A Call for Collaborative Landscape Analysis

Deon Garrett
Icelandic Institute for Intelligent Machines
and the School of Computer Science, Reykjavik University Reykjavik, Iceland
deon@iiim.is

ABSTRACT

In developing effective search and optimization algorithms, it is crucial that the specific features of the problem be taken into account. This observation has led to a great deal of research in how to abstract away trivial details in favor of the core concept that describes such features, with the goal of developing a more general theory of search algorithm performance. However, our efforts have not taken advantage of the great developments in data-driven machine learning that have arisen in the past decade or so. Rather, most work still starts from a clean slate and focuses on collecting and analysing only the limited landscape data that each researcher deems useful for each specific problem. In this position paper, I argue for the development of an open repository of this data – open both in the sense of freely available to all researchers as well as in the sense of an "open-world" assumption concerning the types of data to be collected and analyzed. This paper discusses some of the important issues that would need to be resolved to build such a system in a way that would provide the most value for the field.

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]

General Terms

Algorithms, Experimentation, Standardization

Keywords

Machine Learning, Landscape Analysis

1. INTRODUCTION

It has been known for some time that problem-dependent structures can have a large impact on the performance of search algorithms. One consequence of this has been that a number of researchers have worked on analysis of particular combinations of problems and algorithms with the goal of producing a more general theory to guide the selection of suitable algorithms. However, to date, it has proven difficult to generalize the results of any particular study. Often, the particular tools and techniques developed for one problem are difficult to apply to others, and what information is valuable for a given search algorithm may not be valuable for another. As a result, most researchers have not taken a cumulative approach to landscape analysis in which data from earlier projects or experiments directly informs newer work. It has been left to the researcher to track down relevant papers, often reimplementing landscape analysis metrics from scratch or adapting them to different problems, and applying them to newer problems or algorithms.

In the machine learning community, there has been a rapid shift towards data-driven approaches. There are countless algorithms for detecting subtle patterns in various types of data, and modern computing power enables many of these algorithms to work on vast data sets using desktop calibre hardware. Landscape and search algorithm analysis can produce data in a form that should be usable by these methods, but we need a concerted effort on the part of researchers to aggregrate our data together in a consistent form amenable to this type of analysis. In this paper, I hope to make the case for why we should attempt to build such a collaborative approach, and I will present what I see as the major issues that would need to be addressed.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GECCO'12 Companion, July 7–11, 2012, Philadelphia, PA, USA.
Copyright 2012 ACM 978-1-4503-1178-6/12/07...$10.00.

2. PREVIOUS WORK

Obviously, there is a great deal of existing work relating to landscape and search space analysis, all of which is relevant to the proposed goal. The issue is how to facilitate greater collaboration around the output of all these disparate research studies. There have been a few steps in the right direction.

In GECCO 2011, Mersmann et. al. [15] presented a very nice paper describing a machine learning approach to one of the core problems facing those of us working on landscape analysis – how to tie observable properties of a search space (what they call "low-level" features) with the higher-level and more abstract ideas that we tend to associate with being well or poorly suited for particular types of search strategies. An obvious path along which this work will continue to develop is to then learn mappings on these high-level features to help predict which search algorithms will be most effective.

However, there were also a number of limitations in that paper. Only the BBOB problems were considered, discussed below. These benchmarks are nice in that they are designed

in terms of "groups" which makes it somewhat easier to extract hypotheses around algorithm performance, but many of the problems are trivial to solve, and there's little reason to believe that one can learn much from a sphere model that will be directly relevant to, e.g., scheduling problems with real-world constraints. In addition, while they used a multi-objective algorithm as part of the learning step, the analysis was confined to only single-objective problems. In short, we need to both generalize this work to arbitrary problems and search algorithms as well as define some consistent representations that allow researchers to easily take advantage of each other's work with as little friction as possible.

For the past few years, there has been an ongoing workshop series on black-box optimization benchmarking (BBOB) [9, 7, 8]. In this program, researchers are given a set of benchmark problems in various languages and a fairly complete set of tools for comparing performance measures across the set of algorithms applied. However, thus far there have been fairly substantial limitations. Most importantly, the focus has been exclusively on real-valued single-objective optimization problems. In addition, as the focus of the BBOB suite is strictly on comparing performance of algorithms, it has no particular focus towards lower-level features. That is, we expect intuitively that the performance of some algorithm on some problem is strongly dependent on the component features of both. The goal of understanding problems is to be able to predict the BBOB (and other) results from a more fundamental understanding of the problem.

In addition to the BBOB problems, there have been several other widely accepted benchmark problems, many of which have already been subjected to intensive studies of various properties of their fitness landscapes. For instance, the CEC benchmarks [17, 18] have been updated multiple times and cover a much wider range of problem types. The CEC-2005 benchmarks have been completely characterized in terms of their fitness landcapes [16]. However, as of yet, these studies have not been carried out in a systematic way allowing other researchers to easily combine the different approaches taken in these analyses.

This paper argues that we need to decide on a way to allow the type of research done by Mersmann and colleagues to be *shared* throughout the community, as well as to support extensions to that work. Ideally, there would be a shared corpus – as each researcher developed new analysis methods or approached new problems, the insights gained would be directly combined with the work of numerous other researchers. I want to be able to pull data concerning the relationship between autocorrelation and local search performance from numerous problems contributed by numerous researchers as effortlessly as possible.

In that work, the authors constructed a grid, where each row represented a particular problem instance, and each column contained some type of information about the instance, in particular very low-level features observed directly. This data was then used to map these low-level features onto higher-level notions such as "smoothness". I am proposing that this basic idea be generalized and adapted into a structure that (a) allows a richer set of information regarding landscape structure and algorithm performance to be directly represented, and (b) facilitates a much greater degree of collaboration in both obtaining the raw data as well as in the analysis of the data. Note that in their work, Mersmann et. al. chose features that were universally applicable across their entire study, a task made easier by the restricted domain from which the test functions were drawn. One consequence of this is that there is actually very little information carried along with each feature. Each is, in essence, nothing more than a number attached to a label. Details of how the feature was sampled, how it is to be interpreted, etc., are left to be documented external to the data, which is an acceptible strategy given that the entire study is described uniformly in a single paper. This simplifies matters greatly, and we could choose to do something similar in designing a system for collaborative landscape analysis. Many of the problems associated with settling on static schemata mostly go away if we abandon the idea of fully specifying what each feature means.

In recent years, a number of research groups have turned to the problem of generalizing metaheuristic performance, with the goal of automating the design of good optimization algorithms. Commonly referred to as hyperheuristics [5], the convention here is to build algorithms to search the space of search algorithms. Thus for each new problem to be solved, a researcher or practitioner would simply run a search algorithm to find the best algorithm and set of parameters for the problem. Obviously, there are parallels to be drawn here. Last year, there was a hyperheuristics challenge [4] in which, similar to the BBOB work, participants were invited to submit their algorithms to compete across a range of problems, including some not known to the teams beforehand. This type of interoperability will be an important part of enabling researchers working in landscape analysis to capitalize on each other's work in similar ways.

Additionally, it would be a great boon to the hyperheuristics community directly if large amounts of data concerning how different search algorithms perform on different problems. Even more crucially, the availability of such data along with modern data-mining and machine learning techniques holds the promise for advancing the state of knowledge on what types of search algorithm components work best on what types of low and mid-level search space structures.

Looking beyond motivation into potential implementation strategies, the PISA project [2, 12, 11, 1] is another potentially relevant source of inspiration. While it has no particular support for landscape analysis, it does feature a similar flavor in that it intends to support researchers working on different optimization techniques and using different code bases to more easily share information. In the case of PISA, this information is mostly concentrated on a repository of common algorithms and performance comparison methods rather than landscape analysis, but it could be extended to support this functionality or serve as an inspiration for a different implementation.

Note that PISA is intended, not as a landscape analysis framework, but as a framework for implementing and testing new algorithms. In that niche, there are numerous other projects that have been more successful in terms of attracting active users. The ECJ system [14] and the ParadiseEO system [6, 13] are particularly noteworthy in this regard. However, both are self-contained, requiring that users live within those systems. Choosing to build a collaborative framework on one of these systems is to dictate language and platform choices to the users, excluding those who wish to choose differently from participating as easily. It may well be worth considering those platforms as additional options, but any open system for facilitating the type of collaboration

we want must provide reasonably convenient access independent of language or platform selection by its users. PISA is an interesting choice in this respect, as it attempts to be completely agnostic. However, the programming model it imposes is likely part of the reason it never attracted the volume of users as the other systems. This, likewise, must be a consideration in deciding how to build a collaborative landscape analysis tool.

For the remainder of this paper, I will assume (perhaps wrongly) that there is general agreement that the basic idea is desirable, and present the issues, options, and trade-offs of various implementations as I see them.

3. WHAT LEVEL OF SUPPORT?

The high-level goal is to better enable researchers working on search space and landscape analysis to make use of each other's work. There are a number of avenues available to attempt to reach this goal, with varying degrees of structural support. We can roughly categorize these options along a single dimension – the amount of automated or semi-automated functionality provided.

At the minimum level of support, the community could establish a single hub for activity related to landscape analysis. Several communities maintain bibliographies, for instance, where any relevant published paper can be added to a growing list. In addition to a list of published papers, such a hub could also host source code, datasets, or whatever else was deemed to be of use to the community.

The downsides are obvious – it requires a large amount of human effort to keep such a list maintained, particularly as the community it serves grows. The emergence of wikis helps greatly in this regard, and it is the view of the author that this should be viewed as a minimum step to be taken. However, it goes a relatively short distance towards the type of analysis I would like to see made possible. It still requires researchers to manually track down source code, potentially from different contributors (e.g., I want to look at the work done by group X and the work done by group Y on instances of problem P). This means that problem instances need to be obtained, code must potentially be ported from one framework to another, results carefully checked by hand against published data, etc. These are all the things we have to do today to build on each other's work, only made slightly simpler by having a more reliable place from which one can get the first step started.

To build significantly toward that goal, a more automated approach is needed. Ideally, a researcher should be able to directly use the results obtained by other researchers, without this costly middle step of obtaining, customizing, and reproducing each other's data and code. Rather, if one wants to look at the relationship between, for instance, tabu search with a particular neighborhood operator and the smoothness of the fitness landscape, he or she should be able to obtain all the data contributed by other users relevant to that particular experimental goal. In the optimum case, such data could potentially be retrieved directly in a form suitable for use with Weka [10] or another machine learning toolkit where the relationship between the variables of interest could be more easily explored. Certainly, it is still important that source code and problem instance data be made available, both for reproducibility as well as the inevitable need for "tweaks", but the automated retrieval of relevant information would be a boon to research into search algorithm performance.

Consider a complete static schema that could represent any possible analysis technique, problem type, search algorithm, and performance metric. Ignoring for the moment the enormous difficulty of defining such a schema, this would allow very sophisticated tools to be developed. Researchers could simply describe their work in terms of this schema, and then upload the results to a central repository where other researchers could use a very rich set of semantics to obtain, analyze, and build on such data. Unfortunately, of course, it isn't possible to define such a general schema, at least not in a way that is both usable and useful. Instead, we need to consider different trade-offs and determine how best to allocate the limited efforts of those in the community to achieve the goal. The following section examines this trade-off in more detail.

4. DEFINING A REPRESENTATION

At the core, the ability to more effectively share our data and results in the proposed way comes down to to one key issue: how do we define a common language that is rich enough to allow each person to express his or her particular data, and yet constrained enough so that any two people can have some hope of reusing each other's work? Our current approach – each researcher starting from scratch – is very well optimized for the first of those goals, but yields insurmountable obstacles to the second. So in reality, the problem is mostly about how to constrain the language people use internally so that we can be more consistent. Note that the second major issue (implementation and tool support) will be discussed in Section 6.

As mentioned in the discussion of the Mersmann et.al. paper, there the authors were able to mostly sidestep the issue because (a) they were in complete control of which features they wanted to included in the dataset, and (b) their analysis was restricted to a very particular type of question. In attempting to generalize this idea to the wider realm of arbitrary analysis, we have to admit a much broader notion of what types of information will be needed, not only now but also in the future.

Consider an example of the type of landscape analysis that researchers actually perform. For simplicity, let's look at perhaps the best known idea in the field – fitness-distance correlation (FDC) as performed by Boese [3]. In FDC analysis, the goal is to estimate the degree to which being near the optimum in terms of objective function value is correlated with being close to the optimum in terms of the number of steps needed for a candidate search algorithm to move there from the given search point. To estimate this correlation, we start a search algorithm from some set of randomly generated search points, for each recording the distance to the nearest optimal solution in objective space as well as the number of moves needed to get there under our search algorithm. There are thus a number of parameters we need to specify to determine the outcome of our analysis.

- Type of Analysis (Fitness-Distance Correlation)
- Problem class (TSP)
- Problem instance
- Search algorithm for probes

- Neighborhood structure
- List of probes
  - Starting point of probe
  - Nearest optimal solution
  - Number of steps to reach optimum
- Fitness-Distance Correlation coefficient

Note that this list is but one possibility. We might decide not to include the individual search points, or we might include additional information such as whether the TSP instance was Euclidian, symmetric, etc.

In addition, we would want to include information, perhaps in a separate data structure, information on algorithm performance. Data such as the list shown above allows us to collect information about the FDC coefficients for a potentially large number of problems, but does not address the next step of associating algorithm performance with these properties. Again, to pick a specific but arbitrary example:

- Search Algorithm (Robust Tabu Search)
- Problem class (TSP)
- Problem instance (...)
- List of trials
  - Number of evaluations
  - Best solution found
  - Evaluations to find best

If we imagine all this data being put into some form that could be queried in a flexible way, we could suddenly allow people to easily look at all of the algorithms that have been tested on problems where the FDC coefficient was less than -0.4, for example.

One obvious solution for representing this type of data in a somewhat standard format would be to define an XML wrapper for the relevant data, perhaps similar to that shown in Figure 1. XML is terribly flawed in a number of ways, but it does have the advantage of being very well understood by pretty much everyone, which is important when considering a format intended to foster collaboration. Other formats could of course be selected instead, but I'll stick with XML for the purposes of illustrating the basic concepts in this paper.

This somewhat bloated example demonstrates several issues. First, it is necessary that the schema be general enough to describe several different types of datasets relevant to landscape analysis in general. The example makes some attempt at that goal by providing general tags for things like the problem class and encoding, but it will be very difficult in general to get it exactly right. Things like "probes", the way that solutions are represented, etc., may differ with other problems and datasets, and the schema would need to anticipate this generality. It is rather easily seen that no single pre-specified design will cover all possible cases, so the assumption of an open-world will be essential. However, there are two ways that such an assumption may be manifest.

```
<dataset>
  <metric>Fitness-Distance Correlation</metric>
  <problem>
    <class>TSP</class>
    <encoding>permutation</encoding>
    <instance>foo123</instance>
  </problem>
  <analyzer>
    <type>local search probes</type>
    <search-type>Next-descent</search-type>
    <neighborhood>2-opt</neighborhood>
  </analyzer>
  <probes>
    <data-point>
      <start-point>
        <solution>0 1 2 3 4 5 6 7</solution>
        <fitness>495</fitness>
      </start-point>
      <end-point>
        <solution>0 4 6 2 7 1 3 5</solution>
        <fitness>307</fitness>
      </end-point>
      <nearest-optimum>
        <solution>3 5 2 7 6 1 0 4</solution>
        <fitness>289</fitness>
      </nearest-optimum>
    </data-point>
    ...
  </probes>
  <result>
    <value>0.317</value>
  </result>
</dataset>
```

Figure 1: Sample XML for a run collecting FDC data on an example of the traveling salesperson problem. Note that this schema is quite detailed, which provides a great deal of information concerning the data point, but at the expense of generality. It is difficult to design such a detailed format that is applicable to a wide range of different datasets.

One way that we might accomodate arbitrary datasets is to dispense with the notion of a specified schema entirely. Instead, we simply allow each individual user to specify whatever data makes sense in a simple, basically untyped and unchecked format like keyword-value pairs. Represented in XML, the corresponding idea would be that any syntactically valid XML would be acceptable, but the semantic aspects (tag names, hierarchical structure, etc.) would be completely arbitrary and left to each user.

The upside is obviously the great flexibility such a system entails. Arbitrarily complex analysis and sampling techniques could be specified using exactly the set of attributes needed. However, this flexibility would come at the price of consistency. If I want to look at FDC analysis on TSP versus the quadratic assignment problem, I need to be able to rely on the fact that everyone who has submitted data in this area has used the same names and formats to describe the same ideas. Consistency becomes a social problem rather than a technical one. That is, it shifts the burden to the researcher to search the existing tags to see if someone

else has already contributed data similar to his or her own in some way, and then match the tags accordingly. This process could be facilitated via tool support as well – the submission system could parse each submitted contribution and automatically generate a listing of all tags used in the entire system, for instance. However, the ultimate burden would fall to the users, and it's not clear that such a system would retain enough structure to be useful.

The other approach to allowing open-ended data collection is to provide much more rigidly defined schemata for the cases we are aware of, and then allow the individual researchers within the community to extend them as necessary. Again, the advantages and disadvantages are both fairly easily seen. As long as everyone uses the same sets of information, the repository will provide the maximum benefit. However, it seems quite difficult to determine how best to build such a system up front. Ideally, the need for extension would be minimized; otherwise this method degenerates to the schema-free method above. Thus it becomes important to design schemata that are not brittle in the face of non-substantive changes. That is, if two people are applying very similar techniques on very similar problems, the minor differences cannot cause them to need to extend the given system in different ways, and preferably not have to extend it at all.

In both of these approaches, there is another problem to be addressed – how to "backport" new information onto existing data. That is, if user X submits a dataset containing a particular type of analysis on a particular problem set, and then user Y submits a slightly different set of analyses on the same problems, how should the data be represented? Should the different types of data be merged into a single record for that problem instance or kept separate? If they are merged, how should we handle overlaps (two researchers contribute the same analysis on the same problem instance)?

Another way to look at the problem is to imagine the data, not at the level of file formats and schemata, but as the input to a machine learning technique. Viewed in that way, we have a set of rows, each of which is a single data point describing the outcome of some experiment. For example, we might have one row that shows the performance characteristics of a genetic algorithm using Edge-Assembly Crossover (EAX) on a symmetric TSP instance. That TSP instance may have a number of known properties – perhaps we (or someone else) have looked at the FDC coefficient, estimated the number of local optima, and estimated the barrier height using a 2-opt local search heuristic. However, other properties have not been examined yet – other problems may have been subjected to autocorrelation analysis to estimate smoothness, but for our instance that data is not known. Viewed in this way, we have a table: the columns may contain some attribute relating to the structure of the search space according to some metric, some aspect of the performance of a given search algorithm, or human-assigned labels such as problem class, while each row corresponds to a particular problem instance about which we hope to record information.

In the machine learning view, such a dataset is merely a set of labeled columns with numbers or other measures for each data point. Figure 2 shows an example of the kinds of information we might store in this way. The advantage of this type of representation, aside from the obvious benefit of being well-suited for classification and data mining techniques, is that it is also relatively easy to extend. One can simply "fill in the gaps" in the data by testing particular analysis methods or algorithms on known problem instances and posting the results.

The goal in this view is to construct this table in the most consistent way possible such that we may use it to derive insights, probably via the use of data-mining and machine learning methods, about the relationship between search algorithms and problem features. Note that this is a somewhat simplified view. In all likelihood, the data would not be a single table, but rather a set of related tables, perhaps even stored in a relational database somewhere.

The existing body of work on landscape analysis gives us a reasonable common ground to build on. There are key concepts that are relevant to a wide range of methods for search space analysis – local searches used as "probes", with the associated notions of local search neighborhoods, distance metrics, and encodings. In short, there are general classes of landscape analysis methods. We could define one such class as the set of methods that perform one or more local search steps and report a numeric measure at the end.

Keeping with Mersmann et. al.'s terminology [15], we can consider "low-level" features as those that can be directly observed by some process. One of the key goals of any landscape analysis method is the ability to tie these low-level features into more abstract and meaningful high-level features that cannot be directly observed. For instance, we might deploy some number of random-walk probes and measure the autocorrelation along each path. This feature can be thought of as an approximation for local smoothness, a higher-level feature of the problem likely to have a meaningful relation to the performance of different types of search algorithms.

Because of the perceived difficulty in getting the modeling step right, my intuition is that the preferable solution is to focus explicitly on modeling low-level observed features and search algorithm performance. In this view, there are essentially only two types of records. The first contains problem identifiers of some sort, probably at least the general class of problem (e.g., TSP) as well as the specific instance of the problem. In addition, this type of record would contain only very simple feature measurements, essentially nothing more complex than just a "feature-id" and "feature-value" pair. Expressed in hypothetical XML, this version would look something like that shown in Figure 3.

Compared to the much more detailed snippet in Figure 1, this version contains very little information. All of the details concerning how each feature was measured have been removed, presumably to be made available via associated documentation. However, there are a number of benefits as well. First, the schema is extremely general. With a bit of thought, one could likely define a data format that was both general enough to support a wide range of use cases, while being specific enough to support automated checking (e.g., does the feature type already exist in the system?).

This model does mean that there will be some burden on the researcher to look around a bit to determine if the metric he or she wishes to include has already been submitted under another name. However, this burden could be lessened somewhat by having the upload process parse the data and regenerate dynamically parts of the system documentation. That is, an up-to-date list of problem classes, feature types, etc., could be maintained directly from the submitted data.

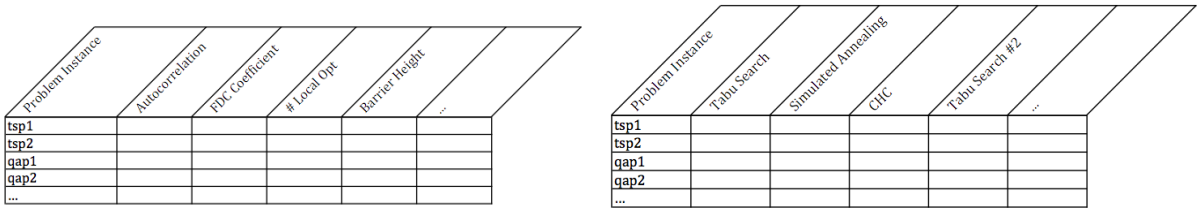

Figure 2: Sample of a data format useful for machine learning algorithms. One contains mappings of search space features to problem instances (left), the other from search algorithms performance to problem instances (right).

```
<dataset>
  <problem>
    <type>TSP</type>
    <instance>foo123</type>
  </problem>
  <features>
    <feature>
      <type>FDC</type>
      <value>0.317</value>
    </feature>
    <feature>
      <type>Barrier-height</type>
      <value>2.49</value>
    </feature>
  </features>
</dataset>
```

Figure 3: A much simpler form of feature representation.

In addition, the system could refuse to accept novel types of certain tags until the user submitted some basic documentation covering their meaning. These processes would of course require some degree of effort up front as part of building the initial system, but the work required is fairly minimal, and they are mostly optional niceties anyway.

Another limitation of almost all the relevant work is that there has been a fairly strong focus on single-objective optimization. Many of the same techniques can be applied to multi-objective optimization, but there are entire new classes of analysis that are valuable as well. Basically, multi-objective optimization involves two goals: move towards better solutions and spread out across the set of mutually incomparable solutions. For the former goal, there is a very direct analogue back to single-objective optimization. In both cases, the goal is to move "downhill". In multi-objective search, there is no single direction corresponding to downhill, but many search algorithms impose a direction as a key step in the search, and once this is done, the analysis is quite similar (although the search space itself can be, and often is, radically different). However, measuring diversity generally requires different types of techniques. Because the output of multi-objective optimization is always a set, any fitness-based metrics must be recast into set-based metrics. Thus, where many single-objective analysis methods take the form $\mathcal{F}(s \in S) \rightarrow \mathbb{R}$, where S is the set of candidate solutions, multi-objective techniques most often take an entire set as input, mapping it onto one or more numbers representing the proximity to the desired Pareto-optimal set as well as the diversity of solutions in the given set.

5. DATA CONSISTENCY ISSUES

It seems necessary for any such system to be very open in terms of what types of data it allows. It is certain that no predetermined set of properties will be able to naturally anticipate all the varied things that clever researchers will want to do going forward. It's likely that no suitable static representation could even encompass all the known approaches that have already been done, at least not in any useful manner. However, it seems equally well-established that a completely unspecified system will tend toward chaos quite quickly. These observations lead naturally to the ideas described in the previous section – attempt to provide a useful set of static descriptors that seem to make sense in general, but allow for customization in a well-defined way.

Aside from the issue of figuring out a suitable schema or schemata that can accomodate the needed flexibility while maintaining as much consistency as possible, there is another issue that may need to be addressed. Software bugs will inevitably arise, leading researchers to contribute incorrect data to the repository. Thus one requirement is that the data be traceable. Every piece of data contributed must be tagged in a way that allows identification of its source.

If the data is distributed as a loosely coupled set of, e.g., zip files, erroneous data is easy to fix – one simply retracts the affected files. However, if we attempt to merge results in any way, it may become tricky to handle errors in the data. For example, suppose that researcher A contributes a FDC analysis on a set of TSP instances and shows the performance using a particular search algorithm. Researcher B, interested in the relationship between the number of local optima and algorithm performance, contributes that data to the repository. There are multiple ways the new information could be integrated, some of which make it difficult to recover in the case that it is later discovered that the data was incorrect.

6. TOOL SUPPORT

Obviously, whatever use such a system has extends only so far as people actually use and contribute to it. One of the key factors affecting that would likely be how much of a change it requires to the normal workflow of each researcher. At a minimum, data must be collected and distributed in a standardized format. I think to get the greatest benefit though, more than that needs to be done. In particular, I

think it is crucial to be able to merge two sources referring to the same thing. That is, if person A collects metrics X and Y for some problem, and person B collects metric Z on the same problem, it must be simple to obtain a single dataset contain all of X, Y, and Z for that problem.

This opens a number of possibilities. How should we resolve two different researchers supplying the same data (e.g., A does X and Y while B does Y and Z – which Ys do we take?). What happens if B comes along later and says that his or her data for attribute Z is actually incorrect? If the system is merging results as they come in, it must store enough information to be able to separate them again at some point in the future, or alternately, checkpoint everything.

6.1 Client Support

Different researchers will in general choose different languages and platforms for their own use, and clearly, any solution intending to foster greater collaboration needs to allow for that. One way to do this is to provide code that manages the process of storing the feature and performance data in an appropriate format, and provide ways for users to call that code.

The approach taken by PISA is to build their system as independent communicating processes which can be written in any language and which communicate with one another via the filesystem. My general impression is that most people don't like this model that much, feeling that it imposes something of a "weird" constraint on the way they normally write software. However perhaps the domain here is different enough that the filesystem is a more natural way of passing the feature space and performance data around the different components of a system. If that is the case, we could simply provide ready-to-run code to read formatted files and massage the data however made the most sense.

Another approach is to, instead of striving for maximum flexibility at the expense of natural usability, simply distribute small modules in a few common languages such as C++ and Java. Such modules could provide very basic functionality, such as reading and writing the defined schemata. Here is where the more popular heuristics frameworks such as ECJ [14] and ParadisEO [6, 13] might be involved – once a set of functionality was decided on, specific modules for these systems could be provided in addition to a more general method of interacting with the repository.

Much of the difficulty in making this entire idea into something workable revolves around how to define things tightly enough that everyone is on the same page, and yet loosely enough that people can do interesting things with it. It is my belief that given the nature of the problem, there is a strict limit on what can be done at a library or framework level. The very idea of measuring properties of a landscape can require quite detailed interactions with the rest of the code in a system, and providing that sort of cohesion in an agnostic module is not a realistic goal. Instead, I think the best option is to define a few small APIs for reading and writing the standard file formats. For instance, there could be an API function to write a particular type of record (e.g., a data point for a FDC run). The user would collect this data via whatever mechanism he or she would normal use, and then call only the single API call passing in the required data. As researchers need to extend the defined behaviors, they can simply write and submit code to handle their extensions. As we generally have to write out our data in some format anyway, the overhead imposed to write it in a compatible format is not so great.

The other side of client support is the question of how to enable researchers to obtain data from the repository in a form that they can directly use. Models like the BBOB suite that simply offer individual downloads work well which each piece is self-contained. However, if I want to look at all the available data for a particular problem, it may be distributed over many different contributed datasets. Tool support would be needed to provide easy ways to combine (or split, or otherwise process) data from the repository.

6.2 Server Support

Probably the most important factor governing what sort of infrastructure we should build is in how capable we want the central repository to be. If we take the BBOB approach (researchers simply send data and source code as files that are linked from a web page), then the server needs are basically covered by an off-the-shelf web server. The client needs then consist simply of whatever functionality is deemed useful to get data into a consistent format, which as discussed above, may involve fairly involved and intricate logic.

Alternately, if the server was more sophisticated, performing some sort of processing or filtering on the data coming in, both the client and server sides might require more sophisticated processing. Ideally, I think this is the preferred state of affairs. Given a strict schema, the server could accept uploaded files, parse them, and perform more intelligent operations such as merging any new data with compatible existing records. This would allow fairly sophisticated queries on the data as well (e.g., "show me all the results for the CHC algorithm on problems with plateaus."). However, this approach requires a great deal more effort up front and probably requires considerable effort to ensure that the stricter data schemata required have been adequately thought through. For these reasons, as well as issues surrounding security and traceability that would need to be considered, I don't think this is a reasonable first goal.

7. CONCLUSIONS

It is my view that a more coordinated effort could provide very large benefits for the landscape analysis community. By standardizing on a common language for modeling problem structure and algorithm performance, we could much more quickly build a large and varied corpus of data sufficient to begin to see real results from the application of machine learning tools to the problem of predicting the concordance between the operational principles of a search algorithm and the inherent structure of a given optimization problem.

In order to make this a reality, we would need to decide on what the general form of the system will be. Do we want to track only low-level features or should higher-level derived features be present as well? How specific and detailed should the data collected be? How much work should the system try to do – just host static files in a compatible set of formats, or process submitted data to provide more sophisticated data validation, querying, and manipulation?

I have provided no real answers to these questions here, and arriving at good answers will likely require much more thought and the input of those in the community who think that the idea is worth pursuing. However, I believe strongly that there is much to be gained from aggressively pursuing a

data-driven approach to building models of search algorithm performance. As a community, we generate a substantial amount of data, but currently, we aren't exploiting it very well.

That said, it is the feeling of the author that the right approach is probably to try and start small. To develop a bare-bones interchange format that captures very simple measurements of low-level, directly observed features as well as very high-level *"algorithm X found solutions of quality Y on problem Z"* performance metrics. This data should then be submitted to a centralized repository maintained by the community itself with rudimentary tool support for things like basic data validation and querying.

The specifics of how this should work probably cannot reliably be imposed from above. Rather, I hope that the argument I've made here will spark agreement within the community and entice like-minded researchers to undertake further discussions and come to a more considered understanding of the issues in a way that meets the goals set forth here.

8. ACKNOWLEDGMENTS

This type of speculative paper could not have been published in a more traditional form, and what value it does have lies almost entirely in the discussion it enables rather than the paper itself. The author would like to thank the organizers of the Understanding Problems workshop and the reviewers for encouraging "blue-sky" position papers such as this, as well as for their helpful comments and suggestions.

9. REFERENCES

[1] PISA: A platform and programming language independent interface for search algorithms. `http://www.tik.ee.ethz.ch/pisa/?page=pisa.php` (referenced in March, 2012).

[2] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler. PISA — a platform and programming language independent interface for search algorithms. In C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Evolutionary Multi-Criterion Optimization (EMO 2003)*, Lecture Notes in Computer Science, pages 494–508, Berlin, 2003. Springer.

[3] K. D. Boese. Cost versus distance in the traveling salesman problem. Technical Report TR-950018, University of California at Los Angeles, 1995.

[4] E. Burke, M. Gendreau, M. Hyde, G. Kendall, B. McCollum, G. Ochoa, A. Parkes, and S. Petrovic. The cross-domain heuristic search challenge - an international research competition. In X. Yao and C. A. C. Coello, editors, *Proceedings of Learning and Intelligent Optmization (LION5)*, volume 6683 of *Lecture Notes in Computer Science*, 2011.

[5] E. K. Burke, E. Hart, G. Kendall, J. Newall, P. Ross, and S. Schulenburg. *Hyper-heuristics: An emerging direction in modern search technology*, pages 457–474. Kluwer, 2003.

[6] S. Cahon, N. Melab, and E.-G. Talbi. Paradiseo: A framework for the reusable design of parallel and distributed metaheuristics. *Journal of Heuristics*, 10(3):357–380, 2004.

[7] N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking 2009: Experimental setup. Technical Report RR-6828, INRIA, 2009.

[8] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Pošík. Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009. In *Proceedings of the 2010 Genetic and Evolutionary Computation Conference (GECCO '10)*, 2010.

[9] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Technical Report RR-6829, INRIA, 2009.

[10] G. Holmes, A. Donkin, and I. Witten. Weka: A machine learning workbench. In *Proceedings of the Second Australia and New Zealand Conference on Intelligent Information Systems*, 1994.

[11] J. Knowles, L. Thiele, and E. Zitzler. A tutorial on the performance assessment of stochastic multiobjective optimizers. 214, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, July 2005.

[12] S. Künzli, S. Bleuler, L. Thiele, and E. Zitzler. *Application of Multi-Objective Evolutionary Algorithms*, chapter A Computer Engineering Benchmark Application for Multiobjective Optimizers, pages 269–294. World Scientific, December 2004.

[13] A. Liefooghe, M. Basseur, L. Jourdan, and E.-G. Talbi. Paradiseo-moeo: A framework for evolutionary multi-objective optimization. In *Proceedings of the Evolutionary Multiobjective Optimization Conference (EMO 2007)*, pages 386–400, 2007.

[14] S. Luke, L. Panait, G. Balan, S. Paus, Z. Skolicki, E. Popovici, J. Harrison, J. Bassett, R. Hubley, and A. Chircop. ECJ: A Java-based evolutionary computation research system, October 2007. http://cs.gmu.edu/∼eclab/projects/ecj/.

[15] O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, and G. Rudolph. Exploratory landscape analysis. In *Proceedings of the 2011 Genetic and Evolutionary Computation Conference (GECCO '11)*, pages 829–836, July 2011.

[16] C. L. Müller and I. F. Sbalzarini. Global characterization of the cec 2005 fitness landscapes using fitness-distance analysis. In *Proceedings of the 2011 EvoStar Conference, volume 6624 of Leture Notes in Computer Science*, pages 294–303. Springer, April 2011.

[17] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Tech. rep., Nanyang Technological University (NTU), May 2005.

[18] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y.-P. Chen, C.-M. Chen, and Z. Yang. Benchmark functions for the CEC '2008 special session and competition on large scale global optimization. Tech. Rep., University of Science and Technology of China, 2007.