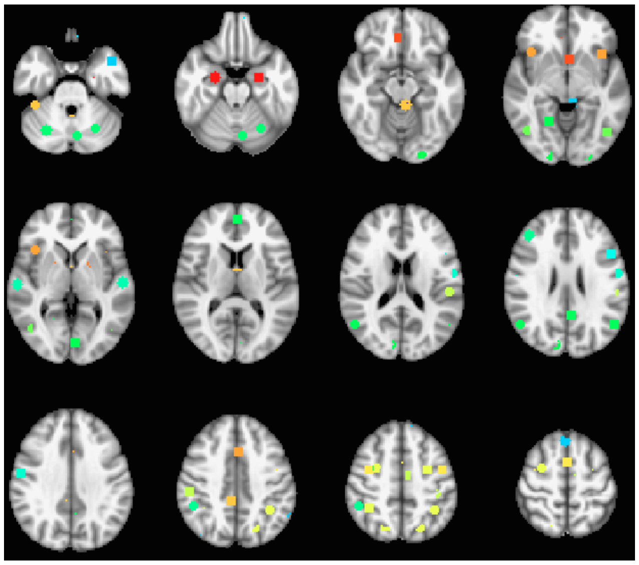
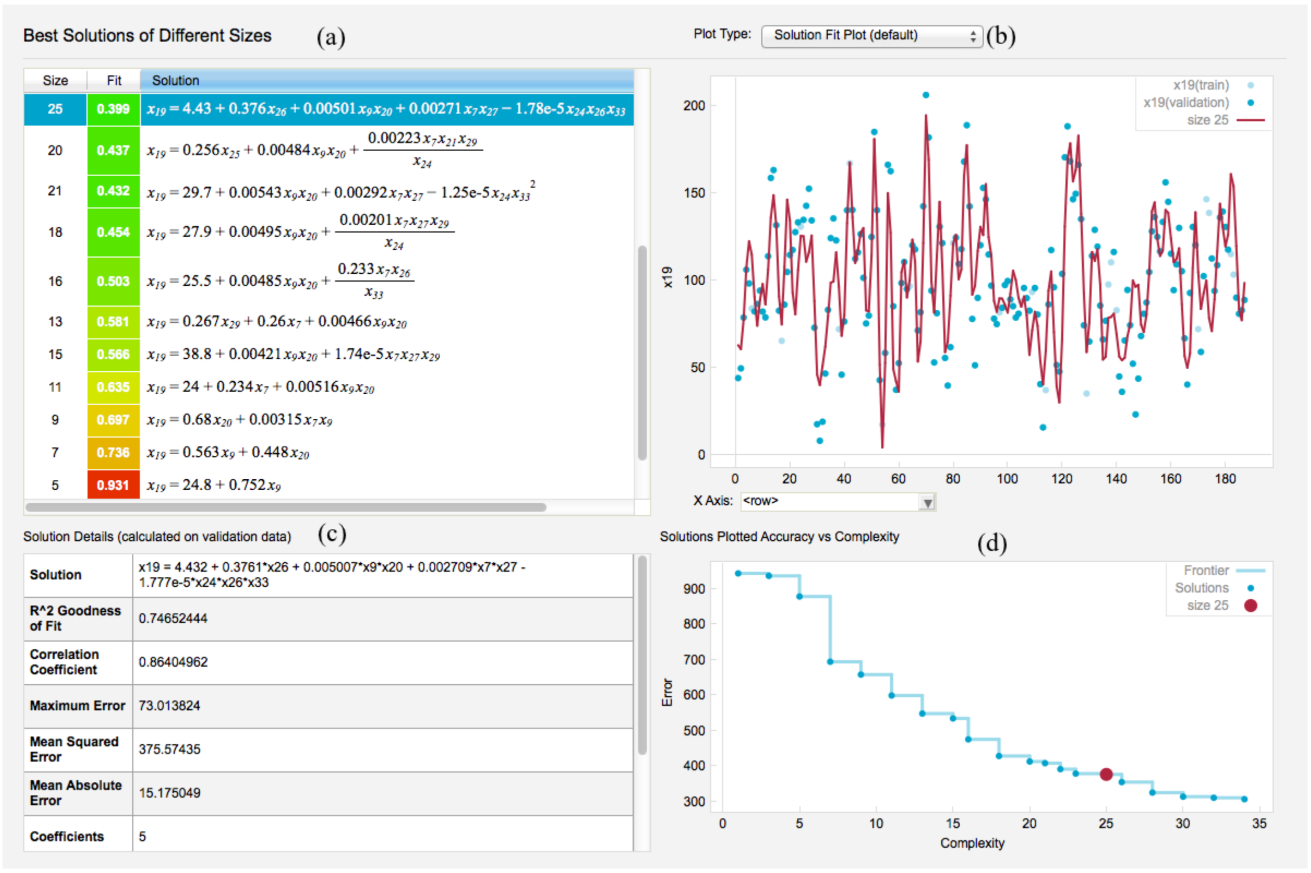


Figure 2: Visualization of ROI. Axial cross sections showing many of the ROI derived from the ICN in Laird et al. (2011).

2.2. Genetic programming

GP is a biologically inspired, population-based machine learning algorithm. It is most commonly employed for symbolic regression: the algorithm searches for models explaining some quantity of interest (e.g., average BOLD signal from an ROI in the brain) as a function of some other possibly related observable quantities, statistics, or summary data (e.g., BOLD signals from other ROI). The algorithm proceeds by evolving the functional forms of a population of potential models, which are initially constructed at random from user-specified mathematical building blocks (available variables, arithmetic functions, parameter constants, etc.). In brief, the models that better explain the data produce more offspring, leading to a gradual reduction of error within the population. We show a representative set of models produced by this approach in Figure 3(a). A key advantage of the technique is that no assumption (e.g., linearity) is imposed on the form of solutions, other than the choice of building blocks from which


Best Solutions of Different Sizes (a)

Size	Fit	Solution
25	0.399	$x_{19} = 4.43 + 0.376x_{26} + 0.00501x_9x_{20} + 0.00271x_7x_{27} - 1.78\text{e-}5x_{24}x_{26}x_{33}$
20	0.437	$x_{19} = 0.256x_{25} + 0.00484x_9x_{20} + \frac{0.00223x_7x_{21}x_{29}}{x_{24}}$
21	0.432	$x_{19} = 29.7 + 0.00543x_9x_{20} + 0.00292x_7x_{27} - 1.25\text{e-}5x_{24}x_{33}^2$
18	0.454	$x_{19} = 27.9 + 0.00495x_9x_{20} + \frac{0.00201x_7x_{27}x_{29}}{x_{24}}$
16	0.503	$x_{19} = 25.5 + 0.00485x_9x_{20} + \frac{0.233x_7x_{26}}{x_{33}}$
13	0.581	$x_{19} = 0.267x_{29} + 0.26x_7 + 0.00466x_9x_{20}$
15	0.566	$x_{19} = 38.8 + 0.00421x_9x_{20} + 1.74\text{e-}5x_7x_{27}x_{29}$
11	0.635	$x_{19} = 24 + 0.234x_7 + 0.00516x_9x_{20}$
9	0.697	$x_{19} = 0.68x_{20} + 0.00315x_7x_9$
7	0.736	$x_{19} = 0.563x_9 + 0.448x_{20}$
5	0.931	$x_{19} = 24.8 + 0.752x_9$

Solution Details (calculated on validation data) (c)

Solution	x19 = 4.432 + 0.3761*x26 + 0.005007*x9*x20 + 0.002709*x7*x27 - 1.777e-5*x24*x26*x33
R^2 Goodness of Fit	0.74652444
Correlation Coefficient	0.86404962
Maximum Error	73.013824
Mean Squared Error	375.57435
Mean Absolute Error	15.175049
Coefficients	5

Figure 3: Screen shot of the GP package Eureqa during a search for models of the activity in ROI 19 in a single subject, as a function of activity in the other 51 regions. (a) The current set of models along the Pareto front of accuracy vs. parsimony, shown in (d) where each point represents a model and the red point represents the highlighted model. (b) Data from ROI 19 (points) over the 6-minute time series for this subject (x-axis in scans, 2 seconds each). The highlighted model is shown in red, and statistics for this model's fit appear in (c).

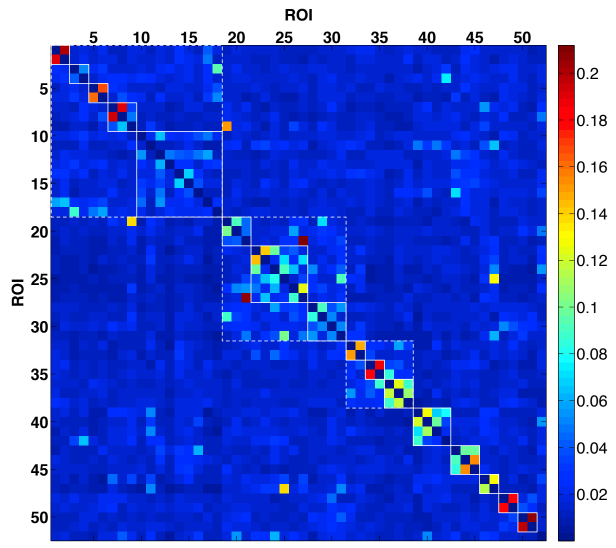
they can be made (we use arithmetic operations in the present work).

Typically, some measure of error (e.g., root mean square error) constitutes a model's explanatory fitness, and some measure of its size (e.g., number of operators, constants, and variables in the equation) represents its parsimony. The next generation of potential models is obtained by mutation (e.g., a single change of variable or operator) and recombination (i.e., swapping of function components between models) of the current set of *non-dominated* solutions: those models for which no simpler model in the population has less error. This set of non-dominated models is said to approach the ideal *Pareto front* of fitness versus parsimony as the population evolves. An important aspect of GP is that the result of a single search is this entire set of potential models, providing a trove of information for statistical analysis. Figure 3 is a screenshot of the off-the-shelf GP package Eureqa from Schmidt and Lipson (2009) performing a search (Eureqa version 0.97 Beta was used to generate the results reported in this study).

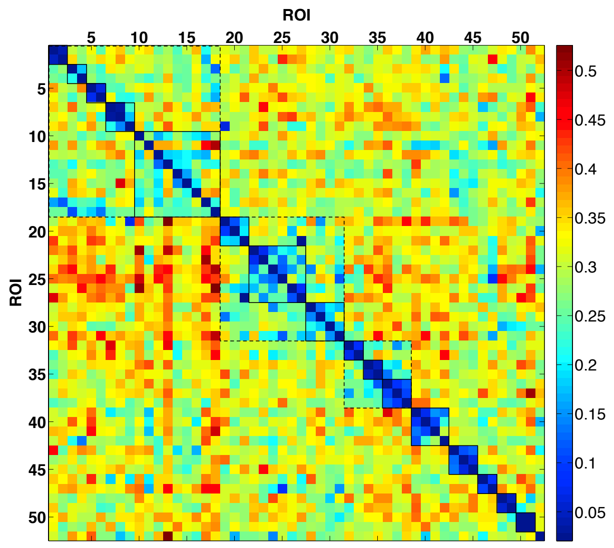
To apply GP to the fMRI data, for each of the 242 subjects we extract a single BOLD signal time series from each of the 52 selected ROI by averaging over the voxels within that ROI. Then the GP algorithm is run 52 times, one for each ROI, using all other ROI as potential explanatory variables. Note that the algorithm has no knowledge of the hypothesized networks from which these regions were chosen.

We describe the computational expense of the algorithm in terms of core-hours, i.e., the number of hours required for a single processor core to perform the necessary computation. Specifically, twelve core-hours of search were performed for each region, amounting to 624 core-hours per subject, and over 17 total core-years of computation were required for the population of 242 subjects. This yielded roughly 12 thousand Pareto fronts comprised of a quarter million models for statistical analysis.

The results of this analysis characterize the entire population of 242 subjects. Alternatively, results can be aggregated over phenotypic groups to produce group-level characterizations, or many GP searches can be run for a single individual to produce a subject-level characterization. We report results of population- and group-level analyses in Section 3, and discuss an example subject-level



(a) Interaction map



(b) Subsampling RSD

Figure 4: Functional interaction map. (a) Interaction map across all 242 subjects, and (b) map of RSD (relative standard deviation) of the interaction rates over 100 subsamples with 100 randomly selected subjects each. Solid outlines indicate ICN and dashed outlines indicate functional groupings of ICN from Laird et al. (2011).

analysis in Appendix B.

2.3. *Analysis*

The output of the GP algorithm poses a challenge for interpretation. Here we present a coarse statistical analysis of this rich mathematical characterization. For each ROI, we count the number of models for that ROI, across the Pareto fronts for all 242 subjects, that have a particular (other) region on the right-hand side of the equation. We compute this count for each of the other 51 regions.

For example, consider the GP search for models of ROI 19 within a single subject illustrated in Figure 3. Upon completion, all 20 of the models along the Pareto front for this subject had at least one term containing ROI 9, and 17 models had terms containing ROI 20. In the subject pool as a whole, the total counts are 2990 and 1984, respectively. Specifically, of the roughly 5000 models for ROI 19 across all subjects, about 60% have terms containing ROI 9, and about 40% have terms containing ROI 20. Note that these frequencies are not properly normalized, because most models contain several ROI. Thus we normalize by the sum of the counts for all ROI. In the case of ROI 19, this sum is 22016.

The result is a vector for each ROI that describes, in a statistical sense, its relative dependence on each of the other regions. We interpret this vector as a distribution of likely interaction, and define the computed values to be relative *interaction rates* (IR). Note that both linear and nonlinear interactions, as well as weakly and strongly weighted basis functions, are counted equally. We form an interaction map by stacking these IR row vectors to visualize interaction across all 52 ROI, shown in Figure 4(a). The value in row 19 column 9, for example, is $2990/22016 \approx 0.136$, depicted as a yellow square.

Note that the IR map is not symmetric by construction (though it appears nearly so), and indeed the value in row 9, column 19 is $0.148 \neq 0.136$. We interpret a row of the IR map as a distribution of relative *dependence* of the corresponding ROI on each of the other regions. We interpret a column, on the other hand, as a measure of the *influence* of the corresponding ROI on each of the other regions. By averaging the IR map with its transpose, we produce a symmetric, *overall* IR map (not shown) that can be used in hierarchical analysis. We examine the interaction map, and provide results of hierarchical analysis, in the next section.

3. Results

Figure 4(a) shows the interaction map generated by the normalized frequency analysis of the NFM procedure, summarizing ROI interaction across all 242 subjects. To test the robustness of the computed interaction map, we form 100 random subsamples (with replacement) from the pool of 242 subjects, each with 100 subjects. For each sample, we perform the same counting procedure to produce the interaction map corresponding to that sample. A heat map of relative standard deviation (RSD) of IR over the 100 subsamples is shown in Figure 4(b).

The strong block-diagonal structure of the interaction map corresponds directly to the grouping of ROI into ICN. For example, regions 39-42, which form a partial block in the figure, are the four ROI that make up the default mode network (ICN 13) in Laird et al. (2011). Robustness (across subjects) of intra-network interaction is supported by the matching block-diagonal structure of low subsampling RSD (mean intra-network RSD $< 20\%$), for all but ICN 2 (ROI 3-4) and ICN 5 (ROI 10-18). In addition to the strong primary block-diagonal structure, there is a secondary structure of lighter blocks that group ICN together. For example, regions 32-38 are composed of the strong blocks 32-33, 34-35, and 36-38 (corresponding to ICN 10, 11 and 12 respectively). There is a lighter block structure that suggests interaction among these three ICN which are, in fact, *together* responsible for visual processing. The secondary structure visible for regions 19-31 is comprised of ICN 6-8, which perform motor and visuospatial tasks. Each of these examples shows a matching secondary structure of moderate subsampling RSD (mean inter-network RSD $< 30\%$), indicating fairly robust inter-network interaction as well.

3.1. Hierarchical analysis

To further illustrate and clarify the hierarchical organization suggested by the interaction map, we generate the dendrogram in the top of Figure 5 by HCA (hierarchical cluster analysis, implemented in MATLAB with the nearest distance algorithm), using the reciprocal of the overall IR between each pair of ROI as the distance between them. For example, ROI 1 and 2 have an approximate overall IR of 0.2, and thus the distance between them is 5. We emphasize that the organization of ROI into networks, and clustering of those networks into functional groups described in Laird et al. (2011), are both captured by NFM. Some examples:

- The red group forms the visual cluster. ROI 32 and 33, the lateral occipital cortices, form one network (ICN 10), while ROI 34-35, the occipital poles, and ROI 36-38, the lingual gyrus, right cuneus and right fusiform gyrus, respectively, form two other networks (ICN 11 and 12) from within the visual cluster.
- Regions 39-42 (the orange group) form ICN 13, the default mode network, and interact with ROI 4, the ventromedial prefrontal cortex, from ICN 2.
- The green group to the far left includes all but one of the ROI from the motor and visuospatial complex. Interaction of this complex with the middle cingulate cortex (mCC, ROI 9) and the network composed of ROI 46 and 47, thought to be responsible for multiple cognitive processes such as attention and inhibition, is indicated as well, suggesting that this interaction was common among many of the subjects.
- ICN 1 (ROI 1,2), 3 (ROI 5,6), the first two regions from ICN 4 (ROI 7-9), ICN 14 (ROI 43-45), 16 (ROI 48,49), and 17 (ROI 50,51) are also indicated.
- Many of the regions from ICN 5 (ROI 10-18) interact with ICN 1 (ROI 1-2), and also form a loose interaction group with ICN 14 (ROI 43-45), the cerebellum, the most robust connection of which appears to be between ROI 16 and 43.

The robustness of each of the interactions discussed in this list is supported by low interaction rate subsampling RSD, shown in Figure 4(b).

3.2. Impact of nonlinearity

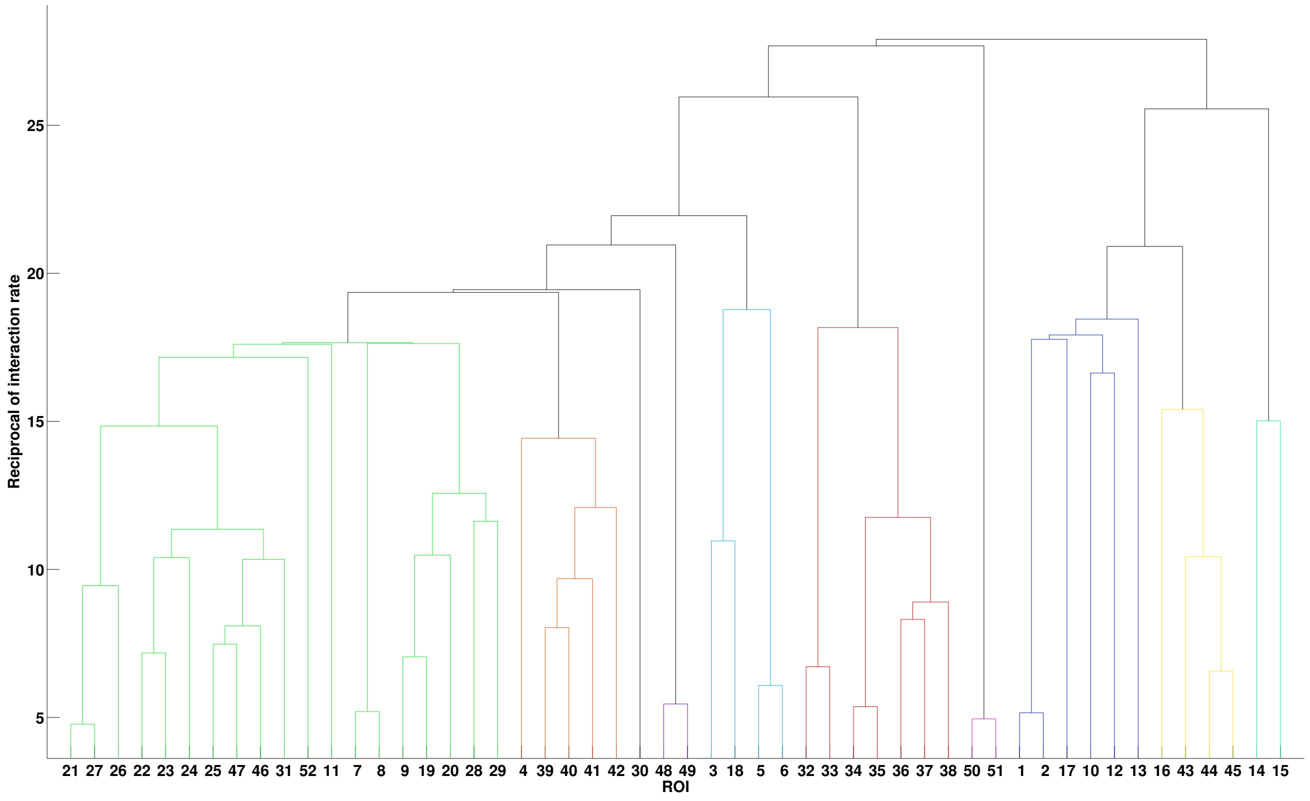
In this section we demonstrate that the NFM procedure both captures the hierarchical structure of ROI interaction indicated by linear analyses, *and* reveals nonlinear interactions not discoverable by such methods. To accomplish this, we compare the population-level hierarchy generated by NFM with the results of an analogous linear procedure involving pairwise correlation analysis. Furthermore, we validate nonlinear relationships suggested by NFM in a stepwise multiple regression, and an elastic net regularized regression, the results of which we describe at the end of this section.

3.2.1. Comparison with correlation analysis

For each of the 242 subjects, we compute the correlation matrix for the 52 ROI time series. Squaring the elements of the correlation matrix and normalizing each row (after setting the diagonal to zero) provides the relative explained variance (relative R^2) of the ROI corresponding to that row by each of the other 51 ROI. The average of the 242 normalized subject matrices is interpreted as the linear version of the population-level IR map generated by NFM. As with IR, the reciprocal of relative explained variance can be considered a distance between ROI (higher relative R^2 means closer). The resulting hierarchy generated by HCA is shown in the bottom of Figure 5.

As expected, much of the large-scale structure revealed by NFM is also indicated by the linear correlation analysis. The similarity of the generated hierarchies supports the validity of the models discovered by GP (i.e., the algorithm is not excessively overfitting the data), and the subtle differences between them suggest potentially interesting interactions that are missed if linearity is assumed. In the following we investigate one of these differences.

Interaction of the mCC (ROI 9) with the motor visuospatial complex is evident in both hierarchies. However, in the linear analysis it appears more closely connected with its own ICN (ROI 7,8, the bilateral anterior insula), and only with the posterior dorsomedial prefrontal cortex (dmPFC, ROI 19) from ICN 6. In contrast, NFM reveals that activity in the mCC is related to more components of the motor system. The nonlinear models generated by GP show a strong connection between the mCC, posterior dmPFC, and the paracentral lobule (PL) of the primary motor cortex (ROI 29 from ICN 8) shown in red, green, and blue, respectively, in Figure 6. Specifically, about 20%



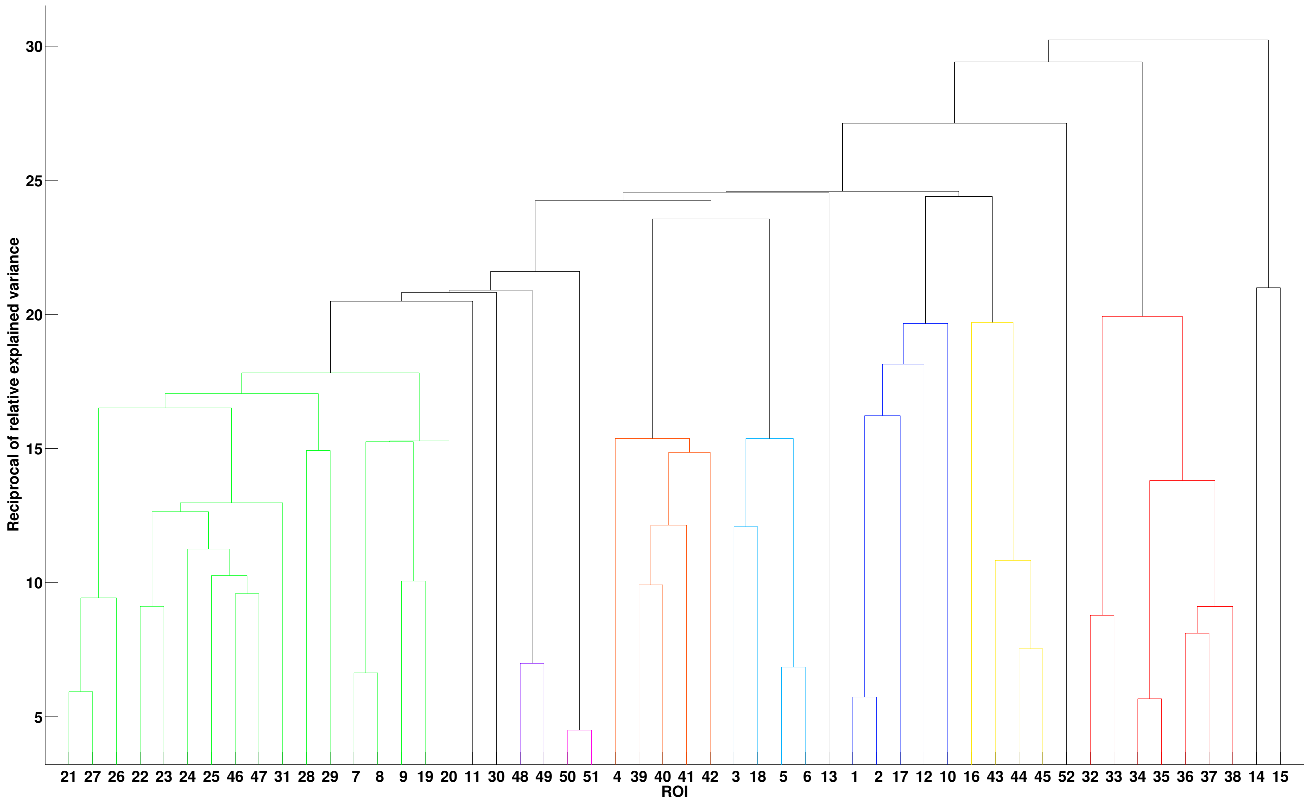


Figure 5: Hierarchical cluster analysis (HCA) of interaction among ROI, generated with NFM (top) and correlation analysis (bottom).

of all of the models generated for the activity in the posterior dmPFC, across all subjects and levels of complexity, contain *both* the mCC and the PL as explanatory variables.

For many of these models, the mCC and PL only show up as linear terms, so it is reasonable to wonder why the correlation analysis did not pick up this interaction. The vast majority of models containing mCC and PL in only linear terms *also* contain nonlinear terms in other ROI. It is the posterior dmPFC *along with* these nonlinear terms that is correlated with the mCC and PL. Thus the interaction is hidden from linear analyses. Furthermore, many of the models *do* contain nonlinear terms involving the mCC and PL. In fact, the product of the activity in these two regions shows up in 78 models for the posterior dmPFC across 21 different subjects, and it is always additive. This term is involved in models across the spectrum of complexity, including instances where it is the *only* term.

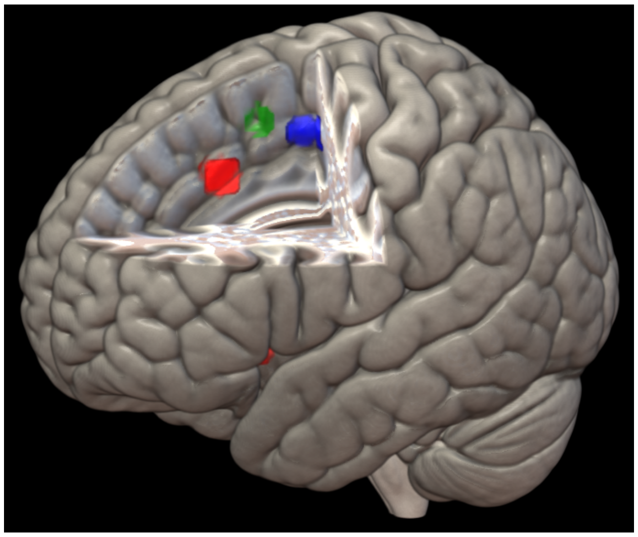


Figure 6: NFM reveals a nonlinear interaction among these three ROI: mCC (red), posterior dmPFC (green), and PL in the primary motor cortex (blue).

3.2.2. Validation of nonlinear terms

To validate first order nonlinearity (pairwise product and quotient terms, as well as reciprocals) suggested by NFM, we first randomly assign 100 subjects to a training group, and 100 different subjects to a testing group. NFM results are aggregated over the training group to produce an IR map and hierarchy (not shown) summarizing ROI interaction within the training group as a whole. The roughly 2000 specific models generated by NFM for each region (approximately 20 models per training subject) are then used to inform the modeling of ROI activity in that region within the testing group, by stepwise regression.

For each ROI and each testing subject, we first perform a standard stepwise linear regression using the other 51 ROI as regressors. We then perform a stepwise regression including all first order nonlinear terms suggested by NFM over the training group in addition to the 51 linear regressors. Statistics of the linear and nonlinear models are compared to determine the effect of including these first order terms. To illustrate, we describe results of the validation procedure for the posterior dmPFC (ROI 19) here.

We show a histogram of increase in the percentage of explained variance for the nonlinear versus linear regression models for the posterior dmPFC in Figure 7. The inclusion of first order nonlinear terms suggested by NFM over the training group increases the percentage of explained variance for *every* test subject, with a mean increase of 12.5% and maximum increase of 42%. The nonlinear models contain more terms (mean 46, compared with mean 19 for linear models), so a potential concern is that the increase in R^2 might simply be a result of the additional degrees of freedom. However, for each test subject the nonlinear model F-statistic is also greater than that of the linear model (mean increase of 83.5, maximum increase of 1000), and comparisons of adjusted R^2, which account for differences in degrees of freedom, show only slightly smaller increases for all test subjects. This suggests that the increase in explained variance is due to explanatory power of the nonlinear terms, and not simply the additional degrees of freedom in the nonlinear models.

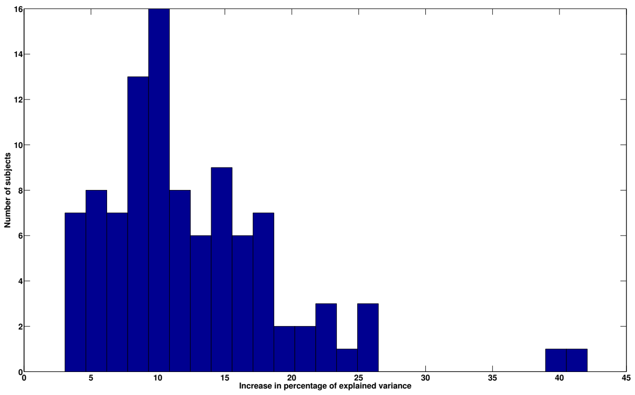


Figure 7: Histogram of increase in explained variance. The inclusion of first order nonlinear terms suggested by NFM over the training group, in a stepwise regression analysis for the posterior dmPFC in the testing group, increases the percentage of explained variance for *every* test subject, with a mean increase of 12.5% and maximum increase of 42%.

To further support the validity of the nonlinear terms suggested by NFM, we apply a similar testing approach using a machine learning algorithm called elastic net regularized regression. In contrast to stepwise regression, regularization allows for the inclusion of highly correlated explanatory variables, while simultaneously discounting regressors with very small coefficients. Regularized models can have more explanatory power or fewer terms (or both), with respect to those from stepwise regression. We see each of these scenarios in the present case. Using the same training and testing groups, and modeling the same ROI (the posterior dmPFC), elastic net regularization produces linear models with an average of 45 terms that explain roughly the same amount of variance (on average) as the nonlinear models generated with stepwise regression, improving upon the explanatory power of their stepwise counterparts. However, the regularized *nonlinear* models provide that same explanatory power with a mean of only 26 terms. Furthermore, the regularized nonlinear model is preferable to the regularized linear model, as determined by the Akaike information criterion, *for every single test subject*.

3.3. Group-level variation

Variation among individuals (illustrated in Appendix B) suggests that statistics of interaction rates among ROI may differ between phenotypic groups. The hierarchical organization of ROI induced by IR might illuminate, in such cases, variation in functional dynamics associated with demographic, behavioral, or genetic characteristics. An example illustrating this potential is provided by the contrast between drinking (D) and non-drinking (ND) adolescents from the IMAGEN dataset. In Figure 8, we show hierarchies for the top and bottom 100 subjects in terms of lifetime drinking score, determined by self-report questionnaire, corresponding to those who have had 2 or more lifetime drinks, and those who have had 1 or fewer, respectively. The two hierarchies are similar to one another (and comparable to the population level hierarchy), but subtle differences between them suggest group-differentiating factors.

- The ROI pair 3,18, the subgenual anterior cingulate cortex (ACC) and fornix body, respectively, are coupled in both the D and ND groups. However, their arrangement in the hierarchies is different, as we'll describe in a moment, resulting from the following two distinguishing interaction rates.

- For the ND group, there is a 22% lower IR between ROI 6, the left globus pallidus, and the fornix body, ROI 18. We note that this reduced interaction is completely missed by pairwise correlation analysis, (which indicates a slightly reduced interaction among *drinkers*, see Appendix C), and thus appears to be an entirely nonlinear effect.

- In contrast, there is a 33% higher intra-network IR within ICN 2, comprised of ROI 3-4, the subgenual ACC and the ventromedial prefrontal cortex (vmPFC), respectively, among non-drinkers. Though this difference is also indicated by correlation analysis, only about half of the effect is captured (a 16% elevation).

- These two differences in interaction cooperate to shuffle the hierarchical arrangement of ROI in the D versus ND group. The subgenual ACC and fornix body are most closely associated with the default mode network in non-drinkers, through the vmPFC. Among drinkers, in contrast, they are grouped directly with the bilateral globus pallidus of ICN 3 (ROI 5-6). In other words, in drinkers there is a tighter coupling among ROI most strongly linked to reward and thirst tasks as reported in Laird et al. (2011). The relevant ROI are shown in Figure 9.

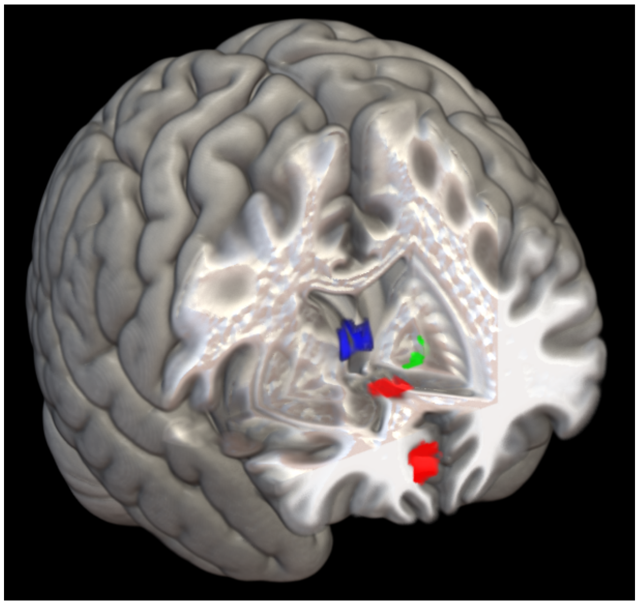
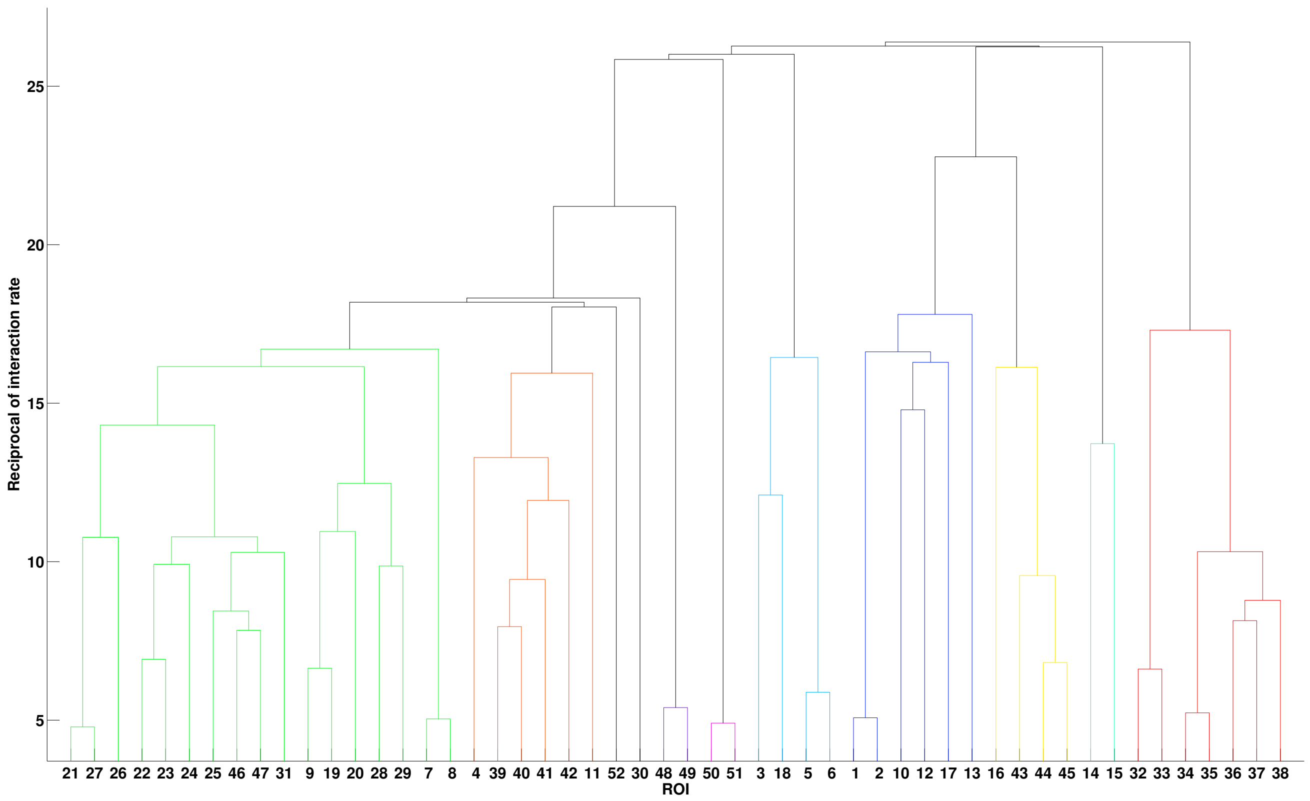


Figure 9: Interaction between the subgenual ACC (top red) and vmPFC (bottom red) is lower among drinkers, who also show elevated interaction between the left globus pallidus (green) and fornix body (blue), an apparently nonlinear effect.

- The largest single difference between the D and ND groups is a 74% elevated IR between the right angular gyrus (ROI 41 in the default mode network) and ROI 11, the posterior cingulate cortex, among drinkers. These ROI are shown in red and green, respectively, in Figure 10. About half of this effect is captured by correlation analysis.

4. Discussion

The large extent of ICN reproduction, and their hierarchical organization into functional groups using an entirely different approach than that described in Laird et al. (2011), provides strong evidence for the analytical potential of NFM. Furthermore, the technique reveals nonlinear interactions that are not discoverable with standard linear techniques, or without prior hypotheses. Such relationships could provide a new window into brain function, and this highlights the potential of the methodology as a *hypothesis generator*. Of course proper care must be taken (with regard to independence of observations, etc.) in the ensuing investigations of such data-driven hypotheses. Nonetheless, hypothesis generation is a powerful tool



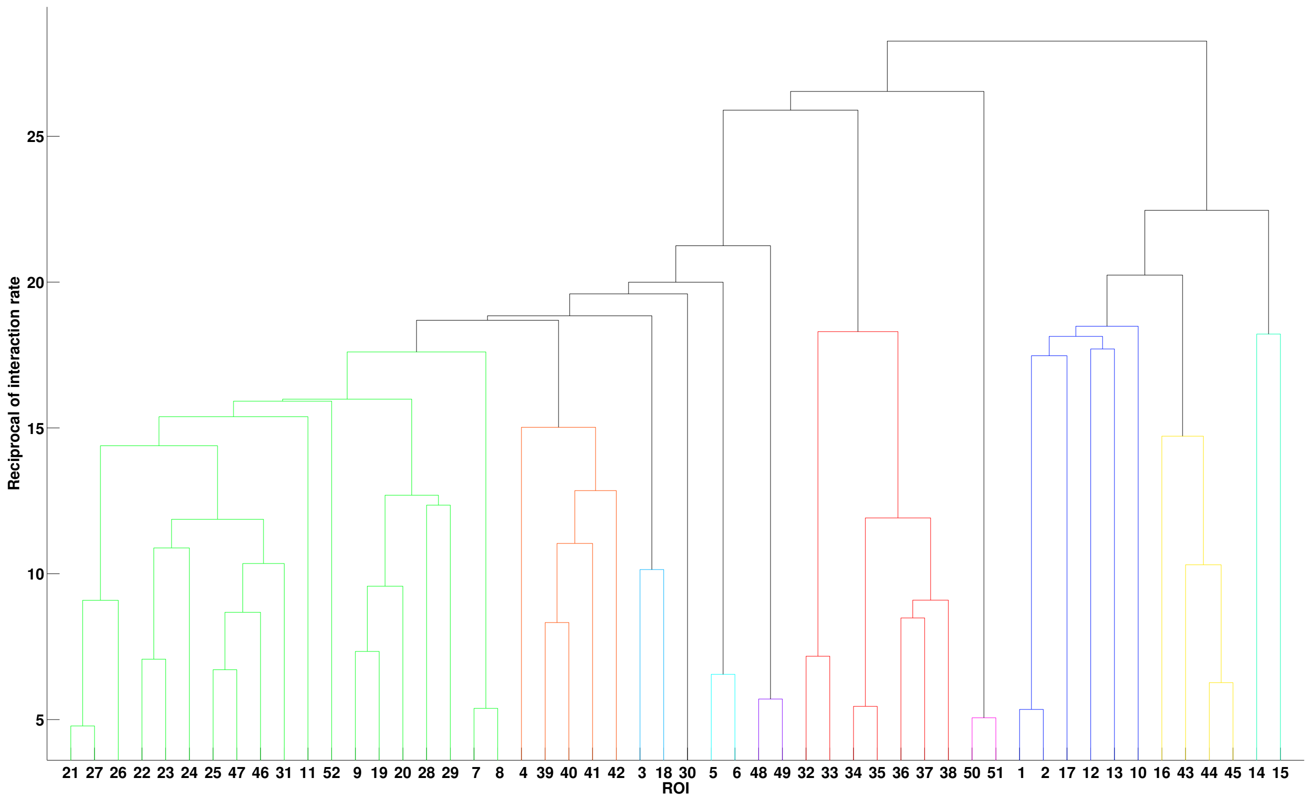


Figure 8: Hierarchies for groups with high (top) and low (bottom) alcohol consumption rates, defined by two or more lifetime drinks and one or fewer lifetime drinks, respectively.

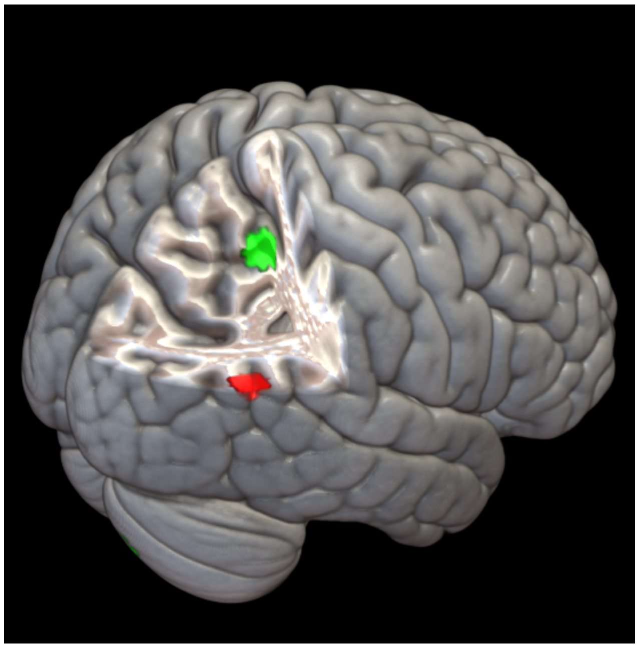


Figure 10: Interaction between the right angular gyrus from the default mode network (red), and posterior cingulate cortex (green) is 74% higher among drinkers.

for scientific exploration, and has been used recently to inform biomedical research, such as in Abedi et al. (2012) and Spangler et al. (2014).

In addition to providing insight on its own, the NFM procedure complements other modes of analysis. A potentially promising extension, especially for a hybrid version capable of voxel level analysis (discussed in Appendix D), would be to use it in conjunction with graph-theoretic analyses such as those described in Bassett and Bullmore (2006), Stam and Reijneveld (2007), and van den Heuvel et al. (2008). The general technique, as detailed in Bullmore and Sporns (2009) and Rubinov and Sporns (2010), is to compute pairwise correlations among all voxels, set a threshold above which two voxels are considered connected, and calculate various network summary measures (e.g., degree distribution, assortativity, diameter, etc.). By simply replacing correlations in these networks with interaction rates determined by NFM, the assumption of linearity is left behind.

Finally, it is important to note that the specific forms of the models in the output of the GP algorithm have been analyzed simplistically in the present work. A major potential benefit of NFM is the insight that might be gained from precisely analyzing these mathematical descriptions of the relationships among ROI in the brain. Of course, ascribing meaning to any particular one of these models would have to be done cautiously. However, given the results we describe here, obtained by a coarse treatment, the *collection* of models determined by GP may offer a number of as yet undiscovered insights. This seems a potentially fruitful avenue for future theoretical research.

5. Conclusions

Results produced in our study suggest that there is potential analytical power in the use of NFM, or some modification thereof, in the neuroimaging domain. The procedure we investigated here utilizes commercially available, out-of-the-box GP software, and preliminary statistical analysis of its output. Many improvements and extensions are possible, only some of which we have suggested in this work. Reproduction of recent results constitutes a measure of cross-validation, and the preliminary results presented demonstrate the unique capability of NFM to discover non-linear relationships among regions of the brain that hold promise for illuminating differences in brain function between subject groups. Further, the mathematical characterizations we have achieved, which are not limited by linear or univariate assumptions, are ripe for future investigation.

Acknowledgements

This work was supported by DARPA grant FA8650-11-1-7155, and the Vermont Advanced Computing Core, NASA (NNX-08AO96G) at the University of Vermont, which provided High Performance Computing resources. The IMAGEN study receives research funding from the European Communitys Sixth Framework Programme (LSHM-CT-2007-037286).The authors also wish to acknowledge NASA for funding in support of NAA through the Vermont Space Grant Fellowship; Mike Schmidt, creator of the GP package Eureqa used in this study; and Ilknur Icke, Trevor Andrews and Richard Watts for crucial conceptual and technical support.

Appendix A. Table of ROI

In Table A.1 we list all 52 ROI investigated in this study by number, give their anatomical names, indicate the ICN from within which they were defined, and provide visual representations of their locations within the brain. Due to its length, the table appears after the References.

Appendix B. Subject-level variation

Figure B.11 contains individual subject interaction hierarchies for two different (randomly selected) subjects, generated by 100 random restarts of the GP algorithm and subsequent normalized frequency analysis. Though the two hierarchies are quite different from one another, they do show some network organization similar to that illustrated in the population level hierarchy (top of Figure 5).

- Portions of the visual cluster (ROI 32-38) are intact in each case.

- Many of the two-region networks remain together, e.g. ICN 1 (ROI 1,2), ICN 16 (ROI 48,49), and though associated with other ROI, also ICN 3 (ROI 5,6) and ICN 17 (ROI 50,51)
- The default mode network (ROI 39-42) is mostly intact in each subject.

The interaction profile of the default mode network illustrates an interesting distinction between the two subjects. For the top subject, the network is fully intact, interacting with ROI 4 (consistent with the population level hierarchy), and also interacting with ROI 26 from the motor visuospatial complex. For the bottom subject, three of the four ROI in the network remain together, but interact instead with several other ROI from the emotional interoceptive class instead of ROI 4, specifically ROI 10, 12, and 16, and a different ROI from the motor visuospatial complex as well (ROI 29 instead of 26). By themselves, these dendrogram comparisons offer no conclusive evidence regarding connections between cognitive processes. However, an experiment could be designed to test if any inferences can be made from such distinctions. For example, the administration of post-scan surveys might grant some interpretability to the specifics of these single-subject interaction hierarchies.

Appendix C. Linear HCA of alcohol consumption

Here we demonstrate that the shuffling of the interaction hierarchy in drinking (D) versus non-drinking (ND) adolescents discovered by NFM is not uncovered by linear correlation analysis. To perform group-level correlation analysis, the normalized relative R^2 matrices for each subject (described in Section 3.2) are averaged over the 100 subjects in each group. Recall that these matrices are generated for each subject by computing the correlation matrix for the 52 ROI time series, squaring the elements, and normalizing each row (after setting the diagonal to zero). The reciprocal of relative explained variance can be considered a distance between ROI (higher relative R^2 means closer), and the resulting D and ND hierarchies generated by HCA are shown in the top and bottom, respectively, of Figure C.12.

Comparison with Figure 8 suggests that this linear analysis *partially* uncovers a distinguishing difference in interaction between drinking and non-drinking adolescents. Specifically, among non-drinkers, a higher intra-network interaction within ICN 2, comprised of the subgenual ACC and the vmPFC (ROI 3 and 4, respectively) is detected here. The result is an indirect coupling, within non-drinkers, of the default mode network (ROI 39-42) and the complex comprised of ROI 3,18,5,6, through the vmPFC.

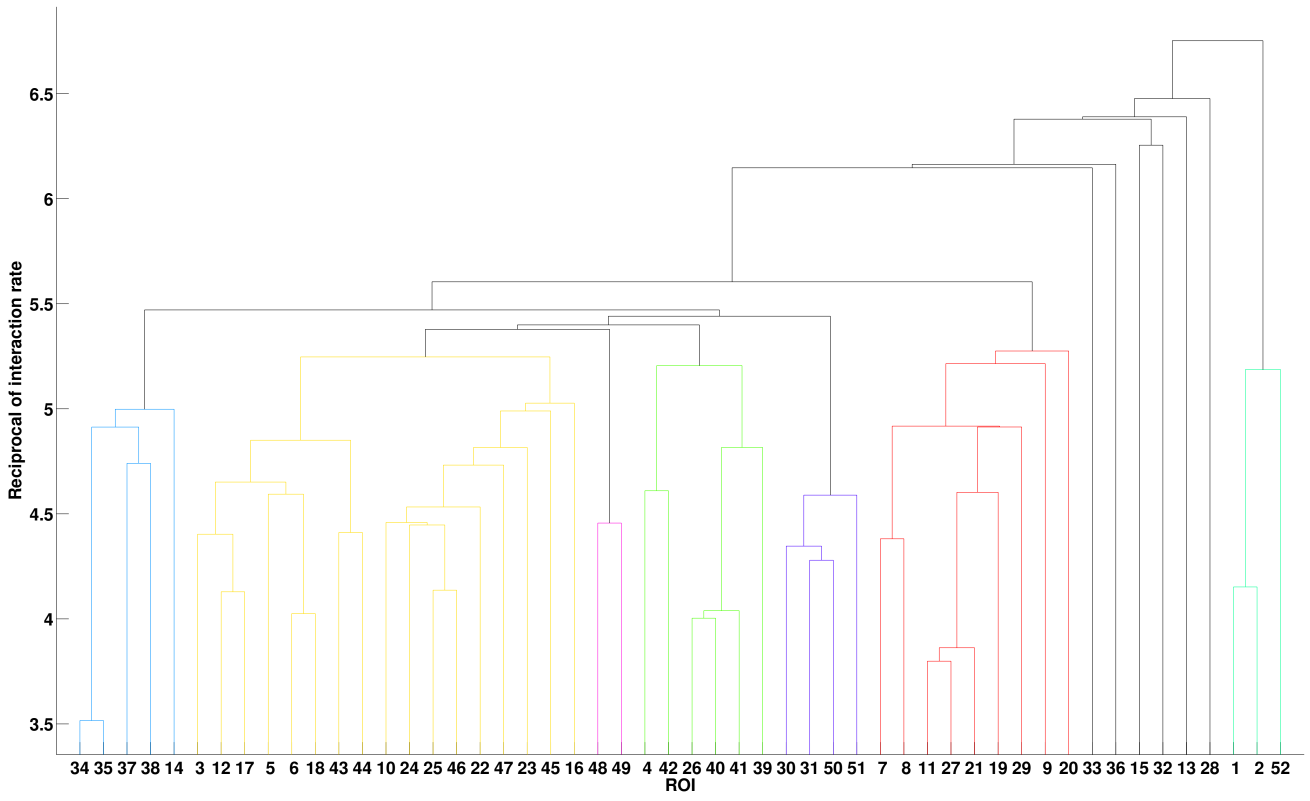
The results of NFM provide further insight in two important ways. First, the elevated interaction within ICN 2 among non-drinkers is detected at twice the strength. Second, the main interaction responsible for grouping the complex of ROI 3,18,5,6, specifically the interaction between the left globus pallidus (ROI 6) and fornix body (ROI 18), is lower among non-drinkers. This second effect is entirely missed by correlation analysis, suggesting that it is nonlinear in nature. The result of capturing these effects *together*, as shown in Figure 8, is a breakup of the complex in non-drinkers, for whom ROI 3,18 are separated from the bilateral globus pallidus of ICN 3 (ROI 5-6). This breakup is suggestive, as each of these ROI is associated with emotion, reward, and interoceptive processes such as thirst, and experiments reporting activity in ICN 5, including the fornix body (ROI 18), predominantly involved interoceptive stimulation, as reported in Laird et al. (2011).

Appendix D. Improvements and modifications

The GP implementation we used for this study is the commercially available package Eureqa from Nutonian, as described in Schmidt and Lipson (2009). Though much of its behavior can be controlled through the interface or command line, it is proprietary code and thus somewhat of a black box. There are many reasons why a dedicated, open source implementation of GP would be more desirable.

A major challenge for this method of analysis is the computational expense of running a large number of GP searches. Generating the IR map for a single subject requires a large number of random restarts for each ROI. For example, running 100 restarts for each of the 52 ROI in this study, allowing 1 core-hour for each search, requires over 10 hours with access to 500 dedicated processors. The procedure as described here is likely computationally prohibitive for running analyses on large numbers of subjects, or for larger collections of ROI. Intelligent stopping criteria, and many other approaches to the mitigation of computational expense, have been reported at length in the GP literature, an example of which is the use of graphics processors reported in Harding and Banzhaf (2007). It may also be possible to determine an ideal (and smaller) number of restarts that balances computation time with the statistical power of the resulting IR map.

It should also be noted that for collections of ROI much larger than that considered here, in addition to the computational expense resulting from more required searches, each search will take much longer to produce meaningful models due to the larger number of possible explanatory variables. A hybrid method of symbolic regression employing a machine learning algorithm called FFX (Fast Function Extraction) described in McConaghy (2011) as a first pass, and then GP, has great potential for the treatment of higher dimensional data, e.g., large numbers of ROI. A prototype of this method was reported in Icke et al. (2014). FFX is a deterministic algorithm that builds up models with nonlinear terms (e.g., products of ROI signal) in a prescribed fashion and evaluates explanatory power at



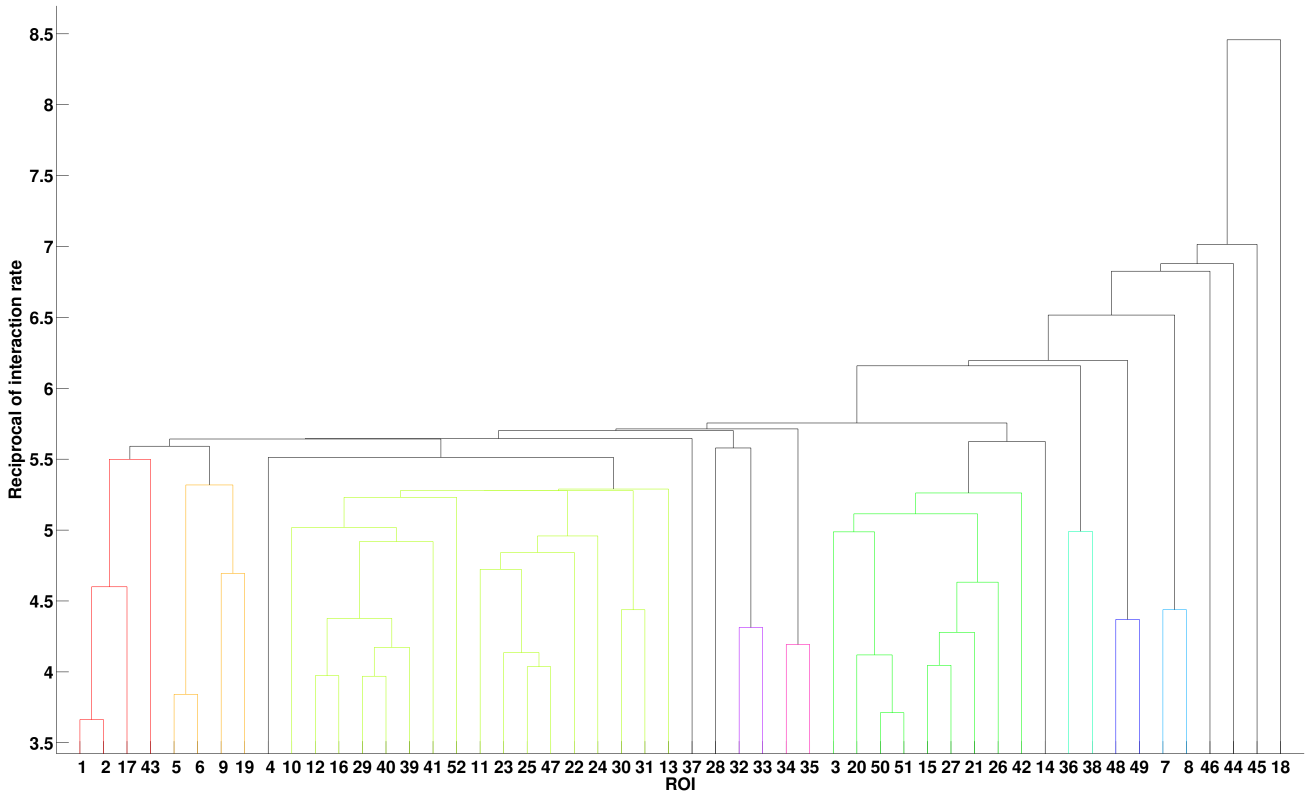
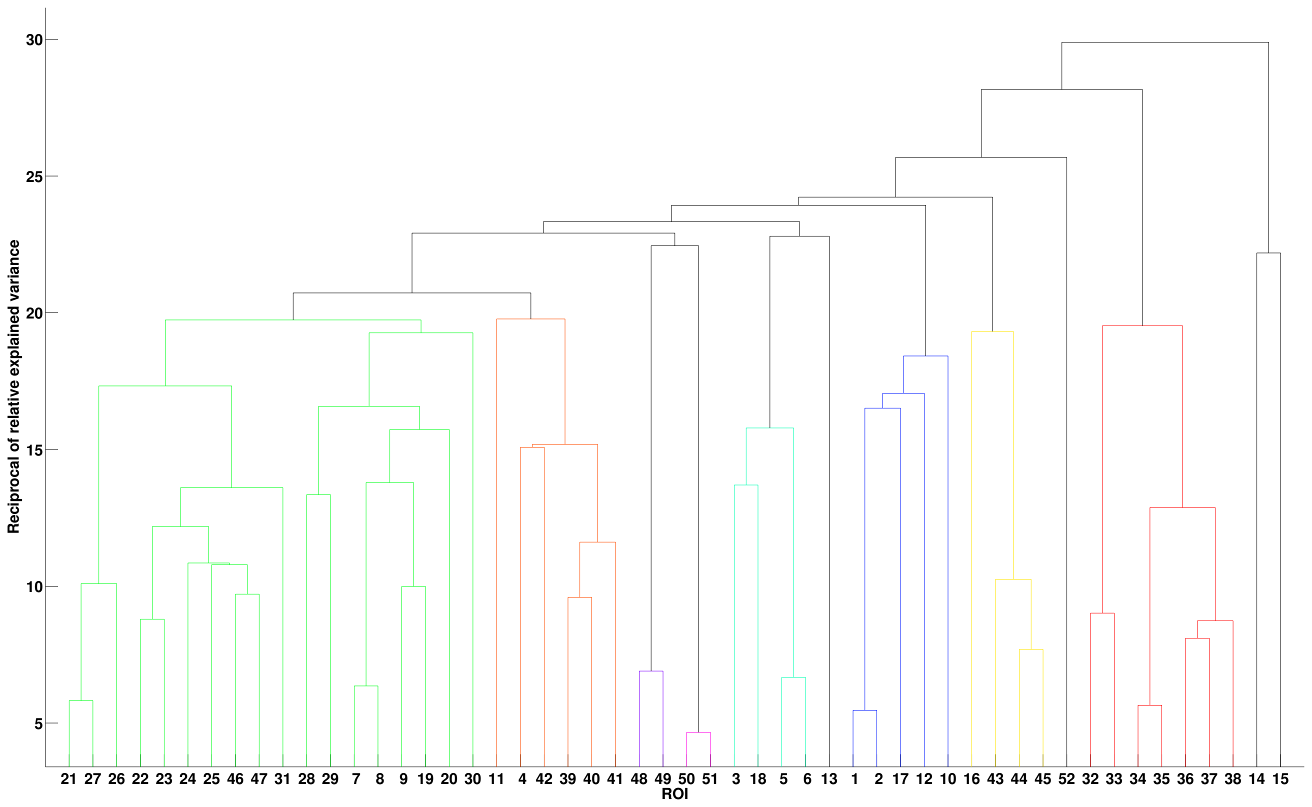


Figure B.11: Example hierarchies for two different individual subjects. Note the large degree of variation between the two.



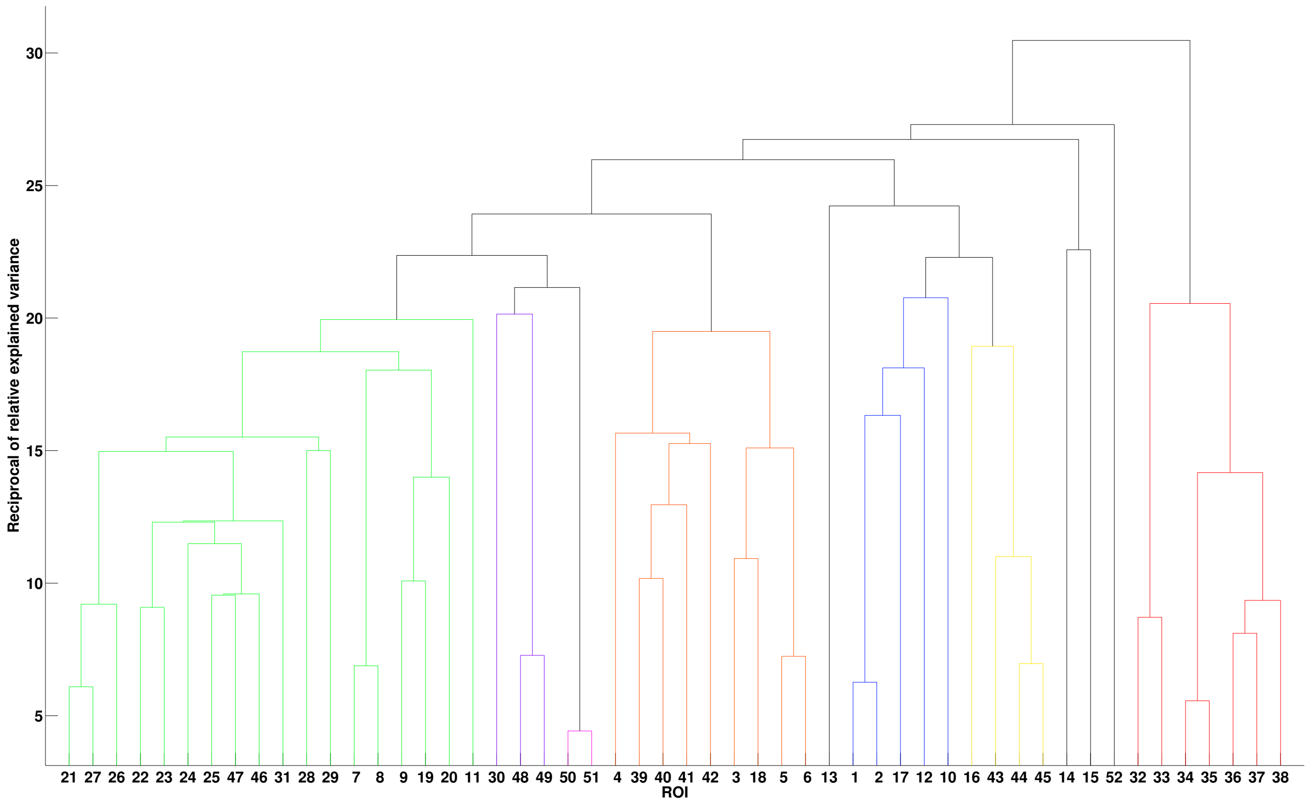


Figure C.12: Linear hierarchies for groups with high (top) and low (bottom) alcohol consumption rates, defined by two or more lifetime drinks and one or fewer lifetime drinks, respectively.

each stage. By ruling out ROI that are likely not explanatory at each stage, the algorithm reduces the dimensionality of the search. In other words, at the cost of reduced breadth in the search space, the algorithm provides huge reductions in computation time in addition to reducing the number of variables that will eventually be injected into the GP algorithm. Implemented effectively, this hybrid algorithm could eliminate the necessity of ROI selection completely by allowing direct regression over voxel signals.

An ever-present concern in the analysis of fMRI is the level of noise in the data. Particularly in the case of regressing over voxel signals, low signal-to-noise ratio is a major challenge, and indeed GP efficacy is diminished in such circumstances. However, there has been some work on modifying the GP algorithm to better manage noisy data, an example of which is the inclusion of noise generators called *stochastic elements* with user-defined distributions (e.g., Gaussian or uniform) as potential explanatory "variables". These generators can themselves end up inside complex functions within the models, providing those models the capability of reproducing realistic noise distributions more likely to be at play than the typical Gaussian. There is no guarantee that this modification will prove beneficial in the case of fMRI, but it has been shown, in Schmidt and Lipson (2007), to effectively identify exact underlying analytical models in the presence of nonlinear, non-Gaussian and nonuniform noise.

References

Abedi, V., Zand, R., Yeasin, M., Faisal, F. E., 2012. An automated framework for hypotheses generation using literature. BioData Min 5 (1), 13.

Bassett, D. S., Bullmore, E., Dec 2006. Small-world brain networks. Neuroscientist 12 (6), 512–523.

Birn, R. M., Murphy, K., Handwerker, D. A., Bandettini, P. A., 2009. fmri in the presence of task-correlated breathing variations. NeuroImage 47 (3), 1092 – 1104, brain Body Medicine.

Bullmore, E., Sporns, O., 03 2009. Complex brain networks: graph theoretical analysis of structural and functional systems. Nat Rev Neurosci 10 (3), 186–198.
URL `http://dx.doi.org/10.1038/nrn2575`

Calhoun, V., Adali, T., Pearlson, G., Pekar, J., 2001. A method for making group inferences from functional mri data using independent component analysis. Human Brain Mapping 14 (3), 140–151.
URL `http://dx.doi.org/10.1002/hbm.1048`

Friston, K., Harrison, L., Penny, W., 2003. Dynamic causal modelling. NeuroImage 19 (4), 1273 – 1302.

Harding, S., Banzhaf, W., 2007. Fast genetic programming on gpus. In: Ebner, M., ONeill, M., Ekrt, A., Vanneschi, L., Esparcia-Alczar, A. (Eds.), Genetic Programming. Vol. 4445 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 90–101.

Hui, K. K., Liu, J., Makris, N., Gollub, R. L., Chen, A. J., I. Moore, C., Kennedy, D. N., Rosen, B. R., Kwong, K. K., 2000. Acupuncture modulates the limbic system and subcortical gray structures of the human brain: Evidence from fmri studies in normal subjects. Human Brain Mapping 9 (1), 13–25.

Hyvärinen, A., Pajunen, P., 1999. Nonlinear independent component analysis: Existence and uniqueness results. Neural Networks 12 (3), 429 – 439.

Icke, I., Allgaier, N., Danforth, C. M., Whelan, R. A., Garavan, H. P., Bongard, J. C., 2014. A deterministic and symbolic regression hybrid applied to resting-state fmri data. In: Riolo, R., Moore, J., Kotanchek, M. (Eds.), Genetic Programming Theory and Practice XI. Springer, Ch. 9, pp. 155–173.

Kanwisher, N., Stanley, D., Harris, A., 1999. The fusiform face area is selective for faces not animals. NeuroReport 10 (1).

Kruggel, F., Zysset, S., von Cramon, D., 2000. Nonlinear regression of functional {MRI} data: An item recognition task study. NeuroImage 12 (2), 173 – 183.

Laird, A. R., Fox, P. M., Eickhoff, S. B., Turner, J. A., Ray, K. L., McKay, D. R., Glahn, D. C., Beckmann, C. F., Smith, S. M., Fox, P. T., 2011. Behavioral interpretations of intrinsic connectivity networks. J. Cognitive Neuroscience, 4022–4037.

Liu, P., Zhou, G., Zhang, Y., Dong, M., Qin, W., Yuan, K., Sun, J., Liu, J., Liang, J., von Deneen, K. M., Liu, Y., Tian, J., 2010. The hybrid glmica investigation on the neural mechanism of acupoint st36: An fmri study. Neuroscience Letters 479 (3), 267 – 271.

McConaghy, T., 2011. Ffx: fast, scalable, deterministic symbolic regression technology. In: Riolo, R., Vladislavleva, E., Moore, J. (Eds.), Genetic Programming Theory and Practice IX. Springer, Ch. 13.

McKeown, M. J., Sejnowski, T. J., 1998. Independent component analysis of fmri data: Examining the assumptions. Human Brain Mapping 6 (5-6), 368–372.

Mueller, W. M., Yetkin, F. Z., Hammeke, T. A., Morris, G. L. I., Swanson, S. J., Reichert, K., Cox, R., Haughton, V. M., 1996. Functional magnetic resonance imaging mapping of the motor cortex in patients with cerebral tumors. Neurosurgery 39 (3).

Mukamel, R., Gelbard, H., Arieli, A., Hasson, U., Fried, I., Malach, R., 2005. Coupling between neuronal firing, field potentials, and fmri in human auditory cortex. Science 309 (5736), 951–954.
URL `http://www.sciencemag.org/content/309/5736/951.abstract`

Murphy, K., Harris, A. D., Wise, R. G., 2011. Robustly measuring vascular reactivity differences with breath-hold: Normalising stimulus-evoked and resting state {BOLD} fmri data. NeuroImage 54 (1), 369 – 379.

Nolde, S. F., Johnson, M. K., D'Esposito, M., 1998. Left prefrontal activation during episodic remembering: an event?related fmri study. NeuroReport 9 (15).

Rio, D. E., Rawlings, R. R., Woltz, L. A., Gilman, J., Hommer, D. W., 2013. Development of the complex general linear model in the fourier domain: Application to fmri multiple input-output evoked responses for single subjects. Computational and Mathematical Methods in Medicine 2013, 16.
URL `http://dx.doi.org/10.1155/2013/645043`

Rubinov, M., Sporns, O., 2010. Complex network measures of brain connectivity: Uses and interpretations. NeuroImage 52 (3), 1059 – 1069, computational Models of the Brain.

Schmidt, M., Lipson, H., 2009. Distilling free-form natural laws from experimental data. Science 324 (5923), 81–85.
URL `http://www.sciencemag.org/content/324/5923/81.abstract`

Schmidt, M. D., Lipson, H., 7-11 Jul. 2007. Learning noise. In: Thierens, D., Beyer, H.-G., Bongard, J., Branke, J., Clark, J. A., Cliff, D., Congdon, C. B., Deb, K., Doerr, B., Kovacs, T., Kumar, S., Miller, J. F., Moore, J., Neumann, F., Pelikan, M., Poli, R., Sastry, K., Stanley, K. O., Stutzle, T., Watson, R. A., Wegener, I. (Eds.), GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation. Vol. 2. ACM Press, London, pp. 1680–1685.

Schumann, G., Loth, E., Banaschewski, T., Barbot, A., Barker, G., Buechel, C., Conrod, P., Dalley, J., Flor, H., Gallinat, J., Garavan, H., Heinz, A., Itterman, B., Lathrop, M., Mallik, C., Mann, K., Martinot, J.-L., Paus, T., Poline, J.-B., Robbins, T., Rietschel, M., Reed, L., Smolka, M., Spanagel, R., Speiser, C., Stephens, D., Stroehle, A., Struve, M., Williams, S., Desrivieres, S., 2010. The imagen study: reinforcement-related behaviour in normal brain function and psychopathology. Molecular Psychiatry 15 (12), 1128 – 1139.

Smith, S. M., Fox, P. T., Miller, K. L., Glahn, D. C., Fox, P. M., Mackay, C. E., Filippini, N., Watkins, K. E., Toro, R., Laird, A. R., Beckmann, C. F., 2009. Correspondence of the brain's func-

tional architecture during activation and rest. Proceedings of the National Academy of Sciences 106 (31), 13040–13045.
URL http://www.pnas.org/content/106/31/13040.abstract

Spangler, S., Wilkins, A. D., Bachman, B. J., Nagarajan, M., Dayaram, T., Haas, P., Regenbogen, S., Pickering, C. R., Comer, A., Myers, J. N., Stanoi, I., Kato, L., Lelescu, A., Labrie, J. J., Parikh, N., Lisewski, A. M., Donehower, L., Chen, Y., Lichtarge, O., 2014. Automated hypothesis generation based on mining scientific literature. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '14. ACM, New York, NY, USA, pp. 1877–1886.
URL http://doi.acm.org/10.1145/2623330.2623667

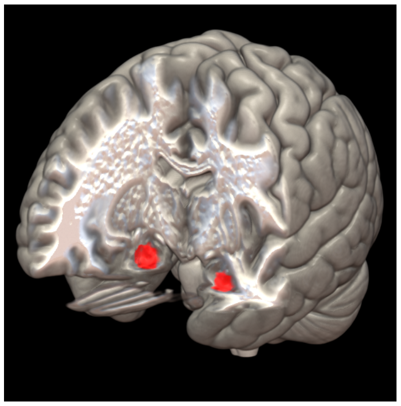
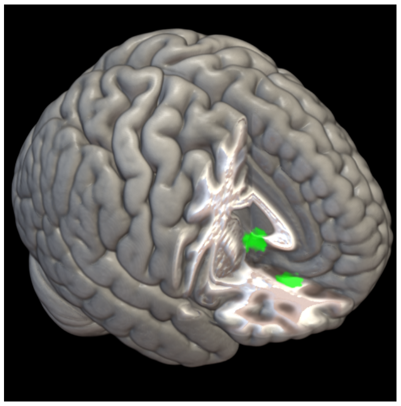
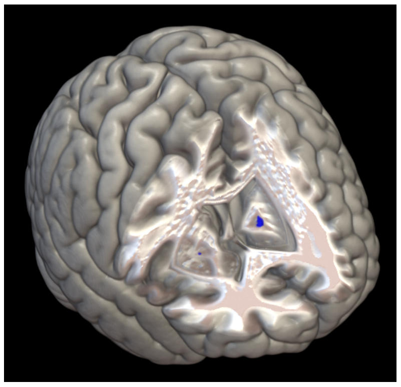
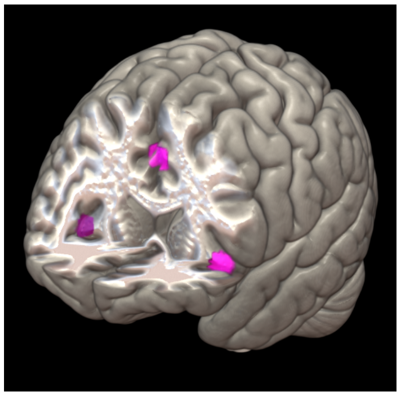
Stam, C. J., Reijneveld, J. C., 2007. Graph theoretical analysis of complex networks in the brain. Nonlinear Biomed Phys 1 (1), 3.

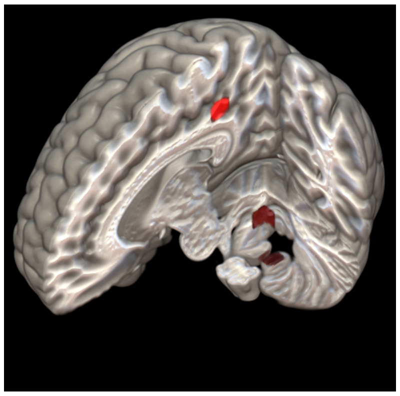
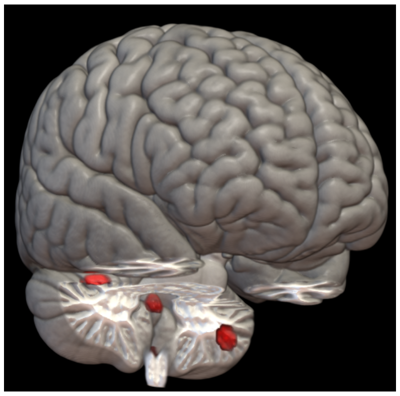
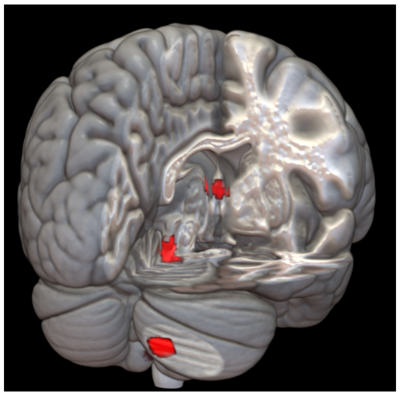
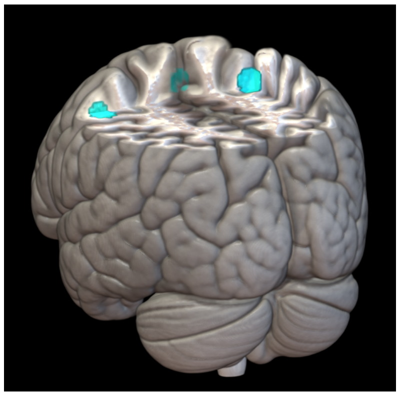
van den Heuvel, M. P., Stam, C. J., Boersma, M., Hulshoff Pol, H. E., Nov 2008. Small-world and scale-free organization of voxel-based resting-state functional connectivity in the human brain. Neuroimage 43 (3), 528–539.

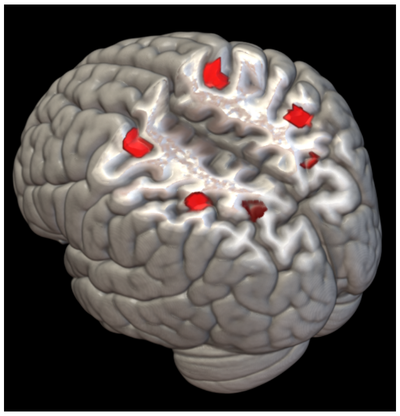
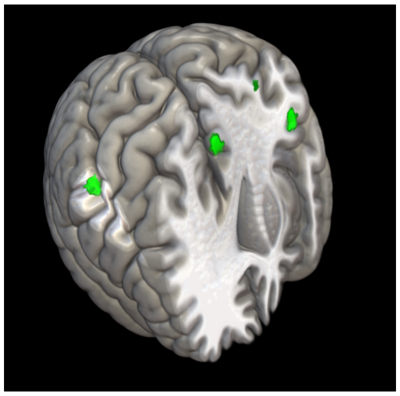
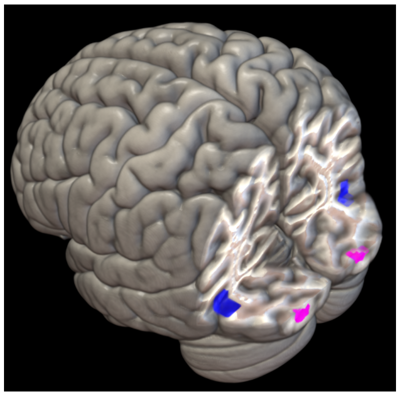
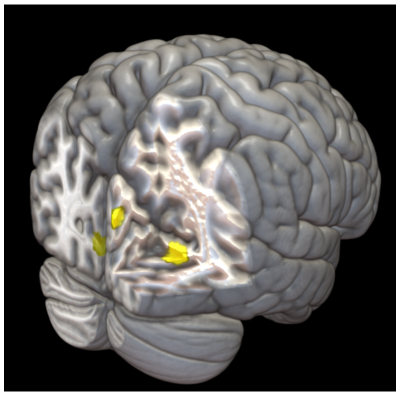
Whelan, R., Conrod, P. J., Poline, J.-B., Lourdusamy, A., Banaschewski, T., Barker, G. J., Bellgrove, M. A., Buchel, C., Byrne, M., Cummins, T. D. R., Fauth-Buhler, M., Flor, H., Gallinat, J., Heinz, A., Ittermann, B., Mann, K., Martinot, J.-L., Lalor, E. C., Lathrop, M., Loth, E., Nees, F., Paus, T., Rietschel, M., Smolka, M. N., Spanagel, R., Stephens, D. N., Struve, M., Thyreau, B., Vollstaedt-Klein, S., Robbins, T. W., Schumann, G., Garavan, H., 06 2012. Adolescent impulsivity phenotypes characterized by distinct brain networks. Nat Neurosci 15 (6), 920–925.
URL http://dx.doi.org/10.1038/nn.3092

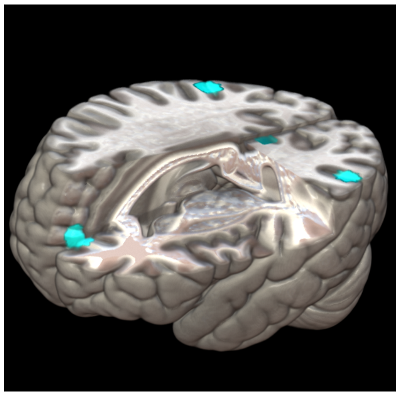
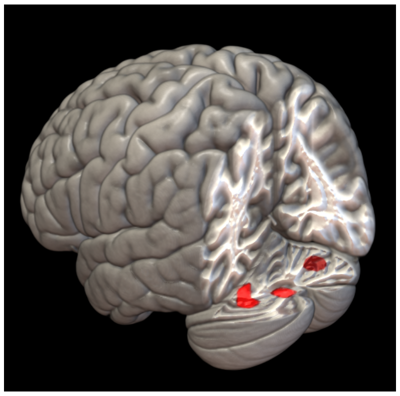
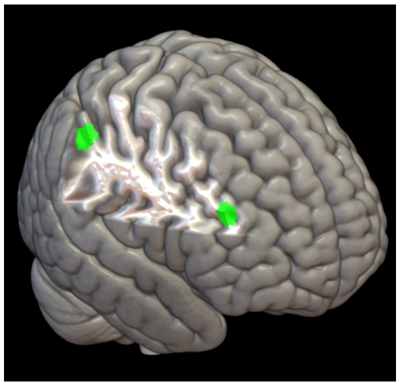
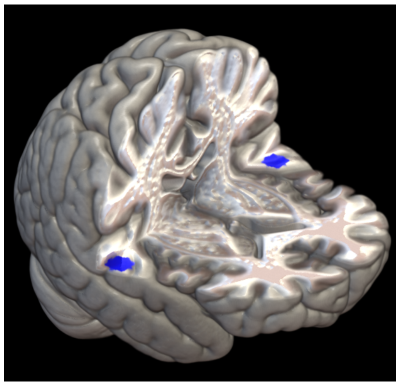
Whelan, R., Watts, R., Orr, C. A., Althoff, R. R., Artiges, E., Banaschewski, T., Barker, G. J., Bokde, A. L. W., Buchel, C., Carvalho, F. M., Conrod, P. J., Flor, H., Fauth-Buhler, M., Frouin, V., Gallinat, J., Gan, G., Gowland, P., Heinz, A., Ittermann, B., Lawrence, C., Mann, K., Martinot, J.-L., Nees, F., Ortiz, N., Paillere-Martinot, M.-L., Paus, T., Pausova, Z., Rietschel, M., Robbins, T. W., Smolka, M. N., Strohle, A., Schumann, G., Garavan, H., the IMAGEN Consortium, 08 2014. Neuropsychosocial profiles of current and future adolescent alcohol misusers. Nature 512 (7513), 185–189.
URL http://dx.doi.org/10.1038/nature13402

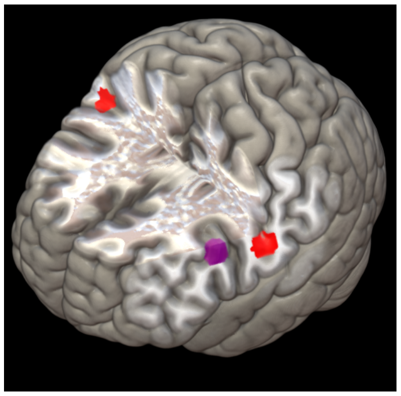
Table A.1: Table of ROI

ROI	ICN	ROI Description	Visualization
1	1	left anterior hippocampus	
2	1	right anterior hippocampus	
3	2	subgenual anterior cingulate cortex, anterior caudate	
4	2	ventromedial prefrontal cortex, medial frontal gyrus	
5	3	right globus pallidus	
6	3	left globus pallidus	
7	4	right anterior insula	
8	4	left anterior insula	
9	4	middle cingulate cortex, dorsomedial prefrontal cortex	

ROI	ICN	ROI Description	Visualization
10	5	inferior cerebellum	
11	5	posterior cingulate cortex	
12	5	inferior vermis	
13	5	inferior vermis	
14	5	anterolateral cerebellum	
15	5	anterolateral cerebellum	
16	5	posterior cerebellum	
17	5	inferior colliculus, anterior vermis	
18	5	fornix (body)	
19	6	posterior dorsomedial prefrontal cortex	
20	6	left superior precentral gyrus	
21	6	right posterior superior parietal cortex	

ROI	ICN	ROI Description	Visualization
22	7	left superior parietal cortex	
23	7	right precuneus	
24	7	left superior parietal cortex	
25	7	right superior parietal cortex	
26	7	left posterior dorsolateral prefrontal cortex	
27	7	right posterior dorsolateral prefrontal cortex	
28	8	left postcentral gyrus	
29	8	paracentral lobule	
30	8	anterior inferior parietal cortex	
31	8	right postcentral gyrus	
32	10	right lateral occipital cortex	
33	10	left lateral occipital cortex	
34	11	left occipital pole	
35	11	right occipital pole	
36	12	lingual gyrus	
37	12	right cuneus	
38	12	right fusiform gyrus	

ROI	ICN	ROI Description	Visualization
39	13	posterior cingulate cortex	
40	13	left angular gyrus	
41	13	right angular gyrus	
42	13	anterior dorsomedial prefrontal cortex	
43	14	right superior cerebellum	
44	14	vermis	
45	14	left superior cerebellum	
46	15	right middle frontal gyrus	
47	15	right supramarginal gyrus	
48	16	left superior temporal gyrus	
49	16	right superior temporal gyrus	

ROI	ICN	ROI Description	Visualization
50	17	right inferior pre and post central gyrus	
51	17	left inferior pre and post central gyrus	
52	18	left inferior frontal gyrus	

4 Symbolically regressing satellite imagery.

Large-scale imaging of large-scale regions of the Earth's surface presents many opportunities for advances in understanding. In a series of manuscripts we have shown that symbolic regression is particularly useful in this domain.

In [9] we demonstrated that symbolic regression could be adapted such that it draws intelligently on sensors that are more or less costly to query. The resulting model can successfully adapt its prediction as the cost and/or availability of different sensing systems fluctuate over time.

In [3] we demonstrated that symbolic regression is particularly useful for predicting an environmental variable of interest—in this case, the amount of water contained in snow pack—across a wide region.

Finally, in [6] we have demonstrated that symbolic regression can act as a form of compressed sensing: it can be successfully trained even if the data set is underdetermined (more features than observations). This works by enabling symbolic regression not just to optimize the structure of the model, but also to optimize the structure of sub-regions across the surface of the Earth to be modeled within which observations are averaged (compressed). This produces particularly intuitive models, because a lay observer can gain intuition into the model by observing what regions it draws averages from for prediction.

4.1 Relevance for U.S. defense and security.

It would be useful for military personnel or stakeholders to not only obtain predictions about environmental variables that change over time—snow, vegetation, rainfall—but to easily understand how a model predictions those variables may change in the near future. The methods presented in the three manuscripts that follow provide one such method to do so.

4.2 Yousefi *et al.* "A Genetic Programming Approach..." (2015).

A technical manuscript describing how symbolic regression can balance prediction and cost effectiveness follows.

A Genetic Programming Approach to Cost-Sensitive Control in Resource Constrained Sensor Systems

Afsoon Yousefi Zowj
Dept. of Computer Science
University of Vermont
ayousef1@uvm.edu

Josh C Bongard
Dept. of Computer Science
University of Vermont
jbongard@uvm.edu

Christian Skalka
Dept. of Computer Science
University of Vermont
skalka@cs.uvm.edu

ABSTRACT

Resource constrained sensor systems are an increasingly attractive option in a variety of environmental monitoring domains, due to continued improvements in sensor technology. However, sensors for the same measurement application can differ in terms of cost and accuracy, while fluctuations in environmental conditions can impact both application requirements and available energy. This raises the problem of automatically controlling heterogeneous sensor suites in resource constrained sensor system applications, in a manner that balances cost and accuracy of available sensors. We present a method that employs a hierarchy of model ensembles trained by genetic programming (GP): if model ensembles that poll low-cost sensors exhibit too much prediction uncertainty, they automatically transfer the burden of prediction to other GP-trained model ensembles that poll more expensive and accurate sensors. We show that, for increasingly challenging datasets, this hierarchical approach makes predictions with equivalent accuracy yet lower cost than a similar yet non-hierarchical method in which a single GP-generated model determines which sensors to poll at any given time. Our results thus show that a hierarchy of GP-trained ensembles can serve as a control algorithm for heterogeneous sensor suites in resource constrained sensor system applications that balances cost and accuracy.

Categories and Subject Descriptors

I.2 [**Computing Methodologies**]: Artificial Intelligence

Keywords

Genetic Programming, Resource Constrained Sensor Systems, Cost-Sensitive Control, Sensor Fusion

1. INTRODUCTION

Resource constrained sensor systems (RCSS) such as Wireless Sensor Networks have revolutionized environmental monitoring by combining low cost with flexibility in sensor capabilities [29]. They have been used in diverse environmental monitoring applications and continue to be adapted in new fields. Because RCSS are often, even typically, deployed in remote locations, and thus rely on combinations of battery power and energy harvesting, a major challenge in RCSS design is to minimize system power consumption.

Minimizing power consumption can be accomplished in a variety of ways, in particular by adapting sensor control strategies that optimize the balance between measurement accuracy and the cost of powering sensors [28]. In this paper, we propose new sensor control algorithms for RCSS with heterogeneous sensor suites that balance cost and accuracy, obtained using genetic programming (GP) techniques.

By "heterogeneous sensor suite", we mean RCSS equipped with multiple types of sensors for prediction of the same phenomena. Each of these sensors is characterized by its accuracy in relation to the phenomena, and a cost of use which is often measured by its power consumption. Such systems support multi-modal sensor fusion, a well-studied technique where data from multiple sensor modalities (types) is combined to predict a single variable [28]. The contribution of our work is a consideration of cost in multi-modal sensor fusion, and the development and testing of associated control algorithms. These algorithms will call upon particular sensors only when needed, and otherwise rely on the cheapest available sensors at any given time. Our problem is distinguished from adaptive sampling [28], in that the latter is concerned with optimally modulating sampling frequency of a given sensor, not choosing between a suite of possible sensors.

While various multi-modal sensor fusion applications exist, we are especially interested in the Snowcloud system which combines snow density telemetry with snow depth and air temperature sensors to predict areal snow water equivalent (SWE) [22]. We envision extending Snowcloud to incorporate ground based light detection and ranging (LIDAR) scanning [4] to be used for SWE estimation as part of its sensor suite. However, while LIDAR yields more accurate data than existing Snowcloud telemetry, it does so at significant additional power cost. Thus, the challenge is to commit these resources only at optimal times. It is also a refinement of multi-modal sensor fusion, since we are mainly interested in settings where available data gathering techniques differ in accuracy, with less accurate sensors being cheaper than more accurate ones.

A fundamental component of our approach is the use of prediction *uncertainty* to drive sensor usage. We propose a scheme whereby predictions are attempted using lower-cost sensors at first. If uncertainty is below an acceptable threshold, then the prediction is used. Otherwise we switch to higher-cost sensors, make a new prediction based on those inputs, evaluate uncertainty again, and continue to move the burden of prediction to more accurate and costly sensors as needed. This scheme is discussed in detail in Section 3.4 and described graphically in Figure 2. Note that while the Snowcloud system is an intended application of this scheme, it

GECCO '15, July 11 - 15, 2015, Madrid, Spain
© 2015 ACM. ISBN 978-1-4503-3472-3/15/07...$15.00
DOI: http://dx.doi.org/10.1145/2739480.2754751

can be generalized to any RCSS application using heterogeneous sensor suites comprising sensors with varying cost and accuracy.

To quantify uncertainty we are aided by machine learning ensemble methods– we use entropy in ensemble predictions as a proxy for uncertainty [21]. To obtain predictive models themselves, in this work we use genetic programming (GP). This is largely due to characteristics of our intended application space. Previous work has demonstrated that the relationships between snow cover and the topographic and meteorological factors that influence it include nonlinearities [24], while the spatial distribution of SWE is nonlinear because it is influenced simultaneously by various forcing effects [25]. Nonlinear predictors are therefore desirable. Furthermore, recent results [6] show that GP has advantages over other approaches (such as C4.5) due to associated techniques for preventing overfitting, e.g. treating model size minimization as an objective [11]. Although C4.5 only supports classification, sufficiently fine classification granularity can achieve competitive performance on regression problems, and this approach is popular in the environmental science community [6]. Finally, GP is appealing due to its white-box nature: it can potentially provide physical insights into modeled phenomena.

An alternative approach to our problem is to not rely on external measures of entropy to switch between sensors, but to treat cost as an additional objective in a multi-objective optimization problem. We explore this option in our work, in direct comparison to the hierarchical approach. However, due to the "curse of dimensionality", adding another optimization dimension may have deleterious effects on prediction performance, especially since selection for size to avoid overfitting already imposes a multi-objective optimization regime [5]. We therefore hypothesize that a hierarchical approach will outperform a non-hierarchical approach in settings where multiple sensors with differing predictive abilities, and we explore this comparison in our experiments.

2. RELATED WORK

Previous work on adaptive sampling [28] has aimed to reduce sampling rates in RCSS applications to balance sensor cost and accuracy. In particular, Alippi *et al.* [3] have tried to find the optimal adaptive frequency of sampling for avalanche monitoring. It has further been claimed that compressed sensing — sending aggregated data instead of raw data — performs better in conjunction with reducing sampling rates, rather than just reducing the sampling rate alone [15]. A variety of methods for compressed sensing [8] have been proposed. Although these methods have achieved cost reduction in monitoring, they are not applicable to our problem since we intend not to change the rate of sampling one sensor type, but rather to reduce sampling cost by switching between available sensors of different type and accuracy.

Another line of work focuses on finding the optimal location for sensors in distributed deployments, in order to maximize accuracy while minimizing deployment densities. Krause *et al.* [13] have used a probabilistic method to predict the communication cost for a given deployment topology. Papadimitriou *et al.* [17] have employed GP and a Bayesian statistical method to minimize entropy over a set of sensor locations. In contrast, our work is concerned with reducing the cost of sampling from an available set of sensors at any given time, not with reducing the densities of sensor topologies.

In work on so-called multi-modal sensor fusion, data from multiple sensors in a potentially heterogeneous suite are aggregated to monitor a specific measurement application [26, 9]. This method has been widely used, for example in visual monitoring [16, 18] and target tracking [19, 23]. Data-fusion focuses on sensor applications that need to compute the correlation between multiple sensor modules and cannot be measured by a single sensor. However, these works do not consider the cost of using different sensors, or minimizing cost.

Cost sensitive multi-modal sensor fusion methods have been developed to balance cost against accuracy, with an eye towards providing fault tolerance [12]. However, we are not concerned with fault tolerance, but strictly between selecting sensors from heterogeneous suites. Willett *et al.* [28] use a small number of sensors to send their readings to a fusion center, and based on the correlation among the sensed data, the fusion center decides which additional sensors should be activated. The same concept has also been tried in a distributed fashion [14]. However, sensing costs in these cases are a function of the number of sensors sampled, not their type.

Perhaps most related to our work is that of Wang *et al.* [27]. They propose a method to find the optimal set of sensors to be polled, using a hybrid tree, where non-leaf nodes act as a decision tree and leaves are standard regression models using a subset of sensors. However, these trees support decision making based on external constraints, i.e. which sensors to use depending on an organization's goals and resources. In contrast, our models are intended to support sensor control in RCSS during deployments.

Outside of the adaptive sampling and sensor fusion fields, multi-objective optimization has been used for cost-sensitive modeling. For example Kim [11] set error as one objective and tree size as another, as we do here. Zhao [30] sets the false negative rate and false positive rate as the two objectives. However, these works do not consider the hierarchical approach that we do.

3. METHODS

This section provides a formalization of the problem, how genetic programming is applied to solve it, and the two variants of genetic programming that we compare in this work. All of the material for replicating the work described here is available online [1].

3.1 Problem Formalization

Let us assume that t values of some environmental phenomenon $\mathbf{g}$ (the ground truth) are known at time steps $1, \ldots t$. These values are stored in $\mathbf{g} = g_1, \ldots g_t$. Let us further assume there are k sensors $s_1, \ldots s_k$ available that can be used to predict $\mathbf{g}$. Let $r_i^{(t)}$ denote the reading of sensor i taken at time t. Moreover, let $s^{(t)}$ and $r^{(t)}$ denote a subset of sensors, and readings taken from them, at time t. We denote the amount of variance of $\mathbf{g}$ explained by sensor i as $v_{r_i}^{(\mathbf{g})}$. This value is determined by linearly regressing only r_i against g. Finally, let $e_i = 100(1 - v_{r_i}^{(\mathbf{g})})$ and c_i represent the prediction error and cost of using sensor i respectively. Using this formulation, e_i represents the percentage of prediction error incurred by just using sensor i to predict $\mathbf{g}$.

The cost of a sensor c_i is usually inversely proportional to its error e_i, so for the work reported below, we set $c_i = v_{\mathbf{r}_i}^{(\mathbf{g})}$ for each sensor. In certain sensor deployments there may be other factors that affect c_i such as power consumption, market price, effort required to collect a sensor's reading, proprietary issues, and so on.

We suppose that an ordering of sensors exists such that s_1 is the least expensive sensor with the highest error and s_k is the most expensive sensor with the lowest error. Formally,

$$\forall i, j \,.\, 1 \leq i < j \leq k \rightarrow e_i > e_j \wedge c_i < c_j.$$

Let us denote the prediction of a model using a subset of sensors at time t by $p^{(t)}$, i.e., $p^{(t)}$ is a function on $\mathbf{r}^{(t)}$. Then, the error of each sampling $e^{(t)}$ would be

$$e^{(t)} \triangleq |p^{(t)} - g^{(t)}|.$$

The cost of each sampling, $c^{(t)}$ is the cumulated cost of all sensors $s_i \in \mathbf{s}^{(t)}$ that were polled for that sampling:

$$c^{(t)} \triangleq \sum_{j \in \{i | s_i \in \mathbf{s}^{(t)}\}} c_j.$$

It is desired that each sampling $\mathbf{s}^{(t)}$ entails low error and cost. That is, the following equality is desirable:

$$\operatorname*{argmin}_{\mathbf{s}^{(t)}} e^{(t)} = \operatorname*{argmin}_{\mathbf{s}^{(t)}} c^{(t)}.$$

Our goal is to design models which combine and transform sensor readings to accurately predict the outcome measure, but can also intelligently determine which sensors to poll when cheap, less accurate sensors exhibit uncertainty about the current prediction.

3.2 General Genetic Programming approach

Genetic programming has widely been employed for regression tasks in which the functional form of the equations relating inputs to outputs is unknown. Here, inputs are sensor values and output is a prediction for a given outcome measurement.

Although many recent improvements have been proposed for GP, here we have kept the genetic programming algorithm simple and instead focused on comparing GP-generated hierarchical and non-hierarchical models. Thus, GP is restricted to the four simple algebraic operators, and each evolutionary trial is initialized with a fixed-sized population of randomly-generated solutions containing three nodes. Maximum tree depth is not set since the tree size is considered as an objective in multi-objective optimization. The crossover rate is set to 0.2 and no fitness stall is considered. If the number of non-dominated solutions reaches 50% of the population size, the training restarts. At the conclusion of each generation, four values are computed for each solution: (1) *error* on training data as defined below, (2) the combined *cost* of the sensors used to make the prediction, (3) the *size* of the solution, and (4) the *age* of the solution. We now discuss each in turn.

Error: Let n be the population size and j range over $\{1, \cdots, n\}$. Let t_j be some solution tree. We represent the error of sampling at time t using solution t_j with $e_{t_j}^{(t)}$. Moreover, d^{train} and d^{test} denote the training dataset and testing dataset, respectively. Then, we define the error on training data using solution t_j by $e_{t_j}^{\text{train}}$ and as the average of $e_{t_j}^{(t)}$ on all samples in d^{train}, i.e.,

$$e_{t_j}^{\text{train}} \triangleq \sum_{g^{(t)} \in d^{\text{train}}} \frac{e_{t_j}^{(t)}}{|d^{\text{train}}|}.$$

Each solution t_j was allowed to use a subset (possibly empty) of available sensors. The cost of each solution depends on the sensors that are employed and the sampling.

Cost: As described in the following sub-sections, the current readings of the sensors may trigger readings from additional sensors. Thus, different $r_i^{(t)}$ may cause t_j to need different $\mathbf{s}^{(t)}$. The average cost of a tree on training data $c_{t_j}^{\text{train}}$ is thus defined as the cost of all of the sensors that have been used to predict the outcome for each training instance, averaged over all instances in the training data set:

$$c_{t_j}^{\text{train}} \triangleq \sum_{\mathbf{r}^{(t)} \in d^{\text{train}}} \sum_{l \in \{i | s_i \in \mathbf{s}^{(t)}\}} \frac{c_l}{|d^{\text{train}}|}.$$

If a solution uses a sensor more than once, no extra cost is incurred: because the sensor has already been polled, its output is already available and can thus be re-used as often as required.

Size: To avoid bloat, solution size was incorporated into the fitness objectives during the optimization process [7].

Age: We employed the Age-Fitness Pareto Optimization (AFPO) method [20], which injects a new randomly-generated solution into the population at each generation and compares the solutions with same age in an effort to guard against convergence. Each solution's age is defined as the number of generations since its oldest ancestor was injected into the population. A new solution produced by mutating an existing solution inherits the same age as its parent. If two existing parents are crossed to produce two new offspring, the offspring inherit the age of the older of the two parents. AFPO is an multiobjective optimization method as solution age is used as an additional fitness objective during optimization.

Optimization. At the end of each generation, the Pareto front is computed according to the objectives used, and the dominated solutions are discarded. Multi-objective optimization with all four objectives described above could easily lead to population collapse in the sense that all members of the population could become non-dominated. To guard against this eventuality, one possibility is to restart the evolutionary run with new solutions if no dominated solutions are detected in the population at the end of a given generation. Alternatively, a very large population size can be employed. However, both of these solutions greatly increase the computational effort required to obtain satisfactory solutions to the given problem. To avoid this situation, different multi-objective optimization approaches has been proposed. One of the simplest non-parametric approaches is to reduce the number of objectives by multiplying objectives together and using the result in the optimization process [10]. In this experiment, since error is the most important outcome, error is used for the primary objective and the second objective is the result of multiplying cost, size and age together.

Once the dominated solutions are deleted, the empty slots in the population are then filled by mutating and crossing copies of the non-dominated solutions. Tournament selection is used to select parents from the front for these operations. After the last generation, age is discarded when computing members of the Pareto front, since the goal is to use only small, accurate and cost-effective solutions for prediction, regardless of their age.

3.3 Non-hierarchical GP

A naive approach to cost-sensitive modeling using GP would be to evolve individual trees that add conditional and comparative operators to the base set of operators, and allow the tree to poll the values of all sensors if desired, as shown in Figure 1.A. In this way, different parts of the solution tree will be visited depending on the current values of the sensors. Successful solutions may evolve which only visit nodes containing references to expensive sensors—which are then polled—if less expensive sensors report certain combinations of values that signal these sensors are unlikely to predict well given the current circumstances. $e_{t_j}^{\text{train}}$ and $c_{t_j}^{\text{train}} * \text{age} * \text{size}$ are employed as the two main objectives in the optimization process.

Figure 1.B shows an hypothetical example of a GP solution t_j that has evolved to encode a useful conditional. In this example, an inexpensive sensor s_1 is first polled. If its reported value $r_1^{(t)}$ is below some threshold, the reading of a more expensive sensor s_2 is going to be used. It is assumed here that s_1 tends to make poor predictions of the outcome if its reading is below 1.43. If this threshold is exceeded, $r_1^{(t)}$ is then used to predict the outcome.

Conditional operators should, indirectly, encode the differential effects on the available sensors, and the relative costs of those sen-

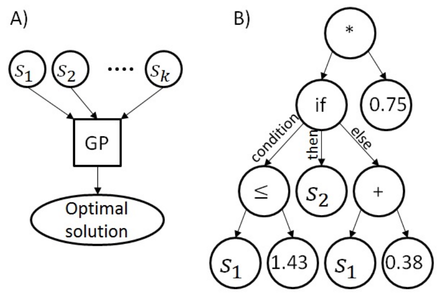


Figure 1:A)Non-hierarchical framework.B)A non-hierarchical sample solution.

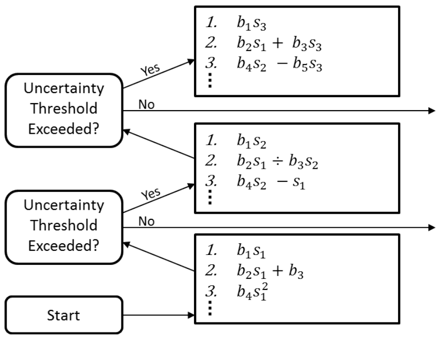


Figure 2: Hierarchical framework.

sors. Note that this is possible even if GP does not have direct access to these differential effects and costs, as they are indirectly reflected in the errors and costs incurred when each solution is evaluated. This issue is worth mentioning in that these effects are complex, non-linear and noisy, and even field experts cannot define them precisely.

3.4 Hierarchical GP

An alternative approach to reconciling prediction error and prediction cost is to build a hierarchy of models: models in the lower layers only have access to inexpensive sensors, while models in the upper layers have access to a greater subset of the sensors, including more expensive ones. When deployed, the overall model returns a prediction from a lower layer if the inexpensive sensors are confident of their combined prediction. If they are not, predictions are drawn from a higher layer.

Briefly, constructing such a model proceeds in two phases:

1. First, build a set of k layers, one for each sensor modality. For each layer i, run GP to find a set of accurate and low-cost solutions that use one or more sensors from the set $s_1, s_2, \ldots s_i$.

2. Define conditions which determine which layer should be allowed to provide the prediction, given the current environmental conditions.

Figure 2 illustrates what such a hierarchical model looks like. At the outset of attempting to provide a prediction for the current environmental conditions, the models stored in the lowest layer are evaluated, which only have access to the least expensive sensor s_1. If the certainty of their combined predictions is acceptable, return the combined prediction of these models. Otherwise, evaluate the models at the next layer, which have access to s_1 and the next least expensive sensor s_2. If these models are acceptably confident in the prediction, return their combined prediction; otherwise, evaluate the solutions at the next layer, and so on. If the top layer is reached, the combined predictions of the models found there are returned as the overall prediction, regardless of their level of certainty. The incremental construction of these models is described next.

Starting with the least expensive sensor s_1, GP is used to find the best models for converting $r_1^{(t)}$ to $g^{(t)}$. When GP terminates, the final non-dominated solutions are then organized as a group named layer L_1. The same process is repeated for s_2, except for the fact that since s_1 is already polled in L_1, it may be incorporated into models during evolution without incurring an extra cost for the solution tree that makes use of it. Similarly, for each sensor s_i, a separate GP run is performed with sensors s_1 to s_i available as input to construct layer L_i. These layers are then organized in a hierarchical fashion. The order of layers is based on the cost of the most expensive sensor they are representing, from L_1 to L_k. Suppose each layer L_i consists of n_i solutions and the jth solution t_j in L_i is denoted as $t_{i,j}$. Let $p_{t_{i,j}}^{(t)}$ denote the prediction of $g^{(t)}$ that $t_{i,j}$ provides. Then, the final prediction of layer L_i for $g^{(t)}$ is

$$p_{L_i}^{(t)} \triangleq \sum_{j=1}^{n_i} \frac{p_{t_{i,j}}^{(t)}}{n_i}.$$

The error that corresponds to $p_{L_i}^{(t)}$ is

$$e_{L_i}^{(t)} \triangleq |p_{L_i}^{(t)} - g^{(t)}|.$$

In the second phase, a conditional must be formulated to determine whether the current layer should return its prediction, or whether the burden of prediction should be passed up to the next layer. One common method for measuring how confident an ensemble of models is, is to compute the variance in their predictions [21]: if variance is low, and those models are sufficiently independent of one another, there is a greater likelihood that their combined predictions can be trusted. If variance is high, this is likely the result of differing assumptions encoded in the models, which cannot all be true reflections of the hidden relationship being modeled. Note the assumption here that the models are relatively independent: a set of identical models will never exhibit a variance in their predictions, regardless of how accurate the individual models are. We can be somewhat confident of the independence of our models, as they are produced by the AFPO algorithm: models with differing ages are likely to arrive on the final Pareto front used to build each layer, and such differently-aged genomes are likely to be somewhat independent because of their different genetic origins.

Formally: Let $p_{L_i}^{\text{train(t)}}$ and $e_{L_i}^{\text{train(t)}}$ denote $p_{L_i}^{(t)}$ and $e_{L_i}^{(t)}$ using $\mathbf{r}^{(t)}$ on d^{train}, respectively. Similarly, $p_{L_i}^{\text{test(t)}}$ and $e_{L_i}^{\text{test(t)}}$ respectively denote $p_{L_i}^{(t)}$ and $e_{L_i}^{(t)}$ using $\mathbf{r}^{(t)}$ on d^{test}. Moreover, assume $v_i^{\text{train(t)}}$ and $v_i^{\text{test(t)}}$ are the variances of all $p_{t_{i,j}}^{(t)}$s on d^{train} and d^{test}. Also, v_i^{train} denotes $v_i^{\text{train(t)}}$ averaged over all the samplings in d^{train}.

To determine whether the burden of prediction should remain with the current layer or passed off to a higher layer, we measure the difference in prediction variance between the models when presented with the training data (v_i^{train}) or with the testing data, i.e. the current environmental conditions ($v_i^{\text{test(t)}}$). When $v_i^{\text{test(t)}}$ is almost the same as v_i^{train}, there is a high probability that $e_{L_i}^{\text{test(t)}}$ is an approximation of $e_{L_i}^{\text{train(t)}}$, and we can be relatively confident that these

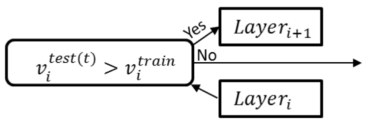


Figure 3: Using the difference between training data prediction variance and test data prediction variance as the condition for switching between model layers.

models will yield a good collective prediction of $g^{(t)}$. When the variance of test data prediction is significantly higher than prediction on the training data, this signals that the solutions in that layer are exhibiting increased disagreement regarding the current environmental conditions. This could be due to the fact that a specific sensor is not physically able to predict under the current conditions, or the solutions have not been trained for the current situation. In such an eventuality it would be advantageous to switch to the next layer, in the hope that its models will exhibit more confidence in their ability to predict the current conditions. In this paper, the variance is considered as a proxy for entropy, but any other entropy related metric could be used instead. Figure 3 illustrates how this intuition is encoded into the switching condition in the hierarchy of layers.

By considering the amount of difference between prediction variance on training and testing data, we can dynamically tune how conservative or liberal the overall hierarchical model is: if little difference is tolerated, the burden of prediction will often be passed to higher layers, resulting in expensive yet accurate predictions; if much difference is tolerated, lower levels will tend to predict, resulting in less expensive and less accurate predictions. The advantage of this approach is that the amount of tolerance could be dynamically tuned based on the current available budget for sensing.

For example, for larger budgets, more cost could be expended in order to obtain more accurate results. In this regard, the tolerance of the difference between variances could be decreased, transferring the burden of prediction to higher layers. Similarly for small budgets, the tolerance would be increased. Through this adjustment, more disagreement would be tolerated and less accurate predictions would be obtained for lower cost. To implement this dynamic tuning given a fluctuating budget, a tolerance parameter $\tau \in [0, 1]$ is defined, reflecting the tolerance of disagreement between the solutions of a given layer. Equation (1) demonstrates how this parameter is used to determine which level should be activated for prediction.

$$p^{(t)} = \begin{cases} p^{(t)}_{L_i} & \text{if } v_i^{\text{train}} < |1-\tau| \cdot v_i^{\text{test}(t)} \\ p^{(t)}_{L_{i+1}} & \text{otherwise} \end{cases} \tag{1}$$

It should be noted that in the present work, the same value for τ is used at the interstices between each pair of layers. However, different values for τ could be employed between different layers to enable the model to respond better to changes in the overall available budget. The extreme cases occur when $\tau = 0$ or $\tau = 1$. The former ensures that the conditional is only true when the prediction variance on the testing data is greater than the prediction variance on the training data which has a high probability of occurring. Thus, the method tends to extract the predictions from the solutions on the uppermost layer. The latter ensures that the conditional is only true whenever the variance on the testing data is finite, which is always true. In this case, the first layer always provides the prediction. Values greater than $\tau = 1$ are not investigated in this work, but are possible. Greater τ value increases the probability of the conditional to be true. $\tau = \infty$ causes the conditional to always be true, thus the method always collects predictions from the last layer.

Table 1: Available sensors and their features.

Name	Equation Template of $r_i^{(t)}$	Cost
s_3	$g^{(t)}$	3
s_2	$b_{2,1}g^{(t)} + b_{2,2}$	2
s_1	$b_{1,1}(g^{(t)})^2 + b_{1,2}g^{(t)} + b_{1,3}$	1

4. RESULTS

The proposed methods are evaluated over two set of experiments, using a synthesized dataset and ten actual datasets. This section summarizes these datasets, experimental setups, and quantitative results.

4.1 Synthesized Data

In these experiment, the proposed methods have been evaluated on a synthetic system monitored by three different sensors. Table 1 shows these three sensors, their readings in relation to $g^{(t)}$, and their cost.

To create the training and testing datasets, at first coefficients in the equations of the sensor relations, i.e., $b_{i,j}$s, were randomly selected in the range $[0, 1]$. Then, random numbers were generated for $g^{(t)}$, and used to calculate the sensor readings based on the given template and selected coefficients. The training and testing dataset sizes were 150 and 50, respectively, and each experiment were repeated 40 times.

Non-hierarchical setup. The population size is 100 and is trained for 300 generations. The optimization process during the last generation does not consider *age* as an objective and Pareto front is selected using *error* and *cost* $\times$ *size* as two separate objectives. After training, the knee of the non-dominated solutions is selected and tested using the testing dataset.

Hierarchical setup. The population size for each layer is 100 and each layer was trained for 100 generations, for the sake of fairness in comparisons. Similarly to the non-hierarchical setup, during the last generation, *age* is not considered in the Pareto optimization process, and non-dominated solutions are selected based on *error* and *cost* $\times$ *size* as two separate objectives. After training, for each layer L_i, the variance of the solutions output on training data v_i^{train} is computed and stored as the threshold of switching to the next layer L_{i+1}. This variance is not computed for the layer corresponding to the most expensive sensor, i.e., L_3, since there are no more sensors to be called. The experiment was repeated 40 times for each of the different tolerance parameters $\tau = 0.0, 0.1, 0.2, 0.4, 0.6, 0.8$.

4.1.1 Results on Synthesized Data

Average error. The average error of the non-hierarchical method is $e^{\text{test}}_{t_j}$, where t_j is the final selected solution. The average error of the hierarchical method is the average of $e^{\text{test}}_{L_i}$, where L_i is the last layer reached in the hierarchy, during the sampling. As can be seen in Figure 4, the largest difference in error occurs at maximum tolerance i.e. $\tau = 0.8$ where the error of the hierarchical method is 1.34% higher than the non-hierarchical method. The hierarchical method tends to achieve lower average error when the tolerance parameter is $\tau < 0.4$. P-values obtained for different tolerance

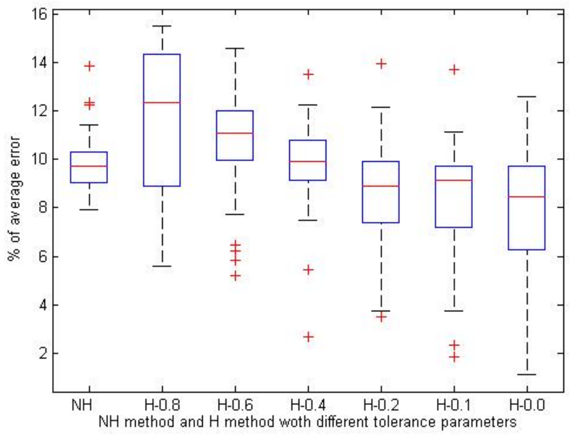


Figure 4:Average error on the test data for the non-hierarchical and the hierarchical methods with different tolerance parameters. Statistical significance of these results are represented in Table 2.A)

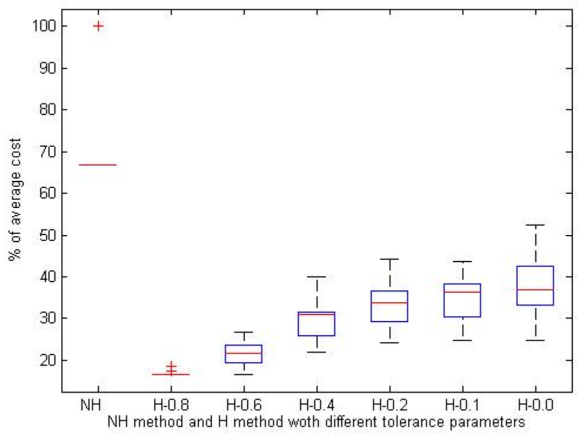


Figure 5:The average cost on the test data for the non-hierarchical and the hierarchical methods with different tolerance parameters. Statistical significance of these results are represented in Table 2.B)

Table 2: A) P-values considering error of the non-hierarchical and the hierarchical methods with different tolerance parameters. B) P-values considering cost of the non-hierarchical and the hierarchical methods with different tolerance parameters.

A	P-values	B	P-values
$\tau = 0.8$	0.013414	$\tau = 0.8$	$\ll 0.001$
$\tau = 0.6$	0.046626	$\tau = 0.6$	$\ll 0.001$
$\tau = 0.4$	0.635566	$\tau = 0.4$	$\ll 0.001$
$\tau = 0.2$	0.001309	$\tau = 0.2$	$\ll 0.001$
$\tau = 0.1$	$\ll 0.001$	$\tau = 0.1$	$\ll 0.001$
$\tau = 0.0$	$\ll 0.001$	$\tau = 0.0$	$\ll 0.001$

parameters are represented in Table 2.A) and show that $\tau = 0.4$ is the boundary where the hierarchical method begins to outperform the non-hierarchical method.

Average cost. By considering t_j as the final selected solution in the non-hierarchical method, the average cost is $c_{t_j}^{\text{test}}$. The average cost of the hierarchical method is the average of $c_{L_i}^{\text{test}}$, when the last layer reached during the sampling is L_i. In order to compare both methods and understand how much of the potential cost each method uses, the cost of each method is represented as the percentage of cost of using all available sensors. Figure 5 shows that the average cost of the hierarchical method is significantly lower than the non-hierarchical method (at most 54.88% and at least 33.81% lower cost). Table 2.B) summarizes the p-values to show how significantly the cost of the hierarchical method is lower than the non-hierarchical method.

4.2 Actual Data

In this experiment, ten datasets are selected from the UCI database repository [2] based on the number of instances and features from the regression section. Table 3 summarizes these datasets and their features. For these datasets, we treat the individual features as individual sensors. Each experiment in this section were repeated 30 times.

In order to determine the accuracy of each sensor s_i in predicting $g^{(t)}$, the value of $v_{\mathbf{r}_i}^{(\mathbf{g})}$ is calculated for each available sensor of each dataset, using linear regression. The greater $v_{\mathbf{r}_i}^{(\mathbf{g})}$ is, the better that sensor can predict $g^{(t)}$.Table 4 summarizes the values of $v_{\mathbf{r}_i}^{(\mathbf{g})}$ for all of the sensors of the Auto MPG dataset, as an exampl. We define the cost of each sensor in these datasets as $v_{\mathbf{r}_i}^{(\mathbf{g})}$.

Table 3: Used UCI datasets.

DS No.	DS Name	No. of Instances	No. of sensors	$g^{(t)}$ Average
DS_1	Auto MPG	398	7	23.51457
DS_2	Housing	506	13	22.53281
DS_3	Forest Fires	517	12	0.031663
DS_4	Energy Efficiency	768	8	22.3072
DS_5	Concrete Compressive Strength	1030	8	35.81796
DS_6	Solar Flare	1389	9	0.300188
DS_7	Airfoil Self-Noise	1503	5	124.8359
DS_8	SkilCraft1 Master Table Dataset	3395	19	4.184094
DS_9	Wine Quality	4898	11	5.877909
DS_{10}	Parkinson's Telemonitoring	5875	17	29.01894

Table 4: Value of $v_{\mathbf{r}_i}^{(\mathbf{g})}$ for all the sensors of Auto MPG dataset.

DS No.	s_1	s_2	s_3	s_4	s_5	s_6	s_7
Auto MPG	0.1766	0.3175	0.3356	0.5951	0.6012	0.6467	0.6918

Table 5: Minimum and maximum amount of variance a sensor accounts for, in each dataset.

DS No.	min $v_{\mathbf{r}_i}^{(\mathbf{g})}$	max $v_{\mathbf{r}_i}^{(\mathbf{g})}$
DS_1	0.1766	0.6918
DS_2	0.0307	0.5441
DS_3	0.0002	0.2578
DS_4	0.0076	0.7911
DS_5	0.0112	0.2478
DS_6	0.000	0.096
DS_7	0.0157	0.1527
DS_8	0.0005	0.4542
DS_9	0.0001	0.1897
DS_{10}	0.0037	0.0263

Table 6: Average error percentages and the corresponding P-values for the hierarchical and the non-hierarchical methods.

DS No.	NH: error %	H: error %	P-value
DS_1	20.85	25.81	$\ll 0.001$
DS_2	25.90	28.93	$\ll 0.001$
DS_3	126.49	202.12	0.565
DS_4	29.19	36.70	$\ll 0.001$
DS_5	35.29	39.68	0.393
DS_6	110.63	111.08	0.223
DS_7	0.00	0.00	0.082
DS_8	37.59	28.65	0.194
DS_9	10.79	10.67	0.197
DS_{10}	32.11	29.88	0.423

Table 7: Average cost percentages and the corresponding P-values for the hierarchical and the non-hierarchical methods.

DS No.	NH: cost %	H: cost %	P-value
DS_1	38.89	12.33	0.022
DS_2	23.18	1.26	$\ll 0.001$
DS_3	6.83	15.81	$\ll 0.001$
DS_4	32.90	4.18	$\ll 0.001$
DS_5	53.63	28.63	0.004
DS_6	0.00	0.98	0.040
DS_7	11.58	7.35	0.005
DS_8	2.55	0.00	$\ll 0.001$
DS_9	0.02	0.00	$\ll 0.001$
DS_{10}	20.62	3.88	0.009

Non-hierarchical setup. The population size is 200 and for each dataset with k features, it is trained for $200 * k$ generations.

Hierarchical setup. The population size for each layer is 200. Similar to synthesized data experiments, In order to equalize search effort in both methods, each layer was trained for 200 generation. After training, a subset of the non-dominated solutions with least error are selected and organized in the corresponding layer. The cardinality of this subset is 2% of the population size. This experiment was conductedof for tolerance parameter $\tau = 0.1$. This value is selected based on the results in 4.1 and will be discussed in more detail in Section 5.1.

4.2.1 Results on Actual Data

Average error. The average error for the non-hierarchical and the hierarchical methods are $e^{\text{test}}_{t_j}$ and $e^{\text{test}}_{L_i}$ respectively, where t_j is the final selected solution in the non-hierarchical method and L_i is the last layer reached during the sampling in the hierarchical method. Table 6 summarizes the average error of both methods on all the datasets as a percentage of error. It can be seen that for some datasets, the average error of the hierarchical method is higher than the average error of the non-hierarchical method. However, the P-value for the two-tailed t-test shows that generally, this difference is not significant. There are three cases where the difference is significant i.e., DS_1, DS_2 and DS_4.

Average cost. Similar to 4.1.1, the average cost is represented as the percentage of the maximum possible cost. Table 7 summarizes the percentage of the average cost each method uses for prediction. The cost of the hierarchical method is significantly lower in all cases except for DS_3 and DS_6.

5. DISCUSSION

Our results in all experiments suggest that the hierarchical method is better at balancing cost and accuracy than the non-hierarchical approach. We believe this is because meaningful sensor control conditions for managing cost are complex and require considerable computational effort to be discovered. Using hand-tuned prediction uncertainty to drive sensor control is more effective. Furthermore, our results show that the latter approach better supports dynamic adaptation to changes in available energy, through modulation of tolerance. The non-hierarchical approach cannot adapt to such changes without retraining from scratch, or aggressive online learning. As mentioned in Section 3.2, in these experiments a basic genetic programming approach was deployed. We anticipate that if we were to use a more powerful underlying GP approach, the error of both hierarchical and non-hierarchical models would be reduced.

In the remainder of this Section we discuss results as they pertain specifically to experiments with synthesized and actual data.

5.1 Synthesized Data

Average error. As can be seen in Figure 4, the hierarchical method achieved significantly better accuracy than the non-hierarchical method for $\tau < 0.4$. In general, results show that higher tolerance allows the algorithm to accept more uncertainty in the prediction and rely on less expensive sensors which are less accurate. This avoids the use of more expensive sensors, but causes average error to rise. A tolerance of $\tau < 0.4$ is apparently the threshold where average error in the hierarchical method exceeds that of the non-hierarchical method.

Average cost. Results reported in Figure 5 show that the hierarchical method significantly outperforms the non-hierarchical method with regard to cost on this dataset, even when tolerance is low. This suggests that the use of variance in ensemble predictions to serve as a proxy for prediction uncertainty is not easy to learn, and serves as a good mechanism for control. Results in Figures 4 and 5 suggest that $\tau = 0.1$ is a "sweet spot" for balancing cost and accuracy, though the value could be increased or decreased if greater frugality or accuracy were needed, respectively.

5.2 Actual Data

For testing on actual data, we fixed $\tau = 0.1$ due to results on synthetic data demonstrating a nice balance between cost and accuracy with this tolerance level.

Average error. Table 6 shows that the average error of the hierarchical and the non-hierarchical methods were not significantly different, except for datasets DS_1, DS_2 and DS_4 where the latter method achieves better prediction accuracy. This is probably due to the characteristics of these datasets, where the difference between the least prediction variances $v^{(\mathbf{g})}_{\mathbf{r}_i}$s and the greatest ones is large. The majority of sensors in these datasets are not informative but have low costs and the remaining sensors are informative enough but come with very higher costs. Thus, lower levels of the hierarchy "struggle" compared to upper ones in terms of accuracy. Nevertheless, accuracy rate with the hierarchical method is still competitive even in these cases, and cost reduction is significant. Also, it can be seen that as the size of the datasets grows, the difference between the error rate of the non-hierarchical and the hierarchical methods decreases, and in the three largest datasets the hierarchical method also achieves lower error rates.

Average cost. The hierarchical method achieved significantly lower cost than the non-hierarchical method on all the real world datasets, as shown in Table 7, except for DS_3 and DS_6. As represented in Table 5, in these two datasets, just a small subset of sensors are relatively informative. Since the tolerance parameter for the hierarchical method is low, the hierarchical method employs more informative sensors. Taken together, results shown in Tables

6 and 7 clearly indicate an advantage of the hierarchical method for balancing cost and accuracy.

6. CONCLUSION AND FUTURE WORK

All resource constrained sensor systems have to face a trade-off between measurement accuracy and the cost of sensor sampling. In networks supporting multiple sensor types, it is therefore desirable to develop cost-sensitive control algorithms that sample more expensive sensors only when necessary. In this paper, a hierarchical method is proposed where GP solutions are sorted in a hierarchy of layers based on the cost of the sensors they use. Switching to the next more expensive layer takes place only if the prediction variance indicates uncertainty at lower layers. We compare this method to a non-hierarchical GP method where cost is treated as an additional optimization objective in fitness selection. In experiments using a synthesized dataset and ten real datasets, the hierarchical method is shown to have significantly lower prediction costs than the non-hierarchical method. As the datasets grow bigger and more complex, competitive and sometimes lower error rates are achieved by the hierarchical method. Future work includes consideration of how to dynamically tune the balance of cost and accuracy based on available energy and budget. Other directions for future work include methods for online learning to support adaptation of control algorithms to particular deployments, and application of hierarchical control algorithms in real resource constrained sensor system deployments.

Acknowledgements

This work was supported in part by the NSF awards PECASE-0953837 and INSPIRE-1344227.

7. REFERENCES

[1] github code public repository. http://git.io/vfmGB. Accessed: 2015-04-18.

[2] UCI machine learning repository. http://archive.ics.uci.edu/ml/datasets.html. Accessed: 2015-02-03.

[3] C. Alippi, G. Anastasi, C. Galperti, F. Mancini, and M. Roveri. Adaptive sampling for energy conservation in wireless sensor networks for snow monitoring applications. In *IEEE 4th International Conference on Mobile Adhoc and Sensor Systems, MASS 2007, 8-11 October 2007, Pisa, Italy*, pages 1–6, 2007.

[4] E. H. Bair, R. E. Davis, D. C. Finnegan, A. L. LeWinter, E. Guttmann, and J. Dozier. Can we estimate precipitation rate during snowfall using a scanning terrestrial lidar? In *International Snow Science Workshop*, pages 923–929, Anchorage, AK, 2012.

[5] Brockhoff, Dimo, Zitzler, and Eckart. Are all objectives necessary? On dimensionality reduction in evolutionary multiobjective optimization. In *Parallel Problem Solving from Nature-PPSN IX*, pages 533–542. Springer, 2006.

[6] D. Buckingham, C. Skalka, and J. Bongard. Inductive learning of snowpack distribution models for improved estimation of areal snow water equivalent. *Journal of Hydrology*, 2015. Accepted for Publication.

[7] E. D. de Jong and J. B. Pollack. Multi-objective methods for tree size control. *Genetic Programming and Evolvable Machines*, 4(3):211–233, 2003.

[8] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[9] D. Hall and J. Llinas. *Multisensor data fusion*. CRC press, 2001.

[10] G. Hornby. ALPS: the age-layered population structure for reducing the problem of premature convergence. In *Genetic and Evolutionary Computation Conference, GECCO 2006, Proceedings, Seattle, Washington, USA, July 8-12, 2006*, pages 815–822, 2006.

[11] D. Kim. Structural risk minimization on decision trees using an evolutionary multiobjective optimization. In *Genetic Programming*, pages 338–348. Springer, 2004.

[12] F. Koushanfar, S. Slijepcevic, M. Potkonjak, and A. Sangiovanni-Vincentelli. Error-tolerant multimodal sensor fusion. In *IEEE CAS Workshop on Wireless Communication and Networking*, pages 5–6, 2002.

[13] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *Proceedings of the 5th international conference on Information processing in sensor networks*, pages 2–10. ACM, 2006.

[14] S. Maleki, A. Pandharipande, and G. Leus. Two-stage spectrum sensing for cognitive radios. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2010, 14-19 March 2010, Sheraton Dallas Hotel, Dallas, Texas, USA*, pages 2946–2949, 2010.

[15] M. L. Malloy and R. D. Nowak. Near-optimal adaptive compressed sensing. *IEEE Transactions on Information Theory*, 60(7):4001–4012, 2014.

[16] A. Martinelli. Vision and IMU data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination. *IEEE Transactions on Robotics*, 28(1):44–60, 2012.

[17] C. Papadimitriou, J. L. Beck, and S.-K. Au. Entropy-based optimal sensor location for structural model updating. *Journal of Vibration and Control*, 6(5):781–800, 2000.

[18] C. Pohl and J. V. Genderen. Review article multisensor image fusion in remote sensing: concepts, methods and applications. *International Journal of Remote Sensing*, 19(5):823–854, 1998.

[19] H. Ren, D. Rank, M. Merdes, J. Stallkamp, and P. Kazanzides. Multisensor data fusion in an integrated tracking system for endoscopic surgery. *IEEE Transactions on Information Technology in Biomedicine*, 16(1):106–111, 2012.

[20] M. Schmidt and H. Lipson. Age-fitness pareto optimization. In R. Riolo, T. McConaghy, and E. Vladislavleva, editors, *Genetic Programming Theory and Practice VIII*, volume 8 of *Genetic and Evolutionary Computation*, pages 129–146. Springer New York, 2011.

[21] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 287–294, New York, NY, USA, 1992. ACM.

[22] C. Skalka and J. Frolik. Snowcloud: A complete data gathering system for snow hydrology research. In *Real-World Wireless Sensor Networks*, pages 3–14. Springer, 2014.

[23] D. Smith and S. Singh. Approaches to multisensor data fusion in target tracking: A survey. *IEEE Trans. Knowl. Data Eng.*, 18(12):1696–1710, 2006.

[24] H. Tabari, S. Marofi, H. Z. Abyaneh, and M. R. Sharifi. Comparison of artificial neural network and combined models in estimating spatial distribution of snow depth and snow water equivalent in Samsami basin of Iran. *Neural Comput. Appl.*, 19(4):625–635, 2010.

[25] U. Tappeiner, G. Tappeiner, J. Aschenwald, E. Tasser, and B. Ostendorf. GIS-based modelling of spatial pattern of snow cover duration in an alpine area. *Ecol. Model.*, 138:265–275, 2001.

[26] E. L. Waltz and D. M. Buede. Data fusion and decision support for command and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(6):865–879, 1986.

[27] D. Wang, H. Ahmadi, T. F. Abdelzaher, H. Chenji, R. Stoleru, and C. C. Aggarwal. Optimizing quality-of-information in cost-sensitive sensor data fusion. In *Distributed Computing in Sensor Systems, 7th IEEE International Conference and Workshops, DCOSS 2011, Barcelona, Spain, 27-29 June, 2011, Proceedings*, pages 1–8, 2011.

[28] R. Willett, A. Martin, and R. Nowak. Backcasting: adaptive sampling for sensor networks. In *Proceedings of the Third International Symposium on Information Processing in Sensor Networks, IPSN 2004, Berkeley, California, USA, April 26-27, 2004*, pages 124–133, 2004.

[29] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Comput. Netw.*, 52(12):2292–2330, 2008.

[30] H. Zhao. A multi-objective genetic programming approach to developing pareto optimal decision trees. *Decision Support Systems*, 43(3):809–826, 2007.

4.3 Buckingham *et al.* “Inductive machine learning...” (2015).

A technical manuscript describing how symbolic regression can be employed to predict environmental variables over large areas follows.

Journal of Hydrology 524 (2015) 311–325

Contents lists available at ScienceDirect

Journal of Hydrology

journal homepage: www.elsevier.com/locate/jhydrol

Inductive machine learning for improved estimation of catchment-scale snow water equivalent

David Buckingham *, Christian Skalka, Josh Bongard

Department of Computer Science, University of Vermont, Burlington, VT 05405, USA

A R T I C L E I N F O

Article history:
Received 5 December 2014
Received in revised form 18 February 2015
Accepted 21 February 2015
Available online 2 March 2015
This manuscript was handled by Konstantine P. Georgakakos, Editor-in-Chief, with the assistance of Kun Yang, Associate Editor

Keywords:
Snow water equivalent
Machine learning
Wireless sensor network
Snowpack modeling
Genetic programming

S U M M A R Y

Infrastructure for the automatic collection of single-point measurements of snow water equivalent (*SWE*) is well-established. However, because *SWE* varies significantly over space, the estimation of *SWE* at the catchment scale based on a single-point measurement is error-prone. We propose low-cost, lightweight methods for near-real-time estimation of mean catchment-wide *SWE* using existing infrastructure, wireless sensor networks, and machine learning algorithms. Because snowpack distribution is highly nonlinear, we focus on Genetic Programming (GP), a nonlinear, white-box, inductive machine learning algorithm. Because we did not have access to near-real-time catchment-scale *SWE* data, we used available data as ground truth for machine learning in a set of experiments that are successive approximations of our goal of catchment-wide *SWE* estimation. First, we used a history of maritime snowpack data collected by manual snow courses. Second, we used distributed snow depth (*HS*) data collected automatically by wireless sensor networks. We compared the performance of GP against linear regression (LR), binary regression trees (BT), and a widely used basic method (BM) that naively assumes non-variable snowpack. In the first experiment set, GP and LR models predicted *SWE* with lower error than BM. In the second experiment set, GP had lower error than LR, but outperformed BT only when we applied a technique that specifically mitigated the possibility of over-fitting.

1. Introduction

There has been extensive research on techniques for measuring and modeling snow because it affects many hydrological, atmospheric, and biological processes (Tappeiner et al., 2001). The accurate estimation of snow water equivalent at the catchment scale is useful in many applications, including agricultural planning, metropolitan use, flood risk evaluation, planning of hydropower production potential, weather forecasting, and climate monitoring (Marofi et al., 2011; Schmucki et al., 2014). More than 1/6 of people globally depend on seasonal snow or glaciers for water supplies (Bales et al., 2006), and in the western United States the majority of surface water resources is derived from snowmelt (Serreze et al., 1999). However, snow has declined across much of the US over the last half-century (Pierce et al., 2008). The current severe drought in California, with record low snowpack measurements over three years, threatens water supplies throughout the state (Boxalla, 2014) and highlights the importance of snowpack research. Snow both influences climate and responds directly to climate change (Engeset et al., 2004). While climate change warrants increased snowpack monitoring, existing techniques perform poorly under extreme climatic conditions (Molotch et al., 2005; Balk and Elder, 2000), and it has been argued that the stationarity of hydrological processes can no longer be assumed (Milly et al., 2008). Furthermore, high costs of data gathering constrain the temporal and spatial granularity of estimation methods. New techniques are needed.

We propose new low-cost techniques for estimating catchment-wide snow water equivalent using machine learning algorithms, especially genetic programming. These algorithms use data gathered from existing sensor infrastructure, and possibly short-term deployments of wireless sensor networks. The manipulation of large data sets in order to gain insight into snow accumulation, melt, and runoff has been highlighted as a necessary next step in mountain hydrology (Dozier, 2011). The long-term, overarching goal of our research project is to achieve better near-real-time (NRT), estimation of *SWE* at the catchment scale. By NRT, we mean automated reporting at fine-grained timescales, for example hourly. By better, we mean more accurate estimation without significantly increased infrastructure cost. Our strategy is to generate snow telemetry datasets using short-term, low-cost field campaigns that can be used by machine learning algorithms

* Corresponding author.
E-mail addresses: dbucking@uvm.edu (D. Buckingham), skalka@cs.uvm.edu (C. Skalka), josh.bongard@uvm.edu (J. Bongard).

http://dx.doi.org/10.1016/j.jhydrol.2015.02.042
0022-1694/© 2015 Elsevier B.V. All rights reserved.

to generate snowpack models. Following field campaigns and the termination of associated measurement techniques, these models can be used for NRT *SWE* estimations with no new instrumentation overhead.

The key idea behind our approach is that machine learning models are able to induce relationships between input parameters and an output value, if such exist, on the basis of the ground truth data if provided. The machine learning method we emphasize is genetic programming (GP), which generates equations relating a dependent variable to a set of independent variables.

In our case, we argue that if we obtain multiple years of "true" average *SWE* for a catchment, machine learning will be able to induce a meaningful mathematical relation between telemetry, such as proximal snow pillow reading(s), and true average *SWE*. Then, in years when true average *SWE* is not available, inputs such as snow pillow readings can be translated into average *SWE* estimates for the catchment. This approach assumes interannual continuity in snow distributions over a catchment, which has been demonstrated by previous research (Scipión et al., 2013; Tappeiner et al., 2001; Schirmer et al., 2011). Because accurate measurements of mean catchment *SWE* are generally unavailable at this time, we use snow course and wireless sensor network data as proxies for true average *SWE* to serve as ground truth for machine learning.

Thus, the ideal we aim for is a generally applicable technique for inducing models that take as input parameters existing infrastructure NRT telemetry, such as snow pillow readings, meteorological data, and date/time information, and output measurements of *SWE* at those locations. This would allow more accurate *SWE* estimation to be provided without additional cost beyond that of the initial field campaign for obtaining a ground truth dataset (Fig. 1).

Several theoretical and practical challenges exist on the way to achieving this goal. The purpose of this paper is to address them and make progress in three particular ways.

First, we explore the issue of what sort of machine learning approaches are best in this context. In general, we argue that techniques that are able to model nonlinear relationships are needed due to the known nonlinear nature of snow distribution in alpine environments (Tappeiner et al., 2001; Marofi et al., 2011). We also argue that so-called white-box tools are best, since these can provide physical insights for scientists (Schmidt et al., 2011). Furthermore, we emphasize resiliency against over-fitting, which is especially important given that the datasets available for machine learning may be relatively small.

Second, we investigate what sort of input parameters should be used by *SWE* estimation models, especially in light of practical concerns, i.e. available telemetry and datasets. In fact, availability of data is a key issue in this effort, and defines what is possible. We acknowledge the importance of terrain effects in determining snowpack distribution, influencing both accumulation and ablation patterns (Winstral et al., 2013; Fassnacht et al., 2003; Marks et al., 1999). However, because all snow sensors and courses are on flat or nearly flat ground, we did not include topographic data as explicit inputs to our models. We emphasize the flexibility of inductive machine learning, which can accommodate arbitrary new input modalities. Only those that are predictive of the dependent variable of interest will be significantly incorporated into the generated models. In this paper we focus on several potential snow telemetry and meteorological inputs in order to demonstrate the applicability of our techniques to catchment-scale *SWE* estimation, while considering the potential for future work to explore other inputs such as topographic data.

Third, we grapple with the issue of ground-truth for catchment-scale *SWE* and usable datasets. Constraints on our goal were imposed by the availability of snowpack data. We are not aware of catchment-wide *SWE* datasets with sufficiently fine time granularity to support our ideal scenario. Although datasets such as those provided by the Cold Land Processes Field Experiment (National Snow & Ice Data Center, 2014) and numerous others provide catchment-scale snowpack measurements, their time granularity is on the order of several months at least. Airborne techniques in general are cost-prohibitive for real-time reporting (Bühler et al., 2011). Although satellites are used to measure snow-covered area and albedo (Dozier and Painter, 2004), satellite retrievals of *SWE* are not feasible. Manual snow courses provide better temporal resolution than airborne methods (e.g. biweekly) but at low spatial resolution: snow courses measure *SWE* at a single location. We highlight the Snowcloud wireless sensor network, which measures *HS* (an effective predictor of *SWE*) in NRT (e.g. hourly) at multiple locations distributed over an area of interest. However, this technology is new, and available data collected by Snowcloud deployments is limited.

2. Background and contributions

Here we briefly define and summarize the machine learning methods used in this work. These techniques are described in more detail, with special emphasis on GP, in Section 4. The basic method (BM) assumes the spatial homogeneity of *SWE*. It naively estimates mean catchment-wide *SWE* to be the same as the single-point *SWE* measurement taken at a snow pillow. Linear regression (LR) fits a least-squares linear model to training data (Hastie et al., 2009). The prediction is a weighted linear combination of the input variables. Binary regression trees (BT) are nonlinear models which are generated using training data (Hastie et al., 2009). A BT model partitions a set of predictions according to the input variables such that a given set of input values results in a specific prediction. Genetic Programming (GP) is a symbolic regression algorithm that uses training data to iteratively improve a population of nonlinear models through a combination of stochastic variation and performance-based selection (Koza, 1992).

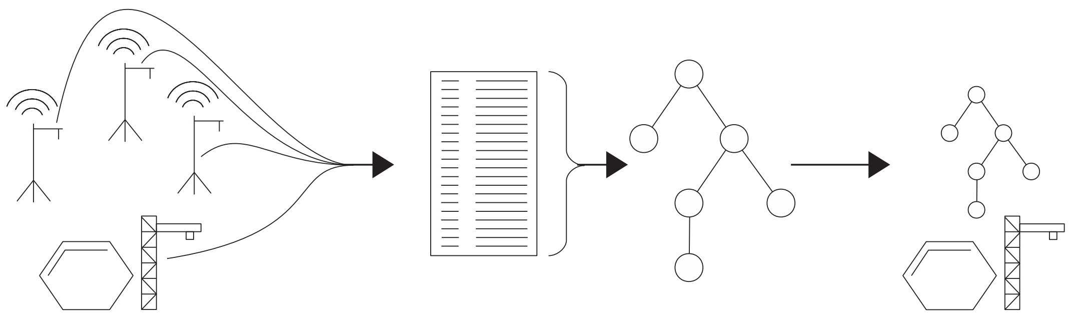


Fig. 1. First, the Snowcloud WSN is deployed in an area near a snow pillow. Next, data generated by Snowcloud, by the pillow, and potentially other sources, are used by machine learning to generate a model of snowpack distribution. Finally, after Snowcloud has been removed, the model is used to estimate snow levels in the area where Snowcloud had been deployed.

In our ideal situation we would use a large set of accurate measurements of mean catchment *SWE* as ground truth to train and evaluate models that predict mean catchment *SWE* in NRT. However, the only *SWE* measurements available at this spatial scale are generated by airborne techniques with time resolutions that are insufficient for machine learning (e.g. twice per year). Because machine learning needs a large number of samples for model training and because we want to predict *SWE* in near-real-time, we required much more frequent measurements. We therefore developed a series of experiments using *available* snowpack data in lieu of NRT catchment-scale *SWE* measurements to explore successive approximations of our ideal scenario. Approximations of average catchment *SWE*, obtained via snow courses and distributed ground-based sensor readings, serve as ground truth for machine learning in our experiments. Implicit in our work is the importance of new methods for obtaining NRT catchment-scale *SWE* ground-truthing via low-cost distributed sensor networks. As data from NASA's Airborne Snow Observatory (NASA Airborne Snow Observatory, 2015) become available for a range of years, they will provide an ideal data set for our approach.

First, we used snow course measurements, which involve the manual collection of *SWE* and/or *HS* at a single location, as a proxy for catchment-wide *SWE*. Although snow courses do not directly measure snowpack distribution at the catchment scale, they are likely to provide measurements that are *closer* to mean catchment *SWE* than snow pillows measurements are. Snow courses take multiple measurements over approximately 200 m, so they involve a much larger sample size than the single-point measurements of snow pillows. Furthermore, pillow under-measurement or over-measurement errors may occur when the base of the snow cover is at melting temperature (Johnson and Marks, 2004). Thus, we used snow course data as a first approximation of mean catchment *SWE* to provide ground-truth data for machine learning. We generated models that use readily available information such as meteorological telemetry and snow pillow measurements as input variables. This approach, which is explored in Experiment Set I, would allow for shorter or less frequent snow courses or for their discontinuation and, because it uses previously collected data, incurs no data gathering costs.

Second, we used *HS* data collected by the Snowcloud (Skalka and Frolik, 2014) wireless sensor network (WSN) at sites in Norway and California, each for only one snow season, as a proxy for catchment-wide *SWE* data. Snowcloud is a WSN-based data gathering system for snow hydrology, notable for its low-cost and ease of deployment, developed and operated by the University of Vermont. A network of light-weight sensor towers (nodes) is deployed over an area of interest for a short-term field campaign to collect spatially distributed measurements of relevant meteorological processes (Fig. 3). In addition to *HS*, Snowcloud measures air temperature, soil temperature, and solar radiation. Mesh wireless communication allows data from the entire network to be collected wirelessly by communication with a single node.

We used measurements collected from Snowcloud over the course of a single snow season to generate ground-truth estimates for model-training. Note that it could be desirable to collect data over multiple seasons as models trained on multi-year data may be more robust against internal-annual variations in snowpack distribution. Once a model has been obtained, the WSN may be recovered for re-deployment at another site. Unlike pillows and snow courses, Snowcloud collects NRT data from multiple locations, potentially capturing more of the variability of snowpack distribution than is possible with single-location measurements. Thus, we use Snowcloud data as a second approximation of catchment mean *SWE* to provide ground-truth data for machine learning. This technique is explored in Experiment Set II.

Recent research by Kerkez et al. (2012) and Welch et al. (2013) has developed new sensor placement strategies for monitoring snow. Although these methods were not employed in the experiments discussed in this paper, they should be considered in future applications of our techniques.

2.1. Suitability of machine learning

Snow pillows are large, expensive, permanent installations that measure *SWE* at a single location. The infrastructure for the automatic collection of *single-point SWE* is well established. For example, there are 830 Snowpack Telemetry (SNOTEL) sites in the United States (Surveyor, 2014) and another 124 snow pillows operated by the California Department of Water Resources. However, the extrapolation from single-point measurements to surrounding areas is error prone. The spatial distribution of alpine snow cover is highly variable (Balk and Elder, 2000; Elder et al., 1991; Jost et al., 2007), due to a variety of environmental forcing effects, such as topography (Anderton et al., 2004), canopy cover (Moeser, 2010), and wind and solar exposure (Moeser, 2010; Moeser et al., 2011).

Meromy et al. (2013) studied 15 snow stations across the western United States and found that snow station biases were frequently greater than 10% of the surrounding mean observed snow depth. The flat-field areas where snow pillows are commonly located are usually not typical of more complex nearby terrain, causing the majority of such stations to overestimate snow depth in their vicinity (Grünewald et al., 2013). Molotch and Bales (2005) studied the areas surrounding six SNOTEL stations in the Rio Grande headwaters. They found that only a small fraction of grid elements were representative of mean grid *SWE* during accumulation, and that no elements were representative of mean grid *SWE* during both accumulation and ablation. SNOTEL stations in the Rio Grande headwaters preferentially represent densely forested areas and experience snow cover persistence that is 14% greater than the mean persistence of the watershed (Molotch and Bales, 2006). Rittger (2012) found that errors based on statistical relationships between point measurements of snow and streamflow in the Sierra Nevada can reach 25–70% in one out of five years.

The relative importance of separate processes which govern snow distribution varies over the course of a snow season. Elder et al. (1991) summarize the various processes and explain how their influence changes over time. During the winter, accumulation and redistribution processes dominate. Precipitation is determined by regional climate and latitude as well as by local orographic effects, and redistribution by wind, avalanches, and sloughs are the primary causes of spatial heterogeneity. In the spring, however, snow distribution is controlled mainly by ablation. Of the many energy sources, solar and longwave radiation dominate. This energy decreases water in a basin through sublimation and when runoff leaves the basin. It also redistributes SWE, affecting spatial variability. These dynamics highlight the need for NRT modeling of snowpack, as the forcing effects that establish snow distribution vary drastically over the course of a snow season.

However, the significant *consistency* of snowpack *between* years encourages investment into the development of reusable statistical models. Strong inter-annual consistency in the spatial distribution of snow (Scipión et al., 2013), in *SCA* (Tappeiner et al., 2001), and in the snow depth patterns of maximum accumulation (Schirmer et al., 2011), have been observed in the Swiss and Italian Alps. In the western United States, consistent wind directions can produce stable snow accumulation patterns from year-to-year (Winstral and Marks, 2014). These findings suggest a strong link between accumulation patterns and geophysical terrain and indicate that site-specific snow distribution models may be able to accurately characterize snowpack distribution over multiple years.

Nevertheless, long-term changes in the patterns of snow distribution may be caused by factors such as changes in vegetation or climate change. Therefore, it may occasionally be necessary to rerun GP and generate a new model. Techniques such as retroactive *SWE* calculation (Rittger et al., 2011) could be used to detect when previous models begin to perform poorly, indicating that secular variability in the dynamics of snow distribution warrants the development of a new model.

It may be desirable to produce non-site-specific models. Trained at catchments where ground truth data is available, and making use of predictor variables that vary between catchments, such as topography, such models could then be applied to catchments where no independent measurement of mean catchment SWE exists. However, we did not incorporate topography because the snow pillows are all on flat or nearly flat ground. Our work focuses on site-specific models and use model inputs that vary over time at a given catchment.

2.2. Why GP?

It has been demonstrated that the relationships between snow distribution and the topographic and meteorological forcing effects include nonlinearities (Tappeiner et al., 2001), and the spatial distribution of *SWE* is nonlinear because it is influenced simultaneously by numerous processes including accumulation, ablation, and snow drifting (Marofi et al., 2011). GP can produce both linear and nonlinear models. If the data used to train GP contain only linear relationships, the resulting models will be linear, and the performance of GP will be similar to that of LR.

White-box models, such as those produced by GP, can be interpreted by human analysis, potentially yielding new information about the modeled data (Schmidt et al., 2011). Some nonlinear regressors, such as artificial neural networks, produce models that are difficult or impossible to interpret. GP trees, however, can be expressed as mathematical equations (Fig. 2). It is possible that by examining these equations domain experts could gain novel insight into the processes governing snow distribution.

Unlike regression techniques that constrain the form of the regressor, GP can combine operators, variables, and constants into arbitrary arrangements. GP does not require any assumptions about the form that a model should take: it is left open to inductive search. By generating models that use predictor variables in unexpected ways, GP may help discover previously unknown relationships among variables.

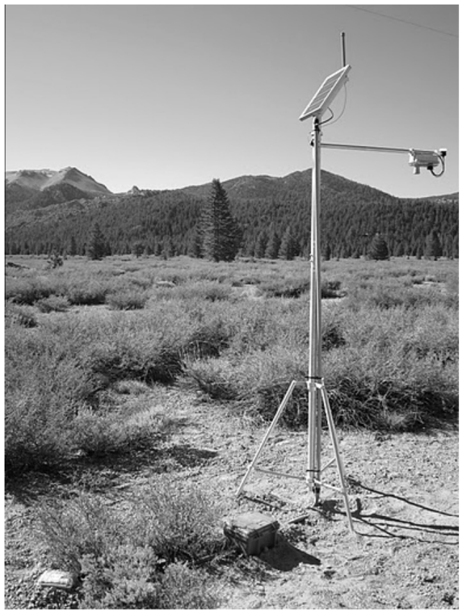


Fig. 3. Snowcloud WSN sensor tower. A complete sensor stand with solar-recharged battery power, wireless mesh communication, and multiple sensor modalities. October 2011, Mammoth Lake, CA.

Finally, as we will discuss further, GP may be augmented with multi-objective optimization, which constrains GP to produce parsimonious models. This mitigates against over-fitting, a significant concern in the case that relatively small datasets are available for machine learning.

While many regression techniques possess one or more of these desirable qualities, GP possesses all of them, making it an ideal candidate for snowpack modeling.

2.3. The primacy of snow depth

While *SWE* is a product of *HS* and density (ρ), it has been shown that *HS* is the essential determining metric for *SWE* estimation. Models have been developed to derive ρ estimates from *HS* measurements (Logan, 1973; Sturm et al., 2010), and measurements of *HS* are highly predictive of *SWE* (Adams, 1976). Analysis of the spatial variability of *HS* and ρ has revealed that the variability of *HS* is significantly greater than that of ρ (López-Moreno et al.,

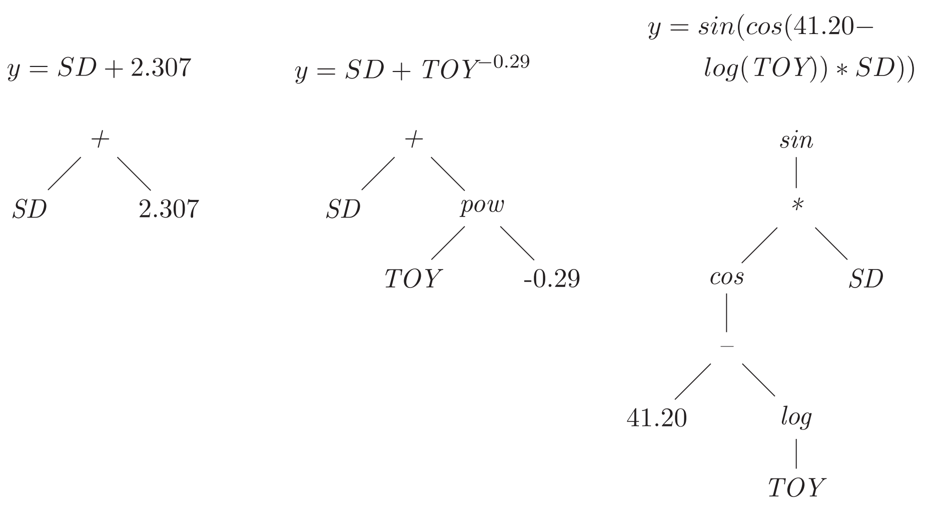
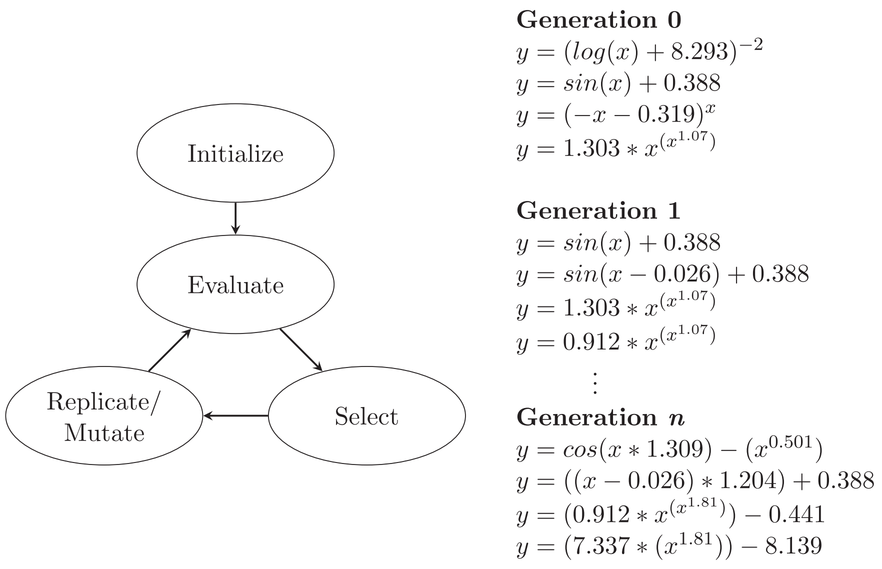


Fig. 2. These example GP trees were manually selected from the final populations of GP runs conducted for Experiment Set II. The leftmost tree represents a simple linear model. The middle tree is a nonlinear model. The rightmost tree is a more complex nonlinear model.



Generation 0

$y = (log(x) + 8.293)^{-2}$
$y = sin(x) + 0.388$
$y = (-x - 0.319)^{x}$
$y = 1.303 * x^{(x^{1.07})}$

Generation 1

$y = sin(x) + 0.388$
$y = sin(x - 0.026) + 0.388$
$y = 1.303 * x^{(x^{1.07})}$
$y = 0.912 * x^{(x^{1.07})}$
$\vdots$

Generation *n*

$y = cos(x * 1.309) - (x^{0.501})$
$y = ((x - 0.026) * 1.204) + 0.388$
$y = (0.912 * x^{(x^{1.81})}) - 0.441$
$y = (7.337 * (x^{1.81})) - 8.139$

Fig. 4. Genetic programming algorithm. The figure on the left demonstrates the iterative process through which GP modifies a population of solutions. On the right, a population of four models evolves as each iteration of the GP cycle produces a new generation.

2012). Variation of *SWE* is therefore overwhelmingly a product of *HS* variation (Moeser et al., 2011; Molotch et al., 2005; Sturm et al., 2010; Elder et al., 1991, 1998). The effect of ρ variation on *SWE* is small by comparison, and estimates of areal *SWE* derived from one or several *SWE* measurements can be greatly improved by incorporating a larger number of *HS* measurements (Elder et al., 1998; Moeser et al., 2011), which are much less labor intensive than manual *SWE* measurements (Sturm et al., 2010). Snowcloud, which provides ground-truth data Experiment Set II, measures *HS*. Therefore, as has been done elsewhere, we use *HS* as a "surrogate for *SWE*" (Winstral et al., 2002).

2.4. Related work

Moeser et al. (2011) explored three models for estimating *SWE* in the area around a meteorological station using ground based measurements. The first model used meteorological data such as air temperature and solar radiation, tree canopy cover measurements, and *HS* measurements collected by the Snowcloud WSN, as well as a single-point *SWE* measurement. The second model used multiple *HS* measurements and single-point *SWE* measurements, but no meteorological or tree canopy data. The third model used meteorological and tree canopy data, along with multiple *HS* measurements, but no single-point *SWE* measurement. It was found that increasing the number of *HS* measurements can improve areal *SWE* measurements because *HS* varies more than snow density. While this work used linear modeling; our work expands upon it by developing nonlinear models.

Marofi et al. (2011) compared three methods for modeling *SWE*: multivariate nonlinear regression (MNLR), artificial neural networks (ANN), and a neural network-genetic algorithm (NNGA), where genetic algorithms were used to parameterize ANNs and the learning process. ANN performed better than MNLR, suggesting that computational intelligence approaches may outperform MNLR for modeling *SWE*. NNGA performed better than ANN, suggesting that evolution-inspired genetic algorithms can be used to develop effective models of *SWE*. Tabari et al. (2010) estimated *HS* and *SWE* using multiple methods and also found that NNGA provided the best results. Unlike neural networks, GP produces white box models.

Tappeiner et al. (2001) compared the performance of LR-based and ANN-based snowpack models, which used topographic and meteorological data to estimate *SWE*. The authors compared the results of LR with ANN to estimate the degree of necessary nonlinearity in *SWE* modeling. The ANN performed significantly better than LR, demonstrating nonlinearity in the relationships between topographic and meteorological variables and *SWE*.

Several studies have used binary regression trees to model snowpack. Winstral et al. (2002) derived terrain-based parameters from digital elevation models (DEM) which were used as input variables to binary regression trees. One parameter was based on maximum upwind slopes relative to seasonally averaged winds. Another measured upwind breaks in slope from a given location. Binary tree models based on these terrain-based parameters as well as elevation, solar radiation, and slope performed better than models based only on elevation, solar radiation, and slope. Elder et al. (1998) modeled the distribution of *SWE* by merging remotely sensed snow-covered area data with binary tree models applied to field measurements of *HS* and *SWE*. Balk and Elder (2000) combined binary regression trees with kriging of manual snow survey measurements and snow-covered area determined by aerial photographs, to estimate *SWE*. Anderton et al. (2004) used binary regression trees to relate *HS* and disappearance date to terrain indices. They found that the topographic effects on snow redistribution by wind primarily determined *SWE* distribution at the start of the melt season which, more than melt rates, determined the patterns of snow disappearance. Molotch et al. (2005) compared binary regression tree models using various sources of DEMs and found that using DEMs from different sources leads to significant differences in modeled snowpack distribution. The most significant differences were on ridge-tops, where the elevation values differed across DEMs.

In Experiment Set II we compare the performance of BT to GP. Unlike this previous work which used binary regression trees to produce spatially distributed models of snowpack, our models predict a single value: mean *HS* measured by a wireless sensor network.

Marks et al. (1999) also developed spatially distributed models. They used topographic data to determine estimates of radiation, temperature, humidity, wind, and precipitation for use in a coupled energy and mass-balance model called ISNOBAL.

Recent research has made significant advances in simulating the effects of wind on snow distribution. Winstral et al. (2009) developed a simplified wind model that uses upwind topography to accurately predict wind speeds. Winstral et al. (2013) developed

Table 1
CDEC snow course site descriptions.

ID	EL (m)	Name	Asp.	Exposure
$\mathcal{CAP}$	2438	Caples Lake	SW	open meadow, low brush
$\mathcal{GRZ}$	2103	Grizzly Ridge	N	meadow in scattered timber
$\mathcal{KTL}$	2225	Kettle Rock	S	sloping, open meadow
$\mathcal{MSH}$	2408	Mount Shasta	SE	grassy and rocky meadow
$\mathcal{NTH}$	2835	North Lake	SE	grassy meadow
$\mathcal{SPD}$	1585	Lake Spaulding	level	grassy meadow
$\mathcal{HIG}$	1838	Highland Lakes	NW	medium sized meadow in dense timber
$\mathcal{HYS}$	2012	Huysink	W	open meadow on one leg, opening in timber on second leg

a snow distribution algorithm that uses terrain structure, vegetation, wind, and precipitation data to simulate wind-affected snow accumulation. It accurately predicted disparate snow distribution caused by inhomogeneous precipitation and redistribution by wind. Winstral and Marks (2014) analyzed the effects of wind on snow distribution. They found that high wind speeds increased snow depth variability, that forested sites decreased variability by moderating wind effects, and that consistent wind directions produced accumulation patterns that were stable between years.

Sturm et al. (2010) used *HS*, day of the year, and climate classes, such as Alpine, Maritime, and Tundra, to estimate snowpack density. Estimated snowpack density was used to convert *HS* measurements into *SWE* estimates.

Guan et al. (2010) found that atmospheric rivers (ARs), are associated with intense storms that contribute a large percentage of snow during most years. Because AR storms are relatively warm, the participation of AR participation into snowfall versus rainfall is sensitive to minor variation in surface air temperature.

Rittger et al. (2011) combined satellite-based measurements of snow-covered area with energy balance calculations to retroactively calculate distributed SWE at the date of maximum accumulation, using the "reconstruction" technique originally developed by Martinec and Rango (1981). This calculation was then used to evaluate the accuracy of two real-time models. They found that at elevations below 1500 m, the real-time models overestimated *SWE* because of early season melt, and at elevations above 3000 m, the real-time models underestimated *SWE* because they do not sample these higher elevations. It is possible that this technique could be used to evaluate the effectiveness of the inductive learning methods that we describe in this work.

3. Training data and model inputs

Inductive machine learning requires substantial datasets for developing and evaluating models, and we acquired extensive hydrological and meteorological data for use in our experiments. We focused on two types of available datasets that are approximations of mean catchment SWE. First, we consider a record of CDEC snow courses from the Sierra Nevada. We observe that CDEC snow courses are intended to provide an estimation of SWE at a particular elevation (USDA, 2014), though in fact they are linear transects of SWE samples. Second, we consider a record of Snowcloud sensor network readings from Norway and California. Snowcloud provides distributed coverage of snow depth readings for the deployment area, as well as fine time granularity, and can support better estimations of mean catchment *SWE* than periodic snow courses.

3.1. Experiment Set I data

Experiment Set I used data collected from eight sites across California. There were three main types of data: *SWE* from manual snow courses, *SWE* measurements from snow pillows, and air temperature data.

The California Data Exchange Center (CDEC) provided an extensive database of snow data. The snow courses that we used, which are described in Table 1, were performed monthly, were about 200 meters long, and consisted of 10 measurements, the mean of which was recorded. CDEC also maintains single-point *SWE* measurement data from snow pillows at sites throughout California. Of the 404 snow course sites, 59 are co-located with snow pillows.

The National Climate Data Center (NCDC) maintains meteorological data, such as air temperature, wind speed, and solar radiation measurements, collected at weather stations across the United States. We used data from the four NCDC stations which are located within 30 km of CDEC snow courses. We arbitrarily chose a 30 km cutoff because we suspected that meteorological activity within that distance might be predictive of measurements at the snow course. The models generated by machine learning will not make significant use of input data that is not predictive.

Significant gaps exist in the NCDC database, and of the various sensor modalities, air temperature data is the most complete. Using more meteorological inputs and necessarily fewer data samples, we had previously been unable to generate effective models of *SWE*. For Experiment Set I, therefore, air temperature was the only meteorological input. Air temperature is known to be a highly effective predictor of melt rate because it is correlated with long-wave atmospheric radiation, the most important energy source for snowmelt (Ohmura, 2001). Air temperature is made accessible to the models by three variables: *minTemp7*, *maxTemp7*, and *meanTemp7*, which aggregate daily values over the seven days inclusively preceding the day for which *SWE* is estimated.

We used the temporal and spatial intersection of available data from these three sources (CDEC snow courses, CDEC snow pillows, NCDC air temperature data) to construct eight datasets, based on eight snow course sites. These snow courses were selected because they are coincident with either snow pillow data, NCDC air temperature data, or both, over a range of time that includes a large number of samples points (greater than 100 except for one site). The constructed datasets are summarized in Table 2.

3.2. Experiment Set II data

Experiment Set II used *HS* data collected by four Snowcloud sensor nodes in Sulitjelma, Norway between January and April, 2013. Each node sampled *HS* every six hours. We averaged *HS*

Table 2
Experiment Set I data summary by CDEC site.

ID	Pillow	NCDC base	Dist (Mi)	Samples	Years
$\mathcal{CAP}$	YES	N/A	N/A	177	1970–2011
$\mathcal{GRZ}$	YES	N/A	N/A	207	1970–2011
$\mathcal{KTL}$	YES	N/A	N/A	159	1979–2011
$\mathcal{MSH}$	NO	Mount Shasta	5.98	137	1973–2011
$\mathcal{NTH}$	NO	Bishop Airport	18.27	147	1973–2011
$\mathcal{SPD}$	NO	Blue Canyon Nyack	4.56	174	1977–2011
$\mathcal{HIG}$	YES	Mount Shasta	18.31	75	1980–2012
$\mathcal{HYS}$	YES	Blue Canyon Nyack	9.79	111	1984–2011

Table 3
Snowcloud deployment coordinates.

Sulitjelma, Norway			Sagehen, CA		
Tower	Lat.	Long.	Tower	Lat.	Long.
1	67.0981	16.0488	1	39.43161	−120.23975
2	67.0983	16.0497	2	39.43155	−120.23936
3	67.0983	16.0482	3	39.43140	−120.23976
4	67.0987	16.0487	4	39.43173	−120.23882
			5	39.43173	−120.23864
			6	39.43204	−120.23872

measurements from the four nodes (Table 3) and then over each day to produce 93 estimates of mean catchment *HS*. These values served as ground-truth *HS* for experiments at Sulitjelma.

Approximately 16 km away from the Sulitjelma Snowcloud deployment site is Storstilla nedanför Balvatn in Nordland County, station number 164.12.0 (Balvatn). The Balvatn station records both *HS* and *SWE*. Daily *HS* measurements collected at Balvatn compose the *HS* input variable to models developed for Sulitjelma in Experiment Set II.

Six Snowcloud wireless sensor network sensor nodes were deployed within the Sagehen Creek Field Station, near Truckee, California, from January to May, 2010. Each node reported daily *HS* measurements, which we averaged to generated 99 estimates of mean catchment *SWE*. These values served as ground-truth *HS* for experiments at Sagehen. Note that the same WSN data were used by Moeser (2010).

In order to assess the significance of the *source* of single-point *HS* input variables, we developed models for estimating mean *HS* at the Sagehen Snowcloud deployment using inputs from two different CDEC sites, *Independence Camp* ($\mathcal{IDC}$) and *Huysink* ($\mathcal{HYS}$). $\mathcal{IDC}$ is approximately 5.5 km away from the Snowcloud deployment and, like Sagehen, is on the Eastern side of the Sierra crest. $\mathcal{HYS}$ is approximately 30 km away, on the Western side of the crest.

3.3. Time of year

Because the dynamics underlying snowpack distribution vary over the course of a snow season, for example between periods dominated by deposition and periods dominated by ablation, we introduce time of year (*TOY*) as an independent variable for both experiment sets. This allows models to distinguish parts of the snow season. Time of year is an integer value expressing the number of days since January 1.

3.4. Preparation of datasets

We define a dataset, *D*, for each experiment (each row of Table 6 and each location in each row of Table 5). Elements of a dataset *D* take the form of a 3-tuple, $\langle T, \theta, \vec{p}\rangle$, where *T*, time, specifies a calendar date, θ is an estimate of the true value of the independent variable, and $\vec{p}$ is a vector of predictor variables. Although *T* is used to generate predictor variables such as *TOY* and air temperature statistics, it is not itself a predictor variable and is therefore not included in $\vec{p}$. *T* is unique in *D* so that no two data samples in *D* have the same *T*:

$$\forall \langle T_1, \theta_1, \vec{p}_1\rangle, \langle T_1, \theta_2, \vec{p}_2\rangle \in D \qquad \theta_1 = \theta_2 \qquad \text{and} \qquad \vec{p}_1 = \vec{p}_2 \tag{1}$$

In Experiment Set I, θ is an approximation of mean catchment *SWE* derived by manual snow course. In Experiment Set II, θ is an approximation of mean catchment *HS* derived from Snowcloud WSN measurements.

Depending on the experiment, $\vec{p}$ includes some combination of *HS* measured at a snow pillow, *SWE* measured at a snow pillow, *TOY* (an integer value derived from *T*), and air temperature, (which is composed of three variables: *minTemp7*, *maxTemp7*, and *meanTemp7*). The *Model inputs* columns of Table 5 and Table 6 specify the contents of $\vec{p}$ for each experiment.

In order that a model developed from *D* may be evaluated on new, unseen data, *D* is divided into training, ϱ, and testing, τ, subsets. The training set is twice as large as the testing set. However, GP and BT require that ϱ be further divided into grow, *g*, and selection, *s*, subsets:

$$\varrho = g \cup s \qquad \text{and} \qquad g \cap s = \emptyset \qquad \text{and} \qquad |g| = |s| \tag{2}$$

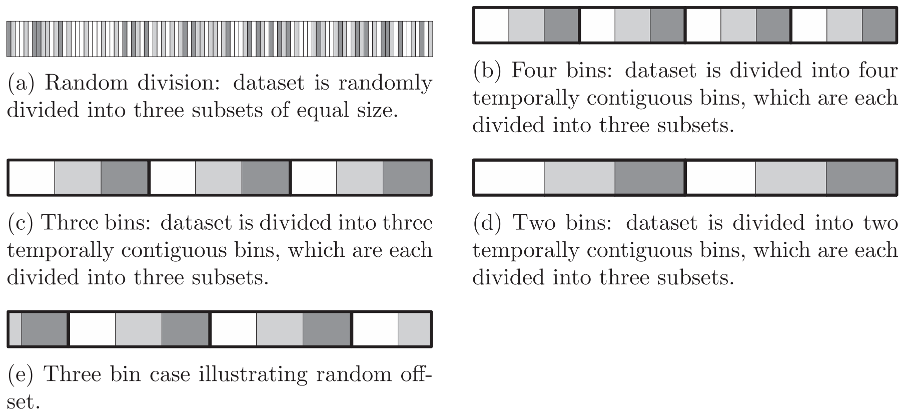
In all experiments, *D* is first divided into g, s, and τ:

$$D = g \cup s \cup \tau \quad \text{and} \quad g \cap s \cap \tau = \emptyset \quad \text{and} \quad |g| = |s| = |\tau| \tag{3}$$

For BM and LR, *g* and *s* are simply combined into ϱ and used as training data. As discussed in more detail in Section 4, in the case of GP and BT *g* is used to generate a set of models and *s* is used to determine which one should be kept and evaluated on τ. In any case, ϱ is used to obtain a single model, which is then exposed to τ to evaluate its ability to predict unseen data.

We explored several methods for dividing *D* into g, s, and τ. In Experiment Set I and in the first part of Experiment Set II (Experiment Set II: *Random Division*), the chronologically ordered *D* is randomly shuffled and then divided into thirds, as illustrated by Fig. 5a. This method has the effect that a large portion of the training data is likely to be temporally proximal to testing data.

As discussed further in Section 5, we found in Experiment Set II that the temporal proximity between ϱ and τ caused machine learning to map *TOY* values to estimates of *HS*. The models memorized the data rather than capturing the relationships among the data. We therefore conducted Experiment Set II: *4 Bins*. Instead of shuffling *D*, we maintained its ordering and divided it into four



(a) Random division: dataset is randomly divided into three subsets of equal size.

(b) Four bins: dataset is divided into four temporally contiguous bins, which are each divided into three subsets.

(c) Three bins: dataset is divided into three temporally contiguous bins, which are each divided into three subsets.

(d) Two bins: dataset is divided into two temporally contiguous bins, which are each divided into three subsets.

(e) Three bin case illustrating random offset.

Fig. 5. Techniques for dividing a chronologically ordered dataset into g, s, and τ (white, light gray, and dark gray respectively).

chronologically contiguous bins. Each bin is then subdivided into three chronologically contiguous subsets which are assigned to g, s, and τ. This method is illustrated by Fig. 5b. We also conducted Experiment Set II: *3 Bins* and Experiment Set II: *2 Bins*, as illustrated in Fig. 5c and d. As we move from Experiment Set II: *Random Division* to Experiment Set II: *2 Bins*, the division of D transitions from finer to coarser temporal granularity. As this granularity becomes coarser, it becomes more difficult for machine learning to use *TOY* to simply memorize data. However, it also becomes more difficult for models to capture the variation of the dynamics of snowpack distribution over the course of a snow season.

In order to introduce stochasticity into the division D and thus allow the repetition of experiments to produce a distributed sample of results, a randomly generated offset shifts the starting point of the division. Fig. 5e illustrates the effect of this offset in the case of three bins.

4. Calculation

In this section we first describe how we compared the performance of different snowpack modeling techniques. We then describe the various modeling techniques that we used, with special emphasis on GP.

4.1. Comparing estimation methods

In order to compare the performance of two machine learning techniques, M and M', on a dataset D, D is divided into complementary subsets ϱ and τ. Methods M and M' are applied to ϱ to produce estimators $\hat{\theta}$ and $\hat{\theta}'$. This process may be deterministic or nondeterministic. In Experiment Set I and Experiment Set II: *Random Division*, nondeterminism is introduced by the random division of D. GP introduces further nondeterminism by the stochasticity of the GP algorithm. The BT algorithm is deterministic when a single input variable is used, but nondeterministic when applied to multiple input variables. Estimators $\hat{\theta}$ and $\hat{\theta}'$ are applied to τ to determine the mean absolute errors of the estimators $\mathrm{MAE}(\hat{\theta})$ and $\mathrm{MAE}(\hat{\theta}')$, as we will discuss in Section 4.2.

This process of randomly dividing D and applying M and $M\prime$ to obtain $\mathrm{MAE}(\hat{\theta})$ and $\mathrm{MAE}(\hat{\theta}')$ is repeated 30 times, resulting in vectors of estimator errors $\vec{e}_M$ and $\vec{e}_{M'}$ each with cardinality 30. We consider $\vec{e}_M$ and $\vec{e}_{M'}$ to be statistical samples of errors drawn from the population of errors that method M and M' could produce given D. We chose to collect 30 samples because a sample size of at least 30 allows the Central Limit Theorem to be safely applied without assuming a normal population distribution, permitting the application of the one-sample t-test to calculate confidence intervals and the paired two-sample t test to test hypotheses.

The means of $\vec{e}_M$ and $\vec{e}_{M'}$ are unbiased estimates of the true population means μ_M and μ'_M. To find out if M' outperforms M on dataset D we pose the hypotheses:

$$H_0 : \mu'_M = \mu_M \qquad (\textit{Null hypothesis})$$
$$H_a : \mu'_M < \mu_M \qquad (\textit{alternative hypothesis})$$

and apply the Student's t-test for paired samples to $\vec{e}_M$ and $\vec{e}_{M'}$. If the Null hypothesis is rejected, we say that method M' produces lower error (performs better) on dataset D than does M. We report the p-value, the probability that we have performed a Type I error by rejecting a true Null hypothesis.

4.2. Evaluating estimator error

Recall that an element d of dataset D takes the form $\langle T, \theta, \vec{p}\rangle$ and that D has been divided into ϱ and τ. An estimation method M is applied to $\varrho \subset D$ to generate an estimator $\hat{\theta}$, which is a function from predictor variables $\vec{p}$ to dependent variable y, an estimate of θ.

$$\hat{\theta} : \vec{p} \to y \qquad y \approx \theta$$

The error of $\hat{\theta}$ on an input vector is the difference between the estimate it produces and ground truth.

$$\mathrm{E}_{\hat{\theta}}(\vec{p}) = \hat{\theta}(\vec{p}) - \theta \tag{4}$$

The error is calculated on each sample in τ to determine the mean absolute error of the estimator:

$$\mathrm{MAE}(\hat{\theta}) = \frac{\sum_{i=1}^{k} |\mathrm{E}_{\theta}(\vec{p}_i)|}{k} \tag{5}$$

where

$$\tau = (d_1, \ldots, d_k) \qquad \text{and} \qquad \vec{p}_i \in d_i \in \tau \subset D$$

4.3. Basic method

The basic method (BM) assumes that *SWE* as measured at a snow pillow is representative of catchment-wide *SWE*. It naively estimates ground truth (snow course-derived) *SWE* to be the same as the independent variable (snow pillow-derived) *SWE* measurement. Error in the predictive power of BM expresses the difference between snow pillow measurements and snow course *SWE* measurements. If x represent *SWE* measured at the snow pillow, then

$$x \in \vec{p} \qquad \text{and} \qquad \hat{\theta}(\vec{p}) = x \tag{6}$$

Unlike the more sophisticated machine learning techniques, BM does not make use of training data to generate a model.

4.4. Linear regression

Linear regression (LR) fits a least-squares linear model to training data which is then evaluated on test data (Hastie et al., 2009). LR expresses the linear relationships between independent and dependent variables. We used the *gsl_multifit_linear* function from the GNU Scientific Library (GSL, 2014) to perform LR. We include LR in order to gain insight into the data we are using. LR will perform less well than nonlinear techniques only if the modeled data contain nonlinear relationships.

4.5. Genetic programming

GP is an evolutionary algorithm, inspired by biological evolution, that iteratively evolves populations of parse trees to perform symbolic regression (Koza, 1992) (see Fig. 4). In this work, the trees are snowpack models, estimator functions, that use available independent variables to estimate mean *SWE* (Experiment Set I) or *HS* (Experiment Set II) at the catchment scale. Tree terminals are input variables and constants, while internal nodes are arithmetic operators. The operators we used are listed in Table 4.

Table 4
GP parameters.

Parameter	Value
population size	1000 (Experiment Set I), 2000 (Set II)
number of generations	3000 (Experiment Set I), 10,000 (Set II)
max tree size	30
mutation operators	crossover (60%), mutation (40%)
binary operators	addition, subtraction, mult., division, power
unary operators	log, exponential, sine, cosine,
terminals	independent variables, constants values

We used the lil-gp Genetic Programming System (System, 2013), an open source implementation of GP, in order that we might make any needed modifications. We modified lil-gp to implement multi-objective Pareto optimization.

GP begins by generating a starting population of randomly constructed trees. Each tree in the population is evaluated on training data to determine its fitness, defined as the inverse of mean error. Trees are selected according to their size and fitness to produce the population for the next generation. Genetic operators make stochastic modifications to the new trees, randomly perturbing their fitness values. The genetic operators we used were *mutation* and *crossover*. Mutation, which is applied to 40% of new trees, selects a subtree at random and replaces it with new, randomly generated subtree. In crossover, which is applied instead of mutation 60% of the time, two parent trees exchange subtrees, resulting in two novel offspring. Crossover allows recombination of subtrees from existing models while mutation introduces new subtrees to the population, maintaining genetic diversity. Because it is likely that subtrees taken from existing, partially evolved models will be more useful than new, randomly generated subtrees, crossover is applied more frequently than mutation. This process is iterated over many evolutionary generations, each time replacing the population with a new population of altered trees. Over time, this produces populations of increasing fitness.

The average wall-clock time for one experiment using the Vermont Advanced Computing Core (VACC) supercomputer was 333 s for Experiment Set I (3000 generations) and 1207 s for Experiment Set II (10,000 generations). The total wall-clock time for all of Experiment Set I was approximately 89 h. The total wall-clock time for all of Experiment Set II was approximately 321 h. Because GP is a stochastic optimization method, its computation complexity is unclear. However, recent work has begun to address this problem (Neumann et al., 2011; Durrett et al., 2010).

One challenge facing GP, like all techniques for deriving a model from training data, is over-fitting. An over-fit model performs well on training data but does not generalize well and fails on unseen data. It memorizes values instead of capturing the mathematical relationships among the data.

The size of a GP model (number of nodes in a tree) constrains its complexity and fitness. Trees that are too small are too simple to accurately model the data and are under-fit. They perform poorly on both training and testing data. Trees that become too large perform extremely well on training data but, due to over-fitting, perform poorly on unseen data. Somewhere between these extremes lies the best, non-over-fit model.

In order to explore the gradient from small, under-fit models to large, over-fit models, we added multi-objective Pareto optimization to lil-gp. Pareto optimization applies evolutionary pressure toward multiple simultaneous goals, in this case low error and small model size, by producing a population (front) of non-dominated models. A tree is dominated by another tree if it is inferior by all objectives, i.e. it is both larger and has lower fitness. A Pareto front (non-dominated front) consists of a set of trees such that no tree is dominated by any other tree on the front. The non-dominated trees are selected at each GP generation so that each population is a non-dominated front, including the final population. The result of GP is therefore a set of trees of various sizes. We set an absolute upper bound at size 30 because we had observed that models with size larger than 30 were consistently over-fit. Arranged from smallest to largest, the error of these trees on the training data decreases monotonically. Error on unseen data, however, will decrease only to a point, and will then increase beyond some tree size as the models become over-fitted.

At this point is the tree size that will maximize performance on ϱ without over-fitting. Models no bigger than this can express features common to both training and testing data but cannot express features that are unique to the training data. However, this size threshold is not known while generating models because test data are not available. It must remain *unseen* for model testing. We therefore developed a novel *selection set* method for selecting a single model from the Pareto front. In the *selection set* method, the training data are further divided into two subsets of equal size, a growth set, g, and a selection set, s (Eq. 2). GP is applied to g to obtain a Pareto front. Each model on the front is then evaluated on s. GP returns the model that performs best (lowest error) on s. We used the *election set* method in all experiments.

4.6. Binary regression trees

We include BT in Experiment Set II in order to compare GP to another nonlinear, less computationally demanding, modeling technique. Erxleben et al. (2002) compared the performances of four spatial interpolation methods to estimate *SWE* and found that a method combining binary regression trees with geostatistical methods was more accurate than other methods. We used the DecisionTreeRegressor class of the Scikit-learn machine learning module for Python (Pedregosa et al., 2011). This software implements the Classification and Regression Trees (CART) algorithm, which is similar to C4.5 (Hastie et al., 2009). BT is parameterized by the maximum tree depth; we used default options for other parameters. As with GP, the data for BT was divided into g, s, and τ. For each experiment, a set of trees was trained on g such that the nth tree had a maximum depth of n. The maximum value of n was determined by incrementing n until further increase did not result in larger trees. The maximum value of n varied between 7 and 13.

Like the Pareto front produced by GP with multi-objective optimization, this methods results in a gradient of models ranging from very small models with high error on g to very large models with low error on g. Each is evaluated on s and the one with the lowest error is returned by BT to be evaluated on τ in order to determine model error. Thus, we applied the same *selection set* method to BT as to GP in order to discourage over-fitting and to provide similar exposure to the data so that the performance of the techniques may be compared. Note, however, that in the case of GP, multi-objective optimization applies pressure toward model parsimony continuously over the course of the evolution of a population of models. In the case of BT, the selection set method was applied once to a set of models after they have been generated.

5. Experiments: descriptions and results

In this section we describe the experiments we conducted and report the results.

5.1. Experiment Set I

In Experiment Set I measurements from snow courses provided ground-truth *SWE* data. We developed models to predict snow course *SWE* at eight different sites in California where snow courses had been conducted (Table 1). Three sites ($\mathcal{CAP}, \mathcal{GRZ}, \mathcal{KTL}$) are located at snow pillows but are not near any NCDC weather stations. Three sites ($\mathcal{NTH}, \mathcal{SPD}, \mathcal{MSH}$) are near NCDC stations but are not at snow pillows. Two of the snow course sites ($\mathcal{HYS}$ *and* $\mathcal{HIG}$) are located at snow pillows and are also near NCDC stations.

First, we conducted experiments at sites with snow pillows but without weather stations ($\mathcal{CAP}, \mathcal{GRZ}, \mathcal{KTL}$). These experiments explored how well linear and nonlinear models predict snow

Table 5
Experiment Set I summary.

Experiment	Model inputs	Locations
a	air temp.	$\mathcal{MSH}, \mathcal{NTH}, \mathcal{SPD}, \mathcal{HIG}, \mathcal{HYS}$
b	*TOY*	all
c	pillow	$\mathcal{CAP}, \mathcal{GRZ}, \mathcal{KTL}, \mathcal{HIG}, \mathcal{HYS}$
d	air temp., *TOY*	$\mathcal{MSH}, \mathcal{NTH}, \mathcal{SPD}, \mathcal{HIG}, \mathcal{HYS}$
e	air temp., pillow	$\mathcal{HIG}, \mathcal{HYS}$
f	*TOY*, pillow	$\mathcal{CAP}, \mathcal{GRZ}, \mathcal{KTL}, \mathcal{HIG}, \mathcal{HYS}$
g	air temp., *TOY*, pillow	$\mathcal{HIG}, \mathcal{HYS}$

course-derived ground truth *SWE* using only snow pillow measurements. Inputs to the models were pillow *SWE* and *TOY*. At each site we developed models with three combinations of input variables: *TOY* alone, pillow *SWE* alone, and *TOY* combined with pillow *SWE*. In each case, we compared the performance of GP, LR, and BM.

Second, we conducted experiments at sites near weather stations but without snow pillows ($\mathcal{KTL}, \mathcal{MSH}, \mathcal{NTH}$). These experiments explored how well linear and nonlinear models predict snow course-derived ground truth *SWE* using air temperature data without access to snow pillow *SWE* measurements. Inputs to the models were *air temperature* and *TOY*. At each site we develop models with three combinations of input variables: temperature alone, *TOY* alone, and temperature combined with *TOY*. In each case, we compare the performance of GP to LR.

Third, we conducted experiments at sites that are near weather stations and have snow pillows ($\mathcal{HIG}$, HYS). These experiments explored how well linear and nonlinear models predict snow course-derived ground truth *SWE* using both pillow *SWE* measurements and air temperature data. Inputs to the models were *SWE*, *air temperature*, and *TOY*. At each site we develop models with seven unique combinations of input variables: temperature alone, *TOY* alone, pillow *SWE* alone, temperature and *TOY* together, temperature and pillow *SWE* together, *TOY* and pillow *SWE* together, and, finally, temperature, *TOY*, and pillow *SWE* together.

Table 5 summarizes Experiment Set I. Each experiment was repeated 30 times to generate error samples for each method. Figs. 6–9 plot the mean values of the samples. Error bars indicate 95% confidence intervals, i.e. sample mean $\pm(\text{SEM} \times 1.96)$. GP and LR had similar error, but both had lower error than BM with p-value less than 0.001 in all cases.

The mean ground truth *SWE* value in mm at each site was: $\mathcal{CAP}: 1145, \mathcal{GRZ}: 1256, \mathcal{KTL}: 687, \mathcal{MSH}: 1747, \mathcal{NTH}: 337, \mathcal{SPD}: 697, \mathcal{HIG}: 594, \mathcal{HYS}: 1065$..

5.2. Experiment Set II

In Experiment Set II models predicted *HS* instead of *SWE*. While research on the influence of meteorological factors on snowpack distribution is extensive (Logan, 1973; Elder et al., 1991;

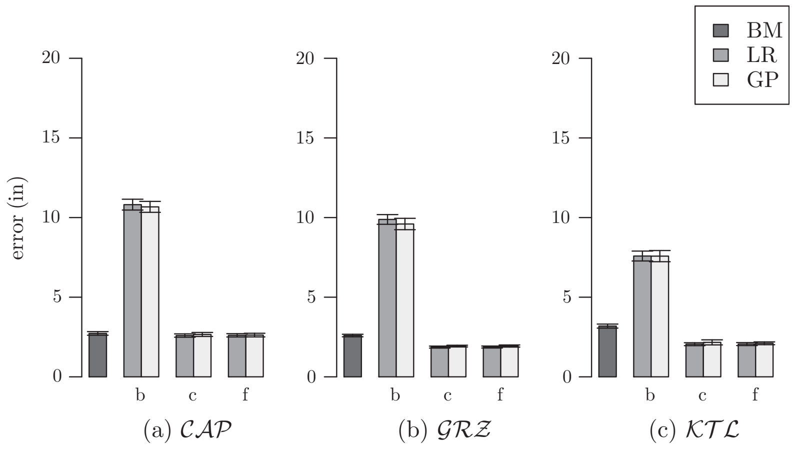


Fig. 6. Experiment Set I results: $\mathcal{CAP}, \mathcal{GRZ}, and \mathcal{KTL}$.

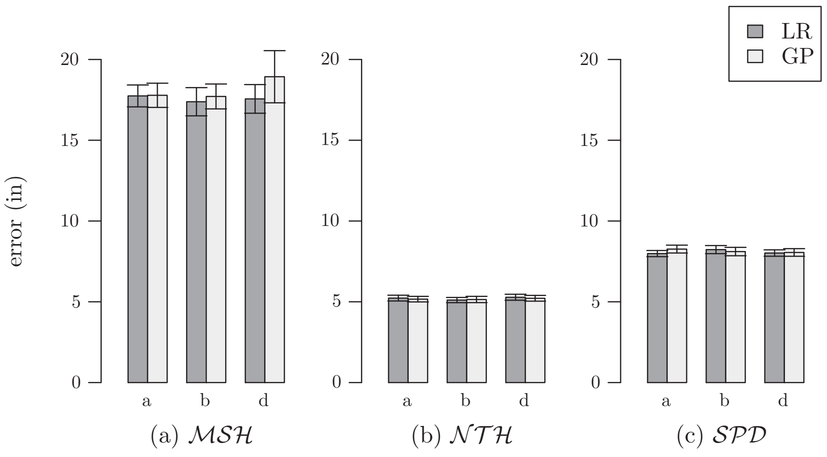


Fig. 7. Experiment Set I results: $\mathcal{MSH}, \mathcal{NTH}, and \mathcal{SPD}$.

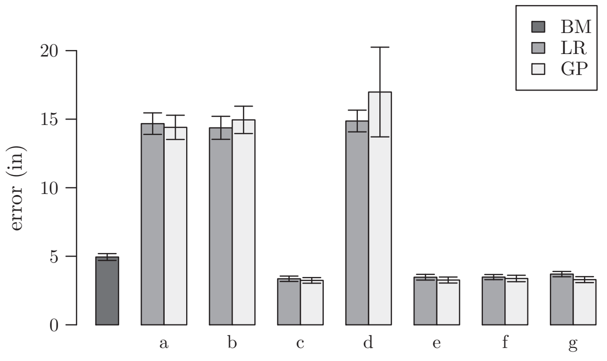


Fig. 8. Experiment Set I results: $\mathcal{HIG}$.

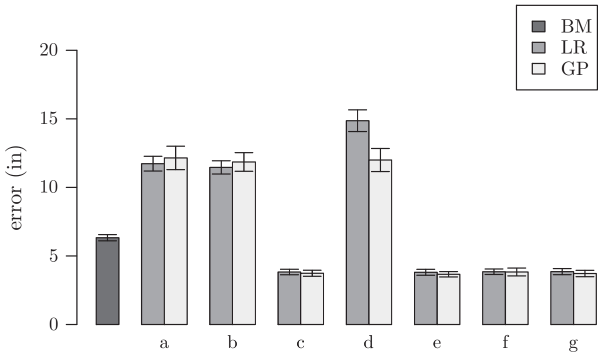


Fig. 9. Experiment Set I results: $\mathcal{HYS}$.

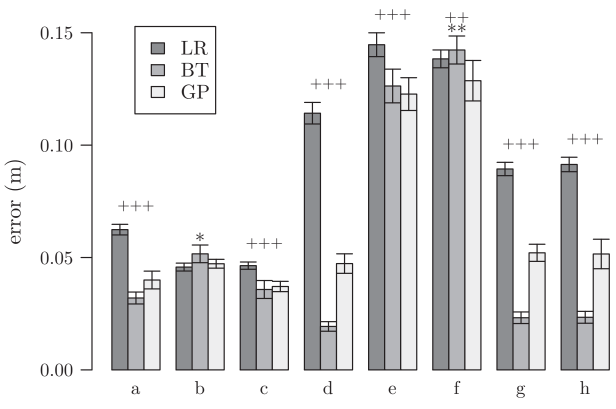


Fig. 10. Experiment Set II (random division) model error.

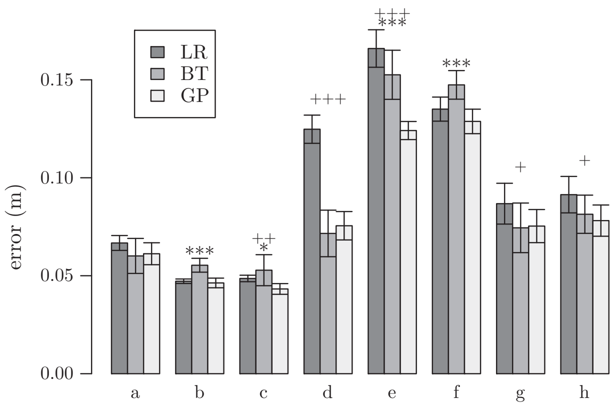


Fig. 11. Experiment Set II (four bins) model error.

Schmucki et al., 2014; Hock and Noetzli, 1997), the inclusion of meteorological inputs does not always improve snowpack model performance (Moeser, 2010), and the inclusion of air temperature data did not improve model performance in Experiment Set I. Therefore, in Experiment Set II we focus on *TOY* and single-point *HS* measurements as predictors of mean catchment *HS*. Instead of manual snow course data as in Experiment Set I, ground-truth data are derived from *HS* measurements collected by the Snowcloud WSN. We compared the performance of three machine learning techniques: LR, BT, and GP.

We developed estimators to predict *HS* at two sites: Sulitjelma, Norway and the Sagehen Experimental Forest, California. At Sulitjelma, model inputs were combinations of *HS* at Balvatn and *TOY*. At Sagehen, model inputs were combinations of *HS* at $\mathcal{HYS}$, *HS* at $\mathcal{IDC}$, and *TOY*. Table 6 summarizes Experiment Set II. We repeated each experiment four times (*Random Division*, *4 Bins*, *3 Bins*, *2 Bins*) and each of these 30 times to generate error samples.

Figs. 10–13 plot the mean values of the samples, i.e. the error of the modeling techniques on testing data. Error bars indicate 95% confidence intervals, i.e. sample mean $\pm(\text{SEM} \times 1.96)$. Stars indicate *p*-values for the Student's paired *t*-test with the hypothesis the GP does not have lower error than BT, i.e. the probability that GP does not outperform BT. One star, $*$, indicates that *p* is less than 0.05, $**$ indicates that *p* is less than 0.01, and $***$ indicates that *p* is less than 0.001. Similarly, plus signs indicate *p*-values for the hypothesis that GP does not have lower error than LR, i.e. the probability that GP does not outperform LR. One plus sign, +, indicates that *p* is less than 0.05, and ++ indicates that *p* is less than 0.01. The mean ground truth *HS* value at Sulitjelma was 1.1900 m. The mean ground truth *HS* value at Sagehen was 0.728 m.

Figs. 14–17 plot the mean sizes of the models whose performance is reported in Figs. 10–13. In the case of GP and BT, these

Table 6
Experiment Set II summary.

Experiment	Location	Model inputs
a	Sulitjelma, Norway	*TOY*
b	Sulitjelma, Norway	*HS* at Balvatn
c	Sulitjelma, Norway	*HS* at Balvatn, *TOY*
d	Sagehen, California	*TOY*
e	Sagehen, California	*HS* at $\mathcal{HYS}$
f	Sagehen, California	*HS* at $\mathcal{IDC}$
g	Sagehen, California	*HS* at $\mathcal{HYS}$, *TOY*
h	Sagehen, California	*HS* at $\mathcal{IDC}$, *TOY*

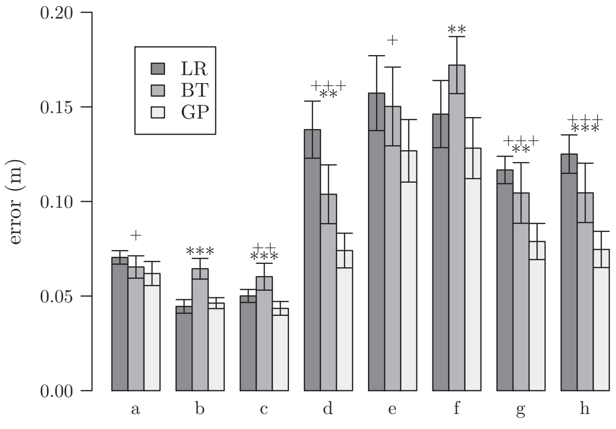


Fig. 12. Experiment Set II (three bins) model error.

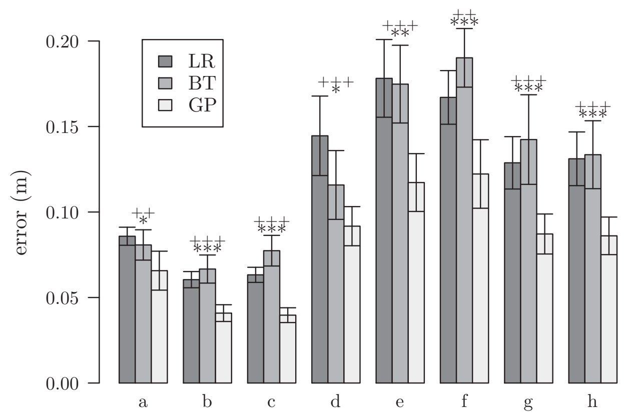


Fig. 13. Experiment Set II (two bins) model error.

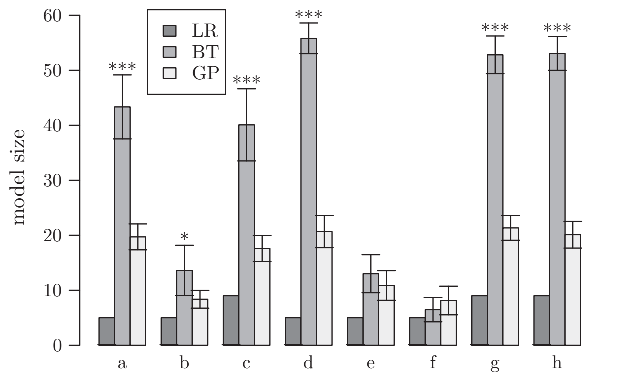


Fig. 14. Experiment Set II (random division) model size.

are the models selected using the *selection set* method. For GP, model size is the number of nodes in the GP tree. For BT, model size is the number of nodes in the binary tree. For LR, model size is the number of operators and values, specifically 5 in the case of a single independent variable and 9 in the case of two independent variables. Stars indicate *p*-values for the Student's paired *t*-test with the hypothesis the GP models are not smaller than BT models. One star, $*$, indicates that *p* is less than 0.05, $**$ indicates that *p* is less than 0.01, and $***$ indicates that *p* is less than 0.001.

6. Discussion

In this section we discuss the results of our experiments, offer some hypotheses to explain our findings, and suggest possible next steps for continued research.

6.1. Experiment Set I

In Experiment Set I GP performed at least as well as other methods in all experiments. This result was expected because GP is capable of generating the same models as LR and BM. We did not perform hypothesis tests comparing GP with LR because visual inspection of error means and 95% confidence intervals (Figs. 6–9) suggests that the methods performed similarly. At the sites where a snow pillow was present, the performance of BM was evaluated. At all of these sites, in all of the experiments where pillow *SWE* was an input variable (b, c, f), both LR and GP performed better (*p*-value less than 0.001) than BM.

These results suggest that machine learning techniques can be used to develop models that predict mean catchment *SWE* more accurately than BM. In general, models performed better when snow pillow data were included. However, GP did not outperform LR.

Because LR performed as well as GP in Experiment Set I, we suspected strict linearity among the explanatory relationships in the data. We hypothesize that because snow courses measure *SWE* only at a single location, they failed to capture existing nonlinearities, and that even though the relationships underlying snowpack distribution are nonlinear, our Experiment Set 1 data is linear. We therefore did not further pursue nonlinear modeling, such as BT, in Experiment Set 1.

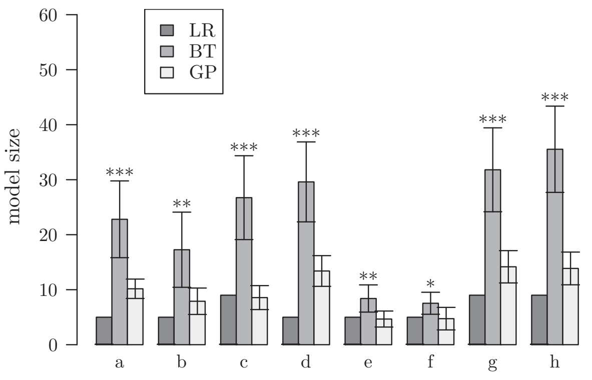


Fig. 15. Experiment Set II (four bins) model size.

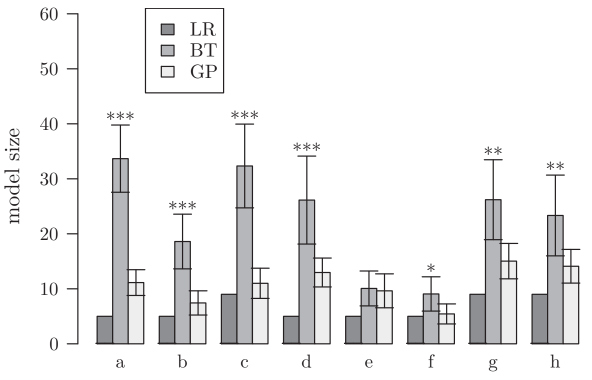


Fig. 16. Experiment Set II (three bins) model size.

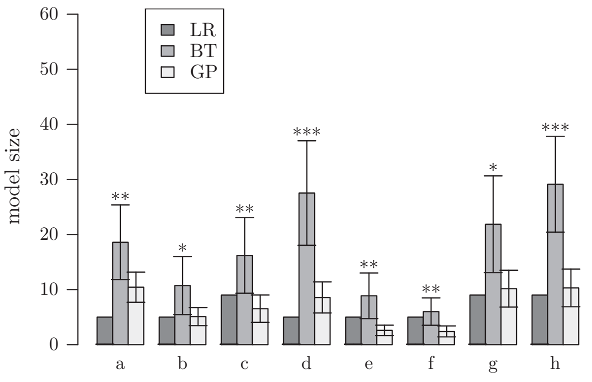


Fig. 17. Experiment Set II (two bins) model size.

6.2. Experiment Set II

First we conducted Experiment Set II: *Random Division*. GP outperformed LR in every experiment except in Norway when the only model input was *HS* at Balvatn. In every experiment in California where *TOY* was an input, BT has much lower error than either GP or LR. In all experiments where *TOY* was an input, the resulting BT models were very large. GP also had lower error and larger model sizes when *TOY* was used then when *TOY* was not used. We had originally introduced the *TOY* variable to allow models to distinguish different parts of the season. However, we hypothesized the BT, and to a lesser extent GP, were abusing the *TOY* variable to memorize snow data by mapping *TOY* data to ground truth *HS*. Even though training and testing data were technically distinct, many of the samples in the testing data were temporally or spatially proximal to samples in the training data. The testing data were not truly unseen with respect to the *TOY* variable. Even though models generalized well to the testing data, they were over-fitting to the *TOY* variable and would likely not generalize to truly unseen data, e.g. from another snow season.

To test this hypothesis and address the possible problem of over-fitting to the *TOY* variable, we repeated Experiment Set II three more times. In Experiment Set II: *4 Bins*, *3 Bins*, and *2 Bins*, we successively decreased the temporal overlap between training and testing data and increase the coarseness of the temporal granularity of the division into training and testing data. Proceeding through this sequence, it became more difficult for machine learning to memorize *HS* data by over-fitting to the *TOY* variable. At the same time, BT error increased and the performance of GP with respect to BT improved. These results suggest that GP is more resilient against over-fitting than BT, possible as a result of multi-objective optimization. Furthermore, when the ability of machine learning to exploit the *TOY* variable by memorizing *HS* the data were minimized, GP significantly outperformed both LR and BT.

6.3. Future work

We believe that the preliminary results discussed in this work are promising and warrant further research into of the applicability of GP to snowpack modeling.

This work should be expanded into a multi-year study. Although Experiment I used snow course data collected over several years, Snowcloud data used in Experiment II was limited to single snow season. A multi-year study would allow models trained on Snowcloud data during one or several years to be evaluated on unseen data from another year. Models trained on multi-year data may be more robust to application in future years than are models trained on single-year data. Even without collecting more data, Experiment Set I could be modified so that models are trained on data from earlier years and tested on data from later years.

Beyond those discussed here, there are many machine learning techniques that should be applied to the problem of catchment-scale *SWE* estimation. GP possesses a unique combination of desirable qualities, but its performance should be compared against other methods such as ANNs, nonlinear multiple regression, and FFX (McConaghy, 2011), a non-evolutionary symbolic regression technology.

The only meteorological input to our models was air temperature. However, meteorological data involving wind, solar radiation, humidity, etc. are available for many locations and have been shown to influence snow distribution (Logan, 1973; Elder et al., 1991; Schmucki et al., 2014; Hock and Noetzli, 1997). Future work should incorporate more potential meteorological predictors of *SWE* and *HS*.

Topographic features significantly shape snow distribution, and models of this relationship have been developed and used extensively (Winstral et al., 2013; Marofi et al., 2011; Chang and Li, 2000; Tabari et al., 2010; Anderton et al., 2004; Grünewald et al., 2013; Molotch et al., 2005; Elder et al., 1998). Although topographic data was not an explicit input in our experiments, models developed with our techniques that use input variables to predict distributed snow measurements likely express some of the relationship between topography and snowpack distribution. Previous efforts to model snowpack using topographic data have derived explicit model inputs from DEMs. The possibility that GP could play an active role in determining which topographical features to use should be explored. GP might discover new methods for extracting information from DEMs that is predictive of snowpack distribution. It is possible that machine learning could use topographic and other data to produce non-cite-specific models, which are trained on data from one or more site and then applied to other sites.

Schwaerzel and Bylander (2006) developed high-order statistical functions for GP to model financial data. These allowed GP models to dynamically select and aggregate a slice of time series data. Future work should apply these techniques to allow GP to determine how to select and aggregate meteorological and topographic data. We made air temperature available to GP by means of functions that aggregate daily measurements over an arbitrary seven day window. Instead, GP could inductively discover how models should dynamically select and aggregate a section of time series data according to changing circumstances.

7. Conclusion

In this paper we have described novel, low-cost methods for catchment-scale *SWE* estimation using machine learning algorithms. The commonly used method of estimating catchment-scale *SWE* from a single point measurement is error-prone because of the spatial heterogeneity of snowpack distribution. We envision an approach wherein short-term field campaigns collect ground-truth data for generating snowpack models which can subsequently augment existing NRT snow telemetry. Toward this end, we explored a suite of machine learning techniques to extrapolate estimates of mean catchment *SWE* from single point *SWE* measurements and other available data and pursued three key research directions. First, we addressed the question of which machine learning approaches are best for this problem. Second, we discussed and pursued the use of a range of possible input parameters. Finally, we grappled with the issue of ground-truthing given limited datasets.

We compared the performance of a basic method (BM) which assumes no spatial variability of *SWE*, linear regression (LR), Genetic Programming (GP), and binary regression trees (BT). We emphasize GP because it produces nonlinear, white-box models without requiring assumptions about model form. GP can be augmented with multi-objective optimization to constrain model complexity and mitigate over-fitting. We found that machine learning techniques generally outperformed BM, demonstrating the spatial variability of *SWE*. Nonlinear techniques outperformed linear models in Experiment Set II, but not in Experiment Set I, suggesting that there are nonlinear relationships among the modeled data used in Experiment Set II. Snowpack distribution at the catchment scale has been shown to be highly nonlinear. It is possible that the spatially distributed sampling technique (Snowcloud wireless sensor network) used for ground-truthing in Experiment Set II captured some of the nonlinearity of snowpack distribution, while the single-location sampling (manual snow courses) used for Experiment Set I did not.

When we naively divided our data at random to generate training and testing data, BT had much lower error than GP in experiments where time of year (*TOY*) was an input variable. In these cases, BT models were much larger than PG models and we suspected that they were memorizing data by mapping *TOY* to snow depth. When we instead divided the data into more temporally contiguous training and testing data in order to prevent this behavior, BT model size decreased and GP outperformed BT.

We emphasize that GP can flexibly incorporate new predictors of catchment-scale *SWE* into the models generated, augmenting its capacity to extrapolate estimates of mean catchment-wide *SWE* from a single point measurement. Genetic programming will make use of input data that helps explain the dependent variable while ignoring data that does not. Our choice of independent variables was a result of intuitive guesses combined with constraints on available data. Topographic information was ruled out because we were unable to determine the precise locations of snow pillows. Multiple forms of meteorological data were available, but air temperature was the most complete, allowing us to compose datasets large enough for effective machine learning. However, the inclusion of air temperature did not have a significant impact on model performance in our first experiment set, and so we did not use any meteorological data in our second experiment set.

Because it has been shown that the forcing effects underlying snowpack distribution change over the course of a snow season, we introduced time of year (*TOY*) as an independent variable so that models can distinguish seasonal differences. However, we found that nonlinear models used *TOY* to memorize the data by mapping *TOY* to ground truth measurements instead of expressing the underlying relationships of snowpack distribution. The ideal solution to this problem would be a multi-year study using spatially distributed data collected by Snowcloud. However, given the limitation of a one year dataset, we modified how data was divided to constrain the temporal proximity of training and testing data.

We conducted two sets of experiments, using available data, as successive approximations of our goal of near-real-time catchment-scale *SWE* estimation. When ground truth was obtained from distributed sampling techniques and when we were careful to mitigate overfitting to the *TOY* variable, GP outperformed other techniques.

Acknowledgments

We would like to acknowledge several individuals for contributions to the content of this paper. Dr. Ian Brown, Stockholm University. Dr. Jeff Frolik, University of Vermont. Dr. Jeff Dozier, University of California. Jeff Brown, University of California. David Moeser, WSL-Institut für Schnee- und Lawinenforschung SLF in Davos, Switzerland. Dr. Keith Klepeis, University of Vermont. Rune Engeset, Norwegian Water Resources and Energy Directorate. Heidi Bache Stranden, Norwegian Water and Energy Directorate.

We acknowledge the Vermont Advanced Computing Core which is supported by NASA (NNX 06AC88G), at the University of Vermont for providing High Performance Computing resources that have contributed to the research results reported within this paper. We acknowledge the support from DARPA through grants W911NF-11-1-0076 and FA8650-11-1-7155. We acknowledge the support from NSF through grant #PECASE-0953837. We acknowledge the support of the Air Force Office of Scientific Research through a YIP grant. This work was supported by NASA under Cooperative Agreement #NNX10AK67H-S02.

References

Adams, W.P., 1976. Areal differentiation of snow cover in east central Ontario. Water Resour. Res. 12, 1226–1234.
Anderton, S., White, S., Alvera, B., 2004. Evaluation of spatial variability in snow water equivalent for a high mountain catchment. Hydrol. Process. 18, 435–453.
Bales, R.C., Molotch, N.P., Painter, T.H., Dettinger, M.D., Rice, R., Dozier, J., 2006. Mountain hydrology of the western united states. Water Resour. Res., 42
Balk, B., Elder, K., 2000. Combining binary decision tree and geostatistical methods to estimate snow distribution in a mountain watershed. Water Resour. Res. 36, 13–26.
Boxalla, B., 2014. California Snowpack Hits Record Low. <http://articles.latimes.com/2014/jan/30/local/la-me-brown-water-20140131>.
Bühler, Y., Christen, M., Kowalski, J., Bartelt, P., 2011. Sensitivity of snow avalanche simulations to digital elevation model quality and resolution. Ann. Glaciol. 52, 72–80.
Chang, K.T., Li, Z., 2000. Modelling snow accumulation with geographic information system. Int. J. Geogr. Inf. Sci. 14, 693–707.
Dozier, J., 2011. Mountain hydrology, snow color, and the fourth paradigm. Eos Trans. Am. Geophys. Union 92, 373–374.
Dozier, J., Painter, T.H., 2004. Multispectral and hyperspectral remote sensing of alpine snow properties. Annu. Rev. Earth Planet. Sci. 32, 465–494.
Durrett, G., Neumann, F., O'Reilly, U., 2010. Computational complexity analysis of simple genetic programming on two problems modeling isolated program semantics. CoRR abs/1007.4636.
Elder, K., Dozier, J., Michaelsen, J., 1991. Snow accumulation and distribution in an alpine watershed. Water Resour. Res. 27, 1541–1552.
Elder, K., Rosenthal, W., Davis, R.E., 1998. Estimating the spatial distribution of snow water equivalence in a montane watershed. Hydrol. Process. 12, 1793–1808.
Engeset, R., Tveito, O.E., Alfnes, E., Mengistu, Z., Udns, H.C., Isaksen, K., Frland, E.J., 2004. Snow map system for Norway, in: Proc. Nordic Hydrol. Conf., p. 12.
Erxleben, J., Elder, K., Davis, R., 2002. Comparison of spatial interpolation methods for estimating snow distribution in the colorado rocky mountains. Hydrol. Process. 16, 3627–3649.
Fassnacht, S., Dressler, K., Bales, R., 2003. Snow water equivalent interpolation for the colorado river basin from snow telemetry (snotel) data. Water Resour. Res., 39
lil-gp Genetic Programming System, 2013. <http://garage.cse.msu.edu/software/lil-gp/>
Grünewald, T., Stotter, J., Pomeroy, J., Dadic, R., Baños, I.M., Marturià, J., Spross, M., Hopkinson, C., Burlando, P., Lehning, M., 2013. Statistical modelling of the snow depth distribution in open alpine terrain. Hydrol. Earth Syst. Sc., 17
GSL, 2014. GNU Scientific Library. <http://www.gnu.org/software/gsl/>
Guan, B., Molotch, N.P., Waliser, D.E., Fetzer, E.J., Neiman, P.J., 2010. Extreme snowfall events linked to atmospheric rivers and surface air temperature via satellite measurements. Geophys. Res. Lett., 37
Hastie, T., Tibshirani, R., Friedman, J., Hastie, T., Friedman, J., Tibshirani, R., 2009. The Elements of Statistical Learning, vol. 2. Springer.
Hock, R., Noetzli, C., 1997. Areal melt and discharge modelling of Storglaciären. Sweden. Ann. Glaciol. 24, 211–217.
Johnson, J.B., Marks, D., 2004. The detection and correction of snow water equivalent pressure sensor errors. Hydrol. Process. 18, 3513–3525.
Jost, G., Weiler, M., Gluns, D.R., Alila, Y., 2007. The influence of forest and topography on snow accumulation and melt at the watershed-scale. J. Hydrol. 347, 101–115.
Kerkez, B., Glaser, S.D., Bales, R.C., Meadows, M.W., 2012. Design and performance of a wireless sensor network for catchment-scale snow and soil moisture measurements. Water Resour. Res., 48
Koza, J.R., 1992. Genetic Programming. Massachusetts Institue of Technology, Cambridge, MA.
Logan, L., 1973. Basin-wide water equivalent estimation from snowpack depth measurements. Role Snow Ice Hydrol, vol. 107. IAHS AIHS Publ., pp. 864–884.
López-Moreno, J., Fassnacht, S., Heath, J., Musselman, K., Revuelto, J., Latron, J., Morán-Tejeda, E., Jonas, T., 2012. Small scale spatial variability of snow density and depth over complex alpine terrain: implications for estimating snow water equivalent. Adv. Water Resour.
Marks, D., Domingo, J., Susong, D., Link, T., Garen, D., 1999. A spatially distributed energy balance snowmelt model for application in mountain basins. Hydrol. Process. 13, 1935–1959.
Marofi, S., Tabari, H., Abyaneh, H.Z., 2011. Predicting spatial distribution of snow water equivalent using multivariate non-linear regression and computational intelligence methods. Water Resour. Manag. 25, 1417–1435.
Martinec, J., Rango, A., 1981. Areal distribution of snow water equivalent evaluated by snow cover monitoring. Water Resour. Res. 17, 1480–1488.
McConaghy, T., 2011. FFX: Fast, scalable, deterministic symbolic regression technology. In: Genetic Programming Theory and Practice IX. Springer, pp. 235–260.
Meromy, L., Molotch, N.P., Link, T.E., Fassnacht, S.R., Rice, R., 2013. Subgrid variability of snow water equivalent at operational snow stations in the western USA. Hydrol. Process. 27, 2383–2400.
Milly, P., Betancourt, J., Falkenmark, M., Hirsch, R., Kundzewicz, Z., Lettenmaier, D., Stouffer, R., 2008. stationarity is dead: whither water management? Science 319, 573–574.
Moeser, C.D., 2010. Development, Analysis and Use of a Distributed Wireless Sensor Network for Quantifying Spatial Trends of Snow Depth and Snow Water Equivalence Around Meteorological Stations With and Without Snow Sensing Equipment. Master's thesis. University of Nevada – Reno.
Moeser, C.D., Walker, M., Skalka, C., Frolik, J., 2011. Application of a wireless sensor network for distributed snow water equivalence estimation, in: Proc. West. Snow Conf., Stateline, NV, USA.
Molotch, N., Colee, M., Bales, R., Dozier, J., 2005. Estimating the spatial distribution of snow water equivalent in an alpine basin using binary regression tree models: the impact of digital elevation data and independent variable selection. Hydrol. Process. 19, 1459–1479.
Molotch, N.P., Bales, R.C., 2005. Scaling snow observations from the point to the grid element: implications for observation network design. Water Resour. Res., 41.

Molotch, N.P., Bales, R.C., 2006. Snotel representativeness in the rio grande headwaters on the basis of physiographics and remotely sensed snow cover persistence. Hydrol. Process. 20, 723–739.

NASA Airborne Snow Observatory, 2015. Measuring Spatial Distribution of Snow Water Equivalent and Snow Albedo. <http://aso.jpl.nasa.gov/>.

National Snow & Ice Data Center, 2014. Clpx overview. <http://nsidc.org/data/clpx/>.

Neumann, F., O'Reilly, U.M., Wagner, M., 2011. Computational complexity analysis of genetic programming – initial results and future directions. In: Riolo, R., Vladislavleva, E., Moore, J.H. (Eds.), Genetic Programming Theory and Practice IX. Springer New York, Genetic and Evolutionary Computation, pp. 113–128.

Ohmura, A., 2001. Physical basis for the temperature-based melt-index method. J. Appl. Meteorol. 40, 753–761.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: machine learning in Python. J. Mach. Learn. Res. 12, 2825–2830.

Pierce, D.W., Barnett, T.P., Hidalgo, H.G., Das, T., Bonfils, C., Santer, B.D., Bala, G., Dettinger, M.D., Cayan, D.R., Mirin, A., et al., 2008. Attribution of declining western US snowpack to human effects. J. Clim., 21

Rittger, K., Kahl, A., Dozier, J., 2011. Topographic distribution of snow water equivalent in the sierra nevada, in: Proc. West. Snow Conf., Western Snow Conference.

Rittger, K.E., 2012. Spatial Estimates of Snow Water Equivalent in the Sierra Nevada. Ph.D. Thesis. University Of California Santa Barbara.

Schirmer, M., Wirz, V., Clifton, A., Lehning, M., 2011. Persistence in intra-annual snow depth distribution: 1. Measurements and topographic control. Water Resour. Res. 47, W09516.

Schmidt, M.D., Vallabhajosyula, R.R., Jenkins, J.W., Hood, J.E., Soni, A.S., Wikswo, J.P., Lipson, H., 2011. Automated refinement and inference of analytical models for metabolic networks. Phys. Biol. 8, 055011.

Schmucki, E., Marty, C., Fierz, C., Lehning, M., 2014. Evaluation of modelled snow depth and snow water equivalent at three contrasting sites in Switzerland using SNOWPACK simulations driven by different meteorological data input. Cold Reg. Sci. Technol. 99, 27–37.

Schwaerzel, R., Bylander, T., 2006. Predicting Financial Time Series by Genetic Programming with Trigonometric Functions and High-Order Statistics, GECCO.

Scipión, D., Mott, R., Lehning, M., Schneebeli, M., Berne, A., 2013. Seasonal small-scale spatial variability in alpine snowfall and snow accumulation. Water Resour. Res. 49, 1446–1457.

Serreze, M.C., Clark, M.P., Armstrong, R.L., McGinnis, D.A., Pulwarty, R.S., 1999. Characteristics of the western united states snowpack from snowpack telemetry (snotel) data. Water Resour. Res. 35, 2145–2160.

Skalka, C., Frolik, J., 2014. Snowcloud: a complete data gathering system for snow hydrology research. In: Real-World Wireless Sensor Networks. Springer, pp. 3–14.

Snow Surveyor, 2014. <http://www.water.ca.gov/floodmgmt/hafoo/hb/sss/surveyor.cfm>.

Sturm, M., Taras, B., Liston, G.E., Derksen, C., Jonas, T., Lea, J., 2010. Estimating snow water equivalent using snow depth data and climate classes. J. Hydrometeorol., 11.

Tabari, H., Marofi, S., Abyaneh, H.Z., Sharifi, M.R., 2010. Comparison of artificial neural network and combined models in estimating spatial distribution of snow depth and snow water equivalent in Samsami basin of Iran. Neural Comput. Appl. 19, 625–635.

Tappeiner, U., Tappeiner, G., Aschenwald, J., Tasser, E., Ostendorf, B., 2001. GIS-based modelling of spatial pattern of snow cover duration in an alpine area. Ecol. Model. 138, 265–275.

USDA, 2014. Snow Surveys and Water Supply Forecasting. <http://www.nrcs.usda.gov/wps/portal/nrcs/detail/or/snow/?cid=nrcs142p2_046152>.

Welch, S.C., Kerkez, B., Bales, R.C., Glaser, S.D., Rittger, K., Rice, R.R., 2013. Sensor placement strategies for snow water equivalent (SWE) estimation in the american river basin. Water Resour. Res. 49, 891–903.

Winstral, A., Elder, K., Davis, R.E., 2002. Spatial snow modeling of wind-redistributed snow using terrain-based parameters. J. Hydrometeorol. 3, 524–538.

Winstral, A., Marks, D., 2014. Long-term snow distribution observations in a mountain catchment: assessing variability, time stability, and the representativeness of an index site. Water Resour. Res. 50, 293–305.

Winstral, A., Marks, D., Gurney, R., 2009. An efficient method for distributing wind speeds over heterogeneous terrain. Hydrol. Process. 23, 2526–2535.

Winstral, A., Marks, D., Gurney, R., 2013. Simulating wind-affected snow accumulations at catchment to basin scales. Water Resour. Res. 55, 64–79.

4.4 Kriegman *et al.* “Evolving spatially aggregated...” (2016).

A technical manuscript describing how symbolic regression of environmental data can produce models interpretable by laypersons follows.

Evolving Spatially Aggregated Features From Satellite Imagery for Regional Modeling

Sam Kriegman, Marcin Szubert, Josh C Bongard, and Christian Skalka

University of Vermont, Burlington VT 05405, USA,
sam.kriegman@uvm.edu

Abstract. Satellite imagery and remote sensing provide explanatory variables at relatively high resolutions for modeling geospatial phenomena, yet regional summaries are often desirable for analysis and actionable insight. In this paper, we propose a novel method of inducing spatial aggregations as a component of the machine learning process, yielding regional model features whose construction is driven by model prediction performance rather than prior assumptions. Our results demonstrate that Genetic Programming is particularly well suited to this type of feature construction because it can automatically synthesize appropriate aggregations, as well as better incorporate them into predictive models compared to other regression methods we tested. In our experiments we consider a specific problem instance and real-world dataset relevant to predicting snow properties in high-mountain Asia.

Keywords: spatial aggregation, feature construction, genetic programming, symbolic regression

1 Introduction

Regional modeling focuses on explaining phenomena occurring at a regional, as opposed to site-specific or global scales [11]. Regional models are of interest in many remote sensing applications, as they provide meaningful units for analysis and actionable insight to policymakers. Yet satellite imagery and remote sensing provide variables at relatively high resolutions. Consequently, studies often involve decisions concerning how to integrate this information in order to model regional processes. Considering measurements at each individual spatial unit as a separate model feature can result in a high dimensional problem in which high variance and overfitting are major concerns. For this reason, spatial aggregation is often applied in this setting to uniformly up-sample variables to be consistent with the response. Although in averaging variables across all spatial units in the region, we discard information which could in turn diminish prediction accuracy and our understanding of underlying phenomena.

Rather than strictly incorporating individual spatial units or uniformly up-sampling, it might instead be beneficial to construct features of a regional model using particularly important subsets of geographical space. In this paper, we move away from uniform up-sampling aggregations towards more flexible and interesting aggregation operations predicated on their subsequent use as features of a regional model. We propose a novel method of inducing spatial aggregations as a component of the machine learning process, yielding features whose construction is driven by model performance rather than prior assumptions.

In experiments designed to explore these techniques, we consider a specific problem and real dataset: estimating regional Snow Water Equivalent (SWE) in high-mountain Asia with satellite imagery. Improved estimation of SWE in mountainous regions is critical [3] but is difficult due in part to complex characteristics of snow distribution [2].

2 Methods

We take a comparative approach to the SWE problem, considering ridge regression, lasso, and GP-based symbolic regression[1]. For each regression model, we consider a filter-based method of feature construction in addition to a second, more dynamic method. For linear regression, we incorporate a wrapper approach in which constructed features and the regression model are induced in separate learning processes, with feedback between the two. For symbolic regression, we use an embedded approach where constructed features and the regression model are induced simultaneously over the course of an evolutionary run.

The Dataset. The SWE dataset[2] is derived from data collected by NASA's Advanced Microwave Scanning Radiometer (AMSR2/E) and Moderate Resolution Imaging Spectroradiometer (MODIS) for March 1 - September 30, in 2003 - 2011, over an area that spans most of the high mountain Asia. We have three explanatory variables measured daily across a 113×113 regular grid for 1935 days: (1) mean and (2) standard deviation of sub-pixel Snow Covered Area [4,10], as well as (3) an estimate of SWE derived from passive microwaves [15]. Our response variable is regional SWE, an attribute of the entire study region, represented as a single value for each of the 1935 days. The response was "reconstructed" by combining snow cover depletion record with a calculation of the melt rate to retroactively estimate how much snow had existed in the region [9].

2.1 Regression Models

Ridge regression [5] is similar to ordinary least squares (OLS) but subject to a bound on the L_2-norm of the coefficients. Because of the nature of its quadratic constraint, ridge regression cannot produce coefficients exactly equal to zero and keeps all of the features in its model. Lasso (Least Absolute Shrinkage and

[1] The source code necessary for reproducing our results is available at `https://github.com/skriegman/ppsn_2016`.

[2] Raw satellite data was pre-processed by Dr. Jeff Dozier (UCSB) using previously reported techniques and is available upon request.

Selection Operator, [16]) modifies the ridge penalty and is subject to a bound on the L_1-norm of the coefficients. The geometry of this L_1-penalty has a strong tendency to produce sparse solutions with coefficients exactly equal to zero. In many high dimensional settings, lasso is the state-of-the-art regression method given its ability to produce parsimonious models with excellent generalization performance. For both lasso and ridge regression, the parameter constraining the coefficients is set through cross-validation.

Genetic Programming (GP, [7]) is a very flexible heuristic technique which can conveniently represent free-form mathematical equations (candidate regression models) as parse trees. GP's inherent flexibility is well-suited for our particular problem because it can efficiently express spatial aggregations and seamlessly combine them into the learning process with minimal assumptions. Furthermore, the "white box" nature of GP may provide physical insights about this complex problem that is currently lacking, as in other domains [1, 13].

To search the space of possible GP trees we use a variant of Age-Fitness Pareto Optimization (AFPO, [12]). AFPO is a multiobjective method that relies on the concept of genotypic age, an attribute intended to preserve diversity. We extend AFPO to include an additional objective of model size, defined as the syntactic length of an individual tree. The size attribute protects parsimonious models which are less prone to overfitting the training data. The GP algorithm therefore identifies the Pareto front using three objectives (all minimized): age, error (fitness), and size. For the fitness objective, we use a correlation-based function rather than pure error, and define $f_{COR} = 1 - |\phi(\hat{s}, s)|$, where $\phi(\hat{s} - s)$ denotes Pearson correlation between model predictions ($\hat{s}$) and actual values of our response (s), regional SWE. Correlation has recently been shown to outperform error-based search drivers given that if a model makes a systematic error it could be easily eliminated by linearly scaling the output and therefore should be protected [14]. Accordingly, for all GP implementations, we apply a linear transformation after f_{COR} -driven evolution has concluded, by using an individual program (model) output as the single input of OLS on the training data.

Our implemented GP experiments used ramped half-and-half initialization with a height range of $2-6$ and an instruction set including unary ($\{\sin, \cos, \log, \exp\}$) and binary functions ($\{\times, +, -, /\}$). One thousand individuals in the population are subject to crossover (with probability 0.75) and mutation (with probability 0.01) over the course of 1000 generations. There is a static limit on the tree height (17) as well as the tree size (300 nodes). Each experiment consists of 30 evolutionary runs, from which the best model (lowest training f_{COR}) is selected. The selected model is then transformed using OLS, and subsequently validated using unseen test data.

Standard Methods. Ridge regression, lasso, and GP may be performed on the raw data using each variable at each individual spatial unit as a separate feature. We denote these methods as Standard Ridge (SR), Standard Lasso (SL) and Standard GP (SGP). SR, SL and SGP each have access to $113 \times 113 \times 3 = 38307$ features, but only 1720 observations in each fold of data.

2.2 Feature Construction Methods

Feature construction is a well studied problem and the utility of genetic programming for feature construction has been recognized in many previous studies [8]. The key difference in our work from this past work is the nature of the data being modeled. We presume that there exist spatial autocorrelations of varying size and shape that, if aggregated to improve the signal to noise ratio, yield features supporting more accurate predictions.

In a regional model, we can construct features by aggregating higher dimensional variables across space. However, it is not entirely clear what kind of aggregations are useful as features of a predictive model. Grouping variables based on similarity or dissimilarity does not necessarily produce useful regional features. In this paper, we make an assumption about the importance of distance and continuity in effective spatial aggregations, based on Tobler's first law of geography [17] which states that "everything is related to everything else, but near things are more related than distant things." Accordingly, we limit the space of possible spatial aggregations to be an average of values within a circular spatial area defined by its centerpoint and radius. However, where to aggregate, how many aggregations to perform, and how to combine the aggregates must still be determined manually or decided during model optimization. We view filters and wrappers as intermediary steps in relaxing assumptions towards our embedded approach, which automates all three of these aspects.

The Filter Method. Filter-based feature construction methods transform or "filter" the original variables as a preprocessing step, prior to modeling. Our filter for the SWE problem represents a static up-sampling transformation of the original variables. Each variable is decomposed in space by a grid of overlapping circles[3] of equal radii centered on a square lattice pattern of points (see Figure 1a,c,e for example). Each constructed feature corresponds to the average (arithmetic mean) of a particular variable sampled within a particular circle of space. Units that reside in an overlapping region of two separate circles are included in the calculation of both features. Since there are three explanatory variables in the SWE dataset, an $R \times R$ grid corresponds to $p = 3R^2$ constructed features. The constructed features are then used as inputs for ridge regression, lasso, and GP, which we will refer to as Filtered Ridge (FR), Filtered Lasso (FL), and Filtered GP (FGP). We will also specify the value of R used in a particular model instance as a subscript, e.g. FR_{15} denotes Filtered Ridge with R=15. We consider filters with $R \in \{1, 2, \ldots, 20\}$, however note that the standard methods are essentially filters with $R = 113$, albeit with the non-overlapping square pixels.

The Wrapper Method. Wrapper-based feature construction methods incorporate feedback from the fit of the model. We implement wrappers around both

[3] The shape of circles are in reality so-called "small circles," as they lie on the surface of earth.

ridge regression and lasso in order to enable the circular sampling regions to define their own center and radius. The circles are no longer fixed on a grid with a predetermined size. Instead, each constructed feature is uniquely parameterized by the coordinates of a center unit (x, y), as a latitude and longitude tuple, and a radius r, as a single value floating point number in km. The center can be any spatial unit in the region, including one at the edge of the raster. The radius is restricted to be within 0 and 1000 km, which is flexible enough to contain only a single unit or span the entire region (see Figure 1b,d for example).

Wrapped Ridge (WR) and Wrapped Lasso (WL) separately use a ridge/lasso-driven hill climbing algorithm to construct features that minimize Mean Absolute Error (MAE), i.e. $\frac{1}{n}\sum_{i=1}^{n}|\hat{s_i} - s_i|$, where s_i is the actual value of our response (regional SWE) and $\hat{s_i}$ is output predicted by the model over n observations. The algorithm uses the same number of circles for each of the three variables, initializing their parameters (x, y, r) randomly. For 1000 iterations, a single constructed feature (circle) is randomly selected and subject to a Gaussian mutation on one of its parameters with standard deviation equal to 25% of the radius and centered at zero. A new ridge/lasso model is then refit on the mutated set of features using a random subset of data sampled without replacement. If the mutation lowered model error on the complementing set of training data left out, then the change is accepted. Otherwise, the mutation is undone. If a proposed mutation to the radius would take it outside the restricted range of $0-1000$ km, then it is "bounced-back" the distance it would have exceeded the boundary. For example, a random mutation that would result in a radius of 1200 km, becomes $1000 - (1200 - 1000) = 800$ km. Thirty restarts are used from which the best model based on training data is selected. We consider $R \in \{1, 2, 3, 4\}$ for wrappers corresponding to $3 \times R^2$ features which really means $3 \times 3 \times R^2$ modifiable parameters.

The Embedded Method. By using GP, we can allow for flexibility with respect to the placement and number of aggregations as well as the way in which they are combined to form a model. However, stochastic optimization methods like GP cannot be easily "refit" in the same manner as deterministic algorithms like ridge regression or lasso. Therefore using wrapper approach for GP is computationally infeasible. Instead, modifications to aggregated features are implemented through mutation-based operators.

In Genetic Programming with Embedded Spatial Aggregation (GPESA) introduced here, our constructed features are represented as parameterized tree terminals, with parameters (x, y, r). Constructed features are randomly initialized in the same manner as the wrapper method, but separately for each terminal of each individual in the population. Greedy Gaussian mutations to the parameters (x, y, r) of a randomly selected constructed feature occur in the population with 20% probability, each generation. Mutations to r have mean zero and a standard deviation of 25%, subject to the bounce-back rule. Similarly, mutations to (x, y) have mean distance zero and a standard deviation of $0.25r$. For 25 iterations, greedy mutations modify the parameterized terminals within a particular GP tree. A modification is accepted if it successfully reduces average error (f_{COR}) on random subsets of training data sampled with replacement. Aside from the stochastic application, another key difference between the wrapper method's hill climbing algorithm and the GPESA's greedy mutations is that the overall regression model stays the same between mutations rather than being "refit" after each mutation.

Validation. In order to validate the generalization of models we partition the dataset into nine overlapping folds. Each fold corresponds to leaving out one year for testing and training on the remaining eight (using years 2003 - 2011). We use MAE on the unseen test data as a metric to assess model performance. To account for a difference in scale across any set of features, all input model features are standardized over time by removing the mean and scaling to unit variance. This means that as wrapper and embedded methods construct new aggregations, the sampled data is scaled over time prior to being averaged over space. Since our goal is near-real-time estimation for a future day, the training values of a feature's mean and variance are reapplied when scaling the same feature in validation.

3 Results

Table 1 displays the test error of each valid regression and feature construction method combination. For filters and wrappers, only the best performing model is displayed and we indicate the particular value of parameter R as a subscript. Since the ultimate goal of our paper is to synthesize a method better than existing approaches, we must statistically compare GPESA to SL, the state-of-the-art linear regression / variable selection algorithm. The null hypothesis of interest here is that of no difference between GPESA and a SL. Therefore we perform yearly Wilcoxon signed rank tests [6] comparing GPESA to SL with Bonferroni correction across the nine years. For five out of the nine test years, GPESA is significantly better than SL, while for the other four years there is no significant difference with SL.

Through displaying only the best testing filters and wrappers, we aim to focus speculation about GPESA performance through a conservative lens. Yet we ultimately view filters and wrappers as intermediary steps "working up" to GPESA. Accordingly, the best test error better represents a bound on the potential performance of a particular intermediary method even though it may not be possible to achieve such performance through a parameter sweep based on the training data. And indeed, across all methods tested, GPESA reported the lowest recorded median mean-absolute error within all but two years (7 of 9) where it has the second lowest.

Year	SR	SL	SGP	FR_4	FL_{19}	FGP_{19}	WR_2	WL_3	GPESA
2003	0.86	0.51	0.35 (0.14)	0.50	0.46	0.44 (0.08)	0.43 (0.10)	0.49 (0.09)	**0.29 (0.09)**
2004	0.47	0.30	0.32 (0.10)	0.34	0.29	0.26 (0.05)	0.37 (0.16)	0.35 (0.16)	**0.17 (0.05)**
2005	0.95	0.44	0.50 (0.13)	0.61	0.40	0.52 (0.06)	0.58 (0.11)	0.63 (0.09)	**0.32 (0.07)**
2006	0.66	0.27	0.41 (0.29)	0.57	0.52	0.36 (0.06)	0.53 (0.11)	0.54 (0.11)	0.27 (0.05)
2007	0.72	0.33	0.44 (0.10)	0.42	0.38	0.34 (0.05)	0.52 (0.13)	0.50 (0.11)	**0.24 (0.06)**
2008	1.46	0.46	0.60 (0.13)	0.71	0.64	0.58 (0.11)	0.70 (0.31)	0.54 (0.26)	0.52 (0.18)
2009	0.81	0.41	0.65 (0.08)	0.90	0.61	0.56 (0.08)	0.98 (0.10)	1.03 (0.09)	0.41 (0.10)
2010	0.62	0.48	0.44 (0.12)	0.43	0.47	0.41 (0.06)	0.43 (0.11)	0.52 (0.11)	**0.32 (0.07)**
2011	0.87	0.48	0.61 (0.17)	0.77	0.60	0.53 (0.10)	0.82 (0.20)	0.93 (0.16)	0.45 (0.12)
Mean	0.82	0.41	0.48	0.58	0.49	0.44	0.58	0.61	0.33

Table 1. Median mean-absolute error with corresponding standard errors in parentheses. Only the best testing filter- and wrapper-based results (choice of R) are displayed. We explicitly compare GPESA with the state-of-art, SL. Bold values indicate significance (at 0.05 level with Bonferroni correction) under a Wilcoxon singed rank test in which the null hypothesis asserts that distribution of the differences between GPESA and SL is symmetrically distributed about 0.

4 Discussion

Our results show that incorporating dynamic aggregations of higher resolution variables into a regional model is beneficial in our particular problem setting, as compared to both uniform up-sampling of variables and a state-of-the-art linear regression technique (SL) that incorporates individual spatial units. SL achieves competitive prediction performance through a sparse linear combination of the individual spatial units, on par with SGP which is not linearly constrained. Ultimately, GPESA performed significantly better (lower median test error) than SL on a majority (5 of 9) of cross validation folds. Moreover, whenever GPESA was not significantly better than SL it was not significantly worse.

A main reason why GPESA has an advantage in this application is the difficulty of knowing a priori what the most important spatial datapoints are, and how to best aggregate them. Additionally, the structure of the model itself is unknown and it depends on the resulting aggregations. Therefore this is not a fixed length optimization problem, which makes it well-suited for GPESA, which can search over different numbers and non-linear combinations of spatial aggregations. While SL can theoretically perform the same aggregation as a GPESA terminal (mean within a radius of a geographical point), SL is restricted to a single linear solution while GPESA is not.

However, it's important to emphasize that the computational cost of GPESA is higher than that of traditional GP and much higher than that of linear regression. In particular, the most expensive operation is the "on the fly" aggregation component of GPESA which makes the fitness evaluation require 500% more time than in SGP. Part of the incurred cost is due to inefficiencies of our implementation that necessitated a copy with all spatial aggregation operations. In future work we will look at reducing this overhead through more efficient data structures (e.g. k-d trees).

Importance of Spatial Data. To better understand the relevance of particular spatial locations, we define the *importance* of a spatial unit for both linear and symbolic methods, separately. For ridge regression and lasso, we can define importance by exploiting the disposition of coefficients to be larger for variables with a stronger correlation to the response, relative to a particular feature set. We define linear regression importance of a particular spatial unit as the average absolute coefficient of features that incorporate the unit into a regression model. While we cannot as easily determine relative importance within nonlinear models, we can instead define importance by exploiting the multiple candidate solutions provided from stochastic multiobjective optimization. We define GP importance of a particular spatial unit as the average absolute correlation $(1 - f_{COR})$ of nondominated solutions that incorporate the unit.

To visualize the importance of spatial information, we generated a series of heatmaps (Figure 1). In Figures 1a, 1c, and 1e we show regional importance values of filter methods for each $R \in \{1, ..., 20\}$, with the relevant value of R annotated in the upper left corner of each box. Note that in lasso- and GP-based approaches, some variables are unused (white), while ridge cannot perform variable selection and uses all. Figures 1b and 1d plot WR and WL for $R \in \{1, 2, 3, 4\}$. Finally, Figures 1e and 1f plot the importance of spatial information in the GP sense, for FGP and GPESA, respectively. Overall, this visualization indicates an agreement among all methods on the relatively higher importance of information in the lower center/right region of the image.

5 Conclusion

In this work we developed a novel method to address the problem of modeling a regional response with high resolution satellite imagery. We moved away from uniform up-sampling aggregations towards more flexible and interesting aggregation operations predicated on their subsequent use as features of a regional model. Our proposed technique, GPESA, is general and intended to apply to a variety of modeling problems on spatially organized data. But as an application example, and as a setting in which to evaluate our techniques, we considered the problem of estimating snow water equivalent in high mountain Asia using satellite imagery. Our results showed that using GP to evolve spatial aggregations outperforms lasso, the state-of-the-art method for directly incorporating individual spatial units into a sparse linear model.

In future work we plan to explore more flexible spatial and temporal aggregations for more predictive modeling in real earth science applications.

Acknowledgements: Thanks to Dr. Jeff Dozier (UCSB) for posing the high-mountain Asia SWE problem and providing associated datasets.

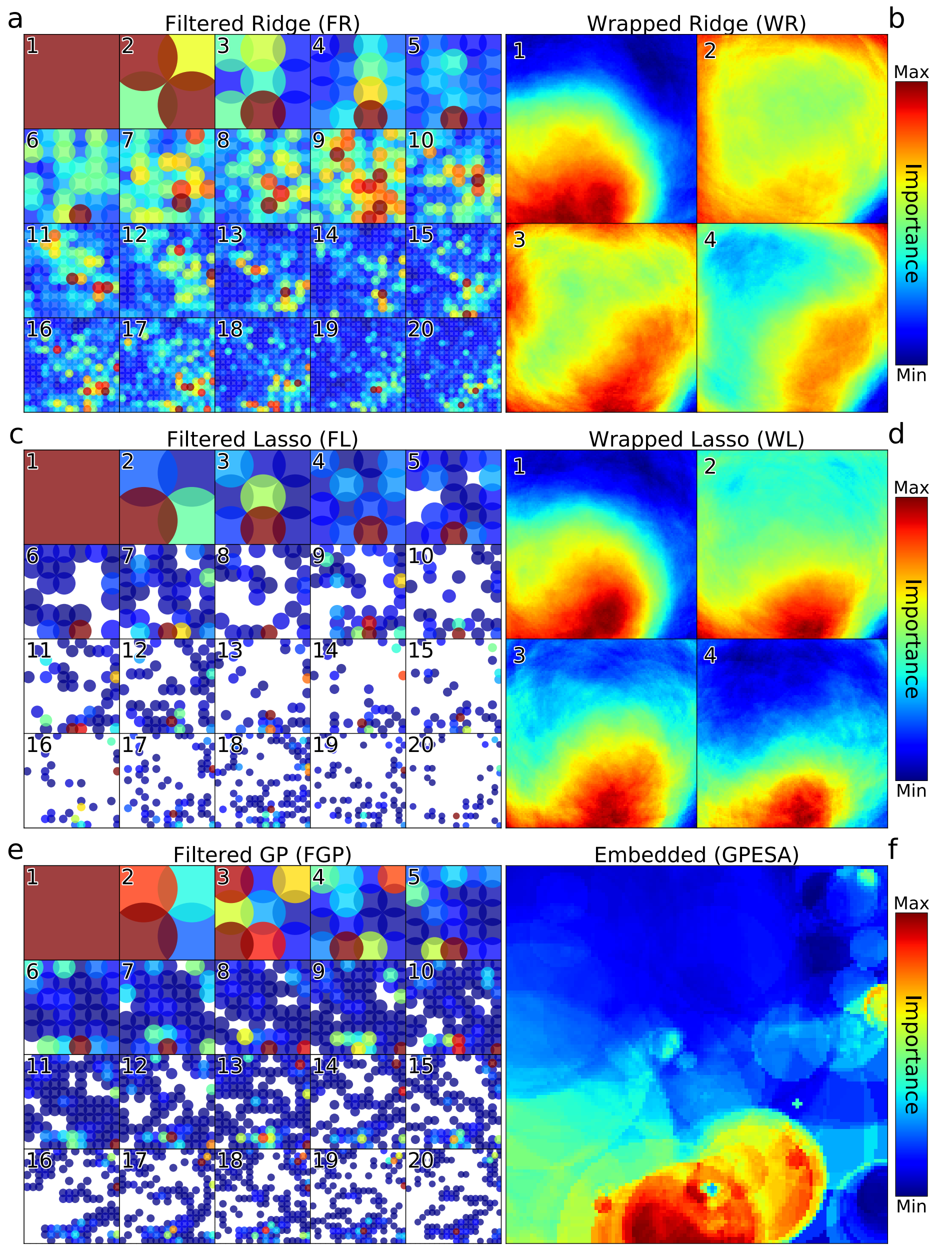


Fig. 1. Importance (defined in Section 4) of spatial units. For filters a.) FR, c.) FL, and e.) FGP, importance is displayed at each resolution $R \in \{1, 2, \ldots 20\}$ and each individual filter subplot is annotated with the corresponding R. For wrappers b.) WR and d.) WL, $R \in \{1, 2, 3, 4\}$. Finally, f.) GPESA, which has no R parameter. White areas indicate spatial units unused in feature construction across all three exploratory variables.

References

1. J. Bongard and H. Lipson. Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 104(24):9943–9948, 2007.
2. D. Buckingham, C. Skalka, and J. Bongard. Inductive learning of snowpack distribution models for improved estimation of areal snow water equivalent. *Journal of Hydrology*, 524:311–325, 2015.
3. J. Dong, J. P. Walker, and P. R. Houser. Factors affecting remotely sensed snow water equivalent uncertainty. *Remote Sensing of Environment*, 97(1):68–82, 2005.
4. J. Dozier, T. H. Painter, K. Rittger, and J. Frew. Time-space continuity of daily maps of fractional snow cover and albedo from MODIS. *Advances in Water Resources*, 31(11):1515–1526, 2008.
5. A. E. Hoerl and R. W. Kennard. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.
6. M. Hollander, D. A. Wolfe, and E. Chicken. *Nonparametric statistical methods.* John Wiley & Sons, 2013.
7. J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, MA, USA, 1992.
8. K. Krawiec. Genetic programming-based construction of features for machine learning and knowledge discovery tasks. *Genetic Programming and Evolvable Machines*, 3(4):329–343, 2002.
9. J. Martinec and A. Rango. Areal distribution of snow water equivalent evaluated by snow cover monitoring. *Water Resour. Res*, 17(5):1480–1488, 1981.
10. T. H. Painter, K. Rittger, C. McKenzie, P. Slaughter, R. E. Davis, and J. Dozier. Retrieval of subpixel snow-covered area, grain size, and albedo from MODIS. *Remote Sensing of Environment*, 113:868–879, 2009.
11. J. Rees, A. Gibson, M. Harrison, A. Hughes, and J. Walsby. Regional modelling of geohazard change. *Geological Society, London, Engineering Geology Special Publications*, 22(1):49–63, 2009.
12. M. Schmidt and H. Lipson. Age-fitness pareto optimization. In R. Riolo, T. McConaghy, and E. Vladislavleva, editors, *Genetic Programming Theory and Practice VIII*, volume 8 of *Genetic and Evolutionary Computation*, pages 129–146. Springer New York, 2011.
13. M. D. Schmidt, R. R. Vallabhajosyula, J. W. Jenkins, J. E. Hood, A. S. Soni, J. P. Wikswo, and H. Lipson. Automated refinement and inference of analytical models for metabolic networks. *Physical biology*, 8(5):055011, 2011.
14. K. Stanislawska, K. Krawiec, and T. Vihma. Genetic programming for estimation of heat flux between the atmosphere and sea ice in polar regions. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, pages 1279–1286. ACM, 2015.
15. M. Tedesco and P. S. Narvekar. Assessment of the nasa amsr-e swe product. *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, 3(1):141–159, 2010.
16. R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.
17. W. R. Tobler. A computer movie simulating urban growth in the detroit region. *Economic geography*, pages 234–240, 1970.

5 Improving symbolic regression.

A large portion of the work conducted under this award involved purely theoretical progress on improving symbolic regression. At the outset of the award, work was completed on hybridizing symbolic- and more traditional regression methods to produce one that is superior to either method working alone [4, 5].

Most recently, we have addressed a major limitation of symbolic regression, which is its scalability. At its heart, symbolic regression uses population-based stochastic optimization: models are randomly modified, and if the modification reduces prediction error, the model is retained; otherwise, it is likely to be discarded. This leads to vast computational waste, as most models produced by these random perturbations perform worse that the originating model.

In [8], we demonstrate that the semantics of different parts of a model can be employed to reduce the randomness of model modification and thus increase the likelihood that a change to a model is beneficial, even for non-convex problems. This is accomplished by removing part of a model, and looking for a complementary part from another model. Connecting these two model parts together keeps the semantics of the resulting model close to the semantics of the contributing models, thus increasingly the likelihood of model improvement.

In [7] we demonstrate how to overcome a major challenge in the field of symbolic regression. In order to stocastically optimization a population of models, it is imperative to maintain diversity in the model population. This is usually done by 'pushing' models away from one another. This however antagonizes the reduction of model error, which 'pulls' the models toward the desired predictions an optimal model should make. In this particular work we show that instead of 'pushing' models away from one another, we incentivize models to spread out, as much as possible, around the desired output of the optimal models. This reduces the antogonism between low error and model population diversity, and leads to significant improvements in the accuracy and parsimony of the final models.

5.1 Relevance for U.S. defense and security.

While the theoretical advances achieved in this part of the project cannot be immediately be applied to defense and security areas of interest, it would be relatively straightforward to incorporate these advances into the three application-specific projects outlined above or incorporated into other applications of symbolic regression of military interest.

5.2 Icke *et al.* "Improving genetic programming..." (2013).

A technical manuscript describing how to hybridize symbolic- and linear regression methods follows.

Improving Genetic Programming Based Symbolic Regression Using Deterministic Machine Learning

Ilknur Icke
Morphology, Evolution and Cognition Lab.
Department of Computer Science
University of Vermont
Burlington, VT 05401
ilknur.icke@uvm.edu

Joshua C. Bongard
Morphology, Evolution and Cognition Lab.
Department of Computer Science
University of Vermont
Burlington, VT 05401
jbongard@uvm.edu

***Abstract*—Symbolic regression (SR) is a well studied method in genetic programming (GP) for discovering free-form mathematical models from observed data. However, it has not been widely accepted as a standard data science tool. The reluctance is in part due to the hard to analyze random nature of GP and scalability issues. On the other hand, most popular deterministic regression algorithms were designed to generate linear models and therefore lack the flexibility of GP based SR (GP-SR). Our hypothesis is that hybridizing these two techniques will create a synergy between the GP-SR and deterministic approaches to machine learning, which might help bring the GP based techniques closer to the realm of *big learning*. In this paper, we show that a hybrid deterministic/GP-SR algorithm outperforms GP-SR alone and the state-of-the-art deterministic regression technique alone on a set of multivariate polynomial symbolic regression tasks as the system to be modeled becomes more multivariate.**

***Index Terms*—symbolic regression, hybrid algorithms, elastic net, regularization**

I. Introduction

Symbolic regression is one the most popular applications of genetic programming and an attractive alternative to standard regression approaches due to its flexibility in generating free-form mathematical models from observed data without any domain knowledge. Indeed, user-friendly genetic programming based symbolic regression (GP-SR) tools such as Eureqa [1] have started to gain more attention from the scientific community over the last couple years. Despite various success stories ([2], [3], [4]) and claims that they will one day 'replace scientists', GP-SR applications (or any evolutionary computation based approach in general) have not yet been widely accepted as standard tools for the data scientists. Although many stochastic optimization algorithms such as stochastic gradient descent (SGD) [5] and metaheuristics such as simulated annealing [6] are well established in the mainstream ML, evolutionary computation methods are generally overlooked. GP suffers from various issues [7] that hinder its applicability to many real-world data science tasks. The theoretical foundations of GP are not as well understood as many of the standard machine learning (ML) algorithms due to the hard to analyze random nature of the technique. Scalability is also a very challenging problem. Efforts to increase scalability of GP via GPUs and cloud computing have been reported (such as in [8], [9]). It is our belief that, if GP-SR is to be a trustable *big learning* [10] tool, it needs to take advantage of the developments in the general ML as well as the parallel and distributed computing techniques.

The idea of studying evolutionary computation techniques from the standard ML perspective is not new. The behavior of GP has been studied in terms of the learning theory in [11] and [12] amongst others. The learnable evolution model (LEM) proposed in [13] is a technique to guide evolutionary processes with standard ML algorithms by creating hypotheses characterizing the differences between high performing and low performing individuals in the population.

Recently, it has been suggested that GP might not be the best option for SR and that stochasticity was not necessarily a virtue. In [14], a deterministic basis function expansion method used in conjunction with a state-of-the-art ML regression algorithm was proposed as an alternative to GP-SR. This algorithm that is known as the Fast Function Extraction (FFX) has been reported to outperform GP-SR on a number of real-world regression problems with dimensionality ranging from 13 to 1468. Our paper shares the same basic ideology, that is, SR should not stray away from the well-established techniques of ML. However, we argue that abandoning the GP approach might not be the best way for SR. Instead, we propose to hybridize the two approaches.

This paper explores one way to incorporate a deterministic ML regression technique into GP-SR in order to improve GP-SR. We report results on a suite of synthetic datasets that were generated to analyze performance as the problem difficulty increases. We believe that analyzing algorithm performance in this manner helps us understand the strengths/weaknesses of the approach before tackling more challenging real-world problems for which the ground truth is hardly ever available.

The organization of this paper is as follows: sections II and III discuss the background and related work. Our proposed algorithm to hybridize the GP-SR and deterministic ML approaches is detailed in section IV. Experimental results are presented and discussed in section V. Finally, section VI discusses conclusions and future work.

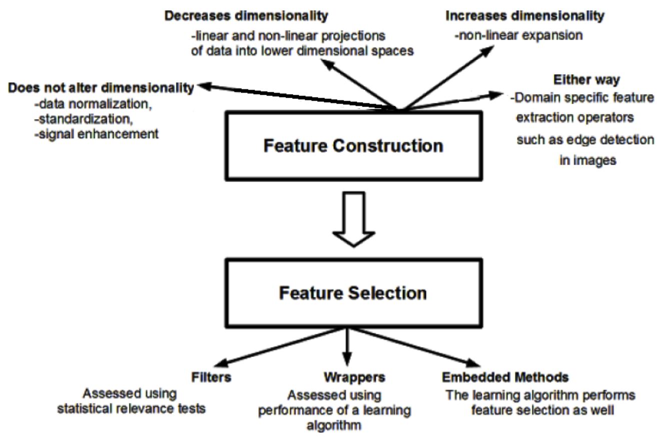

Decreases dimensionality
-linear and non-linear projections of data into lower dimensional spaces
Increases dimensionality
-non-linear expansion
Does not alter dimensionality
-data normalization,
-standardization,
-signal enhancement
Either way
-Domain specific feature extraction operators such as edge detection in images
Feature Construction
Feature Selection
Filters
Assessed using statistical relevance tests
Wrappers
Assessed using performance of a learning algorithm
Embedded Methods
The learning algorithm performs feature selection as well

Fig. 1: Feature extraction as a sequential process of creating features from the input variables and then selecting the most informative features.

II. Background

Data dimensionality poses a great challenge for the numerical and symbolic regression algorithms alike. As the number of predictors increases, it becomes more difficult to identify the informative predictors and to build accurate models. This problem has been well-studied in ML. The task of seeking the best representation for a given dataset in order to optimize the performance of a ML algorithm is known as *feature extraction* [15]. Feature extraction can be seen as a sequential process where new features are first constructed from the input variables and then the most informative ones are selected amongst the constructed features (Fig. 1). Feature construction may or may not increase data dimensionality. If the input variables are suspected to have interactions, it is generally the practice to create additional features via non-linear expansion (such as $x_1 * x_2$).

As for feature selection, the simplest approach is to rank the features with respect to how well they correlate with the predicted variable. This method has the risk of eliminating such features that might not be informative by themselves but might as well be very informative together. Subset selection methods aim to address this issue by considering a subset of features together. These techniques are divided into three main groups: filters, wrappers and embedded methods. *Filters* are pre-processing techniques that, independent from the learning algorithm, select a subset of variables with respect to some criteria such as mutual information. The *wrapper* techniques consider the learning algorithm as a black-box and select the set of features that optimize the performance of the learner. The feature subsets are generated by either forward selection, that gradually adds features or backward elimination, that starts with the whole set of features and eliminates least informative ones. The wrapper approach is computationally expensive as the learning algorithm needs to be executed many times. The *embedded* methods incorporate feature selection within the learning algorithm itself. The decision tree algorithms are the earliest examples of embedded methods. More recent embedded methods utilize the *regularization* technique.

Within the context of the linear regression problem, regularization refers to imposing additional constraints on the coefficients in order to reduce overfitting. In linear regression, given a multivariate dataset $\mathbf{X}_{[M\times N]} = \{\vec{x_1}, \vec{x_2}, ..., \vec{x_N}\}$, a matrix of observations, the response variable $\mathbf{Y}$ is defined as:

$$\mathbf{Y} = f(\mathbf{X}) = \beta_0 + \sum_{j=1}^{N} \beta_j * \vec{x_j}$$

The coefficients are computed via the least squares estimation by minimizing the residual sum of squares over the dataset X:

$$RSS = min_\beta(\sum_{i=1}^{M} y_i - \beta_0 - \sum_{j=1}^{N} \beta_j * x_{ij})^2$$

Since the parameters are computed on the training data, overfitting occurs manifesting itself as large coefficient values. Therefore, an additional constraint on the coefficients is imposed in order to tame the coefficients ($\sum_{j=1}^{N} ||\beta_j||_1 \leq t$). This algorithm that is known as lasso (least absolute shrinkage and selection operator) shrinks the coefficients and also performs feature elimination since the l_1-norm promotes sparsity. Therefore, the coefficients of uninformative features will be close to 0. An l_2-norm constraint is also possible and it is called ridge regression. Ridge regression has the effect of grouping the correlated variables so that they are included in the model together [16], [17]. The elastic net approach [18], [19] is a hybrid of lasso and ridge regression and formulated as:

$$Y = f(X) = \beta_0 + \sum_{j=1}^{N} \beta_j * \vec{x_j} + \lambda_2||\beta||_2^2 + \lambda_1||\beta||_1$$

Generally, λ_1, λ_2 are balanced by defining one single parameter ($0 \leq \alpha \leq 1$) that is called the mixing parameter. At the extreme values of α, elastic net behaves like purely lasso or purely ridge regression. A very large value of λ forces all βs to be 0. As λ is relaxed, the coefficients start to take nonzero values. This sweep of λ values can be visualized as a regularization path (Fig. 2). The algorithm is named elastic net since the "regularization path is like a stretchable net that retains all the big fish"[18].

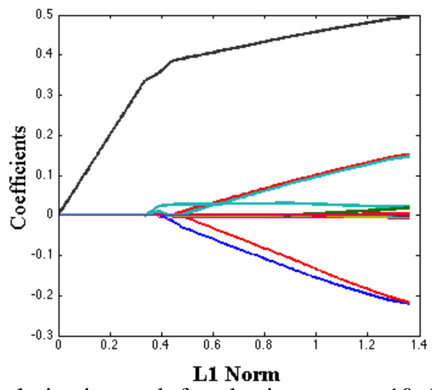


Fig. 2: Regularization path for elastic net on a 10-dimensional dataset. For each λ, the l_1-norm of the coefficients vector versus individual coefficient values are shown. Each line traces the change of coefficient values for one variable. At the beginning, no features are selected. Gradually, more features are added into the models as the coefficients become non-zero.

This basic linear regression algorithm applies to the generalized linear models (GLM) of the form:

$$Y = f(X) = \beta_0 + \sum_{j=1}^{N} \beta_j * b_j(X)$$

where $b_j(X)$ are nonlinear basis functions applied to the input variables in order to construct new features.

III. Related Work

GP-SR inherently performs feature selection when it finds sufficiently accurate data models; any feature that does not appear on the evolved expression can be considered redundant. However, when data dimensionality is high, the search space grows exponentially, making it difficult for GP-SR to find good solutions. The issue of feature selection in GP-SR algorithms have been studied by various researchers. Koza's automatically defined functions (ADF) [20] can be seen as a feature extraction method within the context of SR. A Pareto-GP based variable selection scheme was proposed in [21]. In [22], permutation tests were introduced in GP-SR to discriminate between informative and uninformative variables. In [23], feature selection capabilities of GP-SR and random forests were compared. The authors report that when it finds an accurate model, GP-SR captures the important features that are missed by the random forests algorithm.

The regularization approach has been applied to GP-SR in [24] for polynomial functional form discovery. The authors incorporate a function smoothness term into the fitness function as a way to decrease overfitting. The Fast Function Extraction (FFX) algorithm reported in [14] employs a nonlinear basis function expansion method that creates new features via unary and binary interactions of the input variables. The algorithm does not employ GP-SR to construct the features or the models. The new features are created in a deterministic manner and passed to the elastic net algorithm for model building. The algorithm generates multiple models for the λs on the regularization path. The non-dominated set of these models with respect to accuracy versus complexity are identified as the final models.

The difference of our proposed technique is that we perform feature extraction using an efficient deterministic ML algorithm and pass the features to GP-SR for model building. By taking advantage of the state-of-the-art ML, our algorithm aims to ease the burden of GP-SR in feature extraction and help it excel in model building.

IV. Improving GP-SR using Deterministic ML

The technique we propose in this paper has been largely inspired by the FFX algorithm [14]. However, the author had proposed to eliminate the GP for the symbolic regression problem in favor of a deterministic way to augment the dataset with polynomial features and then use a state-of-the-art machine learning algorithm (elastic net) for model building. In this paper, we propose to hybridize GP with the deterministic ML techniques so as to take advantage of the strengths of both approaches to solve symbolic regression problems more accurately and efficiently in comparison to either technique alone. The outline of the general idea behind FFX is presented in algorithm 1 (for a detailed description of the FFX algorithm, see[14]). The algorithm consists of three stages: feature construction, model building and model selection. The feature construction stage creates new features by applying binary nonlinear interactions (basis functions) and augmenting the original dataset (algorithm 2). It is possible to go beyond the binary interactions; however, this would increase the number of constructed features exponentially. In this paper, we considered only unary and binary features as in [14]).

Algorithm 1: The basic FFX algorithm

```
Input: V={v_1,v_2, ..., v_N}
Output: The set of non-dominated evolved models based on
        validation data error-model complexity (number of
        bases) trade-off
1  [ bases, expandedTrainingDataset] =
   basisFunctionExtraction(trainingDataset)
2  models={}
3  foreach $\alpha \in (0, 0.05, 0.1, ..., 1)$ do
4      models = models ⋃ glmnetfit(variables,trainingDataset)
5      nonDominatedModels =
6      extractParetoFrontier(models,expandedValidationDataset)
7      models=nonDominatedModels
8      models = models ⋃
       glmnetfit(bases,expandedTrainingDataset)
9      nonDominatedModels =
10     extractParetoFrontier(models,expandedValidationDataset)
       models=nonDominatedModels
11 end
```

The model building stage utilizes the coordinate descent elastic net algorithm (glmnet, line 4 of algorithm 1) that was proposed in [19]. As it is shown in Fig. 2, for each value of λ, one can build an expression using the corresponding coefficients. Therefore, the model building stage returns multiple expressions containing different numbers of basis functions.

Algorithm 2: *basisFunctionExtraction* : Polynomial basis function generation as new features form the observed data

```
Input: V={v_1,v_2, ..., v_N}
Output: Expanded Dataset: $V_e$={v_{e1},v_{e2}, ..., v_{eM} }
1  //Generate unary bases
2  foreach $v_1, v_2, ..., v_N$ do
3      unaryBases = unaryBases ⋃ $v_i$
4      foreach $exp_j$ do
5          unaryBases = unaryBases ⋃ $v_i^{exp_j}$
6      end
7      foreach $unaryOperator_k$ do
8          unaryBases = unaryBases ⋃ $unaryOperator_k(v_i)$
9      end
10 end
11 //Generate binary bases
12 foreach $u_i \in unaryBases$ do
13     foreach $u_j \in unaryBases$ do
14         binaryBases= binaryBases ⋃ $u_i * u_j$
15     end
16 end
```

Note that the models are built on the training data. At the model selection stage, the non-dominated set of models with respect to error on validation data versus expression complexity (the number of basis functions or bases for short) are identified.

Our proposed method to hybridize FFX/GP-SR is presented in algorithm 3. The process starts with a *variant* of FFX that was outlined in algorithm 1. From the set of all non-dominated models generated by FFX, all unique features (unary and binary) are extracted (line 2). These are the features that were found by FFX to be the most informative features for the given regression problem. Fig. 3 summarizes the process of identifying these features from the FFX output. Across all the models on the non-dominated set, each base is extracted and the coefficients are eliminated. The identified list of unique unary and binary basis functions are then utilized to create the new dataset with corresponding feature labels. The new dataset is then passed onto the GP-SR for model building.

Algorithm 3: The hybrid FFX/GP-SR algorithm

```
Input: V={v_1,v_2, ..., v_N}
Output: One best model with respect to the validation data
        error and complexity
1 nonDominatedModels = ffx(trainingDataset)
2 bases = extractBasisFunctions (nonDominatedModels,
3                                 validationDataset)
4 newDataset=createNewDataset(bases)
5 bestModel = GP-SR(newDataset)
```

We hypothesized that for higher dimensional problems, pre-processing the dataset using a fast algorithm such as FFX would increase the chances of the GP-SR to succeed as opposed to expecting the GP-SR to perform feature extraction and model building simultaneously. The algorithm shown above may extract many basis functions for high dimensional datasets. In that case, further filtering of the uninformative features created by those basis functions can be done before passing the features to the GP-SR.

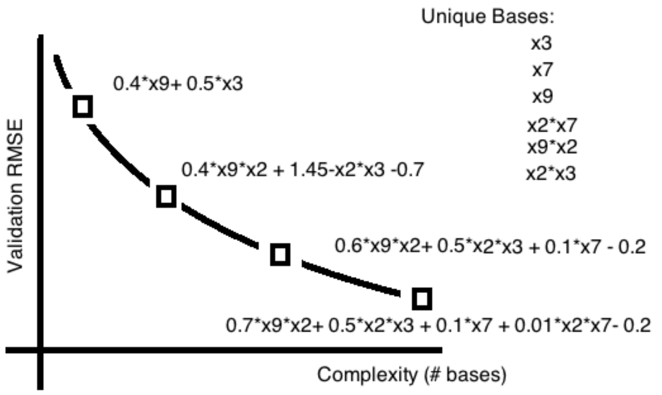

Unique Bases:
x3
x7
x9
x2*x7
x9*x2
x2*x3
0.4*x9+ 0.5*x3
0.4*x9*x2 + 1.45-x2*x3 -0.7
0.6*x9*x2+ 0.5*x2*x3 + 0.1*x7 - 0.2
0.7*x9*x2+ 0.5*x2*x3 + 0.1*x7 + 0.01*x2*x7- 0.2
Validation RMSE
Complexity (# bases)

Fig. 3: Generation of the new dataset based on FFX-generated expressions (extractBasisFunctions, line 2 of algorithm 3). Most frequent bases are extracted from the Pareto frontier and used as features for the new dataset.

V. Experimental Results

We implemented our GP-based Symbolic Regression application using the GPTIPS Matlab package downloaded from [25]. Our version of the FFX algorithm and FFX/GP-SR algorithms were also implemented in Matlab using the glmnet package downloaded from [26] and the GPTIPS package. All experiments were run on a cluster computing environment.

A. Synthetic Benchmark Data Suite and Evaluation Procedure

We tested our algorithms on a systemically generated suite of multivariate polynomial functions in order to analyze the performance as the difficulty of the problem is increased in terms of the number of variables(1-3, 10), order of the polynomial (1-4) and the number of basis functions each polynomial contains (1-4). Examples of such functions are presented in the following sections. For each polynomial, 2500 data points were generated as training points and separate sets of 1250 data points were held aside as validation and test data. All input variables were randomly sampled within the range [0,1].

The evaluation procedure is as follows: for each type of polynomial (such as order 2 with 2 bases), there are 30 different datasets generated by the 30 different polynomials of that type. For each such polynomial, we perform 30 independent GP-SR and FFX/GP-SR runs with 1 minute runtime budget. Since FFX is deterministic it runs only once. For FFX, the final set of non-dominated models are recorded. For GP-SR and FFX/GP-SR the best model with respect to the validation dataset is recorded for each run. In summary, for each type of polynomial, 900 runs of GP and 900 runs of FFX/GP-SR runs are performed.

Unlike the general approach where a close approximation with respect to the prediction error is satisfactory for evaluation of the success, in this paper, we also assess the outcomes in terms of how close the functional form of the hidden target expression is matched. For instance, if the hidden target expression is $\alpha_1 * x_1 + \alpha_2 * x_2 + \beta$, where α_i, β are real valued coefficients, we consider each evolved model with low prediction error that matches this functional form as a successful outcome regardless of the actual values of the coefficients. Namely, the degree of similarity between the hidden ground truth and the evolved polynomials is defined in terms of syntactic similarity.

For FFX, the evaluation is performed based on the whole set of non-dominated models (Fig. 4). If a model with the correct syntactic form exists in this set, then the FFX run is considered a success. For GP and FFX/GP, all 30 runs per unique polynomial are examined. If a model with the correct syntactic form exists in this set, the algorithm is considered successful on discovering that polynomial. We also record the syntactic similarity to the correct polynomial form. The similarity values range between 0 and 1; 1 meaning a perfect match to the true syntactic form and 0 meaning no match at all. For each unique polynomial, the model with best validation error is identified and its syntactic similarity and test error values are recorded as the outcomes for each algorithm.

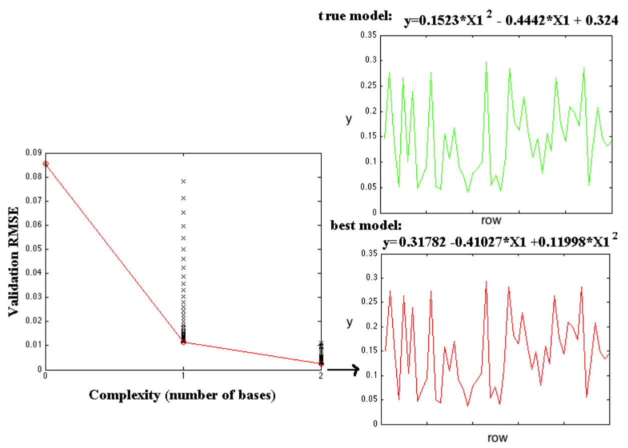


Fig. 4: Result of an FFX run on a second order 1D problem with two basis functions. The final model shown here is selected from the Pareto frontier as the one with lowest validation error.

In the following sections, we present the results separately for five sets of data organized with respect to dimensionality. We start from the simplest case: one variable and various polynomials ranging from linear with a single term to fourth order with four terms. We increase the number of variables to 2,3,10 then to 25 and repeat the same set of experiments. Finally, statistical significance tests are utilized in order to check if hybridizing the GP-SR with FFX helps improve the performance of GP-SR given the data dimensionality.

B. Results on the benchmark problems

Unless otherwise specified, all GP-SR and FFX runs were performed using the following default parameters:

TABLE I: Default GP-SR parameters

Parameter	Value
Representation	GPTIPS [25] Multigene syntax tree Number of genes: 1 Maximum tree depth: 7
Population Size Runtime Budget	500 1 minute
Selection Tournament Size	Lexicographic tournament selection 7
Crossover Operator Crossover Probability	Sub-tree crossover 0.85
Mutation Operator Mutation Probability	Sub-tree mutation 0.1
Reproduction Probability	0.05
Building Blocks	Operators: $\{+, -, *, protected/\}$ Terminal Symbols: $\{x_1, ..., x_N\}$
Fitness	$\frac{1}{N}\sqrt{\sum(y-\hat{y})^2}$
Elitism	Keep 1 best individual

1-dimensional polynomials: The following polynomials are examples of the 1-dimensional hidden target expressions (ground truth) used in our experiments. The polynomials are grouped with respect to the highest order variable interaction. Within each group, the syntactic complexity of the expressions increase as more basis functions are included gradually. For each expression, the number in the paranthesis on the left-hand side indicates how many types of nonlinear interactions (i.e unary, binary,...) are included in that expression.

TABLE II: Default FFX parameters

Parameter	Value
Basis Function Expansion	Exponents : 1 Interactions : Unary, Binary Operators : { }
Elastic Net	α : $\{0, 0.05, 0.1, ..., 1\}$ λ : 100 λ values calculated by glmfit based on α Maximum basis functions allowed : 250
Model Selection	Non-dominated models with respect to validation data error versus number of bases

- order 1 polynomial:

$$(1)\quad y = 0.288 * x_1 + 0.8446$$

- order 2 polynomials:

$$(1)\quad y = 0.14 * x_1^2 + 0.629$$
$$(2)\quad y = 0.12 * x_1 + 0.03 * x_1^2 + 0.29$$

- order 3 polynomials:

$$(1)\quad y = -0.31 * x_1^3 - 0.11$$
$$(2)\quad y = 1.35 * x_1^2 - 0.83 * x_1^3 + 0.139$$
$$(3)\quad y = 0.13 * x_1 + 0.44 * x_1^2 + 0.34 * x_1^3 + 0.39$$

- order 4 polynomials:

$$(1)\quad y = 0.20 * x_1^4 + 0.13$$
$$(2)\quad y = 0.24 * x_1^3 + 0.23 * x_1^4 + 0.39$$
$$(3)\quad y = 0.75 * x_1^2 + 0.30 * x_1^3 + 0.35 * x_1^4 + 0.334$$
$$(4)\quad y = 0.02 * x_1 + 0.13 * x_1^2 + 0.301 * x_1^3 + 0.32 * x_1^4 + 0.91$$

Tables III, IV and V show the number of successful runs for each algorithm for each type of polynomial. As the results of the GP-SR runs indicate, even the 1-dimensional hidden target expressions become more challenging as the number of basis functions increases. Out of the 30 runs, the proportion of successful discovery of the correct functional form declines as the syntactic complexity of the target expressions increase.

TABLE III: Standalone GP-SR runs on 1D datasets (1 minute).

		Bases			
		1	2	3	4
Order of the Polynomial	1	30	-	-	-
	2	30	29	-	-
	3	30	27	19	-
	4	30	27	11	16

TABLE IV: FFX runs on 1D datasets with unary (x_i) and binary interactions ($x_i * x_j$) (average run time: 7 seconds)

		Bases			
		1	2	3	4
Order of the Polynomial	1	30	-	-	-
	2	30	30	-	-
	3	0	0	0	-
	4	0	0	0	0

TABLE V: FFX/GP-SR runs on 1D datasets (1 minute GP run on FFX-generated dataset)

		Bases			
		1	2	3	4
Order of the Polynomial	1	30	-	-	-
	2	30	27	-	-
	3	30	26	19	-
	4	30	28	16	17

It is not surprising that FFX did not succeed at all when the target polynomials were cubic and fourth order, as we have only allowed for unary and binary basis functions. Our goal was to test how much FFX might help GP-SR discover 3rd and 4th order polynomials utilizing binary bases only. Fig. 5 shows that the hybrid did not outperform the plain GP-SR even on the quadratic polynomials as the problem was easy for GP-SR to handle within the given runtime budget.

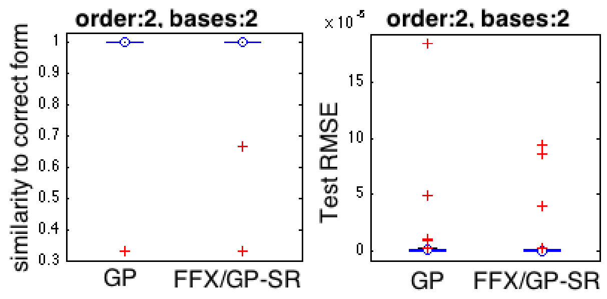


Fig. 5: Summary of runs on second order 1D polynomials with 2 basis functions. According to Wilcoxon rank sum tests, FFX/GP-SR does not outperform GP-SR in 1 minute runtime

2-dimensional polynomials: We repeated the experiments using a set of 30 polynomials for each listed form below:

- order 1 polynomial:

$$(1)\quad y = 0.62 * x_2 - 0.854$$

- order 2 polynomials:

$$(1)\quad y = 0.22 * x_1^2 + 0.05$$
$$(2)\quad y = 0.12 * x_1 - 0.25 * x_1 * x_2 + 0.4$$

- order 3 polynomials:

$$(1)\quad y = 1.67 * x_1^2 * x_2 + 0.46$$
$$(2)\quad y = 0.17 * x_1 * x_2 + 0.369 * x_2^3 - 0.3$$
$$(3)\quad y = 0.03 * x_2 - 0.36 * x_1^2 + 0.22 * x_2^3 + 0.42$$

- order 4 polynomials:

$$(1)\quad y = 2.88 * x_1^2 * x_2^2 + 0.15$$
$$(3)\quad y = 0.4978 * x_1 * x_2^3 - 0.08 * x_1^4 + 0.36$$
$$(3)\quad y = 2.19 * x_1 * x_2^2 - 0.87 * x_2^3 + 0.87 * x_1^2 * x_2^2 + 0.39$$
$$(4)\quad y = 0.13 * x_2 - 1.313 * x_1 * x_2 - 0.1 * x_1^3$$
$$0.4926 * x_1^2 * x_2^2 + 0.19$$

TABLE VI: Standalone GP-SR runs on 2D datasets (1 minute).

		Bases			
		1	2	3	4
Order of the Polynomial	1	30	-	-	-
	2	30	29	-	-
	3	30	22	15	-
	4	30	20	10	2

TABLE VII: FFX runs on 2D datasets with unary (x_i) and binary interactions ($x_i * x_j$) (average run time: 9 seconds)

		Bases			
		1	2	3	4
Order of the Polynomial	1	30	-	-	-
	2	30	16	-	-
	3	0	0	0	-
	4	0	0	0	0

Similar to the 1-dimensional case, we found that FFX/GP-SR did not outperform GP-SR on 2-dimensional polynomial dataset (Fig. 6).

TABLE VIII: FFX/GP-SR runs on 2D datasets (1 minute GP run on FFX-generated dataset)

		Bases			
		1	2	3	4
Order of the Polynomial	1	30	-	-	-
	2	30	30	-	-
	3	30	19	14	-
	4	30	20	11	3

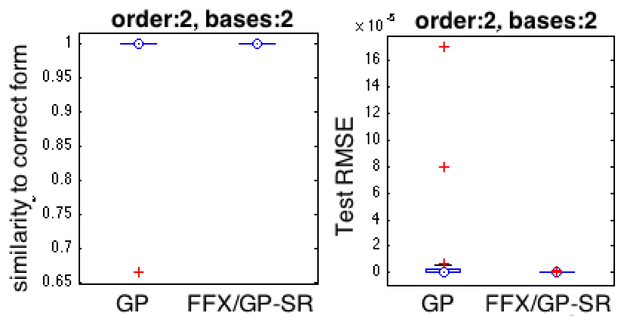


Fig. 6: Summary of runs on second order 2D polynomials with 2 basis functions. According to Wilcoxon rank sum tests, FFX/GP-SR does not outperform GP-SR in 1 minute runtime

3-dimensional polynomials: We repeated the experiments using a set of 30 polynomials for each listed form below::

- order 1 polynomial:

$$(1)\quad y = 0.746 * x_3 + 0.8268$$

- order 2 polynomials:

$$(1)\quad y = 0.54 * x_3^2 + 0.4$$
$$(2)\quad y = 0.8651 * x_1 - 0.61 * x_2^2 - 0.30$$

- order 3 polynomials:

$$(1)\quad y = 0.84 * x_1 * x_2 * x_3 - 0.86$$
$$(2)\quad y = 0.93 * x_1 * x_2 - 0.46 * x_3^3 + 0.88$$
$$(3)\quad y = 0.04 * x_2 - 0.18 * x_2 * x_3 - 0.01 * x_1 * x_2^2 + 0.3$$

- order 4 polynomials:

$$(1)\quad y = 0.20 * x_1 * x_2^3 + 0.91$$
$$(2)\quad y = 0.73 * x_1^2 * x_2 - 0.07 * x_1^2 * x_2 * x_3 + 0.39$$
$$(3)\quad y = 1.2 * x_1 * x_2 + 0.68 * x_1^2 * x_2 +$$
$$0.48 * x_1^2 * x_2 * x_3 + 0.41$$
$$(4)\quad y = 0.35 * x_3 - 0.32 * x_2 * x_3 - 0.35 * x_1 * x_2^2 -$$
$$0.39 * x_3^4 + 0.24$$

TABLE IX: Standalone GP runs on 3D datasets (1 minute)

		Bases			
		1	2	3	4
Order of the Polynomial	1	30	-	-	-
	2	30	25	-	-
	3	30	25	9	-
	4	30	13	12	3

TABLE X: FFX runs on 3D datasets with unary (x_i) and binary interactions ($x_i * x_j$) (average run time: 12 seconds)

		Bases			
		1	2	3	4
Order of the Polynomial	1	30	-	-	-
	2	29	16	-	-
	3	0	0	0	-
	4	0	0	0	0

TABLE XI: FFX/GP-SR runs on 3D datasets (1 minute GP run on FFX-generated dataset)

		Bases			
		1	2	3	4
Order of the Polynomial	1	30	-	-	-
	2	30	26	-	-
	3	30	28	14	-
	4	30	17	12	6

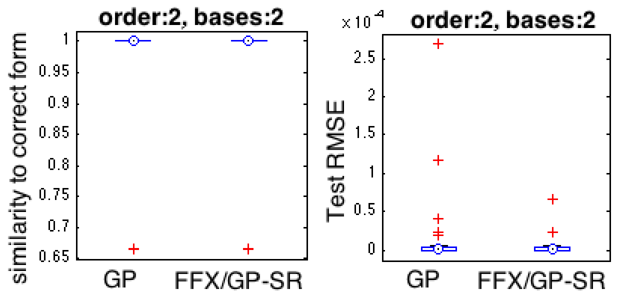


Fig. 7: Summary of runs on second order 3D polynomials with 2 basis functions. According to Wilcoxon rank sum tests, FFX/GP-SR does not outperform GP-SR in 1 minute runtime

Second Order Polynomials from 10 & 25-dimensional Datasets: In order to test our intuition that the FFX/GP-SR would perform better for higher dimensional data, we raised the dimensionality of synthetic data to 10 and 25. In this section, we present results of GP-SR and FFX/GP-SR runs on 30 second order polynomial functions with 2 basis functions such as the following: $y = 0.7 * x_3 - 0.23 * x_9 * x_7 + 0.2$ where $x_i \in x_1, ..., x_{10}$ and $x_i \in x_1, ..., x_{25}$ respectively.

Since we only allowed unary and binary interactions in our FFX implementation, FFX/GP-SR did not significantly do better than GP-SR alone in terms of finding the correct functional form of the hidden polynomials with orders greater than 2. This was evident in 1:3-dimensional polynomial experiments reported in the previous section. Therefore, we only performed runs on the second order polynomials for the 10 and 25-dimensional data. As in the previous sections, the results reported here are aggregated over the runs on 30 different polynomials for a runtime budget of 1 minute..

On the 10-dimensional dataset, the FFX algorithm found the correct syntactic form (identified the correct variables and linear form) for 10 out of the 30 polynomials. The GP-SR algorithm by itself found 14 out of the 30 and the FFX/GP-SR hybrid found 22 out of the 30 target polynomial forms correctly. On the 25-dimensional dataset, the number of times each algorithm found the correct functional form was 18,1 and 26 out of the 30 target polynomials for the FFX, GP-SR and FF/GP-SR respectively.

Fig. 8 and Fig. 9 summarize the comparisons of GP and FFX/GP-SR algorithms based on the similarity to the correct polynomial form and prediction errors. As the dimensionality increases from 10 to 25, the performance of the GP-SR declines sharply in terms of recovering the correct functional form within the given runtime budget of 1 minute. The FFX/GP-SR hybrid, on the other hand, continues to succeed as the dimensionality increases. In summary, the utility of the hybrid algorithm becomes more significant as the number of variables increases. The hybrid algorithm discovers expressions that are significantly more similar to the ground truth and significantly more predictive.

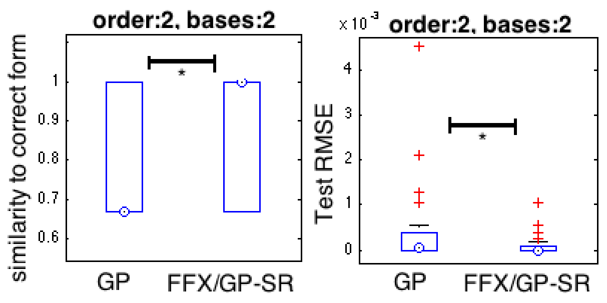


Fig. 8: Summary of runs on second order 10D polynomials with 2 basis functions. The final expressions found by FFX/GP-SR are significantly more similar to the ground truth as opposed to GP-SR alone (Wilcoxon rank sum right-tailed test, $\alpha = 0.05$, p-value:0.0198) and more predictive (Wilcoxon rank sum left-tailed test, $\alpha = 0.05$, p-value:0.005)

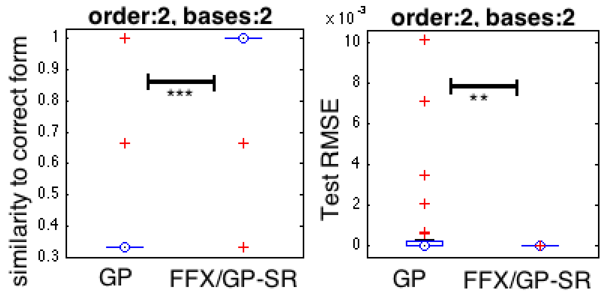


Fig. 9: Summary of runs on second order 25D polynomials with 2 basis functions. The final expressions found by FFX/GP-SR are significantly more similar to the ground truth as opposed to GP-SR alone (Wilcoxon rank sum right-tailed test, $\alpha = 0.001$, p-value $<< 0.001$) and more predictive (Wilcoxon rank sum left-tailed test, $\alpha = 0.01$, p-value$<<$0.01)

C. Discussion

Even though the hybrid algorithm did not provide additional advantage over plain GP-SR on low dimensional datasets, our results indicated that, as the data dimensionality increased (10 and then to 25, in this case), the FFX/GP-SR hybrid performed significantly better in finding more predictive expressions that are more similar to the hidden ground truth in comparison to GP-SR alone. By similar, we mean the success at which the algorithm captures the informative variables and their nonlinear interactions.

Based on our experiment results, we note that even though the FFX algorithm might not always find the correct functional form for the target expressions itself, the rich set of building blocks it provides to the GP-SR has the potential to boost the performance of the GP-SR. Since the GP-SR search space grows exponentially as the number of variables increases, eliminating the uninformative variables beforehand using a deterministic ML algorithm helps ease the burden of discovering the informative variables and constructing the useful nonlinear interactions for model building.

VI. Conclusion

Although GP-SR has been known for a couple of decades, only recently that tools such as Eureqa have started to attract a larger number of scientists due to almost zero-maintenance and user-friendly application interfaces along with various improvements on the metaheuristics and options for parallel and distributed computing. However, GP-SR is yet to be accepted as a standard data analysis tool. In this paper, we argued that the resistance from the ML community is not totally unfounded. First of all, theoretical underpinnings of the GP-SR such as converge proofs are not as well established as standard deterministic algorithms. GP-SR is computationally more expensive compared to most standard ML algorithms and even though many intuitive strategies might be built into the algorithm, there is no guarantee that optimal data models will emerge at the end of the run. More importantly, despite all the success stories, GP-SR techniques do not necessarily outperform the state of the art in ML, especially on high dimensional problems. On the other hand, the strength of GP-SR is in its model-free nature which makes it possible that the algorithm might discover optimal and more intelligible, novel models for the observed data. In summary, it is our belief that stochasticity can be a virtue for SR if it is directed intelligently.

In this paper, we showed that it would be possible to create synergy between the deterministic ML and GP-SR approaches by hybridizing them. The technique presented in this paper is just one way out of many possible options to combine the GP-SR and standard ML for regression problems. Here, we incorporated building blocks extracted by the deterministic regression algorithm into the GP-SR algorithm by means of re-creating the input dataset. Another option is to seed the GP-SR runs with the candidate solutions found by the deterministic approach. Genetic programming can also be used to evolve features (via the generation of the basis functions) that can be fed to the deterministic algorithm for model generation. Our current work focuses on investigating other possible ways to hybridize GP-SR and deterministic ML based approaches in order to address high dimensional real-world regression problems.

Acknowledgment

This research was supported by a DARPA grant #FA8650-11-1-7155. The authors also acknowledge the Vermont Advanced Computing Core which is supported by NASA (NNX 06AC88G), at the University of Vermont for providing High Performance Computing resources that have contributed to the research results reported within this paper.

References

[1] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," *Science*, vol. 324, no. 5923, pp. 81–85, 2009.

[2] J. Koza, "Human-competitive results produced by genetic programming," *Genetic Programming and Evolvable Machines*, vol. 11, pp. 251–284, 2010.

[3] "Robot biologist solves complex problem from scratch," http://cacm.acm.org/careers/136345-robot-biologist-solves-complex-problem-from-scratch/fulltext, October 2011.

[4] R. Dubčáková, "Eureqa: software review," *Genetic Programming and Evolvable Machines*, vol. 12, no. 2, pp. 173–178, Jun. 2011.

[5] "Stochastic optimization: Icml 2010 tutorial," http://www.ttic.edu/icml2010stochopt/.

[6] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[7] M. O'Neill, L. Vanneschi, S. Gustafson, and W. Banzhaf, "Open issues in genetic programming," *Genetic Programming and Evolvable Machines*, vol. 11, no. 3-4, pp. 339–363, Sep. 2010.

[8] "Genetic programming on general purpose graphics processing units," http://www.gpgpgpu.com/.

[9] D. Sherry, K. Veeramachaneni, J. McDermott, and U.-M. O'Reilly, "Flex-gp: genetic programming on the cloud," in *Proceedings of the 2012 European conference on Applications of Evolutionary Computation, EvoApplications'12*, 2012, pp. 477–486.

[10] "Big learning, algorithms, systems and tools," http://biglearn.org/.

[11] N. Amil, N. Bredeche, C. Gagn, S. Gelly, M. Schoenauer, and O. Teytaud, "A statistical learning perspective of genetic programming," in *Genetic Programming*, ser. Lecture Notes in Computer Science, L. Vanneschi, S. Gustafson, A. Moraglio, I. Falco, and M. Ebner, Eds. Springer Berlin Heidelberg, 2009, vol. 5481, pp. 327–338.

[12] M. Castelli, L. Manzoni, S. Silva, and L. Vanneschi, "A quantitative study of learning and generalization in genetic programming," in *Genetic Programming*, ser. Lecture Notes in Computer Science, S. Silva, J. Foster, M. Nicolau, P. Machado, and M. Giacobini, Eds. Springer Berlin Heidelberg, 2011, vol. 6621, pp. 25–36.

[13] R. S. Michalski, "Learnable evolution model: Evolutionary processes guided by machine learning," *Machine Learning*, vol. 38, pp. 9–40, 2000.

[14] T. McConaghy, "Ffx: fast, scalable, deterministic symbolic regression technology," *Genetic Programming Theory and Practice IX, Edited by R. Riolo, E. Vladislavleva, and J. Moore,Springer*, 2011.

[15] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[16] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society (Series B)*, vol. 58, pp. 267–288, 1996.

[17] K. P. Murphy, *Machine Learning A Probabilistic Perspective*. The MIT Press, 2012.

[18] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society Series B*, vol. 67, no. 2, pp. 301–320, 2005.

[19] T. H. Jerome Friedman and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *Journal of Statistical Software*, vol. 33, no. 1, January 2010.

[20] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*. Cambridge, MA, USA: MIT Press, 1992.

[21] G. Smits, A. Kordon, K. Vladislavleva, E. Jordaan, and M. Kotanchek, "Variable selection in industrial datasets using pareto genetic programming," in *Genetic Programming Theory and Practice III*, ser. Genetic Programming, T. Yu, R. Riolo, and B. Worzel, Eds. Springer US, 2006, vol. 9, pp. 79–92.

[22] R. K. McRee, "Symbolic regression using nearest neighbor indexing," in *Proceedings of the 12th annual conference companion on Genetic and evolutionary computation,GECCO '10*, 2010, pp. 1983–1990.

[23] S. Stijven, W. Minnebo, and K. Vladislavleva, "Separating the wheat from the chaff: on feature selection and feature importance in regression random forests and symbolic regression," in *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation,GECCO '11*, 2011, pp. 623–630.

[24] N. Nikolaev and H. Iba, "Regularization approach to inductive genetic programming," *Evolutionary Computation, IEEE Transactions on*, vol. 5, no. 4, pp. 359 –375, 2001.

[25] D. Searson, "Gptips package for matlab," https://sites.google.com/site/gptips4matlab/.

[26] H. Jiang, "Glmnet for matlab," http://www-stat.stanford.edu/ tibs/glmnet-matlab/.

5.3 Icke *et al.* “Modeling hierarchy...” (2013).

A technical manuscript describing how symbolic regression can find hidden hierarchy in data follows.

Modeling Hierarchy using Symbolic Regression

Ilknur Icke
Morphology, Evolution and Cognition Lab.
Department of Computer Science
University of Vermont
Burlington, VT 05401
ilknur.icke@uvm.edu

Joshua C. Bongard
Morphology, Evolution and Cognition Lab.
Department of Computer Science
University of Vermont
Burlington, VT 05401
jbongard@uvm.edu

***Abstract*—Symbolic Regression is an attractive modeling approach because it can capture and present, mathematically, relationships between variables of interest. However, given n variables to model, symbolic regression returns a flat list of n equations. As the number of state variables to be modeled scales, interpretation of such a list becomes difficult. Here we present a symbolic regression method that detects and captures hidden hierarchy in a given system. The method returns the equations in a hierarchical dependency graph, which increases the interpretability of the results. We demonstrate that two variations of this hierarchical modeling approach outperform non-hierarchical symbolic regression on a synthetic data suite.**

***Index Terms*—hierarchy, dependency graph, data mining**

I. Introduction

Hierarchical relationships abound in natural and man-made systems. Hierarchy is thought to be a fundamental characteristic of many complex systems such as biological organisms [1], ecological systems [2], the Internet, and traffic networks [3] and, arguably, social organizations [4]. The human visual system is known to be organized hierarchically [5], where the lower level components process the sensory stimuli and the higher levels process the output of the lower level components. Many artificial neural network architectures used for pattern recognition tasks were also designed based on this principle [6]. More recently, deep belief networks [7] attempt to discover the hierarchical structure hidden in large data sets by learning several layers of hierarchically organized features. These natural/artificial hierarchical systems have been evolved/trained to be able to respond to a wide variety of stimuli. In this scheme, each component is specialized in processing a subset of the inputs coming from the lower level.

Our goal is to be able to automatically reverse engineer hierarchical systems in order to understand which inputs each component is processing and uncover the nature of the process (i.e how each component is computing its output from its inputs). In this paper, we focus on systems where the inputs of the individual components are not overlapping (Fig. 1) such as the non-overlapping perceptron or biological neural networks with non-overlapping receptive fields [8], [9].

Here we show that if traditional symbolic regression is applied (in which each variable is modeled separately but allowed to be described as a function of every other variable) little progress is made on increasingly large yet hierarchical

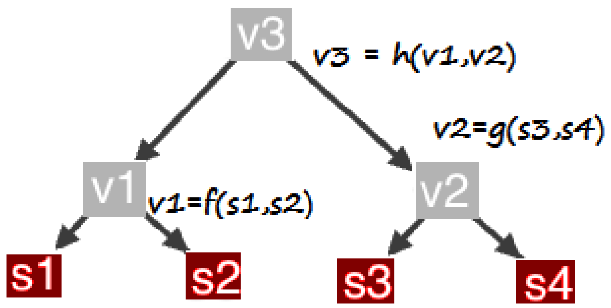


Fig. 1: Dependency graph for a hierarchical system with non-overlapping inputs. The leaf nodes are the stimuli (controlled) and internal nodes are the state variables whose behaviors are observed. The direction of the arrows indicate dependency that is opposite to the information flow in the system.

target systems. This is because as the number of variables increases, more independent runs must be performed (one for each variable), and each run is more difficult because there are more variables to make use of. In contrast we present two variations of symbolic regression adapted for hierarchical systems: the variables that are directly influenced by variables at the lowest level of the hierarchy are identified and modeled first, followed by the variables at the next-highest level but which are restricted to using the variables on the layer below them. We find that hierarchical approach significantly outperforms the traditional symbolic regression paradigm on a number of synthetic datasets that vary in difficulty.

This paper is organized as follows: section II presents the problem in terms of a motivating example and section III discusses the related work. Our proposed approaches for automatic identification of hierarchy are presented in section IV. Experimental results are presented in section V. Finally, conclusions and future work are discussed in section VI.

II. Identifying Hierarchy

Many systems are made up of multiple components that interact with each other by receiving inputs from and/or transmitting outputs to other components. In some cases, these components are hierarchically organized where the information flows in a bottom up manner. In a hierarchical organization, the output of each component depends on the inputs it receives from the components at the lower levels.

Fig. 1 shows a very simple hierarchical system that receives 4 stimuli (s_1, s_2, s_3, s_4). The stimuli are then processed by the two components (v_1 and v_2) and their outputs are passed on to the top component v_3. In this system, the stimuli are provided by the environment (or controlled by an experimenter) and the behavior of v_1, v_2 and v_3 are observed. It is important to note that, only the stimuli are known and the responses of v_1, v_2 and v_3 to these external inputs are observed, but the actual connectivity between the components are not known. The goal is to identify which components are connected and how they are connected.

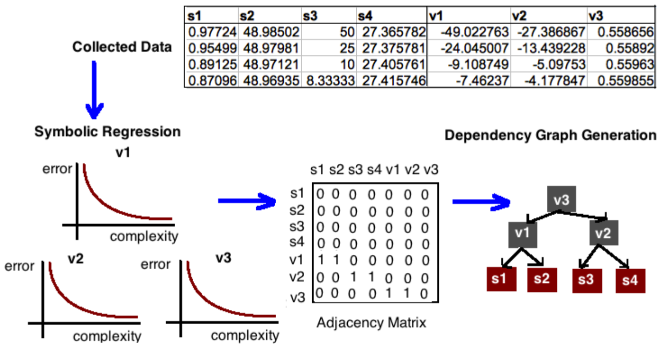
In order to understand the nature of the relationships between the inputs and outputs of each component (the functions f, g and h in Fig. 1), one would employ a regression approach. In this regard, being a free-form modeling technique, symbolic regression can be seen as a more flexible approach as opposed to the mainstream linear regression techniques.

The algorithms we present in this paper are built around a genetic programming based implementation of symbolic regression (SR) in order to model the relationships between the components of an hierarchical system. In symbolic regression, the most predictive variables will appear in the evolved expressions eliminating the non-informative variables. In the context of this paper, if a variable is predictive of another, we say that the predicted variable depends on it. The nature of this dependency can be further examined by looking at the evolved expression relating the predicted variable to its predictors.

III. Related Work

A genetic programming based method for modeling the ODEs for gene regulatory networks was presented in [10]. The algorithm independently evolves expressions to explain each state variable on the observed time series data and does not explicitly model hierarchy. This work was followed by several other papers that produced flat lists of ODEs when exposed to multivariate datasets [11], [12]. A linear genetic programming based reverse engineering algorithm for neuronal networks was presented in [13]. Starting from the observed data, the algorithm tries to infer the structure and parameters of the system. Each state variable is evolved with respect to the neuroscience domain knowledge that is built into the algorithm which limits the use of the algorithm beyond that specific problem domain.

The idea of building variable interaction networks from multivariate data was explored in several papers. An algorithm that extracts a linear dependency structure from multivariate data without explicitly modeling hierarchy was reported in [14]. The limitation of the algorithm is that it uses linear regression in order to construct a linear dependency tree or forest of the variables. Linear models are easier to build and they are intuitive, however there is no guarantee that the phenomenon under study is governed by linear relationships. In [15], multiple genetic programming based symbolic regression runs are executed for each variable separately and a variable interaction network is built by identifying the most relevant variables for a given target variable in terms of a measure of relative frequencies of variable appearances in the expressions modeling that target variable. The algorithm does not assume any specific network topology such as hierarchy.



s1	s2	s3	s4	v1	v2	v3
0.97724	48.98502	50	27.365782	-49.022763	-27.386867	0.558656
0.95499	48.97981	25	27.375781	-24.045007	-13.439228	0.55892
0.89125	48.97121	10	27.405761	-9.108749	-5.09753	0.55963
0.87096	48.96935	8.33333	27.415746	-7.46237	-4.177847	0.559855

$$\begin{array}{c|ccccccc}
 & s1 & s2 & s3 & s4 & v1 & v2 & v3 \\
\hline
s1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
s2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
s3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
s4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
v1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
v2 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
v3 & 0 & 0 & 0 & 0 & 1 & 1 & 0
\end{array}$$

Adjacency Matrix

Fig. 2: The workflow of the NSR algorithm

IV. Symbolic Regression Approach to Model Hierarchy in Multivariate Data

In this section, we present three approaches to automatically extract the hierarchical relationships in multivariate data. The Naive SR algorithm models each state (non-stimuli) variable separately using symbolic regression. After the run is completed, the hierarchy is extracted by examining the expressions and identifying which variables depend on each other. The other two approaches are iterative that aim to enforce hierarchy during the search for the optimal symbolic expressions.

A. *The Naive SR*

In identifying the relationships between multiple variables, a straight-forward approach is to model each variable separately in terms of all other variables using symbolic regression. In doing this, one would expect that the best models would reveal the most informative (highly predictive) variables for each modeled variable. After the symbolic regression phase is done, constructing the hierarchy is just a matter of post-processing. Because each variable is modeled independently, the algorithm does not impose any constraints on the connectivity. The workflow of the Naive SR is presented in Fig. 2 and the steps of the method are outlined in algorithm 1.

After each non-stimulus variable is modeled separately on the training dataset using symbolic regression (the evolve phase), the post-processing phase begins. At this stage, the set of all non-dominated models for each variable are evaluated on the validation dataset. Then, the best model for each variable is chosen (line 8 of algorithm 1) as the model with the lowest error on the validation dataset. The ties are broken in favor of the simplest model. Following the selection of the best models, the variables appearing in these models are identified as the predictors for each respective modeled variable (line 10). An adjacency matrix is then built (line 13) based on these identified predicted variable-predictor mappings. Finally, the algorithm returns the adjacency matrix representing the connectivity between the variables along with the set of best models evolved for each non-stimulus variable.

Algorithm 1: Naive SR (NSR) Algorithm

```
Input: V={v_1,v_2, ..., v_N}
Output: Dependency graph/forest G and a set of
        expressions E={v_i=f_i(v − {v_i})}
1  while time budget not exceeded do
2      foreach v_i do
3          Evolve v_i = f_i(V- {v_i}) on training set
4      end
5  end
6  E={}, D={ }
7  foreach v_i do
8      g_i=FindBestOnParetoFront(f_i) on validation set
9      E=E ∪ {g_i}
10     d_i =ExtractPredictorSet(g_i)
11     D = D ∪ {d_i}
12 end
13 G=BuildGraph(V,D)
```

Algorithm 2: Hierarchical SR (HSR) Algorithm v.1

```
Input: V={v_1,v_2, ..., v_N}
Output: Dependency graph G and a set of expressions
        E={v_i=f_i(v − {v_i})}
1  E={}
2  I={s_1,s_2,...s_N}
3  C=V, D={ }
4  NumEpochs=#Dependent Variables
5  timeBudget = totalTimeBudget/NumEpochs
6  while not all dependent variables are modeled do
7      while time budget not exceeded do
8          foreach v_i do
9              Evolve v_i = f_i(V- {v_i}) on training set
10         end
11     end
12     g_i=FindBestModel(f_i) on validation set
13     E=E ∪ {g_i}
14     d_i =ExtractPredictorSet(g_i)
15     D = D ∪ {d_i}
16     C = C - {v_i}
17     I= I - {d_i }
18     I =I ∪ {v_i}
19 end
20 G=BuildGraph(V,D)
```

B. *Hierarchical SR version 1 (h1)*

In this method, we enforce hierarchical extraction of the dependencies in an iterative manner. At each iteration, dependencies for one non-stimulus variable are discovered. The algorithm starts with only the stimuli as the set of available independent variables. After first iteration, the variable (v_i) that is best explained by a subset of these inputs is determined. Then, all predictors for variable (v_i) are removed from the set of available independent variables in accordance with our constraint that inputs can not overlap. Next, v_i is added to the list of independent variables. The algorithm stops after each (non-stimulus) variable has been modeled using symbolic regression. The method is outlined in algorithm 2.

As opposed to the naive algorithm, the HSR version 1 works in epochs. At the end of each epoch, one variable that is modeled the best is selected and eliminated from the set of dependent variables. This selection is done by identifying one best model across all models for all variables (line 12 of algorithm 2). For N dependent variables, there are N non-dominated sets at the end of each epoch. A combined non-dominated set is then generated in terms of validation set error and the expression complexity (Fig. 3). Throughout this paper, expression complexity is computed as the number of nodes in the expression tree.

Once the best model on the combined non-dominated set is selected (the model with the lowest validation data error), the corresponding dependent variable and its predictors are identified. Since these predictors can only be used to model the identified dependent variable due to our non-overlapping inputs assumption, they are removed from the list of possible inputs for modeling other variables. The identified dependent variable is added to the list of possible predictors for other variables that are waiting to be modeled. This process actively enforces the extraction of hierarchical relationships and generates a set of easily interpretable expressions instead of a flat list of unstructured expressions.

Reconsidering the motivating example in Fig. 1, it is easy to see that the top-level component v_3 can also be modeled as $v_3 = h'(s_1, s_2, s_3, s_4)$. However, such an expression will be dominated by the less complex but equally fit expressions for v_1 and v_2 upon combining all non-dominated sets as in Fig. 3.

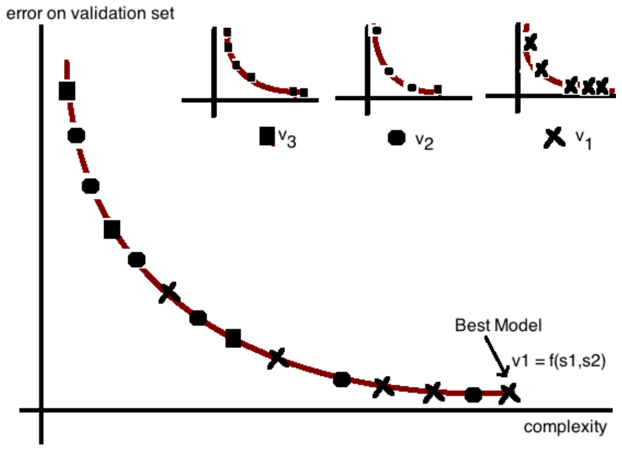


Fig. 3: Selection of the best model on the combined non-dominated set in HSR version 1

C. *Hierarchical SR version 2 (h2)*

The first version of the hierarchical SR algorithm pools all evolved model in an epoch and makes the selection based on the best model on the validation data set. In the second version of the hierarchical algorithm, we modify this selection strategy. Instead of identifying one best model across all variables and then extracting the predictors, we try rather the opposite approach. We first identify the best non-dominated set and then find the set of most frequently occurring predictors across all non-dominated models for the corresponding modeled variable. An additional evolve step is performed in order to find the best expression based on the identified predictor variables only.

The reasoning behind this strategy is that making use of the statistics about how the current set of dependent variables are used across the Pareto front, rather than just how those variables are used in the model with lowest error, might improve its performance compared to h1. The method is outlined in algorithm 3. Similar to the h1 algorithm, the h2 algorithm also runs in epochs. The only difference is in the selection of the best model and the predictor variables at the end of each epoch.

Algorithm 3: Hierarchical SR (HSR) Algorithm v.2

```
Input: V={v_1,v_2, ..., v_N}
Output: Dependency graph G and a set of expressions
        E={v_i=f_i(v − {v_i})}
1  E={}
2  I={s_1,s_2,...s_N}
3  C=V, D={ }
4  NumEpochs=#Dependent Variables
5  timeBudget = totalTimeBudget/NumEpochs
6  while not all dependent variables are modeled do
7      while time budget not exceeded do
8          foreach v_i do
9              v_i =Evolve( f_i(V- {v_i}) ) on training set
10         end
11     end
12     b_i=FindBestModeledVariable(v_i) on validation set
13     d_i =ExtractPredictorSet(b_i)
14     b_i= Evolve( f_i({d_i}) ) on training set
15     g_i=FindBestModel(f_i) on validation set
16     E=E ∪ {g_i}
17     D = D ∪ {d_i}
18     C = C - {b_i}
19     I= I - {d_i }
20     I =I ∪ {b_i}
21 end
22 G=BuildGraph(V,D)
```

The selection process is summarized in Fig. 4. For each modeled variable, all non-dominated models are evaluated on the validation dataset and new non-dominated sets based on the validation dataset error versus expression complexity are built (line 12 of algorithm 3). For each non-dominated set (ns), the fitness is computed as the weighted sum of the error on validation dataset:

$$fitness(ns) = \sum_{i=1}^{M} error * complexity$$

The non-dominated set with lowest weighted error is selected for further processing. Ties are broken in favor of the smallest total expression complexity. Again, reconsidering the motivating example in Fig. 1, where v_3 can also be modeled as $v_3 = h'(s_1, s_2, s_3, s_4)$, the weighted error fitness will penalize v_3 because of the higher total complexity of the expressions.

The next step considers the models that are in the non-dominated set for the selected variable only. A histogram of unique occurrences of each predictor is generated and the set of most frequently occurring predictors are selected by identifying the cut-off point on the histogram. The final model is then generated via symbolic regression using only the selected predictors as the terminal symbols (lines 14-15).

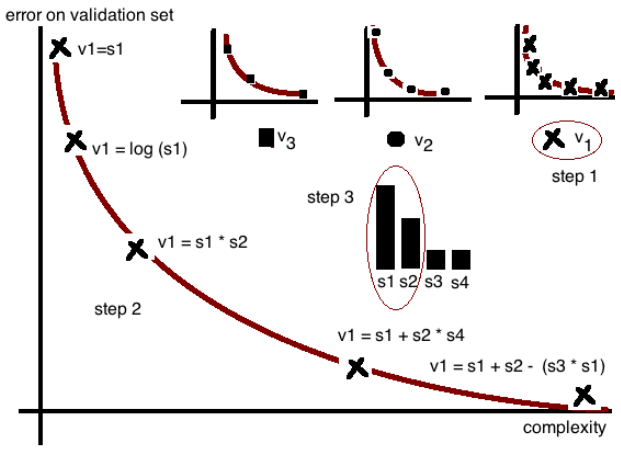


Fig. 4: Selection of the best variable in HSR version 2 (line 12 of algorithm 3). Selection of the best non-dominated set (step 1), building the predictor-frequency histogram (step 2), selecting the most frequently occurring predictors (step 3).

V. Experimental Results

In this section, we first discuss the conceptual differences between the naive way of modeling hierarchical relationships in multivariate data versus actively enforcing hierarchical modeling on an example dataset. Then, we outline our experimental procedures in generating a large benchmark synthetic data suite with varying levels of difficulty and comparing the three algorithms. As it is stated in [16], challenging benchmark datasets are needed for genetic programming research. Ideally, applying algorithms to real-world data is preferable. However, especially in the case of testing new algorithms, we believe that it is very important to have control over the data generation process so that analysis of the strengths and weaknesses of the algorithms can be easier. Finally, we present and discuss our findings on these synthetic benchmark datasets.

A. An example 16-Input Binary Tree System

A simple 16-input synthetic system has been generated using the following expressions representing the relationships between the variables:

$$
\begin{aligned}
&layer1:\\
&\quad v_1 = s_1 + s_2\\
&\quad v_2 = s_3 - s_4\\
&\quad v_3 = s_5 + s_6\\
&\quad v_4 = s_7 + s_8\\
&\quad v_5 = s_9 + s_{10}\\
&\quad v_6 = s_{11} + s_{12}\\
&\quad v_7 = s_{13} + s_{14}\\
&\quad v_8 = s_{15} + s_{16}\\
&layer2:\\
&\quad v_9 = v_1/v_2\\
&\quad v_{10} = v_3 * v_4\\
&\quad v_{v11} = v_5 - v_6\\
&\quad v_{v12} = v_7 + v_8\\
&layer3:\\
&\quad v_{13} = v_9/v_{10}\\
&\quad v_{v14} = v_{11} - v_{12}\\
&layer4:\\
&\quad v_{15} = v_{14} + v_{13}
\end{aligned}
$$

The resulting dependency graph is shown in Fig. 5. The binary tree (arity=2) consists of 4 layers, 16 stimuli and 15 internal nodes which are the variables to be modeled. The total number of edges in the tree is $arity * internal\ nodes = 30$.

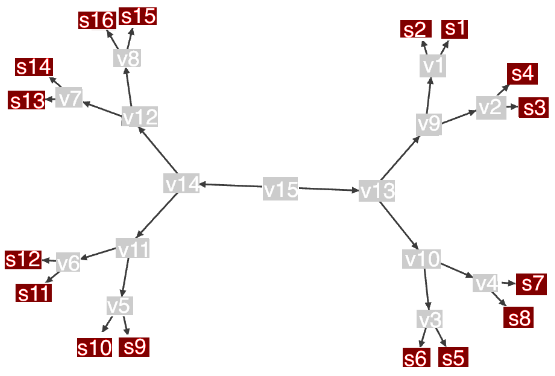


Fig. 5: True dependency graph for the synthetic 16-input binary tree system

Fig. 6 shows the best dependency graph generated by the naive SR algorithm in terms of the error on the test dataset. The test error is calculated as the average error across all modeled variables. Despite the low error, the constructed dependency graph is very dissimilar to the original graph shown in Fig. 5. A closer look at the constructed graph and the generated models for the variables reveals many redundant and cyclic dependencies as well as a number of ignored stimuli. This example shows that even though the given dataset is perfectly hierarchical, the NSR algorithm might fail to capture this hierarchy. Therefore, solely minimizing the prediction error without any constraints on the connectivity might be deceptive for the purposes of modeling hierarchical systems. On the other hand, when multiple runs are performed, it was possible for the algorithm to find the correct dependency graph structure in some of the runs along with the lowest possible test dataset error. However, for this algorithm, a low test error does not always mean that the hierarchy in the underlying system is captured.

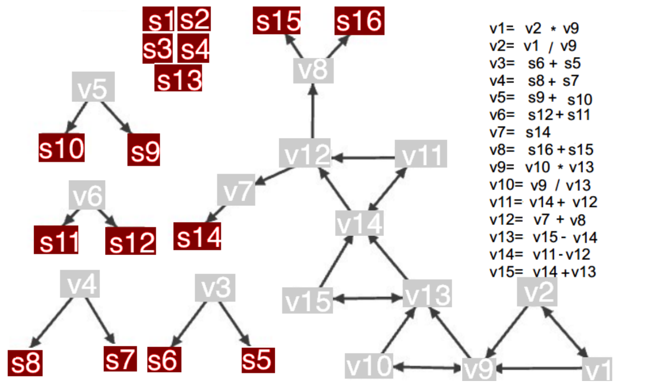


Fig. 6: Dependency graph with lowest test set error (rmse: 0.025) generated by NSR

Fig. 7 shows the dependency graph with the highest test set error generated using the HSR version 1 (h1) algorithm. In terms of the prediction accuracy, this system is worse than the one discovered using the naive approach (Fig. 6). However, the constructed graph almost perfectly captures the true system. The high prediction error was caused by just one misplaced edge (s_{15} is erroneously tied to v_6 instead of v_8). Therefore, for h1 and h2 algorithms, a high test set error does not always mean failure in terms of how close the hierarchy is captured.

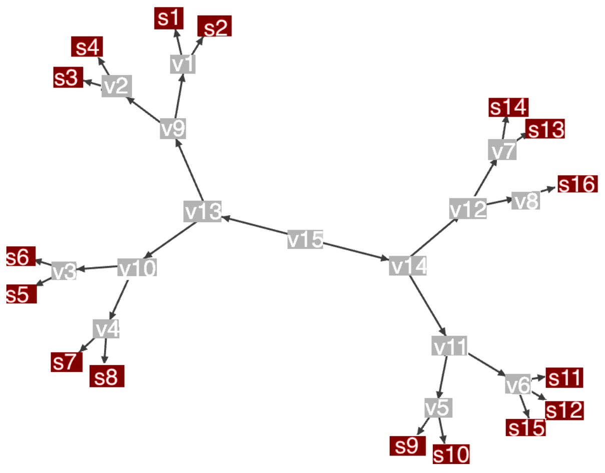


Fig. 7: Dependency graph with highest test set error (rmse: 1.662) generated by HSR version 1

B. Synthetic Benchmark Problems

In order to study the behavior of the algorithms across multiple datasets with varying difficulty, we generated 30 synthetic datasets for each combination of arity=2,3,4,5 and layers=3,4,5,6. A total of 480 datasets were generated with random stimuli as inputs to the systems and randomly generated expressions to represent the functions of the state variables. For the sake of simplicity, the expressions included only {+,-,* and protected / } operators without any constant values. We also kept the degree of nonlinearity constant across all datasets by enforcing that only binary nonlinear interactions were allowed in the randomly generated expressions for the state variables. For instance, for an arity-5 system, the hidden expression for a state variable can be $v_1 = s_1 + s_2 - s_4 * s_5 - s_3$, but not $v_1 = s_1 * s_2 / s_4 * s_5 - s_3$. By design, the difficulty of the dataset increases as the data dimensionality increases (between 4 - 3125 inputs).

TABLE I: Number of inputs, variables to be modeled and number of edges for the generated synthetic benchmark trees

		Layers			
		3	4	5	6
Arity	2	4, 3, 6	8, 7, 14	16, 15, 30	32, 31 ,62
	3	9, 4, 12	27, 13, 39	81, 40, 120	243, 121, 363
	4	16, 5, 20	64, 21, 84	256, 85, 340	1024, 341, 1364
	5	25, 6, 30	125, 31, 155	625, 156,789	3125, 781, 3905

Each dataset was divided into training, validation and test partitions as follows: for each n-arity tree system, all expressions for the state variables (internal nodes of the tree) were evaluated for 5000 randomly generated stimuli creating a 5000x($(arity^{layers} - 1)/(arity - 1)$) dataset. For every 4 rows, the first two rows were included in the training set, while the third and fourth rows are included in the validation and test sets respectively. The training,validation and testing partitions consisted of 2500, 1250 and 1250 rows each.

C. Results on Benchmark Problems

We ran each algorithm 30 times on all 480 datasets for a run time budget of 10 minutes per run for a total of 43200 runs on a cluster computing environment. All algorithms were implemented in C++. The baseline symbolic regression implementation that is used by all three algorithms utilizes the standard tree based representation with sub-tree crossover and mutation operators. Fixed values for population size (500), crossover probability (0.9) and mutation probability (0.1) were used across all experiments. Root mean squared error (rmse) versus expression complexity trade-off along with the age of the individuals were used as the multi-objective fitness function. Similar to the AFPO algorithm [17], a new random individual is added to the population at the beginning of each generation. For n state variables to be modeled, the population included n sub-populations, each of which were set up to evolve expressions for one state variable.

We report the results for each tree structure separately in Fig. 8. For each arity-layer pair, the results are pooled over 30 different trees and 30 runs for each tree resulting in a total of 900 runs per algorithm. In each case, the results are presented from three perspectives: the percentage of the edges that were correctly discovered, the prediction error of the generated tree on the test set, the distribution of test set error versus the percentage of the edges correctly discovered.

The statistical significance of the results are reported as follows: for the percentage of correct edges, we compare each pair of algorithms using the left-tailed Wilcoxon rank sum test with Bonferroni correction and unequal variances assumption. In those cases where the h1 and h2 algorithms are significantly better than the naive algorithm, and when the h2 algorithm is significantly better than the h1 algorithm, the significance is presented using $*$ sign ($* * *$:$\alpha = 0.001$, $**$:$\alpha = 0.01$, $*$:$\alpha = 0.05$). A similar comparison is performed on the test error results, the only difference being the Wilcoxon rank sum test to be a right-tailed test since lower test error indicates better performance in this case.

The top row shows the results for 2-arity tree systems with varying numbers of layers. In terms of capturing the hierarchy, h1 and h2 almost always discover the correct connectivity matrix and consistently outperform the naive algorithm. As far as the test error is concerned, h1 and h2 clearly outperform the naive algorithm only when the ratio of the total nodes to the stimuli gets too large for the naive algorithm to deal with. This happens when the height of the tree increases to 5. The heatmaps show that the naive algorithm mostly finds low-error models at the expense of missing many edges.

As the arity of the trees increase, the trees get broader, as a result, the number of leaves (stimuli) increases. Since this will increase the number of possible predictors, the search becomes more and more difficult for all three algorithms. Accordingly, the h1 and h2 algorithms start to lose their advantage in faithfully modeling the hierarchy as the arity increases. Another source of difficulty for all algorithms is the increase in tree height. As the trees get taller, the number of leaves (stimuli) also increases. Except for 2-arity systems (top row in Fig. 8), h1 and h2 failed to complete within the assigned run time budget in the case of the tallest trees.

D. Discussion

Our results on the synthetic benchmark datasets clearly show that for binary input problems (arity 2), the hierarchical SR algorithms consistently outperform the naive SR as the problem difficulty increases (more tree layers). However, the lack of scaling to much higher arity problems is due to a number of reasons. High data dimensionality is a big challenge for symbolic regression. In additional experiments (not shown here), increasing the time budget from 10 minutes to 1 hour did not significantly increase the performance of h1 and h2. Therefore, efficient feature selection schemes for high dimensional data within symbolic regression is definitely an area for improvement.

Specifically, the results indicate that as the trees get broader and taller, the initial h1 and h2 algorithms do not, in their current form, continue to outperform the naive approach. This is due to the fact that in these cases, the number of epochs significantly increases, which will in turn decrease the

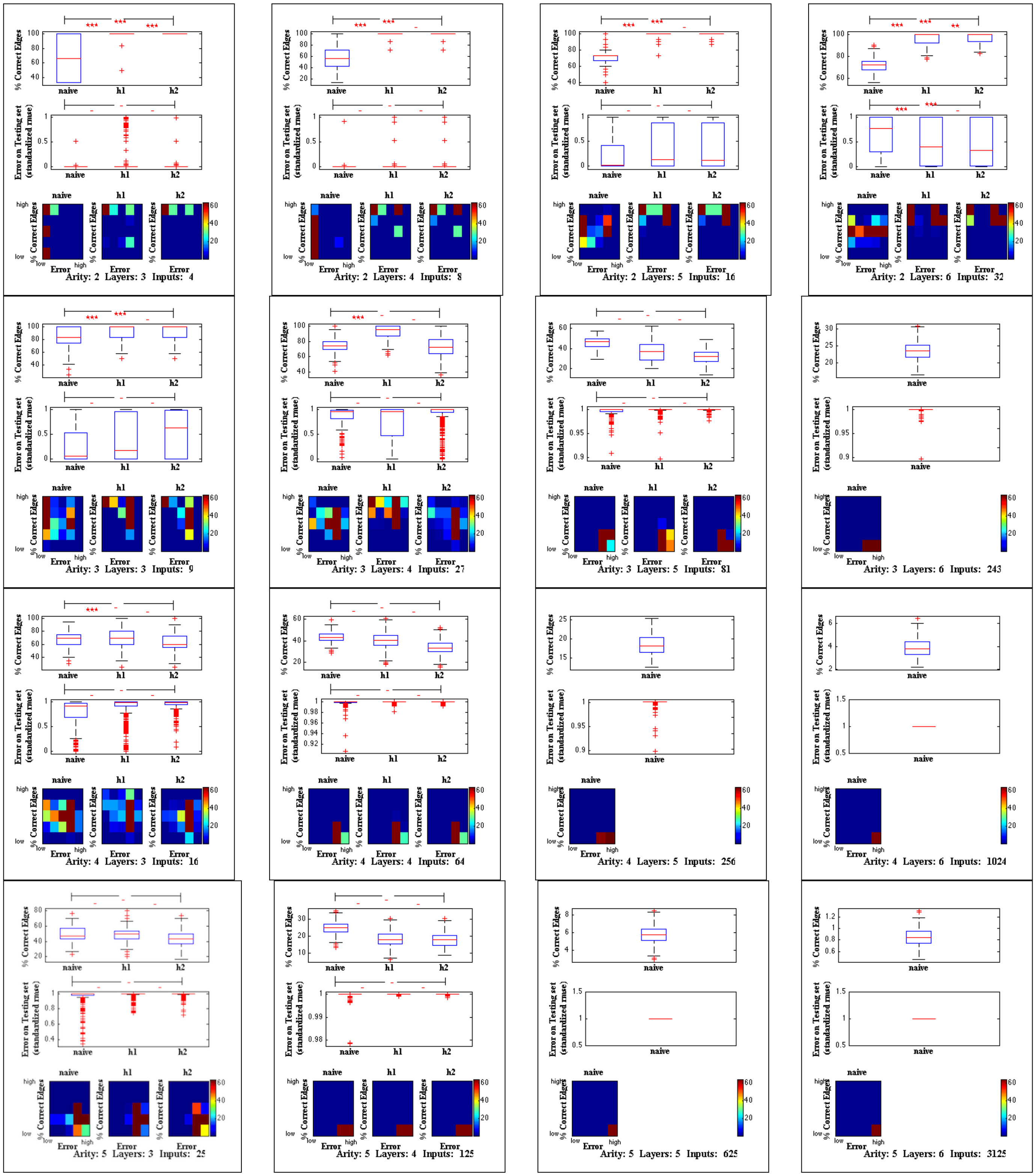


Fig. 8: Comparison of the algorithms across varying data dimensionality and hierarchical organization given 10 minutes run-time budget. Problem difficulty increases from left-right and top-bottom. For each problem type, the plots show the percentage of the edges that were correctly discovered (top), the prediction error of the generated tree on the test set (middle), the distribution of test set error versus the percentage of the edges correctly discovered (bottom). The naive approach mostly fails to recover the hierarchical network topology. The hierarchical approach outperforms the naive approach for easier problems (small arity and/or short trees). All three algorithms perform poorly for more difficult problems (larger arity and taller trees).

amount of time for each individual epoch. This increases the risk that h1 or h2 will select the wrong predictors. Indeed h2 was initially devised to reduce the risk of selecting the wrong dependent variables during an epoch. It was thought that providing h2 with statistics about how the current set of dependent variables are used across the Pareto front, rather than just how those variables are used in the model with lowest error, would improve its performance compared to h1. However this was not found to be the case: instead, h1 significantly outperformed the naive algorithm on seven of the tree structures (Fig. 8) while h2 only significantly outperformed the naive algorithm on five of them.

We hypothesize that h2 may be further improved by incorporating diversity maintaining measures that increase the size of the membership on the Pareto front. This should improve the reliability of the statistics computed across the front and thus improve the probability of selecting the correct set of dependent variables. We also plan to explore alternative methods for selecting the cutoff point in the predictor-frequency histogram. Additionally, in h2, predictor-frequency histograms constructed from previous epochs are discarded; in future work we will investigate ways to re-use information from previous histograms to produce more accurate histograms in the current epoch. Also, for both h1 and h2, selecting more than one accurately modeled variable in each epoch will speed up the algorithms by reducing the number of epochs.

Finally, we note that our initial implementations did not utilize any parallel and distributed techniques. Our algorithms have the potential to scale to real-world problems upon utilizing GPUs [18] and/or cloud computing [19].

VI. Conclusion

Extracting and visualizing the relationships in a hierarchical system as a dependency graph improves the intelligibility of the overall model, compared to the flat list of equations produced by traditional symbolic regression. Our results clearly show that in order to find hierarchy, one needs to explicitly search for it rather than waiting for the hierarchical models to emerge in an unconstrained search such as in the naive SR. Moreover, it was found that explicitly seeking hierarchy in a data set leads to more accurate models compared to traditional symbolic regression. These algorithms were tested against a large number of synthetic datasets with increasing difficulty in terms of data dimensionality and hierarchical organization. Even though the intuition suggests that the HSR version 2 (h2) algorithm would be more robust since it considers multiple models in making the selection for the best modeled variable, our experimental results on 480 synthetic datasets showed no clear advantage over the HSR version 1 (h1) which makes the selections based on one best model at each stage.

The focus of our current work is to further explore more efficient ways for the selection process at each stage and to extend the algorithm to model more general systems that exhibit mixtures of hierarchy and network connectivity. Ultimately, our goal is to apply our algorithms to real-world problems such as functional brain connectivity and gene expression networks.

Acknowledgment

This research was supported by a DARPA grant #FA8650-11-1-7155. The authors also acknowledge the Vermont Advanced Computing Core which is supported by NASA (NNX 06AC88G), at the University of Vermont for providing High Performance Computing resources that have contributed to the research results reported within this paper.

References

[1] R. J. Sommer, “Homology and the hierarchy of biological systems.” *Bioessays*, vol. 30, no. 7, pp. 653–8, 2008.

[2] H. Hirata and R. E. Ulanowicz, “Information theoretical analysis of the aggregation and hierarchical structure of ecological networks,” *Journal of Theoretical Biology*, vol. 116, no. 3, pp. 321 – 341, 1985.

[3] E. Ravasz and A.-L. Barabási, “Hierarchical organization in complex networks,” *Phys. Rev. E*, vol. 67, p. 026112, Feb 2003.

[4] S. Buldyrev, N. Dokholyan, S. Erramilli, M. Hong, J. Kim, G. Malescio, and H. Stanley, “Hierarchy in social organization,” *Physica A: Statistical Mechanics and its Applications*, vol. 330, no. 3-4, pp. 653 – 659, 2003.

[5] S. J. Kiebel, J. Daunizeau, and K. J. Friston, “Perception and hierarchical dynamics,” *Frontiers in Neuroinformatics*, vol. 3, no. 20, 2009.

[6] M. Kaiser, C. C. Hilgetag, and R. Kötter, “Hierarchy and dynamics of neural networks,” *Frontiers in Neuroinformatics*, vol. 4, no. 112, 2010.

[7] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations,” in *Proceedings of the 26th Annual International Conference on Machine Learning,ICML ’09*, 2009, pp. 609–616.

[8] T. R. Hancock, M. Golea, and M. Marchand, “Learning nonoverlapping perceptron networks from examples and membership queries,” *Mach. Learn.*, vol. 16, no. 3, pp. 161–183, Sep. 1994.

[9] M. Schmitt, “On the sample complexity for nonoverlapping neural networks,” *Mach. Learn.*, vol. 37, no. 2, pp. 131–141, Nov. 1999.

[10] E. Sakamoto and H. Iba, “Inferring a system of differential equations for a gene regulatory network by using genetic programming,” in *Proc. Congress on Evolutionary Computation*. IEEE Press, 2001, pp. 720–726.

[11] J. Bongard and H. Lipson, “Automated reverse engineering of nonlinear dynamical systems,” *PNAS, Proceedings of the National Academy of Sciences of the United States of America*, vol. 104, no. 24, pp. 9943–9948, 12 Jun. 2007.

[12] B. A. McKinney, J. E. Crowe, H. U. Voss, P. S. Crooke, N. Barney, and J. H. Moore, “Hybrid grammar-based approach to nonlinear dynamical system identification from biological time series,” *Phys. Rev. E*, vol. 73, p. 021912, Feb 2006.

[13] A. G. Floares, “A reverse engineering algorithm for neural networks, applied to the subthalamopallidal network of basal ganglia,” *Neural Networks*, vol. 21, no. 2-3, pp. 379–386, Mar. 2008.

[14] J. Tikka and J. Hollmén., “Learning linear dependency trees from multivariate time-series data,” *In Proceedings of the Workshop on Temporal Data Mining: Algorithms, Theory and Applications (in conjunction with The Fourth IEEE International Conference on Data Mining),Brighton, UK*, November 2004.

[15] G. Kronberger, S. Fink, M. Kommenda, and M. Affenzeller, “Macroeconomic time series modeling and interaction networks,” in *Proceedings of the 2011 international conference on Applications of evolutionary computation - Volume Part II, EvoApplications’11*, 2011, pp. 101–110.

[16] J. McDermott, D. R. White, S. Luke, L. Manzoni, M. Castelli, L. Vanneschi, W. Jaskowski, K. Krawiec, R. Harper, K. De Jong, and U.-M. O’Reilly, “Genetic programming needs better benchmarks,” in *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference, GECCO ’12*, 2012, pp. 791–798.

[17] M. D. Schmidt and H. Lipson, “Age-fitness pareto optimization,” in *Proceedings of the 12th annual conference on Genetic and evolutionary computation,GECCO ’10*, 2010, pp. 543–544.

[18] “Genetic programming on general purpose graphics processing units,” http://www.gpgpgpu.com/.

[19] D. Sherry, K. Veeramachaneni, J. McDermott, and U.-M. O’Reilly, “Flex-gp: genetic programming on the cloud,” in *Proceedings of the 2012 European conference on Applications of Evolutionary Computation, EvoApplications’12*, 2012, pp. 477–486.

5.4 Szubert *et al.* “Reducing antagonism between...” (2016).

A technical manuscript describing how to reduce antagonism between model diversity and accuracy follows.

Reducing Antagonism between Behavioral Diversity and Fitness in Semantic Genetic Programming

Marcin Szubert
Dept. of Computer Science
University of Vermont
mszubert@uvm.edu

Anuradha Kodali
UC Santa Cruz / NASA Ames
Research Center

Sangram Ganguly
BAERI / NASA Ames
Research Center

Kamalika Das
UC Santa Cruz / NASA Ames
Research Center

Josh C. Bongard
Dept. of Computer Science
University of Vermont

ABSTRACT

Maintaining population diversity has long been considered fundamental to the effectiveness of evolutionary algorithms. Recently, with the advent of novelty search, there has been an increasing interest in sustaining behavioral diversity by using both fitness and behavioral novelty as separate search objectives. However, since the novelty objective explicitly rewards diverging from other individuals, it can antagonize the original fitness objective that rewards convergence toward the solution(s). As a result, fostering behavioral diversity may prevent proper exploitation of the most interesting regions of the behavioral space, and thus adversely affect the overall search performance. In this paper, we argue that an antagonism between behavioral diversity and fitness can indeed exist in semantic genetic programming applied to symbolic regression. Minimizing error draws individuals toward the target semantics but promoting novelty, defined as a distance in the semantic space, scatters them away from it. We introduce a less conflicting novelty metric, defined as an angular distance between two program semantics with respect to the target semantics. The experimental results show that this metric, in contrast to the other considered diversity promoting objectives, allows to consistently improve the performance of genetic programming regardless of whether it employs a syntactic or a semantic search operator.

Keywords

genetic programming; program semantics; novelty search; diversity; geometric crossover; symbolic regression

1. INTRODUCTION

In analogy to the importance of genetic diversity in natural evolution, preserving population diversity has long been perceived as being crucial to the performance of evolutionary algorithms. Intuitively, maintaining a diverse pool of candidate solutions provides better exploration of the search space and thus gives more opportunities to discover novel, potentially fitter individuals. On the other hand, losing diversity can lead to the well-known problem of *premature convergence*, where a population stagnates at local optima and is unlikely to make any further progress.

A number of diversity maintenance techniques have been proposed to mitigate the problem of premature convergence [10, 26]. Most of these methods modify the selection process by promoting the individuals that are most different from the rest of the population. One particular approach relies on multiobjective evaluation of individuals with two objectives: the original fitness of the solution and some measure of its *novelty* designed to promote diversity. Although earlier studies measured novelty by comparing genotypes [6], recent work has successfully employed novelty metrics based on the distance between behaviors [16, 17, 23].

However, since behavioral novelty promotes increasing distance between behaviors while the fitness function typically rewards minimizing distance to the target behavior, we hypothesize that in some cases these two objectives can be overly antagonistic with each other. Consequently, promoting diversity can result in spreading individuals over the behavioral space and slowing down the convergence of the search process. In other words, under certain conditions, employing such conflicting objectives may result in excessive exploration of the entire behavioral space and insufficient exploitation of its most promising regions.

In this paper, we investigate the relationship between behavioral diversity and fitness of evolved individuals in the context of genetic programming (GP), where behavior of an individual can be identified with program semantics. In particular, we attempt to determine whether and under what conditions promoting behavioral diversity can adversely affect the search effectiveness. To this end, we consider four diversity promoting objectives and examine how each of them, used along with the fitness objective, affects the performance of tree-based GP. Moreover, we compare the fitness of programs evolved with two types of search operators: traditional subtree-swapping crossover and locally geometric semantic crossover. Since fitness landscapes induced by the latter are supposedly smoother and easier to search with the fitness objective alone, we expect to observe different effects of promoting diversity.

GECCO '16, July 20-24, 2016, Denver, CO, USA
© 2016 ACM. ISBN 978-1-4503-4206-3/16/07…$15.00
DOI: http://dx.doi.org/10.1145/2908812.2908939

The results obtained on a set of symbolic regression problems demonstrate that some diversity objectives can be indeed detrimental to the search performance, supposedly because of being overly antagonistic with the fitness objective. In particular, we show that using straightforward Euclidean semantic novelty metric can lead to reduced performance with respect to the conventional genetic programming. By contrast, the introduced angular semantic novelty metric, designed to be less antagonistic with the fitness objective, allows to consistently improve both fitness and generalization performance, regardless of the employed search operator.

The remainder of this paper is structured as follows. The next section describes the paradigm of semantic genetic programming and presents geometric semantic operators. Section 3 gives a brief overview of diversity maintenance methods applied in GP and introduces the two aforementioned semantic novelty metrics. Sections 4 and 5 describe experimental setup and present the results. Finally, sections 6 and 7 provide discussion and concluding remarks.

2. SEMANTIC GENETIC PROGRAMMING

Standard tree-based GP searches the space of programs using traditional operators of subtree-swapping crossover and subtree-replacing mutation. These operators are designed to be generic and produce syntactically correct offspring regardless of the problem domain. However, their actual effects on the behavior of the program, and thus its fitness, are generally hard to predict. Because of the complex genotype-phenotype mapping characterized by low locality, even a minimal change at the syntax level may diametrically alter program semantics. Such large phenotypic changes are often considered problematic because, according to Fisher's geometric model [7], the probability of the mutation being beneficial is inversely proportional to its magnitude.

Recently, many alternative search operators have been proposed that take into account the effect of syntactic modifications on program semantics [1, 3, 15, 22, 30]. In order to control the scope of behavioral change, most of these methods adopt common definition of program semantics, known as *sampling semantics* [30], which is identified with the vector of outputs produced by a program for a sample of possible inputs. For instance, in supervised learning, where n input-output pairs are given as a training set $T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, semantics of a program p is equal to vector $\mathbf{s}(p) = [p(\mathbf{x}_1), \ldots, p(\mathbf{x}_n)]$, where $p(\mathbf{x})$ is a result obtained by running program p on input $\mathbf{x}$. Consequently, each program p corresponds to a point in n-dimensional semantic space and a metric d can be adopted to measure semantic distance between two programs. Furthermore, fitness of a program p can be calculated as a distance between its semantics $\mathbf{s}(p)$ and the target semantics $\mathbf{t} = [y_1, \ldots, y_n]$ defined by the training set, i.e., $f(p) = d(\mathbf{s}(p), \mathbf{t})$.

Importantly, the information about program semantics can be exploited not only at the level of search operators but also for other purposes, e.g, to maintain semantic diversity [11], to initialize the population [2] or to drive the selection process [18]. All such semantic-aware methods are collectively captured by the umbrella term of semantic genetic programming [31]. Recently, a paradigm of behavioral program synthesis [13] has been proposed, which extends semantic GP by using information not only about final program results but also about behavioral characteristics of program execution.

2.1 Geometric Semantic Operators

One particularly interesting class of semantic-aware search operators are geometric semantic operators introduced by Moraglio *et al.* [22]. These operators not only incorporate knowledge about program semantics but also exploit geometric structure of the semantic space endowed by a metric-based fitness function. As a result, fitness landscapes seen by these operators are smooth conic landscapes, which are in principle easy to search. In particular, a geometric semantic crossover under the metric d guarantees that semantics of each offspring p' is located in the d-metric segment connecting semantics of its parents p_1 and p_2, i.e., $d(\mathbf{s}(p_1), \mathbf{s}(p_2)) = d(\mathbf{s}(p_1), \mathbf{s}(p')) + d(\mathbf{s}(p'), \mathbf{s}(p_2))$

Although *exact* geometric crossover has been proposed [22], its practical applicability is limited because it leads to exponential growth of the program size. For this reason, alternative operators exist that employ heuristic methods to produce an *approximately* geometric offspring [14, 15]. Previous studies demonstrate that such approximately geometric operators can be still effective while producing much shorter offspring programs than exactly geometric ones.

2.2 Locally Geometric Crossover

In this paper we use Locally Geometric Crossover (LGX) proposed by Krawiec and Pawlak [15]. This operator is arguably the easiest to implement among existing approximately geometric crossover operators. Before applying a crossover, a library of short programs (procedures) must be created. Typically, a static library is generated by enumerating all possible trees lower than a predefined height. Alternatively, a dynamic library could be created at each generation from all subtrees existing in the population.

Given two parents p_1 and p_2, the operator starts by identifying their structurally common region, i.e., the largest region where the parent trees have the same topology. Two crossover points are selected by drawing a pair of corresponding nodes from the common region. Then, for the subtrees p'_1 and p'_2 rooted at the crossover points, semantics of the midpoint between them (i.e., semantically intermediate subprogram) is calculated as $\mathbf{s}_m = (\mathbf{s}(p'_1) + \mathbf{s}(p'_2)) / 2$. The library is searched for programs that are semantically closest to $\mathbf{s}_m$ according to adopted metric d. From a set of k closest programs found in a library, a random one is selected and used to replace subtrees p'_1 and p'_2 in both parents, producing two offspring. In a rare situation when both subtrees p'_1 and p'_2 are semantically equivalent, a random procedure is drawn from a library.

3. PROMOTING DIVERSITY IN GP

Diversity maintenance has been a long-standing issue in GP and a number of methods have been proposed to preserve diversity in a population [4]. Most of the early studies in this area focus on genotypic diversity, which refers to structural differences between programs in a population [6, 21]. In recent years, with the advent of semantic GP, more attention has been paid to semantic or behavioral diversity [2, 9, 11, 19]. The notion of semantic diversity is particularly important in GP, because the mapping between programs and their semantics is usually a complex, non-injective function. In particular, since many syntactically different programs may exhibit the same behavior, genotypic diversity does not necessarily imply semantic diversity while the converse is often true.

Despite the assumed importance of semantic diversity [31], there have been few empirical investigations into effects of promoting behavioral diversity on the effectiveness of genetic programming. Moreover, almost all of the studied methods are limited to ensuring that the genetic operators do not produce offspring that is semantically equivalent to their parents [1, 30, 11, 9]. To the best of our knowledge, the only exception is the work of Nguyen et al. [24]. The authors apply both syntactic and semantic distance metrics in the fitness sharing mechanism and demonstrate that only using the latter improves GP performance on selected symbolic regression problems.

Here, rather than fitness sharing we adopt multiobjective approach treating diversity as a separate objective. In the following we describe four considered variants of multiobjective GP, which differ only with respect to the objective used to encourage diversity. In particular, two of the objectives (age and structural density) have already proved successful in improving GP performance. Additionally, we propose two other objectives which are essentially behavioral novelty metrics designed to promote semantic diversity.

3.1 Age-Fitness Pareto Optimization

Age-Fitness Pareto Optimization (AFPO, [27]) is a multiobjective method that relies on the concept of *genotypic age* of an individual, defined as the number of generations its genetic material has been in the population [10]. The age attribute is intended to protect young individuals before being dominated by older already optimized solutions. Each randomly initialized individual starts with age of one which is then incremented by one every generation. An offspring inherits age of the older parent.

The AFPO algorithm is based on the ParetoGP method which was originally proposed to address the issue of bloat in GP [28]. The algorithm starts with a population of n randomly initialized individuals. In each generation, it proceeds by selecting random parents from the population and applying crossover and mutation operators (with certain probability) to produce $n-1$ offspring. The offspring, together with a single randomly initialized individual, are added to the population extending its size to $2n$. Then, Pareto tournament selection is iteratively applied by randomly selecting a subset of individuals and removing the dominated ones until the size of the population is reduced back to n. To determine which individuals are dominated, the algorithm identifies the Pareto front using two objectives (both minimized): age and fitness (distance to the target semantics).

3.2 Density-Fitness Pareto Optimization

Recently, Burks and Punch [5] proposed an alternative variant of the AFPO algorithm called Density-Fitness Pareto Optimization (DFPO). This method relies on the idea of a *genetic marker*, which refers to concatenated fragments of a program tree. The authors used markers based on the topmost part of a tree and calculated *structural density* of each individual as a fraction of individuals in the population that share the same marker. Employing such a density measure as a minimized objective is intended to maintain a specific form of structural diversity focused on the rooted portions of the trees. According to the reported results obtained on three different problems (including symbolic regression), using density instead of age allows DFPO to further improve the performance achieved by AFPO.

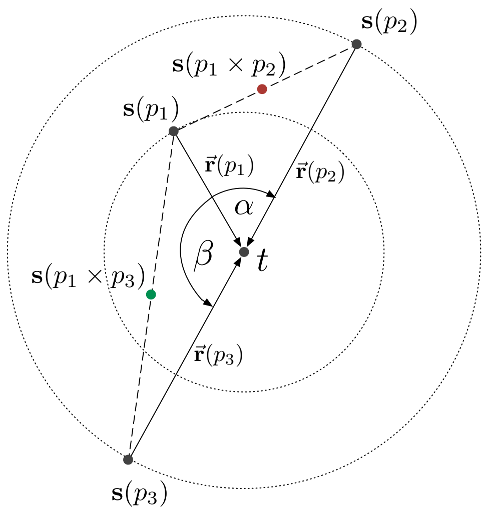


Figure 1: Residual vectors in two-dimensional semantic space where fitness is expressed using Euclidean distance.

3.3 Novelty-Fitness Pareto Optimization

Inspired by novelty search [16], we propose two behavioral novelty metrics that can be used as search objectives. Since both objectives refer to the distribution of programs in the semantic space, maximizing them is intended to promote some form of behavioral diversity. The bi-objective algorithm employing fitness and a behavioral novelty objective is termed Novelty-Fitness Pareto Optimization.

Euclidean Semantic Novelty. Since in this work we focus on real-valued symbolic regression problems, the semantic space is n-dimensional real space. Consequently, we can calculate behavioral distance between programs as a Euclidean distance between their semantics. We define Euclidean semantic novelty of a program p as a mean Euclidean distance between its semantics $\mathbf{s}(p)$ and semantics of its k nearest neighbors in the semantic space:

$$\rho(p) = \frac{1}{k}\sum_{i=1}^{k} d(\mathbf{s}(p), \mathbf{s}(\mu_i)),$$

where k is user-defined parameter and μ_i is i-th nearest program with respect to the semantic distance.

Angular Semantic Novelty. The second proposed novelty metric focuses on angles in the semantic space (see Fig. 1). Measuring angular distance between program semantics has been recently applied in GP [25] but not for the purpose of maintaining diversity. For each program p, we define *residual vector* $\mathbf{r}(p)$ as a difference between target semantics and the program semantics, i.e., $\mathbf{r}(p) = \mathbf{t} - \mathbf{s}(p)$. We define angular semantic novelty of a program p as a mean angle between its residual vector $\mathbf{r}(p)$ and residual vectors of its k nearest neighbors with respect to the angular distance:

$$\rho(p) = \frac{1}{k}\sum_{i=1}^{k} \arccos \frac{\mathbf{r}(p) \cdot \mathbf{r}(\mu_i)}{\|\mathbf{r}(p)\|\|\mathbf{r}(\mu_i)\|}.$$

Table 1: Symbolic regression benchmarks.

Problem	Objective function
Quartic	$4x^4 + 3x^3 + 2x^2 + x$
Nonic	$\sum_1^9 x^i$
R1	$(x+1)^3/(x^2 - x + 1)$
R2	$(x^5 - 3x^3 + 1)/(x^2 + 1)$
Keijzer-4	$x^3 e^{-x} \cos(x) \sin(x)(\sin^2(x)\cos(x) - 1)$

We expect that using this novelty metric as an additional search objective can be beneficial for two reasons. First, this objective is less conflicting with fitness than Euclidean semantic novelty — a population of very fit individuals can at the same time exhibit high angular semantic diversity. Second, promoting large angles between residual vectors makes it more likely that the parents occupy the opposite slopes of the fitness landscape, which is advantageous for geometric semantic crossover. For instance, consider three programs illustrated in Fig. 1. Let us assume that p_1 is the first parent and we need to pick the second parent among programs p_2 and p_3, which are equally fit (have the same distance to the target semantics). By considering possible offspring $p_1 \times p_3$ and $p_1 \times p_2$, it can be observed that fitness of the geometric offspring is inversely proportional to the angle between residual vectors of its parents.

4. EXPERIMENTAL SETUP

The main goal of the experiments is to investigate whether and how promoting particular forms of diversity affect the fitness of programs evolved with tree-based GP. For this purpose, we analyze the performance of multiobjective diversity promoting methods described in Section 3 and compare them to the standard GP driven by the fitness objective alone. All the considered algorithms were implemented as an extension[1] of the Distributed Evolutionary Algorithms in Python (DEAP) framework [8].

4.1 Symbolic Regression Problems

We consider five univariate symbolic regression problems that are adopted from previous studies [5, 20]. Selected benchmarks (see Table 1) include polynomial, rational and trigonometric functions. For each problem, fitness was calculated as Euclidean distance to the target semantics on 20 training cases distributed equidistantly in the $[-1, 1]$ interval. The only exception is Keijzer-4, for which the training cases were sampled from the range $[0, 10]$.

4.2 Genetic Programming Variants

We compare the performance of the following five variants of tree-based GP. Four of them rely on multiobjective fitness evaluation where one of the objectives actively promotes some form of diversity. These setups differ only with respect this objective. All the other settings remain unchanged and they are summarized in Table 2.

GP. To observe the relative impact of promoting diversity, as a baseline method we use standard generational tree-based GP with single-objective tournament selection.

[1]The source code necessary for reproducing our results is available at https://github.com/mszubert/gecco_2016.

Table 2: Genetic programming settings

Parameter	Value
population size	256
generations	1000
initialization	ramped half-and-half height range $2-6$
instruction set	$\{+, -, \times, /, \exp, \log, \sin, \cos\}$
tournament size	7
crossover probability	0.9
reproduction probability	0.1
mutation probability	0.0
node selection	90% internal nodes 10% leaves
maximum tree height	17
maximum tree size	300
number of runs	100

AFPO. Age-Fitness Pareto Optimization algorithm described in Section 3.1.

DFPO. Density-Fitness Pareto Optimization algorithm (see Section 3.2). To calculate the density objective, genetic markers were constructed using first three levels of each tree.

ESNFPO. Novelty-Fitness Pareto Optimization (see Section 3.3) with Euclidean semantic novelty objective using $k = 15$ nearest neighbors to calculate novelty score.

ASNFPO. Novelty-Fitness Pareto Optimization (see Section 3.3) with angular semantic novelty objective using $k = 15$ nearest neighbors to calculate novelty score.

4.3 Search Operators

To gain deeper understanding about usefulness of diversity under different conditions, we combine each of the considered GP variants with the following search operators.

Standard syntactic crossover. Traditional subtree-swapping crossover operator with Koza-style node selection: 0.9 probability of choosing an internal node [12].

Geometric semantic crossover. Locally geometric semantic crossover (LGX, see Section 2.2) based on a static precomputed library of procedures. The library is generated by enumerating all possible trees of height at most 3, built from the given instruction set. When queried with a desired semantics, library returns a random program among $k = 8$ with closest semantics.

4.4 Diversity Measures

To analyze the relationship between behavioral diversity of a population and fitness of evolved programs, the following diversity measures were calculated for each generation.

Median Euclidean Semantic Distance. To assess Euclidean semantic diversity we calculate median of semantic distances between each pair of programs in the population.

Mean Angular Semantic Distance. Angular semantic diversity is measured as a mean angle between residual vectors of each pair of programs in the population.

5. RESULTS

In order to conduct an accurate analysis of the relationship between diversity and performance, we conducted 100 independent runs (with different random seeds) of each of 10 considered configurations (5 GP variants $\times$ 2 crossover operators) on each of 5 symbolic regression problems.

5.1 Search Performance

Figure 2 shows the average best-of-generation fitness (calculated as a Euclidean distance to the target) achieved by particular methods on different benchmark problems, with 95% confidence intervals marked as semi-transparent bands. The left part of the figure illustrates the results obtained with traditional subtree-swapping crossover. Clearly, each of the considered diversity promoting methods significantly improves the performance of the standard GP algorithm. The best performance is achieved either by DFPO or ASNFPO, depending on the problem. The impact of promoting diversity on the fitness of evolved solutions is much less clear in the case of LGX crossover (right part of Figure 2). The only method that consistently improves the results of the baseline GP algorithm on all considered problems is ASNFPO. All the other diversity preserving approaches are detrimental to the search performance at least on some benchmarks.

Further observations can be made by comparing the results achieved by the same algorithm but equipped with different crossover operators. The largest performance improvement is observed for the standard GP algorithm, which when equipped with the LGX operator, achieves significantly higher convergence speed and final performance. As a matter of fact, it converges so quickly, that if the runs were stopped after 100 generations, it would be the best of the considered setups. Besides GP, the only other method that regularly benefits from replacing traditional syntactic crossover with geometric semantic crossover is ASNFPO. Importantly, the synergistic interplay of LGX crossover and the angular semantic novelty objective leads to the best overall results in terms of the ultimate achieved fitness.

5.2 Diversity Analysis

To analyze the relationship between behavioral diversity and fitness, we assessed diversity of populations evolved by particular methods using measures listed in Section 4.4. Table 3 shows Spearman correlation coefficients calculated between behavioral diversity measured at selected generations and best fitness in the last generation of each run.

In the context of the Euclidean semantic distance measure (left part of Table 3), correlation is stronger for semantic crossover than for standard syntactic crossover. More importantly, at the end of runs correlation is positive — large Euclidean semantic diversity is seen with high (bad) fitness values. This observation is consistent with relatively weak performance achieved by ESNFPO method which uses Euclidean semantic novelty objective to promote behavioral diversity. Taken together, these results suggest that high levels of Euclidean semantic diversity can not be considered as being generally beneficial to the search performance.

Moreover, it can be noticed that at the beginning of runs correlation coefficients are much lower (sometimes even negative) and only later start to increase. Therefore, behavioral diversity may play different role at different evolutionary times. Indeed, further analysis revealed that in the most successful runs, Euclidean semantic diversity stays relatively high in the early, exploratory phase of evolution but then gradually decreases which corresponds to exploitation of the most promising parts of the behavioral space. Thus, high diversity at the beginning of the evolution may be not only less harmful but even advantageous. On the other hand, keeping diversity high throughout entire runs typically leads to inferior performance.

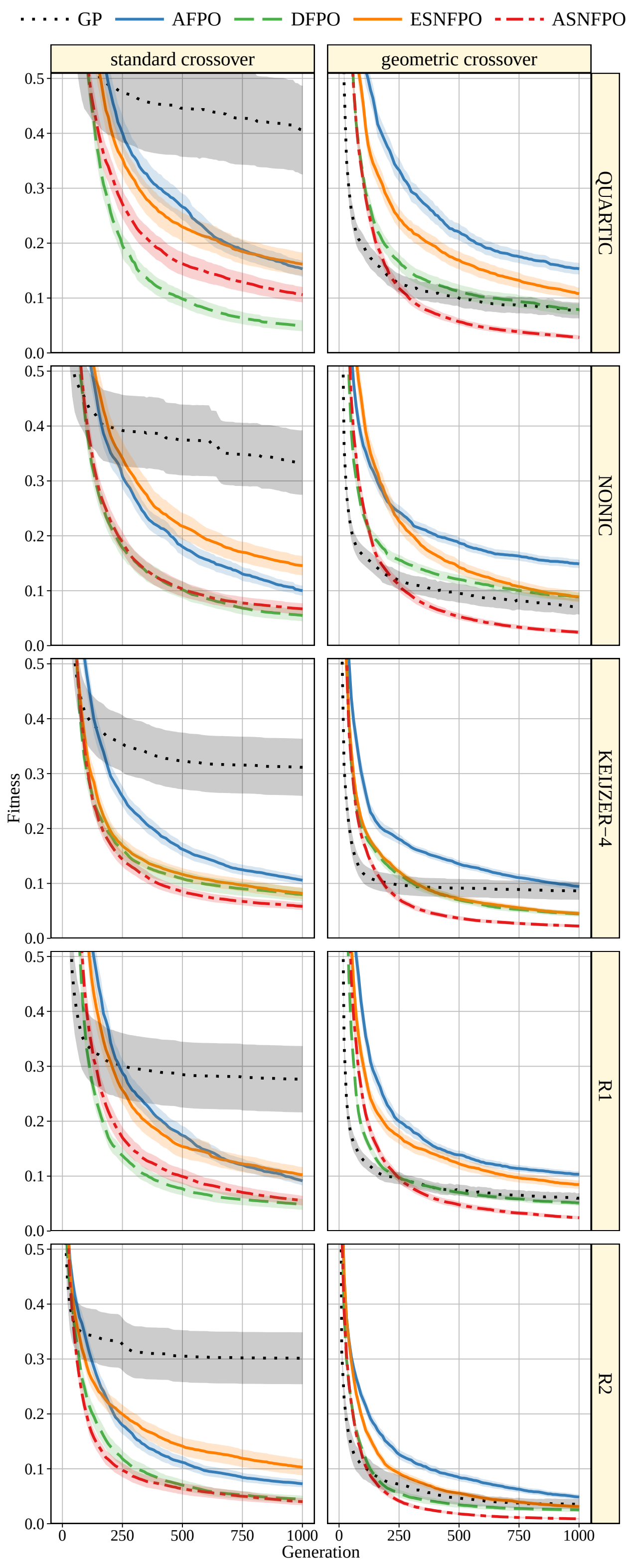


Figure 2: Average best fitness achieved by different variants of multiobjective GP equipped with either standard syntactic crossover (left column) or locally geometric semantic crossover (right column).

Table 3: Correlation between best fitness (lowest error) in the last generation and behavioral diversity measured at selected generations as: 1) median euclidean semantic distance 2) mean angular semantic distance.

	Median Euclidean Semantic Distance										**Mean Angular Semantic Distance**									
	Standard syntactic crossover					**Geometric semantic crossover**					**Standard syntactic crossover**					**Geometric semantic crossover**				
generation	QUA	NON	KEI	R1	R2	QUA	NON	KEI	R1	R2	QUA	NON	KEI	R1	R2	QUA	NON	KEI	R1	R2
0	−.017	−.034	−.003	−.003	+.031	+.002	+.044	−.012	−.066	+.003	+.010	−.002	+.042	−.010	+.041	−.031	+.061	−.015	−.101	+.001
10	+.010	−.003	−.283	−.227	−.353	−.119	−.203	+.004	−.127	−.131	+.153	+.173	−.427	−.009	−.220	−.464	−.533	−.115	−.499	−.421
25	−.072	−.082	−.250	−.222	−.201	−.109	−.272	+.036	+.026	−.196	−.263	−.187	−.375	−.300	−.547	−.585	−.642	−.324	−.610	−.570
50	+.021	+.071	−.261	−.037	−.078	−.065	−.222	−.227	+.232	−.300	−.392	−.358	−.477	−.448	−.609	−.651	−.675	−.607	−.680	−.624
100	+.018	+.109	−.242	+.015	−.002	+.119	+.027	−.233	+.334	−.080	−.457	−.450	−.600	−.515	−.649	−.654	−.647	−.610	−.677	−.599
250	+.087	+.204	−.121	+.134	+.134	+.393	+.381	+.000	+.502	+.214	−.519	−.441	−.639	−.533	−.650	−.630	−.560	−.599	−.607	−.578
500	+.123	+.265	−.031	+.204	+.180	+.489	+.567	+.177	+.558	+.257	−.587	−.463	−.663	−.478	−.635	−.533	−.456	−.513	−.506	−.530
1000	+.175	+.296	+.020	+.243	+.229	+.630	+.694	+.251	+.567	+.380	−.549	−.411	−.658	−.376	−.607	−.284	−.251	−.266	−.301	−.324

The second form of behavioral diversity we investigate is angular semantic diversity. The right part of Table 3 illustrates relatively strong negative correlation between this diversity measure and final fitness of evolved programs, regardless of the type of employed crossover operator. Since high levels of angular semantic diversity are frequently seen with low (good) fitness, we can hypothesize that this form of diversity facilitates genetic programming. Together with high performance of the ASNFPO method, these results provide empirical evidence that angular semantic diversity tends to be more useful than Euclidean semantic diversity.

5.3 Generalization Performance

In order to assess generalization performance of evolved programs, we calculated the root-mean-square error committed by the best-of-run individuals on 1 000 tests drawn uniformly from the same range as for the training set. Table 4 shows median training error, test error and size (number of nodes) of the individuals evolved by particular methods. To confirm statistically significant differences between the results obtained by the five compared GP variants, for each problem and crossover operator we conducted the Kruskal-Wallis test followed by a post-hoc analysis using pairwise Mann-Whitney tests (with sequential Bonferroni correction). We set the level of significance at $p \leq 0.05$. Table 4 shows with an underline the results that were found significantly better than those achieved by every other GP variant.

On most problems, the significantly lowest test error is obtained by either DFPO or ASNFPO. Interestingly, while DFPO achieves the highest generalization performance in the context of standard crossover, ASNFPO is the winner among methods paired with the LGX operator. These results suggest that there is a synergy between particular variation operators and diversity promoting methods. Traditional syntactic crossover is able to exploit structural diversity maintained by DFPO, whereas semantic crossover benefits from angular semantic diversity. Another important observation is that ASNFPO is the only method that achieves higher generalization performance than standard GP on all problems, regardless of the crossover operator.

Finally, by comparing training and test errors achieved on particular benchmarks, we can observe that AFPO and DFPO methods overfit less than the other methods. One reason explaining less severe overfitting is that these two methods tend to produce shorter programs than the other methods (especially when equipped with the LGX operator). In particular, AFPO usually produces the significantly smallest trees among the considered methods.

6. DISCUSSION

One of the most interesting findings from experiments is the discrepancy between results obtained with different crossover operators (see left vs. right part of Fig. 2 and upper vs. lower part of Table 4). With traditional crossover, all the considered diversity promoting methods improve the performance of standard GP. On the other hand, with geometric crossover, ASNFPO is the only algorithm that consistently outperforms standard GP on all five symbolic regression problems. These findings raise the following questions: Why is angular semantic novelty so effective in the context of geometric crossover? Why are other diversity objectives beneficial with one crossover operator while being detrimental with another? We attempt to answer these questions by referring to the notion of fitness-diversity antagonism.

For the purpose of this discussion, let us say that there is an antagonism between fitness and diversity in a given population if improving fitness of any single individual is impossible without reducing population diversity. Under this definition, angular semantic diversity is never antagonistic with fitness. Indeed, by moving program semantics straight in the direction of the target (along residual vectors), angular semantic diversity does not change while fitness of any solution can be arbitrarily improved. In contrast, Euclidean semantic diversity is at least sometimes antagonistic with fitness — there are populations which can not be optimized without reducing their diversity. Indeed, minimizing error pulls individuals toward the target semantics but maximizing Euclidean diversity scatters them away from it.

Intuitively, one could expect that antagonistic diversity objectives would be detrimental to the search performance. However, this may not be the case in deceptive fitness landscapes, where local fitness gradient is misleading. In such a situation, increasing semantic distance to the target (fitness) can in fact reduce the distance to the target measured in the search space seen by specific operators. We hypothesize that semantically-blind standard crossover induces relatively rugged and deceptive landscape. This hypothesis would to some extent explain why any type of diversity objective, regardless of its antagonism, improves the search performance in the context of this crossover operator.

On the other hand, according to Moraglio et al. [22], geometric semantic operators see cone landscapes which are easy to search by fitness objective alone as they are not deceptive at all. Even though we employ *approximately* geometric crossover, we expect that the corresponding fitness landscape is still much smoother than the one induced by traditional crossover. In such landscapes fitness-diversity

Table 4: Median training error, test error and size of best-of-run individuals. For each problem and crossover operator the best results are shown in bold. Underline indicates statistically significant superiority.

		QUARTIC			NONIC			KEIJZER-4			R1			R2		
		TRAIN	TEST	SIZE	TRAIN	TEST	SIZE	TRAIN	TEST	SIZE	TRAIN	TEST	SIZE	TRAIN	TEST	SIZE
crossover standard	GP	0.071	0.088	124	0.059	0.062	94	0.057	0.284	229	0.040	0.059	92	0.060	0.064	**75**
	AFPO	0.031	0.034	**72**	0.022	0.027	**84**	0.024	0.111	**121**	0.019	0.019	**69**	0.016	0.016	77
	DFPO	**0.008**	**0.009**	123	**0.009**	**0.013**	138	0.015	**0.089**	161	**0.009**	**0.009**	100	0.008	**0.008**	135
	ESNFPO	0.026	0.050	128	0.029	0.052	125	0.015	0.210	143	0.018	0.030	111	0.016	0.022	87
	ASNFPO	0.019	0.059	136	0.011	0.043	137	**0.012**	0.169	166	**0.009**	0.023	127	**0.007**	0.015	125
crossover geometric	GP	0.011	0.033	284	0.010	0.029	295	0.012	0.415	300	0.010	0.019	257	0.005	0.005	183
	AFPO	0.033	0.032	**78**	0.034	0.032	**63**	0.020	**0.198**	**144**	0.023	0.021	**66**	0.010	0.009	**61**
	DFPO	0.016	0.016	89	0.019	0.018	86	0.010	0.340	248	0.011	0.011	83	0.005	0.005	80
	ESNFPO	0.023	0.028	185	0.018	0.023	188	0.010	0.508	287	0.017	0.018	149	0.006	0.007	153
	ASNFPO	**0.005**	**0.008**	186	**0.005**	**0.011**	226	**0.005**	0.379	293	**0.005**	**0.007**	202	**0.002**	**0.002**	192

antagonism is much more likely to be detrimental. This would explain weak performance achieved by using antagonistic Euclidean semantic novelty objective. Since angular semantic novelty, by contrast, is the only diversity objective known to be non-antagonistic, it proves successful in the context of the geometric crossover operator.

Finally, let us discuss two other reasons that could explain aforementioned discrepancy in results. First, by analyzing how fitness of perfectly geometric offspring depends on the angular distance between its parents (cf. Fig. 1), we expect that geometric crossover operator is able to effectively exploit angular semantic diversity. Another synergistic combination involves structural diversity (promoted by the DFPO algorithm) and traditional syntactic crossover operator. Both combinations of diversity objective and search operator result in superior performance when compared to other considered methods. Second, the reason why diversity maintenance plays such an important (and beneficial) role in GP equipped with traditional crossover is that in our experiments we do not employ any mutation operator which could supply new genetic material and explicitly sustain genetic diversity in a population. In absence of mutation, we expect that standard GP with subtree-swapping crossover is particularly vulnerable to the problem of premature convergence. This problem is less severe with locally geometric crossover because it relies on a large library of procedures which provides the population with new subtrees acting as a simple diversity preserving mechanism.

7. CONCLUSIONS

In recent years, the issue of behavioral diversity and its impact on the performance of evolutionary algorithms has been studied in many different contexts [17, 23, 29]. To the best of our knowledge, this is the first study that investigates the role of behavioral diversity in genetic programming equipped with semantic search operators. The main goal of this work was to determine whether and under what conditions promoting behavioral diversity can adversely affect the performance of GP applied to symbolic regression.

The most important finding is that using an additional diversity promoting objective can be indeed detrimental to the search performance. However, such a situation was observed only when both of the following conditions were met. First, a specific search operator was employed, which supposedly induced a smooth, non-deceptive fitness landscape. Second, the behavioral diversity objective was inherently antagonistic with the fitness objective. On the other hand, by introducing a non-antagonistic angular semantic novelty objective, we were able to improve the results regardless of the employed search operator. Importantly, this objective was the only one that proved successful in the context of locally geometric crossover operator.

The major limitation of this study is that our experimental investigations were conducted using a small set of five univariate symbolic regression benchmarks. Although promoting angular semantic diversity proved useful in this context, further work is needed to verify whether these results could be extended to more complex real-world problems. In particular, it would be interesting to analyze how much dimensionality of both feature space and semantic space impacts the performance of particular diversity promoting methods. Another direction of future research would be to investigate the importance of behavioral diversity for other semantic search operators.

In a broader perspective, our investigation indicates that a diversity objective needs to be carefully chosen with respect to the problem at hand and employed search algorithm. As demonstrated by this study, using objectives that are antagonistic with fitness was detrimental to the performance of semantic GP. However, we expect that with increasing deceptiveness in the fitness landscape, the consequences of using antagonistic objectives become more difficult to predict. In particular, one could hypothesize that in highly deceptive fitness landscapes antagonistic diversity objectives are more likely to be beneficial. This would be consistent with previous studies demonstrating that in extremely deceptive cases a successful way to increase search effectiveness is to ignore fitness and use a novelty objective alone [16].

Acknowledgments

This work was supported by the National Aeronautics and Space Administration under grant number NNX15AH48G.

8. REFERENCES

[1] L. Beadle and C. G. Johnson. Semantically Driven Crossover in Genetic Programming. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2008*, pages 111–116. IEEE, 2008.

[2] L. Beadle and C. G. Johnson. Semantic Analysis of Program Initialisation in Genetic Programming. *Genetic Programming and Evolvable Machines*, 10(3):307–337, 2009.

[3] J. C. Bongard. A Probabilistic Functional Crossover Operator for Genetic Programming. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 925–932. ACM, 2010.

[4] E. K. Burke, S. M. Gustafson, and G. Kendall. Diversity in Genetic Programming: An Analysis of Measures and Correlation with Fitness. *IEEE Trans. Evolutionary Computation*, 8(1):47–62, 2004.

[5] A. R. Burks and W. F. Punch. An Efficient Structural Diversity Technique for Genetic Programming. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 991–998. ACM, 2015.

[6] E. D. de Jong, R. A. Watson, and J. B. Pollack. Reducing Bloat and Promoting Diversity using Multi-Objective Methods. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 11–18. Morgan Kaufmann, 2001.

[7] R. A. Fisher. *The Genetical Theory of Natural Selection*. Oxford University Press, Oxford, 1930.

[8] F.-A. Fortin, F.-M. De Rainville, M.-A. G. Gardner, M. Parizeau, and C. Gagné. DEAP: Evolutionary Algorithms Made Easy. *Journal of Machine Learning Research*, 13(1):2171–2175, 2012.

[9] E. Galvan-Lopez, B. Cody-Kenny, L. Trujillo, and A. Kattan. Using Semantics in the Selection Mechanism in Genetic Programming: A Simple Method for Promoting Semantic Diversity. In *2013 IEEE Congress on Evolutionary Computation (CEC)*, pages 2972–2979, 2013.

[10] G. S. Hornby. ALPS: The Age-layered Population Structure for Reducing the Problem of Premature Convergence. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 815–822. ACM, 2006.

[11] D. Jackson. Promoting Phenotypic Diversity in Genetic Programming. In *Parallel Problem Solving from Nature, PPSN XI*, volume 6239 of *Lecture Notes in Computer Science*, pages 472–481. Springer, 2010.

[12] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[13] K. Krawiec. *Behavioral Program Synthesis with Genetic Programming*, volume 618 of *Studies in Computational Intelligence*. Springer, 2016.

[14] K. Krawiec and P. Lichocki. Approximating Geometric Crossover in Semantic Space. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 987–994. ACM, 2009.

[15] K. Krawiec and T. Pawlak. Locally Geometric Semantic Crossover: A Study on the Roles of Semantics and Homology in Recombination Operators. *Genetic Programming and Evolvable Machines*, 14(1):31–63, 2013.

[16] J. Lehman and K. O. Stanley. Abandoning Objectives: Evolution through the Search for Novelty Alone. *Evolutionary Computation*, 19(2):189–223, 2011.

[17] J. Lehman, K. O. Stanley, and R. Miikkulainen. Effective Diversity Maintenance in Deceptive Domains. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '13, pages 215–222. ACM, 2013.

[18] P. Liskowski, K. Krawiec, T. Helmuth, and L. Spector. Comparison of Semantic-aware Selection Methods in Genetic Programming. In *Proceedings of th Genetic and Evolutionary Computation Conference*, pages 1301–1307, New York, NY, USA, 2015. ACM.

[19] Y. Martínez, E. Naredo, L. Trujillo, and E. G. López. Searching for Novel Regression Functions. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 16–23. IEEE, 2013.

[20] J. McDermott, D. R. White, S. Luke, L. Manzoni, M. Castelli, L. Vanneschi, W. Jaskowski, K. Krawiec, R. Harper, K. De Jong, and U.-M. O'Reilly. Genetic Programming Needs Better Benchmarks. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 791–798. ACM, 2012.

[21] N. F. McPhee and N. J. Hopper. Analysis of genetic diversity through population history. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1112–1120, 1999.

[22] A. Moraglio, K. Krawiec, and C. G. Johnson. Geometric Semantic Genetic Programming. In *Parallel Problem Solving from Nature - PPSN XII*, volume 7491 of *Lecture Notes in Computer Science*, pages 21–31. Springer Berlin Heidelberg, 2012.

[23] J. B. Mouret and S. Doncieux. Encouraging Behavioral Diversity in Evolutionary Robotics: An Empirical Study. *Evolutionary Computation*, 20(1):91–133, 2012.

[24] Q. U. Nguyen, X. H. Nguyen, M. O'Neill, and A. Agapitos. An Investigation of Fitness Sharing with Semantic and Syntactic Distance Metrics. In *Proceedings of the 15th European Conference on Genetic Programming*, EuroGP'12, pages 109–120, Berlin, Heidelberg, 2012. Springer-Verlag.

[25] S. Ruberto, L. Vanneschi, M. Castelli, and S. Silva. *Genetic Programming: 17th European Conference, EuroGP 2014*, chapter ESAGP – A Semantic GP Framework Based on Alignment in the Error Space, pages 150–161. Springer Berlin Heidelberg, 2014.

[26] B. Sareni and L. Krahenbuhl. Fitness Sharing and Niching Methods Revisited. *IEEE Transactions Evolutionary Computation*, 2(3):97–106, 1998.

[27] M. Schmidt and H. Lipson. Age-Fitness Pareto Optimization. In *Genetic Programming Theory and Practice VIII*, volume 8 of *Genetic and Evolutionary Computation*, pages 129–146. Springer, 2011.

[28] G. Smits and M. Kotanchek. Pareto-Front Exploitation in Symbolic Regression. In *Genetic Programming Theory and Practice II*, chapter 17, pages 283–299. Springer, Ann Arbor, 2004.

[29] M. Szubert, W. Jaśkowski, P. Liskowski, and K. Krawiec. The Role of Behavioral Diversity and Difficulty of Opponents in Coevolving Game-Playing Agents. In *18th European Conference on Applications of Evolutionary Computation*, pages 394–405, 2015.

[30] N. Q. Uy, N. X. Hoai, M. O'Neill, R. I. Mckay, and E. Galván-López. Semantically-based Crossover in Genetic Programming: Application to Real-valued Symbolic Regression. *Genetic Programming and Evolvable Machines*, 12(2):91–119, 2011.

[31] L. Vanneschi, M. Castelli, and S. Silva. A Survey of Semantic Methods in Genetic Programming. *Genetic Programming and Evolvable Machines*, 15(2):195–214, 2014.

5.5 Szubert *et al.* “Semantic forward propagation...” (2016).

A technical manuscript describing how to de-randomize model perturbations to improve optimization follows.

Semantic Forward Propagation for Symbolic Regression

Marcin Szubert[1], Anuradha Kodali[2,3], Sangram Ganguly[3,4], Kamalika Das[2,3], and Josh C. Bongard[1]

[1] University of Vermont, Burlington VT 05405, USA
Marcin.Szubert@uvm.edu
[2] University of California, Santa Cruz CA 95064, USA
[3] NASA Ames Research Center, Moffett Field CA 94035, USA
[4] Bay Area Environmental Research Institute, Petaluma CA 94952, USA

Abstract. In recent years, a number of methods have been proposed that attempt to improve the performance of genetic programming by exploiting information about program semantics. One of the most important developments in this area is *semantic backpropagation*. The key idea of this method is to decompose a program into two parts — a subprogram and a context — and calculate the *desired* semantics of the subprogram that would make the entire program correct, assuming that the context remains unchanged. In this paper we introduce Forward Propagation Mutation, a novel operator that relies on the opposite assumption — instead of preserving the context, it retains the subprogram and attempts to place it in the semantically right context. We empirically compare the performance of semantic backpropagation and forward propagation operators on a set of symbolic regression benchmarks. The experimental results demonstrate that semantic forward propagation produces smaller programs that achieve significantly higher generalization performance.

Keywords: genetic programming, program semantics, semantic backpropagation, problem decomposition, symbolic regression

1 Introduction

Standard tree-based genetic programming (GP) searches the space of programs using traditional operators of subtree-swapping crossover and subtree-replacing mutation [4]. These operators are designed to be generic and produce syntactically correct offspring regardless of the problem domain. However, their actual effects on the behavior of the program, and thus its fitness, are generally hard to predict. For this reason, many alternative search operators have been recently proposed that take into account the influence of syntactic modifications on program semantics [1,11,10,13].

Semantic backpropagation [12,15] is arguably one of the most powerful techniques employed by such semantic-aware GP operators. The two operators based on semantic backpropagation — Random Desired Operator (RDO) and Approximately Geometric Crossover (AGX) have proved to be successful on a number of symbolic regression and boolean program synthesis problems [11,12]. Both operators rely on semantic decomposition of an existing program into two parts — a subprogram and its context. Given a subprogram, both operators attempt to calculate its *desired semantics*, i.e., the values that it should return to make the entire program produce the desired output, assuming that the context remains unchanged. The desired semantics can be then used to find a replacement for the subprogram that improves the overall program behavior.

Despite their superior performance when compared to other GP search operators [15,12,11], backpropagation-based RDO and AGX face a few major challenges that can limit their practical applicability. First of all, they are much more computationally expensive than traditional syntactic operators. Indeed, in order to calculate desired semantics, the target program output needs to be *backpropagated* by traversing the tree and inverting the execution of particular instructions. The computational cost of this operation is similar to the cost of a single fitness evaluation (which is typically the most expensive component of GP). Moreover, using desired semantics to find a subprogram replacement usually requires even more computational effort. Finally, the results reported so far demonstrate that RDO and AGX tend to produce relatively large programs that are difficult to interpret and may suffer from overfitting.

In this paper, we introduce Forward Propagation Mutation (FPM), a novel semantic-aware operator that also relies on program decomposition but works in the opposite manner to semantic backpropagation. Instead of preserving the context and replacing the subprogram, forward propagation retains the subprogram and attempts to place it in the semantically right context. In contrast to semantic backpropagation, the FPM operator does not require an additional tree traversal and thus it incurs less computational overhead. Moreover, the experimental results obtained on a set of univariate and bivariate symbolic regression problems demonstrate that it achieves competitive performance in terms of the training error while producing much smaller programs that usually perform significantly better on the unseen test cases.

2 Semantic Genetic Programming

In order to incorporate semantic-awareness into genetic programming, most of the recently proposed methods adopt a common definition of program semantics, known as *sampling semantics* [13], which is identified with the vector of outputs produced by a program for a sample of possible inputs. In supervised learning problems considered here, where n input-output pairs are given as a training set $T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, semantics of a program p is equal to vector $\mathbf{s}(p) = [p(\mathbf{x}_1), \ldots, p(\mathbf{x}_n)]$, where $p(\mathbf{x})$ is a result obtained by running program p on input $\mathbf{x}$. Consequently, each program p corresponds to a point in n-dimensional semantic space and a metric d can be adopted to measure semantic distance between two programs. Furthermore, fitness of a program p can be calculated as a distance between its semantics $\mathbf{s}(p)$ and the target semantics $\mathbf{t} = [y_1, \ldots, y_n]$ defined by the training set, i.e., $f(p) = d(\mathbf{s}(p), \mathbf{t})$.

The information about program semantics and the structure of the semantic space endowed by a metric-based fitness function can be exploited in many ways to facilitate the search process carried out by GP. Apart from numerous semantic search operators [1,13,10,11], the knowledge about semantics can be used to maintain population diversity [3], to initialize the population [2] or to drive the selection process [7]. All such semantic-aware methods are collectively captured by the umbrella term of semantic genetic programming [14]. Recently, a paradigm of behavioral program synthesis [5] has been proposed, which extends semantic GP by using information not only about final program results but also about behavioral characteristics of program execution.

3 Semantic Backpropagation

One of the most important methods in semantic GP is semantic backpropagation [12]. The key concept behind this method is *program decomposition*: a program p is treated as a function (i.e., it is deterministic and has no side effects) that can be decomposed into two constituent functions (subprograms) p_1 and p_2 such that $p(\mathbf{x}) = p_2(p_1(\mathbf{x}), \mathbf{x})$. In particular, if a program is represented as a tree, such decomposition can be made at each node — the inner function p_1 is expressed by the subtree rooted at the given node, while the outer function p_2 corresponds to the rest of the tree (also termed *context* [9], see left part of Fig. 1).

Semantic backpropagation assumes that the desired program output $p^*(\mathbf{x})$ can be produced by retaining the outer function and replacing just the inner one by another subprogram p_s, i.e., $p^*(\mathbf{x}) = p_2(p_s(\mathbf{x}), \mathbf{x})$. Starting from the desired program output $p^*(\mathbf{x})$, the backpropagation algorithm heuristically inverts the program execution to calculate the desired semantics of the subprogram p_s, i.e., the values it should produce to make the entire program correct. This idea has been employed to design two operators, AGX and RDO, which differ with respect to what they use as the desired program output $p^*(\mathbf{x})$. In this study, we focus on RDO, a mutation operator that assumes that target semantics $\mathbf{t} = [y_1, \dots, y_n]$ is given *a priori* and thus values $p^*(\mathbf{x}_i) = y_i$ can be used as an input for the backpropagation algorithm.

An example of a mutation performed by RDO is illustrated in Fig. 1 and proceeds as follows. First, a random mutation node is selected in the parent program (denoted as a circle with a double border in Fig. 1). The subtree p_1 rooted at this node is removed from the tree and the backpropagation algorithm is applied to calculate the desired semantics of the replacement p_s that would make the offspring program return desired values. The algorithm starts from the root of the tree, where desired semantics is given by $\mathbf{t}$, and follows the path to the removed subtree. For each node it calculates the desired semantics of its child by invoking the Invert function (a detailed description of this function and the RDO operator in general can be found in [12]).

For instance, let us assume that a training set contains just two cases with inputs $\mathbf{x} = [1, 2]$ and desired outputs $\mathbf{t} = [0, 2]$. As shown in Fig. 1, in the first step the algorithm finds out that to produce desired semantics at the root,

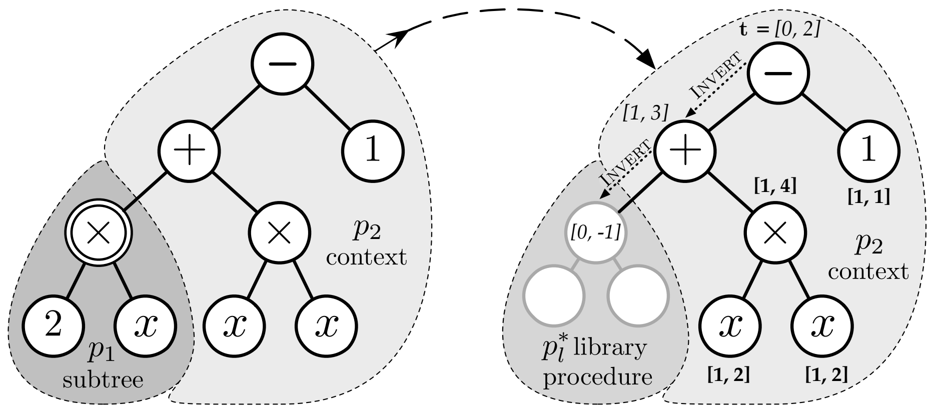


Fig. 1. A mutation performed by Random Desired Operator using semantic backpropagation. Desired semantics are denoted in italics.

knowing that outputs of its right child are equal to $[1, 1]$, the desired semantics of the left child must be equal to $[1, 3]$. This result is used in the subsequent step to calculate desired semantics for the next node. Finally, given desired semantics at the mutation node, the RDO operator attempts to replace the removed subtree with a subprogram that would produce such values. To this end, it employs a precomputed *library* of programs (procedures) that allows to efficiently retrieve a program p_l^* that has the smallest semantic distance to the desired semantics. Additionally, RDO also checks if a single constant real value would provide a better match to the desired semantics than p_l^*.

Importantly, in the process of semantic backpropagation, inverting certain functions can be ambiguous (if the function is not injective) or impossible (if the function is not surjective). As a result, the desired semantics may contain several values for each training case or special *inconsistent* elements. The library must be able to handle such queries efficiently [12,15].

4 Semantic Forward Propagation

Inspired by semantic backpropagation and RDO we propose an alternative mutation operator based on the complementary idea, which we term *semantic forward propagation*. Similarly to RDO, Forward Propagation Mutation (FPM) relies on decomposability of a program p into a subtree p_1 and a context p_2. However, while RDO assumes that a context can be preserved and attempts to replace the subtree, FPM makes the opposite assumption preserving the subtree and building a matching context for it.

The FPM operator starts by choosing a random mutation node in the parent program. The subtree p_1 rooted at this node is extracted from the tree and used as a starting point for creating an offspring. In order to build a new context for this subtree, we assume a fixed structure of the context p_c containing 4 new nodes

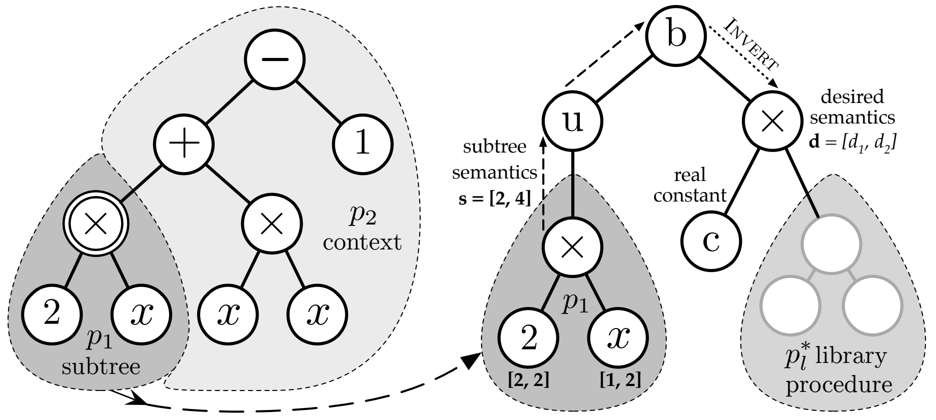


Fig. 2. An operation performed by Forward Propagation Mutation.

and a matching library procedure (see Fig. 2). We apply an exhaustive search to identify a context p_c^* of the assumed structure, that minimizes fitness of the entire offspring program $p_c^* = \arg\min_{p_c} f(p_c \circ p_1)$. To this end, we consider all pairwise combinations of the available unary (e.g., $\{\sin, \cos, \log, \exp\}$) and binary functions (e.g., $\{\times, +, -, /\}$) that could be placed directly above the selected subtree, as nodes u and b, respectively (cf. Fig. 2). Importantly, we extend the unary function set with the identity function $id(x) = x$. If the best found context p_c^* uses this function we skip adding the node u to the tree. For each pair of functions (u, b) placed above the subtree p_1, we *forward propagate* the semantics of the subtree up to the root of the new tree. Then, we apply just a single backpropagation step, using the same INVERT function as in RDO, to calculate desired semantics $\mathbf{d}$ of the other child of the node b, given the the target semantics $\mathbf{t}$ and the forward-propagated semantics $\mathbf{s}(u \circ p_1)$.

Since in this case the desired semantics is usually unambiguous, we can use a different method of searching the library, which could not be easily applied within the RDO operator. Here, we search for the library procedure which achieves highest cosine similarity. In other words, if we treat semantics as an n-dimensional vector, we return library procedure p_l^* that makes the smallest angle with the desired semantics $\mathbf{d}$, i.e.:

$$p_l^* = \underset{p_l \in L}{\arg\min}\ \arccos \frac{\mathbf{s}(p_l) \cdot \mathbf{d}}{\|\mathbf{s}(p_l)\| \|\mathbf{d}\|}.$$

Finally, we add a constant node c to scale the semantics of the library procedure making it closer to the desired semantics, i.e., $c = (\mathbf{s}(p_l^*) \cdot \mathbf{d}) \,/\, \|\mathbf{s}(p_l^*)\|^2$. An alternative, more computationally expensive approach, would be to run simple linear regression for each candidate program in the library, using its semantics as a single explanatory variable and desired semantics $\mathbf{d}$ as a response. This approach would require extending the context structure to accommodate both an intercept and a slope coefficient.

5 Experimental setup

The main goal of the experiments is to compare the performance of RDO and FPM mutation operators on a suite of symbolic regression benchmarks. Additionally, as a control setup we employ traditional subtree-replacing mutation (SRM). All three mutation operators are used along with conventional subtree-swapping crossover in a standard generational tree-based GP algorithm with tournament selection. Each mutation operator is employed in five setups with different values of mutation and crossover probabilities (the source code of our experiments is available at `https://github.com/mszubert/ppsn_2016`).

Most of the GP parameters (summarized in Table 1) are adopted from the recent work on semantic backpropagation [12]. In particular, whenever a random mutation/crossover node needs to be selected, a *uniform depth node selector* is used. Given a program p, it first calculates program's height h, then draws uniformly an integer d from the interval $[0, h]$ and finally selects a random node from all nodes at depth d in program p. This technique has been recently shown to reduce bloat when compared to conventional Koza-I node selectors [6,12].

Moreover, both RDO and FPM use population-based library which is constructed at each generation from all semantically unique subtrees (subprograms) in the current population. Since we impose an upper limit on the tree height (17), when searching the library we ignore all the procedures that would violate this constraint when inserted into the parent program.

We investigate training error, generalization performance (error on 1 000 unseen test cases) and the size of programs produced by using particular mutation operators on 11 symbolic regression benchmarks. We consider six univariate and five bivariate problems that are adopted from previous studies [4,8,12]. Selected benchmarks (see Table 2) include polynomial, rational and trigonometric functions. For each problem, fitness was calculated as root-mean-square error on a number of training cases. The univariate problems use 20 cases distributed equidistantly in the $[-1, 1]$ range, while the bivariate ones use a grid of $10 \times 10 = 100$ points spaced evenly in the $[-1, 1] \times [-1, 1]$ square.

Table 1. Genetic programming parameters

Parameter	Value
population size	256
generations	100
initialization	ramped half-and-half with height range $2 - 6$ 100 retries until accepting a syntactic duplicate
instruction set	$\{+, -, \times, /, \exp, \log, \sin, \cos\}$ (log and / are protected)
tournament size	7
fitness function	root-mean-square error (RMSE)
node selection	uniform depth node selector
maximum tree height	17
number of runs	30

Table 2. Symbolic regression benchmarks.

Benchmark name	Objective function	Variables	Training cases
P4 (Quartic)	$x^4 + x^3 + x^2 + x$	1	20
P7 (Septic)	$x^7 - 2x^6 + x^5 - x^4 + x^3 - 2x^2 + x$	1	20
P9 (Nonic)	$\sum_1^9 x^i$	1	20
R1	$(x+1)^3/(x^2 - x + 1)$	1	20
R2	$(x^5 - 3x^3 + 1)/(x^2 + 1)$	1	20
R3	$(x^6 + x^5)/(x^4 + x^3 + x^2 + x + 1)$	1	20
K11 (Keijzer-11)	$xy + \sin((x-1)(y-1))$	2	100
K12 (Keijzer-12)	$x^4 - x^3 + \frac{y^2}{2} - y$	2	100
K13 (Keijzer-13)	$6\sin(x)\cos(y)$	2	100
K14 (Keijzer-14)	$\frac{8}{2+x^2+y^2}$	2	100
K15 (Keijzer-15)	$\frac{x^3}{5} + \frac{y^3}{2} - x - y$	2	100

6 Results and Discussion

Table 3 presents detailed characteristics of the best-of-run individuals evolved with particular mutation operators. Each row of the table corresponds to a single combination of one of the five GP setups (with different crossover (**X**) and mutation (**M**) probabilities) and one of the three considered mutation operators (either FPM, RDO or SRM). We performed 30 independent GP runs for each of such 15 combinations on each of the 11 symbolic regression problems. To confirm statistically significant differences between the results obtained with particular mutation operators, for each problem and parameters setup we conducted the Kruskal-Wallis test followed by a post-hoc analysis using pairwise Mann-Whitney tests (with sequential Bonferroni correction). We set the level of significance at $p \leq 0.05$. Table 3 shows with an underline the results that were found significantly better than those achieved with the other operators.

The first part of Table 3 shows the average training errors. Although RDO achieves the best overall results for most univariate problems, for the bivariate ones FPM produces more competitive results. Regardless of the parameter settings, the traditional SRM operator leads to the highest training error. Noteworthy, the RDO and FPM operators obtain their best results under different crossover and mutation settings. While both of them benefit from using traditional crossover as an additional variation operator, the performance of FPM decreases when mutation is performed too frequently (i.e., if $\mathbf{M} = 1.0$). To explain this phenomenon let us note that for a given subprogram, the FPM operator builds a context in a deterministic way. As a result, if two semantically equivalent subprograms are selected in the same generation, they will result in identical offspring. Consequently, FPM can lead to creating too many duplicated programs and thus losing diversity in the population. Importantly, although RDO is also deterministic, it is less susceptible to this problem because typically the number of distinct contexts is much larger than that of distinct subtrees.

In order to assess generalization performance of evolved programs, we calculate the root-mean-square error on 1 000 test cases drawn uniformly from the same range as for the training cases. The median test errors committed by the best-of-run individuals are presented in the second part of Table 3. In most cases, the RDO operator (especially for setups that achieve the lowest training error) suffers from substantial overfitting resulting in large test error. Although the FPM operator is also vulnerable to overfitting (in particular on problem P9) it is not as severe as in the case of RDO. With a few exceptions, for each of the considered problems and parameter setups, the FPM operator obtains the highest generalization performance.

Finally, we investigate the average size of best-of-run individuals which is presented in the last part of Table 3. Not surprisingly RDO is the most bloating operator and this is one of the reasons for its poor performance on the unseen test data. On the other hand, in preliminary experiments with imposed program size limit of 300 nodes, we also observed overfitting of the RDO operator. The programs produced by FPM tend to be much smaller. In particular, on two relatively simple problems, P4 and K13, the FPM operator finds short programs that obtain zero test error. Apparently, employing FPM allows to discover solutions that are very close to the original function underlying the training data. However, on all the other problems, the programs produced by RDO and FPM are significantly larger than those created by the traditional SRM operator.

7 Conclusions

Semantic GP operators have proved to be effective on a number of symbolic regression problems [14,13,11]. In this study, we confirmed these observations by analyzing the performance of the RDO operator based on semantic backpropagation [12] and the FPM operator that employs a novel idea of semantic forward propagation. When applied to a suite of symbolic regression benchmarks, both operators significantly outperformed the subtree-replacing mutation operator conventionally applied in GP. However, while both considered semantic operators achieved competitive performance on the training data, the RDO operator was found much more susceptible to overfitting. The proposed FPM operator, on the other hand, consistently produced shorter programs that obtained significantly lower error on the unseen test data.

Despite achieving superior predictive accuracy and producing shorter programs than RDO, the programs constructed with the FPM operator are still too large to be easily understood. This is unfortunate since finding comprehensible solutions has been always considered as one of the primary benefits of using GP instead of black-box machine learning methods. As most semantic-aware operators tend to produce large or very large programs [10], the problem of bloat remains the major challenge that can limit the practical applicability of such methods. Therefore, one of the most important directions of future work is to investigate the performance of RDO and FPM operators combined with parsimony pressure mechanisms that control the complexity of evolved programs.

Table 3. Detailed characteristics of best-of-run individuals produced by particular mutation operators (FPM, RDO, SRM), aggregated over 30 GP runs. Each operator was employed in 5 GP setups with different crossover (**X**) and mutation (**M**) probabilites. Bold marks the best results achieved under certain **X**/**M** settings on particular problems. Underline indicates statistically significant superiority.

			Average training error										
	X	**M**	P4	P7	P9	R1	R2	R3	K11	K12	K13	K14	K15
FPM	0.0	1.0	**0.0011**	0.0072	0.0153	**0.0064**	0.0049	**0.0024**	**0.0626**	**0.0418**	**0.0000**	**0.0031**	**0.0061**
	0.5	0.5	**0.0001**	0.0018	**0.0025**	**0.0012**	0.0018	**0.0006**	**0.0299**	0.0111	**0.0000**	**0.0012**	**0.0007**
	0.5	1.0	**0.0001**	0.0025	0.0037	0.0020	0.0025	0.0007	0.0358	0.0154	**0.0000**	**0.0021**	**0.0007**
	1.0	0.5	**0.0000**	**0.0015**	0.0022	**0.0012**	**0.0013**	**0.0004**	**0.0283**	0.0086	**0.0000**	**0.0012**	**0.0004**
	1.0	1.0	**0.0001**	0.0026	0.0029	0.0018	0.0019	0.0006	0.0334	0.0116	**0.0000**	**0.0017**	**0.0007**
RDO	0.0	1.0	0.0030	**0.0034**	**0.0147**	0.0071	**0.0043**	0.0030	0.0709	0.0444	0.0440	0.0587	0.0302
	0.5	0.5	0.0004	**0.0017**	0.0029	0.0023	**0.0014**	0.0018	0.0455	**0.0090**	0.0038	0.0132	0.0024
	0.5	1.0	**0.0001**	**0.0008**	**0.0004**	**0.0006**	**0.0004**	**0.0002**	**0.0294**	**0.0029**	0.0007	0.0044	0.0015
	1.0	0.5	0.0003	0.0020	**0.0008**	0.0014	0.0015	**0.0004**	0.0504	**0.0063**	0.0015	0.0087	0.0041
	1.0	1.0	**0.0001**	**0.0003**	**0.0003**	**0.0008**	**0.0004**	**0.0004**	**0.0294**	**0.0047**	0.0011	0.0033	0.0008
SRM	0.0	1.0	0.0518	0.0742	0.0758	0.0744	0.0811	0.0097	0.2025	0.3049	0.1552	0.2145	0.0723
	0.5	0.5	0.0323	0.0968	0.0732	0.0834	0.0608	0.0156	0.1769	0.2328	0.1040	0.1138	0.0608
	0.5	1.0	0.0449	0.0926	0.0638	0.0792	0.0880	0.0115	0.1781	0.2128	0.1267	0.1603	0.0748
	1.0	0.5	0.0217	0.0882	0.0715	0.0663	0.0666	0.0078	0.1598	0.2005	0.0866	0.1690	0.0623
	1.0	1.0	0.0282	0.0845	0.0792	0.0698	0.0754	0.0120	0.1942	0.2479	0.1437	0.1724	0.0628

			Median test error										
	X	**M**	P4	P7	P9	R1	R2	R3	K11	K12	K13	K14	K15
FPM	0.0	1.0	**0.0009**	**0.0084**	**0.0342**	**0.0071**	**0.0044**	**0.0026**	**0.0555**	**0.0529**	**0.0000**	**0.0028**	**0.0057**
	0.5	0.5	**0.0000**	**0.0046**	**0.0256**	**0.0025**	**0.0123**	0.0030	**0.0425**	0.0581	**0.0000**	**0.0017**	**0.0008**
	0.5	1.0	**0.0000**	**0.0045**	**0.0142**	**0.0037**	**0.0045**	**0.0015**	**0.0333**	**0.0290**	**0.0000**	**0.0032**	**0.0008**
	1.0	0.5	**0.0000**	**0.0069**	0.0306	**0.0030**	**0.0042**	**0.0017**	**0.0260**	**0.0311**	**0.0000**	**0.0021**	**0.0005**
	1.0	1.0	**0.0000**	**0.0055**	**0.0227**	**0.0025**	**0.0027**	**0.0009**	**0.0300**	**0.0295**	**0.0000**	**0.0024**	**0.0008**
RDO	0.0	1.0	0.0039	0.0593	0.0346	0.0087	0.0145	0.0071	0.1089	0.0988	0.0215	0.0774	0.0185
	0.5	0.5	0.0025	0.5159	0.0469	0.0406	0.0148	**0.0028**	0.0738	**0.0374**	0.0070	0.0252	0.0036
	0.5	1.0	0.0117	0.3084	0.0715	0.1522	0.0652	0.0618	0.0639	0.2124	0.0022	0.0556	0.0097
	1.0	0.5	0.0006	0.0704	**0.0104**	0.0081	0.0319	0.0057	0.0445	0.0364	0.0030	0.0283	0.0014
	1.0	1.0	0.0097	19.486	8E+3	0.0607	0.0466	0.0155	0.0402	0.3878	0.0023	0.0378	0.0026
SRM	0.0	1.0	0.0485	0.1170	0.1017	0.0836	0.0585	0.0123	0.2005	0.2649	0.1986	0.1458	0.0814
	0.5	0.5	0.0240	0.0958	0.0810	0.0730	0.0592	0.0106	0.1770	0.1874	0.1122	0.0988	0.0525
	0.5	1.0	0.0572	0.1865	0.0922	0.0800	0.0694	0.0105	0.1686	0.2037	0.1311	0.1207	0.0882
	1.0	0.5	0.0191	0.0899	0.0785	0.0711	0.0641	0.0101	0.1493	0.1874	0.0998	0.1126	0.0446
	1.0	1.0	0.0257	0.0734	0.0725	0.0739	0.0727	0.0142	0.1894	0.1853	0.1843	0.1723	0.0389

			Average program size										
	X	**M**	P4	P7	P9	R1	R2	R3	K11	K12	K13	K14	K15
FPM	0.0	1.0	172.6	179.0	195.9	162.2	187.9	161.0	210.9	172.4	**9.1**	204.9	207.7
	0.5	0.5	150.4	341.4	322.1	325.3	352.8	347.7	328.3	305.5	**7.4**	326.5	260.6
	0.5	1.0	**78.3**	292.4	271.7	287.9	283.3	265.9	286.4	264.5	**8.5**	258.9	239.6
	1.0	0.5	**44.0**	346.6	354.4	327.2	339.2	311.0	328.0	311.1	**7.8**	298.6	300.0
	1.0	1.0	**99.2**	283.4	271.0	255.7	253.3	270.6	244.8	230.6	**8.9**	239.8	264.4
RDO	0.0	1.0	537.6	690.6	550.8	777.5	2656.9	1203.7	418.6	434.8	85.0	147.2	250.2
	0.5	0.5	503.6	637.9	686.0	493.9	529.6	485.7	358.4	482.4	497.1	346.4	1299.6
	0.5	1.0	626.9	1004.3	934.1	906.7	854.0	747.2	654.2	841.2	464.3	548.6	1137.2
	1.0	0.5	378.6	631.2	588.4	473.0	508.9	486.7	316.8	472.9	311.0	325.5	673.8
	1.0	1.0	645.6	903.6	909.9	668.6	746.4	696.2	542.9	838.7	426.7	514.6	1034.4
SRM	0.0	1.0	**122.9**	**176.1**	**152.4**	**133.8**	**116.2**	**155.9**	**109.3**	**95.1**	95.7	**63.0**	**74.7**
	0.5	0.5	**60.0**	**109.4**	**95.4**	**79.7**	**76.1**	**95.8**	**62.7**	**69.4**	59.6	**53.5**	**57.5**
	0.5	1.0	111.8	**172.8**	**159.6**	**154.3**	**122.3**	**173.6**	**99.2**	**95.3**	96.1	**79.1**	**82.0**
	1.0	0.5	97.9	**106.4**	**107.1**	**96.8**	**89.9**	**137.2**	**89.5**	**86.6**	81.1	**87.6**	**64.4**
	1.0	1.0	119.0	**160.9**	**147.5**	**150.6**	**131.0**	**165.7**	**95.3**	**83.2**	95.9	**80.4**	**96.5**

Acknowledgments

This work was supported by the National Aeronautics and Space Administration under grant number NNX15AH48G.

References

1. Beadle, L., Johnson, C.G.: Semantically Driven Crossover in Genetic Programming. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2008. pp. 111–116. IEEE (2008)
2. Beadle, L., Johnson, C.G.: Semantic Analysis of Program Initialisation in Genetic Programming. Genetic Programming and Evolvable Machines 10(3), 307–337 (2009)
3. Jackson, D.: Promoting Phenotypic Diversity in Genetic Programming. In: Parallel Problem Solving from Nature, PPSN XI, Lecture Notes in Computer Science, vol. 6239, pp. 472–481. Springer Berlin Heidelberg (2010)
4. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA (1992)
5. Krawiec, K.: Behavioral Program Synthesis with Genetic Programming, Studies in Computational Intelligence, vol. 618. Springer (2016)
6. Krawiec, K., O'Reilly, U.M.: Behavioral Programming: A Broader and More Detailed Take on Semantic GP. In: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation. pp. 935–942. GECCO '14, ACM (2014)
7. Liskowski, P., Krawiec, K., Helmuth, T., Spector, L.: Comparison of Semantic-aware Selection Methods in Genetic Programming. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 1301–1307. ACM (2015)
8. McDermott, J., White, D.R., Luke, S., Manzoni, L., Castelli, M., Vanneschi, L., Jaskowski, W., Krawiec, K., Harper, R., De Jong, K., O'Reilly, U.M.: Genetic Programming Needs Better Benchmarks. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 791–798. ACM (2012)
9. McPhee, N.F., Hopper, N.J.: Analysis of genetic diversity through population history. In: Proceedings of the Genetic and Evolutionary Computation Conference. vol. 2, pp. 1112–1120. Morgan Kaufmann (1999)
10. Moraglio, A., Krawiec, K., Johnson, C.G.: Geometric Semantic Genetic Programming. In: Parallel Problem Solving from Nature - PPSN XII, Lecture Notes in Computer Science, vol. 7491, pp. 21–31. Springer Berlin Heidelberg (2012)
11. Pawlak, T.P., Wieloch, B., Krawiec, K.: Review and Comparative Analysis of Geometric Semantic Crossovers. Genetic Programming and Evolvable Machines 16(3), 351–386 (2015)
12. Pawlak, T., Wieloch, B., Krawiec, K.: Semantic Backpropagation for Designing Search Operators in Genetic Programming. IEEE Transactions on Evolutionary Computation 19(3), 326–340 (2015)
13. Uy, N.Q., Hoai, N.X., O'Neill, M., Mckay, R.I., Galván-López, E.: Semantically-based Crossover in Genetic Programming: Application to Real-valued Symbolic Regression. Genetic Programming and Evolvable Machines 12(2), 91–119 (2011)
14. Vanneschi, L., Castelli, M., Silva, S.: A Survey of Semantic Methods in Genetic Programming. Genetic Programming and Evolvable Machines 15(2), 195–214 (2014)
15. Wieloch, B., Krawiec, K.: Running Programs Backwards: Instruction Inversion for Effective Search in Semantic Spaces. In: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation. pp. 1013–1020. GECCO '13, ACM, New York, NY, USA (2013)

6 List of manuscripts generated through this award.

Citations attracted by the manuscripts listed below can be found on PI Bongard's Google Scholar profile.

References

[1] Nicholas Allgaier, Tobias Banaschewski, Gareth Barker, Arun LW Bokde, Josh C Bongard, Uli Bromberg, Christian Büchel, Anna Cattrell, Patricia J Conrod, Christopher M Danforth, Sylvane Desrivières, Peter S. Dodds, Herta Flor, Vincent Frouin, Jürgen Gallinat, Penny Gowland, Andreas Heinz, Bernd Ittermann, Scott Mackey, Jean-Luc Martinot, Kevin Murphy, Frauke Nees, Dimitri Papadopoulos-Orfanos, Luise Poustka, Michael N. Smolka, Henrik Walter, Robert Whelan, Gunter Schumann, Hugh Garavan, and IMAGEN Consortium. nonlinear functional mapping of the human brain. *arXiv preprint arXiv:1510.03765*, 2015.

[2] James P Bagrow, Suma Desu, Morgan R Frank, Narine Manukyan, Lewis Mitchell, Andrew Reagan, Eric E Bloedorn, Lashon B Booker, Luther K Branting, Michael J Smith, Brian F. Tivnan, Christopher M. Danforth, Peter S. Dodds, and Joshua C. Bongard. shadow networks: discovering hidden nodes with models of information flow. *arXiv preprint arXiv:1312.6122*, 2013.

[3] David Buckingham, Christian Skalka, and Josh Bongard. inductive machine learning for improved estimation of catchment-scale snow water equivalent. *Journal of Hydrology*, 524:311–325, 2015.

[4] Ilknur Icke and Joshua C Bongard. improving genetic programming based symbolic regression using deterministic machine learning. In *Proceedings of the 2013 IEEE Congress on Evolutionary Computation*, pages 1763–1770. IEEE, 2013.

[5] Ilknur Icke and Joshua C Bongard. modeling hierarchy using symbolic regression. In *Proceedings of the 2013 IEEE Congress on Evolutionary Computation*, pages 2980–2987. IEEE, 2013.

[6] Sam Kriegman, Marcin Szubert, Josh C Bongard, and Christian Skalka. Evolving spatially aggregated features from satellite imagery for regional modeling. In *International Conference on Parallel Problem Solving from Nature*, pages 707–716. Springer, 2016.

[7] Marcin Szubert, Anuradha Kodali, Sangram Ganguly, Kamalika Das, and Josh C Bongard. reducing antagonism between behavioral diversity and fitness in semantic genetic programming. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, pages 797–804. ACM, 2016.

[8] Marcin Szubert, Anuradha Kodali, Sangram Ganguly, Kamalika Das, and Josh C Bongard. semantic forward propagation for symbolic regression. In *International Conference on Parallel Problem Solving from Nature*, pages 364–374. Springer, 2016.

[9] Afsoon Yousefi Zowj, Josh C Bongard, and Christian Skalka. a genetic programming approach to cost-sensitive control in resource constrained sensor systems. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 1295–1302. ACM, 2015.