Lethalization, penalty and repair functions for constraint handling in the genetic algorithm methodology

A.H.C. van Kampen a, C.S. Strom b, L.M.C. Buydens a,.*

a Laboratory for Analytical Chemistry, Catholic University Nijmegen, Toernooiveld 1, NL-6525 ED Nijmegen, The Netherlands
b Laboratory for Solid State Chemistry, Catholic University Nijmegen, Toernooiveld 1, NL-6525 ED Nijmegen, The Netherlands

Received 5 September 1995; revised 17 January 1996; accepted 18 January 1996

Abstract

A genetic algorithm was designed to find low energy distributions of ions above a crystal surface. To solve this constrained optimization problem, several constrained handling techniques were applied, i.e., lethalization, penalty and repair functions. It was shown that the simple lethalization scheme performs very well, and was at least comparable to some of the penalty functions. This was unexpected because, in general, it is believed that a severe penalization (i.e., lethalization) leads to poor results. An analysis of the fitnesses of trial distributions that violated a constraint as a function of the time, suggested that the properties of genetic based search were responsible for this result. From this it was concluded that a genetic algorithm in combination with lethalization may be an good choice to solve constrained optimization problems if the design of optimal penalty functions is difficult or impossible.

Keywords: Genetic algorithm methodology; Repair functions; Crystal surface

1. Introduction

Ever since John Holland introduced genetic algorithms (GAs) [1–3], this family of global optimization methods have been used as a powerful strategy for solving a wide range of optimization problems [4–7]. They derived their name from the fact that they are more or less based on population genetics: a population of trial solutions is iteratively improved via selection of above-average solutions, originating from modifications of preceding trial solutions.

When constraints are imposed on the problem parameters of the optimization problem, a need for a technique arises to handle these constraints during optimization. Many optimization problems involve constraints, e.g., [8,9], and also in chemistry constrained optimization problems can be identified [10–13].

Different techniques are available for handling constraints, e.g., Lagrange multipliers [14] and mathematical programming [15,16]. However, the problems that are undertaken by GAs, often do not allow the use of these...
methods because the constraints cannot explicitly be defined in terms of the problem parameters, i.e., only after evaluation of a trial solution it may become clear whether or not a constraint is violated. In these situations the GA has to rely on other techniques to handle these constraints.

The first is the use of crossover and mutation operators which are designed to suit a specific constrained optimization problem, and only modify the problem parameters such that the constraints are always satisfied. The most illustrative examples are given by the operators designed for handling combinatorial problems with GAs [3,17]. These kind of operators are not investigated in the current research. The second technique is to represent the optimization problem such that it is impossible to violate a constraint. This approach was not considered explicitly in this paper, although the representation for the application discussed in this paper was such that it contributes to satisfying the constraints. Two other types of constraint handling techniques comprise the repair [18,19] and penalty functions [18,20–23,19,24–26] which are the subject of this paper.

Penalty functions (P) were introduced by Courant [27], and are added as a weighted term to the objective function (G) that must be optimized, i.e., the constrained problem G is transformed into an unconstrained problem (the compound function of P and G). Effectively, for minimization problems, the penalty function increases the value returned by the objective function if a constraint is violated, and is dependent on the degree of violation. There is no general methodology for designing a penalty function for a specific optimization problem, although the functional form and strength of the penalty function are considered to be of great importance.

The use of a penalty function implies that the GA performs its optimization task in a search space defined by the functions P and G. This space may have completely different characteristics than the space defined by the constrained function G alone. This change in search space characteristics is mainly caused by the functional form of the penalty function, and may affect the search performance of the GA.

Another important parameter, affecting the search performance, is the magnitude of the influence of P on G, i.e., the strength of the penalty function. It is well known that when the strength is too large, the search algorithm is not allowed to enter the non-feasible region (not allowed part of search space), and consequently, might spend more time in finding feasible regions than in finding the optimum. For the GA, a large strength may also be the cause for deceptive properties of the search space [3] which may severely hinder the optimization process [21]. However, if the strength is too small, the search algorithm will spent too much time in evaluating invalid solutions. Since solutions of constrained optimization problems are often located at the border of the non-feasible and the feasible region, it is considered important that the search algorithm can move through both regions, i.e., using an optimal penalty strength.

A special kind of penalty function is the so-called lethalization function which assigns a severe but equal penalty value to each solution violating a constraint, i.e., the search algorithm is not allowed to move in the non-feasible region, and therefore is not expected to show a high performance, or to result in an acceptable solution.

An alternative strategy for constraint handling is to use repair functions which check for violations of the constraints by the current set of parameters (trial solution), and if necessary, adjust (repair) these parameters such that the trial solution is mapped back into the feasible region. It is obvious that this strategy requires a heuristic to move the trial solution in such a way that the current set of problem parameters is not distorted too much, preventing deterioration of the search performance. The design of a repair function requires a thorough understanding of the problem. For simple problems (test functions) this may be available, but for complex real-world optimization problems, such as those arising in chemistry and physics, a lack of knowledge about the behavior of the system may lead to an inadequate design of a repair heuristic which consequently will result in a poor performance of the GA. Since repair functions are often custom-tailored to the optimization problem, they are of less general use than the penalty functions. Furthermore, they are often computationally intensive to run.

The application of GAs to constrained optimization problems has mainly concentrated on the design of penalty functions, but still no general recipe for designing these functions resulted. Neither there is a general methodology for designing repair functions. Consequently, one often has to derive at an adequate penalty or repair function by a process of trial and error.
In this paper a repair function and several penalty functions were applied to the problem of finding low energy distributions of ions above a crystal surface. This problem was constrained to find distributions which were free of overlapping ions. In contrast to the discussion above, it is shown that lethalization performs at least comparable to the other penalty functions.

2. Experimental

2.1. Software

The GA application described in this paper was written in standard ANSI C, with use of the toolbox GATES (Genetic Algorithm Toolbox for Evolutionary Search), version 2.0 [28,29]. The APL program SURFPOT [30] was used for calculating Coulomb potentials due to the crystal slices.

2.2. Problem description

Crystals have been known by man from the earliest times, for, as we now realize, many forms of solid matter are crystalline in character, and have excited attention because of their elegant symmetry and their glorious colors. Nowadays, crystals play an important role in the production of solar cells, and are found in, for example, many electronic devices (semi-conductor), digital watches (quartz), and CD-players as part of the laser. In industry, crystallization is used as a purification technique for the production of large quantities of fertilizers, sugar, and many other products. Also in the world of medical sciences, interest in crystal growth processes have grown considerably because teeth, bones, renal calculi, etc., are the result of a crystallization process.

From this it becomes clear that crystals posses very interesting chemical and physical properties which are determined by their microscopic and/or macroscopic structure. Therefore, it is very important that the growth (formation of a crystal out of a mother phase, which may be a vapour, a solution, a melt, etc.) of a crystal, can be controlled external factors, e.g., the temperature, in order to obtain a crystal with the desired structure. Consequently, it is important to understand the growth process of the crystal, i.e., it is important to understand the processes that take place at the interface between the crystal surface and the mother phase [31–39].

The determination of the structure of this interface layer is often performed by Monte Carlo [40,41] or molecular dynamics [42] methods. The GA is, potentially, an additional method to asses the interface structure. The approach taken by the current implementation of the GA was, however, very different from these classical methods. It was, however, not the aim of this paper to discuss these differences. Neither are the physical properties of the simulated system discussed. These will be the subject of a future paper. Consequently, the composition of the interface layer that resulted from the simulation was only discussed in terms of potential energy (the minimization criterion in our optimization), and the number of ions placed in this layer. From these results, the performance of the repair, lethalization, and penalty functions were evaluated, and an analysis to explain their behavior is discussed.

2.2.1. Constraints

In this paper the interfacial layer of a strongly undulated boundary along the (111)-surface 1 of sodium chloride (NaCl) in a melt [30] was considered, which required the determination of the distribution of Na\(^+\) and Cl\(^-\) ions in this layer. The distribution comprising the lowest potential energy was considered to be the most favorable. This distributions of ions in the interfacial volume was subjected to two constraints:

---

1 For more details on the nomenclature of crystal surfaces, see for example [43,44].
CI: The Cl\(^{-}\) and Na\(^{+}\) ions were only allowed to occupy the positions in the interfacial volume in which the potentials due to the crystal were of opposite sign;

C2: The ions, which were represented as a sphere with a Vanderwaals radius, \(r\), in the interfacial volume were not allowed to overlap.

Accordingly, the GA had to be implemented such that these two constraints could be satisfied. The representation of the trial distributions was chosen such that the crossover operator could not violate constraint CI. Furthermore, a new mutation operator was developed to suit the first constraint.

For constraint C2 it turned out to be more difficult to design a suitable representation and/or operator to ensure satisfaction of this constraint, and consequently, a penalty or repair function had to be used.

2.2.2. The potential energy function

In the determination of the distribution of sodium and chloride ions above the crystal surface of NaCl, only the Coulomb potential energy, which is by far the dominant contribution to the total energy of the ionic structure, was considered. Consequently, this optimization comprised the search for the lowest energy distribution, and could be regarded as a variable size subset selection problem, i.e., from all possible positions at which an ion could be placed a subset was chosen for which the number of ions was not fixed.

A set of ions (Na\(^{+}\) and Cl\(^{-}\)) must be placed in the interfacial volume such that they minimized the potential energy, i.e., as negative as possible. The total potential energy was calculated by adding the potential energy due to interactions between the crystal and ions in the interfacial volume (solid–fluid interaction \(E_{SF}\)), and the potential energy due to the pairwise interactions of the ions in this volume (fluid–fluid interaction \(E_{FF}\)): 

\[
E_{pot} = E_{SF} + a \times E_{FF} \\
E_{pot} = \sum_{i=1}^{N} q_{i} V_{i} + a \sum_{i=1}^{N-1} \sum_{j>i}^{N} \frac{q_{i} q_{j}}{r_{ij}}
\]

where \(N\) denotes the number of ions in the distribution, \(V_{i}\) is the potential at point \(i\) with coordinates \((x_{i}, y_{i}, Z_{i})\), \(q_{k}\) is the charge of ion \(k\) (\(-1\) for chloride and \(+1\) for sodium) and \(r_{ij}\) denotes the Euclidean distance between ions at positions \(i\) and \(j\), respectively.

This energy function \(E_{pot}\) was only a very rough approximation of the real physical situation. However, to be able to parameterize this potential energy function (to a certain extent) towards a more realistic situation, a weighting factor \(a\) was introduced, which was set to 0.75 for the current investigations.

\(E_{SF}\) was always negative because \(q_{k}\) and \(V_{i}\) must be of opposite sign according to constraint CI. \(E_{FF}\) could assume either positive or negative values, depending on the distribution. However, a favorable distribution should give rise to a negative fluid–fluid interaction. Consequently, \(E_{pot}\) could either be positive or negative.

2.2.3. Representation of the interfacial volume

The interfacial volume was defined as a three dimensional grid of position points adjacent to the (111)-crystal surface of NaCl, covering a volume of \(m \times n \times l\) unit cells (a unit cell is the basic building block from which the whole crystal can be reconstructed by translation operations only [43,44]). This collection of grid points functioned as a pool of coordinates \(x, y, z\) for placing the particles. Actually, the grid was specified independently for each type of particle (Na\(^{+}\), Cl\(^{-}\)), and had the following properties: \(x_{i}\) and \(y_{i}\) were coordinates defined for a plane parallel to the undulated boundary along the (111)-face of NaCl. This plane covered an area of \(m \times n\) unit cells. The corresponding coordinate \(z_{0}\) of the first plane above the crystal surface, (i.e., the first layer of the grid), was the coordinate corresponding to the distance of closest approach between the crystal and ion, without them having contact. Since the surface of this plane is not flat, the coordinate \(z_{0}\) depended on the position \(x_{i}, y_{i}\), i.e., \(z_{0} = z_{0}(i,j)\). All subsequent planes (grid layers) were obtained by adding a fixed step \(\delta\) to \(z_{0}(i,j)\), i.e., \(z_{k}(i,j) = z_{0}(i,j) + k\delta\). This explains why two separate grids for the sodium and chloride ions were defined, i.e., \(z_{0Na}(i,j) \neq z_{0Cl}(i,j)\).
Fig. 1. String representing a trial distribution with 3 Na⁺ and 3 Cl⁻ ions. Each number denotes an integer index corresponding to specific point x, y, z in the interfacial volume. The value ‘−1’ is a dummy value, which allows for variable size distributions.

At each of the grid points a potential due to the crystal was calculated by the SURFPOT program [30]. Thus, the coordinates of each point of the two separate grids were accompanied by the value of the potential, and served as input for the GA.

For the experiments described in this paper each grid was defined such that it covered 3 X 3 X 1 unit cells, with a resolution of 30 X 30 X 10 points. It can be shown that the distance between two adjacent points in the grid in x, y and z direction were 0.28 Å, 0.28 Å and 0.63 Å respectively. The Vanderwaals radii of Na⁺ and Cl⁻ were taken as 0.95 Å and 1.80 Å. Since each unit cell of NaCl contains 4 Na⁺ and 4 Cl⁻ ions, it was expected to find a distribution with approximately 72 ions.

2.2.4. Encoding of the trial distributions

Each string represented a trial distribution such that the parameters (encoded as integers) on these string denoted indices corresponding to a specific point x, y, z in the grid (interfacial volume).

Although the size of the trial distributions was not fixed, strings with a constant length were used. Although distributions with a size of about 72 ions were finally expected, the number of parameters on the strings provided for a maximum of 200 ions of each type. This was necessary because the trial distributions at the start of the simulation process could contain much more ions.

For the current system with two different types of ions (Na⁺ and Cl⁻), the strings were divided in two separate parts; the left part defined the Na⁺ distribution, and the right part defined the Cl⁻ distribution (Fig. 1). This ensured a direct correspondence with the two separate grids for Na⁺ and Cl⁻.

Each parameter could assume any integer value from 0 to 9000 (30 X 30 X 10 points). However, constraint CI, to which the distribution was imposed, excluded specific values. A parameter value equal to −1 denoted a dummy value with no correspondence to a specific point, and consequently, allowed for variable sized distributions.

2.2.5. Initialization of the trial distributions

In most GA applications, the population of strings is initialized at random at the start of the run. For the current application problem, however, a different type of initialization was developed to ensure that all initial trial distributions obeyed both constraints CI and C2. This initialization procedure operated as follows:

1. select ion p (Na⁺ or Cl⁻ ion) at random;
2. select grid point with opposite sign of the potential at random, and place ion p at this point (constraint CI is satisfied);
3. mark all grid points which are within a distance di# of p. These points are no longer candidates to be selected in the next iteration (step 2).

\[
d_{ij} = \begin{cases} 
  r_p + r_{Na^+}, & \text{for the grid specifying the Na}^+ \text{ distribution} \\
  r_p + r_{Cl^-}, & \text{for the grid specifying the Cl}^- \text{ distribution}
\end{cases}
\]

where \( r_i \) denotes the Vanderwaals radius of particle \( i \);
4. continue with step 1 until both grid systems are completely occupied, i.e. the interfacial volume is filled with ions.

2.2.6. Evaluation of the trial distributions

The quality of the trial distributions was reflected by their potential energy ($E^{pot}$), or the fitness value which was derived from it. By convention, the fitness was maximized during the optimization process, and accordingly, was defined by inverting the sign of the potential energy, i.e., $F = -E^{pot}$. If a lethalization or penalty function was used to reflect violations of constraint $C2$, the fitness value was adjusted accordingly (see below).

2.2.7. Selection of above-average solutions

Once the (penalized) fitness values were assigned to the strings, the selection of above-average strings for the next population could take place. In the current implementation rank-based threshold selection [45] was used which is based on the rank of the strings according to their fitness. In this method a threshold is chosen defining the better fraction of the population, and subsequently only strings from this fraction are selected at random to take place in the new population. In addition the rank-based method was combined with elitism selection [4] which provided for the best strings to be always selected. An elitism fraction is chosen defining the number of best strings that were copied to the new population. This compound selection method (elitism and rank based selection) is shown in Fig. 2.

2.2.8. Crossover and mutation

In the current implementation uniform crossover [4] was used to combine trial distributions. This operators selected a pre-specified number of parameters (indices) at random, and exchanged these with those on the paired string. Due to this re-combination, trial distributions with overlap between the ions could generated. Note that this operator never violated constraint $C1$ because the ion was not separated from its position with the corresponding potential.

Subsequently to re-combination, the trial distributions were subjected to mutation. For the current application a new mutation operator was developed which made semi-random changes to the strings without violating constraint $C1$, but did not impose constraint $C2$.

1. Select parameter $p$ on string at random.
2. Generate integer value $x$, between $-1$ (dummy value) and 9000 (last grid point).
3. Replace value of parameter $p$ with $x$ if this position has the correct potential. Otherwise delete particle $p$ from distribution.

![Fig. 2. Elitism and rank-based threshold selection for selection of above-average solutions from the GA population.](image-url)
2.3. Constraint handling techniques

2.3.1. The repair function

A repair function was designed, which was applied subsequently to the mutation operator, and redistributed the Na\(^+\) and Cl\(^-\) ions such that all overlap between the ions vanished. This repair function was applied to each string in the GA population, and comprised two separate operations:

1. Remove overlapping ions,
2. select ion \( p \) at random from trial distribution;
3. remove all ions which overlap with ion \( p \);
4. mark ion \( p \), so it will not be selected again in step (a);
5. continue with step (a) until all ions are checked for overlap.

Since this procedure deleted ions from the trial distribution, it was possible that new holes in the distribution emerged in which again ions could be placed without overlap. Therefore, a second step was necessary to complete the distribution again.

2. Complete interfacial volume,
1. mark all grid points which are not allowed due to overlap with remaining ions of the current trial distribution;
2. if unoccupied grid points (gaps in the interfacial volume) remain, a new ion is selected from the available types (Na\(^+\) and Cl\(^-\)). Each type of ion has a predefined probability for being selected, which is given by \( P_{Na} \) and \( P_{Cl} = 1 - P_{Na} \). After selection of an ion, it is placed at an unoccupied grid point, which is selected at random;
3. mark all grid points which become forbidden due to overlap with this new ion;
4. continue with step (b).

This procedure to complete the interfacial volume was essentially the same as the procedure for initialization of the strings, but it started out from a partially filled grid. The weighted selection of either a sodium or chloride ion was necessary to bias the ratio of Na\(^+\) and Cl\(^-\) ions in the distribution, although it was impossible to predict the final ratio.

Since this repair function redistributed part of the ions, it could be regarded as a specific kind of mutation. Accordingly, the mutation probability of the regular mutation operator should be decreased in order to maintain an acceptable level of search performance.

2.3.2. Lethalization

The most straightforward method for handling constraint \( C2 \) was to lethalize solutions which violated this constraint, i.e., assigning a large negative fitness value corresponding with a large positive potential energy. The expected drawback of this approach was that no discrimination was made between distributions with different degrees over ion overlap (fitness values are equal). In addition, if a few trial solutions did not violate the constraints, these ‘super-individuals’ could take over the population in a few generations, resulting in premature convergence.

2.3.3. The penalty functions

Two different penalty functions \( P \) were defined:

1. \( P_{count} = 1.0 + N \times \alpha \)
2. \( P_{distance} = 1.0 + \sum_{i,j>i} \left( \alpha - \min \left( 1, \left[ \frac{d_{ij}}{r_i + r_j} \right] \right) \times \alpha \right) \)

where \( N \) denotes the number of constraint violations. The distance \( d_{ij} \) denotes the Euclidean distance between the particles at position \( i \) and \( j \). \( r_i \) and \( r_j \) denote the Vanderwaals radii of the ions at positions \( i \) and \( j \), respectively. The variable \( \alpha \) controls the strength of the penalty value.
The first penalty function $P_{count}$ was based on the number of ions that overlap, whereas the second penalty function $P_{distance}$ also included the degree of overlap. In contrast to the lethalization technique, the use of these penalty functions made it possible to discriminate between solutions with different degrees of constraint violations.

The offset of both functions was taken as 1.0 to prevent a division by zero when they were combined with the potential energy function (see below).

### 2.4. The fitness function

The repair, lethalization, and penalty functions were combined with the potential energy function $E_{pot}$, leading to a final (penalized) fitness value $F$ for each individual string. For the experiments in this paper five different fitness functions were defined:

1. **repair**
   
   \[ F = -E_{pot}; \]

2. **lethalization**
   
   \[ F = \begin{cases} 
   -E_{pot} & \text{if no constraint violations} \\
   0 & \text{otherwise} 
   \end{cases} \]

   where $\Theta$ denotes a large negative value;

3. **penalization**

   \[ F = \frac{-E_{pot}}{P_{count}} \]

   \[ F = \frac{-E_{pot}}{P_{distance}} \]

   \[ F = -\left[\max(E_{pot}) + P_{count}\right] \]

   This last combination of $E_{pot}$ and the penalty function $P_{count}$ ensured that the trial distributions in which the ions did not overlap always had a higher fitness value than distributions with overlapping ions (regardless of the size of the distributions), i.e., the valid distributions were mapped on top of the invalid distributions, and consequently this was called the mapping function. As a result, distributions without overlap between the ions were more likely to be selected for the next generation. This property was not ensured for the other penalty functions. Since the maximum value of $E_{pot}$ was unknown, $\max(E_{pot})$ was replaced with a large positive number $\Omega$, i.e., $F = -[\Omega + P_{count}]$.

### 2.5. The GA parameterization

For the experiments discussed in this paper, a population size of 75 strings was used. As discussed, rank-based threshold selection in combination with elitism was applied. The threshold fraction was set to 0.25 (19 strings), and the elitism fraction was set to 0.05 (4 strings). Uniform crossover was applied with a predefined probability of 0.80, and was parameterized to exchange 250 parameters on application. The mutation operator was applied with a probability of 0.005 if the penalty or lethalization functions were used. If the repair function was used this probability was decreased to 0.0005.
3. Results and discussion

Nine different experiments were performed to establish the performance of the constraint handling techniques when used in combination with the GA. Several characteristic values were tabulated, i.e., the degree of overlap between ions in a distribution (the penalty value $P$), the potential energy ($E_{pot}$), the number of Na$^+$ and Cl$^-$ ions in the interfacial volume, and the number of generations needed to derive the final solution. The results are depicted in Table 1.

From this table it is observed that distributions in which the ions overlapped showed a somewhat lower potential energy than distributions which were free of overlapping ions. The number of sodium and chloride ions deviated from the expected 36/36 ratio, and in most experiments more ions were included in the distribution.

The number of generations to derive at a solution was about equal for every experiment. However, the actual computer time needed showed large differences when the penalty functions were compared to the repair function; a GA run using the repair function typically required 2 weeks to converge, whereas the GA runs using one of the penalty or lethalization functions only required about 2 days.

The result obtained for the repair function (experiment 1) was not as expected. The potential energy was the highest found, and the number of ions largely deviated from the expected value. This distribution was, however, by definition free of ion overlap. In Fig. 3 the evolution of the number of sodium and chloride ions for this repair function is depicted. As a result of the initialization procedure, there was an excess of sodium ions at the beginning of the run, although the number of chloride ions was about the expected value. The repair function was not capable in reducing the number of sodium ions to the expected value, and even increased the number of chloride ions during the optimization process.

The repair function was designed such that it always completely filled the interfacial volume with ions (without overlap), without considering the ratio of Na$^+$ and Cl$^-$. The correct ratio of these ions should result during optimization from the combination of this repair function and the potential energy function. In this experiment a probability of $P_{Na} = 0.2$ ($P_{Cl} = 0.8$) was adopted to bias the ratio towards the expected number of Na$^+$ ions. However, even this small probability did not lead to an acceptable convergence.

From this result it could be concluded that the heuristic used by the repair function was obviously not appropriate. Possible adjustments to this function might require the inclusion of knowledge about the expected ion ratio, or releasing the condition of a completely filled interfacial volume. However, both modifications are not desirable because the exact final ion ratio was unknown, and an interfacial volume with gaps was physically not acceptable. This indicated that for the current problem, the design of an appropriate heuristic for the repair function was indeed difficult because a lack of knowledge about the simulated system led to unexpected results. From

<table>
<thead>
<tr>
<th>Exp.</th>
<th>Function</th>
<th>$\alpha$</th>
<th>$E_{pot}$</th>
<th>$P$</th>
<th>Na$^+$/Cl$^-$</th>
<th>No. of generations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Repair</td>
<td></td>
<td>-3.982</td>
<td>67/48</td>
<td>1804</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Lethalize</td>
<td></td>
<td>-14.39</td>
<td>41/38</td>
<td>2500</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$P_{count}$</td>
<td>0.05</td>
<td>-25.16</td>
<td>1.4</td>
<td>39/36</td>
<td>2203</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>0.25</td>
<td>-12.90</td>
<td>1.0</td>
<td>36/33</td>
<td>2173</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>10.0</td>
<td>-13.83</td>
<td>1.0</td>
<td>38/35</td>
<td>2475</td>
</tr>
<tr>
<td>6</td>
<td>$P_{distance}$</td>
<td>0.5</td>
<td>-15.20</td>
<td>1.05</td>
<td>42/39</td>
<td>2476</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>15.0</td>
<td>-14.69</td>
<td>1.0</td>
<td>41/38</td>
<td>2242</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>100.0</td>
<td>-12.97</td>
<td>1.0</td>
<td>36/33</td>
<td>2450</td>
</tr>
<tr>
<td>9</td>
<td>$-[max(E_{pot}) + P_{count}]$</td>
<td>100.0</td>
<td>-13.84</td>
<td>1.0</td>
<td>39/36</td>
<td>2427</td>
</tr>
</tbody>
</table>

Results of the experiments with the lethalization, repair and penalty functions. The value $\alpha$ denotes the strength of the penalty. $E_{pot}$ denotes the potential energy. $P$ reflects the degree of overlap between the ions in the distribution.
Fig. 3. Evolution of the number of Na\(^+\) and Cl\(^-\) ions for experiment 1 (repair function) with \(P_{Na} = 0.2\). The repair function is not capable of decreasing the excess of ions.

this it was clear that repair functions are not only computer intensive, and dedicated to a specific problem, but may even be very hard to design.

Experiment 2 (lethalization), showed that the final distribution was free of ion overlap, and that the potential energy was about the lowest found. The number of ions in this distribution was also the highest found (if the results of experiment 1 were not considered).

Experiment 3, 4 and 5 denote the results for the \(P_{count}\) penalty function with three different degrees of penalty strengths (\(\alpha\)). As expected, the final distribution was not free of ion overlap (\(P > 1.0\)), when the strength of the penalty value was too small (\(\alpha = 0.05\)). If the penalty strength was taken larger (\(\alpha = 0.25, 1.0\)), only distributions without overlap between the ions remained. The potential energies of these two experiments (4 and 5) were about the same, and it was observed that the distribution with the lower energy corresponded with a larger number of ions.

Experiments 6, 7 and 8 denote the results for the \(P_{distance}\) penalty function with three different chosen strengths. Again, a small penalty strength (\(\alpha = 0.5\)) resulted in violations of constraint \(C2\). The deviation between the potential energies of the valid distributions was somewhat larger than for the penalty function \(P_{count}\).

Fig. 4. Evolution of the number of Na\(^+\) and Cl\(^-\) ions for experiment 7 (\(P_{distance}\) with \(\alpha = 15.0\)).
Like experiments 4 and 5, the distribution with the lowest potential energy corresponded to a larger number of ions. For experiment 7, the evolution of the number of sodium and chloride ions is shown in Fig. 4, for the best trial solution generated during optimization. From this it was observed that this penalty function behaved in a completely different way from the repair function (Fig. 3). After initialization, the trial distribution contained 155 sodium and 29 chloride ions. During the next few generations this number of ions dropped significantly giving a more favorable distribution, i.e., a lower potential energy. After this decrease, additional ions were inserted, and redistributed such that a minimum in energy resulted.

This behaviour suggested an alternative heuristic for the repair function. Instead of starting from a completely filled grid, an initial empty grid might be preferable. This may give the mutation and crossover operators more opportunity to re-distribute the ions without rejection of the new distribution due to a constraint violation.

Finally, in experiment 9 the results of the mapping penalty function are given. No improvement to the other penalty or lethalization functions was observed.

The experiments conducted in this research were all started from identical initial populations, and consequently, the differences observed were only an effect of the differences between the constrained handling techniques. However, in order to decide whether the results of the penalty functions and the lethalization are significantly different, the GA runs should be replicated with different initial populations, whereafter a statistical analysis can be performed. This would reveal the best choice for this specific constrained optimization problem. However, this was not the aim of this research. In addition, it would not contribute to the explanation, or change the significance of the main outcome of the experiments, i.e., the good (unexpected) performance of lethalization. If the results would not be significantly different, it could be concluded that lethalization performs comparable to the other penalty functions. If, on the other hand, the results would be significantly different then lethalization still performs second best.

Assuming that the results are significantly different, inspection of the results for $P_{\text{distance}}$ shows that a high strength ($\alpha = 100.0$) leads to a decreased performance. This would suggest that both the strength and functional form of the penalty functional must be optimal in order to outperform lethalization. If the functional form would not be relevant, increasing the strength would bias $P_{\text{distance}}$ towards lethalization, and consequently, no decrease in performance should be expected. Therefore, the design of an appropriate penalty function is a difficult task because, a priori, no information is available about the optimal functional form and strength.

The results from experiment 2 to 9, thus, show that lethalization performs comparable to other penalty functions, or outperforms non-optimal penalty functions.

Since the solutions of a constrained optimization problem are often located at the boundary between the feasible and non-feasible region, the performance of search strategies that are based on incremental changes of a single trial solution (e.g., descent methods and simulated annealing) is increased if moves through the non-feasible region are allowed. For these strategies a poor performance is observed when lethalization is used, especially for heavily constrained problems, because many attempts to move through the search space are rejected (lethalized) due to a constraint violation. Consequently, the target solution may never be reached if it is shielded by non-feasible regions. Accordingly, these strategies require a more sophisticated penalty function.

The good results that were obtained with lethalization in the GA might be explained if the string fitnesses together with the violation of constraint $C_2$ (ion overlap) are monitored for $P_{\text{distance}}$ ($\alpha = 15.0$) and lethalization. At every generation the strings were sorted according to their potential energy, and the strings that violated constraint $C_2$ were marked. The result for $P_{\text{distance}}$ is shown in Fig. 5. Note that an increased string index corresponds to a lower potential energy (increased fitness). The threshold level (for threshold rank-based selection) is indicated by the horizontal line (19 strings appear above this threshold level). As is observed from this figure, in about 25 generations all distributions with ion overlap disappeared below the threshold and were no longer candidates to be selected in the next generations, i.e., after the selection step all trial distributions in the population were free of ion overlap. New ion overlap was only introduced by the crossover and mutation operators, but even after application of these operators the best 19 strings did not violate the constraint and were be further improved in the next generations. In fact, more than half of the population consisted of distributions in which
ions were not overlapping. Fig. 6 depicts the same information for the lethalization experiment, and showed a similar behaviour. Again, in a very few generations, the best trial solutions became free of ion overlap.

Both $P_{distance}$ and lethalization ‘push’ the invalid trial solutions below the threshold in a few generations, whereafter new solutions were only generated by re-combination and mutation of valid solutions. Once the invalid trial solutions are pushed below the threshold, there is no longer a differentiation in performance between lethalization and $P_{distance}$, i.e., only the first few generations determine the outcome of the experiment. However, this small number of generations is probably too short to result in pronounced differences between lethalization and the other penalty functions. This analysis showed that the properties of genetic based search (i.e., a population of trial solutions and a biased selection method) were responsible for the good performance of lethalization.
From this research it is fair to conclude that the use of a GA and lethalization may be a good choice for solving constrained optimization problems if the design of an optimal penalty function is difficult or impossible. However, additional research is necessary to investigate if these results are applicable to other optimization problems with different levels of constraining. Furthermore the effect of the alternative selection schemes should be investigated.

4. Conclusion

The design of a heuristic for a repair function is difficult if the characteristics of the problem are not fully understood. The use of a GA in combination with lethalization may be a good choice for solving constrained optimization problems when the design of an optimal penalty function is difficult or impossible. The good performance of lethalization may be regarded as an effect of the properties of genetic based search.

Acknowledgements

We would like to thank H.W. Mutsaers for his contributions to this research.

References