Advances in Soft Computing

Editor-in-chief
Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

Further volumes of this series
can be found on our homepage:
springeronline.com

Henrik Larsen, Janusz Kacprzyk, Sławomir Zadrozny, Troels Andreasen, Henning Christiansen (Eds.)
Flexible Query Answering Systems, 2000
ISBN 3-7908-1347-8

Robert John, Ralph Birkenhead (Eds.)
Developments in Soft Computing, 2001
ISBN 3-7908-1361-3

Frank Hoffmann, Mario Köppen, Frank Klawonn, Rajkumar Roy (Eds.)
Soft Computing: Methodologies and Applications, 2005
ISBN 3-540-25726-8

Bernd Reusch (Ed.)
Computational Intelligence, Theory and Applications, 2005
ISBN 3-540-22807-1
Preface

For the 8th time since 1991 we invite researchers to participate in the Dortmund Fuzzy-Days. I am very glad that our conference has established itself as an international forum for the discussion of new results in the field of Computational Intelligence. Again all papers had to undergo a thorough review: each one was judged by five referees to guarantee a solid quality of the programme.

From the beginning of the Fuzzy-Days on, Lotfi A. Zadeh felt associated with the conference. I would like to express my gratitude for his encouragement and support and I am particularly glad that he once again delivers a keynote speech. Much to my pleasure Janusz Kacprzyk, Jong-Hwan Kim, Enrique Trillas and Eyke Hüllermeyer have also agreed to present new results of their work as keynote speakers.

With Prof. Jong-Hwan Kim of the Korean Advanced Institute of Science and Technology (KAIST) a colleague takes part in this year’s Fuzzy-Days to whom I am indebted because he has inspired me and my colleagues to turn our attention towards the exciting field of Robot Soccer. Within a few years Robot Soccer has become an accepted test field for robotic research and the development of Multi Agent Systems. The importance of Computational Intelligence for these applications is evident. We address this topic not only with a dedicated session, but we will also organise a robot soccer match to demonstrate to the participants the entertaining quality of robot soccer as well as the challenging scientific problems.

I wish to thank all participants of the Dortmund Fuzzy-Days for their commitment to the conference and the organisers, namely Mrs Ulrike Lippe, for the excellent job they did. Last but not least, I am obliged to the German research council for their valuable financial support.

July 2004

Bernd Reusch
Contents

Evolutionary Algorithms

An Evolutionary Algorithm for the Unconstrained Binary Quadratic Problems
István Borgulya .......................................................... 3

Application of Genetic Algorithms by Means of Pseudo Gradient
Boriana Vatchova M. Sc. ............................................. 17

Optimization by Island-Structured Decentralized Particle Swarms
Juan F. Romero and Carlos Cotta .................................. 25

Directed Mutation by Means of the Skew-Normal Distribution
Stefan Berlik .............................................................. 35

Rule-Based Fuzzy Inference

Smooth Extensions of Fuzzy If-Then Rule Bases
Thomas Vetterlein ........................................................ 53

Pre-validation of a Fuzzy Model
Farida Benmakrouha .................................................. 61

Multiresolution Fuzzy Rule Systems
Ricardo Nanculef, Carlos Concha, Claudio Moraga, Héctor Allende...... 65
<table>
<thead>
<tr>
<th>Title</th>
<th>Authors</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fuzzy Clustering of Macroarray Data</td>
<td>Olga Georgieva, Frank Klawonn, and Elizabeth Härtig</td>
<td>83</td>
</tr>
<tr>
<td>Fuzzy Clustering: Consistency of Entropy Regularization</td>
<td>Hichem Sahbi and Nozha Boujemaa</td>
<td>95</td>
</tr>
<tr>
<td>Fuzzy Long Term Forecasting through Machine Learning and Symbolic Representations of Time Series</td>
<td>Bernard Hugueney, Bernadette Bouchon-Meunier, and Georges Hébrail</td>
<td>109</td>
</tr>
<tr>
<td>Fuzzy Prototypes Based on Typicality Degrees</td>
<td>Marie-Jeanne Lesot, Laure Mouillet, and Bernadette Bouchon-Meunier</td>
<td>125</td>
</tr>
<tr>
<td>Plenary Talk</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The Power of Zadeh’s Protoforms: Towards General Problem Formulations in Fuzzy Multistage Control and Group Decision Making</td>
<td>Janusz Kacprzyk</td>
<td>141</td>
</tr>
<tr>
<td>Fuzzy Control</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fuzzy Logic Fluid Therapy Control System for Renal Transplantation</td>
<td>A. Yardimci, N. Hadimioglu</td>
<td>143</td>
</tr>
<tr>
<td>Interpolative Fuzzy Reasoning in Behaviour-Based Control</td>
<td>Szilveszter Kovács</td>
<td>159</td>
</tr>
<tr>
<td>Fuzzy Modeling of Offensive Maneuvers in an Air-To-Air Combat Task</td>
<td>S. Akabari, M.B. Menhaj, S. K. Nikravesh</td>
<td>171</td>
</tr>
<tr>
<td>Invited Session: Recent Advances in Theoretical Soft Computing Session Organiser: Vílém Novák</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Approximation of Fuzzy Functions by Extended Fuzzy Transforms</td>
<td>Martin Štěpnička and Stephan Lehmke</td>
<td>187</td>
</tr>
<tr>
<td>Fuzzy Control as a General Interpolation Problem</td>
<td>Siegfried Gottwald</td>
<td>197</td>
</tr>
</tbody>
</table>
Galois Connections with Truth Stressers: Foundations for Formal Concept Analysis of Object-Attribute Data with Fuzzy Attributes
Radim Bělohlávek, Tatána Funioková, and Vilém Vychodil .......................... 205

Fuzzy Transforms in Removing Noise
IrinaPerfilieva and Radek Valášek .............................................. 221

Safe Modelling Of Fuzzy If–Then Rules
IrinaPerfilieva and Stephan Lehmke ............................................. 231

Perception-Based Logical Deduction
Vilém Novák ................................................................. 237

Invited Session: Towards Intelligent Decision Support Systems via Soft Computing
Session Organiser: Janusz Kacprzyk

Fuzzy Object-Oriented Modelling with Metadata Attributes in C♯
Fernando Berzal, Juan Carlos Cubero, Nicolás Marín, and Olga Pons .. 253

Strategies for Decision Making in the Conditions of Intuitionistic Fuzziness
Vassia Atanassova ................................................................. 263

Fuzzy Linguistic Summaries in Text Categorization for Human-Consistent Document-Driven Decision Support Systems
Janusz Kacprzyk and Sławomir Zadrozny ........................................ 271

An Application of Intuitionistic Fuzzy Relational Databases in Football Match Result Predictions
Boyan Kolev, Panagiotis Chountas, Ilias Petrounias, Vassilis Kodogiannis 281

Generalized Net Model for Adaptive Electronic Assessment, Using Intuitionistic Fuzzy Estimations
Anthony Shannon, Daniela Langova-Orozova, Evdokia Sotirova, Krassimir Atanassov, Pedro Melo-Pinto, Taekyun Kim ................. 291

Fuzzy Logic in Decision Support

Analytic Hierarchy Process Based on Fuzzy Analysis
Tomoe Entani, Kazutomi Sugihara and Hideo Tanaka ....................... 301
A Fuzzy-Ga Hybrid Technique for Optimization of Teaching Sequences Presented in ITSs
David Pritchard and Mircea Gh. Negoita ............................... 311

Consistency Conditions for Fuzzy Choice Functions
Irina Georgescu .......................................................... 317

Applications of Fuzzy Systems

A Fuzzy Logic Application to Environment Management System: A Case Study for Goksu Streams Water Quality Assessment
Ahmet Yardimci, A. Muhammetoglu, Hakan Oguz ..................... 327

Combination Rule of Normal Degrees on Automated Medical Diagnosis System (AMDS)
Yutaka Hata, Osamu Ishikawa, Syoji Kobashi and Katsuya Kondo ...... 339

Generation of Representative Symptoms Based on Fuzzy Concept Lattices
Tatiana Kiseliova and Stanislav Krajči ................................. 349

Connectives

On the Direct Decomposability of Fuzzy Connectives, Negations and Implications Based on T-Norms and T-Conorms on Product Lattices
Funda Karačal .............................................................. 357

The Cancellation Law for Addition of Fuzzy Intervals
Andrea Stupňanová ......................................................... 369

Generic View On Continuous T-Norms and T-Conorms
Irina Perfilieva ............................................................. 377

Invited Session: Intelligent Techniques for Knowledge Extraction and Management
Session Organiser: Ernesto Damiani

Mining Class Hierarchies from XML Data: Representation Techniques
Paolo Ceravolo and Ernesto Damiani ..................................... 385
Generalizing Quantification in Fuzzy Description Logics
Daniel Sánchez and Andrea G. B. Tettamanzi .......................... 397

Fuzzy Types: A First Step Towards Lazy Types in the .NET Framework
Fernando Berzal, Juan Carlos Cubero, Nicolás Marín, and M. Amparo Vila ................................................................. 413

Fuzzy Induction via Generalized Annotated Programs
Tomáš Horváth and Peter Vojtáš .............................................. 419

Evaluating Fuzzy Association Rules on XML Documents
Carlo Combi, Barbara Oliboni, and Rosalba Rossato ...................... 435

Plenary Talk
Ubiquitous Robot
Jong-Hwan Kim ................................................................. 451

Fuzzy Image Processing
Combining Fuzzy Logic and Kriging for Image Enhancement
Anwar Majeed Mirza and Badre Munir ...................................... 463

Optical Quality Control of Coated Steel Sheets Using Fuzzy Grey Scale Correlograms
Dirk Lühning ................................................................. 475

Plenary Talk
Fuzzy Methods in Knowledge Discovery
Eyke Hüllermeier ................................................................. 483

Invited Session Evolutionary Algorithms
Session Organiser: Carlos Cotta

Action Games: Evolutive Experiences
Antonio J. Fernández and Javier Jiménez González ...................... 487

Co-Evolving Multilayer Perceptrons Along Training Sets
M.G. Arenas, P.A. Castillo, G. Romero, F. Rateb and J.J. Merelo .... 503
Improving Parallel GA Performances by Means of Plagues
Francisco Fernández and Marco Tomassini .............................. 515

Hybrid Evolutionary Algorithms for Protein Structure Prediction
Carlos Cotta ........................................................................ 525

Invited Session: Aggregation Operators
Session Organiser: Radko Mesiar

Quasi-Copulas on Discrete Scales
Juliana Mordelová ............................................................... 537

Basic Classification of Aggregation Operators and Some Construction Methods
Radko Mesiar ..................................................................... 545

Homogeneous Aggregation Operators
Tatiana Rückschlossová ...................................................... 555

1-Lipschitz Aggregation Operators, Quasi-Copulas and Copulas with Given Opposite Diagonal
Erich Peter Klement and Anna Kolesárová .............................. 565

Fuzzy Measures and Choquet Integral on Discrete Spaces
Yasuo Narukawa, Vicenç Torra ............................................. 573

Neural Networks

Modular Neural Network Applied to Non-Stationary Time Series
Héctor Allende, Rodrigo Salas, Romina Torres, and Claudio Moraga .... 585

A Feedforward Neural Network Based on Multi-Valued Neurons
Igor Aizenberg, Claudio Moraga, and Dmitriy Paliy ..................... 599

Least-Squares Support Vector Machines for Scheduling Transmission in Wireless Networks
Jerzy Martyna ..................................................................... 613

Neural Networks for the Control of Soccer Robots
Man-Wook Han and Peter Kopacek ............................................ 621
Contents XIII

Neuro-Fuzzy Systems

Universal Approximator Employing Neo-Fuzzy Neurons
Vitaliy Kolodyazhniy, Yevgeniy Bodyanskiy, and Peter Otto ...........631

Combined Learning Algorithm for a Self-Organizing Map with Fuzzy Inference
Yevgeniy Bodyanskiy, Yevgen Gorshkov, Vitaliy Kolodyazhniy, Andreas Stephan ........................................................ 641

Fuzzy/Neural Connection Admission Controller for Multimedia Traffic in Wireless ATM Networks
Jerzy Martyna ............................................................ 651

Fuzzy Mathematics

Limits of Functional Sequences in the Concept of Nearness Relations
Martin Kalina ................................................................ 669

On the Law of Large Numbers on IFS Events
Beloslav Riečan ............................................................ 677

An Axiomatic Approach to Cardinalities of IF Sets
Pavol Kráľ ................................................................. 681

Fuzzy Optimization

Sensitivity Analysis for Fuzzy Shortest Path Problem
Tatiana Starostina and Stephan Dempe .............................. 695

Fuzzy Coloring of Fuzzy Hypergraph
Leonid Bershtein, Alexander Bozhenyuk, and Igor Rozenberg ........ 703

Nonlinear Optimization with Fuzzy Constraints by Multi-Objective Evolutionary Algorithms
F. Jiménez, G. Sánchez, J.M. Cadenas, A.F. Gómez-Skarmeta, J.L. Verdegay ................................................................. 713

Poster Contributions

Comparison of Reasoning for Fuzzy Control
Bohdan Butkiewicz ...................................................... 725
Evolving Scientific Knowledge
Jürgen Paetz .................................................... 733

Coding of Chaotic Orbits with Recurrent Fuzzy Systems
Alexander Sokolov, Michael Wagenknecht ......................... 739

Genetic-Based Tuning of Fuzzy Dempster-Shafer Model
Z.A. Sosnowski and J.S. Walijewski ................................ 747

A Novel Design for Classifying Multi-Field Internet Packets
Using Neural Networks
Modjtaba Rouhani, M.J. Solaymanpour, and Naser Nemathakhsh .... 757

Modeling Uncertainty in Decision Support Systems for
Customer Call Center
Alexander Holland .................................................. 763

A New GA-Based Real Time Controller for the Classical
Cart-Pole Balancing Problem
N. Seifipour and M. B. Menhaj ..................................... 771

Depth Control of Anesthesia with an Adaptive Neuro-Fuzzy
System
A. Yardimci, N. Hadimioglu, Z. Bigat, S. Ozen ...................... 787

Ultrasound Intensity and Treatment Time Fuzzy Logic
Control System for Low Cost Effective Ultrasound Therapy
Devices
Ahmet Yardimci and O. Celik ..................................... 797

Author Index ....................................................... 809
Session

Evolutionary Algorithms
An Evolutionary Algorithm for the Unconstrained Binary Quadratic Problems

István Borgulya

University of Pécs, Faculty of Business and Economics, H-7621 Pécs, Rákóczi út 80, Hungary.
borgulya@ktk.pte.hu

Abstract. In this paper a new evolutionary algorithm (EA) is described for the unconstrained Binary Quadratic Problem, which is to be used with small, medium and large scale problems as well. This method can be divided into two stages, where each stage is a steady-state EA. The first stage improves the quality of the initial population. The second stage uses concatenated, complex neighbourhood structures for the mutations and improves the quality of the solutions with a randomized k-opt local search procedure. The bit selection by mutation is based on an explicit collective memory (EC-memory) that is a modification of the flee-mutation operator (Sebag et al. 1997). We tested our algorithm on all the benchmark problems of the OR-Library. Comparing the results with other heuristic methods, we can conclude that our algorithm belongs to the best methods of this problem scope.

Keywords: Binary quadratic programming; Large-size problems; Evolutionary algorithm.

1 Introduction

The general formulation the unconstrained binary quadratic programming problem (BQP) is the following:

$$\text{Max } f(x) = x^T Q x + c^T x$$

where $$x \in \{0, 1\}^n$$, $$Q \in \mathbb{R}^{n \times n}$$ is an $$n \times n$$ symmetric rational matrix.

BQP has a central role in combinatorial optimization. A large number of problems can be formulated as maximization of quadratic real values function in 0-1 variables. For that reason, BQP has been referred to as “the mother of all the combinatorial problems” (Bjorndal et al. 1995). E.g. BQP is equivalent to many classical combinatorial optimization problems such as maximum cut, maximum clique, maximum vertex packing, and maximum vertex cover. As important application we can see e.g. machine scheduling, capital budgeting, financial analysis and molecular conformation problems.
The techniques which can be used to find the exact optimal solution are the branch and bound and branch and cut methods (e.g. (Helmberg and Rendl 1998), (Horst et al. 2000), Pardalos and Rodgers 1990)). We can also use linearization techniques as well (e.g. the reformulation-linearization technique (Sherali and Adams 1998)), which converts nonlinear mixed-integer or zero-one problems into linear ones. This technique can be used not only to construct exact solution algorithms, but also to design powerful heuristic procedures.

Generally problems of sizes larger then n=100 cannot be solved in an acceptable time. Since most real-world problems are large-size problems, heuristics are used to find good solutions within a reasonable time (Glover et al. 2002). A large number of heuristic methods have been developed for solving the BQP. Various heuristic methods are also frequently used, such as one-pass heuristics (Glover et al. 2002), simulated annealing (SA) (Beasley 1999), (Katayama and Narihisa 2001), tabu search (TA) (Beasley 1999), (Glover et al. 1998), evolutionary algorithm (EA) and versions (e.g. genetic algorithm (GA), evolutionary strategy (ES)) (Lodi et al. 1999), (Merz and Freisleben 1999), scatter search (Glover 1997), memetic algorithms (MA) (Merz and Katayama 2001), as well as differently iterated search techniques (e. g. the parthenogenetic algorithms (PA) (Katayama and Narihisa 2001) or various subgradient-type methods (Shor 1998).

In this paper, we present a new heuristic method to solve the BQP. This heuristic is an EA that consists of 2 consequent stages. The first stage improves the quality of the initial population. The second stage uses concatenated, complex neighbourhood structures for the mutations, improves the quality of the solutions with a randomized \( k\)-opt local search procedure and uses a special filter and restart technique. The bit selection by mutation is based on an explicit collective memory (EC-memory) that is a modification of the flee-mutation operator (Sebag et al. 1997). The efficacy of the method was studied on the OR-Library benchmarks: the small, medium and large scale problems were all successfully solved. Comparing the results to others methods, we can state that our algorithm belongs to the best heuristic’s methods of this problem scope.

In section 2, we describe our EA in general. Section 3 includes implementation details of our EAs. In Section 4, we present our computational experience, and we compare our results with other heuristic’s results. Section 5 contains concluding remarks.

2 The principle of the new evolutionary algorithm

2.1 The structure of the algorithm

Hybrid EAs are frequently used for solving combinatorial problems. These methods improve the quality of the descendent solution for example with the application of a local search procedure, SA, or TS. The constitution of these
systems corresponds to an extension of an EA: for instance a local search procedure is applied at every step of the EA cycle. The new EA unlike former hybrid EAs based on a single stage, uses a 2-stage algorithm structure in order to speed up convergence and to produce higher quality results. The first stage is a quick “preparatory” stage that is designated to improve the quality of the initial population. The second stage is a hybrid EA with some special operators.

Let us discuss the 2 EAs (stages) in more detail:

1. The first stage forms some solutions at random and then tries to improve them by randomly generating descendents. The descendent may replace the most similar one of the former solutions.
2. The second stage is a hybrid ES. The algorithm uses two different recombination operations, and concatenated, complex neighbourhood structures for the mutations. The recombination operation is a uniform or single-point recombination or otherwise simple copy-making. In selecting the parents, priority is given to the best, highest objective/fitness function value: the algorithm selects the fittest solution with 0.5 probability and another solution with $0.5/t$ probability (where $t$ is the size of the population). By mutation we applied varying number of bit-flip and a special bit-flip (bit-flip-flop). We form the neighbourhood structure using: some bit-flip-flops + some bit-flips. The quality of the solutions is improved with a local search procedure. We applied the randomized $k$-opt local search (Merz and Katayama 2001). Finally in order to keep the diversity of the population we use a filter and a restart procedure. The filter selects only the best of the solutions close to each other, the other ones are deleted. The restart begins the second stage again, if the fittest solution didn’t change in the last generations. It replaces the weakest solutions with new ones (70% of the population), and it applies the local search procedure on a part of the new individuals.

2.2 EC-memory

There are many variants of EC-memory methods that memorises the past events and/or past successes of the evolution process. We choose to adopt the method of (Sebag et al. 1997) that memorises the past failures of evolution through a virtual individual, the virtual loser. Let us see the principle of this method.

Let X, Y, Z and T be individuals, where X has a high fitness, and Y, Z and T all have a low fitness. Let the average of Y, Z and T be noted VL for virtual loser. The probability of mutating bit $i$ in individual X should reflect how much it discriminates $X_i$ from VL, that is, it should increase with $1 - |VL_i - X_i|$. 
The EC memory is updated by relaxation from a fraction of the worse individuals in the current population. If $\alpha$ denotes some relaxation factor, $it$ is the number of generation and $dVL$ is the average of these worse individuals,

$$VL^{it+1} = (1-\alpha) \cdot VL^{it} + \alpha \cdot dVL$$  (e.g. $\alpha=0.2$).

(Sebag et al. 1997) use a flee-mutation operator based on $VL$. In this mutation operator each bit to mutate is selected by tournament and the probability of mutating of one bit depends on both the bit itself and the individual at hand.

In our algorithm we use the virtual loser too. We don’t apply the flee-mutation operator, but in our mutation structure we use the virtual loser by bit selection. We use two variants of the bit selection:

- The bit to mutate is selected randomly and its probability of mutating depends on the quantity $p_i.(p_i = 1 - |VL_i - X_i|.)$
- Only the bits with the highest $p_i$ values are selected and their probability of mutating depends on the quantity $p_i$.

Applied to the same problem, different bit selection mutations generally succeed with different efficacies. Even though their concurrent random application might increase computational cost with some of the problems, on average it increases efficacy. Therefore our algorithm randomly selects between the two types of bit selections for each mutation.

The new algorithm constructed this way, named EBQP (Evolutionary algorithm for the BQP) may be used both for small, medium and large scale BQP. Its structure and function are different from the former methods used to solve the BQP, as it uses an evolutionary framework, and it applies a special transformation for the mutations.

The EBQP got its final form after a number of assays. We tested the 1-opt and k-opt local search procedures, and tested the mutation operation with and without the EC-memory. The test results made us conclude that the usage of the randomized k-opt local search yields 10 times better results (more accurate) than the 1-opt local search. With the new mutation operator based on EC-memory the efficiency of the algorithm proved to be better than without EC-memory: on the benchmark set the quality of the solutions was improved by 6-7 %, and the computational cost (measuring the average running time) was decreased by 25 %.

3 The new algorithm

3.1 The characteristics of the EAs

The main functions and characteristics of the EAs are the following:
Initial population. The same population and individuals are used in all stages. The first individuals of the P population are randomly generated from S. These are the first “solutions”.

Fitness function. The algorithm uses the objective function \( f(x) \) as fitness function.

Selection operator. In the first stage descendents are randomly selected from S, without the application of any further operators (recombination, mutation). In the second stage the algorithm selects two different parents from the population: the first of them is the most appropriate solution with 0.5 probabilities.

Recombination operator. In the second stage the algorithm chooses from two options:

- With 0.8 probability it chooses the two parents from the mating pool and the descendent is constructed with uniform or single-point recombination.
- With 0.2 probability it chooses the best parent from the mating pool and the descendent is built by copy-making.

Mutation operator. In the second stage the algorithm has two options:

- After discrete or single-point recombination it applies the mutation operator with \( p_{rm} \) probabilities. The value of \( p_{rm} \) is adjusted by the user, and it is low by easy and high by difficult problems (e.g. \( p_{rm} \) is 0.05 or 0.95)
- After copy-making it applies the mutation operator.

The applied transformations are the following: bit-flip a single bit (variable) is flipped randomly in the descendent; bit-flip-flop if there are two randomly chosen bits, the \( i \)th and \( j \)th variables in the descendent having different values, the bits \( i \) and \( j \) will be flipped.

The bit-flip is to be considered as the random succession of sequential flips. The path of the flips follows the hypotenuses of the hypercube of \( n \) dimensions, and it gets from one vertex to another one step by step through the neighbouring vertexes. At each step the Hamming distance between the vertexes is 1. The bit-flip-flop is a special variant of the bit-flip transformation. Its path follows the diagonal lines of the hypercube, and instead of the neighbouring vertexes it flips to the transversal points. At each step the Hamming distance between the vertexes is 2.

The complex, multiple transformation is as follows:

\[
\text{some bit} - \text{flip} - \text{flops} + \text{some bit} - \text{flips}
\]

where the bit-flip-flop transformation is executed only with the probability of 0.5, and the bit-flip and bit-flip-flop transformations are executed varying times.

Our algorithm applies the two variants of bit selection with 0.5-0.5 probability. Therefore a bit-flip-flop transformation is executed when the probability of mutation for the first selected variable is higher than \( p_i \) (\( p_i = 1 - (VL_i - X_i) \)). At the bit-flip transformation, the chosen variable will also be flipped by the probability of \( p_i \).
Begin
calculate gains $g_i$ for all $i$ in $\{1, \ldots, n\}$
repeat
\[x_{\text{prev}} = x, \ G_{\text{max}} = 0, \ G = 0, \ C = \{1, \ldots, n\}\]
repeat
Generate a random permutation $RP[\cdot]$ of the set $\{1, \ldots, n\}$
For $j = 1$ to $n$
\[k = RP[j]\]
if $g_k > 0$ then
\[G + g_k, \ G_{\text{max}} = g, \ x_k = 1 - x_k, \ x_{\text{best}} = x, \ \text{update gains } g_i \text{ for all } i\]
\[C = C \cup \{k\}\]
fi
find $j$ with $g_j = \max_{i \in C} g_i$
\[G = G + g_j\]
If $G > G_{\text{max}}$ then $G_{\text{max}} = G, \ x_{\text{best}} = x$ fi
\[x_j = 1 - x_j, \ C = C \setminus \{j\}, \ \text{update gains } g_i \text{ for all } i\]
until $C = \emptyset$
if $G_{\text{max}} > 0$ then $x = x_{\text{best}}$ else $x = x_{\text{prev}}$ fi
until $G_{\text{max}} \leq 0$
end

Fig. 1. The randomized $k$-opt-local-search algorithm (Merz and Katayama 2001)

The virtual loser is defined after the termination of the first stage. It is periodically updated by using the weakest individuals. In the updating procedure we use 20% of the population.

Local-search. In the EBQP we applied the randomized $k$-opt-local-search algorithm (Merz and Freisleben 2002), (Merz and Katayama 2001). The algorithm is shown in Figure 1, $x$ is a given solution vector of length $n$ and $g$ is a gain vector of length $n$, that is stored with $x$. The $k$th gain of $g$ denotes the cost of the neighbour solution, when a single bit $k$ is flipped in the current solution:

\[g_k = q_{kk} (\bar{x}k - xk) + \sum_{i=1, i \neq k}^{n} q_{ik} (\bar{x}k - xk)\]

where $\bar{x}k = 1 - xk$

Moreover, $g$ is efficiently updated each time a bit is flipped. The gains $g_i$ do not have to be recalculated each time. Instead, it is sufficient to calculate the difference of the gains $\Delta g_i$:
\[ g_i' = \begin{cases} 
-g_i & \text{if } i = k \\
g_i + \Delta g_i(k) & \text{otherwise} 
\end{cases} 
\]

with \( \Delta g_i(k) = 2q_{ik}(\bar{x}_i - x_i)(x_k - \bar{x}_k) \)

**Reinsertion.** In the first stage, the algorithm compares the descendent with the most similar solution. If the descendent is better then the former solution, it is replaced by the descendent. In the second stage the algorithm compares the descendent with the best parent from the mating pool, or if the number of the individuals is less then the population size, it increases the number of the individuals (after restart).

**Filtering.** We added a check to the EBQP (in every \( kn \)th generation). In order to keep the diversity of the population we select only the best of the solutions close to each other, the other ones are deleted \((x\text{ and }x'\text{ are close to each other if the } d^H(x, x') \text{ is less than a predefined value, e.g. } n/4)\). This filtering speeds up the convergence, too. The new individuals substituting the deleted ones are generated from the old individuals with 10 bit-flip mutation.

**Restart strategy.** If no new best individual in the population was found for more than 20 generations, the EBQP begins the second stage with an other population. The individuals excepting the best 30% of the population are deleted, and new individuals are generated. The first \( kn \) individuals are generated in the same way as the descendents and one individual is generated in a generation. The rest of the new individuals are generated from the old individuals with 10 bit-flip mutations in the next \( kn \)th generation. So the population size will be full again at the time of the next check (at the next \( kn \)th generation).

**Stopping criteria.** The algorithm is terminated if the running time (in CPU seconds) is more than a prefixed time limit.

### 3.2 The main steps of the algorithm

Let us introduce the following notations:

- Let us denote the 2 stages as EA1 and EA2.
- Let the population of the \( it \)th generation be denoted by \( P(it) \), and let \( x_1, x_2, \ldots, x_t \) be the individuals of the \( P(it) \) population.
- Let us denote the complex neighbourhood transformation based on the virtual loser \((\text{some bit-flip-flops+some bit-flips})\) of the \( q \) descendent by \( Nhs(q) \).
- The two transformations are executed randomly, maximum \( \min(n/2, 50) \) times. The probability of applying the first transformation is 0.5.
- The measure of the similarity of the two solutions \( x \) and \( z \) is given by \( H(x, z) = 1/(1 + d^H(x, z)) \) where \( d^H(x, z) \) is the Hamming distance of the solutions.
- Let us denote the procedure, which filters out and deletes the close solutions \((d^H(x, x') < n/4)\) by Filter. At the end, the procedure generates new solutions to replace the deleted ones.
Parameters

6 parameters affect the run of the algorithm:

- $t$ – the size of the population.
- $itt$ – a parameter of the second stage. If the number of iterations ($it$) reaches $itt$, the second stage begins.
- $kn$ – a parameter which determines the timing of checks. We use the Filter and Restart procedure only at every $kn$th iteration.
- $m$ – a parameter of the $k$-opt local search. The outer repeat loop can run only $m$ times.
- $p_{rm}$ – a parameter of the second stage. The probability of the mutation operator after recombination.
- $timelimit$ – a parameter for the stopping condition. The procedure is finished if the running time (in CPU seconds) is more than $timelimit$.

Variables:
- $it$ – the number of the iterations.

Procedure EBQP($t$, $itt$, $kn$, $p_{rm}$, $timelimit$, opt, optp)

\begin{verbatim}
  it:=0
  Let $x_i \in \{0,1\}^n$ (i=1,...,t), $P(it) \leftarrow \{x_1 , \ldots , x_t \}$.
  Compute $f(x_1), \ldots , f(x_t)$.

  Do $itt$ times
    Selection of a $q \in S$.
    Compute $f(q)$.
    Let $H(q,x_z):=\max_j H(q,x_j); j, z \in \{1,2,\ldots,t\}$

    If $f(q) > f(x_z)$ then $x_z := q$ fi.
    $it:=it+1$, $P(it) \leftarrow P(it-1)$.
  od.

  Initial value of VL
  /* second stage:
  Repeat
  Do $kn$ times
    Two parents $x_i, x_j$ selection
    Generation of the $q$ descendant.
    $Nhs(q)$. Local search.
    Compute $f(q)$.
    If $|P(it)|<t$ then $P(it)= P(it) \cup q$
    else
      if $f(q) > f(x_i)$ then $x_i := q$ fi.
    fi
    $it:=it+1$, $P(it) \leftarrow P(it-1)$.
  od.
\end{verbatim}
Update of VL
Filter, Restart.
until “running time” > timelimit.
optp = the best x individual, opt = f(optp)
end

3.3 Details of the implementation

When describing the algorithm, some heuristic solutions and the shortcuts of the computations were not described. Let us see them now one-by-one.

Speeding-up the computation.

The algorithm speeds up calculating the value of the objective function values faster. The base of the reduction of the computing time is the possibility, that the recombination and the mutation operators change only a certain fragment of the variables. The descendents are produced by some consequential bit transformations from one of the parents. Therefore it is enough to know the \( f(x) \) function value of the \( x \) parent only, and the \( f(x') \) function value of the \( x' \) descendent could be constructed from the former one by bit transformations. In summary, the algorithm calculates the \( f(x') \) function values from \( f(x) \) by using the \( g_k \) gain values of consequential bit transformations.

Speeding-up the local search.

To reduce the running time of the randomized \( k\)-opt local search (Figure 1), the terminal condition of the inner repeat loop can be modified so that the loop is terminated if there were no new \( x_{\text{best}} \) values for more than 50 iterations and the outer repeat loop can repeat only 1 or 5 times (the \( m \) value is 1 by easy and 5 by difficult problems).

Heuristic in the mutation

The EBQP executes the first element of the \( \text{bit-flip-flop} + \text{bit-flip} \) with the given probability, and the two transformations are repeated by a mutable number of times. After testing numerous possible alternatives, using the test problems from the OR-Library, we found \( \text{bit-flip-flop} \) transformation to be the most efficient at 0.5 probabilities. Defining the number of the transformations arouses a more complex problem. To demonstrate its complexity we can choose a parthenogenetic algorithm (Katayama and Narihisa 2001) as an example. This algorithm uses a parameter dependent on the task for the maximal number of bit transformations in a mutation. We found that the best results were attained at different parameter values at different task groups, with the algorithm proving most efficient when the maximum number was between 40 and 80.
We did not wish to choose the maximal number of the transformations as a parameter. Instead, we searched for a maximal number giving a result of average acceptability, but applicable at each task. After testing several options, two and fifty proved to be as the most apt number of maximal bit transformations. At the 5-10 element problem groups glov500, B1000 and B2500 of the OR-Library the average relative percentage deviation of the solution from the best known solution was individually calculated (The test problems are detailed in Section 4). Subsequently, the results were averaged for both the groups and individual tasks. Based on the comparative tests, we found 50 bit transformations to be the most apt.

4 Solutions of test problems

Test problems

We tested the EBQP with the benchmark set of the OR-Library (http://www.ms.ac.uk/info.html). Although we managed to solve each individual BQP problem in the OR-Library, we wish to show only three benchmark sets. This are two sets from Beaslay with 1000 and 2500 dimensions and the set from (Glover et al. 1998) with 500 dimension (notation: B1000, B2500, glov500). In each sets there are 5 or 10 instances.

Parameter selection

To achieve a quick and accurate solution we need appropriate parameter values. Studying some of the more complex problems of the benchmark set we analyzed the process of the convergence how the parameter values were affecting the convergence, the finding of the global optimum and the speed of the calculation.

So we analyzed the size of the population which should be used in EBQP to achieve a good trade-off between the probability of finding best-known solutions and the necessary run-time to do so. In general, best behaviour was obtained with a size of population \((t)\) between 30 and 60. The size of the population of 30 was found appropriate for the small - medium \((\leq 1000)\) and the size of the population of 60 was found appropriate for larger \((>1000)\) number of dimensions.

As a next step, we studied the frequency of checks \((kn\) parameter) and the \(p_{rm}\) probabilities of the mutation. At the time of the check several functions are executed. This means running the VL update, the Filter and eventually the Restart procedure. It is inexpedient to run them often, because then the VL looses efficacy and we waste too much time on filtering. We found that the frequency of the checks is optimal around every 10 generations \((kn=10)\). There were many possible values found for the \(p_{rm}\) probabilities. We found, that for the half of the test problems, namely the easier tasks \(p_{rm} = 0\) was
suitable. In the other half of the problems, especially the difficult and 2500 dimension problems $p_{rm} = 1$ was more appropriate. We can further refine parameter settings with the $m$ parameter of the $k$-opt local search. The $m$ value can be chosen 1 by easy and 5 by difficult problems.

Finally, we studied the number of iterations of the first stage. We found the first stage to be necessary, because improving the first solutions is beneficial to the quality of the results and usually decreases the scatter of the result. However, 30 iterations are enough for improving the initial population.

**Computation results**

We tested the algorithm on the test problems from the OR-Library. The algorithm managed to find solutions which are either best known or within 0.03 % of the best known solutions for 100% of the test problems. Table 1 includes the results of the chosen test-problems. The table presents mean values calculated from a number of runs: all problems (tasks) were run 10 times. We used a predefined time limit of 60 seconds for each instance of the set glov500, 300 seconds for each of B1000 and 2000 seconds for each of B2500 (The EBQP was implemented in Visual Basic and ran on a Pentium 4 1.8 GHz with 256 MB RAM).

The Table 1 and 2 show the average solution values for large scale problems. In Table 1 we give the problem name, the task number in the problem group, the best known solution, the average relative percentage deviation of the solution from the best known solution (AD). The Table 2 shows the average running time in seconds to the best solutions.

**Table 1.** Average solution values for the large scale problems

<table>
<thead>
<tr>
<th>Task</th>
<th>Glov500</th>
<th>B1000</th>
<th>B2500</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Best known</td>
<td>AD</td>
<td>Best known</td>
</tr>
<tr>
<td>1</td>
<td>61194</td>
<td>0</td>
<td>371438</td>
</tr>
<tr>
<td>2</td>
<td>100161</td>
<td>0</td>
<td>354932</td>
</tr>
<tr>
<td>3</td>
<td>138135</td>
<td>0</td>
<td>371236</td>
</tr>
<tr>
<td>4</td>
<td>172771</td>
<td>0</td>
<td>370675</td>
</tr>
<tr>
<td>5</td>
<td>190507</td>
<td>0</td>
<td>352760</td>
</tr>
<tr>
<td>6</td>
<td>359629</td>
<td>0</td>
<td>359629</td>
</tr>
<tr>
<td>7</td>
<td>371193</td>
<td>0</td>
<td>371193</td>
</tr>
<tr>
<td>8</td>
<td>351994</td>
<td>0.0042</td>
<td>1484199</td>
</tr>
<tr>
<td>9</td>
<td>349337</td>
<td>0.0011</td>
<td>1482413</td>
</tr>
<tr>
<td>10</td>
<td>351415</td>
<td>0</td>
<td>1483355</td>
</tr>
</tbody>
</table>

**Comparative results**

As for comparison, we chose different methods: the TS and SA heuristics by Beasley (TS_B, SA_B) [1], the SA heuristic by Katayama and Narihisa
Table 2. Average running time for the large scale problems

<table>
<thead>
<tr>
<th>task</th>
<th>Glov500</th>
<th>B1000</th>
<th>B2500</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>27.6</td>
<td>62</td>
<td>821</td>
</tr>
<tr>
<td>2</td>
<td>14.0</td>
<td>22</td>
<td>1825</td>
</tr>
<tr>
<td>3</td>
<td>26.3</td>
<td>84</td>
<td>697</td>
</tr>
<tr>
<td>4</td>
<td>37.5</td>
<td>170</td>
<td>634</td>
</tr>
<tr>
<td>5</td>
<td>10.5</td>
<td>152</td>
<td>945</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>95</td>
<td>1509</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>115</td>
<td>1612</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>210</td>
<td>844</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>241</td>
<td>1254</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>151</td>
<td>1288</td>
</tr>
</tbody>
</table>

(SA_KN), the genetic local search by Merz and Freisleben (GLS_MF), the parthenogenetic algorithm by Katayama and Narihisa (PA), and the memetic algorithm by Merz and Katayama (MA).

The comparison was encumbered by the use of various programming languages, operating systems and computers. Only one appropriate aspect of comparison could be found, namely the average relative percentage deviation of the solution from the best known solution, so our table of comparison (Table 3) is based on the results of comparable accuracies. (The TA_B and SA_B program ran on Silicon Graphics, R4000 CPU with 100 MHz. The GLS_MF ran on Pentium II PC, 300MHz. The SA_KN algorithm was implemented in C and ran on a Sun Ultra 5/10, UltraSPARC-IIi 440 MHz under OS Solaris 7. The PA and MA programs ran on Sun a Ultra 5/10 UltraSPARC-IIi 440MHz, too).

We compared the average results. Analyzing the results of Table 3 we can confirm that the MA algorithm has the best, and the PA has the second best results in general. In case of the B2500 problems the PA has the best, and at the B1000 problems the MA has the best results. The EBQP has also good results: his result is the second at the B1000 problems, and the third at the B2500 problems.

Regarding that the MA and PA are the most effective methods of BQP; we can conclude that the EBQP also belongs to the best available methods. The results of the EBQP we can improve probably. Chosen a more appropriate recombination operator, or other local search procedure, probably we can improve the quality of the solution, and we can reduce the running time. The choice of the \( p_{rm} \) probability and the \( m \) parameter of the \( k\)-\( opt \) local search we can automate too. With the help of competing subpopulations we can use different \( p_{rm} \) and \( m \) values parallel, and the algorithm searches the most appropriate values.
Table 3. Comparative results for large scale problems

<table>
<thead>
<tr>
<th>Method</th>
<th>Glov500</th>
<th>B1000</th>
<th>B2500</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD_B (best) (Beasley 1999)</td>
<td>0.0720</td>
<td>0.0860</td>
<td>0.0640</td>
</tr>
<tr>
<td>TS_B (best) (Beasley 1999)</td>
<td>0.0020</td>
<td>0.0370</td>
<td>0.0920</td>
</tr>
<tr>
<td>GLS_MF (Merz and Freisleben 1999)</td>
<td>-</td>
<td>0.0290</td>
<td>0.0610</td>
</tr>
<tr>
<td>SA_KN (Katayama and Narihisa 2001)</td>
<td>0.0220</td>
<td>0.0290</td>
<td>0.0210</td>
</tr>
<tr>
<td>EBQP</td>
<td>0</td>
<td>0.0013</td>
<td>0.0051</td>
</tr>
<tr>
<td>PA (Katayama and Narihisa 2001a)</td>
<td>0</td>
<td>0.0022</td>
<td>0.0002</td>
</tr>
<tr>
<td>MA (Merz and Katayama 2001)</td>
<td>0</td>
<td>0.0000</td>
<td>0.0004</td>
</tr>
</tbody>
</table>

5 Summary

In this paper we presented a new heuristic algorithm, named EBQP for solving the BQP problem. The concept of the EBQP differs from the previous ones: the structure of the new algorithm is an evolutionary framework consisting of 2 stages, for the solution of BQP we introduce a problem-specific mutation operator and we use a \( k-opt \) local search procedure. So we use a complex neighbourhood structure for the mutations, where we can concatenate different neighbourhoods, and the bit selection by mutation is based on an explicit collective memory (EC-memory). The progression of the algorithm is influenced by 6 parameters, more of which have the same value for the different problems.

We can conclude that the EBQP was successfully tested with different kinds of BQP. The algorithm managed to find solutions which are either best-known or within 0.03% of the best known solutions for 100% of the benchmark problems of the OR-Library. Comparing the results with other heuristic methods, we can conclude that the EBQP belongs to the best evolutionary algorithms of this problem scope.

Acknowledgments

The Hungarian Research Foundation OTKA T 042448 and the ERFP-DD2002-HU-S-01 project supported the study.

References

Application of Genetic Algorithms by Means of Pseudo Gradient

Boriana Vatchova M. Sc.

str. bl. 2, 1113 Sofia, Bulgaria

Keywords: genetic algorithms, populations, chromosome, fitness function, crossover operator, mutation and selection, pseudo-gradient.

Introduction

Genetic algorithms (GA) and their capabilities for solving optimization problems have been thoroughly investigated recently in a number of application areas. These algorithms represent a class of stochastic algorithms based on methods of biological evolution. The chromosome is one of the smallest particles in the kernel of every cell and it defines the genetics of living organisms. Nature has the ability of finding suitable chromosomes through natural selection. Analogically, genetic algorithms can find optimal solutions by means of rational computational iterations [1, 2, 3, 4].

The iterations associated with genetic algorithms can be represented through a series of populations of arguments of the optimization task. The latter are called “computer implemented chromosomes” or only “chromosomes”. Every new population of chromosomes (values of arguments of the optimization task) can be derived analogically to biological procedures in nature, known as crossover, mutation and selection.

For the first two procedures, there aren’t routine methods for a reasonable choice. In this case, methods for random search are applied and selection is based on an operation for comparing the values of the fitness functions for the separate chromosomes in populations. Besides this, genetic algorithms usually remove chromosomes with lower values of their fitness function. [5]

This is a drawback of these algorithms because the removal of such chromosomes implies some loss of information received so far.

The purpose of this paper is to keep this kind of information for a limited number of iterations, as it is the case in nature.
For the purpose of more objective assessment of the new chromosomes, the information received from previous generations of chromosomes, including the ones with lower values of their fitness function has to be kept. When hereditary information is used successfully from previous generations of chromosomes, it is normal to expect an acceleration of the speed of genetic algorithms due to a reduction of the number of iterations.

1 Essence of the Method

In most genetic algorithms, the initial population is generated through random selection. After a limited number of initial iterations based on crossover and mutation operators, an assessment is done for the less perspective sub-domains of definition area and their rapid discharge from chromosomes. These chromosomes are replaced by chromosomes in the more perspective sub-domains. It is possible to generate logical rule in the algorithm which will determine the future concentration of chromosomes in the perspective sub-domains by calculating the values of the pseudo gradient of optimization function. It is assumed in this case that the chromosomes in the next populations will be concentrated in sub-domains of all local optimums. In this way, an additional genetic algorithm must ensure that the chromosomes leave the sub-domains in which local optimums are not perspective candidates for a global optimum.

2 Pseudo-Gradient Function

Every population of chromosomes in genetic algorithms is based on a random choice which includes crossover probability $P_C$ and mutation probability $P_M$.

This peculiarity is taken from nature but the probability for a reasonable choice is not very high, especially in the case of mutation. This implies a low speed of movement of chromosomes to the optimum of the function.

The choice of every next generation of chromosomes has to be made wisely. This can be done by means of an additional operator which can direct chromosomes very fast to a local, and after that, to the global optimum. This operator will calculate the gradient of the function on the basis of available information about previous populations.

Given the gradient of a function $\nabla f(x,y,z,\ldots)$, after a certain number of populations, e.g. 3 or 4, the next population is regenerated from the following pseudo grad algorithm:

$$
x_{s+1} = x_s + k_s \nabla f_s(x_s,y_s,z_s,\ldots)
$$

$$
y_{s+1} = y_s + k_s \nabla f_s(x_s,y_s,z_s,\ldots)
$$

$$
z_{s+1} = z_s + k_s \nabla f_s(x_s,y_s,z_s,\ldots)
$$

.................................................................

.................................................................

B.Vatchova