Evo_Indent Interactive Evolution of GNU indent Options

W. B. Langdon

Department of Computer Science, CREST centre, King's College, London, WC2R 2LS, UK

ABSTRACT

Evo_Indent `http://www.dcs.kcl.ac.uk/staff/W.Langdon/evo_indent/` is a PHP web server based user driven genetic algorithm which finds good C code layouts generated by GNU indent. Either the refactored source can be used or the user's preferred indent command options can be saved and re-used to pretty print other program text files.

Categories and Subject Descriptors

D.2.6 [**Software Engineering**]: Programming Environments—*Interactive environments*

General Terms

Human Factors

Keywords

Evolutionary Algorithms, (1+3)-ES, mutation, chromosome reordering, user driven fitness, personalised software, customised interface, prettyprint, understandability, comprehension, refactoring, SBSE

1. INTRODUCTION

Typically low level machine code, assembly languages and older high level languages, such as Fortran, have fixed layout rules. In contrast, most modern programming languages (C, C++, PHP, Java, etc.) are layout independent. However good source code layout, particularly indentation, aids comprehension. Therefore there are many recommendations for good program layout. Indeed some commercial companies include program source layout as part of their inhouse mandatory styles. However many C programmers are free to choose their own programming style. Indeed tools like emacs can be customised to suit each individual.

The GNU indent program has detailed knowledge of C. It uses this to layout users' source files. In particular it uses indentation to highlight the block structure of the code. indent is freely available and ported to many platforms and operating systems. In particular, it runs both on SUN webservers and PCs. To accommodate user preferences, it supports 60 options, many of which require a parameter value.

Although indent provides sensible defaults for its options, deciding on the best combination for a particular purpose is difficult. In practice, most users will either accept one of the default setting or resort to trying out an individual indent option on the source file and seeing what effect it has. Although indent makes backup copies of the files it changes, experimenting with command lines is both tedious and error prone. However should the user alight on a nice set of options, these can be saved in an `.indent.pro` file, thus enabling their convenient reuse on other C source files.

Evo_Indent automates this experimentation, in a safe, principled, rapid and user friendly manner. (Cf. Fig 1.) It appears to be the first application of interactive evolution to software refactoring.

The next section describes how Evo_Indent works, whilst Section 3 describes its benefits. Technical details of the artificial evolutionary system including genetic representation and mutation are delayed until Section 4. Section 5 considers the implication of our choice of a web based design and considers local alternatives. We conclude in Section 6.

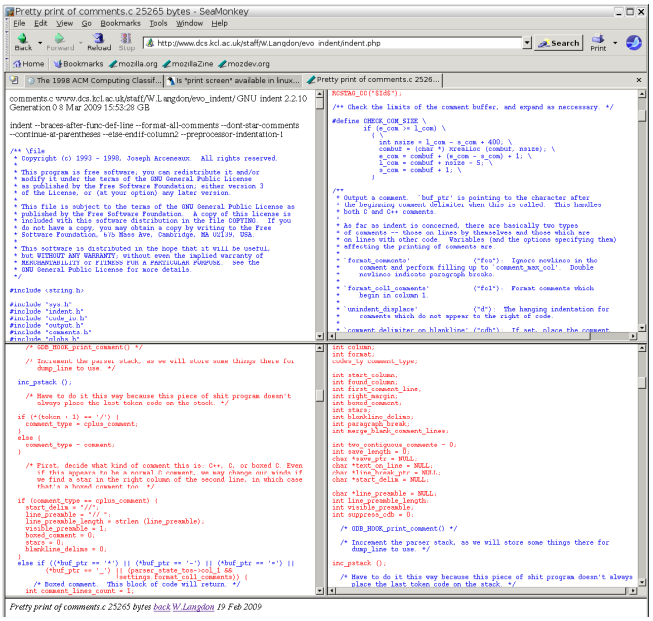

Figure 1: Evo_Indent main operational screen. Three new mutants and their parent are displayed and the user is invited to choose their preferred layout. Phenotypic differences between parent and offspring are highlighted in red. The indent command line is displayed at the top of each scrollable frame. The frames are automatically scrolled to the first difference.

GECCO'09, Late Breaking paper July 8–12, 2009, Montreal, Canada
ACM 978-1-60558-505-5/09/07.

2. INTERACTIVELY EVOLVING USER PREFERRED GNU INDENT OPTIONS

Evo_Indent consists mainly of two screens. The first is a form written in traditional HTML which allows the user to upload their C source or library (`.h`) file.

The second screen uses PHP to generate HTML dynamically, cf. Fig 1. This is where the evolution takes place. HTML frames are supported by most browsers and these are used to partition the browser window into four main areas. These four are used as scrollable windows onto the four current evolved layouts of the user's file. A small fifth frame at the bottom is used to display a little status information.

2.1 Starting Evolution with a Random Population

After the user's file is uploaded a population of four random indent command lines is created. Each is assigned to one of the four main frames. For each, indent is run with the random command line and its processed output is sent to the frame.

indent's output and the original file are compared and differences highlighted in red. This makes changes introduced by indent more visible and so reduces user fatigue, cf. Section 3.4. The source code is also scanned for characters which the browser might interpret as HTML commands and these are replaced by HTML escape codes.

The whole source code is wrapped in an HTML hyperlink. This is displayed by the browser in its usual way and so is immediately recognisable as a hyper link to the user. When the user activates the link on one of the four frames, the browser signals this to the webserver, which directs the signal to Evo_Indent's PHP script.

2.2 Creating a New Generation

When the user's signal arrives, the chosen member of the population is redisplayed in the same frame without its earlier red highlighting. Its chromosome is mutated three times to create three children. These are displayed in the other three screens. In the new population, red is used to highlight differences from the parent.

2.3 Ending the (1+3)-ES

Evolution continues for as long as the user wishes. At any point the user may stop and, using their browser, either save the current command line or indent's output or both.

It is even possible, using the browser's back button, to recover a previous population and proceed with a new evolutionary path. (See also Section 3.4.)

3. ADVANTAGES OF EVO_INDENT

3.1 Safety

A copy of the user's file is uploaded to our web server. All experimental operations take place in the controlled web server environment. Only when a satisfactory layout has been achieved need the user transfer the new version back to their PC. The user may also save the evolved indent command options to their own `.indent.pro` file. `.indent.pro` makes it easy to run indent on the PC with the user's individual evolved preferences. Also `.indent.pro` files can be shared, enabling customised styles to be shared by a group of C programmers.

3.2 Convenience

The web environment uses an evolutionary algorithm to generate new indent options in a principled way. The user is presented not with the command options but with their effect on the user's file. Changes in layout are highlighted.

Evo_Indent allows the user to choose between up to four options simultaneously. Only a single mouse click is need to choose an option.

All the management of intermediate files and finding differences between options is done automatically on the server without the user being concerned.

3.3 Speed

Operation via a remote server might be expected to impose delays. However with high speed PCs, modern browsers and typical high speed network connection, in practice the time to up load files and transfer screens of information to the user's browser are barely noticeable.

3.4 Reduced User Fatigue

A common problem with interactive evolution is user fatigue. That is, typically the user of the system quickly loses interest in the system.

By allowing the user to upload their own files and control how they are laid out we have given them a stake in the system. After all it is their code that is being re-factored for their benefit.

The user indicates their choice by simply clicking on it. This is done in exactly the same way that the user clicks on links. Thus they will be entirely familiar with the mechanism that their browser provides. Almost any point in the whole screen can be clicked on. No time need be wasted accurately positioning the mouse.

Whilst obviously dependent on communications and server loading, typically all four new frames are presented to the user in less than a second of their mouse click.

Most of the time is spent by the user deciding which of the options to choose. We position each new layout at the start of changes from the previous generation. Nevertheless, typically users will want to scroll through the changes before deciding. This is the most time consuming part.

The evolution can be stopped at any point and its results harvested by the user.

If the user changes their mind, their browser's forward and back buttons allow them to step backwards and forwards through the evolution. However only one active path through the evolution is supported. If the user makes a new choice a new set of random mutations are created. Whilst back tracking to earlier populations is still possible, the previous forward steps become inaccessible.

4. MUTATING INDENT'S COMMAND LINE

4.1 indent's Command Options

Excluding help, version identification and C types, from the user's point of view indent has 60 command line options.

There are four settings options: `gnu` (default) `linux`, `kernighan-and-ritchie-style` and `berkeley-style` which provide suitable defaults and thus effect multiple other options. There is also one settings option which disables an integer option. These five options can either be asserted or not present.

There are eighteen command options which have integer values. Again they can only be in use or not. (If not their default value is used).

The remaining command options are Boolean and can be either explicitly set, explicitly turned off or omitted. Again if they are not present, their default value is in use.

indent processes its command line in order. Thus, particularly for the five settings options, where an option appears in the command line is important. However there is no need for an option to appear more than once, since then it is sure to nullify the effect of its previous occurrence.

4.2 Evo_Indent's Chromosome

The chromosome consists of an ordered list of genes. There is exactly one gene for each of the 60 indent command options. For a settings option, the gene says if it is on (+) or not present. For a Boolean, the gene says if it is on (+), negated (-) or neither. For an integer, the gene holds its value and also says if it is on (+) or the default value is to be used. Note an integer gene continues to hold the option's value even if the option is not asserted (and therefore indent is using its default value).

4.3 Creating a Random Population

The initial four chromosomes are created at random. However since the user's time is so precious they are checked to try and ensure that they represent different command options. In practice detailed knowledge of the operation of indent would have to be used to make this totally water tight. It is usually sufficient to check only that the active options are different. (A possible future enhancement would be to check that the output generated by indent, i.e. the four phenotypes, are different.)

4.4 Creating a Random Chromosome

The five settings genes which control other genes (cf. Section 4.1 are placed in a random order then added to the start of the chromosome. For each there is a 10% chance that it will be asserted (+). (Thus there is 41% chance that none of the five will be used.)

Similarly the other 55 genes are placed in random order and for each there is a 10% chance that it will be asserted and a 90% chance it will take its default value. Active Booleans are equally likely to be on (+) as off (-). The value of active integers is given by mutating their default value.

4.5 Mutating Integer Values

If the current value (i) is less than ten a new value is chosen equally from $i-2, i-1, i+1, i+2$.

If i greater than 10, a new integer is chosen from the ranges $i \times 0.833333 \ldots i \times 0.990099$ or $i \times 1.01 \ldots i \times 1.20$.

4.6 Mutating Chromosomes

As with the initial population (Section 4.3) we try to ensure each new child command line is different from its parent.

To mutate a chromosome we start by randomly re-ordering it. One might suggest mutations should be small changes and choosing a random permutation uniformly is not a small change. However typically command options do not interfere with each other so many gene swaps have no effect. Thus a total random reordering has less effect than might at first be expected.

The new chromosome is processed in its new order. For each gene there is 10% chance of it being mutated.

If an integer is currently asserted it can either (with equal likelihood) be disabled or its value can be mutated (using the algorithm described in Section 4.5). If the integer option is not asserted, then it will be. Also, with equal likelihood: 1) it will retain its current value. 2) its current value will be mutated (cf. Section 4.5). Or 3) its value will be replaced by mutating its default value. (I.e. a new value will be generated in the same way as was used to create the initial population.)

If a Boolean is asserted (+) it has a 50% chance of becoming negated (-) and a 50% chance of being removed and thus reverting to its default value. Similarly negated (-) and unasserted Booleans have equal chances of being changed to one of their two other possibilities.

If a settings gene is chosen for mutation, if it is active it will be disabled. Conversely if it not present it will be enabled. Note these are potentially large changes.

Since the initial generation starts with few genes asserted, over the generations the number of asserted options tends to increase.

4.7 Displaying the Command Line

Chromosomes are always converted to command lines one gene at a time in the same order as they occur in the chromosome.

Almost all of indent's command line options have both a short name and at least one long form. The short names consist of two or three letters and so are rather cryptic. Therefore when displaying an Evo_Indent chromosome the longer indent name is used.

5. DISCUSSION

The current system has the advantages of being fast and operational on common web browsers. However this means it does not support fisheye views to enable the user to rapidly concentrate upon particular parts of their programs or specialised printing options like kerning. These would require specialised browser support and tend to be computationally demanding.

While interactive operations should in principle be handled on the user's PC rather than on a web server, the heavy use of files and system utilities make it very difficult to implement in the user's browser and very difficult to obtain portability between browsers. Nevertheless good performance can be obtained despite the inevitable network overhead.

Unlike, emacs, indent is command line based and designed to reformat files in a single operation. This makes its operation under PHP on a web server viable. It would be nice to target either emacs or eclipse but successful interactive evolution would probably be difficult requiring tight integration between them and the evolutionary environment and so be infeasible in a web server environment.

6. CONCLUSIONS

Evo_Indent `http://www.dcs.kcl.ac.uk/staff/W.Langdon/evo_indent/` allows the user to navigate the huge space ($\gg 2^{60}$) of GNU indent command line options in a controlled, rapid and user friendly fashion.

Acknowledgment

I would like to thank Youssef Hassoun and Jian Ren.