Hands-on Large Scale Optimization in Python

From Beginning to Giving Up

Kunlei Lian

2023-02-18

Table of contents

Preface

1 Environment Setup

In this chapter, we explain the steps needed to set up Python and Google OR-Tools. All the steps below are based on MacBook Air with M1 chip and macOS Ventura 13.1.

1.1 Install Homebrew

The first tool we need is Homebrew, 'the Missing Package Manager for macOS (or Linux)', and it can be accessed at https://brew.sh/. To install Homebrew, just copy the command below and run it in the Terminal.

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install
```

We can then use the `brew --version` command to check the installed version. On my system, it shows the info below.

```
~/ brew --version
Homebrew 3.6.20
Homebrew/homebrew-core (git revision 5f1582e4d55; last commit 2023-02-05)
Homebrew/homebrew-cask (git revision fa3b8a669d; last commit 2023-02-05)
```

1.2 Install Anaconda

Since there are several Python versions available for our use and we may end up having multiple Python versions installed on our machine, it is important to use a consistent environment to work on our project in. Anaconda is a package and environment manager for Python and it provides easy-to-use tools to facilitate our data science needs. To install Anaconda, run the below command in the Terminal.

```
~/ brew install anaconda
```

After the installation is done, we can use `conda --version` to verify whether it is available on our machine or not.

```
~/ conda --version
conda 23.1.0
```

1.3 Create a Conda Environment

Now we will create a Conda environment named 'ortools'. Execute the below command in the Terminal, which effectively creates the required environment with Python version 3.10.

```
~/ conda create -n ortools python=3.10
Retrieving notices: ...working... done
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /opt/homebrew/anaconda3/envs/test

  added / updated specs:
    - python=3.10


The following packages will be downloaded:

    package                    |              build
    ---------------------------|-----------------
    setuptools-67.4.0          |     pyhd8ed1ab_0        567 KB  conda-forge
    ------------------------------------------------------------
                                           Total:        567 KB

The following NEW packages will be INSTALLED:

  bzip2              conda-forge/osx-arm64::bzip2-1.0.8-h3422bc3_4
  ca-certificates    conda-forge/osx-arm64::ca-certificates-2022.12.7-h4653dfc_0
  libffi             conda-forge/osx-arm64::libffi-3.4.2-h3422bc3_5
  libsqlite          conda-forge/osx-arm64::libsqlite-3.40.0-h76d750c_0
  libzlib            conda-forge/osx-arm64::libzlib-1.2.13-h03a7124_4
  ncurses            conda-forge/osx-arm64::ncurses-6.3-h07bb92c_1
  openssl            conda-forge/osx-arm64::openssl-3.0.8-h03a7124_0
  pip                conda-forge/noarch::pip-23.0.1-pyhd8ed1ab_0
  python             conda-forge/osx-arm64::python-3.10.9-h3ba56d0_0_cpython
```

```
    readline              conda-forge/osx-arm64::readline-8.1.2-h46ed386_0
    setuptools            conda-forge/noarch::setuptools-67.4.0-pyhd8ed1ab_0
    tk                    conda-forge/osx-arm64::tk-8.6.12-he1e0b03_0
    tzdata                conda-forge/noarch::tzdata-2022g-h191b570_0
    wheel                 conda-forge/noarch::wheel-0.38.4-pyhd8ed1ab_0
    xz                    conda-forge/osx-arm64::xz-5.2.6-h57fd34a_0


 Proceed ([y]/n)?
```

Type 'y' to proceed and Conda will create the environment for us. We can use `cnoda env list` to show all the created environments on our machine:

```
  ~/ conda env list
# conda environments:
#
base                     /opt/homebrew/anaconda3
ortools                   /opt/homebrew/anaconda3/envs/ortools
```

Note that we need to manually activate an environemnt in order to use it: `conda activate ortools`. On my machine, the activated environment `ortools` will appear in the beginning of my prompt.

```
  ~/ conda activate ortools
(ortools)  ~/
```

1.4 Install Google OR-Tools

As of this writing, the latest version of Google OR-Tools is 9.5.2237, and we can install it in our newly created environment using the command `pip install ortools==9.5.2237`. We can use `conda list` to verify whether it is available in our environment.

```
(ortools)  ~/ conda list
# packages in environment at /opt/homebrew/anaconda3/envs/ortools:
#
# Name                    Version                   Build  Channel
absl-py                   1.4.0                    pypi_0    pypi
bzip2                     1.0.8                h3422bc3_4    conda-forge
ca-certificates           2022.12.7            h4653dfc_0    conda-forge
libffi                    3.4.2                h3422bc3_5    conda-forge
```

```
libsqlite                3.40.0                 h76d750c_0      conda-forge
libzlib                  1.2.13                 h03a7124_4      conda-forge
ncurses                  6.3                    h07bb92c_1      conda-forge
numpy                    1.24.2                     pypi_0      pypi
openssl                  3.0.8                  h03a7124_0      conda-forge
ortools                  9.5.2237                   pypi_0      pypi
pip                      23.0.1                 pyhd8ed1ab_0    conda-forge
protobuf                 4.22.0                     pypi_0      pypi
python                   3.10.9          h3ba56d0_0_cpython     conda-forge
readline                 8.1.2                  h46ed386_0      conda-forge
setuptools               67.4.0                 pyhd8ed1ab_0    conda-forge
tk                       8.6.12                 he1e0b03_0      conda-forge
tzdata                   2022g                  h191b570_0      conda-forge
wheel                    0.38.4                 pyhd8ed1ab_0    conda-forge
xz                       5.2.6                  h57fd34a_0      conda-forge
```

Now we have Python and Google OR-Tools ready, we can start our next journey.

2 Introduction

This is a book created from markdown and executable code.

See Knuth (1984) for additional discussion of literate programming.

3 Environment Setup

In this chapter, we explain the steps needed to set up Python and Google OR-Tools. All the steps below are based on MacBook Air with M1 chip and macOS Ventura 13.1.

3.1 Install Homebrew

The first tool we need is Homebrew, 'the Missing Package Manager for macOS (or Linux)', and it can be accessed at https://brew.sh/. To install Homebrew, just copy the command below and run it in the Terminal.

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install
```

We can then use the `brew --version` command to check the installed version. On my system, it shows the info below.

```
~/ brew --version
Homebrew 3.6.20
Homebrew/homebrew-core (git revision 5f1582e4d55; last commit 2023-02-05)
Homebrew/homebrew-cask (git revision fa3b8a669d; last commit 2023-02-05)
```

3.2 Install Anaconda

Since there are several Python versions available for our use and we may end up having multiple Python versions installed on our machine, it is important to use a consistent environment to work on our project in. Anaconda is a package and environment manager for Python and it provides easy-to-use tools to facilitate our data science needs. To install Anaconda, run the below command in the Terminal.

```
~/ brew install anaconda
```

After the installation is done, we can use `conda --version` to verify whether it is available on our machine or not.

```
~/ conda --version
conda 23.1.0
```

3.3 Create a Conda Environment

Now we will create a Conda environment named 'ortools'. Execute the below command in the Terminal, which effectively creates the required environment with Python version 3.10.

```
~/ conda create -n ortools python=3.10
Retrieving notices: ...working... done
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /opt/homebrew/anaconda3/envs/test

  added / updated specs:
    - python=3.10


The following packages will be downloaded:

    package                    |             build
    ------------------------|-----------------
    setuptools-67.4.0          |     pyhd8ed1ab_0         567 KB  conda-forge
    ------------------------------------------------------------
                                          Total:         567 KB

The following NEW packages will be INSTALLED:

  bzip2              conda-forge/osx-arm64::bzip2-1.0.8-h3422bc3_4
  ca-certificates    conda-forge/osx-arm64::ca-certificates-2022.12.7-h4653dfc_0
  libffi             conda-forge/osx-arm64::libffi-3.4.2-h3422bc3_5
  libsqlite          conda-forge/osx-arm64::libsqlite-3.40.0-h76d750c_0
  libzlib            conda-forge/osx-arm64::libzlib-1.2.13-h03a7124_4
  ncurses            conda-forge/osx-arm64::ncurses-6.3-h07bb92c_1
  openssl            conda-forge/osx-arm64::openssl-3.0.8-h03a7124_0
  pip                conda-forge/noarch::pip-23.0.1-pyhd8ed1ab_0
  python             conda-forge/osx-arm64::python-3.10.9-h3ba56d0_0_cpython
```

```
    readline              conda-forge/osx-arm64::readline-8.1.2-h46ed386_0
    setuptools            conda-forge/noarch::setuptools-67.4.0-pyhd8ed1ab_0
    tk                    conda-forge/osx-arm64::tk-8.6.12-he1e0b03_0
    tzdata                conda-forge/noarch::tzdata-2022g-h191b570_0
    wheel                 conda-forge/noarch::wheel-0.38.4-pyhd8ed1ab_0
    xz                    conda-forge/osx-arm64::xz-5.2.6-h57fd34a_0


Proceed ([y]/n)?
```

Type 'y' to proceed and Conda will create the environment for us. We can use `cnoda env list` to show all the created environments on our machine:

```
  ~/ conda env list
# conda environments:
#
base                     /opt/homebrew/anaconda3
ortools                   /opt/homebrew/anaconda3/envs/ortools
```

Note that we need to manually activate an environemnt in order to use it: `conda activate ortools`. On my machine, the activated environment `ortools` will appear in the beginning of my prompt.

```
  ~/ conda activate ortools
(ortools)  ~/
```

3.4 Install Google OR-Tools

As of this writing, the latest version of Google OR-Tools is 9.5.2237, and we can install it in our newly created environment using the command `pip install ortools==9.5.2237`. We can use `conda list` to verify whether it is available in our environment.

```
(ortools)  ~/ conda list
# packages in environment at /opt/homebrew/anaconda3/envs/ortools:
#
# Name                    Version                   Build  Channel
absl-py                   1.4.0                     pypi_0    pypi
bzip2                     1.0.8                 h3422bc3_4    conda-forge
ca-certificates           2022.12.7             h4653dfc_0    conda-forge
libffi                    3.4.2                 h3422bc3_5    conda-forge
```

```
libsqlite              3.40.0           h76d750c_0    conda-forge
libzlib                1.2.13           h03a7124_4    conda-forge
ncurses                6.3              h07bb92c_1    conda-forge
numpy                  1.24.2                pypi_0    pypi
openssl                3.0.8            h03a7124_0    conda-forge
ortools                9.5.2237              pypi_0    pypi
pip                    23.0.1           pyhd8ed1ab_0  conda-forge
protobuf               4.22.0                pypi_0    pypi
python                 3.10.9        h3ba56d0_0_cpython    conda-forge
readline               8.1.2            h46ed386_0    conda-forge
setuptools             67.4.0           pyhd8ed1ab_0  conda-forge
tk                     8.6.12           he1e0b03_0    conda-forge
tzdata                 2022g            h191b570_0    conda-forge
wheel                  0.38.4           pyhd8ed1ab_0  conda-forge
xz                     5.2.6            h57fd34a_0    conda-forge
```

Now we have Python and Google OR-Tools ready, we can start our next journey.

Part I

Benders Decomposition

4 Benders Decomposition

In this chapter, we will explain the theories behind Benders decomposition and demonstrate its usage on a trial linear programming problem. Keep in mind that Benders decomposition is not limited to solving linear programming problems. In fact, it is one of the most powerful techniques to solve some large-scale mixed-integer linear programming problems.

In the following sections, we will go through the critical steps during the decomposition process when applying the algorithm on optimization problems represented in standard forms. This is important as it helps build up the intuition of when we should consider applying Benders decomposition to a problem at hand. Often times, recognizing the applicability of Benders decomposition is the most important and challenging step when solving an optimization problem. Once we know that the problem structure is suitable to solve via Benders decomposition, it is straightforward to follow the decomposition steps and put it into work.

Generally speaking, Benders decomposition is a good solution candidate when the resulting problem is much easier to solve if some of the variables in the original problem are fixed. We will illustrate this point using an example in the following sections. In the optimization world, the first candidate that should come to mind when we say a problem is easy to solve is a linear programming formulation, which is indeed the case in Benders decomposition applications.

4.1 The Decomposition Logic

To explain the reasoning of Benders decomposition, let us look at the standard form of linear programming problems that involve two vector variables, $\mathbf{x}$ and $\mathbf{y}$. Let p and q indicate the dimensions of $\mathbf{x}$ and $\mathbf{y}$, respectively. Below is the original problem $\mathbf{P}$ we intend to solve.

$$(\mathbf{P}) \qquad \text{min.} \quad \mathbf{c}^T \mathbf{x} + \mathbf{f}^T \mathbf{y} \qquad (4.1)$$
$$\text{s.t.} \quad \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} = \mathbf{b} \qquad (4.2)$$
$$\mathbf{x} \geq 0, \mathbf{y} \geq 0 \qquad (4.3)$$

In this formulation, $\mathbf{c}$ and $\mathbf{f}$ in the objective function represent the cost coefficients associated with decision variables $\mathbf{x}$ and $\mathbf{y}$, respectively. Both of them are column vectors of corresponding dimensions. In the constraints, matrix $\mathbf{A}$ is of dimension $m \times p$, and matrix $\mathbf{B}$ is of dimension $m \times q$. $\mathbf{b}$ is a column vector of dimension m.

Suppose the variable $\mathbf{y}$ is a complicating variable in the sense that the resulting problem is substantially easier to solve if the value of $\mathbf{y}$ is fixed. In this case, we could rewrite problem $\mathbf{P}$ as the following form:

$$\text{min.} \quad \mathbf{f}^T \mathbf{y} + g(\mathbf{y}) \tag{4.4}$$
$$\text{s.t.} \quad \mathbf{y} \geq 0 \tag{4.5}$$

where $g(\mathbf{y})$ is a function of $\mathbf{y}$ and is defined as the subproblem $\mathbf{SP}$ of the form below:

$$(\mathbf{SP}) \qquad\qquad \text{min.} \quad \mathbf{c}^T \mathbf{x} \tag{4.6}$$
$$\text{s.t.} \quad \mathbf{A}\mathbf{x} = \mathbf{b} - \mathbf{B}\mathbf{y} \tag{4.7}$$
$$\mathbf{x} \geq 0 \tag{4.8}$$

Note that the $\mathbf{y}$ in constraint (4.7) takes on some known values when the problem is solved and the only decision variable in the above formulation is $\mathbf{x}$. The dual problem of $\mathbf{SP}$, $\mathbf{DSP}$, is given below.

$$(\mathbf{DSP}) \qquad\qquad \text{max.} \quad (\mathbf{b} - \mathbf{B}\mathbf{y})^T \mathbf{u} \tag{4.9}$$
$$\text{s.t.} \quad \mathbf{A}^T \mathbf{u} \leq \mathbf{c} \tag{4.10}$$
$$\mathbf{u} \text{ unrestricted} \tag{4.11}$$

A key characteristic of the above $\mathbf{DSP}$ is that its solution space does not depend on the value of $\mathbf{y}$, which only affects the objective function. According to the Minkowski's representation theorem, any $\bar{\mathbf{u}}$ satisfying the constraints (4.10) can be expressed as

$$\bar{\mathbf{u}} = \sum_{j \in \mathbf{J}} \lambda_j \mathbf{u}_j^{point} + \sum_{k \in \mathbf{K}} \mu_k \mathbf{u}_k^{ray} \tag{4.12}$$

where $\mathbf{u}_j^{point}$ and $\mathbf{u}_k^{ray}$ represent an extreme point and extreme ray, respectively. In addition, $\lambda_j \geq 0$ for all $j \in \mathbf{J}$ and $\sum_{j \in \mathbf{J}} \lambda_j = 1$, and $\mu_k \geq 0$ for all $k \in \mathbf{K}$. It follows that the $\mathbf{DSP}$ is equivalent to

$$\text{max.} \quad (\mathbf{b} - \mathbf{B}\mathbf{y})^T \left(\sum_{j \in \mathbf{J}} \lambda_j \mathbf{u}_j^{point} + \sum_{k \in \mathbf{K}} \mu_k \mathbf{u}_k^{ray} \right) \tag{4.13}$$

$$\text{s.t.} \quad \sum_{j \in \mathbf{J}} \lambda_j = 1 \tag{4.14}$$

$$\lambda_j \geq 0, \ \forall j \in \mathbf{J} \tag{4.15}$$

$$\mu_k \geq 0, \ \forall k \in \mathbf{K} \tag{4.16}$$

We can therefore conclude that

- The **DSP** becomes unbounded if any $\mathbf{u}_k^{ray}$ exists such that $(\mathbf{b} - \mathbf{B}\mathbf{y})^T \mathbf{u}_k^{ray} > 0$. Note that an unbounded **DSP** implies an infeasible **SP** and to prevent this from happening, we have to ensure that $(\mathbf{b} - \mathbf{B}\mathbf{y})^T \mathbf{u}_k^{ray} \leq 0$ for all $k \in \mathbf{K}$.
- If an optimal solution to **DSP** exists, it must occur at one of the extreme points. Let g denote the optimal objective value, it follows that $(\mathbf{b} - \mathbf{B}\mathbf{y})^T \mathbf{u}_j^{point} \leq g$ for all $j \in \mathbf{J}$.

Based on this idea, the **DSP** can be reformulated as follows:

$$\text{min.} \quad g \tag{4.17}$$

$$\text{s.t.} \quad (\mathbf{b} - \mathbf{B}\mathbf{y})^T \mathbf{u}_k^{ray} \leq 0, \ \forall j \in \mathbf{J} \tag{4.18}$$

$$(\mathbf{b} - \mathbf{B}\mathbf{y})^T \mathbf{u}_j^{point} \leq g, \ \forall k \in \mathbf{K} \tag{4.19}$$

$$j \in \mathbf{J}, k \in \mathbf{K} \tag{4.20}$$

Constraints (4.18) are called **Benders feasibility cuts**, while constraints (4.19) are called **Benders optimality cuts**. Now we are ready to define the Benders Master Problem (**BMP**) as follows:

$$(\textbf{BMP}) \qquad \text{min.} \quad \mathbf{f}^T \mathbf{y} + g \tag{4.21}$$

$$\text{s.t.} \quad (\mathbf{b} - \mathbf{B}\mathbf{y})^T \mathbf{u}_k^{ray} \leq 0, \ \forall j \in \mathbf{J} \tag{4.22}$$

$$(\mathbf{b} - \mathbf{B}\mathbf{y})^T \mathbf{u}_j^{point} \leq g, \ \forall k \in \mathbf{K} \tag{4.23}$$

$$j \in \mathbf{J}, k \in \mathbf{K}, \mathbf{y} \geq 0 \tag{4.24}$$

Typically J and K are too large to enumerate upfront and we have to work with subsets of them, denoted by J_s and K_s, respectively. Hence we have the following Restricted Benders Master Problem (**RBMP**):

$$\mathbf{(RBMP)} \qquad\qquad \text{min.} \quad \mathbf{f}^T\mathbf{y} + g \qquad\qquad\qquad (4.25)$$

$$\text{s.t.} \quad (\mathbf{b} - \mathbf{By})^T\mathbf{u}_k^{ray} \leq 0, \ \ \forall j \in \mathbf{J}_s \qquad (4.26)$$

$$(\mathbf{b} - \mathbf{By})^T\mathbf{u}_j^{point} \leq g, \ \ \forall k \in \mathbf{K}_s \qquad (4.27)$$

$$j \in \mathbf{J}, k \in \mathbf{K}, \mathbf{y} \geq 0 \qquad\qquad (4.28)$$
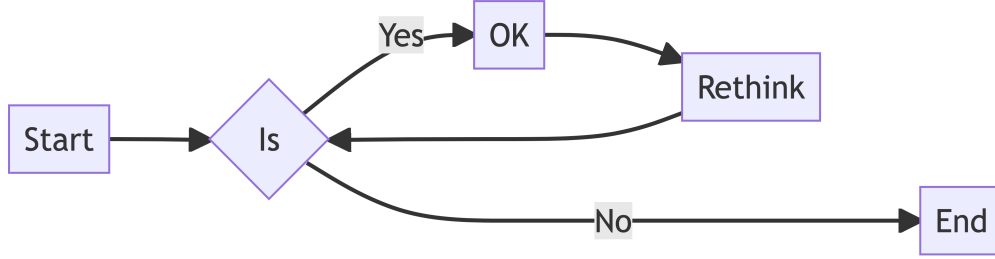

Figure 4.1: Benders decomposition workflow

4.2 A linear programming example

In this section, we will first present a small linear programming problem and solve it directly using the Gurobi API in Python - *gurobipy*. Then we will demonstrate the Benders decomposition approach on this artificial problem. Lastly, we will provide an implementation to solve this problem in gurobipy.

4.2.1 The original problem and its optimal solution

The linear program we examine here is devoid of any practical meaning and is solely used to demonstrate the solution process of Benders decomposition. The problem is stated below, in which $\mathbf{x} = (x_1, x_2, x_3)$ and $\mathbf{y} = (y_1, y_2)$ are the decision variables. We assume that $\mathbf{y}$ is the complicating variable.

$$\begin{aligned}
\text{min.} \quad & 8x_1 + 12x_2 + 10x_3 + 15y_1 + 18y_2 \\
\text{s.t.} \quad & 2x_1 + 3x_2 + 2x_3 + 4y_1 + 5y_2 = 300 \\
& 4x_1 + 2x_2 + 3x_3 + 2y_1 + 3y_2 = 228.75 \\
& 1x_1 + 2x_2 + 1x_3 + 1.5y_1 + 2y_2 = 150 \\
& 3x_1 + 2x_2 + 2x_3 + 1y_1 + 2y_2 = 180 \\
& x_i \geq 0, \ \forall i = 1, \cdots, 3 \\
& y_i \geq 0, \ \forall j = 1, 2
\end{aligned}$$

In this example, $\mathbf{c}^T = (8, 12, 10)$, $\mathbf{f}^T = (15, 18)$ and $\mathbf{b}^T = (300, 228.75, 150, 180)$. In addition,

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & 2 \\ 4 & 2 & 3 \\ 1 & 2 & 1 \\ 3 & 2 & 2 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 4 & 5 \\ 2 & 3 \\ 1.5 & 2 \\ 1 & 2 \end{bmatrix}$$

We first use Gurobi to identify its optimal solution.

```
import gurobipy as gp
from gurobipy import GRB

# Create a new model
env = gp.Env(empty=True)
env.setParam('OutputFlag', 0)
env.start()
model = gp.Model(env=env, name="original_problem")

# Decision variables
x1 = model.addVar(vtype=GRB.CONTINUOUS, name='x1')
x2 = model.addVar(vtype=GRB.CONTINUOUS, name='x2')
x3 = model.addVar(vtype=GRB.CONTINUOUS, name='x3')
y1 = model.addVar(vtype=GRB.CONTINUOUS, name='y1')
y2 = model.addVar(vtype=GRB.CONTINUOUS, name='y2')

# Objective function
model.setObjective(8*x1 + 12*x2 + 10*x3 + 15*y1 + 18*y2,
                   GRB.MINIMIZE)

# Constraints
model.addConstr(2*x1 + 3*x2 + 2*x3 + 4*y1 + 5*y2 == 300)
```

```python
model.addConstr(4*x1 + 2*x2 + 3*x3 + 2*y1 + 3*y2 == 220)
# model.addConstr(1*x1 + 2*x2 + 1*x3 + 1.5*y1 + 2*y2 == 150)
# model.addConstr(3*x1 + 2*x2 + 2*x3 + 1*y1 + 2*y2 == 180)

# Optimize the model
model.optimize()

# Print the results
if model.status == GRB.OPTIMAL:
    print("Optimal solution found!")
    print(f'x1 = {x1.X:.2f}')
    print(f'x2 = {x2.X:.2f}')
    print(f'x3 = {x3.X:.2f}')
    print(f'y1 = {y1.X:.2f}')
    print(f'y2 = {y2.X:.2f}')
    print(f"Total cost: {model.objVal:.2f}")
else:
    print("No solution found.")

# Close the Gurobi environment
model.dispose()
env.dispose()
```

The optimal solution and objective value are as follows.

```{python}
Optimal solution found!
x1 = 14.29
x2 = 0.00
x3 = 0.00
y1 = 0.00
y2 = 54.29
Total cost: 1091.43
```

Knuth, Donald E. 1984. "Literate Programming." *Comput. J.* 27 (2): 97–111. https://doi.or
g/10.1093/comjnl/27.2.97.