Evaluating the Performance of an Evolutionary Tool for Exploring Solution Fronts

Neil Urquhart

Edinburgh Napier University
10 Colinton Road
Edinburgh, UK
EH10 5DT

Abstract. EvoFilter is an evolutionary algorithm based tool for searching through large non-dominated fronts in order to find a subset of solutions that are of interest to the user. EvoFilter is designed to take the output of existing Multi Objective Evolutionary Algorithms and act as a decision support tool for users. Currently EvoFilter is available for all to use on-line [1]. This paper evaluates the performance of EvoFilter by creating a large number of randomised filter specifications which are then applied using EvoFilter and a simple filter to a range of non-dominated fronts created by a portfolio of Multi Objective Genetic Algorithms (MOGAs). The results show that EvoFilter is capable of finding sets of solutions that meet the users' requirements more closely than those found using the simple filter. EvoFilter increases performance on some objectives by including relevant solutions event if these solutions slightly lessen performance on other objectives. The filter discussed in this paper may be accessed at [1].

1 Introduction and Motivation

Real-world optimisation problems are often multi-objective, with objectives that may conflict and may have differing priorities in the view of the end user. Such objectives could include minimising financial costs, CO_2 produced, distance travelled or resources (e.g. staff or vehicles) required. In such cases a common approach is to utilise a multi-objective genetic algorithm (MOGA) which produces a non-dominated front of solutions. Such a front of solutions can have many advantages from a user perspective, the front might allow the user to examine a range of solutions, exploring what is possible both in terms of extreme solutions, which optimise one objective to the maximum or compromise solutions which optimise a number of constraints as best possible. There exists a final decision to be carried out by the user when selecting the solution from within the front, there may be no single solution which is ideal in all aspects. This level of political decision making is difficult to incorporate into the problem formulation and so must be undertaken by planners, with domain expertise, based on factors such as political and legislative pressure, public opinion, financial constraints or corporate policies. For example, if environmental impact conflicts with staff cost, then there exists a level of political decision making required in order to determine which objectives should be prioritised.

The non-dominated front may be large, sizes of over 200 items are not uncommon, in such cases the decision for the user becomes more complex. If the problem has only two dimensions then the front may be plotted as a curve against two axes which gives the user a visualisation which can help them choose. But, with greater than two dimensions, visualisation becomes more complex, one option is to use Parallel Coordinates [2]. which allow visualisation of many dimensions. When using Parallel Coordinates the plot may become crowded when too many solutions are plotted, thus for larger fronts it becomes difficult for the user to make a choice.

In this paper we seek to evaluate the EA based filter, known as *EvoFilter* in order to assess its usefulness alongside a more traditional filtering technique. This evaluation is necessary, should we wish to advocate the use of EvoFilter which is available as an on-line tool [1] for anyone to make use of. As in [3] we use a multi-objective Workforce Scheduling and Routing Problem (WSRP) to test the filter. The WSRP is a useful problem to test the filter on as it has multiple objectives, some of which are conflicting, and when solved using a MOGA produces non-dominated fronts in a range of sizes.

2 Previous Work

A survey of a priori and post priori methods of supporting decision makers faced with a Pareto front of solutions to choose from is presented in [4], the authors use a multi-objective evolutionary algorithm to solve a multi-objective control problem, the output of which is a non-dominated front of potential solutions. Methods of decision support include self-organising maps and subtractive clustering, fuzzy scoring and data envelopment analysis (all post priori approaches) and a guided multi-objective genetic algorithm (a priori). The authors suggest that the post priori approach of subtractive clustering, fuzzy scoring and data envelopment analysis is the most promising approach of those examined.

The EvoFilter algorithm was previously described in [1], it attempts to filter a non-dominated front to just those solutions that are of interest to the user. The user can specify this areas of interest in terms of the *range* of solutions of interest on a particular axis and by specifying the minimum difference between solutions, of interest to the user. The input to the algorithm is an existing non-dominated front, and the output, a smaller front that meets the users' specifications, selected by an Evolutionary Algorithm. A similar approach of reducing the size of the non-dominated front after creation is described in [5], but the approach taken is to partition the front into clusters, using techniques such as k-means, and then select one solution to represent each cluster in a reduced size of front. Cluster analysis is also undertaken by the authors of [6].

The authors of [7] take the approach of generating a smaller front, rather than searching through an existing front, using a method known as the Smart Normal Constraint (SNC) method which generates a reduced size Pareto set, by directing the search to areas of interest.

The concept of using filters to filter out redundant solutions from a Pareto front is introduced in [8]where a two stage process is used to filter solutions from an existing Pareto front. The first stage known as the global filter, removes any non-Pareto solutions, the second stage reduces the size of the front by removing any solutions that do

not meet a pre-defined trade-off requirement. The combination of the two filters results in a small front whose members exhibit the maximum amount of trade-off between objectives. Earlier work with filters includes [9] where the filter is incorporated within an evolutionary algorithm.

The Workforce Scheduling and Routing Problem (WSRP) has been extensively investigated, although similar to vehicle routing problems, the focus of the WSRP is on individuals rather than vehicles. For a comprehensive introduction to the WSRP and an overview of the latest developments, the reader is directed towards [10], [11] and [12]. A number of previous researchers have examined the scheduling and routing of workforces, including home care scheduling[13], security personnel scheduling [14] and technician scheduling [15]. A attempt to model the WSRP as a bi/multi-objective problem can be found in [11], the authors use cost and patient convenience as the twin objectives. The solution cost is the travel cost and staff overtime costs, patient convenience is defined as to whether the member of staff allocated is preferred, moderately preferred or not preferred, with penalties allocated as appropriate. The results presented show a strong relationship between convenience and cost; the more convenient a solution the higher the cost is likely to be.

3 Methodology

3.1 The Workforce Scheduling and Routing Problem

In order to evaluate EvoFilter we apply it to a multi-objective Workforce Scheduling and Routing Problem (WSRP) as previously described in [3]. A set of clients each require a visit, each visit has a specific location (within greater London), a duration and a time window, within which the visit must commence. Each visit must be allocated to an employee, who will undertake the visit, each employee is allocated a set of visits which comprise a days work. An employee may be set to travel by public transport or by motor car, which will determine the times taken to travel between visits and the CO_2 produced.

The output from a portfolio of Multi-Objective Evolutionary Algorithms (MOEAs), are merged to produce a set of non-dominated solutions. Each of the MOEAs within the portfolio uses a problem representation based on a grand tour permutation of visits and travel modes between visits. When constructing a solution each visit is added to the solution in the permutation order, initially, there will be no employees within the solution so the first visit is added as the initial visit of a new employee, the travel mode for that employee being determined by this visit. Each subsequent visit i is then considered, being added to the first employee that can feasibly undertake the visit using the travel mode from their previous visit. If the visit cannot be added to any of the existing employees, a new employee is created and the visit is added to that employee. In this manner a feasible solution is constructed from the problem representation. When a solution has been built, its internal costs, CO_2 emission and impact on traffic may be calculated.

Within our problem we have the following dimensions (each of which which may or may not be used as objectives):

1. Total Cost (£)
2. Staff Cost (£)
3. Travel Cost (£)
4. Staff required
5. CO_2
6. Car use

Note that items 1-4 are internal costs to the organisation providing the service, items 5 and 6 are external costs.

We utilise a portfolio of algorithms to produce non-dominated fronts, the algorithms in the portfolio are based upon NSGA-II [16, 17], SEAMO [18] and SPEA2 [19], see [3] for a full description. It is useful to be able to visualise a Pareto front in order to determine the range of the objectives covered and the trade-offs available. For a 2-dimensional problem the front can be plotted by allocating each dimension to the X and Y coordinates, but for dimensions greater than two, plotting becomes problematic. When considering problems with a more than 2 dimensions than 2 a parallel coordinate plot [2] allows fronts to be visualised and the relationships between the solutions objectives to be explored. Within a parallel coordinates plot a vertical axis is created for each dimension, each solution being represented by a poly-line intersecting each axis at the point appropriate for that solution.

3.2 The EvoFilter Algorithm

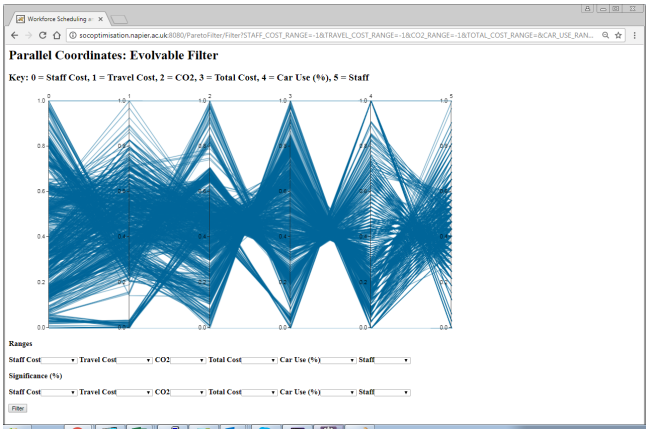
The EvoFiler algorithm is used to select a subset of solutions from a non-dominated front based on a specification set by the user. EvoFilter has been incorporated with a web based tool [1] (see figure 2) which allows the user to specify the ranges within each axis that interest them and to specify a minimum distance on each axis which should separate solutions. The source code for the algorithm is available on line [20]. Selecting a range on an axis is analogous to the practice of "brushing" [2] which allows the user to specify sections of one or more axes in order to select only those poly-lines that pass through the specified sections.

EvoFilter is based upon an evolutionary algorithm , the representation being a binary string, one bit for each member of the Pareto front, a 1 signifies that the solution is filtered a 0 signifies that the solution is to be left in. We use a population size of 20 , creating one child using uniform crossover and a random bit flip mutation in each generation , see algorithm 1 and table 1.

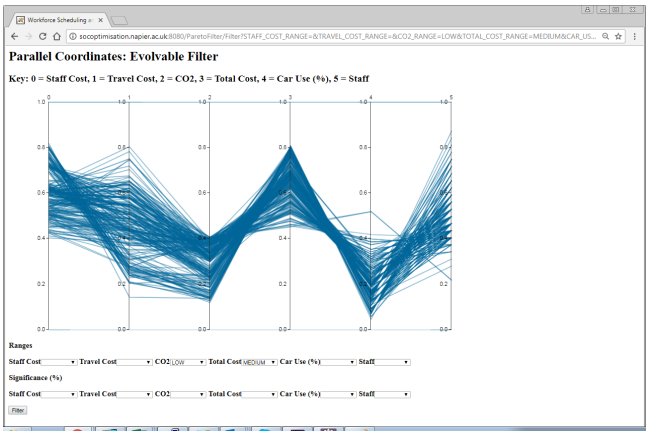
The fitness function examines each item within the front and adds penalties as follows:

1. Any solution not filtered on an axis that is out of range add a penalty of the difference between the item value and the closest range limit
2. Any solution that is filtered, but lies within a range then add a fixed penalty of 0.01
3. Any pair of solutions on an axis that are closer together than the specified criteria add a penalty based on the actual difference, less the minimum specified

The first two items penalise solutions which are unfiltered outside of the specified range on an axis and solutions that are filtered which lie within the specified ranges.

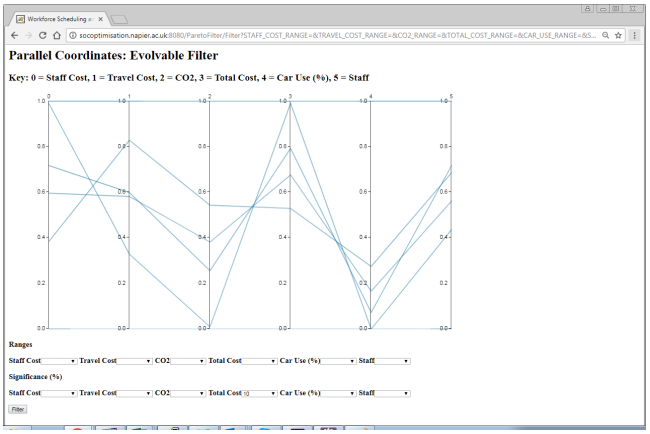


(a)

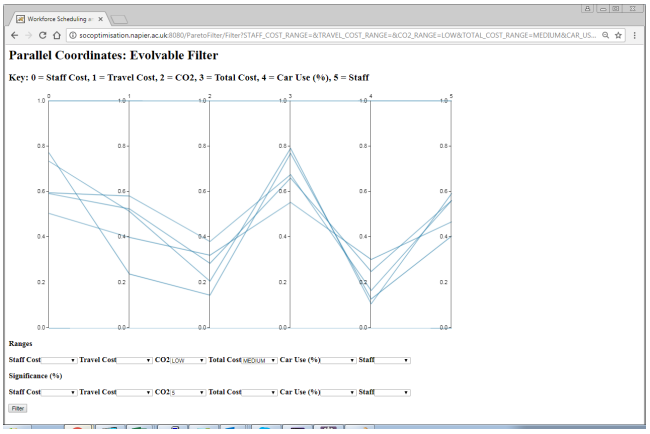


(b)

Fig. 1. EvoFilter is incorporated within a web tool, which may be used to open a front of solutions and find sets of solutions of interest to the user. Figure 1(a) shows a large 6 dimensional front loaded from a .CSV file. We apply a simple range filter to the CO_2 and Total cost dimensions in figure 1(b).



(a)



(b)

Fig. 2. EvoFilter can filter by difference, figure 2(a) shows the effect of placing a requirement of a minimum of 10% difference on the Total Cost axis. A combination of minimum difference and range filtering may be seen in figure 2(b).

Algorithm 1: EvoFilter

```
1  randomise_population();
2  while evals<MAX_EVALS do
3      parent1 = tournament();
4      parent2 = tournament();
5      if rnd() <RECOMBINATION_PRESSURE then
6          child = new Individual(parent1,parent2);
7      else
8          child = parent1.clone();
9      if rnd() <MUTATION_PRESSURE then
10         mutate(child);
11     evaluate(child);
12     rip = replacementTournament();
13     if rip.finCost() >child.finCost() then
14         population.remove(rip);
15         population.add(child);
16 return findBest();
```

Items 1 and 3 allocate penalties which increase depending on the severity of the issue. The algorithm has been implemented in Java (and is contained within a a Java Servlet) can undertake 400,000 evaluations in approx. 10s, allowing a fast response to the user, who may set and adjust the criteria interactively.

Table 1. Parameters used within the Evolutionary Algorithm

Parameter	**Value**
Population size	20
Recombination Pressure	0.5
Mutation Pressure	0.5
Selection tournament size	2
Replacement tournament size	2

3.3 Experimental Methodology

In order to assess the performance of the evolutionary filter we generate random filter specifications and then compare the front produced by the evolutionary filter, with the front produced using a simple filter and the original unfiltered front. In total 75,000 random filter specifications were generated, random specifications were used to simulate the web based environment where a user may pick any combination of range filter and minimum difference filter across the problem dimensions. When generating a random filter specification we first randomly select a non-dominated front to which the filter will be applied, 10 fronts generated by a MOGA (described in section 3.1) were used,

the specifications are given in table 2. Three types of random filter specifications were generated; those with only range filters, those with only minimum difference filters and those with a mixture of both types.

The algorithm has been implemented in Java (and is contained within a a Java Servlet) can undertake 400,000 evaluations in approx. 10s, allowing a fast response to the user, who may set an adjust the criteria interactively.

The range filters were applied randomly to dimensions with a probability of 0.3, if a dimension is selected then the upper and lower bounds of the range are selected randomly in the range 0-1 (in 0.1 increments). A minimum difference filter was applied to a dimension with a probability of 0.2, the minimum difference is selected randomly from the values 0.01, 0.05, 0.1, 0.15 and 0.2.

The simple filter operates as follows:

- For each axis with a range filter specified, any individual solution not within the range is filtered out.

- For each axis with a minimum distance filter, each solution X_i is considered in turn, and compared against every other remaining solution X_j , if the distance between X_i and X_j is less than the minimum specified then X_j is filtered out.

Problem Instance	**Optimisation Criterion**	**Front Size**
offset-r00	CO2 TotalCost CarUse	867
offset-8	CO2 TotalCost CarUse	495
lon-1	CO2 TotalCost CarUse	338
blon-1	CO2 TotalCost CarUse	225
cluster-4	CO2 TotalCost	132
blon-2	CO2 TotalCost	91
offset-1	CO2 TotalCost	53
blon-1	CO2 TravelCost	21
offset-8	CO2 TravelCost	7

Table 2. The non-dominated fronts used within the experiment.

4 Results

We simulate the actions of a user, by creating 75,000 filter specifications, each of which represents a possible interaction with EvoFilter via the web interface. Our results are based on comparing the results produced for each specification

To simulate the range of possible user interactions with the filter 75,000 filter specifications were generated (see section 3.3). Within these filter specifications there will

be a number which specify a filter which eliminates all of the solutions in a front. Table 3 shows the number of instances in which a front was produced after filtering, the first thing that we note is that fronts are only produced in less than 50% of cases. We note that from a users' perspective there is a significant chance that their filter will not produce any solutions, but fast run time of EvoFilter allows them to adjust and re-filter rapidly. More significant are the 5% of cases where EvoFilter can find some results, but simple filter cannot. Such cases suggest that EvoFilter can find some members of the front which closely match the filter, but fall out width the filter in a minor way, but which may still be of interest to the user. Ultimately, it is the domain expertise of the user that will evaluate the usefulness of a particular solution in context.

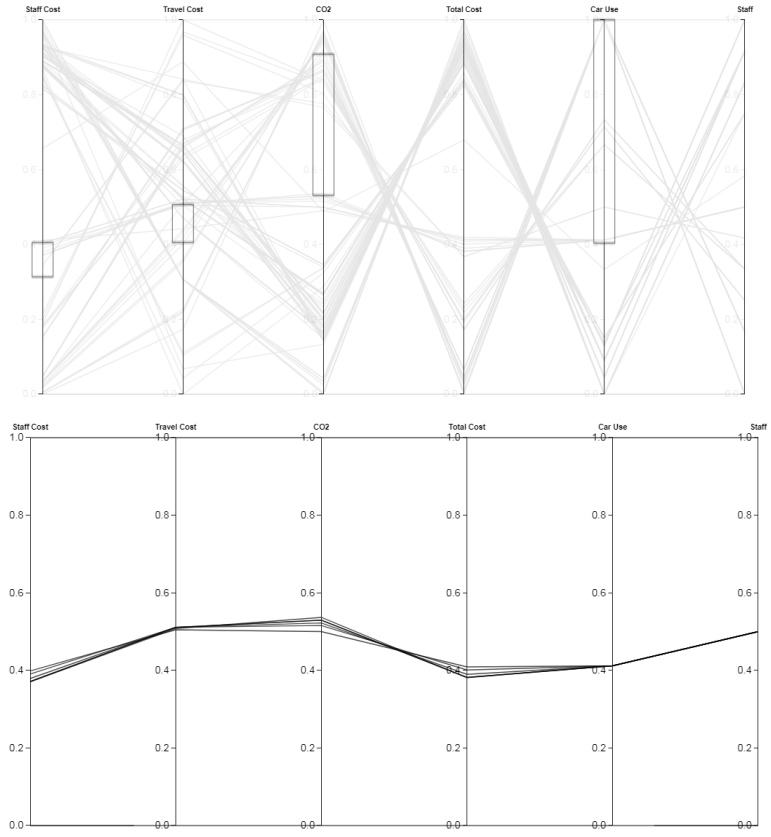


Fig. 3. The upper plot shows the parallel coordinates plot for a non-dominated front (comprising 91 examples). The boxes on the axis denote a set of filter criterion. In this case we note that that a simple filter does not return any solutions, as there are no solutions that intersect all of the areas of interest. The lower plot shows the output from EvoFilter, which has found a small set of solutions which although not an exact match are close enough to be of interest to the user. Plots produced by *http://www.parallelcoordinates.de/*

Filter specifications	75000	
simple filter	31558	42.08%
EvoFlter	35443	47.26%

Table 3. The number of filter specifications which resulted in a front with more than 0 members.

It is useful to get some idea of the reduction in front size that occurs during the filtering process, table 4 gives the average sizes of fronts before and after the application of a filter. On average EvoFilter results in a front that is only 14% of the size of the original, which is smaller than that produced by Simple-filter. As might be expected filters than include elements of both range and difference filtering are the most strict and result in the largest % reduction in front size, least effective are those filters that only reduce the minimum difference between members of the front. We can investigate the front sizes further by examining how, for each filter specification type, the size of the fronts resulting from each of the filter compares, see table 5. Overall, when all filter specifications are considered, we note that approximately equal numbers (37 %) are have a larger front produced by the EvoFilter or the Simple Filter, with 25 % having an equal size for both filters. If we only consider those filter specifications which have a mixture of range and minima difference filters we note that in the majority of cases (70 %) the EvoFilter finds a smaller front that the front found by the Simple-filter.

	All		**Both Filters**		**Range Only**		**Diff only**	
	Absolute	**%**	**Absolute**	**%**	**Absolute**	**%**	**Absolute**	**%**
Original front	237.99		297.67		328.84		216.76	
Simple filter	54.71	22.99%	19.06	6.40%	39.78	12.10%	60.54	27.93%
Evofilter	34.75	14.60%	30.46	10.23%	49.90	15.18%	32.51	15.00%

Table 4. The size of front produced (for instances where a front is produced by the filter). The column marked all shows for all instances where a front was found. The results are further broken down by filter type.

	All		**Both**		**Range**		**Diff**	
Total >0	31282		2246		4304		24732	
evoFilter >simple filter	11851	37.88%	27	1.20%	70	1.63%	11754	47.53%
evoFilter== simple filter	7670	24.52%	661	29.43%	1227	28.51%	5782	23.38%
evoFilter <simple filter	11761	37.60%	1558	69.37%	3007	69.87%	7196	29.10%

Table 5. A comparison of front sizes produced by each type of filter.

The size of front produced by the filters may vary, but it is the content of the front that determines its usefulness to the end user. Ideally a filter is required that allows the smallest number of solutions required to give the user reasonable choice within the

bounds of the filter specification. Traditionally, non-dominated fronts have been compared using the Hyper Volume metric, which measures the area contained behind a non-dominated front. Table 6 shows the average Hypervolumes produced in the experiment. It is worth noting that both filters (EvoFilter and Simple Filter) reduce the Hypervolume by 50%, whilst the filters themselves have broadly similar averages. This suggests that although the fronts are considerably reduced in scope from the original, by filtering the fronts output by the filter are largely the same in scope, regardless of the filter used.

	Orioginal Front	**Simple filter**	**Evol Filter**
All	6.948	3.536	3.567
Both	8.412	2.639	3.182
RangeOnly	8.669	3.036	3.361
DiffOnly	6.516	3.704	3.637

Table 6. Average Hypervolume produced within the experiment.

Although there is difference in size between the fronts produced by the two filtering techniques, what is more important from a users' perspective, examining the difference in the solution space covered by the filtered front. A metric known as range error may be calculated for each dimension of the front as shown in algorithm 2.

Algorithm 2: CalcRangeError

```
foreach dimension d do
    if dMax_d > fMax_d then
        rangeErr_d = (dimMax_d - fMax_d)
    if dMin_d < fMin_d then
        rangeErr_d += (fMin_d - dimMin_d)
```

The range error records any dimensions upon which there are solutions that are outside of the filter range, the further outside the filter the larger the error. Table 7 records the average range errors found in the experiment. We note that the Simple Filter always produced a range error of 0 as it filters out any individual which is outside of the range in any dimension. We note large error values with no filter in place, and very small values when using the EvoFilter. The range error tells us that EvoFilter sometimes allows an individual that has some dimensions that are out of range, but this does not show any advantage in using the EvoFilter. EvoFilter is only of any benefit if the errors recoded in table 7 are counterbalanced by benefits. One major benefit is a greater range of solutions within a filtered dimension, allowing greater choice to the user. Table 8 examines the range Δ of solutions on dimensions that have a range filter. For each filter specification, we compare the fronts produced on thee basis:

– **Δ EvoFilter $>\Delta$ Simple-filter** True if the difference between the maximum and minimum values plotted on the axis for EvoFilter is greater than that for Simple Filter.
– **Δ EvoFilter within the Filter Specification** True if the maximum and minimum values plotted on the axis after using EvoFilter are within the range specified by the filter specification.
– **Δ EvoFilter $>\Delta$ Simple-filter AND Δ EvoFilter is within the Filter Specification** True, if both of the above two conditions are true.

It is the last condition that is of interest to us, that suggests a situation where, on a given axis the EvoFilter has found a set of solutions that offer a wider choice than the Simple-filter, but without violating the range set out in the filter specification. Table 8 shows the number of instances (fronts) where the above three conditions hold true and the number of individual axis as well. We note that when filtering on both minimum difference and range over 50% of instances have at least one axis where EvoFilter outperforms the Simple-filter. We note that the advantages shown in Table 8 should be used to offset the errors shown in table 7.

	Original Front	**Simple filter**	**Evol Filter**
All	23.38	0.00	0.03
Both	97.48	0.00	0.08
RangeOnly	119.08	0.00	0.15

Table 7. Average Range Error

		Criterion A			Criterion B			Criterion C		
	Examples	**Axis**	**Fronts**		**Axis**	**Fronts**		**Axis**	**Fronts**	
All	31282	60146	16545	52.89%	13073	5913	18.90%	5967	3392	10.84%
Both Filters	2246	7869	1580	70.35%	4847	2080	92.61%	2513	1304	58.06%
Range Only	4304	13713	1580	36.71%	8221	3834	89.08%	3454	2088	48.51%
Diff Only	24732	38564	11959	48.35%	0	0	0.00%	0	0	0.00%

Table 8. A comparison as to how the range of data points present after filtering compares with the spread requested by the filter. Criterion A = **EvoFilter Δ >simple Δ**, Criterion B = **EvoFilter Δ within range**. Criterion C **=EvoFilter Δ within range and >simple**

As well as comparing performance on the basis of range filtering, it is also useful to examine performance on the basis of minimum difference filtering. An error value may be calculated for each axis to be subject to minimum difference filtration. To calculate the error the distance between each item on the axis is calculated, if the distance is <than the minimum distance specified the error is deemed to be the $min_dist - actual_dist$ such errors are summed across the entire front. The results of calculating these errors can be seen in table 9. We note that when only filtering on

minimum difference 68% of instances show a lesser error from EvoFilter than from Simple-filter.

	Original	**SimpleFilter**	**EvoFilter**	**Evo diff err. <Simple diff err.**	**%**
All	26.59	3.88	3.64	17054	54.52%
Both Filters	21.52	0.73	1.19	74	3.29%
Range Only	6.00	1.07	1.28	53	1.23%
Diff Only	30.63	4.65	4.27	16927	68.44%

Table 9. Average min difference errors.

5 Conclusions and Future Work

The experiment conducted uses randomly generated filter specifications to simulate users undertaking filtering using the on-line tool [1], we note that only just under half of these situations does EvoFilter find a result (table 3). When dealing with real-world problems there will be a number of cases where solutions simply do not exist - for instance as a general rule within the WSRP problem examined solutions with low CO_2 and low financial cost are rare. In such cases it is usually better to let the user know that no solutions that meet their requirements exist, rather than attempting to find the nearest matches which are too far removed from the users' request to be of use. We do note that EvoFilter can find solutions in 5% cases where the simple filter cannot (see table 3). The knowledge that there is a 50% chance that a users' first set of requirements will not return any solutions, makes the fast runtime of EvoFilter all the more necessary, so that the user may refine their requirements over a number of iterations.

We note that EvoFilter is at its most useful when filtering by range (also known as *brushing* in Parallel Coordinates literature), and that table 8 shows that EvoFilter can find sets of solutions which, although braking user constraints on range on some axis can result in other axis having a greater spread of solutions within range. The primary advantage of EvoFilter is the ability to include solutions which cause a slight breach of the users' requirements on one axis, but compensate for that by increasing performance on another axis.

It is hoped to encourage use of the EvoFilter tool, in conjunction with a number of real-world problems. The file format used to upload fronts could be modified to include more information, including phenotype information, this would allow the user to also specify features within the phenotype that are desirable. and could be included in the filtering criterion. Currently we only search through non-dominated fronts, but these do not represent the entire final population of most MOGAs, we could increase the search space to include other, dominated, solutions, on the basis that they might be a good match for user criterion.

References

1. Urquhart, N.: Evolutionary decision support for multi-objective problems. http://socoptimisation.napier.ac.uk:8080/ParetoFilter/main.jsp (2017)
2. Inselberg, A.: Parallel Coordinates. Visual MultidimensionalGeometry and its applications. Springer (2009)
3. Urquhart, N., Hart, E.: Creating optimised employee travel plans. In: Proceedings of the The 11th edition of the International Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems. (2015)
4. Zio, E., Bazzo, R.: A Comparison of Methods For SelectingPreferred Solutions in Multi-objective DecisionMaking. Number 6 in Atlantis Computational Intelligence Systems. In: Computational Intelligence Systems in Industrial Engineering. Atlantis Press (2012)
5. Cheikh, M., Jarboui, B., Loukil, T., Siarry, P.: A method for selecting pareto optimal solutions inmultiobjective optimization. Journal of Informatics and Mathematical Sciences **2**(1) (2010) 51–62
6. Chaudhari, P.M., Dharaskar, R., Thakare, V.M.: Computing the most significant solution from pareto front obtained in multi-objective evolutionary. (IJACSA) International Journal of Advanced Computer Science and Applications, **1**(4) (October 2010) 63–68
7. Hancock, B.J., Mattson, C.A.: The smart normal constraint method for directly generating a smart pareto set. Structural and Multidisciplinary Optimization **48**(4) (Oct 2013) 763–775
8. Mattson, C.A., Mullur, A.A., Messac, A.: Smart pareto filter: obtaining a minimal representation of multiobjective design space. Engineering Optimization **36**(6) (2004) 721–740
9. Cheng, F.Y., Li, D.: Genetic algorithm development for multiobjective optimization of structures. AIAA Journal **36**(6) (2017/10/31 1998) 1105–1112
10. Castillo-Salazar, J.A., Landa-Silva, D., Qu, R.: Workforce scheduling and routing problems: literature survey and computational study. Annals of Operations Research **239**(1) (2016) 39–67
11. Braekers, K., Hartl, R.F., Parragh, S.N., Tricoire, F.: A bi-objective home care scheduling problem: Analyzing the trade-off between costs and client inconvenience. European Journal of Operational Research **248**(2) (2016) 428 – 443
12. Hiermann, G., Prandtstetter, M., Rendl, A., Puchinger, J., Raidl, G.R.: Metaheuristics for solving a multimodal home-healthcare scheduling problem. Central European Journal of Operations Research **23**(1) (2015) 89–113
13. Rasmussen, M., Justesen, T., Dohn, A., Larsen, J.: The Home Care Crew Scheduling Problem:: Preference-Based Visit Clustering and Temporal Dependencies. DTU Management (2010)
14. Misir, M., S.P.V.K., Berghe, G.V.: Security personnel routing and rostering: a hyper- heuristic approach. Proceedings of the 3rd International Conference on Applied Operational Research, ICAOR11 (2011)
15. Gunther M, N.V.: Application of particle swarm optimization to the british telecom workforce scheduling problem. Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012), Son, Norway (2012)
16. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast elitist multi-objective genetic algorithm: Nsga-ii. IEEE Transactions on Evolutionary Computation **6** (2000) 182–197
17. Seshadri, A.: Nsga-ii: A multi-objective optimization algorithm. Matlab Central File Exchange (2006)
18. Valenzuela, C.: A simple evolutionary algorithm for multi-objective optimization (seamo). In: Proceedings of the Conference on Evolutionary Computation, 2002. (CEC '02). 2002 Congress on. Volume 1. (May 2002) 717–722

19. E. Zitzler, M.L., Thiele, L.: Spea2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In: Evolutionary Methods for Design, Optimisation, and Control, CIMNE (2002) 95–100
20. Urquhart, N.: Evofilter source code. `https://github.com/NeilUrquhart/EvoFilter` (January 2018)