Co-Optimization of Fuels & Engines
FOR TOMORROW'S ENERGY-EFFICIENT VEHICLES

Milestone Completion Report

Completion Date:	12/21/16
Scheduled Completion:	12/31/16
Technology Area:	EE-3V Vehicles Technology Office
Co-Optima Technical Team:	Simulation Toolkit
Principle Investigator:	Ray Grout
Milestone Title:	TK G.3: Complete initial internal release of the Co-optimizer application
Authors:	Ray Grout, Juliane Muller
Participating Laboratories:	NREL, LBNL
Key Words:	Co-optimizer, analysis, simulation, non-linear programming, optimization
Reviewed By:	Matthew McNenly

Abstract:

This milestone reflects the assembly of a non-linear model to optimize fuel composition based on the engine efficiency merit function. An application was built that links open source components to solve the optimization problem that answers the question "what combination of candidate components maximizes the engine efficiency merit function". This work is tied directly to scenario analysis for Thrust I fuels and engines to support the 18 month decision point. The capabilities of this release include a Pareto front analysis for cost-merit tradeoffs, and the ability to determine the component blends that result in maximum merit for a set of "K" values, where "K" is the coefficient appearing in the merit function that describes the engine design. It is currently left to the user to provide values of fuel properties for each component. Other tools exist that can extract such information when available to be exported from the fuel properties database. Finally, in this release, the component fractions are in mole fraction and the properties are blended by linearly weighting by mole fraction.

How milestone was met

A collection of python scripts that implement the co-optimizer functionality are stored in a github repository at: git@github.com:rgrout/cooptimizer.git and is available on request to any member of co-optima. The requirement to request access is because there is a (small) cost associated with adding new users to the github access list. This milestone report is placed on the co-optima SharePoint site and serves as a reference to the availability that is broadly available. The co-optimizer is also configured and available on *Peregrine*, the computing facility at NREL that has become a common platform for simulation activities in co-optima.

Methods

Python; HPC; PyOMO; interior point optimization; NSGAII genetic algorithm, non-linear programming

Conclusions/lessons learned/next steps

This is an initial realization of the co-optimizer application that serves as a concrete implementation that can be used to aid clarity in discussion around the possibilities and use cases. The analysis performed in the course of developing this release was primarily proof of "what could be done" as many of the inputs are not yet available (individual component properties, component cost estimates, etc.). The next technical steps include: (i) implement a broader range of analysis patterns; (ii) establish a reference property database that can be distributed with the co-optimizer; and (iii) integrate more advanced blending models into the co-optimizer. An additional near term next step is to devise a strategy to deal with incomplete and uncertain inputs in a rigorous fashion.

References

Publications/Patents

Co-optimization of fuels and engines
Co-optimizer

This collection of python scripts is the first (internal) release of the co-optimizer, or the co-optima scenario analysis tool.

This release (V0.1) is not intended for broad distribution, and should not be shared outside of co-optima.

The intention of this release is to add concreteness to the co-optimization concepts, and facilitate discussion between the teams as the tool evolves.

Questions, comments, and rants can be directed to Ray.Grout@nrel.gov

Availability:

Currently, the code is hosted on github in a private repository.

For access, follow the following steps:

- Create an account on github.com if you don't have one already
- Email Ray.Grout@nrel.gov and request access. It would be useful if you include a note about any potential usage you have in mind (so that I can save you time if it isn't currently possible, and so it can be added to the wishlist or we can discuss what would be necessary to support it.)
- Clone the git repository at *git@github.com:rgrout/cooptimizer.git*

Prerequisites

- python
- Numpy
- Matplotlib
- PYOMO
- IPOPT
- deap / nsga2 (Installation guide)
- xlrd and xlwt

Setting up to run on Peregrine

The co-optimizer is set up and installed on Peregrine in /projects/optima/applications/co-optimizer. To set up your environment to run the co-optimizer on Peregine, a few modules are necessary:

module load python module load mkl

Then you can use pip to install the prerequisite python tools in your home directory:

pip install --user pyomo pip install --user xlrd pip install --user xlwt pip install --user deap

Finally, to use the pre-build ipopt executable you need to add to your path:

export PATH=$PATH:/projects/optima/applications/co-optimizer/utils/ipopt/bin

Input files

- cooptimizer_input.py
- Properties database input file (example: propDB_fiction.xls)

- Cost input file (example: costDB_fiction.xls)

There are tools included in this distribution to read a file output from the *fuel properties database*; however, since at the moment the properties necessary for the merit function largely need to be filled in by the user, they are not used. Instead, it is left to the user to fill in components and properties in the template files "propDB.xls" and "costDB.xls". The names are arbitrary and are specified in the cooptimizer_input.py file.

Usage and capabilities:

This release is capable of 2 primary capabilities.

1. Firstly, sweeping out a Pareto front to study the cost-merit trade off; that is, finding the composition that maximizes merit for a given cost (or, equivalently, finding the composition that minimizes cost for a given merit).

2. Sweeping across a range of values of "K" in the merit function and finding the composition that maximizes the merit function irrespective of cost (or for no data available)

The mode is selected by setting either:

task_list['cost_vs_merit_Pareto'] = True

or:

task_list['K_vs_merit_sweep'] = True

in the input file *cooptimizer_input.py*. Note that as of this release, only the former is implemented in the GA formulation.

The co-optimizer is run by:

python co_optimizer.py

Methodology

The co-optimizer uses two alternative algorithms to solve the merit function optimization problem, specified in the input file. The 'pyomo' implementation currently uses IPOPT for solving the non-linear interior point optimization problem. However, this method sometimes lacks robustness. On a single failure, the co-optimizer will attempt some heuristics to obtain a successful solution; if this is not possible, it will display an error message. There is also a 'deap_NSGAII' option, which will use the DEAP toolbox implementation of the NSGA2 (Non Sorting Genetic Algorithm II) to find the Pareto front. This method is more robust, but can take longer.

Acknowledgement

The co-optimizer was developed as part of the Co-Optimization of Fuels & Engines (Co-Optima) project sponsored by the U.S. Department of Energy (DOE) Office of Energy Efficiency and Renewable Energy (EERE), Bioenergy Technologies and Vehicle Technologies Offices. (Optional): Co-Optima is a collaborative project of multiple national laboratories initiated to simultaneously accelerate the introduction of affordable, scalable, and sustainable biofuels and high-efficiency, low-emission vehicle engines.