EVS28
KINTEX, Korea, May 3-6, 2015

Using multiobjective optimization for automotive component sizing

R. Vijayagopal, R. Chen, P. Sharer, S.M.Wild, A. Rousseau
Argonne National Laboratory, Argonne, IL, USA, rvijayagopal@anl.gov

Short Abstract

This paper shows how a multiobjective problem is formulated and solved in order to size the components of a vehicle with a split hybrid transmission, such as a Toyota Prius. The goal is to explore feasible design options and the trade-offs between fuel economy and vehicle cost. Eight input variables are provided for this optimization, including plant variables such as maximum power ratings for engine, motors, and battery; final drive ratio; and control variables that determine how the battery energy is utilized. Three constraints are used: achievement of the battery charge balance, ability to trace the drive cycle, and ability to achieve a zero to 60 mph acceleration performance within 10 seconds.

We describe a multiobjective optimization algorithm that we have implemented in Autonomie, a simulation tool developed at Argonne, and we demonstrate its ability to utilize parallel computing capabilities of Matlab. A parallel/distributed-computing infrastructure is used to simultaneously evaluate multiple combinations of input parameters, over multiple drive cycles, thereby reducing the overall time taken to perform the optimization and hence reduce the total solution time. The optimization produces several design choices, which form a Pareto front. The search algorithm ensures that as the number of iterations increases, more and more points are added on or near the Pareto front. All the points that form the front are relevant design choices, and the front characterizes the balance between conflicting goals such as fuel economy and performance.

1 Introduction

The multiobjective optimization problem described in this paper differs from regular optimization problems in several respects. Foremost, it involves minimizing two conflicting goals: that is, the input variables that help reduce one of the goals tends to increase the value of the other goal. In automotive design, such problems are prevalent, and few tools exist to assist in making good decisions.

Besides the manufacturing cost, the two factors that most determine the commercial viability of a vehicle are fuel economy and acceleration performance. Some vehicles excel in only one of these factors, and these designs can be optimized with single-objective optimization exercises with appropriate design constraints. If the vehicle cost and desired acceleration performance are fixed, then optimization can be done with these design qualities as constraints and fuel economy as the single objective. Doing so, however, might result in overlooking many other attractive design choices. Perhaps one can get better acceleration for the same fuel economy at a lower cost, or can get significantly better fuel economy at a slight loss of performance, and slight increase in cost. In order to understand all the attractive optimum design choices, a multiobjective optimization exercise is necessary.

This study seeks to find such optimal design choices for a mid-sized hybrid electric vehicle (HEV) when maximizing fuel economy and minimizing vehicle cost. Argonne National Laboratory has developed a simulation process for such a trade-off analysis. This process is integrated in the simulation framework Autonomie [1]. In this paper we explain the steps involved and demonstrate a specific test case on a hybrid vehicle like a Toyota Prius.

The default vehicle in Autonomie gets over 50 mpg for the UDDS cycle and is considered fuel efficient. However this paper shows that a HEV component sizing is available that has better fuel economy, comparable acceleration performance, and lower cost.

2 Optimization Problem

We describe here the optimization problem and its implementation. We also explain the importance of using a multiobjective approach.

2.1 Problem Description

In a hybrid vehicle, engine and motors can provide the propulsion power. The planetary gear set provides a unique way to combine the torque output from these prime movers. We also assume a final drive ratio between the planetary gears and the wheel.

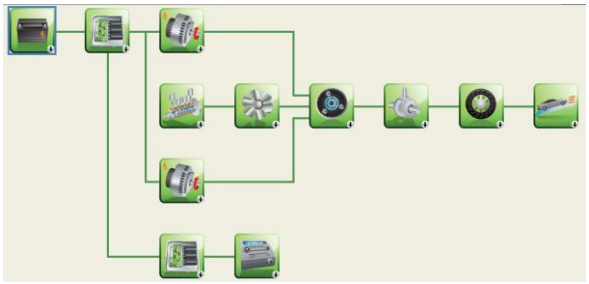

Figure 1. Schematic of the hybrid vehicle powertrain used for this study

In this paper, we have identified two objective functions: fuel economy, calculated in terms of miles per gallon, and vehicle cost, calculated in terms of dollars. Autonomie has cost estimations for all component models (engine, motor, battery etc) which are scaled with the component size. Factors that usually contribute to a better fuel economy in a HEV, larger motor, larger battery etc will cost more. An improvement in fuel economy can involve a trade-off in cost. Other factors like final drive ratio or control parameters may not affect cost but could affect the acceleration performance and fuel economy.

This example shows that the objectives considered in our study are in conflict with each other and that multiobjective optimization is clearly needed in order to characterize the nature of these conflicts.

2.2 Implementation

The implementation of the problem involves defining the input variables, objectives, and constraints. The input parameters are summarized in Table 1.

Table 1. Design variables for the optimization problem

Input Variables	Unit	Default	Min	Max
Engine Power	kW	88	50	200
Final Drive Ratio		4	2	5
Motor2 Power	kW	51	10	100
Motor Power	kW	58	10	100
Battery Energy	kWh	4	1	12
Battery Power	kW	63	16	129
Min Engine operating power	kW	1	0	30
Max power for EV operation	kW	20	0	100

The components are scaled to meet the desired power or energy ratings. As part of this scaling, the cost of the components is also estimated. Table 2 shows the scaling parameters used for estimating the new cost values. Typically the cost is computed as Cost = k*x + c, where c is a constant value and k is the scaling parameter. In the case of the battery, the maximum value from either the power or energy requirement is taken as the cost of the battery.

Table 2. Design variables and their impact on cost

Cost scaling factors	Unit	k
Engine	$/kW	6.2
Final Drive		0
Motor2	$/kW	13
Motor	$/kW	13
Battery (Energy)	$/kWh	120
Battery (Power)	$/kW	22
Min Engine operating power	kW	0
Max power for EV operation	kW	0

Parameters such as the power rating of the engine, motor, and battery may not require much explanation, but the last two parameters in Table 1 refer to control variables that determine the way the vehicle utilizes the hybrid powertrain. The vehicle controller estimates the power that is demanded from engine and motors. If the engine is required to be kept running, then a minimum power output is imposed in order to avoid idling, as well as very low power operations. This minimum power output is determined by the 'minimum engine operating power' parameter. If the estimated power demand from the wheels exceed a certain threshold, the engine is turned on by the vehicle controller. Typically this should not be higher than the maximum power the battery or motor can provide. This threshold value is called 'maximum power for EV operation' in this study and is used as a lookup table in the model because it varies with battery state of charge. This particular lookup table is scaled based on the value of this parameter.

The minimum and maximum limits on the design variables are chosen based on the range of values available for such components. Estimated cost of

the vehicle can be computed based on the component sizes, even without running any simulation. But the other two metrics of interest, fuel economy and performance, are computed after running simulations. Autonomie has default procedures for measuring fuel economy and acceleration performance. The fuel economy is measured on a charge-sustaining run over a UDDS cycle.

All the major component sizes need to be evaluated in order to obtain the optimal design choices. Since only two objectives are involved here, visualizing the output of this problem is easy. Specifically, a Pareto front can be developed as shown in Figure 2, which indicates the trade-offs between fuel economy and vehicle manufacturing cost as achieved by the specific vehicle. The red points in Figure 2, indicate the design choices explored in search for better fuel economy and the blue points show the choices evaluated in an attempt to lower the vehicle cost.

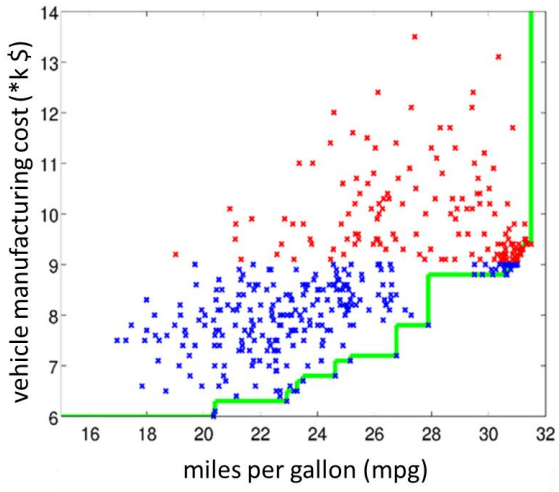

Figure 2. Example of a Pareto front for two conflicting objectives

All the designs that fall on the green line are relevant Pareto optimal solutions; and based on what fuel economy or acceleration performance is needed, the best design that meets these criteria can be selected from this optimal set of designs. Any point in this set can be traced back to a list of valid input variables.

2.3 Multiobjective Optimization

In multiobjective optimization [2], one considers the optimization of multiple objectives simultaneously. Specifically, one seeks the set of Pareto optimal points. A point x is said to be Pareto optimal if no other point y is better in all of the objectives, that is, no y exists such that f(x)>f(y) for all i. Intuitively, for Pareto optimal points, one cannot improve an individual objective without worsening another. The Pareto front corresponds to the objective values of the set of solutions that are Pareto optimal.

3 Optimization Logic

The algorithm we employ is based on modifications to the random search (RS) of Nesterov [3]. This method has favourable convergence guarantees when the function being minimized is convex, including cases when it is nondifferentiable or contaminated by (stochastic) noise. We have also observed reasonable empirical behaviour on several nonconvex problems. In each iteration, RS generates a stochastic direction, estimates the associated directional derivative, and takes a step along the direction that is scaled by this derivative. These basic operations can be arranged so that the two function evaluations in each iteration are performed concurrently. Furthermore, the directional derivative estimates are improved by using the modifications in [4], which compute and employ an estimate of the noise (whether deterministic or stochastic) in an objective of interest [5], so that the expectation of the estimation error is minimized.

Our approach to minimizing multiple objectives simultaneously is in part guided by our desire to exploit additional concurrency. In particular, we assume that the Pareto front is reasonably convex and hence can be recovered by solving a sequence of single-objective problems. Each single-objective problem is a different linear (convex) combination/weighting of the multiple objectives, f = w1f1+w2f2+…wmfm. Given a batch of weights—selected and scaled based on knowledge about the multiple objective functions and about previous single-objective runs—one can solve the associated single-objective problems concurrently, with the number of concurrent problems dictated by the available computational resources. Simulations run in previous batches can be exploited (under the current weights) to warm-start individual RS runs.

The overall logic of the algorithm comprises the following steps.

1. Begin at the best point from previous runs, or pick a random point. x=[x1,x2,……,xn].
2. Add a small step to get [xμ] along a random direction.
3. Try simulation with [x] and [xμ].

4. Get [f] & [fμ].
    a. For multiple objectives f = w1f1+w2f2+……+wmfm.
    b. [x] and [xμ] can be evaluated in parallel.
    c. Different processes in each iteration can be run in parallel.
    d. Different set of weights "w" can be run in parallel.
5. Compute the slope of "f" in the chosen direction.
6. Make a step along the direction as scaled by the slope, to arrive at the next [x].
7. Repeat 2, until the maximum number of iteration is reached.

4 Simulation Framework

Autonomie is a simulation tool developed at Argonne that allows users to plug in their specific models, drive cycles, and processes. Autonomie is built on Matlab, Simulink & Stateflow and can interface with many other simulation tools. Figure 3 shows the Simulink vehicle model built for this study. In addition to being a model-building platform, it provides a convenient framework to perform studies. Several optimization techniques, including the modified random search algorithm, are already integrated with Autonomie. The graphical user interface (GUI), makes it easy to define the optimization problem in Autonomie.

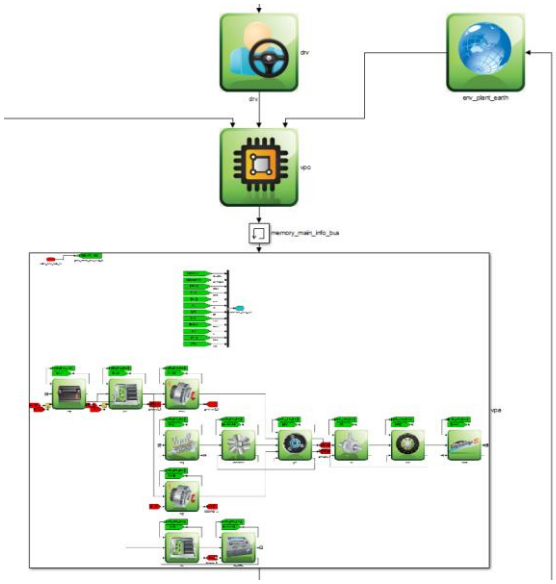

Figure 3. Vehicle controller and vehicle architecture is built using Simulink and Stateflow

The drive cycles over which this model should be simulated can be selected in Autonomie. This study uses two cycles: the UDDS and an acceleration test (0–60 mph). For each of these cycles, specific constraints and objectives are specified. Minimum fuel consumption is the goal, and SOC balance is a constraint for the UDDS cycle. The vehicle controller tried to enforce SOC balance, but this needs to be added as a constraint to ensure that the simulation results are valid. Similarly, the acceleration test has as a goal the minimum time taken to accelerate from 0 mph to 60 mph. All variables that are used in the vehicle or produced as output from the vehicle model are available through the GUI, which can be used to define the optimization problem. The constraints imposed on this study are listed below.

1. SOC balance for UDDS cycle for fuel economy runs
2. UDDS drive cycle trace, within +/- 2mph tolerance at all times
3. 0-60 mph acceleration, under 10 s

After defining the objectives and constraints for each cycle (see Figure 4), all the input variables and the allowable range of inputs need to be provided (see Figure 5).

If the variables are defined within Autonomie, then the editor shows the lower and upper limits that can be selected.

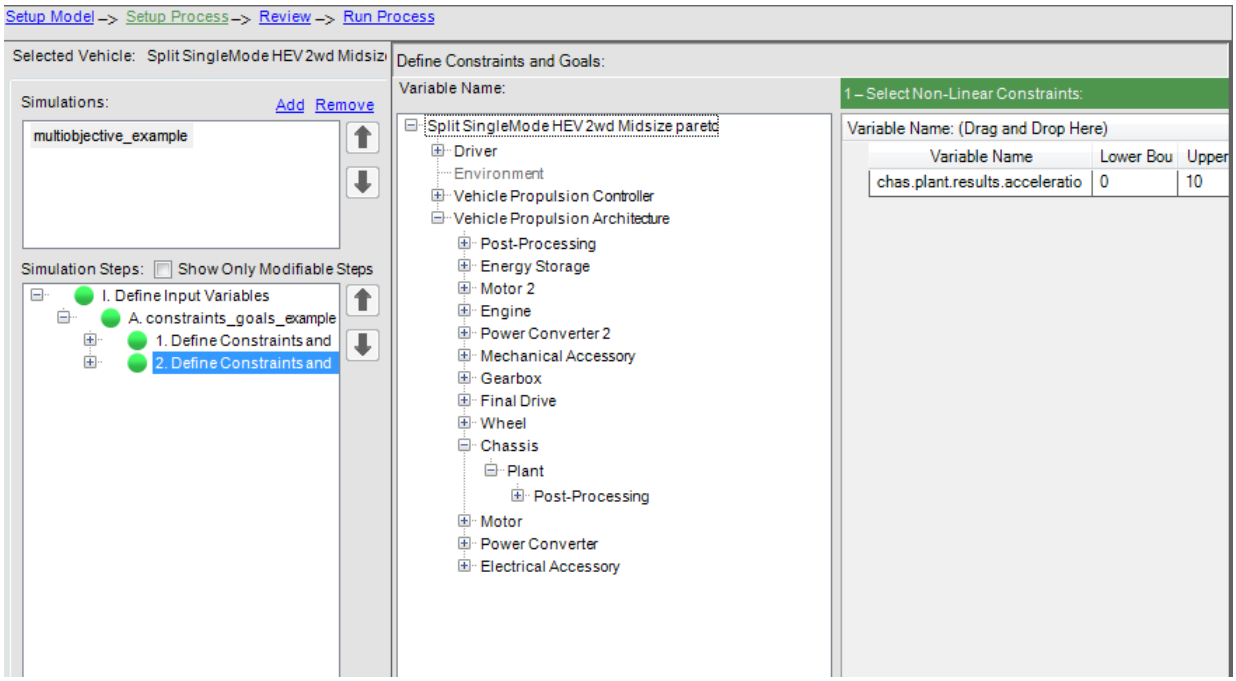

Figure 4. Defining constraints and objectives through the GUI

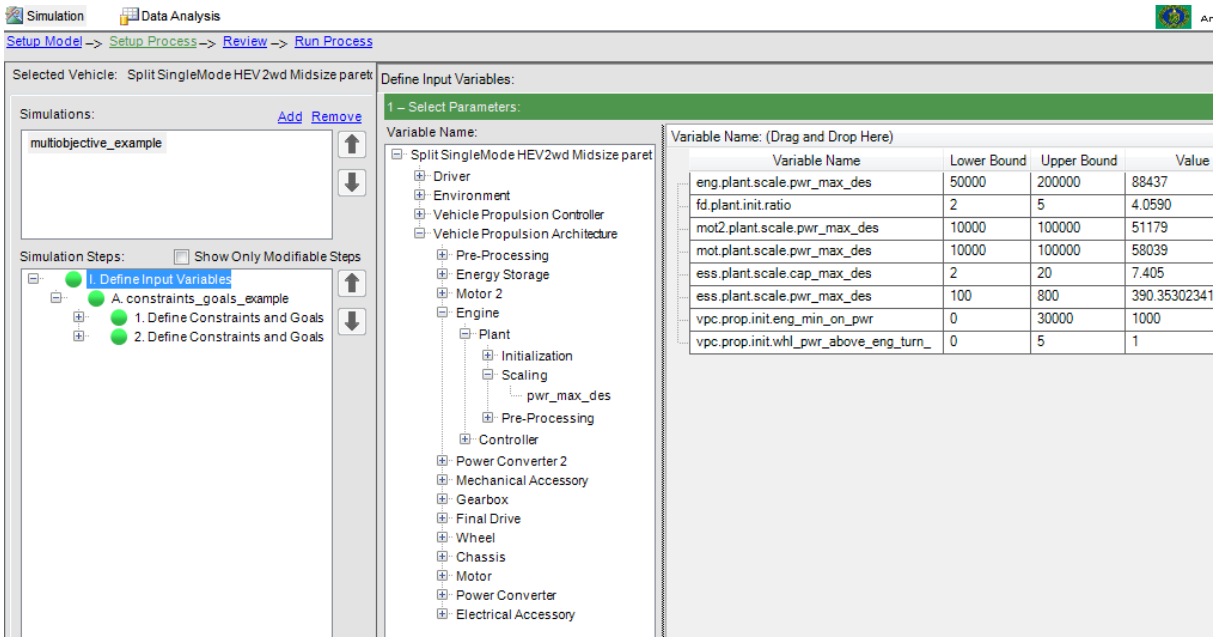

Figure 5. Defining all the input variables for the optimisation problem

5 Running the Simulations

The optimization routine can initiate parallel evaluations of different cycles. If the user has a multi core machine, parallel matlab sessions can be run locally in user's machine itself. If a license for the Matlab parallel computing toolbox and access to a distributed-computing cluster are available, then the simulations can be executed by using these resources. For this study, the simulations were run on a distributed-computing cluster. This study used a Windows-based system but the implementation was tested on linux based clusters too. Figure 6 shows the overview of how this parallel evaluations are achieved. The shared disc space should be accessible from both the distributed-computing cluster and the user's machine. The path for the common locations and files is defined for Windows and UNIX operating systems by using XML files.

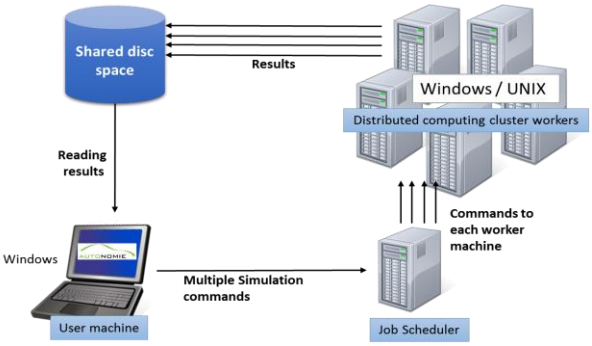

Figure 6. Overview of the distributed computing capability

The user's machine runs the optimization routine, which sends multiple simulation commands to the distributed-computing cluster through a Matlab job manager or a job scheduler used in the distributed-computing cluster. The simulation runs are carried out by the worker machines as scheduled by the job manager. When these simulations are completed, the results are written out to the shared disc space. The user's machine can read these results from the shared disk space, process the results, and specify with new simulation commands.

Our multiobjective optimization algorithm evaluates different weighted combinations of the various objectives. For any given weight, a single cost function is formed as F = w*F1 + (1-w)*F2. The weight, 'w' can be between 0 and 1, with the first objective given a weight w and the second objective given the weight (1-w). Each objective is then minimized by using our single-objective algorithm; see Figure 7.

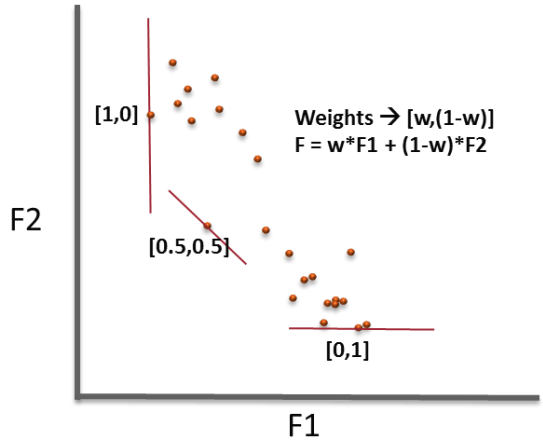

Figure 7. Multiobjective problem split into many single-objective problems with different weights that can be solved concurrently

Each of these weighted combinations can be treated as independent optimization runs. Each separate weight will initiate a separate Matlab session in the user's machine and two worker machines on the distributed-computing cluster. In our test case, the distributed-computing cluster provided 12 workers, to concurrently evaluate six weighted combinations of the two objectives.

6 Results

The optimization runs using the hybrid vehicle model show interesting behaviour. The optimization algorithm provided component sizes that are better suited for the operation of the vehicle over the UDDS cycle. In Figure 8, we plot the percentage change in both fuel consumption and vehicle cost on the x and y axes, respectively. The (0,0) point refers to the baseline vehicle. Any point that has positive x and positive y coordinates is worse than the default vehicle, as it has a higher fuel consumption and cost. All the other outputs present a design choice that has some advantage over the default vehicle.

The points that fall on the dotted black line correspond to the final relevant design choices. These choices illustrate the trade-off available between fuel economy and cost. For example, for a higher vehicle cost, an option exists for obtaining 15% better fuel consumption than the default vehicle; this option is represented by the point at (-15, 4). Similarly, for the same fuel economy as the default vehicle, one can obtain about 4% lower cost. There are also points such as (-15,-2.5), which offer lower fuel consumption as well as lower cost.

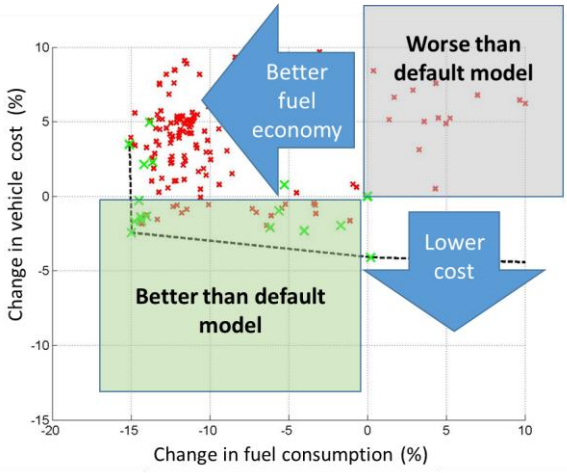

Figure 8. Interpreting the results shown in a Pareto diagram

In Figure 8, we have used three weights [0 0.5 1]. For each of these weights, the RS algorithm employed is run multiple times, with 20 iterations in each. If no improvement in the weighted objective function is noticed over the past 10 consecutive iterations, that particular run is terminated. The best point observed during that run

is marked with a green "x" in the plot. Then the algorithm looks at all the previous results for the best point to start a new run. While evaluating test cases for a particular w, say 1, where all weightage is given to minimizing fuel consumption, the randomness in the RS algorithm may produce an indeterminate design point that could end up as a very good point for a 0.5 weight (where fuel economy and cost are given equal weights). If the algorithm is looking to start a new set of runs for the w = 0.5, then it will begin from the previous point that was actually produced by a parallel run. This sharing of information between parallel runs helps reduce the number of iterations and time taken for each rerun, and hence the total time. Figure 8 shows more improvement on fuel consumption side than on the cost. More evaluations of weights closer to zero will explore the possible improvements in cost as well. The Pareto front is represented by the dotted block line in Figure 8. Several hundred more iterations are needed to obtain a more accurate Pareto front. The exact shape of the front will also depend on the behaviour of the vehicle model.

A Pareto front generated with Random Search algorithm may not show the globally optimal results, but this can provide certain feasible design points which will meet the design constraints. For example, every point on the black dotted line that is marked with a green 'x' in Figure 9 presents a point in the design space that provide some trade-off between vehicle cost and fuel consumption. There are design choices that can provide both cost savings and fuel savings too. Analysing the variation in input parameters for three sample points marked with large blue dots in the figure below can also shows us the trends. They represent the best points observed for the weights we used in this study, [0, 0.5, 1].

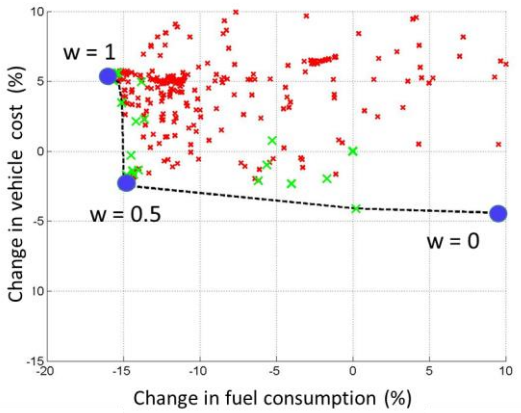

Figure 9. Viable design choices picked from the approximate Pareto front

These three points presents very different design choices. When looking for maximum fuel economy, the optimization logic will not try to save cost. Similarly in an effort to find the lowest cost, a single objective optimization algorithm will not explore options to avoid wasting of fuel. The results presenting tradeoffs between these two extreme positions would avoid unnecessary cost and fuel usage.

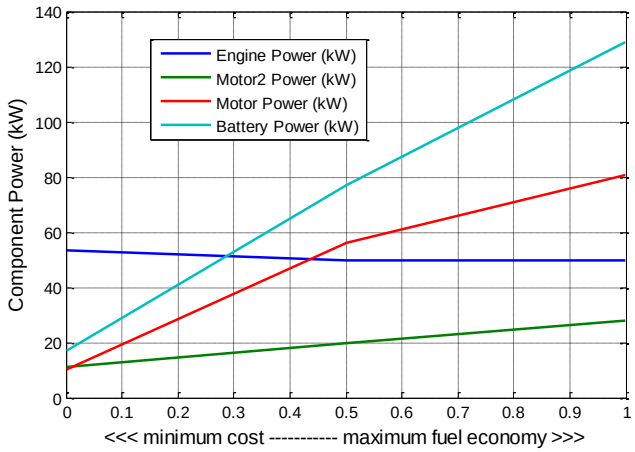

Figure 10. Variation in component powers during the fuel economy vs cost trade-off

On the left extreme of Figure 10, we have the design inputs for a low cost hybrid (w=0). The hybrid powertrain size is close to the minimum limits, but sized large enough to meet the performance requirement of reaching 60mph in under 10s. The middle of the plot, presents a balance of fuel economy and cost (w=0.5). We see larger sized motors and battery. As part of the effort to reduce cost, the engine power is scaled down to the minimum limit. Finally, when we focus just on improving fuel economy, (w=1) we get a solution with very large battery and motors. Parameters such as battery energy and final drive ratio remained almost the same for these solutions. While battery energy affects cost, it is overshadowed by the cost imposed by battery power. Hence these two factors remain at values that yield better fuel economy.

The control variables however show interesting response to varying component sizes. On the left extreme of Figure 11, where fuel economy was not a factor, we see that the minimum power demanded from the engine is not a critical factor. However as soon as we move to a region where fuel economy is given a positive weightage, the minimum engine operating power drops to about 2 kW.

The maximum power up to which the vehicle can operate in EV mode determines the power threshold at which engine will be turned on to provide propulsion power. This value is seen to

increase as the motor power and battery power increases. This brings out a relation between the control variable and component size. For maximum fuel economy the engine is turned on only if the power demanded by the wheels is greater than 50kW. While this may not result in the globally maximal fuel economy, the trends we see these design variables will be useful.

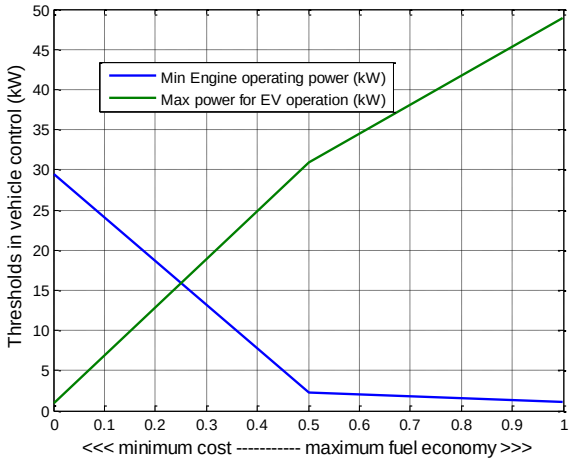

Figure 11. Variation in control parameters during the fuel economy vs cost trade-off

For better fuel economy, we see that the optimisation logic picked a higher degree of hybridization. In contrast, to reduce cost, the algorithm picked minimum values for the component sizes needed to achieve the performance requirements. In the actual vehicle, Engine power may not be reduced to such a low value because there are other performance considerations like ability to climb extended grades, or being able to operate with a faulty battery. The drive cycles and performance requirements we imposed on this study do not demand a big engine. In this study, performance requirement was provided as a constraint rather than an objective. With a different problem definition and imposition of more realistic operational constraints, one can size vehicles that trade-off all three objectives, fuel economy, performance and cost.

7 Conclusion

This paper shows how a multiobjective optimization problem is solved by using modern computational techniques and tools. Sizing of the hybrid vehicle to meet the fuel economy, cost and performance constraints is a complex task. Recent advances made in software, as well as the availability of multicore desktops and distributed computing facilities, make this study possible. Autonomie functions as a simulation framework that does model building and initiates simulations. It also handles the communication between the user's machine and the job manager/scheduler of the distributed-computing cluster.

This exercise resulted in obtaining three design choices that are better than the default vehicle in different aspects. To keep the visualizations simple, we considered just fuel economy and cost as objectives, but more objectives can easily be added.

Acknowledgments

We thank colleagues who helped during the various stages of this study. S.Pagerit, S.Halbach and M.Juskiewicz contributed to developing the GUI needed for defining the optimization problem in Autonomie. Mathworks supported this work by providing feedback on the distributed-computing techniques used in Autonomie.

This work was supported by the U.S. Department of Energy's Vehicle Technology Office. The submitted manuscript has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory ("Argonne"). Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.

References

1. Halbach, S., Sharer, P., Pagerit, P., Folkerts, C., Rousseau, A., Model architecture, methods, and interfaces for efficient math-based design and simulation of automotive control systems, SAE 2010-01-0241, SAE World Congress, Detroit, April 2010 (pdf)
2. M. Ehrgott. Multicriteria Optimization. Springer-Verlag, 2nd edition, 2005.
3. Y. Nesterov. Random gradient-free minimization of convex functions. CORE Discussion Papers 2011001, Universite Catholique de Louvain, Center for Operations Research and Econometrics (CORE), 2011. URL http://www.ucl.be/cps/ucl/doc/core/documents/coredp2011_1web.pdf
4. R. Chen. Derivative-free optimization of noisy function (Unpublished doctoral

dissertation). Lehigh University, Bethlehem, PA, USA.

5. J. J. Moré and S. M. Wild. Estimating derivatives of noisy simulations. ACM Trans. Math. Softw., 38(3):19:1–19:21, April 2012. doi:10.1145/2168773.2168777.

Authors

Ram Vijayagopal is a research engineer at Argonne National Laboratory. He works in the area of advanced vehicle technologies.

Ruobing Chen was a Given's Fellow in the Mathematics and Computer Science Division at Argonne during the time of this study. She received her Ph.D. in Industrial Engineering from Lehigh University, PA in 2015. Currently, she is a Data Scientist at Robert Bosch LLC, Palo Alto, CA.

Stefan Wild obtained his Ph.D. in operations research at Cornell University. He joined Argonne as Director's Postdoctoral Fellow in September 2008. His primary research focus is on algorithms and software for simulation-based optimization problems where derivatives of the objective are unavailable

Aymeric Rousseau received his Master of Science in industrial systems from EIGSI in La Rochelle, France, in 1997. He is currently leading Argonne National Laboratory's Vehicle Modeling and Simulation Group.

Phillip Sharer is senior developer of the Autonomie process architecture. He has been a research engineer at Argonne National Laboratory for 14 years. He received a Master of Science in Engineering from Purdue University, Calumet.