

In this document we show the process of creating AlgoRun containers for 3 different examples of published software: (1) the popular bioinformatics software *Bowtie* (Langmead, 2009), (2) *REACT* (Vera-Licona, 2014), a systems biology software to infer gene regulatory networks and, (3) the *KS algorithm* to solve the transversal hypergraph generation problem (Kavvadias, 2005).

For the first example, *Bowtie*, we show how to create an AlgoRun container, how to run the *Bowtie* AlgoRun container using AlgoRun web interface, how to expose command line options as parameters, some input examples to highlight the use of command line options vs. parameters and finally, how to access the tool deployed with AlgoRun via a RESTful API interface. For the other two examples we show how to create the AlgoRun containers and provide the users with the appropriate links to allow users to deploy and use all the AlgoRun features as presented in the first example.

## 1 Packaging Bowtie Software with AlgoRun

Bowtie (Langmead, 2009) is an ultra-fast memory-efficient short read aligner. The source code is written in C++ and is available under the Artistic License. Download it from

<http://sourceforge.net/projects/bowtie-bio/files/bowtie/1.1.2/>

Unzip the downloaded file. This unzipped file will contain all the source code of Bowtie.

**STEP 1:** Add all Bowtie source files inside the `src` folder.

**STEP 2:** Add the instructions to install the C++ dependencies as well as the instructions to build Bowtie source code to the Dockerfile. Below is how the Dockerfile of Bowtie looks like.

```
1 FROM algorun/algorun
2
3 ADD ./algorun_info /home/algorithm/web/algorun_info/
4 ADD ./src /home/algorithm/src/
5
6 # Install any algorithm dependencies here
7 RUN apt-get update && apt-get install -y build-essential
8 RUN cd /home/algorithm/src && make
```

*Source code: Dockerfile of Bowtie software*

---

### Hints:

1. Dockerfile syntax requires preceding all commands with `RUN` keyword.
  2. To ensure successful installation, always use `apt-get update` before installing packages and use `-y` option in the install command.
  3. Change to `/home/algorithm/src` directory before running any command that operates on the source files inside `src` folder.
- 

**STEP 3:** `manifest.json` file is required to describe the computational algorithm. Comments in the file will guide you to fill the correct values. Below is how the manifest of Bowtie looks like.

```

1 {
2   "manifest_version": "1.2",
3   "algo_name": "Bowtie 1.1.2",
4   "algo_summary": "Bowtie is an ultrafast, memory-efficient alignment program for aligning short DNA sequence reads to large genomes.",
5   "algo_description": "Bowtie is an ultrafast, memory-efficient short read aligner geared toward quickly aligning large sets of short DNA sequences (reads) to large genomes. Check <a href='\"http://bowtie-bio.sourceforge.net/targets/blank\">our website</a> for detailed explanation. This interface is meant to provide a quick and easy access to the computation without having to install Bowtie packages. Command line options are exposed as parameters, which you can configure from the above window.",
6   "algo_website": "http://bowtie-bio.sourceforge.net/",
7   "algo_keywords": ["bowtie", "DNA", "genome", "sequencing", "alignment", "Burrows-Wheeler", "indexing"],
8   "algo_authors": [
9     {
10      "name": "Ben Langmead",
11      "email": "langmead@cs.umd.edu",
12      "profile_picture": "ben.jpg",
13      "personal_website": "http://www.cs.jhu.edu/~langmea/",
14      "organization": "John Hopkins University",
15      "org_website": "https://www.jhu.edu/"
16    },
17    {
18      "name": "Cole Trapnell",
19      "email": "colettrap@uw.edu",
20      "profile_picture": "cole.png",
21      "personal_website": "http://cole-trapnell-lab.github.io/team/cole-trapnell/",
22      "organization": "University of Washington",
23      "org_website": "http://www.uw.edu/"
24    }
25  ],
26   "algo_exec": "ruby bowtie.rb",
27   "algo_input_stream": "direct",
28   "algo_output_stream": "stdout",
29   "algo_parameters": {},
30   "input_type": "algorun:dna-sequence",
31   "output_type": "algorun:aligned-dna-sequence",
32   "algo_image": "algorun/bowtie"
33 }

```

Source Code: *manifest.json* of Bowtie software (comments-skimmed)

## STEP 4:

- **input\_example.txt** file includes a sample input data for users to quickly try the algorithm. Enter **ATGCATCATGCGCCAT** as an example.
- **output\_example.txt** file includes a sample of the expected output for the same input. It makes it easier for users to expect the results. The above input produces the following:

```

0      -      gi|110640213|ref|NC_008253.1| 148810      ATGGCGCATGATGCAT
IIIIIIIIIIIIIIIIII 0      10:A>G,13:C>G

```

## NOTES

- Bowtie source code comes with **e\_coli** index packaged by default. So, use it in the **algo\_exec**. If you included other indexes, it's ok to use them as well.
- Use **direct** in **algo\_input\_stream** to accept input directly from the command line. Bowtie has other options to read the input from a file. However, AlgoRun will automatically present an option to upload a file to the input area in the web interface.
- Use **stdout** in **algo\_output\_stream** to let AlgoRun get the result from the terminal. Bowtie has other options to write the output to a file. However, AlgoRun will automatically present an option to download the result to a file from the web interface.

### STEP 5:

- From the directory where the Dockerfile exists, build Bowtie container using:  
`docker build -t bowtie .`
- You should see a success message as in the following picture.

```
---> 0788ef071b7e
Removing intermediate container e6d52cdf612c
Step 6 : MAINTAINER Abdelrahman Hosny <abdelrahman.hosny@hotmail.com>
---> Running in 5c9d49c5c70f
---> 6e1bfa3d638f
Removing intermediate container 5c9d49c5c70f
Successfully built 6e1bfa3d638f
abdelrahman@abdelrahman-laptop:~/uchc/algorun/examples/bowtie-1.1.2$
```

*Bowtie container build success message*

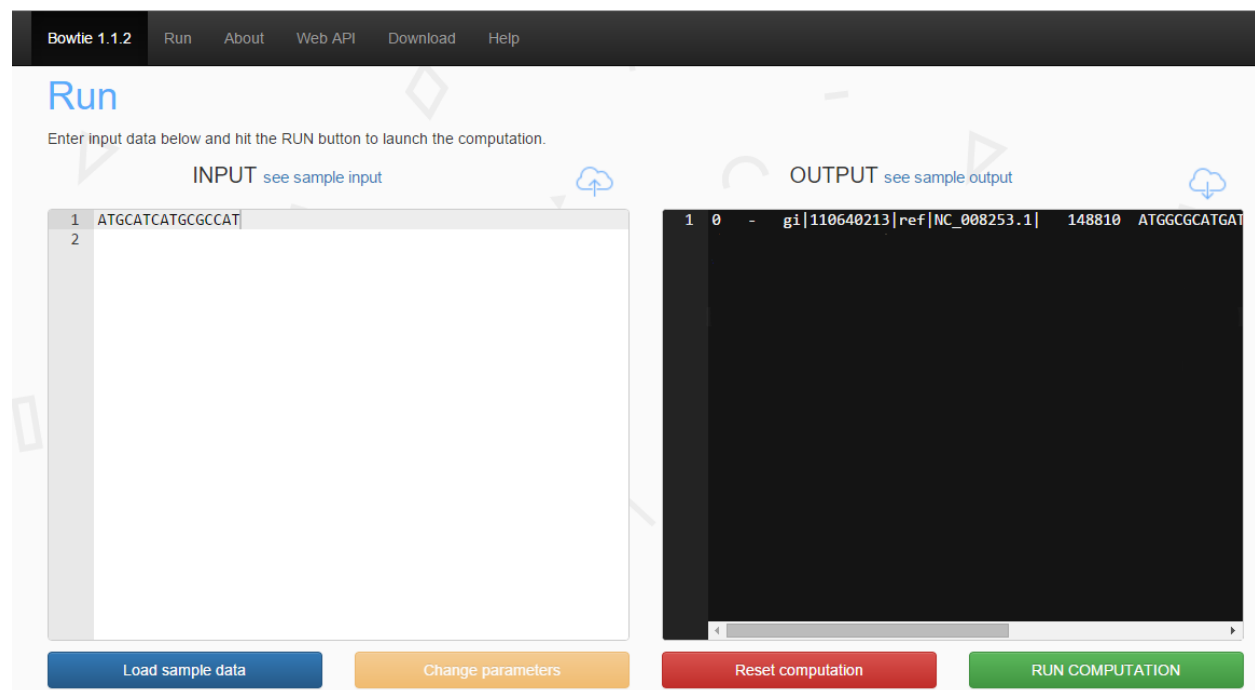
## 2 User Interface

Run Bowtie container using:

`docker run -p 31331:8765 bowtie`

Open the web browser and type <http://localhost:31331>

*Hint: You can use any available port other than 31331. Yet, you must bind it to 8765 port as it is the gateway to AlgoRun.*



### 3 [OPTIONAL] Expose Command Line Options as Parameters

To give flexibility to an implemented algorithm, AlgoRun allows exposing parameters that can be easily changed from the web interface. These parameters will be available as environment variables in the source code.

The power of Bowtie as a very fast DNA sequences aligner comes from the available command line options. So, you can make use of AlgoRun parameters to expose these command line options. You have two options: either to manipulate the source code of Bowtie so that it reads options from environment variables (instead of command line) or to develop a wrapper around Bowtie main executable that will internally translate environment variables to command line options. To do so, follow the below steps:

*The example here uses Ruby programming language to write the wrapper. You can use any other language and apply the same concepts.*

1. Specify parameters and their default values in the manifest file. The adjacent picture shows some parameters.
2. Read the input data<sup>1</sup>. The input data is passed as the first command line argument.
3. Read the environment variables (of the same names you specified in the manifest) and form the options string.
4. Call the executable file and to print the output to the standard output<sup>2</sup>.

```
28 ▾ "algo_parameters": {
29   "Skip": "0",
30   "Only-Align": "all",
31   "Trim-Left": "0",
32   "Trim-Right": "0",
33   "Phred-Quality": "33",
34   "Solexa": "off",
35   "Align-v": "0",
36   "Align-n": "2",
37   "Align-e": "70",
38   "Align-l": "28",
39   "Align-I": "0",
40   "Align-X": "250",
41   "Report-k": "1",
42   "Report-all": "off",
43   "Report-m": "no-limit",
44   "Report-best": "off",
45   "Report-strata": "off",
46   "suppress": "0"
47 },
```

Modify the Dockerfile to install ruby dependency:

```
7 RUN apt-get update && apt-get install -y ruby build-essential
```

Modify `algo_exec` value in the manifest file to:

```
25 "algo_exec": "ruby bowtie.rb",
```

Rebuild Bowtie container using: `docker build -t bowtie .`

<sup>1</sup> Remember that you can use “direct”, “file” or “stdin” alternatives to read the input. Whatever you choose, modify the manifest file accordingly.

<sup>2</sup> You can also write the output to a file and specify the file name in the manifest key “algo\_output\_stream”

```

1 require 'open3'
2
3 # read input data that is passed directly
4 input_data=ARGV[0].strip
5
6 # form the options string by reading environment variables
7 options = ""
8
9 options += " -s " + ENV["Skip"].strip
10 options += " -u " + ENV["Only-Align"].strip unless ENV["Only-Align"] == "all"
11 options += " -5 " + ENV["Trim-Left"].strip
12 options += " -3 " + ENV["Trim-Right"].strip
13 options += " --phred64-quals" if ENV["Phred-Quality"] == "64"
14 options += " --solexa-quals" if ENV["Solexa"] == "on"
15 options += " -n " + ENV["Align-n"].strip
16 options += " -v " + ENV["Align-v"].strip
17 options += " -m " + ENV["Align-m"].strip
18 options += " -e " + ENV["Align-e"].strip
19 options += " -l " + ENV["Align-l"].strip
20 options += " -I " + ENV["Align-I"].strip
21 options += " -X " + ENV["Align-X"].strip
22 options += " -k " + ENV["Report-k"].strip
23 options += " -all" if ENV["Report-all"] == "on"
24 options += " -m " + ENV["Report-m"].strip unless ENV["Report-m"] == "no-limit"
25 options += " --best" if ENV["Report-best"] == "on"
26 options += " --strata" if ENV["Report-strata"] == "on"
27 options += " --suppress " + ENV["suppress"].delete(' ') unless ENV["suppress"] == "0"
28 options.strip!
29
30 # run the algorithm with the options injected
31 command = "bowtie " + options + " e_coli -c " + input_data
32 stdin, stdout, stderr, wait_thr = Open3.popen3(command)
33
34 # print the output to the standard output stream
35 puts stdout.read
36 puts stderr.read

```

Source Code: *bowtie.rb* wrapper code

At this point, options available from Bowtie can be changed by clicking on “Change Parameters” button from the web interface. Visit <http://bowtie.algorun.org> for the final version of Bowtie running inside AlgoRun standard container.

Find the complete example on AlgoRun GitHub repository (<https://github.com/algorun/algorun/tree/master/examples/bowtie-1.1.2>).

The screenshot shows the Bowtie web interface with the following components and annotations:

- 1** access through web browser: Points to the browser address bar showing `bowtie.algorun.org`.
- 2** upload dataset file: Points to the cloud upload icon in the top right.
- 3** download computation result to a file: Points to the download icon in the top right.
- 4** click to see a sample of the input format: Points to the "see sample input" link.
- 5** click to see a sample of the output format: Points to the "see sample output" link.
- 6** input box: Points to the text input area for the dataset.
- 7** algorithm output box: Points to the output text area.
- 8** fill the input box with sample data: Points to the "Load sample data" button.
- 9** click to open parameters window: Points to the "Change parameters" button.
- 10** clear the input and output boxes: Points to the "Reset computation" button.
- 11** run the algorithm on the given dataset in the left: Points to the "RUN COMPUTATION" button.

The interface includes an "INPUT" section with a sample sequence, a "Parameters Configuration" table, and an "OUTPUT" section showing alignment results.

Bowtie web interface

## 4 Input Examples (Command Line Options vs. Parameters)

### 4.1. Example Link: <http://bowtie-bio.sourceforge.net/manual.shtml#example-1--a>

Command line: `./bowtie -a -v 2 e_coli --suppress 1,5,6,7 -c ATGCATCATGCGCCAT`

With AlgoRun: Change **Report-all** to on, **Align-v** to 2 and **suppress** to 1,5,6,7

### 4.2. Example Link: <http://bowtie-bio.sourceforge.net/manual.shtml#example-2--k-3>

Command line: `./bowtie -k 3 -v 2 e_coli --suppress 1,5,6,7 -c ATGCATCATGCGCCAT`

With AlgoRun: Change **Report-k** to 3, **Align-v** to 2 and **suppress** to 1,5,6,7

### 4.3. Example Link: <http://bowtie-bio.sourceforge.net/manual.shtml#example-3--k-6>

Command line: `./bowtie -k 6 -v 2 e_coli --suppress 1,5,6,7 -c ATGCATCATGCGCCAT`

With AlgoRun: Change **Report-k** to 6, **Align-v** to 2 and **suppress** to 1,5,6,7

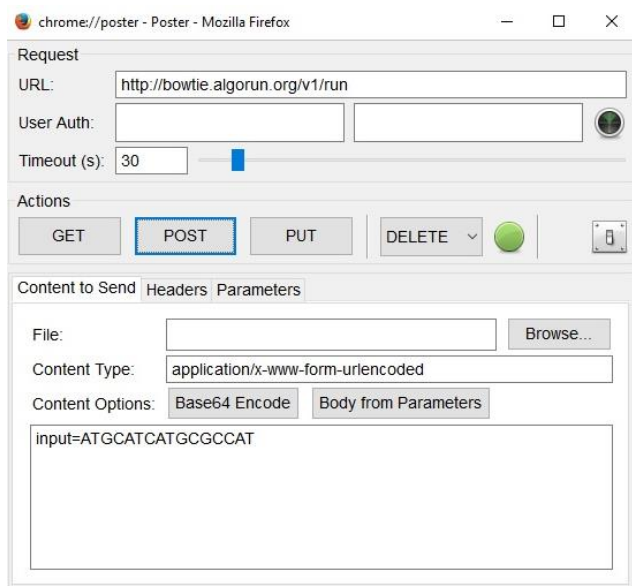
### 4.4. Example Link: <http://bowtie-bio.sourceforge.net/manual.shtml#example-9--a--m-3---best---strata>

Command line: `./bowtie -a -m 3 --best --strata -v 2 e_coli --suppress 1,5,6,7 -c ATGCATCATGCGCCAT`

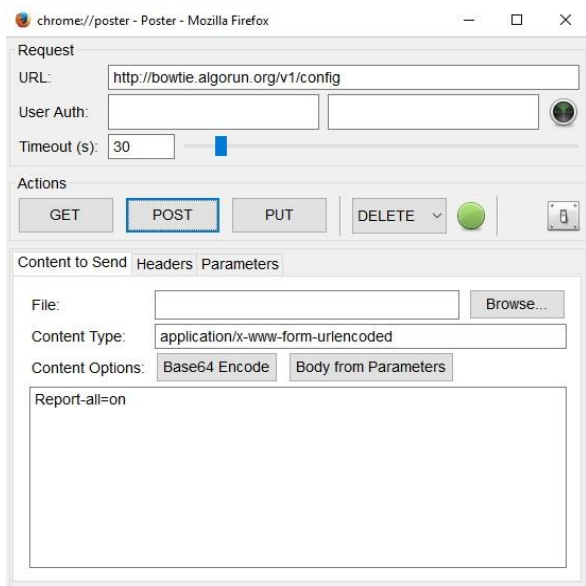
With AlgoRun: Change **Report-all** to on, **Report-m** to 3, **Report-best** to on, **Report-strata** to on, **Align-v** to 2 and **suppress** to 1,5,6,7.

## 5 Running Bowtie through AlgoRun's Web API

In addition to the web user interface available at <http://bowtie.algorun.org>, you can run Bowtie using the web API. Using the web API is useful to perform the computation from different client applications. As an example of running Bowtie through the web API, see the Firefox Poster plugin examples below. Refer to packaging software document for a detailed illustration on web APIs.



**Run Bowtie Computation:** (1) Type the URL of the endpoint `http://bowtie.algorun.org/v1/run` (2) Select **Body from Parameters**. (3) Type `input=ATGCATCATGCGCCAT`. (4) Click **Post** to initiate the request.



**Configure Bowtie Parameters:** (1) Type the URL of the endpoint `http://bowtie.algorun.org/v1/config` (2) Select **Body from Parameters**. (3) Type `Report-all=on`. (4) Click **Post** to initiate the request.



## 6 Packaging REACT Algorithm with AlgoRun

REACT<sup>3</sup> (Vera-Licona, 2014), is a software tool to reverse engineer gene regulatory networks from time series data. The source code is written in C++ and is available on GitHub at:

<https://github.com/veralicona/REACT/tree/master/src>

In addition, the source code includes ruby files as a helper to run the algorithm.

STEP 1: Add all REACT source files inside the `src` folder.

STEP 2: Add the instructions to install the C++ and ruby dependencies as well as the instructions to build REACT source code to the Dockerfile. Use the helper ruby file `ruby /home/algorithm/src/run.rb`  
`make`

```
1 FROM algorun/algorun
2 ADD ./algorun_info /home/algorithm/web/algorun_info/
3 ADD ./src /home/algorithm/src/
4
5 # Install any algorithm dependencies here
6 RUN apt-get update && \
7 apt-get install -y build-essential ruby
8 RUN ruby /home/algorithm/src/run.rb make
```

*Source code: Dockerfile of REACT algorithm*

STEP 3: `manifest.json` describes REACT algorithm. Below is how the manifest of REACT looks like.

```
1 {
2   "manifest_version": "1.2",
3   "algo_name": "REACT",
4   "algo_summary": "Evolutionary Algorithm for Discrete Dynamical System Optimization",
5   "algo_description": "The inference of gene regulatory networks (GRNs) from system-level experimental observations is at the heart of systems
6   biology due to its centrality in gaining insight into the complex regulatory mechanisms in cellular systems. This includes the inference of both
7   the network topology and dynamic mathematical models.",
8   "algo_website": "http://compsysmed.org/Software/EARevEng/REACT.html",
9   "algo_keywords": ["reverse engineering", "cell biology"],
10  "algo_authors": [
11    {
12      "name": "Paola Vera-Licona",
13      "email": "veralicona@uchc.edu",
14      "profile_picture": "",
15      "personal_website": "http://compsysmed.org",
16      "organization": "Center for Quantitative Medicine, UConn Health",
17      "org_website": "http://cqm.uchc.edu/"
18    },
19    {
20      "name": "John J. McGee",
21      "email": "",
22      "profile_picture": "",
23      "personal_website": "",
24      "organization": "Wolfram Alpha",
25      "org_website": "http://www.wolframalpha.com/"
26    }
27  ],
28  "algo_exec": "ruby React.rb",
29  "algo_input_stream": "file",
30  "algo_output_stream": "output.txt",
31  "algo_parameters": {},
32  "input_type": "[superAdam:TimeSeriesSet,superAdam:PolynomialDynamicalSystemSet,(superAdam:DirectedGraph)]",
33  "output_type": "superAdam:PolynomialDynamicalSystemSet",
34  "algo_image": "algorun/react"
35 }
```

*Source Code: **manifest.json** of REACT algorithm (comments-skimmed)*

---

<sup>3</sup> REACT: Reverse Engineering with Evolutionary Computation Tools

#### STEP 4:

- `input_example.txt` file includes a sample input data for users to quickly try the algorithm. Copy and paste an example from: [http://react.algorun.org/algorun\\_info/input\\_example.txt](http://react.algorun.org/algorun_info/input_example.txt)
- `output_example.txt` file includes a sample of the expected output for the same input. The above input produces an output of the format:  
[http://react.algorun.org/algorun\\_info/output\\_example.txt](http://react.algorun.org/algorun_info/output_example.txt)

#### STEP 5:

- From the directory where the Dockerfile exists, build REACT container using:  
`docker build -t react .`

#### [EXTRA STEP] Expose REACT Parameters:

REACT algorithm uses default values for different parameters. To expose these parameters to the user, include them in the manifest file in the “algo\_parameter” key as in the below picture.

```

29 - "algo_parameters": {
30     "HammingPolyWeight": "0.5",
31     "ComplexityWeight": "0.2",
32     "RevEngWeight": "0",
33     "BioProbWeight": "0",
34     "HammingModelWeight": "0.35",
35     "PolyScoreWeight": "0.65",
36     "GenePoolSize": "100",
37     "NumCandidates": "55",
38     "NumParentsToPreserve": "5",
39     "MaxGenerations": "100",
40     "StableGenerationLimit": "50",
41     "MutateProbability": "0.5"
42 },

```

*REACT parameters in the manifest file*

Parameters can be changed by clicking on “Change Parameters” button from the web interface. Visit <http://react.algorun.org> for the final version of REACT running inside AlgoRun standard container.

Find the complete example on AlgoRun GitHub repository

(<https://github.com/algorun/algorun/tree/master/examples/REACT>).



## 7 Packaging KS Algorithm with AlgoRun

KS<sup>4</sup> Kavvadias-Stavropoulos algorithm (Kavvadias, 2005) generates all minimal hitting sets (traversals) of a hypergraph. The source code is written in Pascal and is available on Murakami and Uno's repository<sup>5</sup>.

As the source code is using a dialect of Pascal that is not compatible with the modern compiler, download a helper executable that has been written to come over that problem:

<https://github.com/algorun/algorun/tree/master/examples/ks>

For your convenience, the repository above includes KS algorithm source code as well.

STEP 1: Add KS `thg.pas` source file with the helper `mhs` file inside the `src` folder. The helper

STEP 2: Add the instructions to install the C++ dependencies and Pascal compiler to the Dockerfile. As the helper is written in python, add the instructions to install the python dependencies as well. After that, navigate to the `src` directory and compile the source code file `pc thg.pas`

Adding the instructions to install `python-pip` dependency helps in installing other python packages as `simplejson` in a much easier way.

```
1 FROM algorun/algorun:latest
2 ADD ./src /home/algorithm/src/
3 ADD ./algorun_info /home/algorithm/web/algorun_info/
4
5 # Install any algorithm dependencies here
6 RUN apt-get update && \
7     apt-get install -y --no-install-recommends \
8         build-essential \
9         fp-compiler \
10        python-pip \
11        && rm -rf /var/lib/apt/lists/
12 RUN pip install \
13     simplejson
14 WORKDIR /home/algorithm/src/alg
15 RUN pc thg.pas
```

*Source Code: Dockerfile of KS algorithm*

---

<sup>4</sup> More information: <http://lca.ceid.upatras.gr/~estavrop/transversal/>

<sup>5</sup> Murakami and Uno's repository <http://research.nii.ac.jp/~uno/dualization.html>

**STEP 3:** `manifest.json` describes KS algorithm. Below is how the manifest of KS looks like.

```

1 {
2   "manifest_version": "1.2",
3   "algo_name": "KS",
4   "algo_summary": "KS algorithm for minimal hitting set computations",
5   "algo_description": "Introduced in <a href='\"//doi.org/10.7155/jgaa.00107\"'>An efficient algorithm for the transversal hypergraph generation</a>
6   by Kavvadias and Stavropoulos. Container sources available at the CompSysMed group <a href='\"//github.com/VeraliconResearchGroup/THSGenerationAlgorit
7   hms/tree/master/containers/ks\"'>Github page</a>.",
8   "algo_website": "http://lca.ceid.upatras.gr/~estavrop/transversal/",
9   "algo_keywords": ["minimum hitting set", "hypergraph transversal", "MHS"],
10  "algo_authors": [
11    {
12      "name": "Elias C. Stavropoulos",
13      "email": "estavrop@ceid.upatras.gr",
14      "organization": "Department of Computer Engineering and Informatics, University of Patras",
15      "org_website": "www.ceid.upatras.gr"
16    },
17    {
18      "name": "Dimitris J. Kavvadias",
19      "email": "kavadias@ceid.upatras.gr",
20      "organization": "Department of Computer Engineering and Informatics, University of Patras",
21      "org_website": "www.ceid.upatras.gr"
22    }
23  ],
24  "algo_exec": "./mhs",
25  "algo_input_stream": "file",
26  "algo_output_stream": "out.dat",
27  "input_type": "algorun:mhs-in",
28  "output_type": "algorun:mhs-out",
29  "algo_image": "compsysmed/ks"
30 }

```

Source Code: `manifest.json` of KS algorithm (comments-skimmed)

**STEP 4:**

- `input_example.txt`: Copy and paste the following sample input:
 

```

{
  "sets": [
    [1, 2, 5],
    [3, 2, 4],
    [1, 3]
  ]
}

```
- `output_example.txt`: Copy and paste the following sample output:
 

```

{"transversals": [[2], [3, 4], [4, 5]], "timeTaken": 0.002045721,
"sets": [[1, 2, 5], [3, 2, 4], [1, 3]]}

```

**STEP 5:**

- From the directory where the Dockerfile exists, build KS container using:
 

```
docker build -t ks .
```

## 8 References

Langmead, B. *et al.* (2009). [Ultrafast and memory-efficient alignment of short DNA sequences to the human genome](#). *Genome Biology*, 10:R25.

Vera-Licona, P., Jarrah, A.S., Garcia, LD., McGee, J., Laubenbacher, R. (2014): [An Algebra-Based Method for Inferring Gene Regulatory Networks](#). *BMC Systems Biology*, 8:37.

Kavvadias D. and Stavropoulos E. (2005): [An Efficient Algorithm for the Transversal Hypergraph Generation](#). *J. of Graph Alg & App*, 9:2, 239-264.