Paper

Application of multiple criteria evolutionary algorithms to vector optimisation, decision support and reference point approaches

Marcin Szczepański and Andrzej P. Wierzbicki

Abstract — Multiple criteria evolutionary algorithms, being essentially parallel in their character, are a natural instrument of finding a representation of entire Pareto set (set of solutions and outcomes non-dominated in criteria space) for vector optimisation problems. However, it is well known that Pareto sets for problems with more than two criteria might become complicated and their representation very time-consuming. Thus, the application of such algorithms is essentially limited to bi-criteria problems or to vector optimisation problems with more criteria but of simple structure. Even in such cases, there are problems related to various important aspects of vector optimisation, such as the uniformity of representation of Pareto set, stopping tests or the accuracy of representing Pareto set, that are not fully covered by the broad literature on evolutionary algorithms in vector optimisation. These problems and related computational tests and experience are discussed in the paper. In order to apply evolutionary algorithms for decision support, it would be helpful to use them in an interactive mode. However, evolutionary algorithms are in their essence global and of batch type. Nevertheless, it is possible to introduce interactive aspects to evolutionary algorithms by focusing them on a part of Pareto set. The results of experimental tests of such modifications of evolutionary algorithms for vector optimisation are presented in the paper. Another issue related to vector optimisation problems with more than two criteria is the computational difficulty of estimating nadir points of Pareto set. The paper describes the use of diverse variants of evolutionary algorithms to the estimation of nadir points, together with experimental evidence.

Keywords* — *evolutionary algorithms, vector optimisation, nadir point estimation, reference point techniques.

1. Evolutionary algorithms in vector optimisation: general comments

There are many excellent reviews of evolutionary algorithms used in vector optimisation [3–5, 10, 12]. Most of them, however, treat evolutionary or genetic algorithms as goals in themselves, as given tools that should be further developed and put into use. In this paper, we concentrate rather on the use of such algorithms for solving various tasks of vector optimisation or multiple criteria analysis for decision support.

First, let's recall the traditional distinction between genetic and evolutionary algorithms: genetic algorithms rely on binary representation of individuals, while evolutionary algorithms admit real-valued (computational) representations. For vector-valued representations, evolutionary algorithms are more appropriate. On the other hand, special methods developed for genetic algorithms can be also usefully translated into evolutionary algorithms.

Next, we observe that evolutionary algorithms are applied to vector optimisation in order to obtain accurate representation of the Pareto set (or any modified concept of a non-dominated set). Being inherently parallel, evolutionary algorithms are a natural approach to the problem of representing a complicated set. However, research on truly parallel or distributed implementations of evolutionary algorithms is scarce. Thus, the application of such algorithms is essentially limited to bi-criteria problems or very simple vector optimisation problems with more criteria. Accurate representation of more complicated Pareto sets using evolutionary algorithm still requires huge computation efforts.

On the other hand, practical applications of vector optimisation to decision support require interactive multiple criteria analysis [11], where instead of computing a single Pareto set, various characteristics of selected variants or parts of Pareto sets are needed for subsequent formulations of the problem being analysed. Such cases include utopia points, nadir points, neutral compromise points of Pareto sets and, finally and most importantly for interactive applications – representations of selected segments of Pareto sets. While evolutionary algorithms might be useful for obtaining such characteristics, little attention was given to such applications. Generally speaking, the same fact can be stated as follows: *since evolutionary algorithms are global and non-interactive in their nature, the challenge in their applications for multiple criteria analysis is to make them more local and interactive*. While this paper does not resolve all problems related to this challenge, it tries to move in this direction – by treating evolutionary algorithms not as main goal in itself, but as a way of addressing various tasks of multiple criteria analysis.

2. Modifications of evolutionary algorithms in vector optimisation

2.1. Representation of individual

By *individual* in genetic or evolutionary algorithms, we consider a current solution point together with additional parameters, typically characterising its mutation potential by specifying the dispersion σ. In vector optimisation or multiple criteria analysis, current solution is typically represented by a vector of decision variables $x \in R^n$ and vector of decision outcomes or criteria $q \in R^k$. Dispersion parameters are related to decision variables and can be represented by a vector of the same dimension. Thus, an individual is represented by:

$$ind = (x, \sigma, q) \in R^{2n+k}. \tag{1}$$

2.2. Constraints

Constraints on decision variables (either in equation or inequality form) define the permissible set of decisions:

$$X_0 = \left\{ x \in R^n : \begin{array}{l} g_i(x) \geq 0, i \in I \\ h_i(x) = 0, i \in E \end{array} \right\}. \tag{2}$$

In genetic algorithms, if x is not in X_0, the individual is simply discarded. This may lead, however, to quite long computations if the set X_0 has a complicated structure. Therefore, we shall use a method typically adopted in evolutionary algorithms to represent constraints – applying penalty functions. There are many types of penalty functions (internal, external, exact, shifted, etc. – see e.g. [11]). With evolutionary algorithms that do not need derivatives of optimised functions, it is best to use exact non-differentiable external penalty functions of the type $|h_i(x)|$ and $|g_i(x)_-| = \min\big(g_i(x), 0\big)|$ (with sufficiently large penalty coefficients), which are added to each criterion value – if it is minimised or subtracted – if maximised.

2.3. Cross-breeding

Cross-breeding is a typical evolutionary operation. In vector optimisation, cross-breeding applies to two parent individuals represented by decision variable vectors x_1 and x_2; their successor x' may be determined as follows:

$$x' = ax_1 + (1-a)x_2, \tag{3}$$

where the parameter a is a random variable from the interval $a \in [0,1]$. This is called *basic arithmetic cross-breeding*, while *extended arithmetic cross-breeding* applies to each component x'_i of the vector x' with separately generated random coefficients a_i. There are several other variants of cross-breeding, such as *heuristic cross-breeding*, not discussed here.

2.4. Mutation

In vector optimisation, mutation is applied to every component x_i of the decision variable vector x (usually, mutation is additionally applied to a successor of cross-breeding) by selecting a random variable with a normal distribution and modifying the component x_i by this variable with a corresponding dispersion coefficient:

$$\begin{aligned} \xi_i^x &\in N(0,1), \\ x'_i &= x_i + \sigma'_i \xi_i^x . \end{aligned} \tag{4}$$

Additionally, the dispersion coefficient is modified randomly, but usually slowly decreased after (or before) each mutation. This decreasing modification of dispersion parameters slows down mutations when approaching solutions. In vector optimisation, it results in coming closer to the Pareto set.

2.5. Selection

Selection is responsible for convergence of a genetic algorithm towards optimal solutions and applies to selection of parent individuals (selection in reproduction); there are numerous methods of such selection, not discussed here. In evolutionary algorithms, succession may substitute for selection. This means choosing the μ as best individuals from population $\mu + \lambda$ (so-called $\mu + \lambda$ succession strategy; μ denotes here a population from parent individuals, λ the corresponding population of successors) in some way. Another strategy consists of simply substituting parent population μ by successor population λ (the so-called μ, λ strategy). For vector optimisation purposes, succession is superior to selection.

2.6. Pareto ranking

Succession process includes multiple stages to uniformly approximate Pareto set by an evolutionary algorithm. First, we use *Pareto ranking of a population*, then apply special *niched methods* for preserving uniformity of representation, and finally use special *succession methods*. We will describe all of them below.

Pareto ranking consists of attaching a rank value (the lower the better) to each individual. Goldberg [2] has proposed to give rank 1 to each non-dominated individual in population. Next, we delete the non-dominated individuals and determine non-dominated individuals in remaining part of population, giving them rank 2. We continue the process with increasing rank values until each individual has a rank value. Then we can either select successor population of given number of individuals according to lowest rank values, or – as proposed by Goldberg – determine the probability of reproduction depending on rank value (which is actually a selection, not a succession mechanism).

Another ranking method proposed by Fonseca and Fleming [4] involves assigning each individual a rank value of 1 plus the number of other individuals dominating this individual. This method provides for more differentiation of a population than Goldberg method.

Having a rank value, it is easy to determine a fitness indicator $fit(x)$ – for example, by defining it as inverse of the rank value.

2.7. Niched methods

Having a fitness value or fitness function for Pareto ranking, it is easy to apply the basic principle of evolutionary algorithms – the *survival of the fittest* individuals.

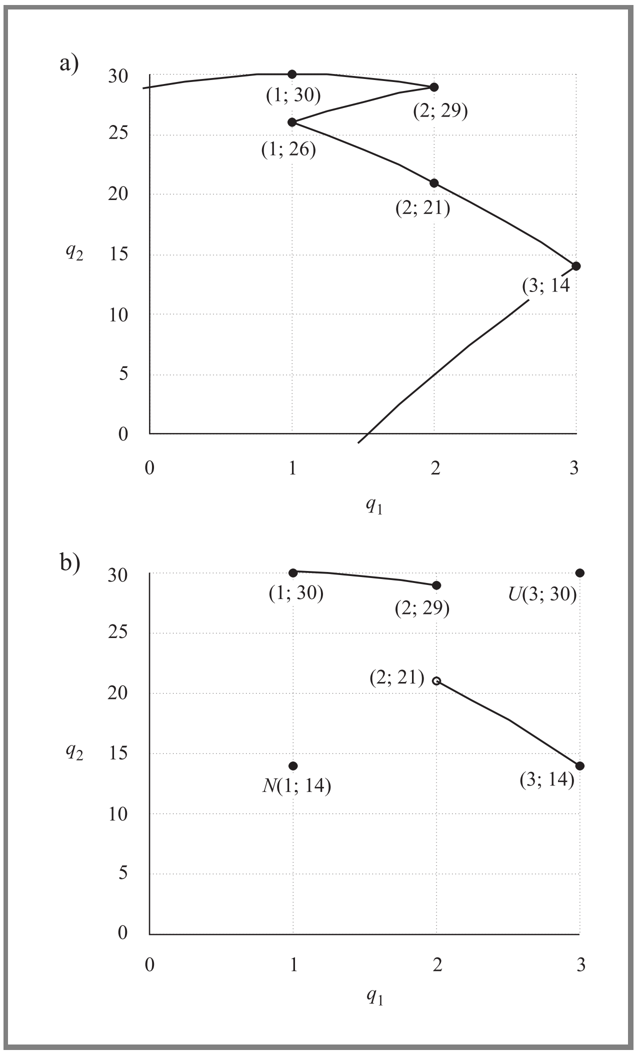

Fig. 1. The set of attainable criteria values (a) and the Pareto set (b) for the nonlinear example.

However, such a method does not result in a uniform representation of the Pareto set. The fittest individuals can form an elite close to each other, representing only an "easy" part of Pareto set. Such degeneration of the *survival of the fittest* principle can be illustrated by a relatively simple, but nonlinear example (Fig. 1). We maximize two criteria functions (with $-0.5 \le x \le 6$):

$$\begin{aligned} \max: q_1(x) &= \left\{ \begin{array}{cc} x+2 & x \le 1 \\ -x+4 & 1 < x \le 3 \\ x-2 & 3 < x \le 4 \\ -x+6 & x > 4 \end{array} \right\}, \\ \max: q_2(x) &= -x^2+10x+5\,. \end{aligned} \tag{5}$$

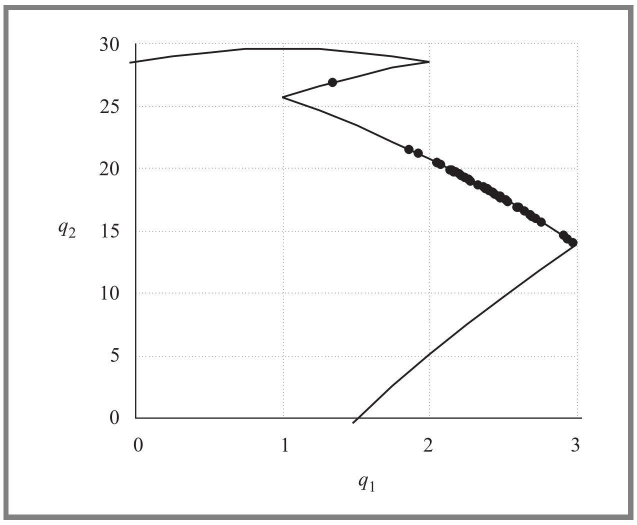

Fig. 2. Non-uniform representation of Pareto set with a simple *survival of the fittest* evolutionary algorithm (population size: 50, 200 generations).

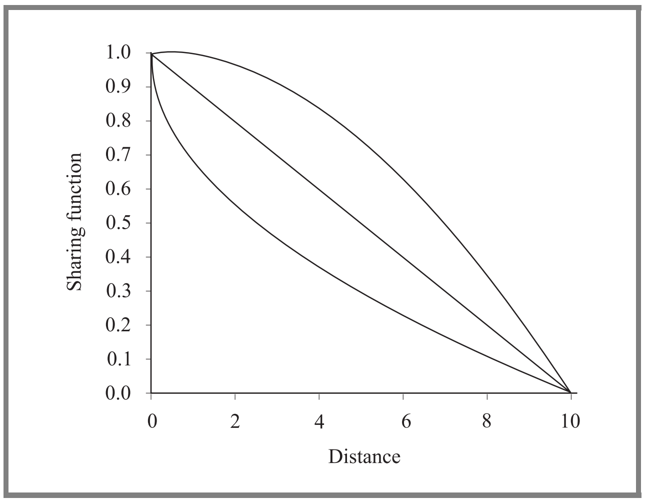

Fig. 3. Examples of sharing functions.

Application of a simple *survival of the fittest* algorithm here results in a degenerated representation of the Pareto set, concentrating on the "easy part" of the set (Fig. 2).

In order to overcome this difficulty, we must penalise the fitness function for individuals being too close to each other.

With this aim, we define a *sharing function* depending on a distance of two individuals, say x and x_0. This sharing function sh must have the following properties:

$$0 \le sh(x - x_0) \le 1, \text{ for any distance } |x - x_0|$$
$$sh(0) = 1$$
$$\lim sh(x - x_0) = 0, |x - x_0| \to \infty. \qquad (6)$$

Sharing functions shown in Fig. 3 belong to the family:

$$sh(x - x_0) = \begin{cases} 1 - \left(\frac{|x-x_0|}{D}\right)^p & |x - x_0| < D \\ 0 & |x - x_0| \ge D \end{cases}, \qquad (7)$$

where D is a diameter of a *niche*.

The so-called *niched methods* consist of modifying fitness values $fit(x)$ for a given individual x, reciprocal to the sharing function:

$$fit'(x) = \frac{fit(x)}{1 + m(x)}, \qquad (8)$$

where $m(x)$ is a sum of sharing functions over other non-dominated individuals y in given population:

$$m(x) = \sum_y sh\big(d(x, y)\big). \qquad (9)$$

Figure 4 illustrates effectiveness of such niched methods in preventing degeneration through cross-breeding of too close individuals.

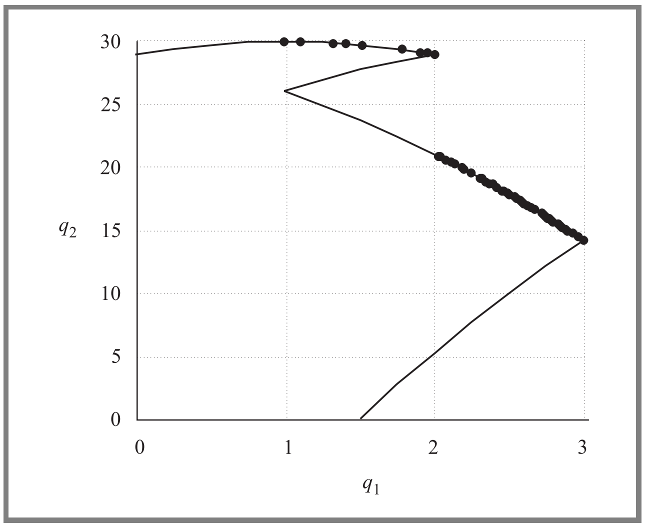

Fig. 4. Effectiveness of a *niched method* with $D = 0.1$ (population size: 50, 200 generations).

We see it is necessary to use niched methods in evolutionary algorithms of vector optimisation not only in order to obtain a uniform representation of Pareto set, but also to prevent degenerate populations resulting from naive direct application of the "survival of the fittest" principle.

2.8. Stopping tests

Before discussing succession methods, stopping tests for entire algorithm should be discussed. Stopping test for evolutionary and genetic optimisation algorithms are much less developed than for analytical optimisation methods. If the optimal value of an optimised function is known (which happens only in very special cases) then the distance from this optimal value can be used for a stopping test. Otherwise, one must limit the number of iterations in the algorithm (number of generations in a genetic or evolutionary algorithm) and hope for a good accuracy. Another stopping test is based on the speed of change of an approximation of the solution: work stops when changes fall below certain level.

For vector optimisation, the issue of stopping tests is more complicated. We can rely on a given number of iterations or generations, but cannot easily use the speed of change, because we approximate or represent an entire Pareto set and the uniformity of this representation is also a goal. A substitute for the speed of change might be a comparison of two subsequent generations and checking how many individuals in the next generation dominate some individuals in the former generation. Figure 5 shows example of such computation.

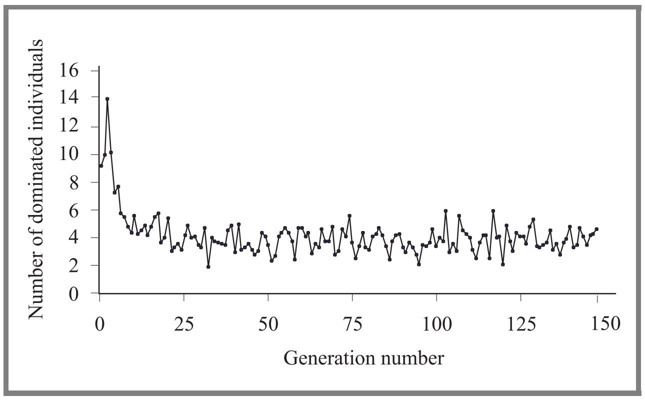

Fig. 5. Average numbers of dominated individuals between generations for a typical evolutionary algorithm.

We see such a stopping test cannot be very reliable. Other tests, however, might be related to special features of vector optimisation. One relates to the uniformity of Pareto set representation, which can be represented by average value of sharing function $m(x)$ as defined by Eq. (9). Another relates to the concept of utopia and nadir points for a Pareto set. For an approximation of Pareto set obtained in a subsequent generation numbered here by i, it is relatively easy (see also point **4**) to compute utopia points q_i^U ("lowest" points dominating entire Pareto set) as well as nadir points g_i^N ("highest" points dominated by the entire Pareto set). If the approximation of a Pareto set converges to the actual Pareto set, the distance between the approximations of utopia and nadir points:

$$un(i) = |q_i^U - q_i^N| \qquad (10)$$

increases and converges to the value characterising the actual Pareto set. We illustrate both of these concepts on an example (we use here an example defined later by Eq. (22)) – see Figs. 6 and 7.

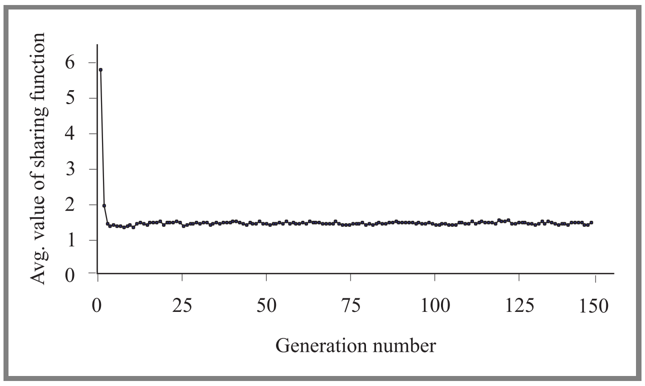

Fig. 6. Average values of sharing function in subsequent generations for Eq. (22).

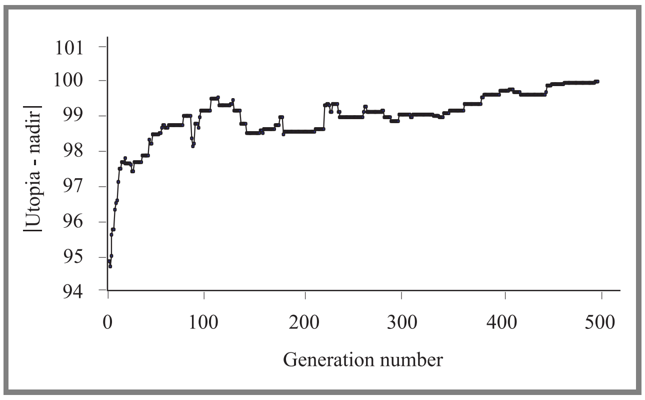

Fig. 7. Utopia-nadir distances in subsequent generations for Eq. (22).

We observe that after a small number of iterations most of the analysed measures oscillate around a constant value and thus are not particularly useful for stopping tests. An exception is the distance of utopia and nadir approximations, which converges to a constant value after a relatively large, but reasonable number of iterations. Thus, *the relative change of the distance of utopia and nadir approximations is the best stopping test for estimating a Pareto set by evolutionary algorithms*.

2.9. Succession methods

Application of niched methods results in decreasing fitness of an individual in densely represented parts of a Pareto set. However, this might lead to concentration on the boundaries of the Pareto set, demonstrated by the following example. Analysing how to choose successors in order to get a uniform representation of a Pareto set, we investigated a simple case: let the Pareto set in three-dimensional space belong to the plane $z = 0$ and be a square $x, y \in < 0; 9 >$. The simplest niched method with the niche diameter of 2 gives the following values of fitness function (100 points arranged in square table) shown in Fig. 8.

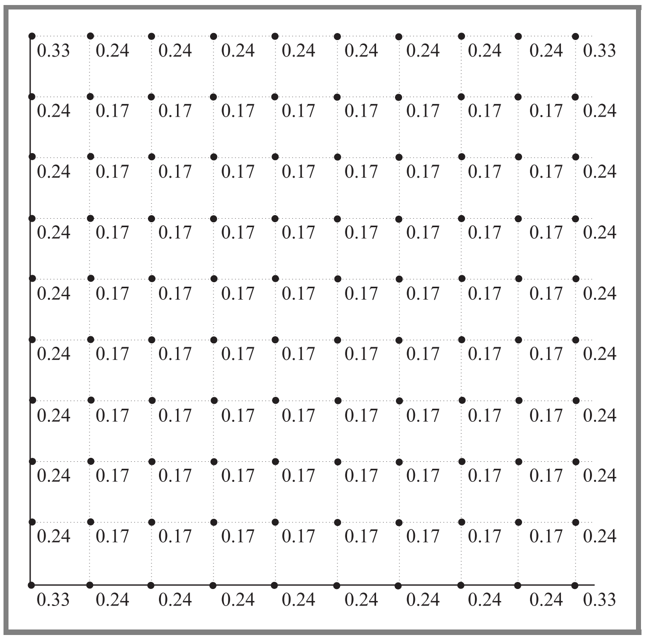

Fig. 8. Values of a fitness function for the simple case considered.

By applying the simplest succession method based on a simple ranking of the individuals to this case, we promote individuals located on the boundary of Pareto set (Fig. 9).

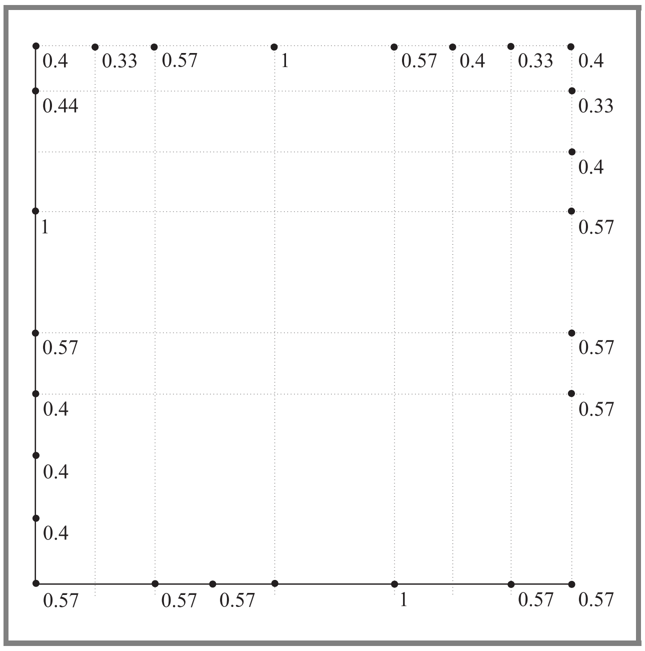

Fig. 9. Successors in the simple case with basic ranking succession rule ($\mu = 0.25$).

Table 1
Comparison of various succession methods

Parameter	Succession method				
	ranking	roulette	tournament	modified fitness	deterministic
	$\mu = 10$				
Computing time [ms]	36.59	4.84	7.40	28.56	1744
Average fitness	0.807	0.807	0.779	0.820	1
	$\mu = 25$				
Computing time [ms]	39.13	8.89	14.30	33.30	1665
Average fitness	0.520	0.579	0.548	0.601	1
	$\mu = 50$				
Computing time [ms]	46.92	19.14	29.70	43.63	1466
Average fitness	0.374	0.367	0.348	0.381	0.469

We can also imagine a deterministic (actually – non-evolutionary) succession rule in which we eliminate in a deterministic loop subsequent individuals, while increasing the fitness of its neighbours. The process is repeated until the population drops to a given number of individuals, as illustrated by Fig. 10.

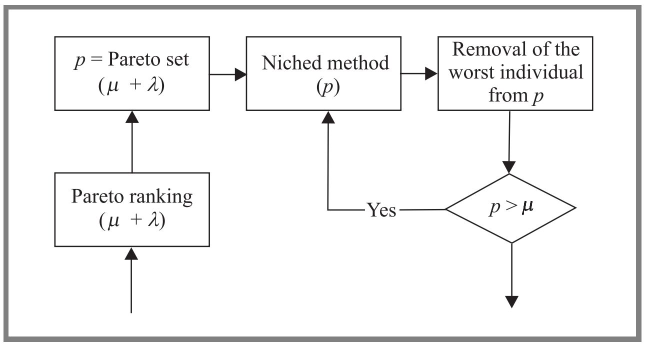
p = Pareto set ($\mu + \lambda$)
Niched method (p)
Removal of the worst individual from p
Pareto ranking ($\mu + \lambda$)
Yes
$p > \mu$

Fig. 10. Block-diagram of a deterministic succession rule.

Another succession rule is obtained by modifying definition of coefficient $m(x)$, needed to determine fitness. Instead of summing it up over all non-dominated individuals as in Eq. (9), it can be summed up only for individuals with lower index numbers on the list:

$$m(x) = \sum_{x=1}^{y-1} sh\big(d(x,y)\big) . \tag{11}$$

That way, the individuals considered first on the list obtain greater fitness indicators (Fig. 11).

Yet another methods of succession for evolutionary vector optimisation can be obtained by modifying roulette and tournament approaches to general evolutionary algorithms. Recall that a roulette approach determines successors (or selects individuals for cross-breeding) randomly, with probability increasing with the fitness indicator. Tournament approach determines successors by selecting randomly k individuals for a tournament and then selecting the tournament winner as the individual with highest fitness indicator (or randomly selects one of them, if there is a tie). Both approaches give similar results in our case (Fig. 12).

The above mentioned methods were compared in terms of their accuracy (defined by uniform coverage of the Pareto set, measured by average value of fitness indicator, that should be highest for a uniform coverage) and computational effort needed to solve this simple case. Table 1 gives results obtained by using a PC with 700 MHz Pentium III processor, after a large number of generations (10 000).

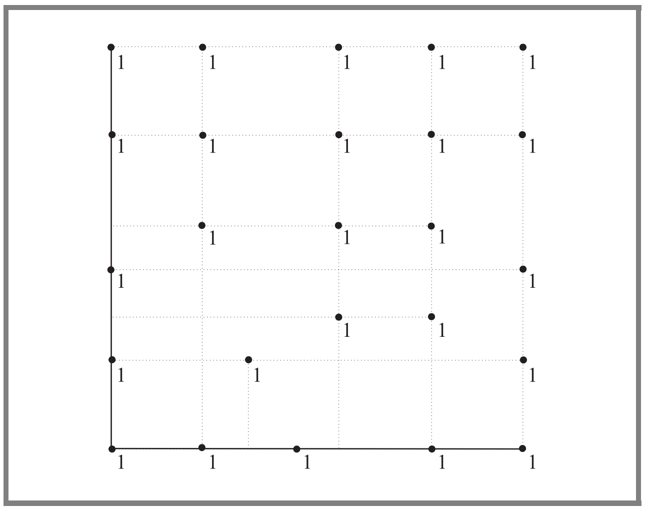

Fig. 11. Successors in the simple case with deterministic succession rule.

The most uniform representation of the Pareto set is obtained by deterministic method, though the required computing time is rather large. Among other methods, simple ranking method gives the least uniform representation – as can be expected since it favours individuals on the edge of Pareto set. For further experiments, either the roulette method (giving shortest computing time) or the determin-

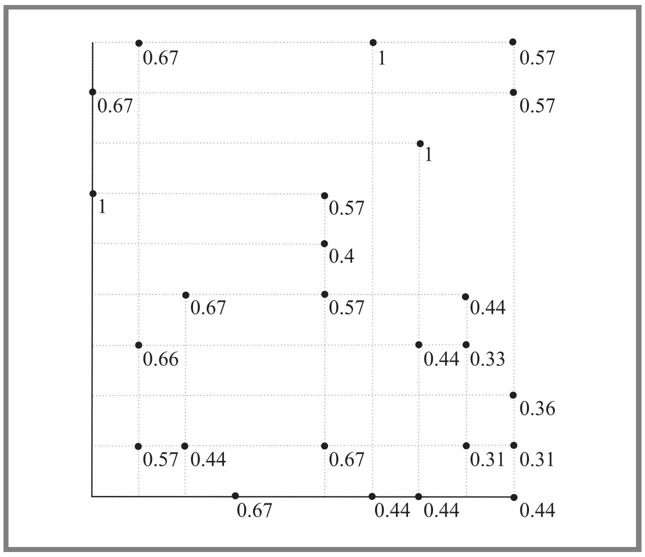

Fig. 12. Successors obtained by a roulette method in the simple case – the tournament method gives similar results.

istic method (ensuring uniform representation), were typically used. We will show later that performance of ranking succession method can be considerably improved if a more sophisticated ranking method is used.

2.10. Accuracy of representing Pareto set

When analysing more complicated Pareto sets than the simplest example presented before, it was observed that evolutionary algorithms do not converge precisely to the actual Pareto set. In a sense, this phenomenon is obvious: due to mutation necessary for evolutionary behaviour, only a few individuals come precisely to the Pareto set; most of them are oscillating just "below" the Pareto set. Even if obvious, this aspect was not sufficiently stressed and analysed in the literature. We give here results of investigating – in some cases for quite a long time with up to 30 000 generations – a simple example with known Pareto set, obtained by linear vector optimisation:

$$\max x_j,\ j = 0,\dots,i,$$
$$\sum_{j=0}^{i} x_j \le 1,$$
$$x_j \ge 0,\ j = 0,\dots,i. \tag{12}$$

We see that for the investigated example with $i = 2$, the average distance form the Pareto set oscillates about $4 \cdot 10^{-3}$ (actually, $3.76 \cdot 10^{-3}$) after only 200 generations (Fig. 13). Naturally, this value depends on the limit values for decreasing the dispersion parameter σ. This is because the oscillation of the distance from the Pareto set results from recombination and (predominantly) the mutation operation. Even if the original population were situated precisely on the Pareto set, mutation would put successors "below" this set, as illustrated by the following simple example (Fig. 14).

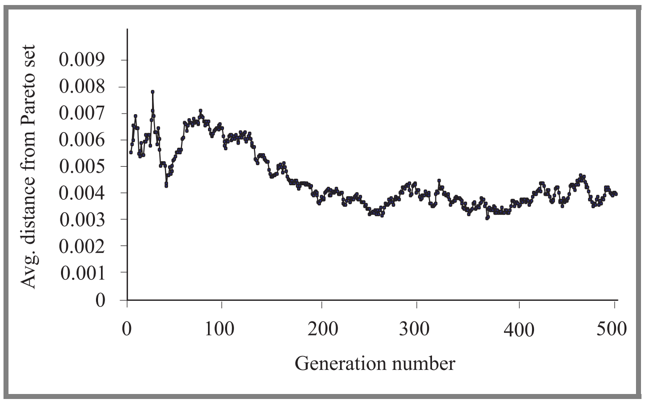

Fig. 13. Average distance from Pareto set for the example defined by Eq. (12) ($i = 2$, $\mu = 100$, $\lambda = 100$, $r = 0.01$).

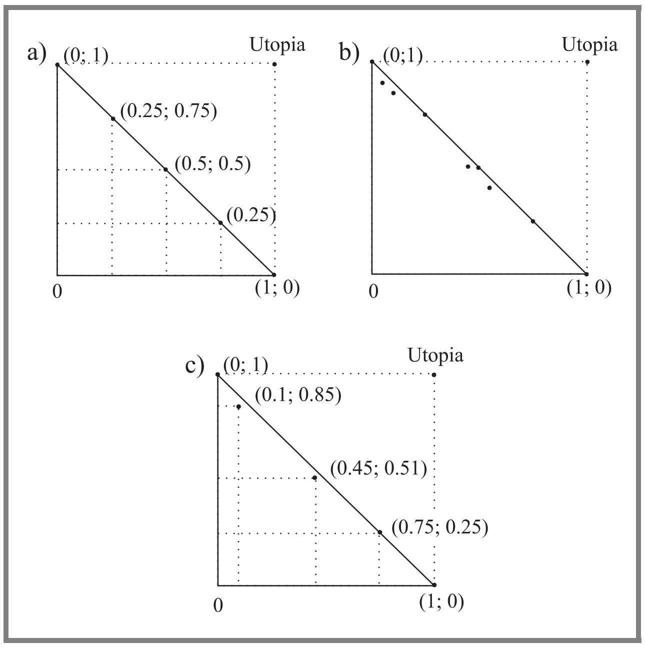

Fig. 14. Population on Pareto set (a), successor population after recombination and mutation (b) the same after succession (c).

We could of course force the algorithm to converge to the precise Pareto set, if we decided to decrease the mutation effect through decreasing dispersion parameter σ to zero. This would result, however, in losing exploratory powers of the evolutionary algorithm, considered a degeneration of the algorithm. Precise dependence of accuracy of approximating Pareto sets on the limit values of dispersion parameters requires further detailed study.

3. Use of reference points and achievement functions in evolutionary algorithms

A powerful and practical way of making vector optimisation algorithms interactive is to combine them with the concepts of reference points and to use order-preserving achievement functions [11]. We will investigate here, how to combine these concepts with evolutionary algorithms in order to either make them more interactive or to eliminate other deficiencies.

3.1. Segments of Pareto sets dominating a reference point

In interactive analysis of Pareto sets, it might be interesting to approximate a part of Pareto set "above" a given reservation point q^{res} – see the example shown in Fig. 15. We have to add constraints:

$$\begin{aligned} f(x) &\geq g_i^{res},\ i=1,\dots,k_1 \text{ (for maximised criteria)},\\ f(x) &\leq g_i^{res},\ i=k_1+1,\dots,k \text{ (for minimised ones)}. \end{aligned} \qquad (13)$$

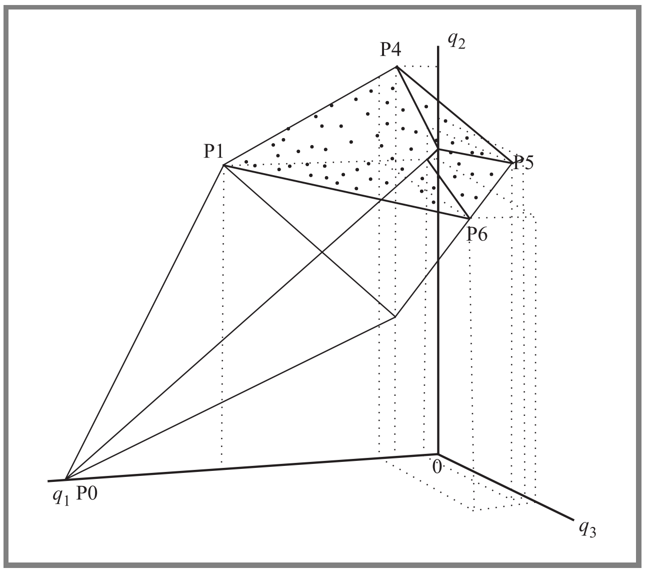

Fig. 15. A part of Pareto set above a given reservation point (0, 24, 0).

Provided that the resulting problem is feasible (the reservation point is not "above" Pareto set), specifying such additional requirement does not complicate the evolutionary algorithm. Additional constraints are simple and can be taken into account as selection conditions. We can also achieve a better approximation accuracy if the reservation point lies close to Pareto set. For the relatively simple examples of Pareto sets considered here, the necessary computational effort does not diminish, however: approximating a part of Pareto set is as expensive as approximating the entire set. On the other hand, the necessary computational effort is reasonable for simple examples. Interactive investigation by approximating first entire Pareto set, and approximating selected parts of it more precisely later is possible.

3.2. Using achievement functions for better ranking and for improving the accuracy of representing Pareto set

Ranking Pareto in evolutionary algorithms can be modified by using an order-consistent achievement function (see also [11]), e.g.:

$$\sigma(q,\overline{q}) = \min_{1\leq i\leq m} \sigma_i(q_i,\overline{q}_i) + \varepsilon \sum_{i=1}^{m} \sigma_i(q_i,\overline{q_i}), \qquad (14)$$

where $\overline{q}$ is a reference point in criteria space. The partial achievement functions can be defined for a simple case as follows:

$$\begin{aligned} \sigma_i(q_i,\overline{q}_i) &= \tfrac{q_i-\overline{q}_i}{q_i^U-q_i^N} \text{ (for maximised criteria)},\\ \sigma_i(q_i,\overline{q}_i) &= \tfrac{\overline{q}_i-q_i}{q_i^N-q_i^U} \text{ (for minimised ones)}, \end{aligned} \qquad (15)$$

where $\overline{q}^U$ and $\overline{q}^N$ are utopia and nadir point vectors or their approximations, respectively. Modification of Pareto ranking is based on the following property of the achievement function:

$$\overline{q} \in Q_0 \Rightarrow \left\{ \begin{array}{c} \max_{q\in Q_0} \sigma(q,\overline{q}) \geq 0 \\ \widehat{q} = \arg\max_{q\in Q_0} \sigma(q,\overline{q}) \geq \overline{q} \end{array} \right\}. \qquad (16)$$

Thus, the value $\sigma(q,\overline{q})$ greater than 0 indicates (approximately), that point q dominates the reference point $\overline{q}$. The value 0 of the achievement functions indicates that point q is either equal or (approximately) equivalent to $\overline{q}$. Because of these properties, the Pareto rank of an individual can be determined by:

$$rank_j^{(t)} = 1 + \sum_{k=1}^{S_j} \sigma(q_k,q_j), \qquad (17)$$

where q_k are individuals dominating q_j, thus $\sigma(q_k,q_J) \geq 0$, and S_j is the number of individuals dominating q_j. This way of ranking takes into account both distance of a given point from Pareto frontier and number of points dominating given point. The disadvantage is that estimation of utopia and nadir points must be available to construct the achievement function, hence this ranking method cannot be used when approximating Pareto set for the first time. It is applicable only to further, interactive analysis of selected parts of Pareto set.

Despite such drawback, the ranking method based on achievement function values has several advantages. It is more sensitive than the classical Golberg ranking method and the Fonseca and Fleming method, which can be illustrated by the simple example (Fig. 16).

Another, more practical advantage of Pareto ranking using achievement function values is that it might improve the accuracy of the entire evolutionary algorithm. We have seen

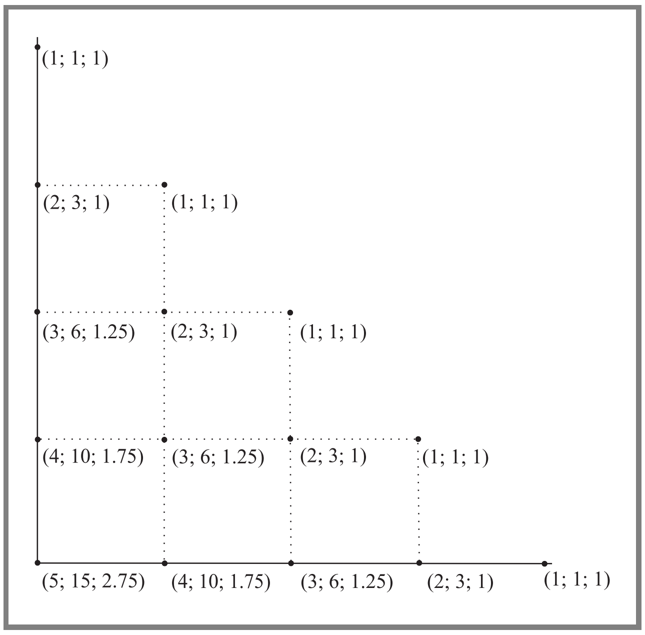

Fig. 16. Example of ranking values obtained by (1st value) Goldberg method; (2nd value) Fonseca and Flemming method; (3th value) by using achievement function values.

before that ranking methods did not behave well as succession mechanisms. A ranking method using achievement function values can perform much better: we can increase the accuracy of the entire evolutionary algorithm by increasing the value of the parameter ε, as suggested by the computation results shown in Table 2.

Table 2
Average distance from Pareto set after 30 000 generations depending on the parameter ε ($i = 2$, $\mu = 100$, $r = 0.01$)

ε	0	0.01	0.1	1	10	100
Average distance from Pareto set $[\cdot 10^{-3}]$	3.82	3.88	3.88	1.18	0.19	0.00012

We see that, using evolutionary algorithm interactively for more precise investigation of a part of Pareto set, we could actually obtain much better accuracy or use much shorter computation times for a ranking method based on achievement function values. On the other hand, very large values of ε (say, changing it from 10 to 100) mean only increasing the absolute value of achievement function, not its character that is dominated then by its linear part. This suggests that similar results would be obtained when using a slightly different form of the ranking formula:

$$rank_j^{(t)} = 1 + \beta \sum_{k=1}^{S_J} \sigma(q_k, q_j)\,, \qquad (18)$$

while increasing the parameter β over its initial value 1.

Thus, *use of ranking values based on achievement functions not only increases flexibility of ranking, but also results in much better accuracy of approximating Pareto set.*

3.3. Neutral compromise points and their neighbourhoods

Given a reservation point q_{res} and an aspiration point q_{asp} in criteria value space, we can define a *relative neutral compromise point* as a point in Pareto set in criteria space being closest to the line joining points q_{res} and q_{asp} (Fig. 17).

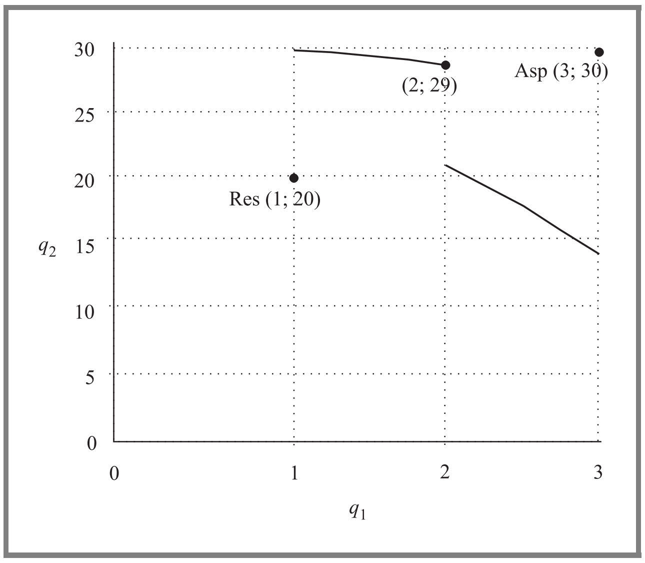

Fig. 17. Example of Pareto set with a reservation, aspiration and a relative neutral compromise points shown.

This point can be obtained by optimising the achievement function $\sigma(q, \overline{q})$ of the form (14) with partial achievement functions defined e.g. as follows:

$$\sigma_i(q_i, \overline{q}_i) = \frac{q_i - q_{asp.i}}{q_{asp.i} - q_{res.i}} \ \text{(for maximised criteria)},$$
$$\sigma_i(q_i, \overline{q}_i) = \frac{q_{asp.i} - q_i}{q_{res.i} - q_{asp.i}} \ \text{(for minimised ones)}. \qquad (19)$$

For more sophisticated forms of partial achievement functions see e.g. [11]. In evolutionary algorithms, we can use the achievement function $\sigma(q, \overline{q})$ as a fitness measure and thus optimise it.

This results in interactive modification of evolutionary algorithms for vector optimisation: the user defines the aspiration and reservation points, the algorithms responds with the relative neutral compromise point or its approximation by a population of points (Table 3). This idea is illustrated by the following example. For the vector optimisation problem defined by Eq. (5), we define reservation and aspiration points as in Fig. 17. The line joining points q_{res} and q_{asp}

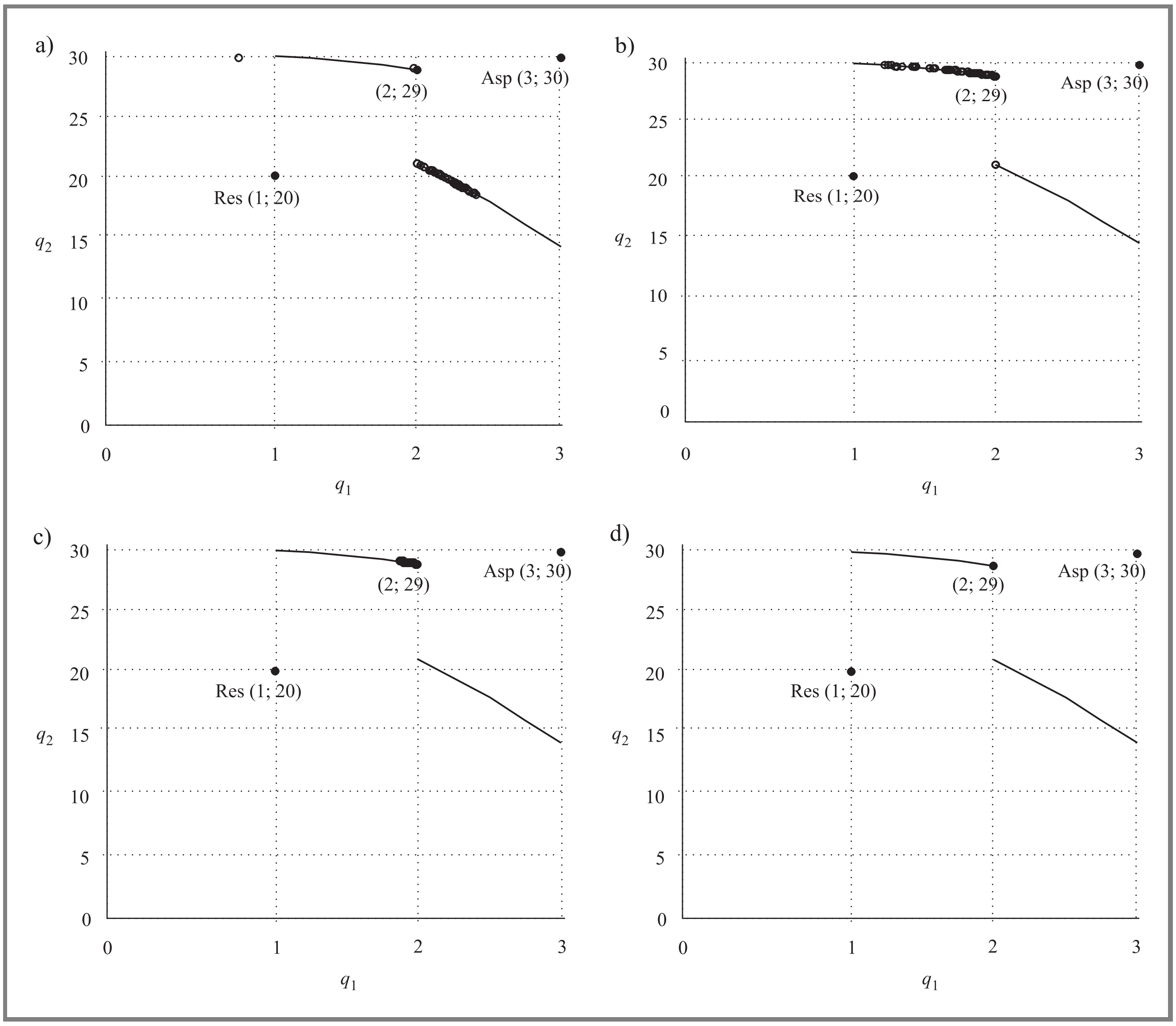

Fig. 18. The approximation of the relative neutral compromise point for the example from Fig. 17. Approximation cloud depending on generation number (a) $n = 5$; (b) $n = 10$; (c) $n = 15$; (d) $n = 30$ ($\mu = 50$, $\lambda = 50$).

Table 3
Diameter of approximation cloud depending on generation number $(\mu = 50, \lambda = 50)$

Generation number	5	10	20	40	60
$\Delta_{q_1}[\cdot 10^{-3}]$	1020	850	45.95	1.82	0.07
$\Delta_{q_2}[\cdot 10^{-3}]$	10640	9230	91.35	3.63	0.16

does not intersect Pareto set, but this makes the example more interesting. An evolutionary algorithm with achievement function used as a fitness measure produces a population approximating the relative neutral compromise point (2, 29) in the criteria space at first, and soon converges to this point (Fig. 18).

3.4. Parameterisation of representing Pareto set or its segment

The approach discussed above can be further parameterised combining a niched method with ranking based on achievement function. The niched method was originally used to provide a uniform representation of Pareto set in a global approach; here we use it to parameterise a local approach. Size of the niche can be related to e.g. the distance between aspiration and reservation points. Use of the niched method results in broadening the dispersion of a population around a neutral compromise point, as illustrated in Fig. 19.

We conclude that the evolutionary algorithms of vector optimisation, though traditionally understood as global and having non-interactive, batch character, can nevertheless be localised and used as local tools of interactive multiple criteria analysis.

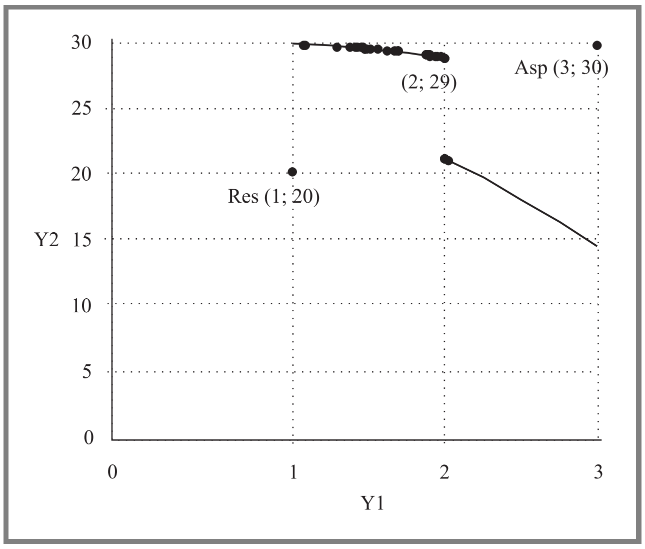

Fig. 19. Dispersion around the relative neutral compromise point for the example from Fig. 17, resulting from a niched approach with niche size equal to 10% of the range between aspiration and reservation points (population size 50, 200 generations).

4. Estimation of utopia and nadir points in evolutionary algorithms

4.1. Definitions and classical computations of utopia and nadir points

We recall that the *utopia point* q^U is defined as the "lowest" point dominating entire outcome set Q_0 (and thus entire Pareto set $\widehat{Q}_0$) in criteria value space. In other words, if some criteria are maximised and other minimised, we define:

$$\begin{aligned} q_i^U &= \max_{x \in X_0} f_i(x),\ i = 1, \ldots, k_1 \\ &\quad (\text{for maximised criteria}), \\ q_i^U &= \min_{x \in X_0} f_i(x),\ i = k_1 + 1, \ldots, k \\ &\quad (\text{for minimised criteria}), \end{aligned} \tag{20}$$

where $f_i(x)$ are criteria functions, k is the number of them (while k_1 is the number of maximised criteria), X_0 is the set of admissible decisions and $Q_0 = f(X_0)$ is the outcome set of attainable criteria vectors.

The *nadir point* is defined as the "highest" point in criteria value space dominated by the entire Pareto set $\widehat{Q}_0$ – and not necessarily the entire outcome set Q_0. This difference explains the difficulty (see e.g. [9]) of precisely calculating the nadir point, since we must perform necessary computations not over entire X_0 or Q_0, but over their efficient subsets $\widehat{X}_0$ or $\widehat{Q}_0$. Thus, if some criteria are maximised and other minimised, we define:

$$\begin{aligned} q_i^N &= \min_{x \in \widehat{X}_0} f_i(x) = \min_{q \in \widehat{Q}_0} q_i,\ i = 1, \ldots, k_1 \\ &\quad (\text{for maximised criteria}), \\ q_i^N &= \max_{x \in \widehat{X}_0} f_i(x) = \max_{q \in \widehat{Q}_0} q_i,\ i = k_1 + 1, \ldots, k \\ &\quad (\text{for minimised criteria}). \end{aligned} \tag{21}$$

We cannot replace $\widehat{Q}_0$ with Q_0 in the equation above, because this might lead to nadir estimation much lower than actual values. On the other hand, computation of precise value of the nadir point is very difficult when using classical methods. There are many methods that approximate nadir point components; the simplest of them is based on using only results of computations related to determining utopia components as in (20) and selecting the worst criteria values encountered during these computations:

$$\begin{aligned} q_i^U &= \max_{x \in X_0} f_i(x),\ \widehat{q}_i = \arg\max f_i(x),\ i = 1, \ldots, k_1 \\ &\quad (\text{for maximised criteria}), \\ q_i^U &= \min_{x \in X_0} f_i(x),\ \widehat{q}_i = \arg\min f_i(x),\ i = k_1 + 1, \ldots, k \\ &\quad (\text{for minimised criteria}), \\ q_i^N &= \min_{1 \le j \le k} \widehat{q}_i^j,\ i = 1, \ldots, k_1 (\text{for maximised criteria}), \\ q_i^N &= \max_{1 \le j \le k} \widehat{q}_i^j,\ i = k_1 + 1, \ldots, k\ (\text{for minimised criteria}), \end{aligned} \tag{22}$$

where q_j denotes the jth component of vector q. This method is accurate if $k = 2$, for bi-criteria problems. However, in other cases it usually gives too optimistic estimations of the nadir value.

Matthias Ehrgott and Dagmar Tenfelde-Podehl [9] have proposed an algorithm computing the nadir point for three (or more) criteria by determining the Pareto sets for (each possible pair of) two criteria. For these bi-criteria Pareto sets, the values of the third missing criterion are attached, the resultant three-dimensional vectors are collected in one set, dominated results deleted, and the nadir values are directly computed from the resulting approximation of Pareto set.

4.2. Evolutionary algorithms and utopia and nadir points

Although the literature on evolutionary and genetic algorithms for vector optimisation is rather rich, it is focused more on the algorithms details than on their use for analysing Pareto set. Thus, an obvious fact was practically overlooked: since we approximate entire Pareto set by an evolutionary algorithm, the computations of utopia and nadir points should be much more easy than when using classical vector optimisation algorithms and should be actually by-products of the evolutionary algorithm applied. The questions that should be investigated are "only" how to provide for necessary accuracy of estimating these points –

especially the nadir point – while limiting the computational effort necessary for this estimation. We shall show on an example that these questions are by no means trivial.
We consider a slightly modified example from [3]:

$$\begin{aligned} \text{maximise}: f_1(x) &= 100 - 7x_1 - 20x_2 - 9x_3 \\ \text{maximise}: f_2(x) &= 4x_1 + 5x_2 + 3x_3 \\ \text{maximise}: f_3(x) &= x_3 \\ 1\tfrac{1}{2}x_1 + x_2 + 1\tfrac{3}{5}x_3 &\le 9 \\ x_1 + 2x_2 + x_3 &\le 10 \\ x_i \ge 0,\ i &= 1,2,3 . \end{aligned} \tag{23}$$

The set of admissible decisions X_0 is illustrated by Fig. 20.

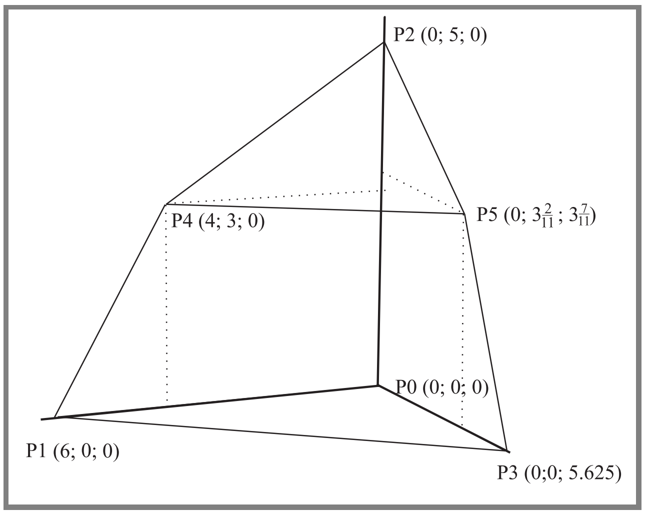

Fig. 20. Set of admissible decisions X_0 for the example defined by Eq. (23).

The set of admissible decisions X_0 is determined by its corner points:

$$\Big\{ \mathrm{P0} = (0;0;0),\ \mathrm{P1} = (6;0;0),\\ \mathrm{P2} = (0;5;0),\ \mathrm{P3} = \big(0;0;5\tfrac{5}{8}\big),\\ \mathrm{P4} = (4;3;0),\ \mathrm{P5} = \big(0;3\tfrac{2}{11};3\tfrac{7}{11}\big) \Big\} .$$

Following the transformation $q = f(x)$ determined by Eq. (23), we can define also the corresponding corner points of the set of attainable criteria values Q_0 (Fig. 21). By direct examination, we can eliminate some of them as not belonging to Pareto set.
We can show in this way that the Pareto set is composed of surfaces determined by the following points in criteria space:

$$\Big\{ \mathrm{P0} = (100;0;0),\ \mathrm{P1} = (58;24;0),\\ \mathrm{P3} = \big(49\tfrac{3}{8};16\tfrac{7}{8};5\tfrac{5}{8}\big) \Big\} \quad \text{and}$$

$$\Big\{ \mathrm{P1} = (58;24;0),\ \mathrm{P3} = \big(49\tfrac{3}{8};16\tfrac{7}{8};5\tfrac{5}{8}\big),\\ \mathrm{P4} = (12;31;0),\ \mathrm{P5} = \big(3\tfrac{7}{11};26\tfrac{9}{11};3\tfrac{7}{11}\big) \Big\} .$$

By direct examination, we can find for these points the utopia point $q^U = \big(100;31;5\frac{5}{8}\big)$ and the nadir point $q^N = \big(3\frac{7}{11};0;0\big)$. Now we shall show the results of computing these points via three variants of evolutionary algorithms.

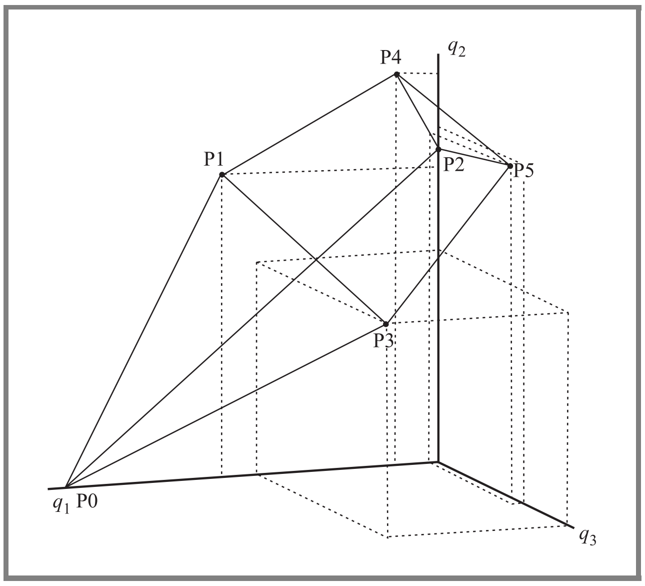

Fig. 21. The set of attainable criteria values Q_0 for the example defined by Eq. (23).

I. Evolutionary computations of utopia point with utopia based nadir approximations

The first variant uses direct determination – see Eq. (19) – of utopia point for an evolutionary approximation of a Pareto set and an indirect – see Eq. (21) approximation of the nadir point based on the data obtained in utopia point determination. An evolutionary algorithm with $(\mu,\lambda) = (200,100)$ and 200 generations gave the following results:

$$q_1^U = (100;\,0;\,0)$$

$$q_2^U = (11.9998;\,30.9999;\,0) \Rightarrow q^U = (100;\,30.9999;\,5.625)$$

$$q_3^U = (49.375;\,16.875;\,5.625)$$

with the corresponding quite inaccurate nadir approximation $q^N = (11.9998;\,0;\,0)$. By increasing the computing effort (measured below as the number of new computations of criteria values, because this, rather than organisation of the algorithm determines the computational effort) we can increase the accuracy of utopia approximations, but accuracy of nadir approximations remains inadequate, as shown in Table 4. Thus, we conclude that this method of nadir approximations is not worth using with evolutionary algorithms.

Table 4
Results of utopia and nadir point approximation by method I

The number of new computations of criteria values	Nadir point approximation	Utopia point approximation
30 000	(12.06; 0; 0)	(100.0; 30.79; 5.617)
60 000	(11.97; 0; 0)	(100.0; 30.95; 5.624)
120 000	(12.00; 0; 0)	(100.0; 31.00; 5.625)

II. Evolutionary computations of utopia point and nadir point

Since an evolutionary algorithm approximates entire Pareto set, we can also simply determine utopia and nadir points directly, according to their definitions, for the subsequent evolutionary approximations of Pareto set (Fig. 22). This simple method needs not, however, be the best, since a uniform approximation of Pareto set does not necessarily cover well the remote corners of this set, which are responsible for utopia and nadir points.

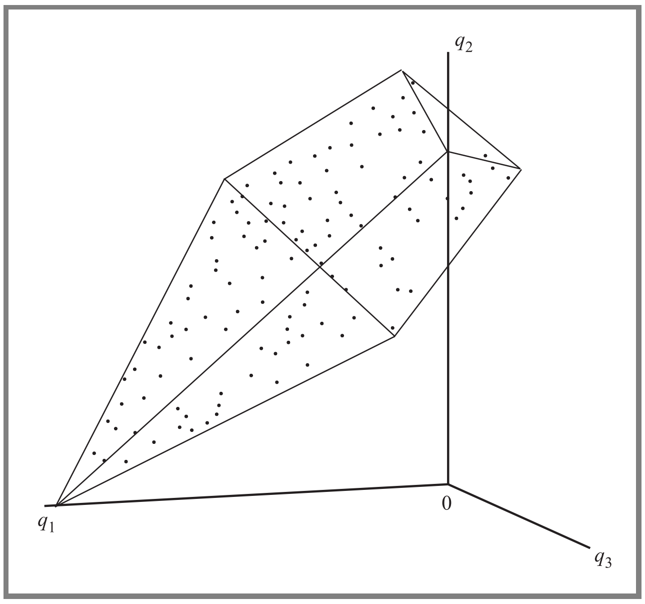

Fig. 22. Approximation of the Pareto set for the example defined by Eq. (23).

Thus, an evolutionary algorithm for vector optimisation must be modified in order to provide for a good approximation of utopia point and particularly the nadir point. It is necessary to increase fitness indicators for individuals with extreme values of criteria components.

Theoretically, such a method should give good approximations of Pareto set together with its utopia and nadir points. However, practical applications show that good approximations of the nadir point remain difficult to obtain. This is illustrated by results (Table 5) of an evolutionary algorithm with direct determination of nadir point for Pareto set approximations in subsequent iterations, with a modification of fitness indicators for individuals with extreme values of criteria vectors components. We observe that accuracy of the nadir point approximation, although much better than in method I, still remains inadequate even after very long computations.

Table 5
Results of nadir point approximation by method II

The number of new computations of criteria values	Nadir point approximation
Arbitrary starting population	
30 000	(6.78; 0; 0)
60 000	(5.90; 0; 0)
120 000	(5.91; 0; 0)
Starting population containing individuals responsible for utopia point	
30 000	(5.38; 0; 0)
60 000	(5.24; 0; 0)
120 000	(5.06; 0; 0)

III. Evolutionary approximations of Pareto sets for smaller number of criteria

The method proposed by Ehrgott and Tenfelde-Podehl [9] was not developed as an evolutionary algorithm, but can be easily combined with evolutionary approaches, involving the following steps:

– for each pair of criteria, Pareto sets are be approximated by using an evolutionary algorithm;

– for each individual in these approximations, the corresponding values of other criteria are computed;

– results obtained this way are combined and dominated points deleted, resulting in an approximation of Pareto set for the original problem;

– utopia and nadir points are computed according to their definitions for this approximation of Pareto set.

The advantage of this method over method II is that approximation of Pareto sets for bi-criteria problems in a natural way provides for more attention paid to extreme values of criteria components.

We illustrate the working of this method by showing the results of such approximations obtained by using an evo-

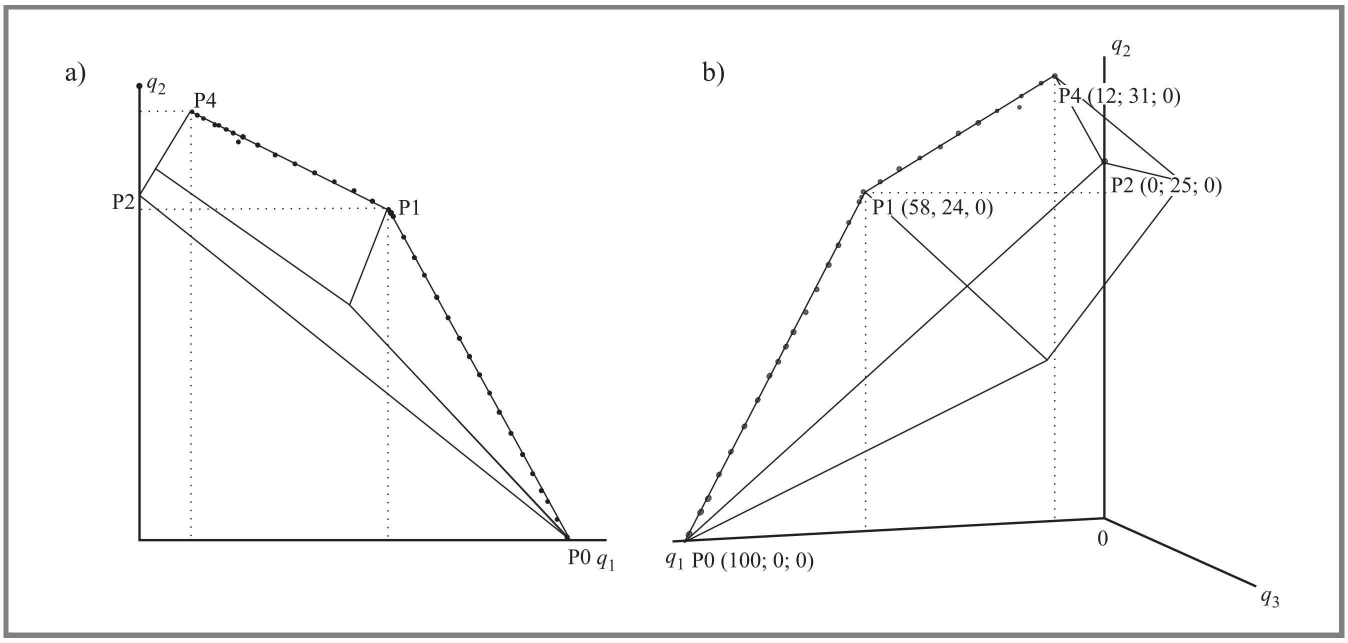

Fig. 23. Approximation of Pareto set for criteria 1 and 2 (a) with three-dimensional presentation (b).

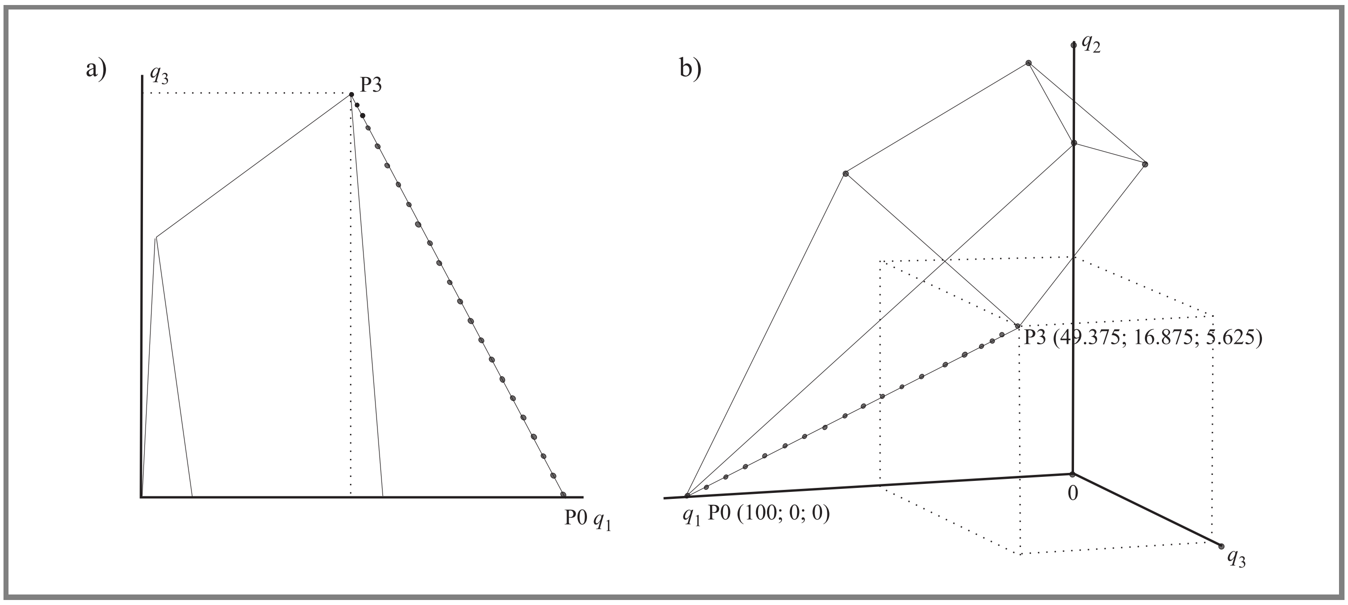

Fig. 24. Approximation of Pareto set for criteria 1 and 3 (a) with three-dimensional presentation (b).

lutionary algorithm with $(\mu,\lambda) = (200,100)$, 200 generations and alternative niche diameters 4,75; 1,55; 0,28) – see Figs. 23–25.

This way, after a large number (360 000) of computations of new criteria vectors, the following approximations were obtained: utopia point $q^U = (100; 30.999; 5.625)$ and nadir point $q^N = (4.36; 0; 0)$. We see that nadir point approximation, though much better than in other methods, still remains inadequate. Moreover, method III requires more computations (three times in this case) than methods II and I, and a fair way of comparing them is to compare nadir approximations after the same number of computations of new criteria vectors. Such a comparison is presented in Table 6.

When we compare the results of these three methods, we see that method III is most promising. The example defined by Eq. (23) might be especially difficult for nadir point approximation, hence we tried another variant

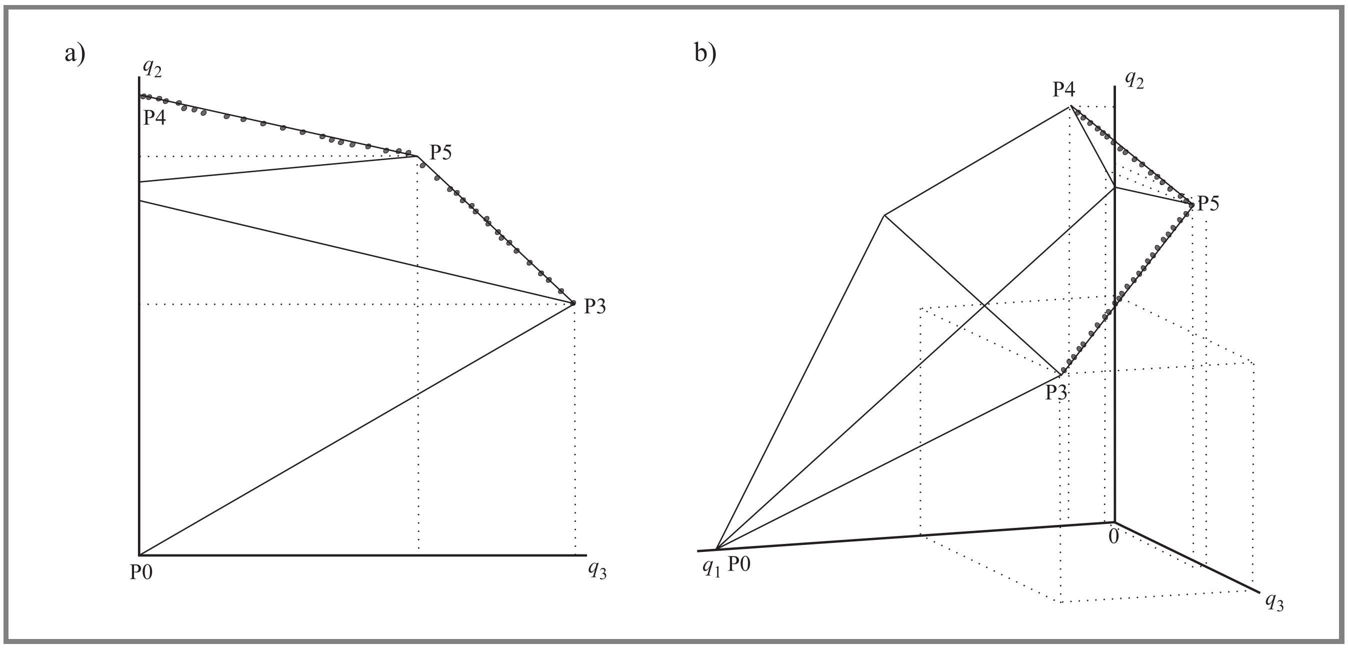

Fig. 25. Approximation of Pareto set for criteria 2 and 3 (a) with three-dimensional presentation (b).

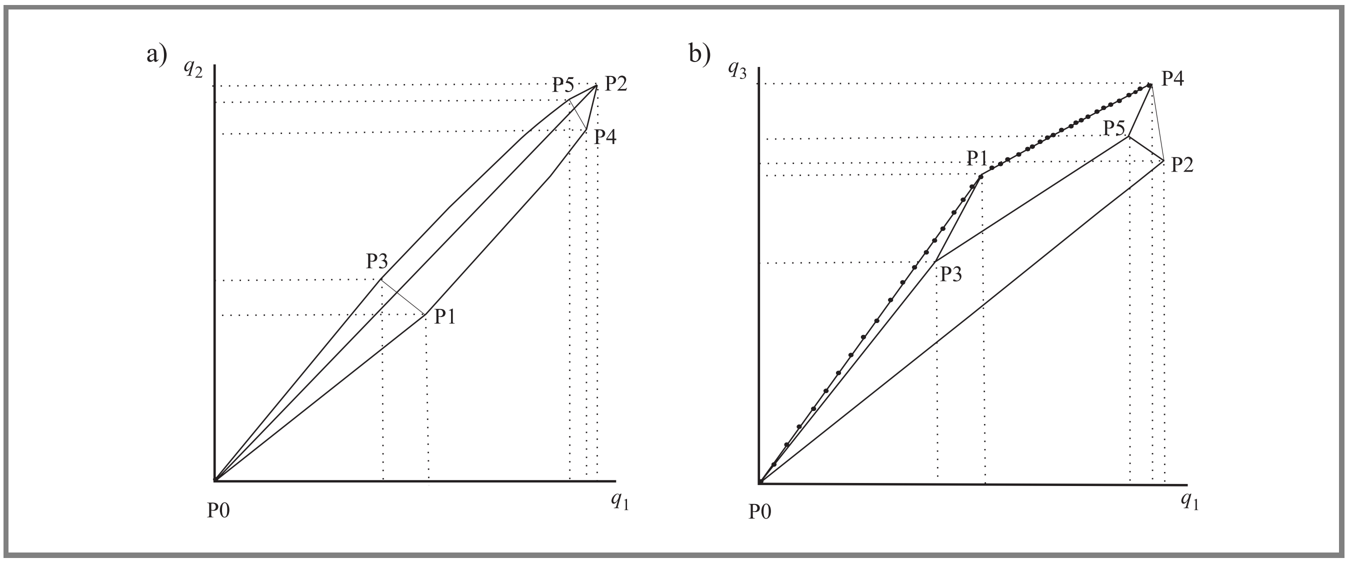

Fig. 26. Pareto sets for criteria 1 and 2 (a) or for criteria 1 and 3 (b), for example defined by Eq. (24).

Table 6
Results of nadir point approximation by method III

Number of new computations of criteria vectors	Nadir point approximation
30 000	(5.01; 0; 0)
60 000	(4.67; 0; 0)
120 000	(4.78; 0; 0)

of this example, at the same time testing the possibility of generalising method III for a larger number of criteria.

The original example from [3] is as follows:

$$\begin{aligned}
\text{minimise}: f_1(x) &= 9x_1 + 19\tfrac{1}{2}x_2 + 7\tfrac{1}{2}x_3 \\
\text{minimise}: f_2(x) &= 7x_1 + 20x_2 + 9x_3 \\
\text{maximise}: f_3(x) &= 4x_1 + 5x_2 + 3x_3 \\
\text{maximise}: f_4(x) &= x_3 \\
1\tfrac{1}{2}x_1 + x_2 + 1\tfrac{3}{5}x_3 &\le 9 \\
x_1 + 2x_2 + x_3 &\le 10 \\
x_i \ge 0, i &= 1,2,3\,. \qquad (24)
\end{aligned}$$

The set of admissible decisions X_0 is the same as in the example defined by Eq. (23) – see Fig. 20. However,

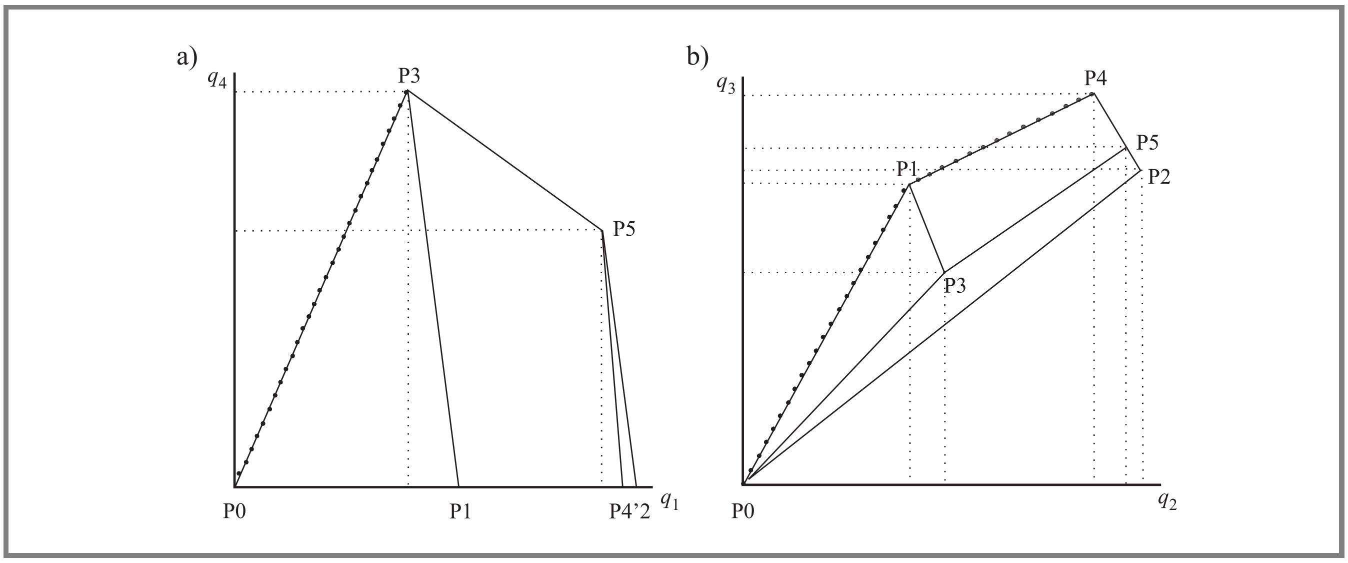

Fig. 27. Pareto sets for criteria 1 and 4 (a) or for criteria 2 and 3 (b), for example defined by Eq. (24).

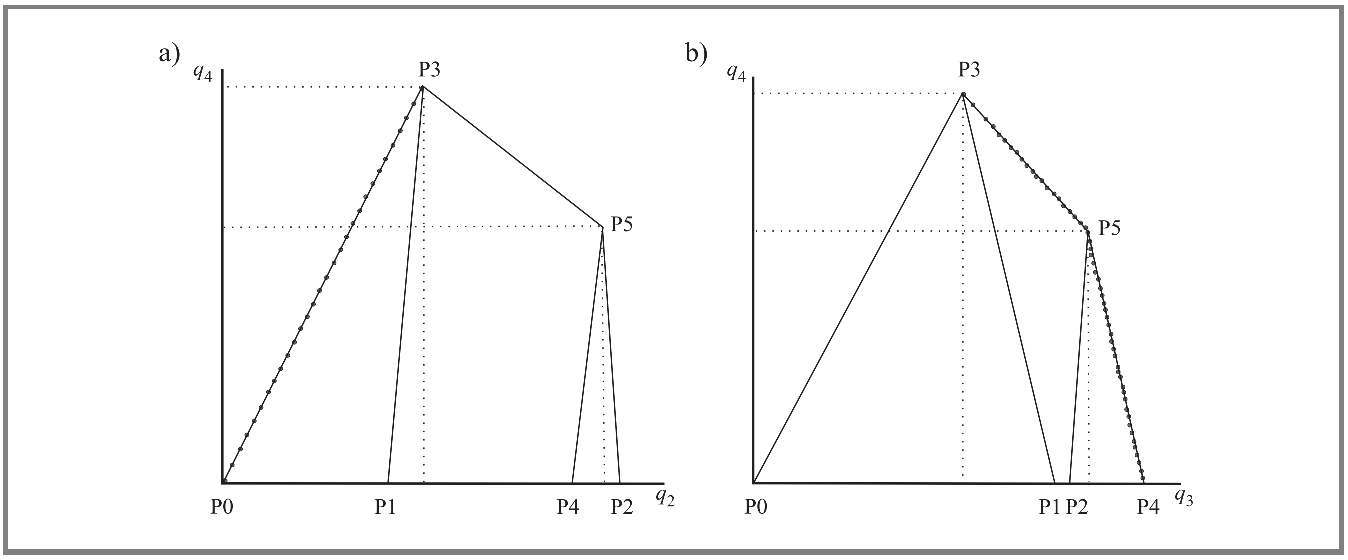

Fig. 28. Pareto sets for criteria 2 and 4 (a) or for criteria 3 and 4 (b), for example defined by Eq. (24).

utopia and particularly nadir points change with each change of criteria and they are here $q^U = \left(0; 0; 31; 5\frac{5}{8}\right)$ and $q^N = \left(94\frac{1}{2}; 96\frac{4}{11}; 0; 0\right)$. The Pareto sets for consecutive bi-criteria problems are shown in Figs. 26–28.

Utopia and nadir points obtained using evolutionary algorithm with $(\mu, \lambda) = (200, 100)$ and 200 generations, and a version of method III for four criteria: $q^U = (0; 0; 30.999; 5.625)$ and $q^N = (94.4998; 95.8747; 0; 0)$. Although the actual number of criteria value computations here increased 6 times (this is the drawback of using method III), we have obtained quite acceptable approximation of utopia and nadir points in this example.

5. Conclusions and future research

We shall point out only few conclusions, in particular those concerning future research:

- Although there is a very rich literature on evolutionary algorithms for vector optimisation, this literature focuses mostly on the tool – specific aspects of evolutionary algorithms, much less on the task – specific issues of vector optimisation, for which an evolutionary approach might be helpful.

- When concentrating on the task, evolutionary algorithms might be usefully extended – e.g. to obtain more precise approximations of selected parts of Pareto set, or better approximations of utopia and nadir points of Pareto set.

- In such extensions of evolutionary algorithms, an essential issue is to make them more interactive (e.g. first approximating entire Pareto set, then a selected part of it). For interactive extensions of evolutionary algorithms, combining them with reference point approaches and achievement function concepts might be useful.
- A particularly difficult issue (not only for evolutionary algorithms, but also in entire vector optimisation) is the determination of nadir points. Classical evolutionary approaches are not sufficient to solve this issue. Combinations of evolutionary algorithms with other approaches of vector optimisation are necessary.
- Many issues outlined in this paper should be treated as starting points only and require deeper future research. Starting from a different perspective, concentrating more on tasks than on tools, the paper serves only as identification of future research issues.

References

[1] K. Deb, "Non-linear goal programming using multi-objective genetic algorithms". Tech. Rep. no. CI-60/98, Department of Computer Science, University of Dortmund, 1998.

[2] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, Mass.: Addison-Wesley, 1989.

[3] C. M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization", *Evol. Comput.*, vol. 3, no. 1, 1995.

[4] C. M. Fonseca and P. J Fleming, "Genetic algorithms for multi-objective optimization: formulation, discussion and generalization" in *Proc. Fifth Int. Conf. Genet. Algor.*, S. Forrest, Ed., San Mateo, USA, 1993, University of Illinois at Urbana – Champaign, Morgan Kauffman, pp. 416–423.

[5] J. Horn, *Multicriterion Decision Making*. Handbook of Evolutionary Computation, IOP & Oxford University Press, 1997.

[6] J. Horn and N. Nafpliotis, "Multiobjective optimization using the niched Pareto genetic algorithm" in *Proc. First IEEE Conf. Evol. Comput., IEEE World Congr. Comput. Intell.*, vol. 1, 1994.

[7] P. Korhonen, "Multiple objective programming support". IR-98-010, International Institute for Applied Systems Analysis, Laxenburg, 1998.

[8] Z. Kowalczuk, T. Białaszewski, and P. Suchomski, "Genetic poliop-timisation in Pareto sense with ranking and niched methods" in *Proc. III Nat. Conf. Evol. Algor. Glob. Optim.*, Warsaw, Poland, 1999.

[9] M. Ehrgott and D. Tenfelde-Podehl, "Nadir values: computation and use in compromise programming". Universitat Kaiserslautern Fachbereich Mathematik, 2000.

[10] D. A. Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithm research: a history and analysis". Tech. Rep. TR-98-03, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1998.

[11] A. P. Wierzbicki, M. Makowski, and J. Wessels, *Model-Based Decision Support Methodology with Environmental Applications*. Dordrecht – Laxenburg: Kluwer – IIASA, 2000.

[12] E. Zitzle, "Evolutionary algorithms for multiobjective optimization: methods and applications". Swiss Federal Institute of Technology (ETH), Zurich, 1999.

Marcin Szczepański was born in August 10, 1976. He has been graduated as Master of Engineering at Warsaw University of Technology at the Institute of Control and Computation Engineering at Optimization and Decision Support Division, in 2001. During his studies, he was especially interested in evolutionary algorithms and decision support.

e-mail: Marcin.Szczepanski@damovo.com
Damovo Poland
Jana Olbrachta st 94
01-102 Warsaw, Poland

Andrzej Piotr Wierzbicki born June 29, 1937 in Warsaw. Graduated as Master of Engineering at the Faculty of Electronics, Warsaw University of Technology (WUT), in 1960. Ph.D. degree at this University in 1964, for a thesis on nonlinear feedback systems; D.Sc. degree in 1968, for a thesis on optimisation of dynamic systems. In 1971–75 a Deputy Director of the Institute of Automatic Control, later a Deputy Dean of the Faculty of Electronics, WUT. In 1975–78 the Dean of the Faculty of Electronics WUT. Since 1978 worked with the International Institute for Applied Systems Analysis in Laxenburg n. Vienna, Austria; 1979–84 as the chairman of the theoretical branch, Systems and Decision Sciences Program, of this Institute. From 1985 back in the Institute of Automatic Control, WUT, as a Professor of optimisation and decision theory. In 1986–91 scientific secretary, currently member of presidium of the Committee of Future Studies "Poland 2000" (in 1990 renamed "Poland in XXI Century") of P.Ac.Sc. In 1991 elected a member of the State Committee for Scientific Research of Republic of Poland and the chairman of its Commission of Applied Research; contributed to basic reforms of Polish scientific system in 1991–94. Deputy chairman of the Council of Polish Foundation for Science in 1991–94, chairman of scientific councils of NASK (National Scientific and Academic Computer Network in Poland) and PIAP (the Industrial Institute of Measurements and Control). In 1991–96 the editor in chief of the quarterly "Archives of Control Sciences" of P.Ac.SC. In 1992 received (as first European researcher) the George Cantor Award of the International Society of Multiple Criteria Decision Making for his contributions to the theory of multiple criteria optimisation and decision support. Since 1996 the General Director of the National Institute of Telecommunications in Poland. In 2000 nominated as a member of the ISTAG (Information Society Technology Advisory Group) at European Commission. Since 2001 chairman of Advisory Group on Scientific International Cooparation of the State Committee for Scientific Research of Poland. Beside lecturing for over 40 years and promoting more than 80 master's theses at WUT (Warsaw University of Technology), he also lectured at the Department of Mathematics, Information Science and Mechanical Engineering of Warsaw University and in doctoral studies: at WUT, the Academy of Mining and Metallurgy, at the University of Minnesota, at the Illinois Technical University, Hagen University, and at the University of Kyoto. He also promoted 18 completed doctoral dissertations. Author of over 180 publications, including 11 books (4 monographs, 7 – editorship or co-authorship of international joint publications, over 50 articles in scientific journals (over 30 in international), 80 papers at conferences (68 at international, including over 48 published as chapters in books). He also authored 3 patents granted and applied industrially. Current interests include parallelisation of optimisation algorithms using multiple criteria approaches, diverse aspects of negotiation and decision support, including e.g. applications of fuzzy set theory for describing uncertainty in decision support models, multimedia software in computer networks, telematics in education, diverse issues of information society and civilisation. Languages: English, German, Russian (each fluent, beside native Polish). Member of IEEE, ISMCDM (International Society of Multiple Criteria Decision Making), SEP (Polish Society of Electrical Engineers), PTM (Polish Mathematical Society), PSKR (Polish Association for the Club of Rome).

e-mail: A.Wierzbicki@itl.waw.pl
National Institute of Telecommunications
Szachowa st 1
04-894 Warsaw, Poland