untilperfect Documentation

Release 0.0.2

Sean Tully

Mar 26, 2019

Contents

CHAPTER 1

Introduction

The unt**ilp**erfect application solves the buffer preparation vessel sizing and assignment problem using mixed **i**nteger **l**inear **p**rogramming.

The source repo is at https://github.com/multipitch/untilperfect.

Builds are hosted at https://pypi.org/project/untilperfect/.

Documentation is hosted at https://untilperfect.readthedocs.io/.

This project was forked from https://github.com/multipitch/dissertation. The dissertation repo was created for my masters dissertation on the subject towards an MSc in Business Analytics from University College Dublin; further development has been forked here so that the dissertation repo remains frozen.

Install via pip:

```
$ pip install untilperfect
```

Provides the *untilperfect* CLI command

```
$ untilperfect --help
usage: model.py [-h] [-b BUFFERS] [-n] [-p PARAMETERS] [-f PATH] [-s SOLVER]
                [-t PROBLEM_TYPE] [-v VESSELS] [-w]

Solves the buffer preparation assignment and selection problem.

optional arguments:
-h, --help            show this help message and exit
-b BUFFERS, --buffers BUFFERS
                      buffers filename (default: 'buffers.csv')
-n, --no-plot         do not generate plot
-p PARAMETERS, --parameters PARAMETERS
                      parameters filename (default: 'parameters.ini')
-f PATH, --path PATH  file path (default: <current working directory>)
-s SOLVER, --solver SOLVER
                      solver to be used (default: 'COIN_CMD')
-t PROBLEM_TYPE, --problem-type PROBLEM_TYPE
                      specify model to solve (default: 'complete'), other
                      model options are 'basic', 'minimized_hold_time',
                      'mimimized_used_volume'
-v VESSELS, --vessels VESSELS
```

(continues on next page)

(continued from previous page)

```
                      vessel filename (default: vessels.csv)
-w, --write           write problem to file in .lp format
```

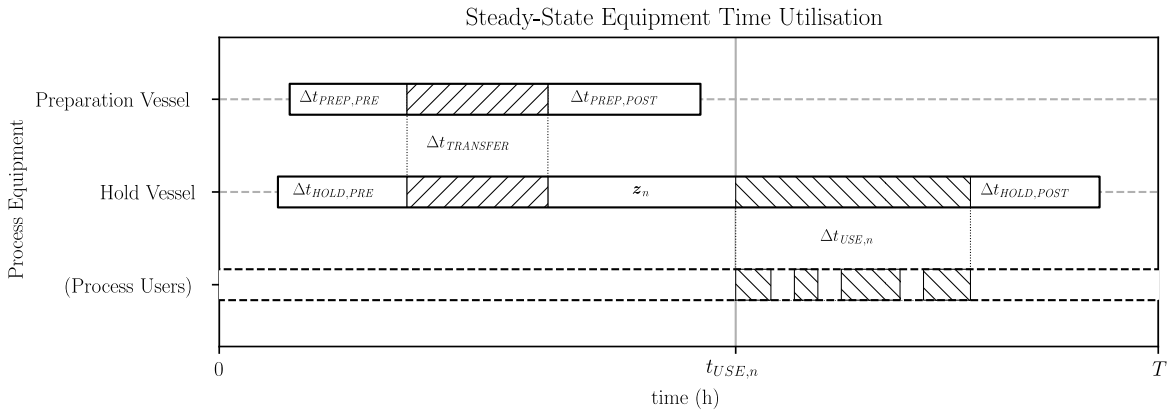
CHAPTER 2

Contents

2.1 Problem definition

The sections below detai lthe “complete” problem. Other problem variants are the “basic” and “minimized hold time” variants, which are discussed afterwards.

2.1.1 Given



Data on buffers e.g.

indices	names	volumes	use start times	use_durations
n	–	U_n	$t^*_{USE,n}$	$\Delta t_{USE,n}$
0	Buffer #1	5825.23	62.86	39.16
1	Buffer #2	10214.75	79.63	25.5
2	Buffer #3	13995.95	17.6	61.7
3	Buffer #4	14619.52	74.28	44.19
4	Buffer #5	4504.94	29.73	36.0
5	Buffer #6	16361.95	5.5	38.78
6	Buffer #7	3464.09	38.25	57.93
7	Buffer #8	13387.42	11.35	36.55
8	Buffer #9	1064.93	61.21	45.84
9	Buffer #10	1654.58	34.88	22.03
10	Buffer #11	23631.53	26.26	37.99
11	Buffer #12	11546.57	94.15	56.41

Data on available vessels

indices	names	volumes	costs
m	–	V_m	c_m
0	1000 L	1000.0	63.10
1	2000 L	2000.0	95.64
2	3000 L	3000.0	121.98
3	4000 L	4000.0	144.96
4	5000 L	5000.0	165.72
5	6000 L	6000.0	184.88
6	8000 L	8000.0	219.71
7	10000 L	10000.0	251.19
8	12000 L	12000.0	280.23
9	16000 L	16000.0	333.02
10	18000 L	18000.0	357.41
11	20000 L	20000.0	380.73
12	22000 L	22000.0	403.14
13	25000 L	25000.0	435.28
14	30000 L	30000.0	485.59

And some additional parameters:

parameter	symbol	value
process cycle time (h)	T	96.0
prep pre duration (h)	$\Delta t_{PREP,PRE}$	12.0
prep post duration (h)	$\Delta t_{PREP,POST}$	1.5
transfer duration (h)	$\Delta t_{TRANSFER}$	2.0
hold pre duration (h)	$\Delta t_{HOLD,PRE}$	8.0
hold post duration (h)	$\Delta t_{HOLD,POST}$	1.5
minimum hold duration (h)	$\Delta t_{HOLD,MIN}$	12.0
maximum hold duration (h)	$\Delta t_{HOLD,MAX}$	60.0
vessel minimum fill ratio	f_{MINUSE}	0.3
maximum prep utilization	f_{MAXUSE}	0.8
max slots	P	5

Support formulae:

$$m \in \mathcal{M}; \quad \mathcal{M} = \{0, 1, 2, \ldots, m, \ldots, (M-1)\}$$

$$n \in \mathcal{N}; \quad \mathcal{N} = \{0, 1, 2, \ldots, n, \ldots, (N-1)\}$$

$$p \in \mathcal{P}; \quad \mathcal{P} = \{0, 1, 2, \ldots, p, \ldots, (P-1)\}$$

$$M = |\mathcal{M}|$$

$$N = |\mathcal{N}|$$

$$P = |\mathcal{P}|$$

$$t_{USE,n} = t^*_{USE,n} \mod T \quad \forall n \in \mathcal{N}$$

$$V_{MAX} = \max\left(V_m \ \forall m \in \mathcal{M}\right)$$

$$\Delta t_{PREP} = \Delta t_{PREP,PRE} + \Delta t_{TRANSFER} + \Delta t_{PREP,POST}$$

2.1.2 Minimize

The total (relative) cost of buffer preparation vessels

$$\boldsymbol{Z} = \sum_{m \in \mathcal{M}} \sum_{p \in \mathcal{P}} c_m \boldsymbol{y}_{mp}$$

2.1.3 Subject to

Each buffer must be prepared in a defined slot

$$\sum_{p \in \mathcal{P}} \boldsymbol{x}_{np} = 1 \quad \forall n \in \mathcal{N}$$

Each slot may contain at most one vessel

$$\sum_{m \in \mathcal{M}} \boldsymbol{y}_{mp} \leq 1 \quad \forall p \in \mathcal{P}$$

Each vessel must be sufficiently large to prepare allocated buffers

$$U_n \boldsymbol{x}_{np} - \sum_{m \in \mathcal{M}} V_m \boldsymbol{y}_{mp} \leq 0 \quad \forall n \in \mathcal{N}, \ \forall p \in \mathcal{P}$$

Each vessel must be sufficiently small to prepare allcoated buffers

$$V_{MAX} \boldsymbol{x}_{np} + f_{MINFILL} \sum_{m \in \mathcal{M}} V_m \boldsymbol{y}_{mp} \leq U_n + V_{MAX} \quad \forall n \in \mathcal{N}, \ \forall p \in \mathcal{P}$$

Each preapration vessel must have a utilization below the maximum utilization limit

$$\Delta t_{PREP} \sum_{n \in \mathcal{N}} \boldsymbol{x}_{np} \leq f_{UTIL} T \quad \forall p \in \mathcal{P}$$

The total duration in each hold vessel must be less than the cycle time

$$\boldsymbol{z}_n \leq T - \Delta t_{HOLD,PRE} - \Delta t_{TRANSFER} - \Delta t_{USE,n} - \Delta t_{HOLD,POST} \quad \forall n \in \mathcal{N}$$

Buffer preparation procedures mustn't clash with one another

$$\begin{aligned} 2\boldsymbol{w}_{nkp} - \boldsymbol{x}_{np} - \boldsymbol{x}_{kp} &\leq -2 \\ \boldsymbol{w}_{nkp} - \boldsymbol{x}_{np} - \boldsymbol{x}_{kp} &\geq -1 \end{aligned} \quad \forall n \in \mathcal{N}, \ \forall k \in \mathcal{N}; \ k > n, \ \forall p \in \mathcal{P}$$

$$\begin{aligned} T\boldsymbol{q}_n - \boldsymbol{z}_n &\geq -t_{USE,n} \\ T\boldsymbol{q}_n - \boldsymbol{z}_n &\leq -t_{USE,n} + T \end{aligned} \quad \forall n \in \mathcal{N}$$

$$
\begin{aligned}
-T\boldsymbol{q}_n + T\boldsymbol{r}_n + \boldsymbol{z}_n &\leq t_{USE,n} + \Delta t_{PREP} \\
-T\boldsymbol{q}_n + T\boldsymbol{r}_n + \boldsymbol{z}_n &\geq t_{USE,n} + \Delta t_{PREP} - T
\end{aligned} \quad \forall n \in \mathcal{N}
$$

$$
\begin{aligned}
T\boldsymbol{q}_n + T\boldsymbol{s}_n - \boldsymbol{z}_n &\leq -t_{USE,n} + \Delta t_{PREP} \\
T\boldsymbol{q}_n + T\boldsymbol{s}_n - \boldsymbol{z}_n &\geq -t_{USE,n} + \Delta t_{PREP} + T
\end{aligned} \quad \forall n \in \mathcal{N}
$$

$$
\begin{aligned}
\boldsymbol{r}_n + \boldsymbol{s}_n - \ \boldsymbol{u}_n &\geq 0 \\
\boldsymbol{r}_n + \boldsymbol{s}_n - 2\boldsymbol{u}_n &\leq 0
\end{aligned} \quad \forall n \in \mathcal{N}
$$

$$
\begin{aligned}
T\boldsymbol{q}_k - T\boldsymbol{q}_n - 2T\boldsymbol{u}_n + 2T\boldsymbol{v}_{nk} + 2T\sum_{p\in\mathcal{P}} \boldsymbol{w}_{nkp} - \boldsymbol{z}_k + \boldsymbol{z}_n \\
\leq t_{USE,n} - t_{USE,k} - \Delta t_{PREP} + 4T \\
T\boldsymbol{q}_k - T\boldsymbol{q}_n + 2T\boldsymbol{u}_n + 2T\boldsymbol{v}_{nk} - 2T\sum_{p\in\mathcal{P}} \boldsymbol{w}_{nkp} - \boldsymbol{z}_k + \boldsymbol{z}_n \\
\geq t_{USE,n} - t_{USE,k} + \Delta t_{PREP} - 2T \\
T\boldsymbol{q}_k - T\boldsymbol{q}_n - T\boldsymbol{r}_n + 2T\boldsymbol{u}_n + 2T\sum_{p\in\mathcal{P}} \boldsymbol{w}_{nkp} - \boldsymbol{z}_k + \boldsymbol{z}_n \\
\leq t_{USE,n} - t_{USE,k} - \Delta t_{PREP} + 4T \\
T\boldsymbol{q}_k - T\boldsymbol{q}_n + T\boldsymbol{s}_n - 2T\boldsymbol{u}_n - 2T\sum_{p\in\mathcal{P}} \boldsymbol{w}_{nkp} - \boldsymbol{z}_k + \boldsymbol{z}_n \\
\geq t_{USE,n} - t_{USE,k} + \Delta t_{PREP} - 4T \\
\forall n \in \mathcal{N},\ \ \forall k \in \mathcal{N};\ k > n
\end{aligned}
$$

2.1.4 Where

The following decision variables are specified.

$$\boldsymbol{q}_n \in \{0,1\} \quad \forall n \in \mathcal{N}$$

$$\boldsymbol{r}_n \in \{0,1\} \quad \forall n \in \mathcal{N}$$

$$\boldsymbol{s}_n \in \{0,1\} \quad \forall n \in \mathcal{N}$$

$$\boldsymbol{u}_n \in \{0,1\} \quad \forall n \in \mathcal{N}$$

$$\boldsymbol{v}_{nk} \in \{0,1\} \quad \forall n \in \mathcal{N},\ \forall k \in \mathcal{N};\ k > n$$

$$\boldsymbol{w}_{nkp} \in \{0,1\} \quad \forall n \in \mathcal{N},\ \forall k \in \mathcal{N};\ k > n,\ \forall p \in \mathcal{P}$$

$$\boldsymbol{x}_{np} \in \{0,1\} \quad \forall n \in \mathcal{N},\ \forall p \in \mathcal{P}$$

$$\boldsymbol{y}_{mp} \in \{0,1\} \quad \forall m \in \mathcal{M},\ \forall p \in \mathcal{P}$$

$$\Delta t_{HOLD,MIN} \leq \boldsymbol{z}_n \leq \Delta t_{HOLD,MAX}; \quad \boldsymbol{z}_n \in \mathbb{R} \quad \forall n \in \mathcal{N}$$

2.2 Variant Problems

2.2.1 Basic

The “basic” variant omits the buffer preparation scheduling constraints and as such does not compute a workable schedule. It merely calculates the vessels required to maintain the specified utilization ratio at minimum cost.

2.2.2 Minimized Hold Time

The "minimized hold time" variant involves two rounds of optimization. Firstly, the "complete" problem is solved, resulting in a minimum cost.

Next, the optimum (minimized) cost is set as a constraint:

$$Z' = \min \boldsymbol{Z}$$

$$\sum_{m \in \mathcal{M}} \sum_{p \in \mathcal{P}} c_m \boldsymbol{y}_{mp} = Z'$$

Then the problem is re-run with the following objective to be minimized:

$$\boldsymbol{Y} = \sum_{n \in \mathcal{N}} \boldsymbol{z}_n$$

2.3 Nomenclature

Symbol	Description
$\boldsymbol{a}_{nk}$	buffers n and k prepared in same vessel (binary)
c_m	(relative) cost of vessel m
f_{MAXUSE}	buffer maximum use duration ratio
$f_{MINFILL}$	vessel minimum fill ratio
f_{MINUSE}	buffer minimum use duration ratio
f_{UTIL}	preparation slot maximum utilisation ratio
k	secondary buffer index $(k \in \mathcal{N}, k \neq n)$
m	vessel size index $(m \in \mathcal{M})$
n	buffer index $(n \in \mathcal{N})$
p	slot index $(p \in \mathcal{P})$
$\boldsymbol{q}_n$	the buffer n hold operation crosses the single-cycle boundaries (binary)
$\boldsymbol{r}_n$	$\boldsymbol{t}_{LOWER,n}$ occurs before $\boldsymbol{t}_{PREP,n}$ in the single-cycle window (binary)
$\boldsymbol{s}_n$	$\boldsymbol{t}_{UPPER,n}$ occurs after $\boldsymbol{t}_{PREP,n}$ in the single-cycle window (binary)
$\boldsymbol{t}_{LOWER,n}$	lower bound of feasible scheduling region for all buffers $k > n$ with respect to buffer n
$\boldsymbol{t}_{PREP,k}$	preparation reference time for buffer k
$\boldsymbol{t}_{PREP,n}$	preparation reference time for buffer n
$\boldsymbol{t}_{UPPER,n}$	upper bound of feasible scheduling region for all buffers $k > n$ with respect to buffer n
$t_{USE,n}$	buffer n time of first use, normalised
$t^*_{USE,n}$	buffer n time of first use, un-normalised
Δt_{FEAS}	maximum feasible buffer use duration
$\Delta t_{HOLD,MAX}$	maximum allowable buffer hold duration
$\Delta t_{HOLD,MIN}$	minimum allowable buffer hold duration
$\Delta t_{HOLD,POST}$	duration of post-use operations in buffer hold procedures
$\Delta t_{HOLD,PRE}$	duration of operations prior to receiving buffer in buffer hold procedures
Δt_{PREP}	total duration of buffer preparation procedures
$\Delta t_{PREP,POST}$	duration of operations post transferring out buffer in buffer preparation procedures
$\Delta t_{PREP,PRE}$	duration of operations prior to transferring out buffer in buffer preparation procedures
$\Delta t_{USE,n}$	duration of use of buffer n
$\Delta t_{TRANSFER}$	duration of transfers from buffer preparation vessel to buffer hold vessel
$\boldsymbol{u}_n$	feasible scheduling window for buffer k with respect to buffer n does not cross cycle boundary (binary)
$\boldsymbol{v}_{nk}$	feasible scheduling window for buffer k with respect to buffer n occurs before buffer n preparation procedure (binary)
$\boldsymbol{w}_{nkp}$	distinct buffers n and k are both made in slot p (binary)
$\boldsymbol{x}_{np}$	buffer n is prepared in slot p (binary)

Continued on next page

Table 1 – continued from previous page

Symbol	Description
$\boldsymbol{y}_{mp}$	a vessel of size m is in slot p (binary)
$\boldsymbol{z}_n$	buffer n hold duration
M	number of vessel sizes
$\mathcal{M}$	set of vessel sizes
N	number of buffers
$\mathcal{N}$	set of buffers
P	number of slots
$\mathcal{P}$	set of slots
T	process cycle time (start–to–start duration)
U_n	volume of buffer n to be prepared
V_m	maximum working volume of vessel size m
V_{MAX}	largest maximum working volume of available vessel sizes
$\boldsymbol{Y}$	secondary objective; sum of buffer hold times, given minimal total vessel cost
$\boldsymbol{Z}$	primary objective; total vessel cost
Z'	minimal total vessel cost

2.4 untilperfect package

2.4.1 Submodules

untilperfect.cli module

cli.py

Command line interface for untilperfect.

untilperfect.cli.**main**()
Provides command line interface to untilperfect.

Returns Status. See pulp.LpStatus.

Return type int

untilperfect.iotools module

iotools.py

Tools for reading and writing of files.

untilperfect.iotools.**column_reader**(*filename*)
Reads columnar data from csv file.

untilperfect.iotools.**get_config_section**(*filename='config.ini'*, *section='DEFAULT'*)
Reads a section from a configuration file.

untilperfect.model module

model.py

This module contains the definition of the buffer preparation vessel assignment problem along with classes for Parameters, Buffers and Vessels. It also contains a function for solving the problem.

class untilperfect.model.**BufferPrepProblem**(*parameters*, *buffers*, *vessels*, *solver=None*)
Bases: object

Determines the optium selection and assignment of prep vessels.

Parameters

- **parameters** (*untilperfect.Parameters*) –
- **buffers** (*untilperfect.Buffers*) –
- **vessels** (*untilperfect.Vessels*) –
- **solver** (*None or pulp.LpSolver, optional*) –

basic(*do_solve=True*)
Solve basic problem (no scheduling) to minimize cost.

Parameters **do_solve** (*bool, optional*) – If set to True (default), solves the problem upon construcion. Set this parameter to false to defer solution (useful if other constraints are to be applied before solving).

Returns If *do_solve* is set to True, returns problem status (see pulp.LpStatus).

Return type int

complete(*do_solve=True*)
Solve complete problem (includes scheduling) to minimize cost.

Parameters **do_solve** (*bool, optional*) – If set to True (default), solves the problem upon construcion. Set this parameter to false to defer solution (useful if other constraints are to be applied before solving).

Returns If *do_solve* is set to True, returns problem status (see pulp.LpStatus).

Return type int

evaluate(*problem_type*)
Generate some useful data from a solved problem.

minimized_hold_time()
Solve complete problem to first minimize vessel cost, then minimize hold times subject to the minimum cost.

Returns If *do_solve* is set to True, returns problem status (see pulp.LpStatus).

Return type int

minimized_used_volume()
Solve complete problem to first minimize vessel cost, then minimize used preparation volume, subject to minimum cost, finally, minimize hold times, subject to minimal cost and minimal used preparation volume.

Returns If *do_solve* is set to True, returns problem status (see pulp.LpStatus).

Return type int

plot(*filename='single_cycle_plot.svg'*)
Create a single-cycle steady-state equipment occupancy plot.

Parameters **filename** (*str, optional*) – Plot file name.

write(*filename='untilperfect.lp'*)
Write problem to file in .LP format.

Parameters **filename** (*str, optional*) – Output file name.

class untilperfect.model.**Buffers**(*data*)
Bases: *untilperfect.model.Data*

Selection of buffers to be prepared.

Parameters **data** (*str or dict*) – Pass either a filename string or a dict of parameters to initialize a Data class instance. The filename should be that of a valid data file (csv format).

set_relative_use_start_times(*cycle_time*)
Calculates use start times relative to cycle.

Parameters **cycle_time** (*float*) – Process cycle time.

class untilperfect.model.**Data**(*data*)
Bases: object

Parent class of 'Buffers' and 'Vessels'. If initialized with a 'dict', reads data from the 'dict'. If initialized with a 'str', treats it as a filename and reads data from the file.

Parameters **data** (*str or dict*) – Pass either a filename string or a dict of parameters to initialize a Data class instance. The filename should be that of a valid data file (csv format).

class untilperfect.model.**Parameters**(*data*)
Bases: object

Problem parameters.

Parameters **data** (*str or dict*) – Pass either a filename string or a dict of parameters to initialize a Parameters instance. The filename should be that of a valid parameters file; in ini format and with a [parameters] section.

class untilperfect.model.**Vessels**(*data*)
Bases: *untilperfect.model.Data*

Selection of preparation vessels available for use.

Parameters **data** (*str or dict*) – Pass either a filename string or a dict of parameters to initialize a Vessels class instance. The filename should be that of a valid data file (csv format).

untilperfect.model.**solve**(*parameters_file='parameters.ini'*, *buffers_file='buffers.csv'*, *vessels_file='vessels.csv'*, *problem_type=<function BufferPrepProblem.complete>*, *solver=None*, *plot=True*, *write=True*, *cli=False*)
Solve BufferPrepProblem.

Parameters

- **parameters_file** (*str, optional*) – Parameters filename.
- **buffers_file** (*str, optional*) – Buffers filename.
- **vessels_file** (*str, optional*) – Vessels filename.
- **problem_type** (*optional*) – Type of problem to solve.
- **solver** (*optional*) – Solver to use; see pulp.LpSolver
- **plot** (*bool, optional*) – Generate steady-state single-cylce equipent occupancy plot and save to file.
- **write** (*bool, optional*) – Write the problem to file in .lp format.
- **cli** (*bool, optional*) – Set to True to return problem status, returns problem object otherwise.

Returns If *cli=True*, returns status (int, see pulp.LpStatus), if *cli=False*, returns pulp.LpProblem instance.

Return type int or pulp.LpProblem

untilperfect.plots module

plots.py

This module contains functions to plot results.

untilperfect.plots.**cyclic_xranges**(*start_time*, *duration*, *cycle_time*)
Where a range crosses the cycle time boundary, split into 2 ranges.

Parameters

- **start_time** (*float*) –
- **duration** (*float*) –
- **cycle_time** (*float*) –

Returns List of operation time ranges.

Return type list

untilperfect.plots.**explanatory_plot**(*filename='explanatory.svg'*)
Plot that explains duration parameters.

untilperfect.plots.**label_style**(*label*)
Wrapper to encode labels in latex format if this setting is active.

Parameters **label** (*str*) –

Returns

Return type str

untilperfect.plots.**matplotlib_init**()
Change some matplotlib settings.

untilperfect.plots.**single_cycle_plot**(*problem*, *filename=None*)
Generate a single-cycle steady-state equipment occupancy plot.

Parameters

- **problem** (*untilperfect.Problem*) –
- **filename** (*str or None, optional*) – If a filename string is specified, saves plot to file, otherwise opens plot in a tk window.

untilperfect.pulptools module

pulptools.py

This module contains a class to handle multidimensional variables in pulp.

class untilperfect.pulptools.**LpVariableArray**(*name*, *dimensions*, *low_bound=None*, *up_bound=None*, *cat=None*, *e=None*)
Bases: object

Multidimensional decision variable array.

evaluate()
Evaluates decision variable values.

2.4.2 Module contents

init.py

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

u

Index