Computational Materials Science 45 (2009) 77–83

Contents lists available at ScienceDirect

Computational Materials Science

journal homepage: www.elsevier.com/locate/commatsci

Properties of a genetic algorithm equipped with a dynamic penalty function

W. Paszkowicz

Institute of Physics, Polish Academy of Sciences, al. Lotników 32/46, 02-668 Warsaw, Poland

ARTICLE INFO

Article history:
Received 22 December 2007
Received in revised form 31 March 2008
Accepted 2 April 2008
Available online 2 September 2008

PACS:
61.05.cp
02.60.Pn

Keywords:
Genetic algorithm
Penalty function
Optimisation
Powder pattern
Indexing
Multiple extrema
Convergence

ABSTRACT

A genetic algorithm aiming for finding the global minimum and multiple deep local minima of a function exhibiting a complex landscape is studied. A feedback dynamic penalty function is used as a means to direct the algorithm to look for new local minima. The penalty is applied in close vicinity of all minima found before the current search stage. The last one, where the population tends to be trapped in, is treated smoothly. The penalty becomes progressively active there causing that the population progressively transfers outside the trapping area. The method ascertains that, unlike in more classical approaches, after finding the global minimum and a number of local ones on the way, the algorithm continues the exploration and identifies new local minima. Performance tests are described for a task of indexing of a powder diffraction pattern. The presented way of constructing the penalty function is to some extent problem specific, but the applied scheme may be adapted to other global search and optimisation problems, in particular to those requiring identification of multiple deep local minima.

1. Introduction

1.1. Genetic algorithms

The concept of genetic algorithms (GAs) [1,2] follows the idea of solving scientific and technical problems by learning from nature. They constitute a powerful tool for global search and high-complexity optimisation tasks in science, technology and elsewhere. Applications of this kind were initialised as early as in 1975 by de Jong [3] and develop quickly since mid 1980s.

The basic idea of GAs consists in a stochastic strategy that mimics the evolution of living organisms (here represented by individuals, i.e., points in the space of variables). In the beginning of calculations, an initial population is generated. Next generations are successively created using simplified principles of Darwinian evolution. A task-specific fitness function $f(x,y,z,\ldots)$ describing the population-member quality (that is a measure of 'adaptation to environment') is defined together with the variable ranges and other problem specific constraints. The $f(x,y,z,\ldots)$ functions exhibiting complex landscapes with many extrema can be studied. Suitable selection criteria make that the minima of the function are more 'attractive' for location of entities (the members of the population). Therefore, mostly sub areas of low f value are explored by the algorithm. GAs are often used for problems that cannot be successfully treated neither in analytical way nor by using hill-climbing search routines, because of function landscape complexity and/or extended ranges of variables.

In classical genetic algorithms, the basic genetic operators used in formation of each new population include selection, crossover and mutation, completed by the principle of elitism: the selection operator ascertains a greater chance to pass the 'good' genes to next generations for better-adapted individuals. It determines which individuals are chosen for crossover. Selection is proceeded on the basis of fitness values (or fitness ranking). The crossover operator determines how the information (bits) is exchanged between two selected individuals (parents). A part of individuals are transferred to next generations without crossover. The mutation operator typically consists of flipping a single bit with a probability called mutation rate, M. If the number of generations is large enough, using this operator ascertains that the variable space is explored as completely as possible. The elitism prevents the best solution from extinction. Global

Abbreviations: LM, local minimum; GM, global minimum; GA, genetic algorithm; PF, penalty function; VPF, variable penalty function; DPF, dynamic penalty function.

E-mail address: paszk@ifpan.edu.pl

0927-0256/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.commatsci.2008.04.033

optimisation methods based on the genetic approach are used for solving static and dynamic tasks. The importance of GAs is due to their ability of solving difficult problems for which the objective function is not defined analytically.

One of principal features of computational search and optimisation problems is the presence of constraints. There are several ways for introducing constraints into the algorithm. An important way is established by penalty functions (PFs) that may be treated as a modification of GAs at the objective function level. The PFs can be built into the definition of the objective function in any single-objective evolutionary global search algorithm. Their use permits for transformation from constrained to unconstrained search/optimisation, providing an effective soft limit of the search area and improving the convergence (for a review see Refs. [4,5]).

1.2. Search for multiple minima

In global search and optimisation, the goal is to find the global optimum for a static or varying-in-time function. For some many-parameter problems it may be quite sufficient to find one of deep local minima. GAs are appropriate for solving complicated search/optimisation problems, where the number of local optima is large or very large. Many computational tasks in science are multiple minima problems (references to some problems – protein folding, peptide conformation, drug assays, atomic clusters – requiring an analysis of multiple local minima can be found in the introduction of the Ref. [6]). In specific cases, the corresponding (typically exhibiting a complex landscape) objective functions have a number of separate minima of equal or similar depth, and the nature of the physical problem requires finding of all these deep minima. Therefore, it is desirable that the search algorithm identifies all, or at least many deep local minima of the studied function. Methods for solving this kind of problems are not frequently addressed. For simple tasks, just a systematic search, possibly hybridised with a refinement procedure, can suffice (a positive example is the algorithm for indexing powder patterns for high and medium symmetry [7] providing a list of solutions found below a predefined threshold). There exist not many numerical methods providing a list of local minima (LM) for problems of high-complexity. Typically, the given search algorithm (e.g., a genetic algorithm) can identify some local minima on the way to the global one, but many other LMs remain undiscovered.

During the search, trapping the population at local minima (LM), and long stagnation periods at each of minima may lead to serious difficulties in achieving a satisfactory exploration efficiency of any algorithm. Getting stuck in one accidental local minimum leads to a failure (due to very long stagnation time) of the calculations and thus should be avoided. The solutions in genetic algorithms also may tend to be trapped at LMs. The natural way of overcoming this difficulty is optimising the algorithm parameters, in particular the M value, before or during the calculation. Many other methods of improving the convergence have been considered, for example those involving the use of asymmetric mutation or oscillating mutation rate [8] or stretching the LMs [9].

GAs themselves (without any modification) are able to discover some local minima on the way to the global one. It has been recognised that GAs are helpful in identifying the local minima (see e.g., Refs. [10,11]). A particularly interesting way to this goal is through the use of dynamic penalty functions (DPFs). This method consists in adding constraints connected with already found LMs. GAs have been shown to be able to find local minima if a dynamic penalty function was introduced [6,10,12]. In one of applications, Manby et al. [12] have shown results for a modified GA yielding multiple energy-minimisation solutions for Al_n and C_n clusters. Using the DPF pushes the genetic algorithm to look for a new LM after a (predefined) stagnation period. Moreover, after reaching the global minimum (GM), the algorithm continues to look for other minima. Some other algorithms also have been reported to be useful in identifying multiple local minima [13–15].

1.3. Static and variable penalty functions in global search and optimisation

The idea of using penalty functions in calculations was for the first time presented in Ref. [16]. They have been sometimes used in optimisation techniques (e.g., in Ref. [17]). Several early approaches of evolutionary computations using the PFs were born in 1993–1995 (cf. the survey in Ref. [18]). The concept, kinds and use of PFs have been reviewed in more or less detail, e.g., in Refs. [5,18–27].

Hard or soft, static or variable (dynamic, adaptive) penalty functions (PFs) can be built into the definition of the objective function in any evolutionary global search algorithm in order to limit the search area and so to improve the convergence. The PFs can have their own parameters. In general, the choice of PF parameter values should depend on the extremum shape and extent in the given class of optimisation tasks. Penalising may be treated as one of methods for constraint handling (see e.g., [19,26,28,29]). Gen and Cheng [18] have presented an early review of the use of penalty functions in GAs. According to their classification, the variable penalties can be adjusted by two factors: degree of violation of constraints and the iteration number. The values of penalty function should be comparable to the values of the objective function. The particular category is the variable penalty where the definition of the penalty involves the information on the previously found extrema (as applied e.g., in Refs. [10,30]). Penalties have been occasionally used also in conjunction with other optimisation methods: penalising can become an element of tabu search [11], as for example the adaptive penalty combined with tabu search [24].

The rate of increase of the penalty components of the objective function has been shown to largely influence the algorithm performance [31]. The techniques using the PFs permit the genetic algorithm to include the infeasible solutions into the search/optimisation procedure. This is particularly important for the case of highly constrained problems, where the offspring tends to occupy the infeasible regions of the space. There are three kinds of penalties distinguished in literature: static (not varying), dynamic, and adaptive (both varying with generation number). The variation may involve a feedback from the search process. The dynamic penalty approach involves a constantly growing penalty. Using penalties in multiobjective tasks is not straightforward because of the fitness based on ranking instead of f value [32] (see also Ref. [33]). The authors of Ref. [34] have reviewed several techniques using the adaptive penalties and have proposed their own scheme employing penalties for a stable penalisation.

The PFs are now built in some general utility genetic algorithm packages such as LHG [35], GAILS [36], a GA teaching tool [37], or BASIC [38], they have also been shown to work with GAs implemented in MATLAB [22]). Their use for specific problems has been reported *e.g.* in a study of process design [39], investigation of indexing procedures [10], engineering optimisation problems [40,41], facility layout problem [42], and nuclear-power-plant safety [19].

In the present work, the properties of a genetic algorithm equipped with a feedback dynamic penalty function are studied. It is shown that its use leads to the possibility of finding multiple local minima of the studied objective function.

2. Concept of dynamic penalty function

The use of variable penalty functions (VPFs) makes that the domain of intensive search is progressively reduced that leads to a faster convergence. As mentioned in the preceding sections, the VPFs have been found to be helpful in identifying multiple deep local minima, that is attractive for solving the problems where such minima may have a real physical meanings, for example in the task of modelling the atomic clusters: some calculated clusters of identical composition but marginally differing by the energy may represent real (stable or metastable) physical objects.

In Ref. [10], the use of a soft penalty function that varied during the calculation (a dynamic penalty function, DPF) has permitted to improve the search efficiency (convergence to the global minimum) for a powder pattern indexing task; the concept from Ref. [10] will be presented here in more detail and the influence of DPF parameters on the efficiency will be discussed. The applied definition of the DPF discourages the algorithm from exploration of the earlier found LMs; moreover, a list of such minima is created. The DPF varies smoothly in order to prevent against the loss of genetic information at the stage of birth of the next generation. Its use leads to reduction of the time of stagnation at the given LM, enabling thus to find multiple LMs and to select the global one from among them. It will be shown for the powder pattern indexing example studied, how the choice of the DPF parameter values influences the opportunity of finding the global optimum in a short cpu time.

Consider a global optimisation problem where $f(\boldsymbol{x})$ is the objective function. The optimisation of $f(\boldsymbol{x})$ is replaced by the optimisation of penalised $F(\boldsymbol{x})$ function

$$F(\mathbf{x}) = f(\mathbf{x}) + p_0(\mathbf{x}) + p_1(\mathbf{x})$$

where

x is a vector representing three real variables, a, b and c,
$p_0(x) + p_1(x)$ denote the penalty terms active in the vicinity of the previously found solutions,

$$P_0(x) = \gamma(f_{av} - f(x)) \quad \text{for all } x \in \Omega_1 \cup \Omega_2 \cup \ldots \cup \Omega_{N-1}$$

$$P_1(x) = \alpha(f_{av} - f(x)) \quad \text{for all } x \in \Omega_N$$

where

f_{av} = function value averaged over the whole population,
Ω represents the field of action of DPF (the penalty range, *i.e.* an area around the given LM, in the present work it is defined as that containing points within the $\pm D$ for each variable, a, b and c),
$\alpha = \Delta N / N_{gmax}$, for N_g fulfilling the condition $N_0 \leqslant N_g \leqslant N_{gmax}$, i.e. for several generations coming just after fulfilling the convergence criterion ($\Delta N = N_g - N_0$, where N_0 = number of generation when the convergence criterion is fulfilled, N_g = number of generation after the convergence criterion is fulfilled, N_{gmax} = generation attenuation parameter),
$\alpha = 0$ otherwise,
γ = penalty strength parameter of value $\geqslant 1$; for the purposes of the present work $\gamma = 1$ was assumed.

The above definition of the feedback-type dynamic penalty function makes that at the first stage $F(x) = f(x)$ in the whole space, and since detection of the first local minimum the penalty terms are progressively added. It is suitable for function of interest but may require some modification for function of different landscape. α linearly varies with ΔN increasing during N_{gmax} generations after fulfilling the convergence criterion. The above-described approach, used in Ref. [10] but not yet described in detail, exhibits some similarities to that of Ref. [43] where the penalty increases proportionally to generation number, and to the method of deforming the objective function around the already found minima, described in Ref. [44]. The main difference in respect to the cited approaches is the smooth increase of the penalty value in the vicinity of the last local minimum found.

In the proposed indexing approach, the search is performed with predefined lower and upper bounds of each of three variables. The operators are defined in a way prohibiting solutions outside these bounds. If the condition $a < b < c$ is not fulfilled after action of an operator, a suitable swapping of the variables is performed.

The DPF varies smoothly, in order to prevent against the loss of genetic information at the stage of birth of the next generation. Its use leads to reduction of the time of stagnation at the given LM, enabling thus to find multiple LMs and to select the global one from among them. If $\gamma = 1$, then at a point close to a LM the $F(x)$ function progressively varies between $f(\boldsymbol{x})$ and f_{av}. (Including f_{av} into a penalty has been reported e.g., in Ref. [23]) makes that the already identified f minima are masked. The reason for using the smooth variations is to prevent against the genetic-information loss that would occur if the changes were abrupt. In this work, the changes are completed after N_{gmax} generations. The smooth variations concern the last identified LM, the earlier-discovered minima are fully masked.

The idea of feedback DPF is illustrated in Fig. 1, using simple cone-like and pyramid-like 2D functions. The leftmost picture shows a simple pyramid shape function f (representing a local minimum) at the moment of decision 'LM found'. During N_{gmax} generations the LM is progressively flattened (masked), so the selection pressure decreases here. The entities residing at the area can become parents of new generations and their children have freedom to leave the penalised area during the process of evolution and explore other LMs.

It is worth noting that the idea of feedback penalty has some relation to tabu search. The main difference is that the classical tabu search:

- forbids the previously found solutions,
- works in discrete-variable space,
- is not a population-based method,
- whereas the above presented approach
- penalises the previous solutions (a less severe condition),
- works in continuous-variable space,
- is a population-based method.

3. Results and discussion

A real-coded algorithm described in Ref. [10] is used for the present tests aiming for demonstration of the DPF properties. The first population is created on the basis of randomly selected entities. A fixed number of generations, N_{stag}, during which a stagnation (stable f value) is observed, is used as a criterion for decision on the local minimum detection. The output files store the current best solution of the population characterised by the lowest f value, the averaged (over the population) F value and all already-detected local minima of f[1]

[1] Distinguishing between f and F in discussing the output of the calculations is of marginal importance, as (i) decision on detection of a new local minimum in any masked area where f and F differ is virtually impossible for the function of the given kind and (ii) the influence of masked areas on averaging is negligible, because in the global search the masked areas cover an extremely small sub-volume of the space and the average F values ascribed to these areas make that these areas only occasionally can have an entity inside.

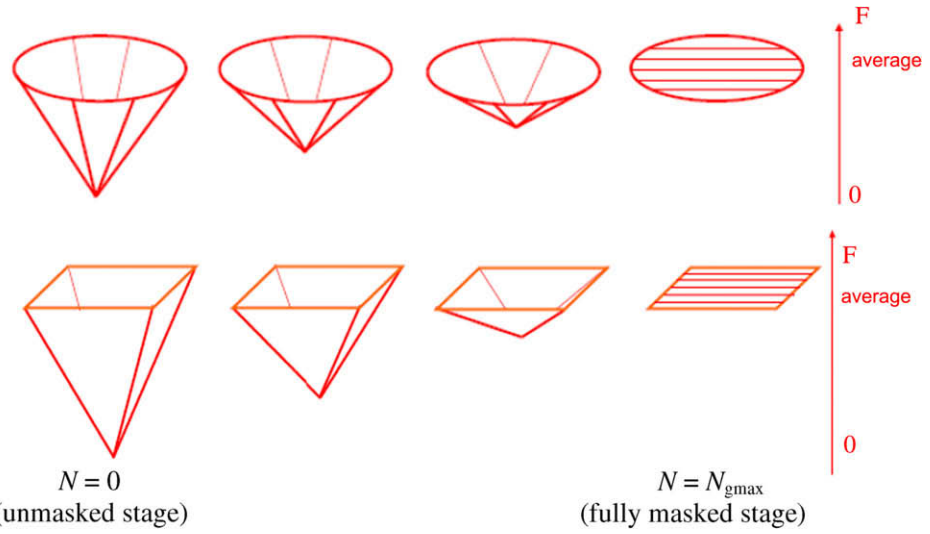

Fig. 1. Simplified scheme of the effect of 2D feedback dynamic penalty function variation at the most recently found cone shaped and pyramid shaped local minima of function *f*. The second, third and fourth pictures from the left display the *F* function values resulting from the action of DPF. The scheme assumes the ideal situation of the zero value at the extremum and average value at the side of the LM. The depth is progressively ('dynamically') reduced during a number of generations, N_{gmax}. The final *F* value corresponds to *f* averaged over the whole population.

The here-considered computational indexing problem is characterised by complex shape of the objective function: the shape exhibits many narrow local minima. The global one is difficult to locate using traditional computing methods. Therefore, GA based numerical approaches are useful here [10,45–47]. For this indexing task, the fitness function *f* is assumed to be equal to the s_N function defined in Refs. [7,8], representing a normalised measure of the difference between the experimental and the trial calculated diffraction pattern:

$$s_N(\vec{x}) = (A/N)\sum_{i=1}^{N} \epsilon_i(\vec{x})/\bar{e}_i(v) \qquad (1)$$

where

- the index N denotes the number of successive initial experimental reflections, being taken into account in the calculation,
- $\vec{x}$ is the vector of variables (for the orthorhombic system: lattice parameters a, b, c),
- A is the normalizing factor,
- $\epsilon_i(\vec{X})$ is the absolute difference between experimentally determined peak position, q_i, and position of the nearest calculated peak, Q_i, i.e., $\epsilon_i = |q_i - Q_i|$, the positions being expressed in (interplanar spacing)$^{-2}$ units,
- V is the unit-cell volume,
- $\bar{e}_i(V)$ $\bar{e}_i(V)$ is the mean distance between the calculated peaks of the trial unit-cell of volume V in the vicinity of the i^{th} experimental peak; the calculation of the volume is based on estimation method described in Ref. [48].

The above defined function has the average value of the order of 0.5 outside the minima. It can be more or less noisy depending on the experimental data. For the most complex low-symmetry single-phase case (triclinic unit cell) the number of variables is six. Possible presence of (indistinguishable) impurity peaks in the pattern causes further difficulties in analysis.

The DPF parameters influence the convergence speed, and explorative properties of the algorithm. The dependence of the optimisation run on the penalty range is illustrated in Fig. 2 for the test case of a powder diffraction pattern of $LiCu_2O_2$. For the tests shown in the figure, this range varies from 0 through 0.002 Å to 0.4 Å. The variation of *F* value for the best entity, with rising number of function calls, N_{calls}, is presented for two different mutation rates, 5% and 20%. The population size was 121, the hard constraints for *a*, *b* and *c* variables were 2.5–15 Å.

For the high mutation rate (Fig. 2a–c), the algorithm finds the global minimum in ~8000 calls as the first one. The algorithm efficiency depends on the penalty parameters applied. In absence of the penalty ($D = 0$ Å), there is no further action (the global minimum is detected, no local ones are identified). For a too small penalty range, the algorithm attempts to point out some solutions at the side an earlier found minimum area – these points ('side minima') have *f* values similar to that at the global optimum as they are very near to it (Fig. 2b). When a reasonable range ($D = 0.4$ Å) is assumed, the algorithm identifies multiple LMs easily: a clear rising trend of their depths is visible in Fig. 2c.

For the actual low *M* value, the algorithm becomes trapped at the first identified LM if $D = 0$ (effective lack of penalty, see Fig. 2d). As shown in Fig. 2e, for $D = 0.002$ Å (a small penalty range) one observes numerous attempts for looking for next LMs around the first one, the discovered local minima of *F* exhibit very similar function values. These minima apparently do not correspond to those of *f* – they represent the lowest *F* values at the side of the masked area. For a physically reasonable value of $D = 0.400$ Å, the GM is easily found in ~33,500 calls (a higher computational expense than in the case of Fig. 2a–c) and 39 deep LMs are also easily found in this run.

Table 1 lists the solutions provided by the runs 2c and 2f (Fig. 2). One observes that those with $c = 12.41$ Å dominate. The physical reason for this dominance is the fact, that the low angle peaks can be indexed only if one variable, *a*, *b* or *c*, is a multiple of 12.41 Å. The lack of indexing (i.e., a lack of existence of a well matching calculated peak) for these peaks leads to a large value of the objective function. For the 2f run, almost all solutions have one variable equal to 12.41 Å, whereas for the 2c run their number is considerably lower. This is attributed to a weaker explorative ability of the algorithm if the mutation rate is low (case 2f): the algorithm tends to find a completely new solution less easily in this case. Due to difference in mutation rate, the 2f run finds the local optima with a greater precision (seen through their lower *f* values), whereas the 2c run presents an increased diversity of solutions. This is attributed to exactly the same reason: the difference

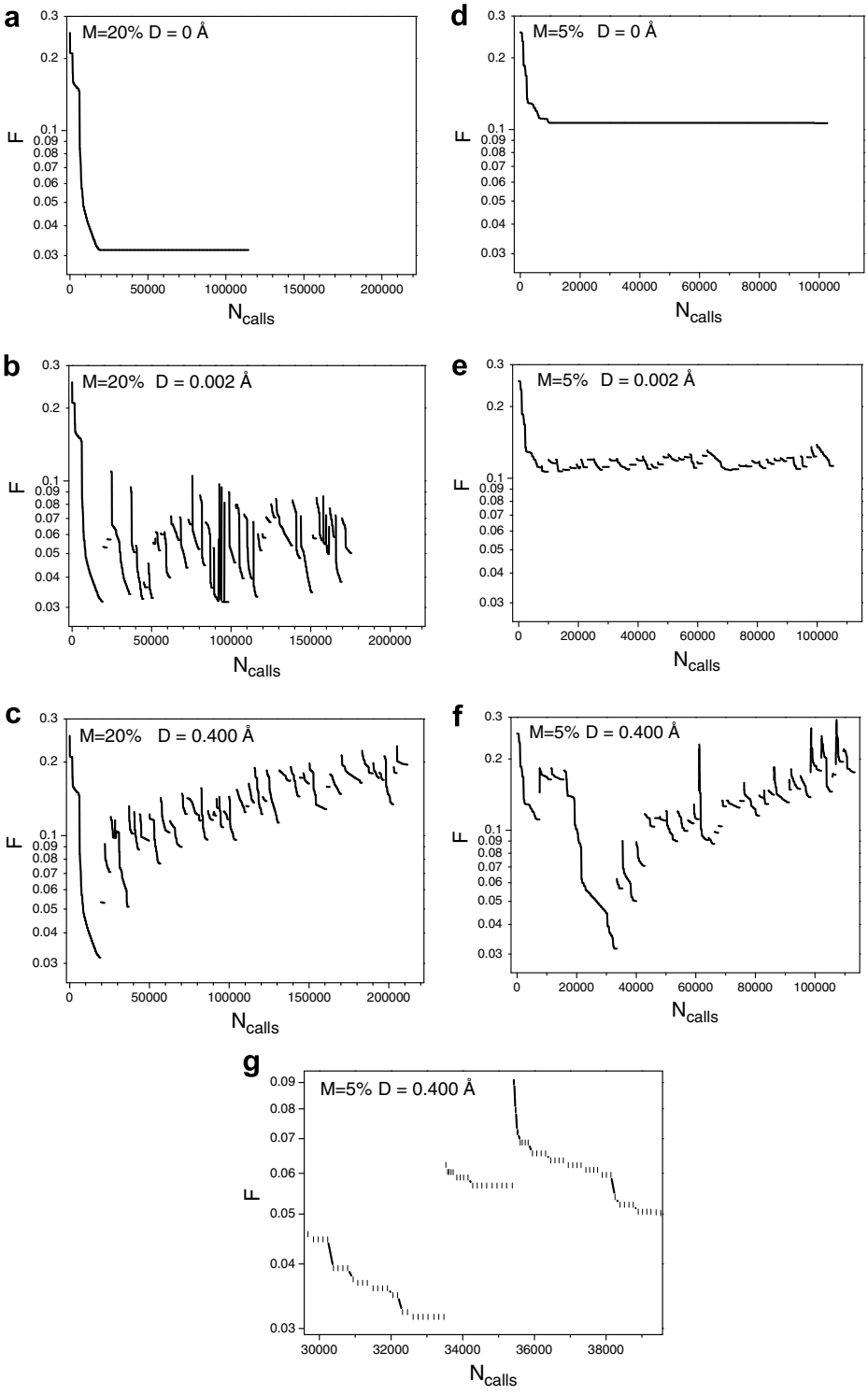

Fig. 2. Results of tests for the indexing task of lithium copper oxide, presenting the dependence of the optimisation process on the penalty range. The objective function value F is plotted (note the logarithmic scale) against the number of function calls, N_{calls}, D (the penalty range) values studied are 0, 0.002 and 0.4 Å. F represents the function value for the best entity. The examples are for mutation rate 20% (a–c) and 5% (d–f). The function value at the global minimum (f_0 (5.713, 5.724, 12.413) = 0.0316) is reached in figures (a–c) and (f). (g) Magnified part of figure (f) illustrating the objective function behaviour before and after adding a new penalty term to the function.

between explorative/exploitative abilities at different mutation rates.

It is noteworthy that the deepest minima (from A to G) are found in both, 2c and 2f runs. It can be thus concluded that finding the deepest solutions does not strongly depend on the mutation rate (only the order of finding does).

In several runs, the given minimum is apparently not refined (has a large F value). This is caused either by the liberal stop criterion or by presence of a shallow sub-minimum in the vicinity of a deeper one. This problem (proper refinement of each local minimum) can be solved by using an additional local-refinement procedure inside the area.

Table 1
Illustration of the search progress observed in the test calculations when the dimension of the domain of DPF varies

Symbol	*i*	*a* (Å)	*b* (Å)	*c* (Å)	*f*
(a) Run 2c					
A	1	5.71242	5.72503	12.41260	0.03164
C	2	5.71282	11.44925	12.41260	0.05307
D	3	5.72076	9.47421	12.41260	0.07072
E	9	6.91351	11.43125	12.41258	0.07678
F	7	4.61275	5.72116	12.41277	0.08769
I	8	3.65295	11.44231	12.41277	0.09544
J	20	6.90509	10.21502	12.40741	0.09594
(K)	14	3.84940	5.72127	12.41335	0.09618
N	6	5.72116	7.33103	12.41260	0.10129
M	10	3.02641	11.43764	12.41302	0.10294
P	18	5.72111	7.93433	12.41314	0.10815
U	13	6.30010	11.43334	12.41328	0.11541
Q	4	3.06598	5.72116	12.41281	0.11555
L	15	5.10863	5.72160	12.41114	0.11586
S	12	5.16347	11.44271	12.41278	0.12269
(X)	31	3.09585	5.54077	13.36144	0.12829
W	17	5.72003	10.94758	12.41141	0.12931
T	22	5.72292	12.40178	13.94563	0.13170
V	23	5.72028	10.39944	12.41212	0.13664
$	25	5.72015	9.01422	12.41372	0.13835
	28	6.20650	9.41440	11.44938	0.14452
	33	3.10476	12.56213	13.34412	0.14798
	32	5.54077	6.19127	13.36144	0.15825
	30	6.20966	11.14464	13.35776	0.16342
	34	3.10427	10.87455	13.37645	0.16440
	27	6.20563	7.87126	8.35500	0.16819
	37	3.09569	8.37095	13.36202	0.16892
	29	5.72469	6.20650	10.84165	0.17231
	36	6.19731	8.35812	13.35805	0.17307
	35	3.16244	7.26918	14.99269	0.17971
	39	6.69892	10.79030	12.63909	0.18035
	40	3.63410	5.28861	7.55906	0.19461
	16	5.72161	12.41054	12.61275	0.22190
	19	5.71865	12.40612	13.35610	0.22551
	26	7.86438	8.36605	12.40253	0.23777
B	5	5.72426	8.57251	12.41260	0.23836
G	11	4.40899	11.43037	12.41350	0.24059
	21	5.72330	6.43863	12.40004	0.24151
	38	6.69703	8.37447	12.63821	0.26309
	24	5.72249	9.90969	12.41091	0.27214
(b) Run 2f					
A	4	5.71437	5.72445	12.41283	0.03159
B	6	5.72485	8.57064	12.41283	0.05022
C	5	5.72468	11.42531	12.41283	0.05679
D	7	5.72099	9.47551	12.41283	0.07072
F	17	4.61271	5.72146	12.41196	0.08759
G	11	4.40930	11.42894	12.41369	0.08988
(H)	16	5.71930	7.68790	12.41283	0.09094
L	18	5.31050	5.72130	12.41307	0.09758
M	12	3.20515	11.44482	12.40634	0.09944
(O)	8	3.56556	5.72110	12.41283	0.10350
I	13	3.70027	11.44222	12.41283	0.10695
Q	10	3.11488	5.72499	12.40808	0.10988
(R)	1	6.53324	11.80959	12.40588	0.11107
S	14	5.23977	11.44021	12.41283	0.11134
T	23	5.71976	12.40490	13.35553	0.11516
V	21	5.72050	10.21216	12.41306	0.12450
W	20	5.72010	10.96226	12.41318	0.12618
(Y)	25	8.08950	8.15282	12.40774	0.12900
N	22	5.71904	7.27551	12.41331	0.13287
(Z)	9	5.72526	12.41283	12.77607	0.13597
(&)	24	5.72254	6.67777	12.40317	0.13722
$	30	5.72006	9.01446	12.41334	0.13805
P	26	5.72096	8.14217	12.41149	0.14132
	28	5.71584	11.85059	12.41403	0.14847
	27	5.72130	6.15003	12.41316	0.15907
	19	4.07094	5.72130	12.41283	0.16082
	2	6.20316	6.52638	11.79840	0.16377
	3	3.14800	6.19421	11.76933	0.16381
E	35	6.91760	11.40939	12.41269	0.16812
U	36	6.28993	11.39032	12.41736	0.17175
	29	5.72300	12.40304	13.84736	0.17458
	40	4.62960	8.38015	12.41343	0.17583
J	33	6.91407	10.10064	12.41613	0.17839
	39	3.62161	9.25427	12.41910	0.19443
	15	5.23977	11.44021	12.41283	0.22055
	31	5.72006	9.01446	12.41334	0.22200
	34	3.36685	6.91407	12.41613	0.24262
	37	6.28993	11.39032	12.41736	0.24870
	32	5.72006	9.01446	12.41334	0.26010
	38	3.08428	12.41071	13.56299	0.28601

Forty solutions listed were obtained in the runs c and f displayed in Fig. 2. The solutions are ordered with the *F* value. The A–Z, and $ symbols denote 26 solutions found in the 2c and 2f output data sets (symbols in parentheses concern unpaired solutions). Underlined are lower values in pairs.

The use of the feedback dynamic penalty is demonstrated above to help in achieving the goals important for the given test application in powder diffraction pattern indexing:

- improving the exploration efficiency permitting for finding the global optimum (reducing the trapping effect),
- smaller sensitivity to the choice of mutation rate value, and
- identifying numerous minima of a complex function.

The value of *D* parameter is intended to ascertain that the masked area covers the domain around a LM, where the *f* values are much smaller than the average. Its exact value is not of high importance – only with too low values, the number of side minima tends to grow, so finding the global solution may become difficult then.

4. Summary

Variable penalty functions are a valuable tool for improving the properties of genetic algorithms. It is demonstrated in the present paper that a genetic algorithm equipped with a feedback penalty is able to find numerous local minima. Moreover, its convergence is improved at conditions of unoptimised low mutation rate.

An interesting issue is the question whether the algorithm of the described type can detect some or multiple local minima. Typically, 'numerous' should be treated as 'not much more than some tens', because (except ill-shaped functions) looking for thousands or more minima seems useless; moreover, the presence of too many terms in the penalty definition prohibitively slows down the calculations. Practically, the search would concern a limited number of deepest minima that are the most important for practical problems.

The principle of smooth 'masking' the last found minimum and complete masking all earlier found minima is easy to implement for a function of recognised properties as that analysed in the present paper. The applied approach may need to be modified for cases of more complex function landscapes. For the considered case, the simplicity of the approach is justified by the approximate knowledge of the extent of the size of the local minimum area, and by the fact that outside the local and global minima the function value is of a known order of magnitude. The observation that the calculations are successful with the use of non-optimised mutation rate shows that an implementation of the penalty of the described kind may be advantageous in various search and optimisation tasks, as less attention can be paid to finding the best *M* value before the GA run.

The described tests enabled determination of the penalty function properties. The obtained results are promising as concerns extending the application to tasks of higher complexity, e.g., for indexing of powder patterns collected for lower symmetry polycrystals.

Despite the presented way of constructing the penalty function is, to some extent, problem specific, the proposed scheme may be adapted or modified for exploration of global and multiple local optima in any other task.

References

[1] J.H. Holland, Adaptation in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor, MI, USA, 1975.
[2] D.E. Goldberg, Genetic Algorithms in Search Optimization and Machine Learning, Addison-Wesley, Reading, MA, USA, 1989.
[3] K.A. De Jong, Analysis of the Behavior of a Class of Aenetic Adaptive Systems (Thesis), The University of Michigan, College of Literature, Science and the Arts, Computer and Communications Sciences Department, Technical Rep. N^0 185, University of Michigan, 1975.
[4] D.W. Coit, A.E. Smith, D.M. Tate, Adaptive penalty methods for genetic optimization of constrained combinatorial problems, INFORMS J. Comput. 8 (1996) 173–182.
[5] C.A. Coello Coello, Theoretical and numerical constraint–handling techniques used with evolutionary algorithms: A survey of the state of the art, Comp. Meth. Appl. Mech. Eng. 191 (2002) 1245–1287.
[6] M.A. Moret, P.M. Bisch, F.M.C. Vieira, Algorithm for multiple minima search, Phys. Rev. E 57 (1998) R2535–R2538.
[7] W. Paszkowicz, Application of optimization to powder pattern indexing, J. Appl. Crystallogr. 20 (1987) 166–172.
[8] W. Paszkowicz, Properties of a genetic algorithm extended by a random self-learning operator and asymmetric mutations: A convergence study for a task of powder-pattern indexing, Anal. Chim. Acta 566 (2006) 81–98.
[9] K.E. Parsopoulos, V.P. Plagianakos, G.D. Magoulas, M.N. Vrahatis, Objective function 'stretching' to alleviate convergence to local minima, Nonlin. Anal. 47 (2001) 3419–3424.
[10] W. Paszkowicz, Application of the smooth genetic algorithm for indexing powder patterns: test for the orthorhombic system, Mater. Sci. Forum 228–231 (1996) 19–24.
[11] M. Pirlot, General local search methods, Eur. J. Oper. Res. 92 (9) (1996) 493–511.
[12] F.R. Manby, R.L. Johnston, C. Roberts, Predatory genetic algorithms, Commun. Math. Comput. Chem. 38 (1998) 111–122.
[13] D. Cvijovic, J. Klinowski, Taboo search: An approach to the multiple minima problem, Science 267 (5198) (1995) 664–666.
[14] S. Salhi, N.M. Queen, A hybrid algorithm for identifying global and local minima when optimizing functions with many minima, Eur. J. Oper. Res. 155 (2004) 51–67.
[15] R. Chelouah, P. Siarry, A hybrid method combining continuous tabu search and Nelder–Mead simplex algorithms for the global optimization of multiminima functions, Eur. J. Oper. Res. 161 (2005) 636–654.
[16] R. Courant, Variational methods for the solution of problems of equilibrium and vibrations, Bull. Am. Math. Soc. 49 (1943) 1–23.
[17] J.A. Snyman, N. Stander, W.J. Roux, A dynamic penalty function method for the solution of structural optimization problems, Appl. Math. Model. 18 (1994) 453–460.
[18] M. Gen, R. Cheng, A survey of penalty techniques in genetic algorithms, in: Proceedings of IEEE International Conference on Evolutionary Computation, Nagoya, Japan, 20–22 May 1996, pp. 804–809.
[19] Z. Michalewicz, A survey of constraint handling techniques in evolutionary computation methods, in: J.R. McDonnell, R.G. Reynolds, D.B. Fogel (Eds.), Proceedings of the 4th Annual Conf. on Evolutionary Programming, MIT Press, Cambridge, MA, 1995, pp. 135–155.
[20] S. Martorell, S. Carlos, A. Sánchez, V. Serradell, Constrained optimization of test intervals using a steady–state genetic algorithm, Reliab. Eng. Syst. Safety 67 (2000) 215–232.
[21] A.E. Smith, D.W. Coit, Penalty functions, in: T. Baeck, D.B. Fogel, Z. Michalewicz (Eds.), Evolutionary computation 2. Advanced algorithms and operators, section 7, Institute of Physics Publishing, Bristol & Philadelphia, 2000, pp. 41–48.
[22] R. Sarker, C. Newton, A genetic algorithm for solving economic lot size scheduling problem, Comput. Ind. Eng. 42 (2002) 189–198.
[23] H.J.C. Barbosa, A.C.C. Lemonge, A new adaptive penalty scheme for genetic algorithms, Inform. Sci. 156 (2003) 215–251.
[24] S. Kulturel-Konak, B.A. Norman, D.W. Coit, A.E. Smith, Exploiting tabu search memory in constrained problems, INFORMS J. Comput. 16 (2004) 241–254.
[25] M. Dolen, H. Kaplan, A. Seireg, Discrete parameter-nonlinear constrained optimization of a gear train using genetic algorithms, Int. J. Comput. Appl. Technol. 24 (2005) 110–121.
[26] Ö. Yeniay, Penalty function methods for constrained optimization with genetic algorithms, Math. Comput. Appl. 10 (2005) 45–56.
[27] E.P. Dadios, J. Ashraf, Genetic algorithm with adaptive and dynamic penalty functions for the selection of cleaner production measures: A constrained optimization problem, Clean Technol. Environ. Policy 8 (2006) 85–95.
[28] A.H.C. van Kampen, C.S. Strom, L.M. C Buydens, Lethalization, penalty and repair functions for constraint handling in the genetic algorithm methodology, Chemometr. Intell. Lab. Syst. 34 (1996) 55–68.
[29] J. Arabas, Wykłady z algorytmów ewolucyjnych (Lectures on Evolutionary Algorithms, in Polish), Wydawnictwa Naukowo-Techniczne, Warsaw, 2001, p. 50.
[30] H. Dakuo, W. Fuli, Feedback penalty function based on genetic algorithm, in: Proceedings 4th World Congress on Intelligent Control and Automation, IEEE, 2002, pp. 3153–3155.
[31] S. Kazarlis, V. Petridis, Varying fitness functions in genetic algorithms: Studying the rate of increase of the dynamic penalty terms, Parallel Problem Solving from Nature – PPSN V, Lect. Notes Comput. Sci. 1498 (1998) 211–220.
[32] A. Konak, D.W. Coit, A.E. Smith, Multi-objective optimization using genetic algorithms: a tutorial, Reliab. Eng. Syst. Safety 91 (2006) 992–1007.
[33] M. Marseguerra, E. Zio, S. Martorell, Basics of genetic algorithms optimization for RAMS applications, Reliab. Eng. Syst. Safety 91 (2006) 977–991.
[34] P. Nanakorn, K. Meesomklin, An adaptive penalty function in genetic algorithms for structural design optimization, Comput. Struct. 79 (2001) 2527–2539.
[35] R. Östermark, A multipurpose parallel genetic hybrid algorithm for non-linear non-convex programming problems, Eur. J. Oper. Res. 152 (2004) 195–214.
[36] Guided Adaptive Iterated Local Search – Method and Framework, Technical Report AIDA-04-05, klausvpp@intellektik.informatik.tu-darmstadt.de.
[37] A. Parandekar, Genetic algorithm-based optimizer: A Java based teaching tool, in: U.-M. O'Reilly (Ed.), Proceedings Graduate Student Workshop, Orlando, FL, 1999, pp. 392–393.
[38] E.G. Shopova, N.G. Valdieva-Bancheva, BASIC – A genetic algorithm for engineering problems solution, Comput. Chem. Eng. 30 (2006) 1293–1309.
[39] K. Wang, A. Salhi, E.S. Fraga, Process design optimisation using embedded hybrid visualisation and data analysis techniques within a genetic algorithm optimisation framework, Chem. Eng. Process. 43 (2004) 657–669.
[40] C.A. Coello Coello, Use of a self-adaptive penalty approach for engineering optimization problems, Comput. Ind. 41 (2000) 113–127.
[41] E.M.R. Fairbairn, M.M. Silvoso, R.D. Toledo Filho, J.L.D. Alves, N.F.F. Ebecken, Optimization of mass concrete construction using genetic algorithms, Comput. Struct. 82 (2004) 281–299.
[42] C.R. Shebanie, II, An integrated, evolutionary approach to facility layout and detailed design, Master Thesis University of Pittsburgh, School of Engineering, 2004.
[43] J.A. Joines, C.R. Houck, On the use of non-stationary penalty functions to solve nonlinearconstrained optimization problems with GA's, Evolutionary Computation, in: Proceedings of the First World Congress on Computational Intelligence, 27-29 06 1994, vol. 2, IEEE, 1994, pp. 579–584.
[44] J. Arabas, Wykłady z algorytmów ewolucyjnych, Lectures on Evolutionary Algorithms, in Polish, Wydawnictwa Naukowo-Techniczne, Warsaw 2001, p. 218.
[45] B.M. Kariuki, S.A. Belmonte, M.I. Mc Mahon, R.L. Johnston, K.D.M. Harris, J. Nelmes, A new approach for indexing powder diffraction data based on whole-profile fitting and global optimization using a genetic algorithm, J. Synchrotron Radiat. 6 (1999) 87–92.
[46] S.A. Belmonte, B.M. Kariuki, M.I. McMahon, R.L. Johnston, K.D.M. Harris, R.L. Nelmes, Powder pattern indexing based on a genetic algorithm optimisation of a whole-profile fit, IUCr Commission Powder Diffr. Newsl. 21 (1999) 4–5.
[47] J.A. Hageman, R. Wehrens, R. De Gelder, L.M.C. Buydens, Powder pattern indexing using the weighted crosscorrelation and genetic algorithms, J. Comput. Chem. 24 (2003) 1043–1051.
[48] W. Paszkowicz, On the estimation of the unit-cell volume from powder diffraction data, J. Appl. Crystallogr. 20 (1987) 161–165. and 529.