IPOMOEA:Intended Package Orientation using Multi-Objective Evolutionary Algorithm in R

M. A. EL-dosuky[1]

[1]**Computer Science Department, Faculty of Computer and Info, Mansoura University, 35516 Mansoura, Egypt,**

Corresponding author:
M. A. EL-dosuky[1]

Email address: mouh_sal_010@mans.edu.eg

ABSTRACT

Programmers' lack of familiarity with what is available in packages may prompt them to reinvent the wheel. This is generally the case in any programming language, but it is a matter of madness with a language described as difficult even by professionals supporting it such as R. R Cookbook says: "But R can be frustrating. It's not obvious how to accomplish many tasks, even simple ones." IPOMOEA is a code that has been written to mitigate this problem. It helps R language developers determine how to perform a specific task, by automating the search in R site for all packages that are likely to contribute to the task implementation. After that, IPOMOEA determines a partial set of results to be the intended package using multi-objective evolutionary algorithm NSGA-II. Not only does it specify the intended package, but also it helps orient programmers and manage packages.

1 INTRODUCTION

> It goes without saying that R (R Core Team, 2017) is a powerful language. It is an effective tool for statistical processing. Furthermore, it has been armed with thousands of packages thanks to the efforts of armies of developers for more than twenty years now (Smith (2017)). However, the programmers' lack of familiarity with what is available in those packages may prompt them to reinvent the wheel. This is generally the case in any programming language, but it is a matter of madness with a language described as difficult even by professionals supporting it. Let's quote what is stated in the preface of R Cookbook by Teetor (2011), in which the author says: "But R can be frustrating. It's not obvious how to accomplish many tasks, even simple ones."

IPOMOEA is a code that has been written to mitigate this problem. It helps R language developers determine how to perform a specific task, by automating the search in R site for all packages that are likely to contribute to the task implementation. After that, IPOMOEA determines a partial set of results to be the intended package using multi-objective evolutionary algorithm (MOEA) NSGA-II Deb et al. (2002). Not only does it specify the intended package, but also it helps orient programmers and manage packages.

2 MODEL

IPOMOEA front-end (Figure 1) has two steps. The developer should provide the keyword. The intended package could be any one that implements a function named after that keyword. IPOMOEA then indulges in automatic search of the R site, as if RSiteSearch() is called. This is possible thanks to Namazu search engine Namazu Project (2011). Then a file, named after the entered keyword, shall be created for storing the results locally. Namazu ranks the result set based on significance using a smart algorithm Inoue et al. (2005).

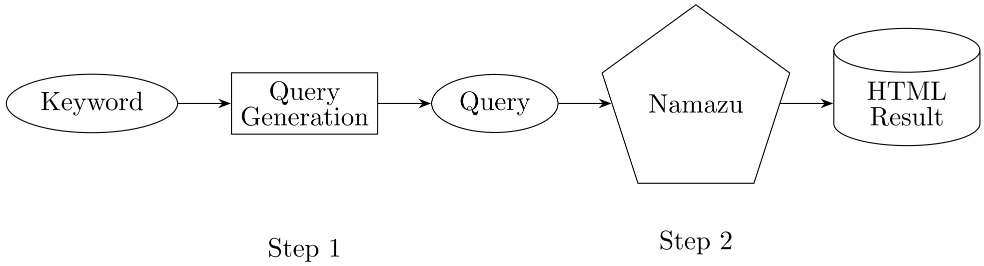

Figure 1. IPOMOEA Front-end

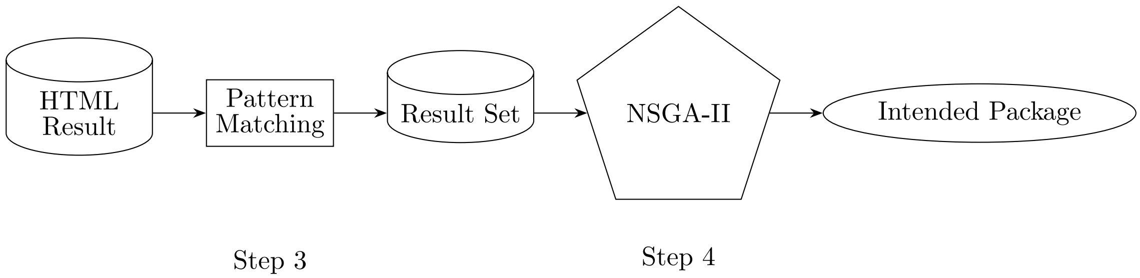

Figure 2. IPOMOEA Back-end

IPOMOEA back-end (Figure 2) has two steps too. IPOMOEA harvests all packages, functions within, and their ranks from HTML results provided from Namazu. Namazu generates results in a fixed format, allowing for parsing them using regular expressionsMcNaughton and Yamada (1960). The intended package is then reached based on MOEA NSGA-II Deb et al. (2002).

IPOMOEA is formalized to be a minimization problem over a set of features:

$$minimize \quad IPOMOEA_{Objective}(Features) \tag{1}$$

subject to

$$IPOMOEA_{Constraint}(Features) = \left(\sum_{f \in Features} f \right) - \theta \geq 0 \tag{2}$$

θ is not just a mere threshold, it is a grouping and/or discriminating factor. As there could be many candidate intended packages, grouping or discriminating them is necessary.

3 IMPLEMENTATION

IPOMOEA code is publicly available **?**. First of all, it makes sure the MCO package (Mersmann et al., 2014) is available. It is the package that provides nsga2() function, implementing NSGA-II Deb et al. (2002). MCO will be installed if it is not already installed.

```
if (!require("mco")) install.packages("mco")
```

Then the developer should provide the keyword. It happened to be

```
%

\begin{lstlisting}
ipomoea_keyword <- 'dickey-fuller'
```

IPOMOEA then indulges in automatic search of the R site, as if RSiteSearch() is called. This is possible thanks to Namazu search engine Namazu Project (2011). Then a file, named after the entered keyword, shall be created for storing the results locally.

```
ipomoea_search <- readLines(paste('http://search.r-project.org/cgi-
    bin/
    namazu.cgi?query=', ipomoea_keyword))
ipomoea_file <- paste(".ipomoea.",ipomoea_keyword,".html")
```

Namazu generates results in a fixed format, allowing for parsing them using regular expressionsMcNaughton and Yamada (1960). For example, the cardinality of result set can be extracted using the following pattern.

```
ipomoea_total_pattern = 'Total <!-- HIT -->([^<]*)<!-- HIT -->'
```

Using grep(), gregexpr(), and gregexpr(), IPOMOEA extracts the cardinality of result set, which happened to be:

48

Result set per se can be extracted using:

```
ipomoea_result_set_pattern = '<dl>(.*)</dl>'
```

Namazu ranks the result set based on significance using a smart algorithm Inoue et al. (2005). To harvest all packages, functions within, and their ranks from the results:

```
ipomoea_packs <- c()
ipomoea_funcs <- c()
ipomoea_ranks <- c()

for (ipomoea_line in ipomoea_resultLines){
    if(startsWith(ipomoea_line, ipomoea_dt)){
        ipomoea_pack <- get_lib_func_score(ipomoea_package_pattern,
            ipomoea_line)
        ipomoea_func <- get_lib_func_score(ipomoea_function_pattern,
            ipomoea_line)
        ipomoea_rank <- get_lib_func_score(ipomoea_rank_pattern,
            ipomoea_line))

        if(!startsWith(ipomoea_func, "00")){
            ipomoea_func_fullname<- paste(ipomoea_pack, "::", ipomoea
                _func, sep='')

            ipomoea_packs <- c(ipomoea_packs, ipomoea_pack)
            ipomoea_funcs <- c(ipomoea_funcs, ipomoea_func_fullname)
            ipomoea_ranks <- c(ipomoea_ranks, as.numeric(ipomoea_rank
                ))
        }
    }
}
```

All the magic is in the meticulously tweaked patterns. The pattern to extract ranks is:

```
ipomoea_rank_pattern <- "<dt>[^(]*[(]score: (.*)[)]</dt>"
```

Top results and their associated scores/ranks are depicted in Figure 3.

To convert the problem into a minimization one, all ranks should be subtracted from the maximum one:

```
ipomoea_max_rank <- max(ipomoea_ranks)
ipomoea_min_rank <- min(ipomoea_ranks)
ipomoea_inverted_ranks <- ipomoea_max_rank - ipomoea_ranks
```


PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.27656v1 | CC BY 4.0 Open Access | rec: 16 Apr 2019, publ: 16 Apr 2019

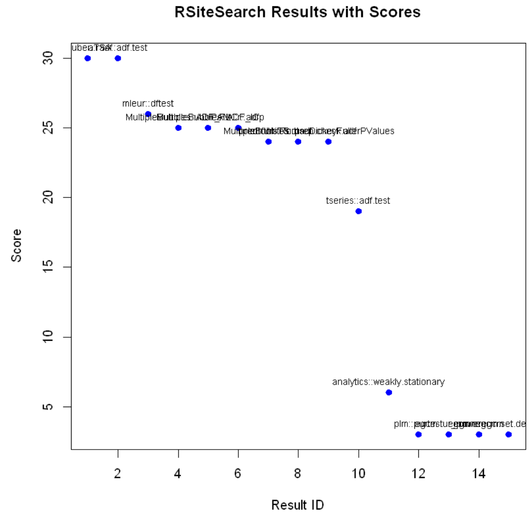

Figure 3. Top results and their associated scores/ranks.

Then the implementation of formalization (Equation 1 and Equation 2). For a matter of simplification, let us limit the feature set to just 2 of them, namely the ID, and the rank/score.

```
ipomoea_objective <- function(ipomoea_features) {
    if(is.wholenumber(ipomoea_features[1])) { ipomoea_features  }
    else { c(as.double(floor(ipomoea_features[1])+1), ipomoea_features[2]) }
}
ipomoea_constraint <- function(ipomoea_features) {
    sum(ipomoea_features) - ipomoea_theta
}
```

Then, the time is ripe to call MCO package (Mersmann et al., 2014) that provides nsga2() function Deb et al. (2002), setting its parameters, and calling it.

```
library(mco)

ipomoea_theta <- 5

ipomoea_generations<-100
ipomoea_dimension_1_lower_bound <-1
ipomoea_dimension_1_upper_bound <- length(ipomoea_ranks)
ipomoea_dimension_2_lower_bound <- 0
ipomoea_dimension_2_upper_bound <- max(ipomoea_inverted_ranks)

ipomoea_intended_packs <- nsga2(ipomoea_objective, 2, 2,
    generations= ipomoea_generations,
    lower.bounds=c(ipomoea_dimension_1_lower_bound, ipomoea_dimension_2_lower_bound),
    upper.bounds=c(ipomoea_dimension_1_upper_bound, ipomoea_dimension_2_upper_bound),
    constraints=ipomoea_constraint,
    cdim=1)
```

Candidate intended package(s) at $\theta = 5$, $\theta = 7$, and the extreme case at $\theta = |ResultSet|$ are shown in Figure 4, Figure 5, and Figure 6 respectively.

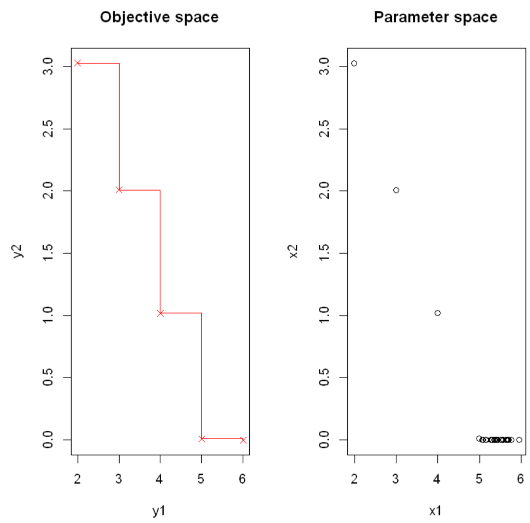

Figure 4. Candidate intended package(s) at $\theta = 5$

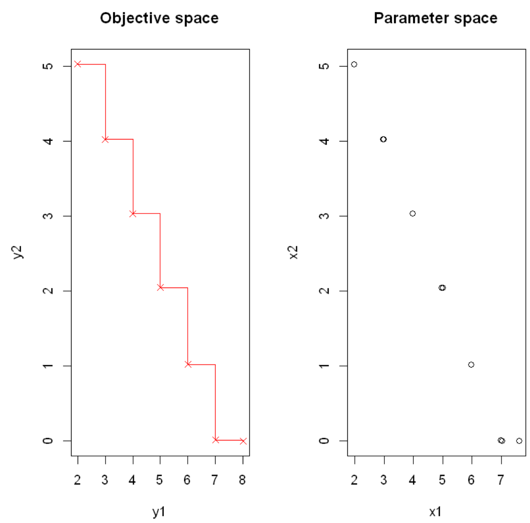

Figure 5. Candidate intended package(s) at $\theta = 7$

Note that candidate intended package(s) is/are distilled spatially at the zeroth level, and temporally after $ipomoea_generations$ iterations. Note also that the intended package actual ID needs to be adjusted before calling it.

```
ipomoea_intended_pack_all_ids <- ipomoea_intended_packs["value"]
ipomoea_intended_pack_inverted_id <- ipomoea_intended_pack_all_ids[
    ipomoea_generations]
ipomoea_intended_pack_actual_id <- length(ipomoea_ranks)
                                   - ipomoea_intended_pack_inverted_id +
                                        1
ipomoea_intended_function = ipomoea_funcs[ipomoea_intended_pack_
    actual_id]
```

Then came the automatic installation of the intended package(s):

```
ipomoea_unique_packs <- unique(ipomoea_packs)
```

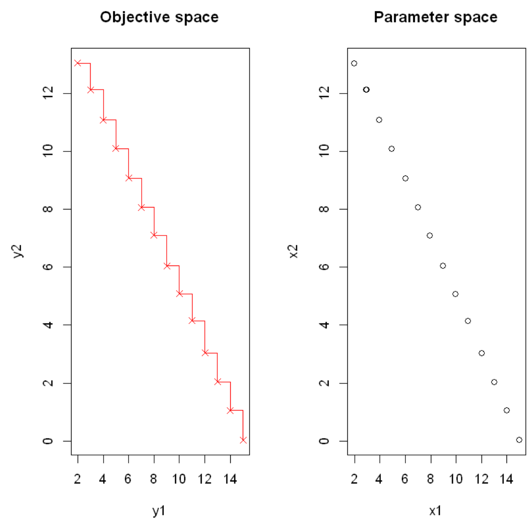

Figure 6. Candidate intended package(s) at $\theta = |ResultSet|$

```
ipomoea_intersection_packs <- (ipomoea_unique_packs %in% installed.packages()[,"Package"])
ipomoea_uninstalled_packs <- ipomoea_unique_packs[! ipomoea_intersection_packs]
if(length(ipomoea_uninstalled_packs)>0) install.packages(ipomoea_uninstalled_packs)
```

4 SUMMARY AND DISCUSSION

IPOMOEA front-end has two steps: query generation out of a keyword, and automatic search of the R site, as if RSiteSearch() is called thanks to Namazu search engine Namazu Project (2011).

IPOMOEA back-end has two steps too: harvesting all packages, functions within, and their ranks from HTML results provided from Namazu using regular expressions, before reaching the intended package using MOEA NSGA-II Deb et al. (2002).

Basic orientation is provided by the automatic installation of the intended package(s).

> This is ongoing project. Incorporating extra-features for selection criteria is a must to meet developers' preferences. Advanced orientation is highly admired. Possible orientation may consider displaying examples of use cases.

REFERENCES

R Core Team (2017). *R : A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197.

Inoue, K., Yokomori, R., Yamamoto, T., Matsushita, M., and Kusumoto, S. (2005). Ranking significance of software components based on use relations. *IEEE Transactions on Software Engineering*, 31(3):213–225.

McNaughton, R. and Yamada, H. (1960). Regular expressions and state graphs for automata. *IRE transactions on Electronic Computers*, (1):39–47.

Mersmann, O. et al. (2014). MCO: Multiple criteria optimization algorithms and related functions.

Namazu Project (2011). *Namazu: a Full-Text Search Engine*. Namazu Project, Japan.

Smith, D. (2017). Microsoft revolution analytics. Accessed: 2018-11-18.

Teetor, P. (2011). *R cookbook: Proven recipes for data analysis, statistics, and graphics.* " O'Reilly Media, Inc.".