Direct Multi-objective Optimization through LS-OPT® using Small Number of Crashworthiness Simulations

Tushar Goel[1], Yih-Yih Lin[2], Nielen Stander[1]
[1]*Livermore Software Technology Corporation, Livermore CA*
[2]*HP, High Performance Computing Division, Richardson, TX*

Abstract

Genetic algorithms typically require a large number of simulations, which would be economically prohibitive for crash simulations without the advent of today's cost-effective multi-core computers. A study is conducted to seek improvements while restricting the number of simulations and exploiting the ability to use parallelization. The parallelization, achieved by simultaneously running multiple simulations for each GA generation on a HP quad-core cluster, resulted in a significant time savings. Furthermore, the optimal distribution of computational effort to achieve the greatest improvement in performance was explored. A crashworthiness simulation of a vehicle with 58,000 element finite element model was used as a test example. Various population sizes and numbers of generations were tried while keeping the total number of simulations constant. The optimization performance is also compared with Monte-Carlo and space filling sampling methods. It is observed that using GA, one can find many feasible and trade-off solutions. It is beneficial to allow a greater number of generations to get good trade-off solutions. Significant improvements in the performance were observed.

Introduction

Genetic algorithms (GAs) in LS-OPT [1] are used as a global optimization method for single- and multi-objective optimization. Since GA is a population-based method, it typically requires a large number of simulations to find an optimal solution. With expensive function evaluations such as vehicle crash simulations, the time to achieve a converged solution can therefore be unreasonably long. The only way to solve this predicament is to process the computational load in parallel with a large number of computer processors. However, a large number of computer processors could be economically prohibitive if the cost of each processor is high. Fortunately, the recent advent of multi-core computers have alleviated this cost concern, for a core in a dual-core-based computer is typically 30 percent less expensive than a single-core computer and a core in a quad-core computer is also typically 30 percent less expensive than a dual-core computer. There are two ways of parallelizing the GA-based optimization process. Firstly, multiple processors can be used to simultaneously analyze different designs (individuals in GA population). Secondly, multiple processors can be used for each simulation (LS-DYNA [2] MPP). These two steps can significantly reduce the 'wall time' of performing a GA simulation.

Furthermore, the performance of the GA might depend on the interplay between diversity and evolution characterized by population size and the number of generations, respectively. Obviously, a small population size would have a lack of diversity in the population and would be prone to convergence to a local optimal design (or local Pareto optimal front). On the other hand, a large population size would require significant computational effort for evolution. The influence of population size on performance has been studied in literature for single objective optimization but not for multi-objective optimization. This issue is particularly important for engineering problems like vehicle crash simulations, which almost always have multiple objectives and a limit on the number of simulations due to the high cost of each simulation.

This interplay between diversity, evolution, and computational expense is the subject of this paper. The computational expense is limited by fixing the number of simulations and the elapsed wall time is reduced by using a cluster of HP ProLiant servers with quad-core Xeon processors to parallelize the optimization process. Each GA generation is analyzed in parallel i.e., all the members in a population are simultaneously analyzed. The significant reduction in the optimization time enabled the study of the convergence properties of a multi-objective optimizer with variation in population size and number of generations.

The next section in the paper provides details of the simulation strategy and test metrics adopted in this study. The section **Test Example** describes the test problem and the test procedure. The details of the hardware system, software interface, and savings in wall time are given in the section **Simulation Details**. The results obtained are described and analyzed in the section **Results and Discussion**. The main findings are recapitulated in the last section.

Test Methodology and Test Metrics

An elitist non-dominated sorting genetic algorithm (NSGA-II) is used as a global optimization method in LS-OPT to solve multi-objective optimization problems. The details of this algorithm [3] and implementation in LS-OPT [4] appear elsewhere. The salient features of this algorithm are the introduction of elitism and use of crowding distance as a diversity preserving mechanism. In this study, a tournament selection operator with a tournament size of two, and real coded crossover and mutation operators are used to create a child population. The distribution index for the simulated binary crossover operator is taken as five. The crossover and mutation probabilities are also kept constant throughout the study.

Table 1 – Different configurations of population-size and number of generations. *Not a GA simulation.

Case	**Population size**	**# of generations**
Random*	1000	1
Space-filling*	1000	1
P20xG50	20	50
P40xG25	40	25
P50xG20	50	20
P100xG10	100	10

To study the impact of diversity and evolution on the performance of NSGA-II, various combinations of population size and number of generations are considered, as given in Table 1. The number of simulations is fixed to have equivalence among different simulations. A budget of a maximum of 1000 simulations is allocated for each simulation to keep the computational expense close to practical limits. Furthermore, the performances of NSGA-II simulations are compared with a random search method. Two strategies, Monte-Carlo method and Space filling design, were used to select 1000 random points in the design space. The simulations for each of these methods are carried out on an HP quad-core cluster. The details of the hardware and software used to conduct simulations are detailed in section on the **Simulation Details**.

Unlike single objective optimization problems where the optimum design is a single solution, multi-objective optimization results in a set of optimal solutions. Thus, special test metrics are needed to compare the results from different simulations. Typically, two criteria, convergence to the Pareto optimal front, and the diversity on the Pareto optimal front, are used to compare the multi-objective optimization results. To compare the two sets of optimal designs, the number of solutions that are dominated in each set is computed using a weak non-domination criterion. The smaller the number of dominated solutions, the better is the convergence property. Secondly, the diversity is characterized by a uniformity measure [3], defined as

$$\Delta = \sum_{i=1}^{N} \frac{\left| d_i - \bar{d} \right|}{N}; \bar{d} = \frac{1}{N} \sum_{i=1}^{N} d_i . \qquad (1)$$

where d_i is the crowding distance of the solution in the function or variable space. The boundary points are assigned a crowding distance of twice of the distance to the nearest neighbor. Two uniformity measures, one in the function space and another in the variable space are considered in this study. A small value of the uniformity measure is desirable.

Test Example

This optimization study is carried out by considering a crashworthiness simulation of a National Highway Transportation and Safety Association (NHTSA) vehicle undergoing a full frontal impact. The finite element model for the full vehicle, containing approximately 54,800 elements, is shown in Figure 1. The LS-DYNA explicit solver is used to simulate the crash.

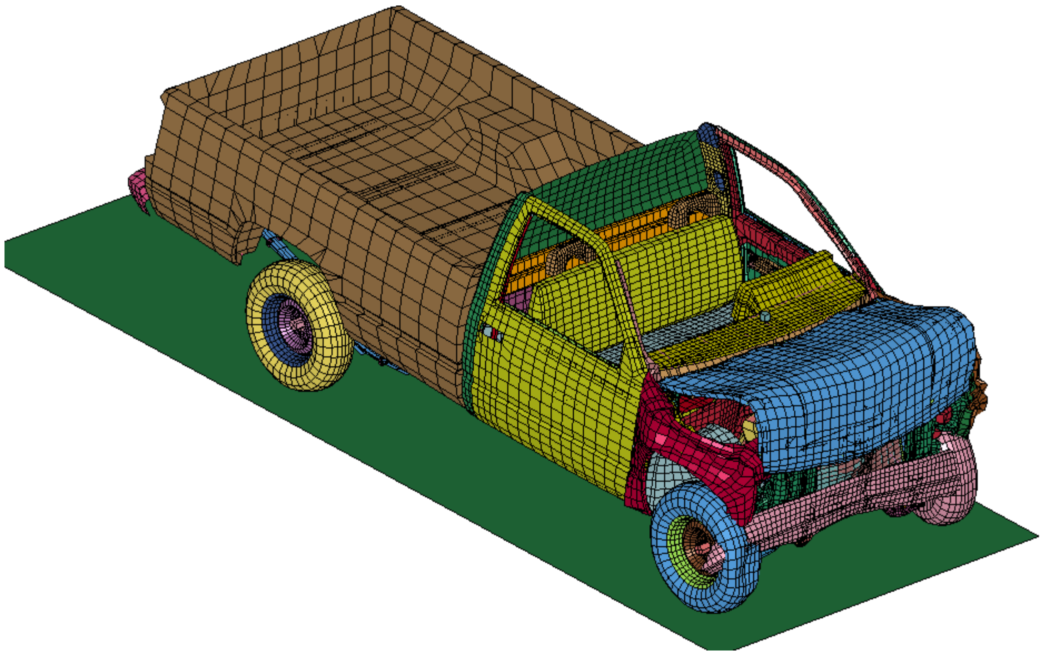
Figure 1: Finite element crash model of a pickup truck

The gauges of structural components in the vehicle are parameterized directly in the solver input file. Nine gauge thicknesses associated with front-right-inner, front-right-outer, front-left-inner, front-left-outer, back-left and back-right rails, bumper, bottom-under radiator MTG, and bottom-center cabin member, are taken as design variables. The parts affected by the design variables are shown in Figure 2. The range of these design variables is chosen as within +/-20 % of the baseline design variable values. The baseline design and the bounds on the variables are given in Table 2.

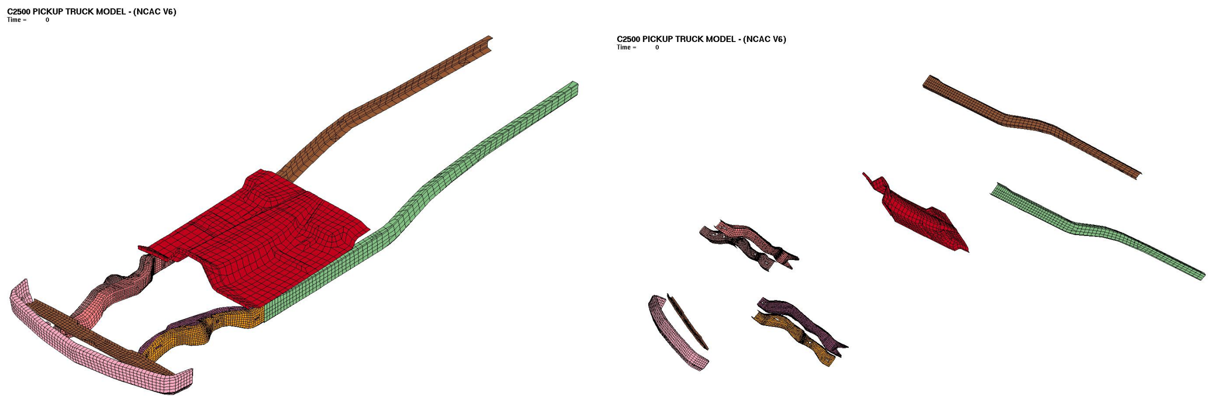
C2500 PICKUP TRUCK MODEL - (NCAC V6)
Time = 0
C2500 PICKUP TRUCK MODEL - (NCAC V6)
Time = 0

Figure 2: Thickness design variables (with exploded view)

Table 2 – Baseline design and bounds on design variables.

Variable description	Name	Lower bound	Baseline design	Upper bound
Rail front-right-inner	T1	2.500	3.137	3.765
Rail front-right-outer	T2	2.480	3.112	3.750
Rail front-left-inner	T3	2.400	2.997	3.600
Rail front-left-outer	T4	2.400	3.072	3.600
Rail right-back	T5	2.720	3.400	4.080
Rail left-back	T6	2.850	3.561	4.270
Bumper	T10	2.160	2.700	3.240
Radiator bottom	T64	1.000	1.262	1.510
Cabin bottom	T73	1.600	1.990	2.400

The crash performance of the vehicle is characterized by considering the maximum acceleration, maximum displacement that links to intrusion, time taken by the vehicle to reach zero velocity state, and different stage pulses. These responses are taken at the accelerometer mounted in the middle of the front seat. To reduce the influence of numerical noise, filtered acceleration is considered and different entities are averaged over two accelerometer nodes. While constraints are imposed on some of these crash performance criteria like stage pulses, it is desirable to optimize the performance with respect to other criteria. Thus a multi-objective optimization problem can be formulated as follows:

Minimize

Mass and peak acceleration;

Maximize

Time-to-zero-velocity and maximum displacement;

Subject to constraints on variables and performance.

The design variable bounds are given in Table 2 and the performance constraints, namely maximum displacements and stage pulses, are specified in Table 3.

Table 3 – Design constraints.

	Lower bound	Upper bound
Maximum displacement (x_{crash})	-	721 mm
Stage 1 pulse(x_{crash})	-	7.48 *g*
Stage 2 pulse(x_{crash})	-	20.20 *g*
Stage 3 pulse(x_{crash})	-	24.50 *g*

The three stage pulses are calculated from the averaged SAE filtered (60Hz) acceleration $\ddot{x}$ and displacement *x* of the accelerometer nodes in the following fashion:

$$\text{Stage } j \text{ pulse} = -\frac{k}{(d_2 - d_1)}\int_{d_1}^{d_2} \ddot{x}\,\mathrm{d}x \quad k = 0.5 \text{ for } j = 1, k = 1.0 \text{ otherwise}; \qquad (2)$$

The integration limits (d_1;d_2) = (0;200); (200;400); (400;Max(displacement)) for j = 1, 2, 3 respectively, represent different structural crash events. All displacement units are mm and the minus sign is used to convert acceleration to deceleration.

During optimization, all objectives and constraints are scaled to avoid dimensionality issues.

Simulation Details

A 640-core HP XC cluster, comprising 80 ProLiant server nodes of two Intel Xeon 5365 quad-core processors (also known as Clovertown, with 2 processors/8 cores), with a 3.0 GHz clock rate, is used in this investigation. HP XC System Software is an HP supported software stack that can be used for the operation and management of HP clusters. XC software includes a Linux distribution designed to be compatible with Red Hat Enterprise Linux Advanced Server. The cluster is shared by many users, and its job flows are scheduled by Platform LSF, which is also included in XC software. Additionally, the cluster is configured with a HP Scalable Files System whose size is 32 TB.

Each crash simulation described in the section **Test Example** is an explicit LS-DYNA simulation, which can be either serial or parallel. We use the serial LS-DYNA simulation in this investigation to obtain better performance, for the elapsed time of an LS-OPT case corresponds to the throughput time, on which the serial LS-DYNA has advantage over the parallel LS-DYNA. As aforementioned, the cluster is shared by many users and under the control of LSF. At the start of each generation, LS-OPT submits to the LSF queuing system the same number of serial jobs as the size of population, N; the next generation is started only after LS-OPT makes sure all jobs in the current generation have finished. There are two methods for a queuing system to allocate compute resource for each generation: the one-simulation-one-allocation method and the many-simulations-one-allocation method. The former is the traditional method, in which each serial simulation is submitted to the LSF queuing system, requesting allocation of a single core (or CPU) by invoking the command "*bsub –n 1 run-serial-job*;" while in the latter method, all necessary computer resource, say M cores are allocated by the command "*bsub –n M run-all-serial-jobs-simultaneously*." (The number M must be equal to or greater than the population size

N.) The traditional one-simulation-one-allocation method is used in this investigation. The advantage and disadvantage of the two methods will be discussed in details in the next section **Results and Discussion**.

Each serial simulation produces an output of 225 MB; and as a result, each of the six cases, requiring 1000 simulations, produced a total output of 225 GB. This demand for a large amount of disc space is met by the 32 TB in the aforementioned HP Scalable File System.

Results and Discussion

Elapsed time with one-simulation-one-allocation method in a shared cluster

Table 4 shows the elapsed times, actually measured, for the 6 cases using the one-simulation-one-allocation method. These measured elapsed times can be understood in terms of the throughput time in a node, which is the time for all jobs, running simultaneously, to finish in the node. The throughput time for a multi-core node is known to depend on the workload: Too much workload will inevitably elongate it. Table 5 shows the throughput times for a Clovertown node with 1, 2, 4, and 8 cores loaded. Note that a node is half populated when 4 cores are loaded, and is fully populated when 8 cores are loaded. Each generation of the four GA cases takes about 5.7 hours, which is close to the throughput time, 5.2 hours, for a fully populated node. This reflects that most allocated nodes in each generation are fully populated. The difference of 0.4 hour in elapsed time per generation and the throughput time is inevitable due to the use of the one-simulation-one-allocation method, since each allocation can not always be instantly satisfied in a shared queuing system. Both random case and space-filling case has only one generation and takes 16 hours to finish. Note that 16 hours is thrice that of the throughput time. If no other user were sharing the cluster, it would take two LSF-allocation rounds to finish for either case, since the cluster has 640 cores. That it takes an extra LSF-allocation round to finish is because the cluster is shared by other users.

Table 4 – Measured elapsed times for the six GA simulation cases.

Case	Population size	Number of generations	Elapsed time (h)	Ave. elapsed time per generation (h)
Random	1000	1	16	16
Space-Filling	100	1	16	16
P20xG50	20	50	334	6.68
P40xG25	40	25	145	5.8
P50xG20	50	20	120	6.0
P100xG10	100	10	54	5.4

Predicted elapsed time with the many-simulation-one-allocation method in a shared cluster

As the previous subsection indicates, the throughput time is the sole factor for determining the elapsed time for an LS-OPT case. From Table 5, it can be seen that the optimal throughput time for an LS-OPT case is when all allocated nodes are half-populated, taking 3.1 hours to complete a generation. However, because most queuing systems have a fill-up allocation scheduling policy, using the one-simulation-one-allocation would most likely result in fully-populated allocated nodes, taking 5.2 hours to complete a generation. This situation can probably be avoided by the many-simulation-one-allocation method, in which a sufficient number of nodes are allocated to satisfy the half-populated-node condition from the start and lasting until the end of the LS-OPT simulation. Applying the many-simulations-one-allocation method optimally, the predicted elapsed times for the 6 cases are shown in Table 6. For both random and space-filling cases, the number 4 is chosen as the number of LSF-allocation rounds; the choice corresponds to an allocation of 250 cores per round, which is dictated by the maximal available number of cores, 640, and the half-populated-node condition. Note that these predicted times are significantly faster than the measured elapsed times with the one-simulation-one-allocation method by a factor of at least 1.7.

Table 5 – Elapsed time for a serial simulation in the 2-processor/8-core Clovertown, running throughput mode.

Number of Simulations	**Elapsed Time (h)**
1	2.8
2	3.1
4	3.1
8	5.2

Table 6 – Predicted optimal elapsed time, using the method of N simulation-one allocation.

Case	**Population size**	**Number of generations**	**Number of cores allocated**	**Number of LSF-allocation rounds**	**Predicted Elapsed time (h)**
Random	1000	1	500	4	12.4
Space-Filling	100	1	500	4	12.4
P20xG50	20	50	40	50	155
P40xG25	40	25	80	25	77.5
P50xG20	50	20	100	20	62
P100xG10	100	10	200	10	31

Comparison of optimization results

A comparison of optimization results obtained from various optimization runs is given in Table 7. While the GA runs are required to use 1000 crash simulations, the actual number of crash simulations is slightly less because calculations for duplicate designs are avoided. The number of feasible designs from the GA runs (not random/space-filling method) is significantly higher than the random selection or space-filling sampling methods. In the current study, allowing more generations (evolution) was useful though the simulation for the case P40xG25 resulted in fewer feasible solutions compared to the case P50xG20. Overall this result is not surprising because the current GA implementation focuses on identifying feasible regions in the design space.

The large number of feasible points did not guarantee more trade-off solutions (TOS) as was evident from the results. Among all GA simulations, the ratio of number of TOS and feasible solutions was the highest for the case P40xG25 and the least for the case P20xG50. The maximum number of TOS was obtained for the case P50xG20. As expected, all GA runs yielded more trade-off solutions than the random sampling or space-filling method.

Table 7 – Comparison of different simulations. POS: Pareto optimal solutions, Δ-uniformity measure.

Case	**Actual # of simulations**	**Feasible points**	**# of candidate POS**	**True candidate POS**	**Δ-var**	**Δ-obj**
Random	1000	2	2	0		
Space Filling	1000	5	4	0	0.076	0.542
P20xG50	997	150	26	25	0.405	0.18
P40xG25	996	67	31	20	0.349	0.191
P50xG20	995	99	32	9	0.274	0.121
P100xG10	995	46	20	15	0.27	0.204

The uniformity measure (Table 7) that characterizes diversity on the Pareto optimal front indicated that the case P50xG20 offered the maximum diversity in trade-off solutions in both variable and function spaces. The other GA simulations have comparable diversity on the Pareto front. Note that, this measure is not evaluated for the random sampling method because of the small number of candidate Pareto optimal solutions.

Table 8 – Relative quality of candidate Pareto optimal solutions obtained from various simulations. A number in (i, j) location of matrix denotes the number of POS of case 'j' (column number) that are dominated by the POS of case 'i' (row number).

Case	**Random**	**Space Filling**	**P20xG50**	**P40xG25**	**P50xG20**	**P100xG10**
Random	0	0	0	0	0	0
Space Filling	0	0	0	0	0	0
P20xG50	2	2	0	7	12	5
P40xG25	2	3	1	0	16	5
P50xG20	2	2	0	2	0	5
P100xG10	1	2	0	4	9	0

A better indicator of the convergence is obtained by performing a non-domination check over all TOS from all six simulations. There were a total of 69 true candidate Pareto optimal solutions. It is obvious from Table 8 that the TOS obtained from random and space-filling sampling methods were poor. Interestingly, the GA runs that allow more evolution contributed more solutions to the true candidate Pareto optimal solutions set, which clearly indicates merits of evolution. The result for case P50xG20 was an exception because many TOS were dominated by other TOS from other GA runs.

A relative measure of the performance of various GA runs is shown in Table 8. One can see that the designs obtained from random Monte-Carlo sampling or space-filling method were the worst because these designs were not better than any GA run. Among all GA runs, clearly the trade-off solution set from the case P20xG50 was the best and the case P40xG25 performed comparably; while the case P50xG20 performed the worst.

Table 9 – Design variables corresponding to the selected optimal design.

Case	T1	T2	T3	T4	T5	T6	T10	T64	T73
Random	2.693	3.247	2.618	3.125	3.870	3.678	2.391	1.330	2.376
Space Filling	2.670	2.917	3.548	2.509	2.724	3.727	2.271	1.104	1.946
P20xG50	3.165	2.710	2.767	2.834	2.829	3.829	2.505	1.345	2.357
P40xG25	3.340	2.679	2.652	2.828	2.978	3.916	2.464	1.215	1.894
P50xG20	2.899	3.197	2.762	2.655	3.036	2.963	2.250	1.331	2.343
P100xG10	2.708	3.018	3.256	2.737	3.478	3.843	2.510	1.504	2.347

To compare the performance of a single design selected from the corresponding trade-off solution sets, a weighted sum of objectives is maximized, when unit weight is assigned to each objective. The design variables, objectives, and constraints corresponding to the optimal designs for all optimization runs are shown in Table 9 and Table 10, respectively. The performance of the baseline design is also shown. It is obvious that no design variable hit the bound. The baseline design was infeasible and resulted in a high peak acceleration. The designs obtained from different optimization methods were feasible and had improved performance. The optimal solutions obtained from different GA simulations were significantly better than the baseline design and designs obtained by sampling. Significant reductions in peak acceleration and increase in time-to-zero-velocity were obtained while approximately maintaining the same or better mass and maximum displacement values.

Table 10 – Performance of selected optimal design for different simulations and baseline design. SP: Stage pulse.

	Objectives				Constraints			
Case	X_{crash}	**Accel**	**Mass**	**Time-to-zero -velocity**	X_{crash}	**SP-1**	**SP-2**	**SP-3**
Random	719.3	112019	1.819	0.084	719.3	6.96	19.87	23.82
Space Filling	719.0	111248	1.803	0.076	719.0	7.02	20.15	23.57
P20xG50	719.3	53645	1.812	0.098	719.3	7.29	19.96	24.28
P40xG25	720.2	52667	1.806	0.095	720.2	7.43	20.02	22.15
P50xG20	717.3	80132	1.806	0.104	717.3	6.86	20.13	23.61
P100xG10	720.0	65615	1.818	0.100	720.0	7.21	19.84	24.08
Baseline	*711.1*	*116601*	*1.812*	*0.094*	*711.1*	*7.90*	*21.18*	*25.23*

Overall, the results show merits of using evolution for multi-objective optimization. For the same amount of computational effort, the evolution helped find more feasible solutions and yielded many trade-off solutions. Probably, the GA simulation P20xG50 performed the best for this example. In general, there may be some variability in the performance because of the dependence on initialization procedures.

Conclusions

The results shown here offer some very interesting conclusions:

1. The capability to run a GA in parallel, which has become cost effective only lately by the advent of multi-core machines, reduces the elapsed time for optimization.

2. The many-simulation-one-allocation method might be much more efficient than the one-simulation-one-allocation for a shared cluster with a queuing system.
3. Very few feasible solutions from random sampling or space filling experimental designs indicated the presence of large infeasible regions.
4. As expected, for the same number of crash simulations the GA yielded a much greater number of feasible and better trade-off solutions compared to the random sampling or space filling designs.
5. The trade-off between population size and generation number for a fixed budget of simulations was not very clear though the results indicated benefits of allowing more evolution.
6. For this example, the GA run with a population size of 20 that is evolved for 50 generations provided the greatest number of feasible and trade-off solutions though the uniformity measures of these individuals were slightly inferior to the other GA simulation with a population size of 50. Nevertheless, the quality of solutions favored the former GA run.
7. Optimized designs significantly reduced peak acceleration and increased time-to-zero-velocity while keeping the mass of the vehicle and the maximum displacement nearly constant.

We note that the above results might have some sensitivity to the choice of initial random seed but the conclusions would largely be valid.

Acknowledgements

The authors would like to thank Subramani Balasubramanyam, Dilip Bhalsod, and Marko Thiele for their help in setting up the test problem.

References

1. Stander, N., Roux, W.J., Goel, T., Eggleston T. and Craig, K.J. LS-OPT® Version 3.3 User's Manual, Livermore Software Technology Corporation, October 2007.
2. Hallquist, J.O. LS-DYNA® User's Manual, Version 971.
3. Deb, K, Multi-Objective Optimization Using Evolutionary Algorithms, Wiley, 2001, New York.
4. Goel T, Stander N, Multi-objective Optimization with LS-OPT, 6th German LS-DYNA Forum, Frankenthal Germany, Oct 11-12 2007.