A NOTE ON MULTI-OBJECTIVE MATHEMATICAL PROGRAMS

Isaac Siwale[1]

Abstract

This note introduces GENO — a commercial solver for multi-objective, non-linear mathematical programs of various types. Four examples are presented that demonstrate GENO's efficacy at finding single-point solutions to the multi-objective problem. Validation of the solutions obtained is provided in an ad hoc fashion.

Key Words: Multi-objective Programming, Multi-criteria Programming, Non-linear Programming.

1 Introduction

Real-world practical problems are often characterised, not by a single objective, but by a set of criteria against which a solution should be assessed. Such problems are known as *multi-objective* or *multi-criterion* optimisation problems. They may formally be defined as the task of finding a *feasible* vector, **x*,** that *simultaneously* optimises the *p* components of an objective or criteria vector, **J**, viz.:

$$\underset{\mathbf{x}\in\mathbf{U}}{\text{Opt}}\ \mathbf{J} = \left(J^1(\mathbf{x}), J^2(\mathbf{x}), J^3(\mathbf{x}), \cdots, J^p(\mathbf{x})\right)^T, \quad p \in \mathbf{N}$$

But whereas the idea of "an optimal solution" is a well-defined concept in the uni-objective context (i.e. to get the "optimal solution" one maximises or minimises a criterion function), the same cannot be said of the multi-objective case. In general, the functions comprising the vector **J** of a multi-objective optimisation problem rank the feasible points in disparate ways: taken pair-wise, some may "compete" in the sense that an improvement with respect to one criterion requires degradation in the quality of the solution as assessed by another; others may "collude" in that an improvement in one entails the same in the other; and yet others may be totally independent. Thus, compared to the uni-objective problem, the mechanism of optimisation in the multi-objective case is no longer "uni-directional" and notion of "an optimal solution" is not so clear-cut. Instead, one encounters several types of solutions, most of which require the model-user (i.e. the person for whom the computed solutions are intended) to articulate his preferences as regards the components of **J** at some stage during the solution process. The traditional approach to solving multi-objective problems thus comprises two fairly distinct stages: a 'search procedure' and a 'decision-making' process. Depending on how the two are combined, three classes of methods are discernable, namely:

1. *A priori Articulation of Preferences*. [Decide-then-Search] The model-user expresses his preferences in terms of an aggregate utility function *prior to optimisation*.
2. *A posteriori Articulation of Preferences*. [Search-then-Decide] The model-user is presented with a set of 'efficient solutions' — a notion normally attributed to Vilfredo Pareto (Coello Coello, 2005). The model-user proceeds to select *a* solution from the given set.
3. *Progressive Articulation of Preferences*. [Decide-and-Search] Preference articulation by the decision maker and solution generation proceed in parallel at inter-leafed steps.

[1] Ike's Research Ltd, 265 Southcroft Road, London, SW16 6QT, England. email: ike@siwale.fsbusiness.co.uk

Most classical research on the numerical solution of multi-objective optimisation problems tended to concentrate on devising techniques for generating the entire set of efficient solutions (the 'Pareto frontier') in a *search-then-decide* solution strategy. The algorithms always begin with a "scalarisation" of the objective vector by some means or other, i.e. the weighted-sum method, ε-pertubation method, Tchybeshev method, Min-Max method, Goal Programming, and so on. These classical methods have some major drawbacks: (1) "scalarisation" of the objective vector means that the resulting solution to the resulting uni-objective problem is dependant on the subjective parameters used in the "scalarisation" process itself; (2) the associated algorithms generate only one solution at a time (which is hopefully on the Pareto-frontier), and so in order to generate a set of such Pareto-optimal solutions (as is required by the classical solution strategies mentioned above), the algorithm has to be applied many times; (3) the efficacy of most of the algorithms is dependant on the shape of the Pareto frontier;[2] (4) if the number of objectives is large, then the computational effort required to generate an efficient set of solutions can be very substantial; (5) presenting a visually simple solution set to the model-user (as required by at least the last two solution strategies outlined above) is difficult except in the bi-objective case.

In summary, classical solution techniques to the multi-objective problem are doubly inadequate: quiet apart from the fact that generating the entire Pareto frontier may be a wasted effort since in the end only one solution is required, the model-user may not be able to select a solution that properly represents his or her preferences after all. New techniques that ease these problems somewhat are however continually being devised. In particular, evolutionary algorithms adequately alleviate most of the difficulties associated with classical solution techniques that have been identified above (see Deb, 2001). Unfortunately, none of the evolutionary algorithm designs that I am aware of are designed to produce a single-solution in a single run; rather, their emphasis is in generating a well-sampled Pareto frontier to present to the model-user because they are all based on either the *decide-then-search* or the *decide-and search* solution strategies. The issue of incorporating the model-user's preferences is an active area of research. But although new designs of multi-objective evolutionary algorithms seem to be going in that direction (Deb, 2006), and given the severe limitations that result from the problem outlined as item 5 above, the case for designing algorithms that generate a single-point solution is strong.

The purpose of this note is to introduce to practitioners a commercial solver called GENO.[3] Inspired by Game-theoretic concepts,[4] the GENO solver implements an evolutionary algorithm that readily computes single-point solutions to multi-objective optimisation problems using the *Equilibrium* and *Compromise* solution concepts. No formal proofs are offered in support of this assertion. Rather, the approach taken here is to show, either via analysis or by comparison to some established results, that the GENO solution is indeed what it purports to be in each case. A brief description of the code's capabilities follows.

---

[2] Kasprzak, E. M. and K. E. Lewis (2001) write: "The weighted-sum method of generating Pareto sets was shown to work well with convex problems decades ago by Geoffrion (1968) and while it is still a very popular method its deficiencies have been noted. Messac, et al. (2000) have effectively illustrated the problems associated with choosing weights for an aggregate objective function and has derived conditions that predict which Pareto solutions can be found using weights. Das and Dennis (1997) note that even with convex problems, taking an even spread of weights will not result in an even spread of points in the Pareto set and this renders some sections of the Pareto frontier difficult to populate" (Paraphrased from p. 3).

[3] GENO is an acronym for ***G***eneral ***E***volutionary ***N***umerical ***O***ptimiser.

[4] Surprisingly, the intuitively obvious connection between game theory and the general multi-criteria optimisation problem has not been exploited as much as one would expect. Of late however, this appears to be changing: see e.g. Basar, T. and Olsder, G. L. (1982); Coello (1996, Chap.2); Andersen, K. A. and M. Lind (1999); Sefrioui, M. and J. Periaux (2000); Conley, J. P., *et al*. (2000).

2 The GENO Code

GENO is a real-coded genetic algorithm that can be used to solve uni- or multi-objective optimisation problems. The problems presented may be static or dynamic in character; they may be unconstrained or constrained by equality or inequality constraints, coupled with upper and lower bounds on the variables. The variables themselves may assume real or discrete values in any combination.

Although the generic design of the algorithm assumes a multi-objective dynamic optimisation problem, GENO may be "specialized" for other classes of problems such as the general static optimisation problem, the mixed-integer problem, and the two-point boundary value problem, by mere choice of a few parameters. Thus, not only can GENO compute different types of solution to multi-objective problems, it may also be set to generate real or integer-valued solutions, or a mixture of the two as required, to uni-objective static and dynamic optimisation problems of varying types. These properties are easily pre-set at the problem set-up stage of the solution process. A detailed description of the algorithm is beyond the scope of this note:[5] rather, the aim here is to demonstrate its capabilities via several numerical examples as follows.

3 Examples

Two types of single-point solutions are presented: Example 1 computes the compromise solution of a bi-objective problem; Example 2 presents comparative results on a well known equilibrium problem; Example 3 presents the compromise solution of a simple test problem; and Example 4 presents a new benchmark solution for a mixed-variable nonlinear program via the equilibrium solution concept of John Nash (1951).

Example 1 [**Source**: Fonseca and Flemming (1995)]

$$\underset{\mathbf{x}}{\text{Opt}}\ \mathbf{f}(\mathbf{x}) = \{f_1(\mathbf{x}), f_2(\mathbf{x})\}$$

Subject to: $\mathbf{x} \in [-2, 2]$

Where: $f_1(\mathbf{x}) = 1 - \exp\left\{-\sum_{i=1}^{n}\left(x_i - 1/\sqrt{n}\right)^2\right\}$;

$f_2(\mathbf{x}) = 1 - \exp\left\{-\sum_{i=1}^{n}\left(x_i + 1/\sqrt{n}\right)^2\right\}$;

$n = 8$.

[5] A free trial-version of the program can be obtained by contacting current vendors at: info@Aptech.com

I. **GENO Output**

Generation	*Objective [1]*	*Objective [2]*
0	1.000000	0.999912
10	0.632121	0.632121
20	0.632121	0.632121
30	0.632121	0.632121
40	0.632121	0.632121
50	0.632121	0.632121
60	0.632121	0.632121
70	0.632121	0.632121
80	0.632121	0.632121
90	0.632121	0.632121
100	0.632121	0.632121

Solution Vector: $\mathbf{x} = (0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000)^T$

Objective Function Value: $\mathbf{f}(\mathbf{x}) = (0.632121, 0.632121)^T$

II. **Remarks**

Preamble

For a multiple objective optimisation problem, the solution ideally ought to be a win-win situation in which each individual objective function attains its best possible value within the constraints of the problem. But, as the name implies, the 'ideal solution' is usually unattainable; all one can hope for is to numerically approach such a point as closely as possible. The 'Euclidean compromise' is a solution concept for the multi-objective optimisation problem that is based on the notion of 'being close to ideal'. The Euclidean compromise solution (ECS) is that point on the Pareto frontier that is closest to the ideal solution, as measured by the Euclidean distance metric. This example serves to illustrate the effectiveness of GENO at computing the ECS of a multiple objective optimisation problem. The example is simple enough to afford an analytical determination of the ECS, against which the performance of GENO may be measured.

The Euclidean Compromise Solution: An Analytical Approach

In order to compute the Euclidean compromise solution, one needs to know the location of the 'ideal point' in the space of objective function values (hereafter called the 'space of outcomes'). For this example, 'Opt' operator denotes minimisation, and it is easy to show that, for all n, the minimum of f_1 is 0, and this is located at:

$$(x_1,..,x_n) = \left(1/\sqrt{n},..,1/\sqrt{n}\right). \tag{1a}$$

Similarly, the minimum of f_2 is 0, and this is located at

$$(x_1,..,x_n) = \left(-1/\sqrt{n},..,-1/\sqrt{n}\right). \tag{1b}$$

The ideal solution is therefore the point (0, 0) in the space of outcomes, and clearly, this is unattainable. To ascertain the point on the Pareto frontier that is closest to the ideal, one may proceed as follows.

According to Fonseca and Fleming (1995, p.51), the Pareto frontier is the set of all vectors **x** such that:

$$(x_1 = x_2 = ... = x_n) \wedge \left(-1/\sqrt{n} \le x_1 \le 1/\sqrt{n}\right). \tag{1c}$$

The distance to the ideal point in the space of outcomes is given by:

$$R(\mathbf{x}) = \sqrt{\left(f_1^2(\mathbf{x}) + f_2^2(\mathbf{x})\right)}\,. \tag{1d}$$

The minimum distance occurs at a solution to the following equations:

$$\frac{\partial R(\mathbf{x})}{\partial x_i} = \frac{1}{2\sqrt{\left(f_1^2(\mathbf{x}) + f_2^2(\mathbf{x})\right)}}\left(2f_1(\mathbf{x})\frac{\partial f_1(\mathbf{x})}{\partial x_i} + 2f_2(\mathbf{x})\frac{\partial f_2(\mathbf{x})}{\partial x_i}\right) = 0\,, \quad \forall i \tag{1e}$$

where:

$$\frac{\partial f_1(\mathbf{x})}{\partial x_i} = 2\left(x_i - 1/\sqrt{n}\right)\exp\left\{-\sum_{i=1}^{n}\left(x_i - 1/\sqrt{n}\right)^2\right\}, \quad \forall i \tag{1f}$$

and:

$$\frac{\partial f_2(\mathbf{x})}{\partial x_i} = 2\left(x_i + 1/\sqrt{n}\right)\exp\left\{-\sum_{i=1}^{n}\left(x_i + 1/\sqrt{n}\right)^2\right\}, \quad \forall i \tag{1g}$$

It follows that:

$$\frac{\partial R(\mathbf{x})}{\partial x_i} = 0 \;\Rightarrow \left(x_i - 1/\sqrt{n}\right)f_1(\mathbf{x})\exp\left\{-\sum_{i=1}^{n}\left(x_i - 1/\sqrt{n}\right)^2\right\} = -\left(x_i + 1/\sqrt{n}\right)f_2(\mathbf{x})\exp\left\{-\sum_{i=1}^{n}\left(x_i + 1/\sqrt{n}\right)^2\right\}, \quad \forall i\,. \tag{1h}$$

By inspection, one can see that the point $\mathbf{x}^* : x_1 = x_2 = \ldots = x_n = 0$ is the *only point on the Pareto frontier* that solves equation (1h), and the objective function values are equal at this point i.e., $f_1(\mathbf{x}^*) = f_2(\mathbf{x}^*) = 0.632121$. As can be seen from the results presented above, GENO easily finds this solution.[6]

Concluding Remarks

The Euclidean Compromise Solution is a member of a class of solutions first suggested by Yu (1973). Although this approach is the subject of much current research, most this effort is directed at generating a *set* of solutions (in a *search-then-decide* strategy) as opposed to producing a single point as advocated in this note. However, classical compromise programming methods do at least identify many more points on the Pareto frontier unlike the weighted-sum approaches which have been shown to be incapable of generating points from all parts of the efficiency frontier in some cases.

Evolutionary algorithms alleviate most of the limitations of classical multi-objective solution methods. GENO is an evolutionary solver that does not use 'scalarization' of the objective vector in the solution process; neither does it need to explicitly estimate the Pareto frontier. Furthermore, the model-user is not required to express any preference *prior*, *post* or *during* the solution process. Rather it is assumed that the user is rational and would therefore prefer the ideal solution if this were attainable: as a compromise, he accepts the compromise solution as being the best that can be done. The algorithm is capable of converging to any point on the Pareto frontier that is closest, by some measure, to the ideal solution—this being ascertained *logically* by the analyst.[7] The current design employs the Euclidean metric but other measures may be used. Finally, it should be noted that, provided the components of the objective vector are properly scaled in outcome space, the solution generated by GENO will always exhibit a "middling characteristic",[8] which is intuitively what is desired of a compromise solution.

As can be seen from the results presented above, the solution generated by GENO is the same as that determined analytically; the algorithm converges to the Euclidean compromise solution within ten generations.

---

[6] The graphics accompanying this example pertain to a different version of the problem in which outcomes are restricted as follows:

$$f_1 \in (-\infty, \infty); \qquad f_2 \in [0.4, 0.6].$$

They were generated by an earlier less efficient version of the program (circa, 1998): they have been included in this presentation merely to provide visual proof that GENO does indeed produce a solution that is located on the Pareto frontier.

[7] See also Example 3 below and Example 3.13 in Siwale (2006)

[8] The term 'middling' was brought to the multi-objective programming lexicon by Schaffer (1984) who noted that his VEGA technique tended to produce solutions that excelled on one criterion, but performed poorly on others. He argues that what is desirable of a compromise solution is that it should have acceptable performance simultaneously on all criteria, i.e. a "middling performance".

Graphic 1: Population Progression towards the Pareto Frontier [9]

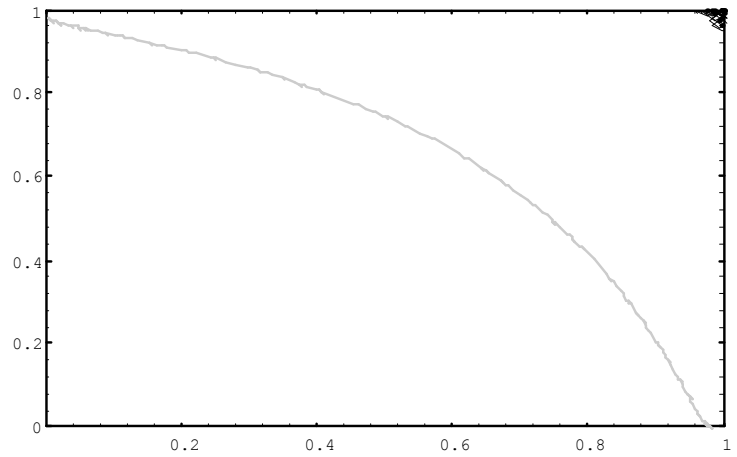

Figure 1a: Population Distribution at Generation 0

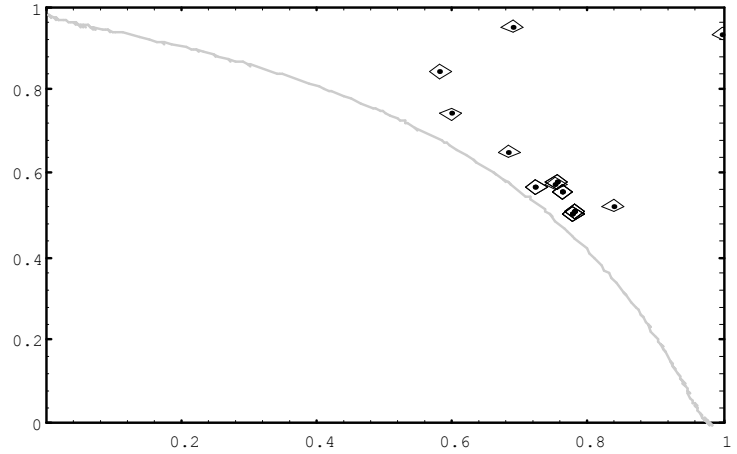

Figure 1b: Population Distribution at Generation 20

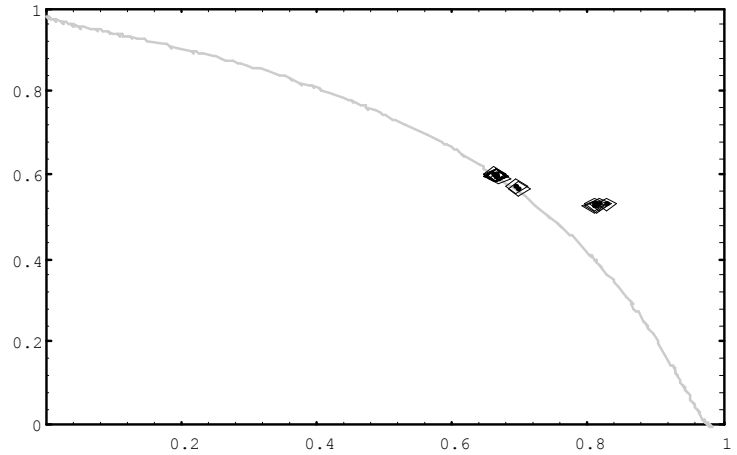

Figure 1c: Population Distribution at Generation 40

[9] The Pareto front is depicted by the grey curve.

Graphic 1: Population Progression towards the Pareto Frontier (Continued)

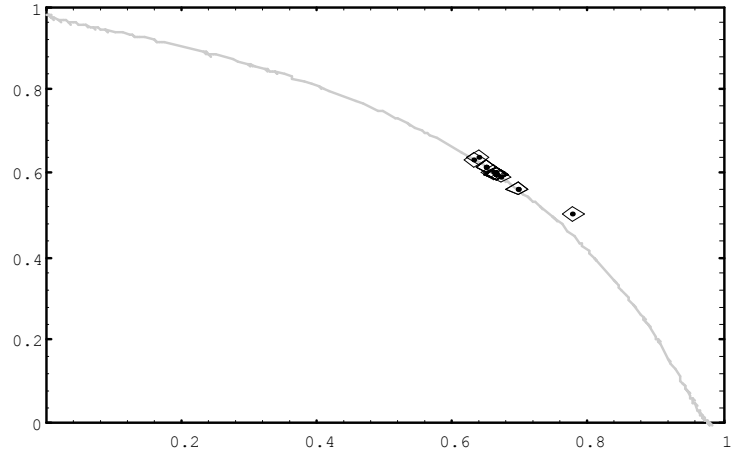

Figure 1d: Population Distribution at Generation 60

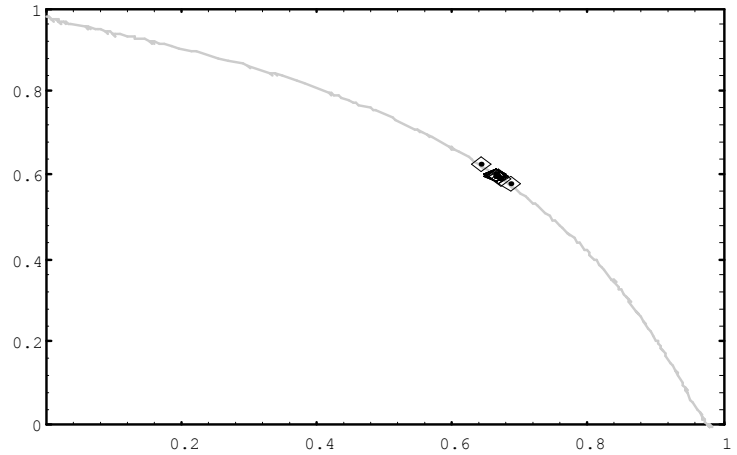

Figure 1e: Population Distribution at Generation 80

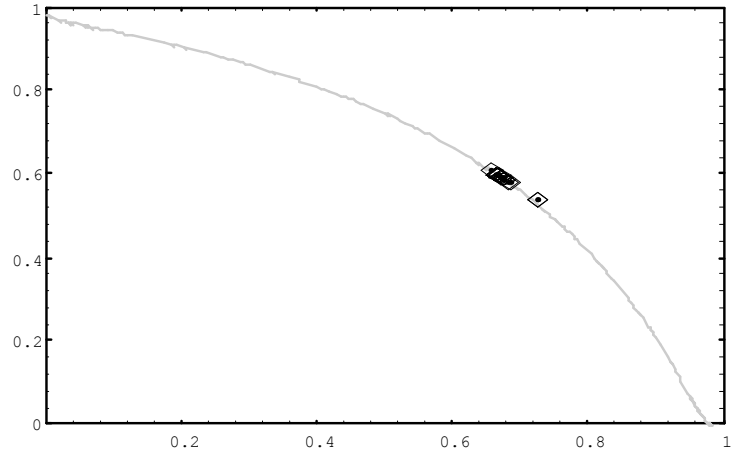

Figure 1f: Population Distribution at Generation 100

Example 2: [**Source**: Murphy, Sherali and Soyster (1982)]

$$\underset{\boldsymbol{q}}{\text{Opt}}\ \mathbf{J}(\boldsymbol{q}) = (J^1, J^2, \cdots, J^5)^{\mathrm{T}}$$

Subject to:

$$J^i(\boldsymbol{q}) = q_i\, p(\boldsymbol{q}) - f_i(q_i)\ ;\quad f_i(q_i) = c_i q_i + \frac{(q_i)^{(1+\alpha_i)}}{(1+\alpha_i)(K_i)^{\alpha_i}}\ ;\quad p(\boldsymbol{q}) = \left(\frac{5000}{\boldsymbol{Q}}\right)^{\beta}\ ;\quad \boldsymbol{Q} = \sum_{i=1}^{n} q_i\ ;\ i = \{1,2,...,5\}$$

Problem-specific Data:

Firm	c_i	K_i	α_i	β
1	10	5	1/1.2	1/1.1
2	8	5	1/1.1	1/1.1
3	6	5	1.00	1/1.1
4	4	5	1/0.9	1/1.1
5	2	5	1/0.8	1/1.1

I. **GENO Output**

Generation	q_1^*	q_2^*	q_3^*	q_4^*	q_5^*
0	36.100000	63.500000	40.500000	40.600000	21.700000
20	36.930000	41.810000	43.710000	42.660000	39.180000
60	36.932500	41.818160	43.706590	42.659240	39.178900
80	36.932511	41.818141	43.706578	42.659240	39.178952
90	36.932511	41.818141	43.706579	42.659240	39.178953
100	36.932511	41.818141	43.706579	42.659240	39.178953

Equilibrium Solution: $\boldsymbol{q}^*$ = (36.932511, 41.818141, 43.706578, 42.659240, 39.178952)$^{\mathrm{T}}$

II. **Remarks**

This example was originally formulated by Murphy, et al. (1982) and has since been numerically solved by Harker (1984), Jörnsten (1991), as well as Kolstad and Mathiesen (1991). To provide a benchmark against which GENO may be compared to these other algorithms, a solution that is accurate to 18 decimal places was found (by solving the system of non-linear equations arising from the optimality conditions) using the *FindRoot* facility of *Mathematica*.[10] The results below clearly show that GENO computes the most accurate solution.

q_i	Benchmark	Murphy, *et al*.	Harker	Jörnsten	Kolstad, *et al*.	GENO
1	36.932510815735757481	36.9120	36.93180	36.9300	36.9350	36.932511
2	41.818141660437635128	41.8200	41.81755	41.8200	41.8182	41.818141
3	43.706578522274216542	43.7050	43.7060	43.7100	43.7066	43.706578
4	42.659239743305114839	42.6650	42.6588	42.6600	42.6593	42.659240
5	39.178952516625022418	39.1820	39.1786	39.1800	39.1790	39.178952

[10] In this case, the problem is convex and the solution is easy to obtain directly from the optimality conditions derived by Rosen (1965).

Example 3 [**Source**: Marco, Désidéri and Lanteri (1999)]

$$\underset{\mathbf{z}}{\text{Opt}}\ \mathbf{f}(\mathbf{z}) = \{f_1(\mathbf{z}), f_2(\mathbf{z})\}$$

subject to: $f_1(\mathbf{z}) = (\mathrm{x}-1)^2 + (\mathrm{y}-3)^2$; $\quad f_2(\mathbf{z}) = (\mathrm{x}-4)^2 + (\mathrm{y}-2)^2$; $\quad \mathbf{z} = (\mathrm{x}, \mathrm{y})^{\mathrm{T}} \in [-5, 5] \times [-5, 5]$

I. **GENO Output**

Generation	*Objective [1]*	*Objective [2]*
0	1.730000	7.930000
10	2.500000	2.500000
20	2.500000	2.500000
40	2.500000	2.500000
80	2.500000	2.500000
100	2.500000	2.500000

Solution Vector: **z** = (2.500000, 2.500000) $^{\mathrm{T}}$

Objective Function Value: **f** (**z**) = (2.500000, 2.500000) $^{\mathrm{T}}$

II. **Remarks**

The Pareto frontier is that set of points that minimise the auxiliary objective:

$$J(\mathbf{z}) = \alpha f_1(\mathbf{z}) + (1-\alpha) f_2(\mathbf{z}), \quad \alpha \in [0, 1]. \tag{3a}$$

The first order optimality conditions are:

$$\partial J(\mathbf{z})/\partial x = 0 \quad \Leftrightarrow \quad x = 4 - 3\alpha\,; \quad \partial J(\mathbf{z})/\partial y = 0 \quad \Leftrightarrow \quad y = 2 + \alpha \tag{3b}$$

Given that $\alpha \in [0, 1]$, then $x \in [1, 4]$, $y \in [2, 3]$, and hence $f_1 \in [0, 10]$, $f_2 \in [0, 10]$. In this example, it is required to minimise both objectives, and therefore the *ideal solution* in the space of outcomes is the point (0, 0). Given a candidate solution, **z***, its Euclidean distance from the ideal point (in the space of outcomes) is:

$$R(\mathbf{z}) = \sqrt{\left(f_1^2(\mathbf{z}) + f_2^2(\mathbf{z})\right)} = \sqrt{[(x-1)^2 + (y-3)^2]^2 + [(x-4)^2 + (y-2)^2]^2}\,. \tag{3c}$$

This distance is least when the following holds:

$$(\partial R(\mathbf{z})/\partial x = 0) \wedge (\partial R(\mathbf{z})/\partial y = 0) \quad \Leftrightarrow \quad x + 3y = 10\,. \tag{3d}$$

There are many points that satisfy condition 3d. But by evaluating equation 3c at points that are on the frontier,[11] and are near the candidate solution, $\mathbf{z}^* = (2.5, 2.5)^{\mathrm{T}}$, one may verify that **z*** is indeed a minimum distance point. The Table below illustrates.

	Z^T = (2.53, 2.49)	Z^T = (2.47, 2.51)	Z^T = (2.5, 2.5)	Z^T = (2.49, 2.503)	Z^T = (2.51, 2.497)
f_1	2.601	2.401	2.5	2.467	2.533
f_2	2.401	2.601	2.5	2.533	2.467
Distance Function	3.53977	3.53977	3.53553	3.53592	3.53592

As can be seen in the Table above the point $\mathbf{z}^* = (2.5, 2.5)^{\mathrm{T}}$ is indeed the Euclidean compromise solution, at least up to the second decimal place. And GENO easily converges to this solution.

[11] Note that for the 'new point' to be on the Pareto-efficient frontier, the perturbations in the variables must satisfy $\delta x + 3\delta y = 0$.

Example 4: [**Source**: Coello Coello (2000)]

$$\min_{\mathbf{x}} J(\mathbf{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_4x_1^2 + 19.84x_3x_1^2$$

Subject to: $x_4 - 240 \le 0$ [12]

$$-x_1 + 0.0193x_3 \le 0$$

$$-x_2 + 0.00954x_3 \le 0$$

$$-\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1{,}296{,}000 \le 0$$

$$x_i \in [0.0625, 99] \cap \{x_i : x_i = 0.0625N,\ N \in \mathbf{Z}\},\ i = 1, 2;\quad x_i \in [10.0, 200],\ i = 3, 4$$

I. **GENO Output** [13]

Generation	*Objective 1*	*Objective 2*
0	1500647.737740	0.000000
100	6059.828502	0.000000
200	6059.715394	0.000000
320	6059.714347	0.000000
340	6059.714347	0.000000
360	6059.714336	0.000000
380	6059.714336	0.000000
400	6059.714336	0.000000
480	6059.714336	0.000000
500	6059.714336	0.000000

Optimal Variable Vector: $\mathbf{x}$ = (0.812500, 0.437500, 42.098446, 176.636596) ^{T}

Objective Function Value: $J(\mathbf{x})$ = 6059.714336

II. **Remarks**

This problem has previously been tackled by Deb (1997) using GeneAS (Genetic Adaptive Search); by Kannan and Kramer (1994) using an augmented Lagrangian multiplier method; and by Sandgren (1988) using a branch and bound technique; and by Coello Coello (2000) using a genetic algorithm. Coello Coello (2000, p.18) presents a comparison of these methods together with his technique: the table below is an extract from there to which has been appended the result by GENO.

	Coello Coello	Deb (1997)	Kannan, *et al*.	Sandgren	GENO
Best Function Value	6069.3267	6410.3811	7198.0428	8129.1036	6059.714336

As can be seen, the solution by GENO is by far the best amongst those considered; in fact, as of this writing, it is the best of all the solutions that I am aware of.[14] Note also that in the final solution vector, x_1 and x_2 are integer multiples of 0.0625 as required.

[12] This constraint is in fact superfluous since $x_4 \in [0, 200]$ is there. But it is retained in order to preserve the original problem specification.

[13] **Legend**: *Objective 1* is the actual function being minimised; *Objective 2* is a merit function for an auxiliary mathematical program associated with the constraints. An entry of 'zero' implies that the constraints are satisfied (for details, see Siwale: 2006, pp.12 - 14).

[14] Hedar and Fukushima (2005, p.19) claim to have found a better solution valued 5868.764836, but it should be noted that their solution ignores the discreteness restriction on x_1 and x_2, and so their algorithm cannot, strictly speaking, be compared to GENO.

4 Summary

This note has introduced GENO — a solver for multi-objective non-linear programs (amongst other types). Several numerical examples solved using GENO were presented and a new benchmark result (**Examples** 4) was identified for designers of other algorithms to aim for.

References

ANDERSEN, K. A. and M. Lind (1999). Computing the NTU-Shapley Value of NTU-games defined by Multiple Objective Linear Programs. *International Journal of Game Theory*, **28**, pp.585-597.

BAŞAR, T. and G. J. Olsder (1999). *Dynamic Non-cooperative Game Theory*. SIAM Classics in Applied Mathematics, **23**, SIAM, Philadelphia

COELLO COELLO, C. A. (1996). An Empirical Study of Evolutionary Techniques for Multi-objective Optimization in Engineering Design. Unpublished Ph.D. Dissertation, Department of Computer Science, Tulane University, U.S.A.

COELLO COELLO, C. A. (2000). Constraint-handling Using an Evolutionary Multi-objective Optimisation Technique. *Civil Engineering and Environmental Systems*, **17**, pp. 319-346.

COELLO COELLO, C. A., G. T. Pulido and E. M. Montes (2005). Current and Future Research Trends in Evolutionary Multi-objective Optimization. [Online] Available from: List of References on Evolutionary Multiobjective Optimization [Accessed January 1, 2007].

CONLEY, J. P., R. McLean and S. Wilkie (2000): "Axiomatic Foundations for Compromise Theory: The Duality of Bargaining Theory and Multi-Objective Programming", Forth coming in *Games and Economic Behaviour*.

DAS, I. and J. Dennis (1997). A Closer Look at Some Drawbacks of Minimising Weighted Sums of Objectives for Pareto Set Generation in Multi-criteria Optimisation Problems. *Structural Optimisation*, **14**, pp. 63-69.

DEB, K. (1997). GeneAS: A Robust Optimal Design Technique for Mechanical Component Design. In D. Dasgupta, and Z. Michalewicz (Eds.). *Evolutionary Algorithms in Engineering Applications*. Springer-Verlag, Berlin.

DEB, K. (1999). Multi-objective Evolutionary Algorithms: Introducing Bias Amoung Pareto-optimal Solutions. *KanGAL Report No. 99002*, Indian Institute of Technology, Kanpur, PIN 208 016, India.

DEB, K. (2006). Reference Point Based Milti-objective Optimization Using Evolutionary Algorithms. *International Journal of Computational Intelligence Research*, **2**, pp. 273-286.

FONSECA, C. M. and P. J. Flemming (1995). An Overview of Evolutionary Algorithms in Multi-objective Optimisation. *Evolutionary Computation*, **3**, pp. 1-16.

GEOFFRION, A. M. (1968). Proper Efficiency and the Theory of Vector Maximization. *Mathematical Analysis and Applications*, **22**, pp. 618-630.

HARKER, P. T. (1984). A Variational Inequality Approach for the Determination of Oligopolistic Market Equilibrium. *Mathematical Programming*, **30**, pp. 105-111.

HEDAR, A. and M. Fukushima. (2005). Derivative-Free Filter Simulated Annealing Method for Constrained Continuous Global Optimisation. In G. Di Pillo and F. Giannessi (Eds.). *Nonlinear Optimisation and Applications 2*. Kluwer, Amsterdam.

JÖRNSTEN, K. O. (1991). A Method to determine Oligopolistic Market Equilibra Based on a Convex Optimisation Formulation. *Optimisation*, **22**, pp. 439-447

KANNAN, B. K. and S. N. Kramer (1994). An Augmented Lagrangian Multiplier Based Method for Mixed Integer Discrete Continuous Optimisation and its Applications to Mechanical Design. *Journal of Mechanical Design. Transactions of the ASME*, **116**, pp. 318-320.

KASPRZAK, E. M. and K. E. Lewis (2001). Pareto Analysis in Multi-objective Optimization Using the Colinearity Theorem and Scaling Method. *Structural and Multi-disciplinary Optimization*, **22**, pp. 208-218.

KOLSTAD, C. O. and L. Mathiesen (1991). Computing Cournot-Nash Equilibra. *Operations Research*, **39**, pp. 739-748

MARCO, N., J.A., Désidéri, and S. Lanteri (1999). Multi-Objective Optimization in CFD by Genetic Algorithms. Rapport de Recherche, No. 3686, Institut National de Rechereche en Informatique et en Automatique.

MESSAC, F. H., J. D. Sundaraj, R. V. Tappeta and J. E. Renaud (2000). The ability of Objective Functions to Generate Non-convex Pareto Frontiers. *AIAA Journal*, **38**, pp. 1084 - 1091.

MURPHY, A., H. D. Sherali and A. L. Soyster (1982). A Mathematical Programming Approach for Determining Oligopolistic Market Equilibrium. *Mathematical Programming*, **26**, pp. 40-47.

NASH, J. (1951). Non co-operative Games. *Annals of Mathematics*, **54**, pp. 287-295.

ROSEN, J. B. (1965). Existence and Uniqueness of Equilibrium Points for Concave *n*-person Games, *Econometrica*, **33**, pp.520-534.

SANDGREN, E. (1988). Nonlinear Integer and Discrete Programming in Mechanical Design. *Proceedings of ASME Design Technology Conference*, Kissimine, Florida, pp. 95-105.

SCHAFFER, J. D. (1984). *Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms*. Unpublished PhD Dissertation, Vanderbilt University, Nashville, U.S.A.

SEFRIOUI, M. and J. Periaux (2000): "Nash Genetic Algorithms: Examples and Applications", *2000 Congress on Evolutionary Computation*, July, IEEE Service Centre, San Diego, California, **1**, pp.509-516.

SIWALE, I. (2006). GENO^{TM} 1.0: The GAUSS User Manual, 4th Edition. *Technical Report No. RD-3-2005*, Ike's Research Ltd, London

YU, P. L. (1973). A Class of Solutions for Group Decision Problems. *Management Science*. **19**. pp. 936-946