# Finding Explanations for Multi-Objective Optimization (in Near-Linear Time)

Authors suppressed for blind review

Institution suppressed for blind review

**Abstract.** CT0 is very fast algorithm for finding trade-offs in multi-objective problems. A human inspecting those explanations can quickly infer changes that can improve their situation. This paper evaluates those explanations using data generated from (a) the POM3 model of agile selection of tasks; (b) four COCOMO-suite predictors for effort, months, defects and risk. CT0 ran orders of magnitude faster than standard optimizers (e.g. 3 seconds vs 150 seconds). Also, the generated explanations were as effective for optimization as the results of standard multi-objective optimizers (NSGA-II and SPEA2).
Based on this study, we recommend CT0 when some succinct summary has to be rapidly generated (e.g. in some interactive design meeting). CT0 could also be useful as post-processor to other optimizers (to generate succinct explanations of their conclusions) or as a optimizer to other optimizers (by constraining those other optimizers to only search the regions recommended by CT0).

**Keywords:** Software engineering, explanation, optimization, multi-objective.

*"If you cannot- in the long run- tell everyone what you have been doing, your doing has been worthless."*
– Erwin Schrodinger

## 1 Introduction

Explaining and the results of multi-objective optimization to a user can be problematic. A typical run of a multi-objective optimizer can process thousands to millions of examples. It is an overwhelming task for humans to certify the correctness of conclusions generated from so many results. Verrappa and Leiter warn that

> "..for industrial problems, these algorithms generate (many) solutions, which makes the tasks of understanding them and selecting one among them difficult and time consuming" [**?**].

Even if explanations are constrained to (say) just a few hundred examples taken from the Pareto frontier, this can still confuse the user. Valerdi notes that it can take days for panels of human experts to rigorously review even a few dozen examples [21]. For example, once had a client who disputed the results of our analysis. They demanded to audit the reasoning but when we delivered the of candidate solutions on the Pareto frontier, they were overwhelmed by the amount of information. Flustered, the client discounted the analysis and rejected our conclusions. From this experience, we learned that to better support decision making in SBSE, we must better explain SBSE results.

Other researchers have recognized the importance of explanation and is known to be a key factor in selecting algorithms. For example, in the field of machine learning, "each time one of our favorite approaches has been applied in industry, each time the comprehensibility of the results, though ill-defined, has been a decisive factor of choice over an approach by pure statistical means, or by neural networks." [3]. Analogous terms to explainability in that community are "comprehensibility", "interpretability" [1] or "understandability" [**?**].

In spite of the importance attributed to the subject, explanation has not been extensively investigated in the context of SBSE. One of the few papers that does is that of Veerappa and Lieter [23] who clustered examples from the Pareto frontier (examples generated from a goal graph representation of requirements for London ambulance services). In this approach, "instead of having to inspect a large number of individual solutions, (users) can look at a much smaller number of groups of related solutions, and focus their attention on the important characteristics of the group rather than the particularities of their individual solutions" [23].

While an insightful study, we are concerned with two issues about the Veerappa and Lieter study: (a) the complexity of clustering and (b) evaluation the value of generated recommendations. Veerappa and Lieter did not evaluate the effects of the recommendations that could be generated by users browsing their clusters. Also, their method could suffer from scalability issues since it a post-processor to a clustering algorithm. Clustering can be a slow process requiring say, $O(N^2)$ comparisons for each generation of the k-means algorithm [**?**].

Accordingly, in this paper, when:

1. Cluster using a near-linear time algorithm;
2. Then, when we infer some recommendations from the clusters, we impose those recommendation back onto the model inputs in order to generate more outputs.

As a starting point in this exploration of explanation, it is important to distinguish between the (1) problem of explaining the output of an multi-objective optimizer (discussed in this paper); from the more complex problem of (2) explaining how that output was generated. To put that in a more colloquial form, we seek to explain eggs, but not the chicken.

Next, a definition of "explanation" is required such that:

1. An explanation system can be designed;
2. It is possible to distinguish a "good" for a "bad" explanation.

In the SE literature, the general consensus in software engineering is that "good" explanations are succinct explanations.

Yet MOEAs fail on that criteria. XXX

Cognitive science theory argues that there is more to "explaining" something that just showing it succinctly. According to Kelly's personal construct theory (PCT) humans explain things via "constructs" that distinguish sets of examples [11]. So, for Kelly, human explanations are not about "things" in isolation but rather the *differences between groups of things*. In data mining, finding differences between things is called *contrast set learning* [17]. Previously, work on constrast learning for single goal SE problems found that very succinct contrast sets could be generated by

- Building a decision tree to separate the different outcomes;
- Identifying leaves containing desired outcome $X$ and undesired outcome $Y$;
- Querying that tree to find branches $B_x$ and $B_y$ that lead to $X, Y$.
- Computing $B_x - B_y$ which selects/rejects for desired/undesired outcomes.

In one spectacularly successful demonstration of this technique [14], it was found decision trees with 6,000 nodes had much superfluous information that was not useful for distinguishing desired and undesired outcomes. Using contrast set learning, the data that generated those decisions trees generated one contrast with only four variables in each (and when applied to test data, that contrast e was successful at pruning away all the undesired outcomes). Other studies with other data sets [15] confirmed the **the law of tiny constrasts**: *the minimal constrast set between things is usually much smaller than a complete description of those things.*

Other work by Leake characterized explaination as

Current MOEA algorithms are "instance-based methods" that return specific examples that perform "best" with respect to the multiple goals. The number of examples generated in this way can be overwhelming.

If a user wants to learn general principles from those examples, some secondary *explanation* process is required to group and generalize those examples. For example, Veerappa and Lieter [23] clustering examples from the Pareto frontier so users (at a minimum) need only browse the centroids of each clusters).

GAs flat vectors, not the trees explored by by ()say) Gouse et al.

Goals is performance just as good but explain better

One caveat before beginning: if the audience for the results of optimization are not human beings, then perhaps an explanation systems is not required. For example, Petke, Harman, Langdon, & Weimer [19] use evolutionary methods to rewrite code such that the new code executes faster. The audience for the rewritten code is a compiler. Such compilers do not argue or and ask questions about the code they are given to process. Hence, that rewrite system does not necessary need an explanation system. That said, a succinct and useful description of the difference between passing and failing runs of the rewrite system could be useful when (e.g.) a human is trying to debug that code rewrite system.

Yet another model of "explanation" not explored here is the "surprise modeling" approach recommended by Voinea&Tulea [2] and others including Horvitz [9]. In that approach, (a) some background knowledge (e.g. summaries of prior actions by users) is used to determine "normal" behavior; (b) users are only presented results that deviated from normal expectations. In analogous research, Koegh [12] argues that *time series discords* (infrequent sequential events in a times series) are a useful way to summarize reports from complex temporal streams. The premise of surprise modeling and reporting discords is that "rare events need to be explored". We do not dispute the importance of exploring such outliers. On the other hand, when forming policies for software projects, we need treatments that are well supported by the data. Hence, our contrast sets report changes in the data that, in our data, were *frequently* seen to lead to change.

Another potential issue with CT0 is correlation-vs-causation conflation. The issue here is that contrast sets will be useless if they report spurious correlations and not true causal effects. Proving that some effect is truly causal is a non-trivial task. The standard Hall criteria for causal effects [18] is so strict that, outside of highly controlled lab conditions, it rarely accepts that any effect is causal. Hence, in software engineering, when researchers talk of causality [4,7,10,24] they use Granger's "predictive causality"; i.e. causality is the ability of predicting values seen in the future from values seen in the past. Elsewhere, Granger causality has been adapted to data mining by organizing cross-validations such that the test sets contain data collected at a later time than the training sets [13]. In this paper, we adapt Granger casualty to search-based methods by testing recommendations learned from $M$ simulations on a subsequent round of $N$ new simulations. Those recommendations satisfy Granger causality when the subsequent round of $N$ simulations are changed in a manner predicted by the recommendations gleaned from the original $M$ simulations.

"Data farming" is a technique used extensively by the U.S. Military [**?**]. Data farming builds a "landscape" of output that can be analyzed for trends, anomalies, and insights in multiple parameter dimensions. In a recent review of search-based and data mining methids in SE, we found numerous examples of data farming [**?**, **?**, **?**, **?**, 5, 6, 8, 16, 20, 22].

In theory. Once a project manager can view their project on the landscape, they can use this visualization to determine

We come to this work after attending a recent seminar at the US Department of Defence's Software Engineering Institute (SEI), Pittsburgh, USA. That seminar reflected on how to best broadcast the lessons learned by SEI to a very broad audience.

In the $21^{st}$ century, it is now impossible to manually browse very large quantities of software project data. For example, as of October 2012, Mozilla Firefox had 800K reports on software projects. While it is now possible to automatically analyze such data with data miners, at some stage a group of business users will have to convene to *interpret the results* (e.g., to decide if it is wise to deploy the results as a defect reduction method within an organization). These business users are now demanding that data mining tools be augmented with tools to support business-level interpretation of that data. For example,

at a recent panel on software analytics at ICSE'12,
industrial practitioners lamented the state of the art in data mining
and software engineering [**?**]. Panelists commented that
"prediction is all well and good, but what about decision
making?". That is, these panelists are more interested in the interpretations
that follow the mining, rather than just the mining.

You are strongly encouraged to use LaTeX $2_\varepsilon$ for the preparation of your camera-ready manuscript together with the corresponding Springer class file `llncs.cls`. Only if you use LaTeX $2_\varepsilon$ can hyperlinks be generated in the online version of your manuscript.

The LaTeX source of this instruction file for LaTeX users may be used as a template. This is located in the "authors" subdirectory in `ftp://ftp.springer.de/pub/tex/latex/llncs/latex2e/instruct/` and entitled `typeinst.tex`. Kindly send the final and checked source and PDF files of your paper to the Contact Volume Editor. This is usually one of the organizers of the conference. You should make sure that the LaTeX and the PDF files are identical and correct and that only one version of your paper is sent. It is not possible to update files at a later stage. Please note that we do not need the printed paper.

We would like to draw your attention to the fact that it is not possible to modify a paper in any way, once it has been published. This applies to both the printed book and the online version of the publication. Every detail, including the order of the names of the authors, should be checked before the paper is sent to the Volume Editors.

### 1.1 Checking the PDF File

Kindly assure that the Contact Volume Editor is given the name and email address of the contact author for your paper. The Contact Volume Editor uses these details to compile a list for our production department at SPS in India. Once the files have been worked upon, SPS sends a copy of the final pdf of each paper to its contact author. The contact author is asked to check through the final pdf to make sure that no errors have crept in during the transfer or preparation of the files. This should not be seen as an opportunity to update or copyedit the papers, which is not possible due to time constraints. Only errors introduced during the preparation of the files will be corrected.

This round of checking takes place about two weeks after the files have been sent to the Editorial by the Contact Volume Editor, i.e., roughly seven weeks before the start of the conference for conference proceedings, or seven weeks before the volume leaves the printer's, for post-proceedings. If SPS does not receive a reply from a particular contact author, within the timeframe given, then it is presumed that the author has found no errors in the paper. The tight publication schedule of LNCS does not allow SPS to send reminders or search for alternative email addresses on the Internet.

In some cases, it is the Contact Volume Editor that checks all the final pdfs. In such cases, the authors are not involved in the checking phase.

### 1.2 Additional Information Required by the Volume Editor

If you have more than one surname, please make sure that the Volume Editor knows how you are to be listed in the author index.

### 1.3 Copyright Forms

The copyright form may be downloaded from the "For Authors" (Information for LNCS Authors) section of the LNCS Website: `www.springer.com/lncs`. Please send your signed copyright form to the Contact Volume Editor, either as a scanned pdf or by fax or by courier. One author may sign on behalf of all of the other authors of a particular paper. Digital signatures are acceptable.

## 2 Related Work

Sayyad

GALE

## 3 Paper Preparation

Springer provides you with a complete integrated LaTeX document class (`llncs.cls`) for multi-author books such as those in the LNCS series. Papers not complying with the LNCS style will be reformatted. This can lead to an increase in the overall number of pages. We would therefore urge you not to squash your paper.

Please always cancel any superfluous definitions that are not actually used in your text. If you do not, these may conflict with the definitions of the macro package, causing changes in the structure of the text and leading to numerous mistakes in the proofs.

If you wonder what LaTeX is and where it can be obtained, see the "*LaTeX project site*" (`http://www.latex-project.org`) and especially the webpage "*How to get it*" (`http://www.latex-project.org/ftp.html`) respectively.

When you use LaTeX together with our document class file, `llncs.cls`, your text is typeset automatically in Computer Modern Roman (CM) fonts. Please do *not* change the preset fonts. If you have to use fonts other than the preset fonts, kindly submit these with your files.

Please use the commands `\label` and `\ref` for cross-references and the commands `\bibitem` and `\cite` for references to the bibliography, to enable us to create hyperlinks at these places.

For preparing your figures electronically and integrating them into your source file we recommend using the standard LaTeX `graphics` or `graphicx` package. These provide the `\includegraphics` command. In general, please refrain from using the `\special` command.

Remember to submit any further style files and fonts you have used together with your source files.

**Headings.** Headings should be capitalized (i.e., nouns, verbs, and all other words except articles, prepositions, and conjunctions should be set with an initial capital) and should, with the exception of the title, be aligned to the left. Words joined by a hyphen are subject to a special rule. If the first word can stand alone, the second word should be capitalized.

Here are some examples of headings: "Criteria to Disprove Context-Freeness of Collage Language", "On Correcting the Intrusion of Tracing Non-deterministic Programs by Software", "A User-Friendly and Extendable Data Distribution System", "Multi-flip Networks: Parallelizing GenSAT", "Self-determinations of Man".

**Lemmas, Propositions, and Theorems.** The numbers accorded to lemmas, propositions, and theorems, etc. should appear in consecutive order, starting with Lemma 1, and not, for example, with Lemma 11.

### 3.1 Figures

For LaTeX users, we recommend using the *graphics* or *graphicx* package and the `\includegraphics` command.

Please check that the lines in line drawings are not interrupted and are of a constant width. Grids and details within the figures must be clearly legible and may not be written one on top of the other. Line drawings should have a resolution of at least 800 dpi (preferably 1200 dpi). The lettering in figures should have a height of 2 mm (10-point type). Figures should be numbered and should have a caption which should always be positioned *under* the figures, in contrast to the caption belonging to a table, which should always appear *above* the table; this is simply achieved as matter of sequence in your source.

Please center the figures or your tabular material by using the `\centering` declaration. Short captions are centered by default between the margins and typeset in 9-point type (Fig. **??** shows an example). The distance between text and figure is preset to be about 8 mm, the distance between figure and caption about 6 mm.

To ensure that the reproduction of your illustrations is of a reasonable quality, we advise against the use of shading. The contrast should be as pronounced as possible.

If screenshots are necessary, please make sure that you are happy with the print quality before you send the files.

Please define figures (and tables) as floating objects. Please avoid using optional location parameters like "`[h]`" for "here".

**Remark 1.** In the printed volumes, illustrations are generally black and white (halftones), and only in exceptional cases, and if the author is prepared to cover the extra cost for color reproduction, are colored pictures accepted. Colored pictures are welcome in the electronic version free of charge. If you send colored figures that are to be printed in black and white, please make sure that they really are legible in black and white. Some colors as well as the contrast of converted colors show up very poorly when printed in black and white.

### 3.2 Formulas

Displayed equations or formulas are centered and set on a separate line (with an extra line or halfline space above and below). Displayed expressions should be numbered for reference. The numbers should be consecutive within each section or within the contribution, with numbers enclosed in parentheses and set on the right margin – which is the default if you use the *equation* environment, e.g.,

$$\psi(u) = \int_o^T \left[\frac{1}{2}\left(\Lambda_o^{-1}u, u\right) + N^*(-u)\right] dt \; . \tag{1}$$

Equations should be punctuated in the same way as ordinary text but with a small space before the end punctuation mark.

### 3.3 Footnotes

The superscript numeral used to refer to a footnote appears in the text either directly after the word to be discussed or – in relation to a phrase or a sentence – following the punctuation sign (comma, semicolon, or period). Footnotes should appear at the bottom of the normal text area, with a line of about 2 cm set immediately above them.[1]

### 3.4 Program Code

Program listings or program commands in the text are normally set in typewriter font, e.g., CMTT10 or Courier.

*Example of a Computer Program*

```
program Inflation (Output)
  {Assuming annual inflation rates of 7%, 8%, and 10%,...
   years};
   const
     MaxYears = 10;
   var
     Year: 0..MaxYears;
     Factor1, Factor2, Factor3: Real;
   begin
     Year := 0;
     Factor1 := 1.0; Factor2 := 1.0; Factor3 := 1.0;
     WriteLn('Year  7% 8% 10%'); WriteLn;
     repeat
       Year := Year + 1;
       Factor1 := Factor1 * 1.07;
       Factor2 := Factor2 * 1.08;
       Factor3 := Factor3 * 1.10;
       WriteLn(Year:5,Factor1:7:3,Factor2:7:3,Factor3:7:3)
     until Year = MaxYears
end.
```

(Example from Jensen K., Wirth N. (1991) Pascal user manual and report. Springer, New York)

[1] The footnote numeral is set flush left and the text follows with the usual word spacing.

### 3.5 Citations

For citations in the text please use square brackets and consecutive numbers: [**?**], [**?**], [**?**] – provided automatically by LaTeX's `\cite` ... `\bibitem` mechanism.

### 3.6 Page Numbering and Running Heads

There is no need to include page numbers. If your paper title is too long to serve as a running head, it will be shortened. Your suggestion as to how to shorten it would be most welcome.

## 4 LNCS Online

The online version of the volume will be available in LNCS Online. Members of institutes subscribing to the Lecture Notes in Computer Science series have access to all the pdfs of all the online publications. Non-subscribers can only read as far as the abstracts. If they try to go beyond this point, they are automatically asked, whether they would like to order the pdf, and are given instructions as to how to do so.

Please note that, if your email address is given in your paper, it will also be included in the meta data of the online version.

## 5 BibTeX Entries

The correct BibTeX entries for the Lecture Notes in Computer Science volumes can be found at the following Website shortly after the publication of the book: `http://www.informatik.uni-trier.de/~ley/db/journals/lncs.html`

**Acknowledgments.** The heading should be treated as a subsubsection heading and should not be assigned a number.

## 6 The References Section

In order to permit cross referencing within LNCS-Online, and eventually between different publishers and their online databases, LNCS will, from now on, be standardizing the format of the references. This new feature will increase the visibility of publications and facilitate academic research considerably. Please base your references on the examples below. References that don't adhere to this style will be reformatted by Springer. You should therefore check your references thoroughly when you receive the final pdf of your paper. The reference section must be complete. You may not omit references. Instructions as to where to find a fuller version of the references are not permissible.

We only accept references written using the latin alphabet. If the title of the book you are referring to is in Russian or Chinese, then please write (in Russian) or (in Chinese) at the end of the transcript or translation of the title.

The following section shows a sample reference list with entries for journal articles [**?**], an LNCS chapter [**?**], a book [**?**], proceedings without editors [**?**] and [**?**], as well as a URL [**?**]. Please note that proceedings published in LNCS are not cited with their full titles, but with their acronyms!

## References

1.
2. Visual data mining and analysis of software repositories, 2007.
3. I. Askira-Gelman. Knowledge discovery: Comprehensibility of the results. In *Hawaii International Conference on System Sciences*, 1998.

4. P. Bhattacharya, M. Iliofotou, I. Neamtiu, and M. Faloutsos. Graph-based analysis and prediction for software evolution. In *Proceedings of the 34th International Conference on Software Engineering*, ICSE '12, pages 419–429, Piscataway, NJ, USA, 2012. IEEE Press.
5. E. Chiang and T. Menzies. Simulations for very early lifecycle quality evaluations. *Software Process: Improvement and Practice*, 7(3-4):141–159, 2003. Available from `http://menzies.us/pdf/03spip.pdf`.
6. L. Chung, B. Nixon, E. Yu, and J. Mylopoulos. *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers, 2000.
7. C. Couto, M. Valente, P. Pires, A. Hora, N. Anquetil, and R. Bigonha. Bugmaps-granger: a tool for visualizing and predicting bugs using granger causality tests. *Journal of Software Engineering Research and Development*, 2, 1024.
8. W. Heaven and E. Letier. Simulating and optimising design decisions in quantitative goal models. In *Requirements Engineering Conference (RE), 2011 19th IEEE International*, pages 79–88, 2011.
9. E. Horvitz, J. Apacible, R. Sarin, and L. Liao. Prediction, expectation, and surprise: Methods, designs, and study of a deployed traffic forecasting service. In *UAI'05*, pages 275–283, 2005.
10. B. Huberman, D. Romero, and F. Wu. Crowdsourcing, attention and productivity. *Journal of Information Science*, 35(6):758–765, December 2009.
11. G. Kelly. *The Psychology of Persona] Constructs. Volume 1: A Theory of Personality. Volume 2: Clinical Diagnosis and Psychotherapy*. Norton, 1955.
12. E. Keogh, J. Lin, and A. Fu. Hot sax: Efficiently finding the most unusual time series subsequence. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, ICDM '05, pages 226–233, Washington, DC, USA, 2005. IEEE Computer Society.
13. M. Lumpe, R. Vasa, T. Menzies, R. Rush, and R. Turhan. Learning better inspection optimization policies. *International Journal of Software Engineering and Knowledge Engineering*, 21(45):725–753, 2011.
14. T. Menzies and Y. Hu. Data mining for very busy people. November 2003. Available from `http://menzies.us/pdf/03tar2.pdf`.
15. T. Menzies and Y. Hu. Just enough learning (of association rules): The TAR2 treatment learner. In *Artificial Intelligence Review*, 2007. Available from `http://menzies.us/pdf/07tar2.pdf`.
16. I. Myrtveit, E. Stensrud, and M. Shepperd. Reliability and validity in comparative studies of software prediction models. *IEEE Trans. Softw. Eng.*, 31(5):380–391, May 2005.
17. P. K. Novak, N. Lavrač, and G. I. Webb. Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *J. Mach. Learn. Res.*, 10:377–403, June 2009.
18. L. Paul and N. Hall. *Causation : A Users Guide*. Oxford University Press, 2013.
19. J. Petke, M. Harman, W. Langdon, and W. Weimer. Using genetic improvement & code transplants to specialise a c++ program to a problem class. In *European Conference on Genetic Programming (EuroGP)*, 2014.
20. M. Shepperd and G. F. Kadoda. Comparing software prediction techniques using simulation. *IEEE Trans. Software Eng*, 27(11):1014–1022, 2001.
21. R. Valerdi. Convergence of expert opinion via the wideband delphi method: An application in cost estimation models. In *Incose International Symposium, Denver, USA*, 2011. Available from http://goo.gl/Zo9HT.
22. A. van Lamsweerde and E. Letier. Integrating obstacles in goal-driven requirements engineering. In *Proceedings of the 20th International Conference on Software Engineering*, pages 53–62. IEEE Computer Society Press, 1998. Available from `http://citeseer.nj.nec.com/vanlamsweerde98integrating.html`.
23. V. Veerappa and E. Letier. Understanding clusters of optimal solutions in multi-objective decision problems. In *Proceedings of the 2011 IEEE 19th International Requirements Engineering Conference*, RE '11, pages 89–98, Washington, DC, USA, 2011. IEEE Computer Society.

24. P. Zheng, Y. Zhou, M. Lyu, and Y. Qi. Granger causality-aware prediction and diagnosis of software degradation. In *Services Computing (SCC), 2014 IEEE International Conference on*, pages 528–535, June 2014.