Sādhanā (2020)45:70
https://doi.org/10.1007/s12046-020-1309-4

Improved performance in multi-objective optimization using external archive

MAHESH B PATIL

Department of Electrical Engineering, Indian Institute of Technology Bombay, Mumbai, India
e-mail: mbpatil@ee.iitb.ac.in

MS received 12 February 2019; revised 8 November 2019; accepted 15 January 2020

Abstract. We show that the use of an external archive, purely for storage purposes, can bring substantial benefits in multi-objective optimization. We first present a new scheme for archive management. We then combine it with the NSGA-II algorithm for solving multi-objective optimization problems and demonstrate significant improvement in performance. Furthermore, we show that the additional computational effort in handling the external archive is insignificant in problems for which objective functions are expensive to evaluate.

Keywords. External archive; NSGA-II; hyperspace; hypercubes.

1. Introduction

An external archive for storing non-dominated solutions plays an important role in a variety of multi-objective evolutionary algorithms (MOEAs). In algorithms based on the genetic algorithm (GA), an external archive is generally used to store the non-dominated solutions and propagate them to the next generation through a suitable selection mechanism [1–16]. In algorithms based on particle swarm optimization (PSO), an external archive is used in the selection of the globally best particle [17–22]. In some of the evolutionary algorithms (e.g., [4, 11, 18]), the external archive also serves as the final output, i.e., the approximate Pareto-optimal (APO) solution set.

The purpose of this paper is to show that an external archive, even when used purely for storage, can lead to a significant performance improvement. In particular, we show that it gives a substantially larger number of APO solutions than the original MOEA. We combine an external archive scheme with NSGA-II [23], one of the industry-standard MOEAs, and present results for a few multi-objective optimization problems. With these results, we demonstrate the advantage of using an external archive.

The paper is organized as follows. In section 2, a brief review of the existing schemes for external archive management is presented. In section 3, a new scheme for archive management that is efficient in terms of memory requirement is described. In section 4, a new algorithm that is a simple combination of the NSGA-II algorithm and the archive management scheme of section 3 is presented. The results obtained with the new algorithm are compared to those of NSGA-II in section 5. Finally, concluding remarks are presented in section 6.

Published online: 16 March 2020

2. Archive management: review

The various archiving strategies reported in the literature for multi-objective optimization may be broadly categorized as follows.

(a) Unconstrained archive: In this case, the archive is not limited, i.e., it can hold any number of solutions (see [1, 8, 19], for example). This scheme has the obvious advantage that all desirable (non-dominated) solutions are preserved, and the efficacy of the search process is not compromised on account of limited archive size. However, the memory requirement for this approach is clearly larger than that for a limited archive. Equally important, the archive operations are more time-consuming due to a larger number of archive members. Special data structures can be employed [8] to speed up computations related to the archive.
In a practical situation, an unlimited archive is generally not required from the utility perspective as long as a sufficiently large number of well-spread APO solutions are produced by the concerned MOEA. It is therefore more common to employ a limited archive.

(b) Limited archive: In this case, an upper limit ($N_A^{\max}$) is placed on the total number of solutions to be accommodated in the external archive. To decide whether to admit a new candidate (C) into the archive, it is compared to the existing solutions (A_i) in the archive. If C is dominated by each A_i, the archive remains

unchanged. If C dominates any of the archive solutions, they are removed from the archive and C is admitted. In the situation where the archive is full (i.e., it has $N_A^{\max}$ solutions already) and C is non-dominated with respect to each A_i, one solution needs to be removed before admitting C. In other words, the archive needs to be "pruned" or "truncated."

Depending on how the non-dominated solutions are organized in the archive, we can have the following sub-divisions of limited archive schemes.

(i) Archive with hypergrid: In this scheme, the archive is divided into an M-dimensional [3, 5, 18, 22] or L-dimensional [4] "hypergrid" where M is the number of objective functions and L is the number of decision variables. The smallest unit or cell of the hypergrid is called "hypercube" or "hyperbox." For pruning of the archive, some measure of the density of solutions in the hypercubes is used, and a solution from a hypercube with a higher density is preferred for removal. In Sect. 3, we will look at the hypergrid scheme of [18] in more detail.

(ii) Archive without hypergrid: In this case, the non-dominated solutions are stored as N_A individual solutions and are not organized into hypercubes. Various approaches have been used for pruning, such as clustering [2, 17], distance to the closest neighbour [6] and crowding distance [9, 11, 13, 16, 21], the intention always being to remove a solution from a dense region of the archive.

Hypergrid-based archive management schemes are attractive from the computational perspective since they involve only *local* calculations for pruning, and in the simplest case, plain *counting* of solutions in a given hypercube [18].

3. Archive management: new scheme

In this section, we present a new scheme for archive management. Before describing the new scheme, however, it is instructive to look at an existing archive management scheme and understand the implementation issues.

3.1 *Issues with archive management*

The archive management scheme used in the multi-objective particle swarm optimization (MOPSO) algorithm [18] is illustrated in figure 1 (a). A hypergrid for minimization of two objective functions f_1 and f_2 is shown in the figure. The number of hypergrid divisions is $N_{f1} = 8$ for the first objective function and $N_{f2} = 10$ for the second. The crosses in the figure represent the non-dominated solutions in the archive.

If a new candidate being considered for entry into the archive falls inside the current hypergrid – the outer rectangle in figure 1 (a) – no changes are required in the hypergrid boundaries. If it falls outside (see the solution marked as "new solution" in the figure), the hypergrid boundaries need to be recalculated, and the existing solutions in the hypergrid need to be relocated, as some of them may now fall in a different hypercube (see figure 1 (b)).

In practice, this grid recalculation would be required more frequently in the initial stages of the MOEA; as the algorithm converges, the hypergrid boundaries would tend to become constant. Nevertheless, a hypergrid scheme that does not require recalculation of the boundaries is desirable.

Another important aspect of hypergrid management is memory requirement. Let $N_{\text{sols}}^{\max}$ be the maximum number of solutions allowed in a given hypercube. With two objective functions, there are $N_{f1}N_{f2}$ hypercubes (see figure 1). Since each hypercube can contain up to $N_{\text{sols}}^{\max}$ solutions, we need to allocate memory for $N_{\text{sols}}^{\max}N_{f1}N_{f2}$ solutions. In the general case, with M objective functions, there are $\prod_{k=1}^{M} N_{fk}$ hypercubes, and we need to allocate memory for

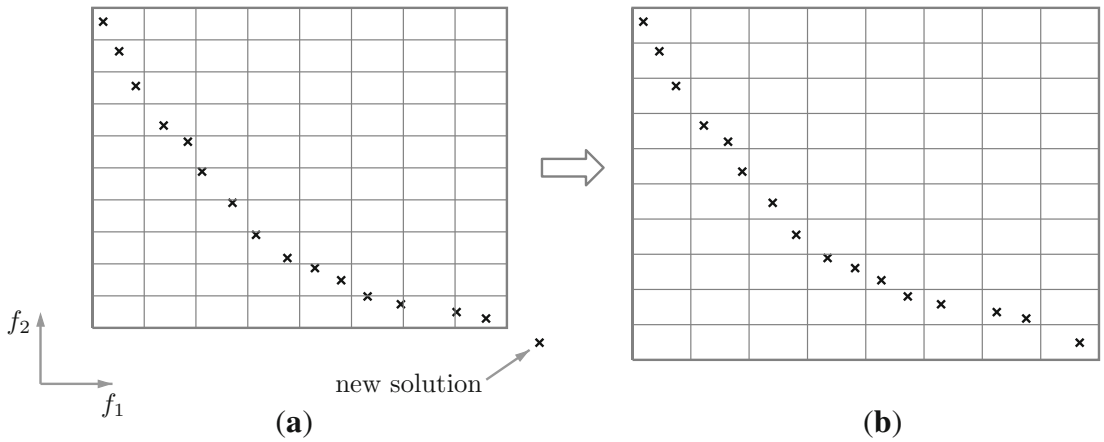

Figure 1. Schematic diagram showing the hypergrid (**a**) before and (**b**) after a new candidate lying outside the current hypergrid is admitted [18].

$N_{\text{sols}}^{\max} \prod_{k=1}^{M} N_{fk}$ solutions. Of the $\prod_{k=1}^{M} N_{fk}$ hypercubes, only a small fraction (typically 10–20 %) would be occupied and the memory allocated for the remaining hypercubes is unused. A scheme with improved memory utilization is therefore desirable.

With these two improvements in mind, viz., avoiding grid boundary recalculation and more efficient use of memory, we propose the following hypergrid management scheme.

3.2 *Proposed archive management scheme*

In the new scheme, shown in figure 2, the hypergrid does not have boundaries. It is characterized by two parameters for each of the M objective functions: a reference value f_k^{ref} and a spacing Δf_k. A given hypercube is specified by M indices, computed from its position in the objective space and the afore-mentioned hypergrid parameters. A maximum of $N_{\text{cells}}^{\max}$ hypercubes (cells) are allowed, and each hypercube can hold up to $N_{\text{sols}}^{\max}$ solutions. Memory allocation for $N_{\text{cells}}^{\max} N_{\text{sols}}^{\max}$ solutions is therefore required in this scheme. Note that the parameter $N_{\text{cells}}^{\max}$ needs to be only as large as the maximum number of occupied cells during the evolution of the population. We will refer to this new scheme as the "fixed hypergrid" (FH) scheme because the hypergrid remains fixed throughout the evolution.

The advantage of the FH scheme can be seen from the results shown in figure 3 for the CTP1 two-objective problem [23]. At the end of the first iteration, we see that 9 hypercubes (cells) have non-dominated solutions (indicated by squares in the figure). As the evolutionary algorithm progresses, some of the previously occupied cells can become empty (see the grey-coloured cells in the figure). Also, some previously empty cells can become occupied because of new non-dominated solutions appearing in the archive. After the second iteration, four of the previously occupied cells become empty, two previously empty cells become occupied, and the total number of cells is 11. At the end of the 40$^{\text{th}}$ iteration, the number of occupied cells is 15, and the number of empty cells (which were occupied at some point) is 9. We would therefore require storage for 15 cells if empty cells are removed during the evolution; if not, we would need storage for 24 cells. In

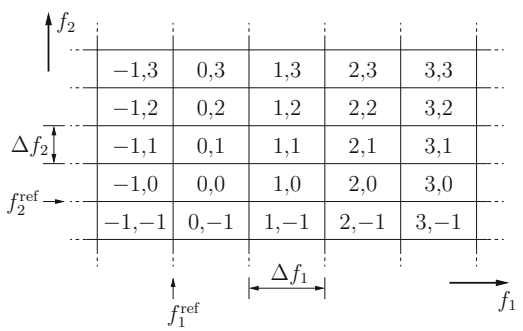

Figure 2. Proposed hypergrid management scheme for $M = 2$ (two objective functions).

contrast, with the adaptive grid scheme described earlier (figure 1), we would have started with a grid of, say, 10×8 or 80 cells (assuming roughly the same resolution as in figure 3), and kept on changing the grid boundaries as new solutions joined the archive.

The FH scheme requires two parameters to be specified by the user for each objective function, viz., a reference (f_k^{ref}) and a resolution (Δf_k). In addition, it requires the maximum number of cells ($N_{\text{cells}}^{\max}$) and maximum number of solutions per cell ($N_{\text{sols}}^{\max}$) to be specified. In practice, the user may have sufficient knowledge about the optimization problem, enabling a judicious choice for the parameters. If not, the user can use a relatively coarse grid (large values of Δf_k), run the MOEA, look at the results and then fine-tune the parameters.

It is possible during the initial stages of the MOEA that a large number of cells, which were once occupied, become empty later because of new solutions entering the archive and dominating the previous solutions. At some point, the total number of cells (both occupied and empty) may exceed $N_{\text{cells}}^{\max}$. When that happens, we can "pack" the hypergrid by removing the empty cells. After the packing operation, the total number of cells would become equal to the number of occupied cells.

Figure 4 illustrates how $N_{\text{cells}}^{\max}$ affects the total number of cells for the CTP1 example. For $N_{\text{cells}}^{\max} = 25$, packing does not take place because the total number of cells is always less than $N_{\text{cells}}^{\max}$. For $N_{\text{cells}}^{\max} = 20$, the total number of cells exceeds $N_{\text{cells}}^{\max}$ at the 6$^{\text{th}}$ iteration, and therefore a packing step is carried out, i.e., the vacant cells are removed.

4. NSGA-II with external archive

The FH scheme can be easily combined, *purely* as a storage mechanism, with an existing MOEA. For this purpose, we choose the NSGA-II (real-coded) algorithm, one of the industry-standard MOEAs. The resulting algorithm (see Algorithm 1) will be referred to as NSGA-II-FH, i.e., NSGA-II with FH.

Algorithm 1 NSGA-II-FH

1: Initialize parent population.
2: Initialize archive.
3: Evaluate parent population.
4: Assign rank and crowding distance to each individual.
5: **for** $i_{\text{gen}} = 1$ to N_{gen} **do**
6: Perform selection and crossover.
7: Perform mutation.
8: Evaluate child population.
9: Merge child and parent populations and obtain
10: mixed population.
11: Perform non-dominated sorting on mixed population
12: and obtain the next parent population.
13: Update archive.
14: **end for**

The only new step in this algorithm, as compared with the NSGA-II algorithm, is step 13, that of updating the archive. Note that the archive does not participate in the evolution of the population; it only stores non-dominated individuals from the population in a cumulative manner, as and when they become available.

Algorithm 2 describes the archive update procedure. The operations involved in archive update are comparisons (to check dominance) and copying of decision variable and objective function values when a new solution is admitted into the archive. The FH scheme has the advantage that it does not require recalculation of hypergrid boundaries. However, locating j_{cell} corresponding to a population member i_{pop} (step 14 in Algorithm 2) is more expensive in this scheme – compared with the hypergrid approach of [18] – because it involves comparing the position of i_{pop} with each of the occupied hypercubes (until a match is found). In the next section, through specific examples, we will discuss how the additional work involved in updating the archive affects the overall performance of the NSGA-II-FH algorithm with respect to the standard NSGA-II algorithm.

Algorithm 2 Update archive

1: **for** $i_{\text{pop}} = 1$ to N **do**
2: **for** $i_{\text{cell}} = 1$ to N_{cells} **do**
3: **for** each occupied solution i_{sol} in i_{cell} **do**
4: Compare i_{pop} and i_{sol} for dominance.
5: **if** i_{pop} and i_{sol} are non-dominating **then**
6: **if** i_{pop} and i_{sol} are identical **then**
7: Next i_{pop}
8: **end if**
9: **else if** i_{sol} dominates **then**
10: Next i_{pop}
11: **end if**
12: Find all solutions in archive dominated by i_{pop}
13: and remove them.
14: Find j_{cell} corresponding to i_{pop}.
15: **if** j_{cell} exists **then**
16: **if** j_{cell} is full **then**
17: Randomly remove one solution from j_{cell}.
18: **end if**
19: Add i_{pop} to j_{cell}.
20: **else**
21: **if** $N_{\text{cells}} = N_{\text{cells}}^{\max}$ **then**
22: **if** there are vacant cells **then**
23: Pack archive (Remove vacant cells).
24: Create a new cell and add i_{pop} to it.
25: **else**
26: Declare "archive full" and stop.
27: **end if**
28: **end if**
29: **end if**
30: **end for**
31: **end for**
32: **end for**

5. Results and discussion

We consider four multi-objective optimization problems. The algorithms NSGA-II and NSGA-II-FH are used for each of the problems, keeping the algorithmic parameters the same, viz., crossover probability $p_c = 0.8$, mutation probability $p_m = 1/L$ (where L is the number of decision variables), distribution index for crossover $\eta_c = 10$ and distribution index for mutation $\eta_m = 10$. The first two problems are test problems commonly used in the literature, and the next two are optimization problems from the electrical engineering domain.

5.1 *VNT problem*

The VNT problem [24] has $L = 2$, $M = 3$ (i.e., two decision variables and three objective functions). The functions given by

$$f_1(\mathbf{x}) = 0.5(x_1^2 + x_2^2) + \sin(x_1^2 + x_2^2)$$
$$f_2(\mathbf{x}) = (3x_1 - 2x_2 + 4)^2/8 + (x_1 - x_2 + 1)^2/27 + 15$$
$$f_3(\mathbf{x}) = \frac{1}{x_1^2 + x_2^2 + 1} - 1.1\exp\left[-(x_1^2 + x_2^2)\right]$$

are to be minimized, with the decision variables in the range $-3 \le (x_1, x_2) \le 3$. We keep the population size constant ($N = 60$) and carry out a *single* run of NSGA-II and NSGA-II-FH for different values of N_{gen} (number of generations). The hypergrid parameters used for the NSGA-II-FH algorithm are $N_{\text{cells}}^{\max} = 1000$, $N_{\text{sols}}^{\max} = 10$, $f_1^{\text{ref}} = f_2^{\text{ref}} = f_3^{\text{ref}} = 0$, $\Delta f_1 = 0.1$, $\Delta f_2 = 0.01$, $\Delta f_3 = 0.1$.

Figures 5 and 6 show the results for the objective functions f_1 and f_3. The true Pareto front is also shown for comparison. The crosses represent APO solutions produced by the concerned algorithm. It can be observed that the NSGA-II-FH method produces a significantly larger number of solutions for each value of N_{gen}. Figure 7 (a) and (b) shows all three objective function plots for a population size $N = 60$ and $N_{\text{gen}} = 200$.

The advantage of using an external archive to store all solutions visited by the population cumulatively is clear from the figures. This was in fact one of the motivations behind using an external archive in the AMGA algorithm [11], for example. The NSGA-II-FH results demonstrate that an external archive is beneficial even when it does not participate in the underlying algorithm.

It needs to be emphasized that the results in figures 5 and 6 are for a single run as opposed to the normal practice of combining the results of several independent runs. With a single run, the maximum number of APO solutions produced by the NSGA-II algorithm is limited by the population size. For the NSGA-II-FH algorithm, the number of APO solutions in the archive is not limited by the population size; it can be much larger, as seen from the

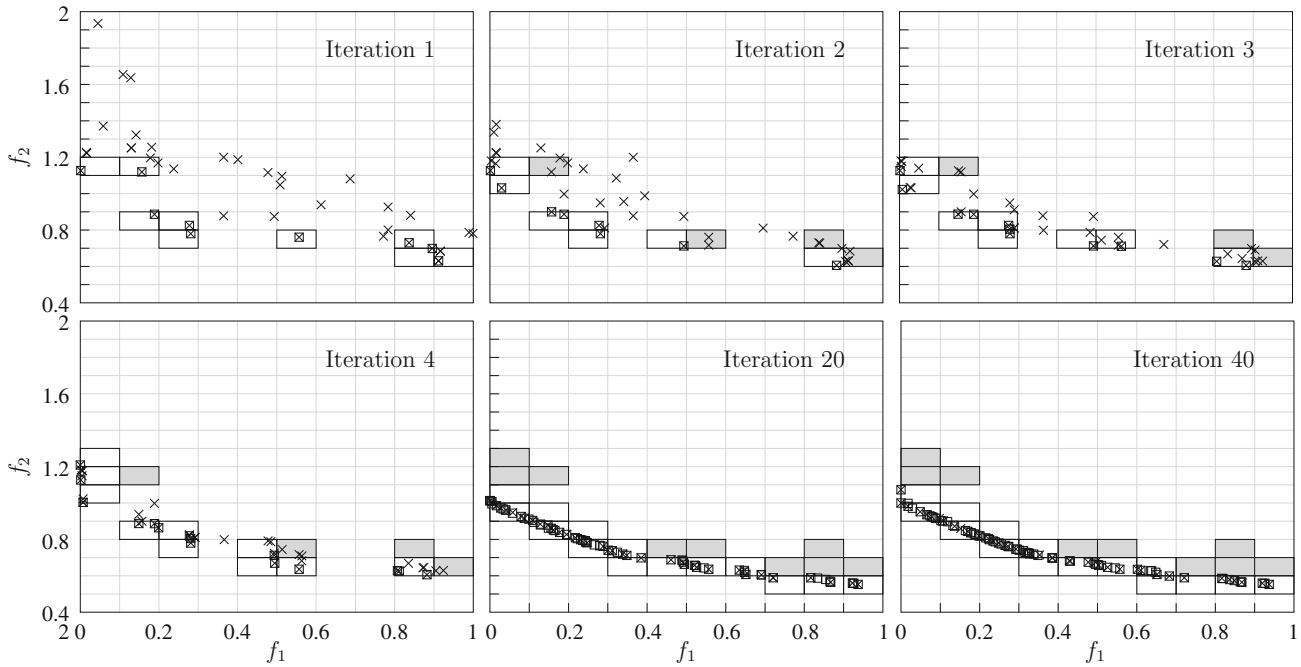

Figure 3. Illustration of the fixed hypergrid scheme using the CTP1 two-objective problem [23]. The real-coded NSGA-II algorithm with a population size of $N = 40$ was used. Crosses indicate the current positions of individuals in the population, and squares indicate all of the non-dominated solutions obtained up to the given iteration.

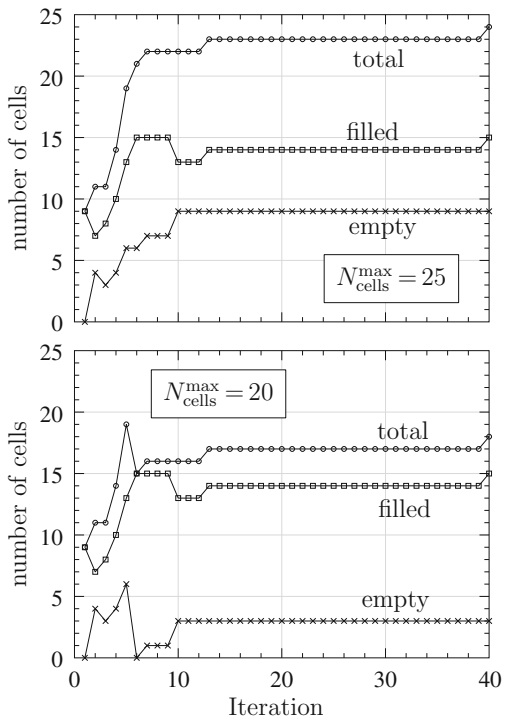

Figure 4. Filled, empty and total number of cells versus iteration for the CTP1 example for two values of $N_{\text{cells}}^{\max}$.

afore-mentioned figures. In other words, NSGA-II gives only a sample – a snapshot – of all the APO solutions visited by the population while NSGA-II-FH gives a more complete picture. This is clearly a major advantage of the NSGA-II-FH algorithm.

5.2 *KUR problem*

In this problem [23], the functions

$$f_1(\mathbf{x}) = \sum_{i=1}^{2}\left[-10\exp\left(-0.2\sqrt{x_i^2 + x_{i+1}^2}\right)\right]$$

$$f_2(\mathbf{x}) = \sum_{i=1}^{3}\left[|x_i|^{0.8} + 5\sin(x_i^3)\right]$$

are to be minimized, with $-5 \le (x_1, x_2, x_3) \le 5$. The results for this example are shown in figure 8 for a population size of 40 and $N_{\text{gen}} = 500$. Once again, the NSGA-II-FH algorithm results in a significantly larger number of APO solutions. The decision maker may in fact find the number of solutions obtained in a single run of the NSGA-II-FH algorithm to be adequate and will therefore not need to perform additional independent runs of the algorithm. In contrast, with the NSGA-II algorithm, the only way to obtain a larger number of APO solutions is to either increase the population size or perform several independent

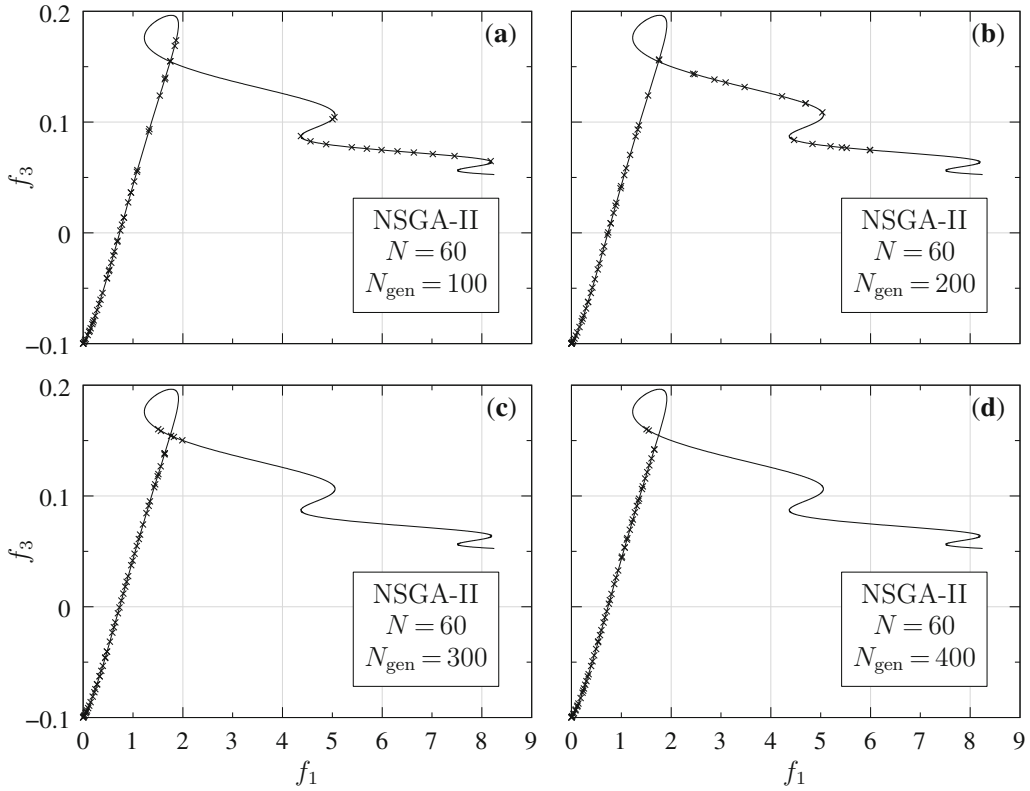

Figure 5. Approximate Pareto-optimal solutions for the VNT problem obtained with the NSGA-II algorithm for a population size $N = 60$ and different values of N_{gen}: (**a**) $N_{\text{gen}} = 100$, (**b**) $N_{\text{gen}} = 200$, (**c**) $N_{\text{gen}} = 300$ and (**d**) $N_{\text{gen}} = 400$.

runs and then combine the resulting APO solutions. In either case, the computation time would increase.

In these two examples, the function evaluation cost is relatively low. In the next two examples, we consider engineering problems in which function evaluation is much more expensive.

5.3 *Inverting amplifier*

Consider the inverting amplifier shown in figure 9 (a). It is well known that a high gain and a high bandwidth for an amplifier are conflicting objectives. If a simple op-amp model is used, the gain versus bandwidth relationship can be obtained analytically. However, if a realistic op-amp model is used, circuit simulation is required.

The optimization problem considered here can be stated as follows. There are two design variables, the resistances R_1 and R_2, with $1\,\text{k}\Omega < R_1 < 10\,\text{k}\Omega$, $5\,\text{k}\Omega < R_2 < 50\,\text{k}\Omega$. There are three objective functions: gain, bandwidth (i.e., the high cut-off frequency f_H) and input resistance of the amplifier. The gain and bandwidth are to be maximized, and the input resistance is to be minimized. Two constraints are imposed: a minimum gain of 5 and a minimum bandwidth of 1 kHz.

Computation of the objective functions involves setting the parameter values (R_1 and R_2) in a circuit file, simulating the circuit using the circuit simulator NGSPICE [25] and then computing the mid-band gain, cut-off frequency and input resistance from the simulator output. Note that, compared with our first two examples, function evaluations are far more expensive in this case.

Each algorithm was run once with $N = 20$ and $N_{\text{gen}} = 100$. The hypergrid parameters for the NSGA-II-FH algorithm were $N_{\text{cells}}^{\max} = 1000$, $N_{\text{sols}}^{\max} = 3$, $f_1^{\text{ref}} = f_2^{\text{ref}} = f_3^{\text{ref}} = 0$, $\Delta f_1 = 1$, $\Delta f_2 = 1\,\text{kHz}$, $\Delta f_3 = 0.5\,\text{k}\Omega$.

Figure 9 (b) and (c) shows the gain versus frequency plots. Figure 10 (a) and (b) shows all three objective functions. It is observed once again that NSGA-II-FH produces a substantially larger number of APO solutions.

5.4 *CMOS inverter cascade*

As the second engineering problem, we consider the design of a cascade of CMOS inverters (see figure 11 (a)) for driving a large capacitive load [26]. For a given number of stages N' and a given value of the load capacitance C_L, the *W/L* ratios of the transistors are to be designed. Simplified analytical treatment for minimizing the delay (between the input and output waveforms) is possible with certain approximations [26]. This procedure gives a first-order design. However, the power consumption of the circuit, which also varies substantially with the *W/L* ratios of the

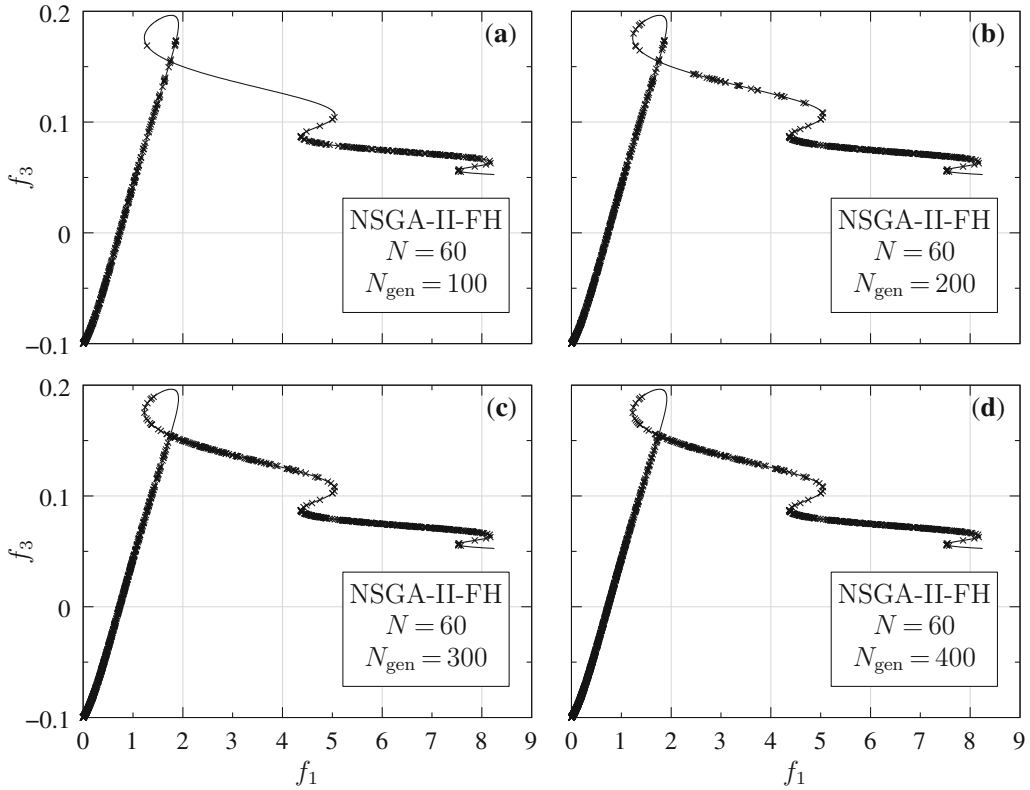

Figure 6. Approximate Pareto-optimal solutions for the VNT problem obtained with the NSGA-II-FH algorithm for a population size $N = 60$ and different values of N_{gen}: (**a**) $N_{\text{gen}} = 100$, (**b**) $N_{\text{gen}} = 200$, (**c**) $N_{\text{gen}} = 300$ and (**d**) $N_{\text{gen}} = 400$.

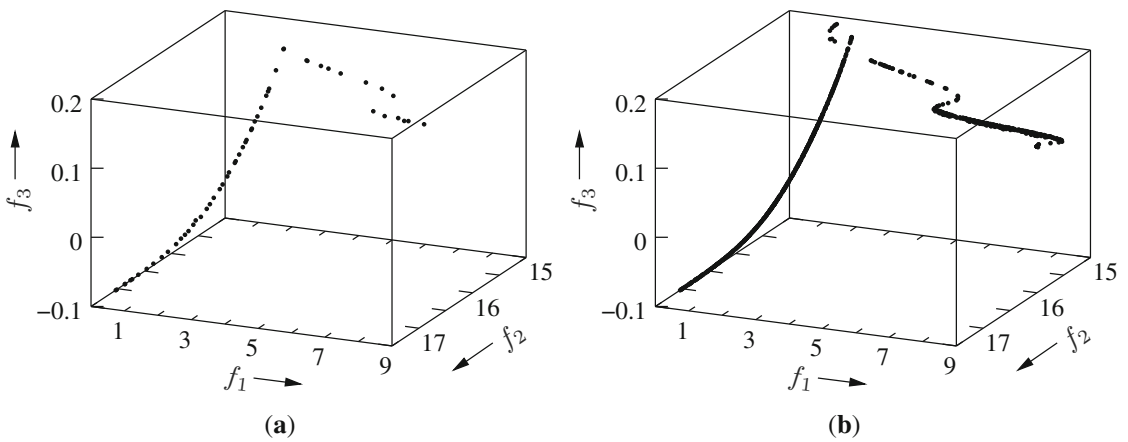

Figure 7. Three-dimensional plots showing f_1, f_2, f_3 for the VNT problem. The population size is $N = 60$ and number of generations is $N_{\text{gen}} = 200$: (**a**) NSGA-II result and (**b**) NSGA-II-FH result.

transistors, is not considered in this first-order design. A multi-objective optimization procedure using circuit simulation is therefore desirable.

The optimization problem can be stated as follows. Minimize the delay (between the input and output waveforms) and power consumption of the circuit (or the power supplied by the V_{DD} supply) subject to the following conditions. (a) The width of the p-transistor of any stage is twice that of the n-transistor of the same stage. (b) The channel length is equal to the minimum feature size λ for all transistors. (c) W_1^n, the width of M_1^n, is equal to 2λ. (d) The width for any transistor cannot exceed $W_{\max}$ (taken as 400λ). Since we assume $W_k^p/W_k^n = 2$, it implies that the upper limit for the n-channel transistor widths is $W_{\max}/2$.

We consider an inverter cascade with $N' = 5$. In this case, the decision variables are the widths of the transistors

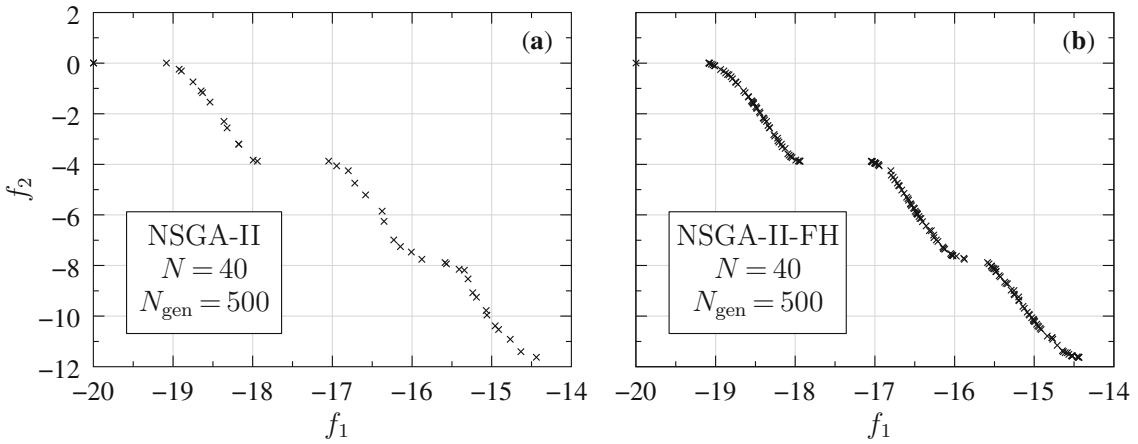

Figure 8. Approximate Pareto-optimal solutions for the KUR problem obtained for a population size $N = 40$ and $N_{\text{gen}} = 500$. (**a**) NSGA-II algorithm and (**b**) NSGA-II-FH algorithm.

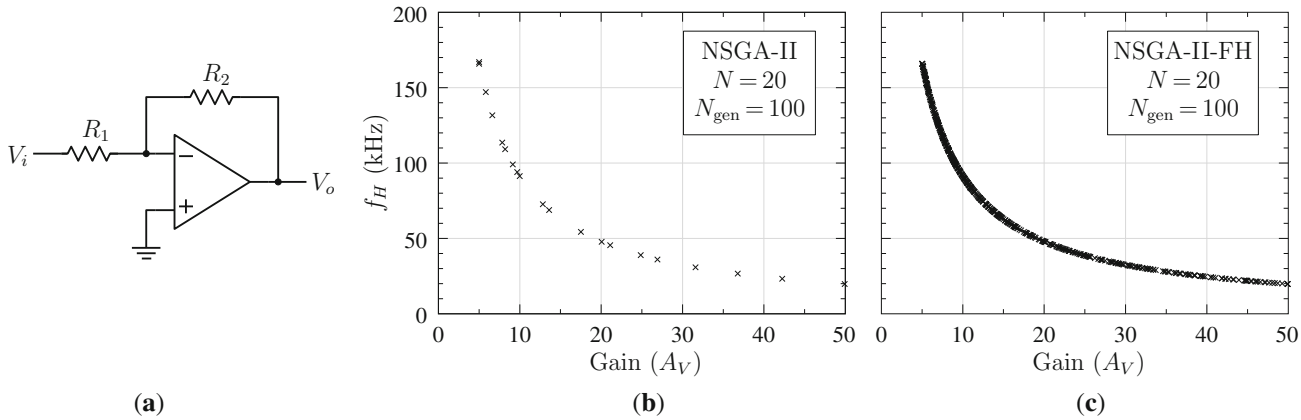

Figure 9. (**a**) Op-amp based amplifier circuit, (**b**) NSGA-II result and (**c**) NSGA-II-FH result obtained for a population size $N = 20$ and $N_{\text{gen}} = 100$.

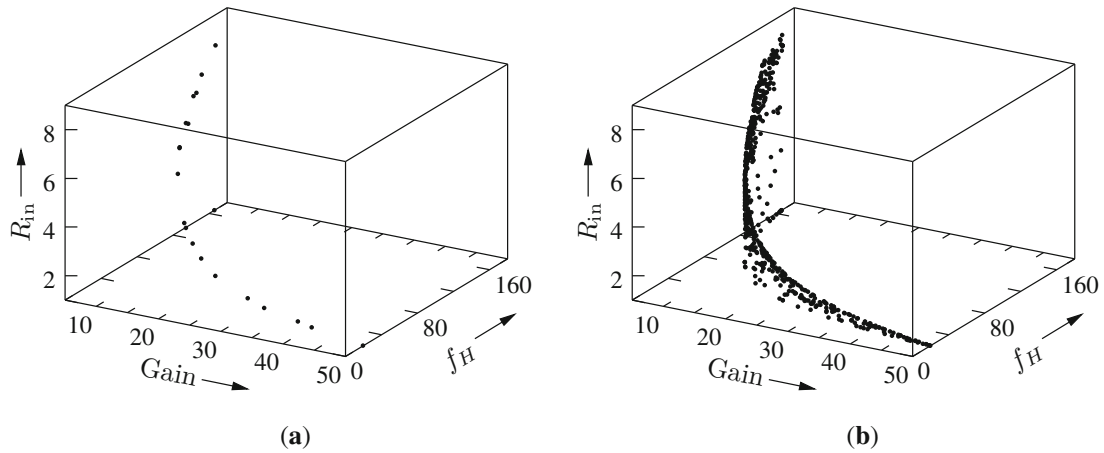

Figure 10. Three-dimensional plots showing all three objective functions for the amplifier problem of figure 9. R_{in} is in kΩ, and f_H is in kHz: (**a**) NSGA-II result and (**b**) NSGA-II-FH result (population size $N = 20$, $N_{\text{gen}} = 100$).

M_2^n, M_3^n, M_4^n, M_5^n in figure 11 (denoted by W_2^n, W_3^n, W_4^n, W_5^n, respectively). The widths of the p-transistors are related to the widths of the n-transistors by $W_k^p/W_k^n = 2$ and therefore not treated as independent parameters. The decision

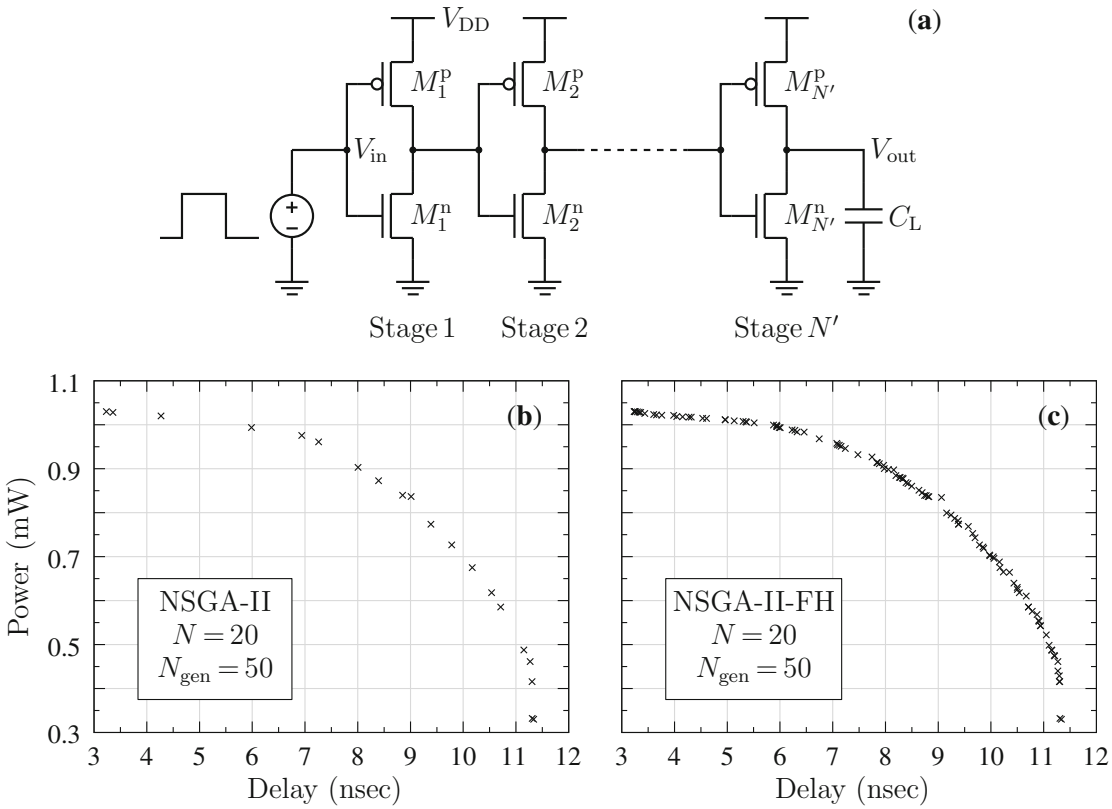

Figure 11. (**a**) Cascade of inverters used in driving a large load capacitance, (**b**) NSGA-II result and (**c**) NSGA-II-FH result obtained for a population size $N = 20$ and $N_{gen} = 50$.

variables take on integer values (multiples of λ) from 2 to 400. Note that simple *enumeration* is not a practical option in this case because of the large size of the decision variable space (about 400^4 or 2.56×10^{10}).

As in the op-amp example, the NGSPICE circuit simulator is used for function evaluation. A circuit file is first prepared. During the optimization process, the circuit file gets modified (with the strings corresponding to W_2^n, W_3^n, W_4^n, W_5^n replaced with their values) and is passed on to the circuit simulator. From the simulation results obtained, the rise and fall times, and power consumption, are computed. The sum of the rise and fall times is treated as the delay objective function value. The results are shown in figure 11 (b) and (c). The advantage of the NSGA-II-FH method is again clearly seen.

The benefit of the NSGA-II-FH algorithm comes with an additional computational cost. To gauge the impact of this additional cost, we present in table 1 the CPU time required for the two algorithms on a desktop computer (Linux) with 3.3 GHz clock and 4 GB RAM without any parallelization. From the table, we observe that NSGA-II-FH takes substantially longer than NSGA-II for the VNT problem. On the other hand, for the amplifier problem, there is virtually no difference between the two because the time taken by function evaluations dominates in this case, and the overhead due to archive manipulation is negligibly small.

Table 1. CPU time taken by NSGA-II and NSGA-II-FH algorithms for the VNT andamplifier problems.

	VNT ($N = 60$)		Amplifier ($N = 20$)	
N_{gen}	NSGA-II	NSGA-II-FH	NSGA-II	NSGA-II-FH
100	13 ms	31 ms	33 s	34 s
200	21 ms	77 ms	67 s	67 s
300	33 ms	128 ms	101 s	101 s
400	41 ms	177 ms	134 s	135 s

6. Conclusion

A new scheme for external archive management in multi-objective optimization has been presented. It is demonstrated that the new scheme, combined with the NSGA-II algorithm purely for storage purposes, produces a substantially larger number of APO solutions as compared with the original NSGA-II algorithm. The new scheme does not need hypergrid boundaries to be specified, thus making grid adjustment unnecessary. It also takes less memory compared with similar hypergrid approaches used earlier.

It is shown that additional CPU time is required for the extra computations related to archive management in the

new NSGA-II-FH algorithm. This additional time is found to be significant for optimization problems in which function evaluations are relatively inexpensive. On the other hand, for many practical problems of interest, the function evaluation time is large, and the overhead due to archive management is then negligible. In these problems, there is virtually no difference between NSGA-II and NSGA-II-FH in terms of computing time. Two examples in this category are also presented.

The new archive management scheme has also been incorporated in the MOPSO algorithm [18] and has been found to be effective for a variety of multi-objective optimization problems. The results will be presented elsewhere.

References

[1] Parks G T and Miller I 1998 Selective breeding in a multiobjective genetic algorithm. In: *Parallel Problem Solving from Nature*, pp. 250–259

[2] Zitzler E and Thiele L 1999 Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Trans. Evol. Comput.* 3: 257–271

[3] Knowles J D and Corne D W 2000 Approximating the nondominated front using the Pareto archived evolution strategy. *IEEE Trans. Evol. Comput.* 8: 149–172

[4] Corne D W, Knowles J D and Oates M J 2000 The Pareto envelope-based selection algorithm for multiobjective optimisation. In: *Parallel Problem Solving from Nature*, pp. 839–848

[5] Corne D W, Knowles J D, Oates M J and Martin J 2001 PESA-II: region-based selection in evolutionary multiobjective optimization. In: *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, pp. 283–290

[6] Zitzler E, Laumanns M and Thiele L 2001 *SPEA2: improving the strength Pareto evolutionary algorithm*. TIK-Rep. 103, pp. 1–21

[7] Laumanns M, Thiele L, Deb K and Zitzler E 2002 Combining convergence and diversity in evolutionary multiobjective optimization. *Evol. Comput.* 10: 263–282

[8] Fieldsend J E, Everson R M and Singh S 2003 Using unconstrained elite archives for multi-objective optimisation. *IEEE Trans. Evol. Comput.* 7: 305–323

[9] Huang V L, Suganthan P N, Qin A K and Baskar S 2005 *Multiobjective differential evolution with external archive and harmonic distance-based diversity measure*. School of Electrical and Electronic Engineering, Nanyang Technological University Technical Report 1–25

[10] Hallam N, Blanchfield P and Kendall G 2005 Handling diversity in evolutionary multiobjective optimization. In: *Proc. IEEE Congress Evolut. Comput.*, vol. 3, pp. 2233–2240

[11] Tiwari S, Koch P, Fadel G and Deb K 2008 AMGA: an archive-based micro genetic algorithm for multi-objective optimization. In: *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, pp. 729–736

[12] Martínez S Z and Coello C A C 2010 An archiving strategy based on the convex hull of individual minima for MOEAs. In: *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1–8

[13] Sharma D and Collet P 2010 An archived-based stochastic ranking evolutionary algorithm (ASREA) for multi-objective optimization. In: *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, pp. 479–486

[14] Tiwari S, Fadel G and Deb K 2011 AMGA2: improving the performance of the archive-based micro-genetic algorithm for multi-objective optimization. *Eng. Optim.* 43: 377–401

[15] Cai X, Li Y, Fan Z and Zhang Q 2015 An external archive guided multi-objective evolutionary algorithm based on decomposition for combinatorial optimization. *IEEE Trans. Evol. Comput.* 19: 508–523

[16] Song M and Chen D 2018 An improved knowledge-informed NSGA-II for multi-objective land allocation (MOLA). *Geo-spatial Information Science*, pp. 1–15

[17] Mostaghim S and Teich J 2003 Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO). In: *Proceedings of the Swarm Intelligence Symposium*, pp. 26–33

[18] Coello C A C, Pulido G T and Lechuga M S 2004 Handling multiple objectives with particle swarm optimization. *IEEE Trans. Evol. Comput.* 8: 256–279

[19] Alvarez-Benitez J E, Everson R M and Fieldsend J E 2005 A MOPSO algorithm based exclusively on Pareto dominance concepts. In: *Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization*, pp. 459–473

[20] Tripathi P, Bandyopadhyay S and Pal S K 2007 Adaptive multi-objective particle swarm optimization algorithm. In: *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 2281–2288

[21] Zhang Y, Gong D-W and Ding Z 2012 A bare-bones multi-objective particle swarm optimization algorithm for environmental/economic dispatch. *Inf. Sci.* 192: 213–227

[22] Han H, Lu W and Qiao J 2017 An adaptive multiobjective particle swarm optimization based on multiple adaptive methods. *IEEE Trans. Cybern.* 47: 2754–2767

[23] Deb K, Pratap A, Agarwal S and Meyarivan T 2002 A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* 6: 182–197

[24] Deb K 2009 *Multi-objective optimization using evolutionary algorithms*. Chichester: Wiley

[25] Nenzi P 2014 Ngspice circuit simulator release 26.

[26] Baker R J, Li H W and Boyce D E 1998 *CMOS circuit design, layout, and simulation*. New York: IEEE Press