Genetic Search Algorithms to Fuzzy Multiobjective Games: a Mathematica Implementation

ANDRE A. KELLER
CNRS UMR8019, Université de Lille 1 Sciences et Technologies
Cité Scientifique 59655 Villeneuve d'Ascq Cedex FRANCE
andre.keller@univ-lille1.fr

Abstract: Genetic stochastic search algorithms (GAs) have soon demonstrated their helpful contribution in finding solutions to the complex real-life optimization problems. In 2005, Mastorakis' method successfully combines the GAs with the Nelder-Mead (NM) simplex optimization technique: the GAs are used first to reach the neighborhood of some global extremum, and the NM algorithm then finds it exactly. Playing games with genetic algorithms has been already proposed: it is a means of seeking better strategies in playing repeated games. These algorithms have been applied extensively for solving Nash equilibria of fuzzy bimatrix games with single objective. The experience shows the ability of the GAs to find solutions to equivalent quadratic programming problems without an exhaustive search. This paper is an attempt to consider the complexity of the real situations, when the decision makers are facing to multiple simultaneous objectives in a fuzzy environment. The software *MATHEMATICA* 7.0.1 is used to implement these techniques in a high-performance computing environment.

Key–Words: genetic algorithm, variable-size simplex algorithm, Nelder-Mead algorithm.

1 Introduction to Evolutionary Optimization

Evolutionary methods have proved their helpfull assistance to complex real-life problems such as with nonlinear bounded optimization problems and decentralized planning systems.

1.1 Bounded Optimization Problems

Let a nonlinear bounded programming problem

$$\min f(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^n$$
$$\text{s.t.} \quad \mathbf{x} \in [\mathbf{x}_l, \mathbf{x}_u].$$

A multimodal example with bounds may be deduced from a weighted combination of two sinc functions $g(x,y) = 3f(x+10, y+10) + 2f(x-5, y+5),\ x, y \in [-20, 10]$ where

$$f(x,y) = 50\frac{\sin(\sqrt{x^2+y^2})}{\sqrt{x^2+y^2}} - \sqrt{x^2+y^2}.$$

The hybridation of GA and classical optimization method is proposed [35] to find the global optimization solution : the GA is used first with a small number of iterations, and second the Newton's method for local optimization is used in this neighbourhood of the solution. The real-valued GA consists in Mathematica routines: a population of chromosomes is

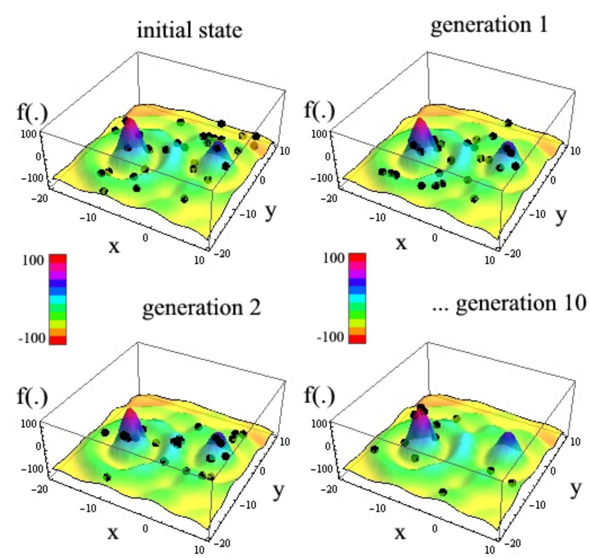

Figure 1: GA iterations

created randomly and the genetic processes of selection, crossover and mutation are then used for each iteration [1]. The GA is ended after an arbitrary 10 iterations calculation. The best result we obtain is $(\hat{x}, \hat{y}) = (-9.17265, -9.71843)$. The exact solution of the global optimization problem being at $(x^*, y^*) = -9.898, -9.966)$, the error of the GA 10 iterations approximation is $(x^* - \hat{x}, y^* - \hat{y}) = (-.052682, -.247571)$. The exact optimization is then reached [2] by using the Mathematica primitive for local maximization

$$\textbf{FindMaximum}[f[x,y], \{\{x,\hat{x}\},\{y,\hat{y}\}\}, \\ Method \rightarrow "Gradient"].$$

1.2 Constrained Optimization Problems

Let the standard nonlinear programming problem with n bounds,p inequality constraints, and $m-p$ equality constraints be [**?**, 36]

$$\begin{aligned} \min_{\mathbf{x}} \quad f(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^n \\ \text{subject to:} \\ g_i\mathbf{x}) \leq 0,\ i = 1, \ldots, p, \\ h_i\mathbf{x}) = 0,\ i = p+1, \ldots, m, \\ \mathbf{x}\ \in [\mathbf{x}_l, \mathbf{x}_u], \end{aligned}$$

where $f,\ g_i,\ h_i\ :\ \mathbb{R}^n \mapsto \mathbb{R}$ are respectively the cost function, the inequality and the equality constraint, and where $\mathbf{x}$ is the vector of n optimization

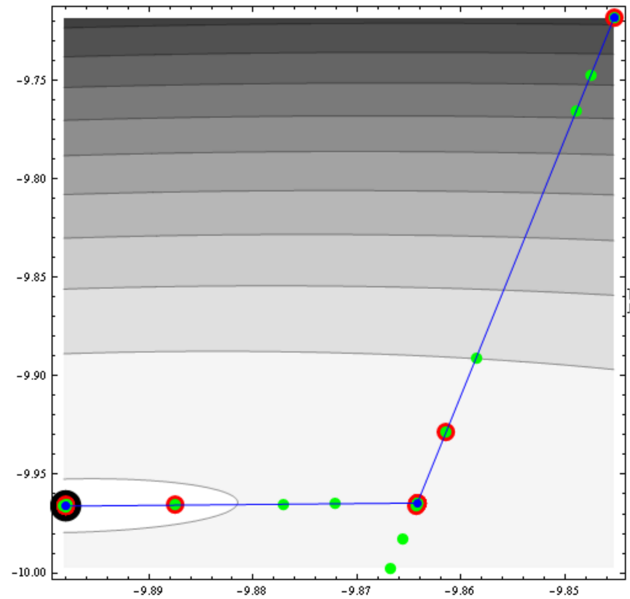

Figure 2: Iterative gradient method

variables. The search space $\mathcal{S} \subseteq \mathbb{R}^n$ is defined by the lower bounds $\mathbf{x}_l$ and upper bounds $\mathbf{x}_u$ of the variables $\mathbf{x}$. It is represented by the q-dimensional rectangle $\mathcal{S} = \prod_{substacki=1}^{q}[x_i^l, x_i^u],\ q \leq n$. For this problem, the set $\mathcal{F} \subseteq \mathcal{S}$ of feasible points is defined by m constraints such that

$$\mathtt{dom}\ \mathcal{F} = \bigcap_{i=1}^{p} \mathtt{dom}\ g_i\ \cap \bigcap_{i=p+1}^{m} \mathtt{dom}\ h_i.$$

It is included in the search space $\mathcal{S}$ defined . We have $\mathbf{x} \in \mathcal{F} \subseteq \mathcal{S} \subseteq \mathbb{R}^n$. Genetic algorithms may introduce penalty function which penalize infeasible solutions [36] , such as

$$f_p(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^{m} C_i d_i^{\kappa},$$

with

$$d_i = \begin{cases} \delta_i g_i(\mathbf{x}),\ i = 1, \ldots, p, \\ |h_i(\mathbf{x})|,\ i = p+1, \ldots, m. \end{cases}$$

where $f_p(\mathbf{x})$ denotes the penalized objective function, C_i a nonzero constant for violation of constraint i, d_i the distance metric of constraint i and κ a user defined parameter. The lagrangian for this problem is defined as

$$\mathcal{L}(\mathbf{x}, \lambda, \mu) = f(\mathbf{x}) + \sum_{i=1}^{p} \lambda_i g_i(\mathbf{x}) + \sum_{i=p+1}^{m} \mu_i h_i(\mathbf{x}),$$

[1] The Mathematica notebook consists in modules such as in the appendix A: 1- the module **createPopulation**$[nSize]$ randomly constructs a population of $nSize$ chromosomes and renders separately the values of (x, y) in $initialPopulation$ and the fitness in $fitList$, 2- **rankPopulation**$[initialPopulation, fitList, pSize]$ sorts the chromosomes according to their fitness, 3- the selection process uses **selectPopulation**$[..., keepRate]$, **rankWeighting**$[...]$ and **selectPairing**$[...]$, 4- the crossover process is using **crossOver**$[...]$ to get two offspring for each mating parent, and 5- the mutation process is using **mutatePopulation**$[..., mutationRate]$, **fitMatingPopulation**$[...]$, **fitMutatedPopulation**$[...]$.

[2] For this example, the global optimization may use the Mathematica primitive for the Nelder-Meade variable simplex algorithm

$$\textbf{NMaximize}[\{f[x,y], x_l \leq x \leq x_u, y_l \leq y \leq y_u\}, \{x,y\}, \\ Method \rightarrow "NelderMead"].$$

Using the Mathematica extra package $Optimization`UnconstrainedProblems`$, the primitive $FindMinimumPlot[-f[x,y], \{\{x,\hat{x}\},\{y,\hat{y}\}\}, Method \rightarrow "Newton]$ shows a 4 steps path between the best and the exact solutions.

where $\vec{\lambda}, \vec{\mu}$ denote the dual variables associated to the constraints.

The GA-based GENOCOP III package (see appendix B) is used for solving the following numerical nonlinear example

$$\min_{x,y} \quad f(x,y) = x^2 + 9y^2, \quad x, y \in \mathbb{R}$$

$$\begin{aligned}
&\text{subject to:}\\
g_1(x,y) &\equiv -2x_y + 1 \leq 0,\\
g_2(x,y) &\equiv -x - 3y + 1 \leq 0,\\
g_3(x,y) &\equiv (x+3)^2 + 3(y+1)^2 - 25 \leq 0,\\
x &\in [-1., 2.],\ y \in [-.5, 1.5].
\end{aligned}$$

The best solution $(\hat{x}, \hat{y}) = (.500091; .166636)$ with $f(\hat{x}, \hat{y}) = .5$ by GA is close to the exact optimum $(x^*, y^*) = (.5, .16666)$ at iteration 100 (see the illustration appendix B).

1.3 Multiple Objectives Optimization Problems

A multiobjective optimization problem (MOP) states that the decision variables **x** optimize a vector function of objective functions $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}))$ subject to constaints variable bounds for **x**. The MOP may be written

$$\min_{\mathbf{x}} \quad \mathbf{f}(\mathbf{x}) = \big(f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})\big), \quad \mathbf{x} \in \mathbb{R}^n$$

$$\begin{aligned}
&\text{subject to:}\\
g_i(\mathbf{x}) &\leq 0,\ i = 1, \ldots, p,\\
h_i(\mathbf{x}) &= 0, i = p+1, \ldots, m,\\
\mathbf{x} &\in [\mathbf{x}_l, \mathbf{x}_u].
\end{aligned}$$

Definition 1 *(Pareto optimality). Let $\mathcal{F}$ be the set of numbers which satifies the constraints, a point $\boldsymbol{x}^* \in \mathcal{F}$ is Pareto optimal if for every $\boldsymbol{x} \in \mathcal{F}$ either $\bigwedge_{i \in \mathbb{N}_k} f_i(\boldsymbol{x}) = f_i(\boldsymbol{x}^*)$ or there is at least one $i \in \mathbb{N}_k$ such that $f_i(\boldsymbol{x}) > f_i(\boldsymbol{x})$.*

1.4 Nested Optimization Problems

A bilevel programming problem (BLP) is concerning a hierachical decision system with two levels. At the lower decision level the decision maker (the follower) try to optimize its own objective function under the given decision pattern of a DM at the upper level (the leader). This Stackelberg situation may be illustrated

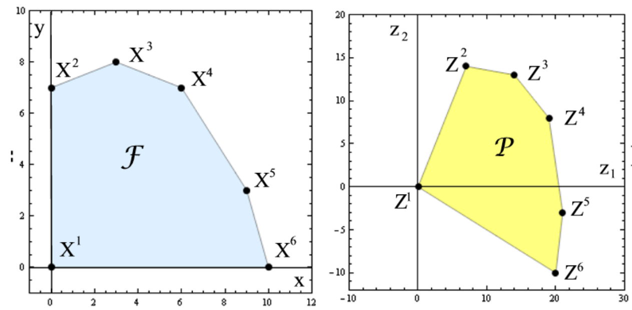

Figure 3: Feasible space and Pareto space

by the following BLP

$$\min_{\mathbf{x}} \quad F(\mathbf{x}, \mathbf{y}), \quad \mathbf{x} \in \mathbb{R}^{n_1}$$

$$\begin{aligned}
&\text{subject to:}\\
&G(\mathbf{x}, \mathbf{y}) \leq 0,\\
&\min_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}), \quad \mathbf{y} \in \mathbb{R}^{n_2}\\
&\text{subject to:} \quad g(\mathbf{x}, \mathbf{y}) \leq 0,
\end{aligned}$$

where the objective functions of the leader and the follower are respectively $F, f : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \mapsto \mathbb{R}$ with respective constraints defined by $G : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \mapsto \mathbb{R}^p$ and $g : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \mapsto \mathbb{R}^q$. The problem is tranformed into a single level problem, by replacing the lower level programming with its Kuhn-Tucker (K-T) conditions. We have

$$\min_{\mathbf{x},\mathbf{y},\mathbf{u}} \quad F(\mathbf{x}, \mathbf{y}), \quad \mathbf{x} \in \mathbb{R}^{n_1}$$

$$\begin{aligned}
&\text{subject to:}\\
G(\mathbf{x}, \mathbf{y}) &\leq 0,\\
\nabla_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) + \mathbf{u}' \nabla_{\mathbf{y}} g(\mathbf{x}, \mathbf{y}) &= 0\\
\mathbf{u}' g(\mathbf{x}, \mathbf{y}) &= 0,\\
g(\mathbf{x}, \mathbf{y}) &\leq 0,\\
\mathbf{u} &\geq 0,
\end{aligned}$$

where $\mathbf{u} \in \mathbb{R}^q$ denotes a vector of K-T multipliers. The following example is drawn from [1]. The BLP is defined by

$$\min_{x} \quad F(x,y) = x - 4y, \quad x \in \mathbb{R}$$

$$\begin{aligned}
&\text{subject to:}\\
&\min_{y} f(y) = y, \quad y \in \mathbb{R}\\
\text{subject to:} \quad g_1(x,y) &\equiv -x - y + 3 \leq 0,\\
g_2(x,y) &\equiv -2x + y \leq 0,\\
g_3(x,y) &\equiv 2x + y - 12 \leq 0,\\
g_4(x,y) &\equiv -3x + 2y + 4 \leq 0,\\
x &\geq 0,\ y \geq 0.
\end{aligned}$$

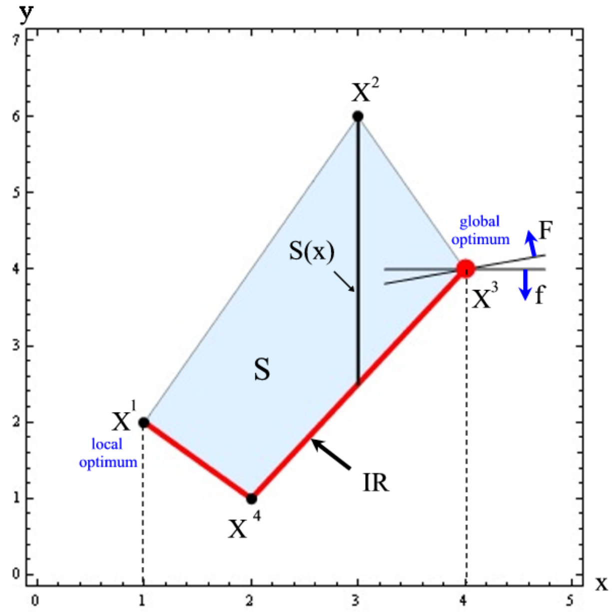

Figure 4: BLP example

The constraint region S is defined by $S \triangleq \{(x,y)|x \in X,\ y \in Y,\ g_i(x,y) \leq 0,\ i = 1,4\}$(Fig.4). The feasible set for the follower for each fixed $\bar{x} \in X$ is $S(x) \triangleq \{y \in Y|g_i(\bar{x},y) \leq 0\}$. For $\bar{x} = 3$, the feasible segment is shown in Fig4. The projection of S onto the leader decision space is $S(X) \triangleq \{x \in X,\ \exists y \in Y|g_i(x,y \leq 0,\ i = 1,4\}$. The follower's rational reaction set for $x \in S(X)$ is $P(x) \triangleq \{y \in Y|y\ \arg\max f(x,\hat{y})|\hat{y} \in S(x)\}$. The inductible region (IR)is $IR \triangleq \{(x,y)|(x,y) \in S,\ y \in P(x)\}$ (see the solid curve in Fig.4). Replacing the K-T conditions of the follower's problem, the BLP is transformed into a single-level programming problem. The lagrangian of the follower's problem is

$$\mathcal{L}(x_0,y,\vec{\lambda},\beta) \equiv f(y) + \sum_{i=1}^{4} \lambda_i g_i(x_0,y) - \beta y,$$

where $\vec{\lambda} = \{\lambda_1,\lambda_2,\lambda_3,\lambda_4\}$. We have the problem

$$\min_{x,y,\vec{\lambda},\alpha,\beta} \quad F(x,y), \quad x \in \mathbb{R}$$

subject to:

$$\nabla_y \mathcal{L}(x_0,y,\vec{\lambda},\beta) = 0,$$
$$\lambda_i g_i(x,y) = 0,\ i = 1,4$$
$$g_i(x,y) \leq 0,\ i = 1,4$$
$$\alpha\, x = 0,\ \beta\, y = 0$$
$$x \geq 0,\ y \geq 0,\ \lambda_i \geq 0,\ i = 1,4,\ \alpha \geq 0,\ \beta \geq 0.$$

For this problem, the Stackelberg solution is $(x^*,y^*) = (4,4)$ with the objectives of $F(x^*,y^*) = -12$ for the leader and $f(y*) = 4$ for the follower (see point X^3 in Fig.**??**. A local optimum is reached at $(x^*,y^*) = (1,2)$ with a better objective $f(1,2) = 2$ for the follower but a worse objective $F(1,2) = -7$ for the leader.

2 Single Objective Fuzzy Matrix Games Using Genetic Algorithms

2.1 Single Objective Fuzzy Matrix Games

Two players I and II have mixed strategies given by the n-dimensional vector $\mathbf{x}$ and the m-dimensional vector $\mathbf{y}$, respectively. Let $\mathbf{e}_n$ be an n-dimensional vector of ones, $\mathbf{e}_m$ having a dimension m. Suppose that the strategy spaces of Player I and II are defined by the convex polytopes $S^m = \{\mathbf{x} \in \mathbb{R}^m_+,\ \mathbf{x}'\mathbf{e}_m = 1\}$ and $S^n = \{\mathbf{y} \in \mathbb{R}^n_+,\ \mathbf{y}'\mathbf{e}_n = 1\}$, respectively. The payoffs of Players I and II are the $m \times n$ matrices $\mathbf{A}$ and $\mathbf{B}$, respectively. The objectives of Player I and Player II will be the programming problems $\{\max_{\mathbf{x}}\ \mathbf{x}'\mathbf{A}\mathbf{y}$ subject to $\mathbf{x}'\mathbf{e}_m = 1,\ \mathbf{x} \geq 0\}$, and $\{\max_{\mathbf{y}}\ \mathbf{x}'\mathbf{B}\mathbf{y}$ subject to $\mathbf{y}'\mathbf{e}_n = 1,\ \mathbf{y} \geq 0\}$, respectively. The expected payoffs of Players I and II are $\mathrm{E}_1(\mathbf{x},\mathbf{y}) = \mathbf{x}'\mathbf{A}\mathbf{y}$ and $\mathrm{E}_2(\mathbf{x},\mathbf{y}) = \mathbf{x}'\mathbf{B}\mathbf{y}$, respectively. Playing safe, the two players will select the strategy for which the maximum losses are minimum.

Definition 2 *A **Nash equilibrium point** is a pair of strategies $(\boldsymbol{x}^*,\boldsymbol{y}^*)$ such that the objectives of the two players are full filled simultaneously. We have*

$$\begin{aligned} \boldsymbol{x}'^*\boldsymbol{A}\boldsymbol{y}^* &= \max_{\boldsymbol{x}}\{\boldsymbol{x}'\boldsymbol{A}\boldsymbol{y}^*|\,\boldsymbol{x}'\boldsymbol{e}_m = 1,\ \boldsymbol{x} \geq 0\} \\ \boldsymbol{x}'^*\boldsymbol{B}\boldsymbol{y}^* &= \max_{\boldsymbol{y}}\{\boldsymbol{x}'^*\boldsymbol{B}\boldsymbol{y}|\,\boldsymbol{y}'\boldsymbol{e}_n = 1,\ \boldsymbol{y} \geq 0\} \end{aligned}$$

Applying the Kuhn-Tucker necessary and sufficient conditions, we have the Equivalence Theorem 3.

Theorem 3 (Mangasarian and Stone (1964)[?]) *(Equivalence Theorem) Let $G = (S^m, S^n, \boldsymbol{A}, \boldsymbol{B})$ be a bimatrix game, a necessary and sufficient condition that $(\boldsymbol{x}^*,\boldsymbol{y}^*)$ be an equilibrium point is the solution of the QP problem*

$$\max_{\boldsymbol{x},\boldsymbol{y},p,q} \quad \boldsymbol{x}'(\boldsymbol{A}+\boldsymbol{B})\boldsymbol{y} - p - q$$

subject to

$$\boldsymbol{A}\boldsymbol{y} \leq p\boldsymbol{e}_n,$$
$$\boldsymbol{B}'\boldsymbol{x} \leq q\boldsymbol{e}_m,$$
$$\boldsymbol{x}'\boldsymbol{e}_m = 1,$$
$$\boldsymbol{y}'\boldsymbol{e}_n = 1,$$
$$\boldsymbol{x} \geq 0,\ \boldsymbol{y} \geq 0,$$

where $p,q \in \mathbb{R}$ are the negative of the multipliers associated with the constraints.

Proof: see appendix **??**. $\square$

The Lemke-Howson 's algorithm (1964) [**?**, **?**, **?**] can be used when computing the Nash equilibrium payoffs.

Definition 4 *Let the expected payoff of Player I be* $D_1 = \{\boldsymbol{x}'\boldsymbol{A}\boldsymbol{y} |\ \boldsymbol{x} \in S^m,\ \boldsymbol{y} \in S^n\}$*. A* ***fuzzy goal*** *for Player I is a fuzzy set* $\tilde{G}_1$ *represented by the membership function (MF)* $\mu_1 : D_1 \mapsto [0, 1]$.

Definition 5 *Let the expected payoff of Player II be* $D_2 = \{\boldsymbol{x}'\boldsymbol{B}\boldsymbol{y} |\ \boldsymbol{x} \in S^m,\ \boldsymbol{y} \in S^n\}$*. A* ***fuzzy goal*** *for Player II is similarly a fuzzy set* $\tilde{G}_2$ *represented by the MF* $\mu_2 : D_2 \mapsto [0, 1]$.

An equilibrium solution is defined with respect to (w.r.t.) the degree of attainment of the fuzzy goals.

Definition 6 *A pair* $(\boldsymbol{x}^*, \boldsymbol{y}^*) \in S^m \times S^n$ *is an* ***equilibrium solution*** *if, for other strategies,we have*

$$\begin{aligned} \mu_1(\boldsymbol{x}'^{*}\boldsymbol{A}\,\boldsymbol{y}^*) &\geq \mu_1(\boldsymbol{x}'\boldsymbol{A}\boldsymbol{y}^*), \text{ for all } \boldsymbol{x} \in S^m \\ \mu_2(\boldsymbol{x}'^{*}\boldsymbol{B}\boldsymbol{y}^*) &\geq \mu_2(\boldsymbol{x}'^{*}\boldsymbol{B}\boldsymbol{y}), \text{ for all } \boldsymbol{y} \in S^n \end{aligned}$$

The expression of the linear MF of the fuzzy goal $\tilde{G}_1$ for Player I is

$$\mu_1(\mathbf{x}'\mathbf{A}\mathbf{y}) = \begin{cases} 1, & \mathbf{x}'\mathbf{A}\mathbf{y} \geq \bar{a} \\ \frac{\mathbf{x}'\mathbf{A}\mathbf{y} - \underline{a}}{\bar{a} - \underline{a}}, & \underline{a} < \mathbf{x}'\mathbf{A}\mathbf{y} < \bar{a} \\ 0, & \mathbf{x}'\mathbf{A}\mathbf{y} \leq \underline{a}, \end{cases}$$

where $\underline{a}$ denotes the worst degree of satisfaction of Player I, whereas $\bar{a}$ denotes the best degree of satisfaction. These values are defined as

$$\begin{aligned} \underline{a} &= \min_{\mathbf{x}\in X} \min_{\mathbf{y}\in Y} \mathbf{x}'\mathbf{A}\mathbf{y} = \min_i \min_j a_{ij} \\ \bar{a} &= \max_{\mathbf{x}\in X} \max_{\mathbf{y}\in Y} \mathbf{x}'\mathbf{A}\mathbf{y} = \max_i \max_j a_{ij} \end{aligned}$$

The expression of the linear MF of the fuzzy goal $\tilde{G}_2$ for Player II is, as well

$$\mu_2(\mathbf{x}'\mathbf{B}\mathbf{y}) = \begin{cases} 1, & \mathbf{x}'\mathbf{B}\mathbf{y} \geq \bar{b} \\ \frac{\mathbf{x}'\mathbf{B}\mathbf{y} - \underline{b}}{\bar{b} - \underline{b}}, & \underline{b} < \mathbf{x}'\mathbf{B}\mathbf{y} < \bar{b} \\ 0, & \mathbf{x}'\mathbf{B}\mathbf{y} \leq \underline{b}, \end{cases}$$

where $\underline{b}$ and $\bar{b}$ also denote the worst and the best degree of satisfaction of Player II, respectively. These values are deduced from similarly using $\mathbf{B}$.

Theorem 7 (Equilibrium solution) *An equilibrium solution* $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ *of the fuzzy bimatrix game, is deduced from the optimal solution* $(\boldsymbol{x}^*, \boldsymbol{y}^*, p^*, q^*)$ *of the QP problem*

$$\begin{aligned} \max_{\boldsymbol{x},\boldsymbol{y},p,q} \quad & \boldsymbol{x}'(\hat{\boldsymbol{A}} + \hat{\boldsymbol{B}})\boldsymbol{y} - p - q \\ & \textit{subject to} \\ & \hat{\mathbf{A}}\boldsymbol{y} \leq p\boldsymbol{e}_n, \\ & \hat{\boldsymbol{B}}'\boldsymbol{x} \leq q\boldsymbol{e}_m, \\ & \boldsymbol{x}'\boldsymbol{e}_m = 1, \\ & \boldsymbol{y}'\boldsymbol{e}_n = 1, \\ & \boldsymbol{x} \geq 0,\ \boldsymbol{y} \geq 0, \end{aligned}$$

where $\hat{\mathbf{A}} = \mathbf{A}/(\bar{a} - \underline{a})$ and $\hat{\mathbf{B}} = \mathbf{B}/(\bar{b} - \underline{b})$.

Proof: see Bector and Chandra [**?**], p. 180. $\square$

2.2 Hybridized Genetic Algorithm's Solution

In the Nishizaki and Sakawa's multiobjective example ([**?**], pp. 93–95), Player I has three pure strategies and Player II four strategies. Let us retain a single objective version, with the following payoffs

$$\mathbf{A} = \begin{pmatrix} 1 & 4 & 7 & 2 \\ 3 & 6 & 1 & 8 \\ 2 & 5 & 3 & 9 \end{pmatrix} \text{ and } \mathbf{B} = \begin{pmatrix} 5 & 1 & 2 & 4 \\ 3 & 4 & 8 & 3 \\ 1 & 8 & 1 & 2 \end{pmatrix}$$

The values of the worst and the best degree of satisfaction are given by $\underline{a} = \underline{b} = 1,\ \bar{a} = 9,\ \bar{b} = 8$. We have the QP problem

$$\begin{aligned} \max_{\mathbf{x},\mathbf{y},p,q} \quad & \frac{1}{8}\mathbf{x}'\,\mathbf{A}\,\mathbf{y} + \frac{1}{7}\mathbf{x}'\,\mathbf{B}\,\mathbf{y} - p - q \\ & \text{subject to} \\ & \frac{1}{8}\,\mathbf{A}\,\mathbf{y} \leq p\,\mathbf{e}_3, \quad \frac{1}{7}\,\mathbf{B}'\,\mathbf{x} \leq q\,\mathbf{e}_4, \\ & \mathbf{x}'\,\mathbf{e}_3 = 1, \quad \mathbf{y}'\,\mathbf{e}_4 = 1, \\ \mathbf{x}' = (x_1, x_2, x_3) \geq 0,\ & \mathbf{y}' = (y_1, y_2, y_3, y_4) \geq 0. \end{aligned}$$

The optimum solutions of the QP problem have been obtained by an iterative method [3]. We have $x^* = (.4795, .2877, .2329)$, $y^* = (.0481, .5948, .2857, .0714)$, $p^* = .5712$, $q^* = .4990$.

[3] The QP problem is solved by using the primitive 'FindMaximum' of the Mathematica package.

3 Multiple Objectives Fuzzy Matrix Games Using Genetic Algorithms

3.1 Bimatrix games definitions

A single objectives bimatrix game [4] (a non-zero-game with two players) is evaluated in a fuzzy environment where the objectives are uncertain. A bimatrix game is represented by $G = (S^m, S^n, \mathbf{A}, \mathbf{B})$. Let $\mathbf{e}_m$ be an m-dimensional vector of ones, $\mathbf{e}_n$ having a dimension n.The players' strategy spaces are the convex polytopes $S^m = \{\mathbf{x} \in \mathbb{R}_{\geq 0}, \mathbf{x}'\mathbf{e}_m = 1\}$ and $S^n = \{\mathbf{y} \in \mathbb{R}_{\geq 0}, \mathbf{y}'\mathbf{e}_n = 1\}$. The list of the r payoff matrices for Player I is represented by $\mathbf{A}^k = (a^k_{ij})_{m\times n},\ k \in K \triangleq \{1, \ldots, r\}$. The list of the s payoff matrices for Player II is represented by $\mathbf{B}^l = (b^l_{ij})_{m\times n},\ l \in L \triangleq \{1, \ldots, s\}$. *Pure strategies* of the players correspond to the rows and columns for each matrix : if Player I chooses a pure strategy $i \in I \triangleq \{1, \ldots, m\}$ and Player II a pure strategy $j \in J \triangleq \{1, \ldots, n\}$, Player I obtains the payoff vector $\left(a^1_{ij}, \ldots, a^r_{ij}\right)$ and Player II the payoff vector $\left(b^1_{ij}, \ldots, b^s_{ij}\right)$. *Mixed strategies* are define by the probabilities $\mathbf{x} \in X \triangleq \{\mathbf{x} \in \mathbb{R}^m |\ \mathbf{e}'_m.\mathbf{x} = 1,\ \mathbf{x} \geq 0\}$ for Player I and $\mathbf{y} \in X \triangleq \{\mathbf{y} \in \mathbb{R}^n |\ \mathbf{e}'_n.\mathbf{y} = 1,\ \mathbf{y} \geq 0\}$ for Player II. For any pair of mixed strategies $(\mathbf{x}, \mathbf{y})$, the Player I's k-th *expected payoff* is $x'.A^k.y$ and the Player II's l-th expected payoff $x'.B^l.y$. The objectives of Player I and Player II will be the Programming problems $\max_{\mathbf{x}} \mathbf{x}'\mathbf{A}\mathbf{y}$ `subject to` $\mathbf{x}'\mathbf{e}_m = 1, \mathbf{x} \geq 0]$, and $\max_{\mathbf{y}} \mathbf{x}'\mathbf{B}\mathbf{y}$ `subject to` $\mathbf{y}'\mathbf{e}_n = 1, \mathbf{x} \geq 0$, respectively.In this study, we assume that the two players have fuzzy goals.

3.2 Nishizaki-Sakawa's model

Definition 8 *Let the **fuzzy goals** for Players I and II be denoted by $\boldsymbol{p}_1 = (p^1_1, \ldots, p^r_1) \in D_1 \subseteq \mathbb{R}^r$ and $\boldsymbol{p}_2 = (p^1_2, \ldots, p^s_2) \in D_2 \subseteq \mathbb{R}^s$. The Player I's* k*th fuzzy goal G^k_1 is a fuzzy set characterized by the membership function (MF)*

$$\mu^k_1 : D^k_1 \mapsto [0, 1].$$

Similarly, the Player II's l*th fuzzy goal G^l_2 is a fuzzy set characterized by the MF*

$$\mu^l_2 : D^l_2 \mapsto [0, 1].$$

A fuzzy goal expresses the player's degree of satisfaction for the corresponding payoff [42]. Players are assumed to specify intervals for their degree of satisfaction, such as $\underline{a} \leq p \leq \bar{a}$ for Player I. For $p < \underline{a}$, Player I's kth MF is $\mu^k(p) = 0$, for $p < \bar{a}$we have $\mu^k(p) = 1$ and for $\underline{a} \leq p \leq \bar{a}$, $\mu^k(p)$ is supposed continuous and strictly increasing. If the Player I's MF of the fuzzy goal $\mu^k(\mathbf{x}\mathbf{A}^k\mathbf{y})$ is a linear function, we then have for any mised strategies $(\mathbf{x}, \mathbf{y})$

$$\mu^k(\mathbf{x}\mathbf{A}^k\mathbf{y}) = \begin{cases} 1, & \mathbf{x}\mathbf{A}^k\mathbf{y} > \bar{a}^k \\ 1 - \frac{\bar{a}^k - \mathbf{x}\mathbf{A}^k\mathbf{y}}{\bar{a}^k - \underline{a}^k}, & \underline{a}^k < \mathbf{x}\mathbf{A}^k\mathbf{y} \leq \bar{a}^k \\ 0, & \mathbf{x}\mathbf{A}^k\mathbf{y} \leq \underline{a}^k, \end{cases}$$

where the payoff $\underline{a}^k$ gives the worst degree of satisfaction for Player I w.r.t. the kth objective. It is computed as

$$\underline{a}^k = \min_{x\in X} \min_{y\in Y} \mathbf{x}\mathbf{A}^k\mathbf{y} = \min_{i\in I} \min_{j\in J}\ a^k_{ij}.$$

On the contrary, the payoff $\bar{a}^k$ will give the best degree of satisfaction for Player I w.r.t. the kth objective and is computed as

$$\bar{a}^k = \max_{x\in X} \max_{y\in Y} \mathbf{x}\mathbf{A}^k\mathbf{y} = \max_{i\in I} \max_{j\in J}\ a^k_{ij}.$$

The fuzzy decision rule by Bellman and Zadeh [5] may then be used to aggregate the goals, such as

$$\mu(\mathbf{x}, \mathbf{y}) = \min_{k\in K} \mu^k(\mathbf{x}, \mathbf{y}) = \min_{k\in K} \left(1 - \frac{\bar{a}^k - \mathbf{x}\mathbf{A}^k\mathbf{y}}{\bar{a}^k - \underline{a}^k}\right).$$

We also have (see Nishizaki and Sakawa [**?**], p.47)

$$\mu(\mathbf{x}, \mathbf{y}) = \min_{k\in K} \left(\sum_{i=1}^{m} \sum_{j=1}^{n} \hat{a}^k_{ij} x_i y_j + c^k\right),$$

where

$$\hat{a}^k_{ij} = \frac{a^k_{ij}}{\bar{a}^k - \underline{a}^k} \texttt{ and } c^k = -\frac{\underline{a}^k}{\bar{a}^k - \underline{a}^k}.$$

Theorem 9 (Equilibrium solution) *An equilibrium solution w.r.t. the degree of attainment of the aggregated fuzzy goal is the optimal solution of a nonlinear programming problem*

$$\begin{aligned} \max_{\boldsymbol{x}\boldsymbol{y},\sigma_1,\sigma_2,p,q} \quad & \sigma_1 + \sigma_2 - p - q \\ & \textit{subject to} \\ \boldsymbol{x}\hat{\boldsymbol{A}}^k\boldsymbol{y} + c^k_1 &\geq \sigma_1,\ k \in \mathbb{N}_r \\ \boldsymbol{x}\hat{\boldsymbol{B}}^l\boldsymbol{y} + c^l_2 &\geq \sigma_2,\ l \in \mathbb{N}_s \\ \hat{\boldsymbol{A}}^k\boldsymbol{y} + c^k_1\boldsymbol{e}_m &\leq p\boldsymbol{e}_m,\ \exists k \in \mathbb{N}_r \\ \hat{\boldsymbol{B}'}^l\boldsymbol{x} + c^l_2\boldsymbol{e}_n &\leq q\boldsymbol{e}_n,\ \exists l \in \mathbb{N}_r \\ \boldsymbol{x}'\boldsymbol{e}_m = 1,&\ \boldsymbol{y}'\boldsymbol{e}_n = 1, \\ \boldsymbol{x} \geq 0,&\ \boldsymbol{y} \geq 0. \end{aligned}$$

[4] This presentation is inspired from Nishizaki and Sakawa [42]

[5] R.E. Bemman and L.A. Zadeh, Decision making in a fuzzy environment, *Management Science*, **17**, 141–164, 1970. One another method for aggregating multiple fuzzy goals is weighting the coefficients.

The optimal solutions are obtained by solving $r \times s$ problems, including each only one of the two classes of r and s inequalities.

3.3 Numerical example

In the following example, two players example [6]. Players I and II have respectively three and four pure strategies, and three different objectives. The goals of the two players are fuzzy. The payoff matrices for Players I and II respectively, are

$$\mathbf{A}^1 = \begin{pmatrix} 2 & 6 & 5 & 7 \\ 2 & 0 & 5 & 4 \\ 4 & 7 & 6 & 9 \end{pmatrix}, \mathbf{A}^2 = \begin{pmatrix} 3 & 6 & 8 & 2 \\ 6 & 2 & 0 & 8 \\ 2 & 9 & 7 & 4 \end{pmatrix}$$

$$\mathbf{A}^3 = \begin{pmatrix} 1 & 4 & 7 & 2 \\ 3 & 6 & 1 & 8 \\ 2 & 5 & 3 & 9 \end{pmatrix}, \mathbf{B}^1 = \begin{pmatrix} 1 & 6 & 1 & 7 \\ 8 & 2 & 3 & 4 \\ 4 & 9 & 3 & 5 \end{pmatrix}$$

$$\mathbf{B}^2 = \begin{pmatrix} 8 & 2 & 0 & 8 \\ 1 & 9 & 7 & 6 \\ 5 & 2 & 8 & 5 \end{pmatrix}, \mathbf{B}^3 = \begin{pmatrix} 5 & 1 & 2 & 4 \\ 3 & 4 & 8 & 3 \\ 1 & 8 & 1 & 2 \end{pmatrix}$$

The values of the worst and the best degree of satis-

program	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
x_1	0.	0.	0.	0.2759	0.2059	0.	0.	0.	0.
x_2	0.	1.	0.2815	0.	0.4239	0.2318	0.	1.	0.2815
x_3	1.	0.	0.7185	0.7241	0.3702	0.7682	1.	0.	0.
y_1	0.	0.3289	0.	0.	0.	0.	0.	0.408	0.
y_2	0.397	0.	0.5207	0.459	1.	0.593	0.397	0.1194	0.5207
y_3	0.126	0.3023	0.0793	0.	0.	0.	0.126	0.2935	0.0793
y_4	0.477	0.3688	0.4	0.541	0.	0.407	0.477	0.1791	0.4
σ_1	0.707	0.405	0.6788	0.6241	0.4252	0.7089	0.707	0.3333	0.6788
σ_2	0.4652	0.5016	0.5325	0.4523	0.5519	0.4753	0.4652	0.5124	0.5325
p	0.7057	0.5279	0.7023	0.2404	0.	0.1809	0.8698	0.649	0.8579
q	0.375	0.125	0.25	0.2715	0.5519	0.25	0.375	0.125	0.25
solution	0.0915	0.2537	0.259	0.5644	0.4252	0.7533	−0.0725	0.0718	0.1034

Figure 5: Optimal solutions

faction are given by $\underline{a} = \underline{b} = 1,\ \bar{a} = 9,\ \bar{b} = 8$. We have the QP problem

$$\max_{\mathbf{x},\mathbf{y},p,q} \ \frac{1}{8}\mathbf{x}'\,\mathbf{A}\,\mathbf{y} + \frac{1}{7}\mathbf{x}'\,\mathbf{B}\,\mathbf{y} - p - q$$

$$\text{subject to}$$

$$\frac{1}{8}\,\mathbf{A}\,\mathbf{y} \le p\,\mathbf{e}_3, \quad \frac{1}{7}\,\mathbf{B}'\,\mathbf{x} \le q\,\mathbf{e}_4,$$

$$\mathbf{x}'\,\mathbf{e}_3 = 1, \quad \mathbf{y}'\,\mathbf{e}_4 = 1,$$

$$\mathbf{x}' = (x_1, x_2, x_3) \ge 0,\ \mathbf{y}' = (y_1, y_2, y_3, y_4) \ge 0.$$

The optimal solutions ... The optimum solutions of the QP problem have been obtained by an iterative

generations:	x_1	x_2	x_3	y_1	y_2	y_3	y_4	p	q
100	0.4007	0.4397	0.1596	0.4077	0.1782	0.2175	0.1966	0.6629	0.468
500	0.4436	0.5416	0.0148	0.3507	0.2482	0.2699	0.1312	0.6603	0.3602
1000	0.4186	0.5807	0.0007	0.3742	0.194	0.3574	0.0744	0.6304	0.335
5000	0.0999	0.8158	0.0843	0.3605	0.1776	0.2699	0.192	0.6704	0.2487
10 000	0.3142	0.6858	0.	0.3911	0.2091	0.2241	0.1757	0.6616	0.2821
50 000	0.0383	0.793	0.1687	0.3652	0.2627	0.2658	0.1063	0.6501	0.2917
100 000	0.0043	0.8574	0.1383	0.3515	0.1776	0.2814	0.1895	0.6715	0.2481
optimum	0.	1.	0.	0.408	0.1194	0.2935	0.1791	0.649	0.125

Figure 6: Best solutions by using GA

method [7]. We have $x^* = (.4795, .2877, .2329),\ y^* = (.0481, .5948, .2857, .0714),\ p^* = .5712,\ q^* = .4990$.

4 Conclusion

A Simple Genetic Algorithm with Mathematica

Principles and Pseudo-code

Genetic algorithms (GAs) are stochastic search techniques which procedures are inspired from the genetic processes of biological organisms by using encodinds and reproduction mechanisms. Theses principles may be adapted to real-world optimization problems (OPs)for which the traditional gradient methods may not be adequate. Let a simple OP be $max\ f(\mathbf{x})$ subject to lower and upper bounds $\mathbf{x}_l \le \mathbf{x} \le \mathbf{x}_u$, where $\mathbf{x}, \mathbf{x}_l, \mathbf{x}_u \in \mathbb{R}^n$. In binary-coded GAs, each parameter value is encoded as a gene (binary string) and concatenate together into a chromosome (a vector of parameter values). The problem is then translated to a combinatorial problem, the points of which are corners of a high-dimensional cube [**?**]. Let $P(t)$ be a population of potential solutions at generation t, and new individuals (offspring)$C(t)$, the peudo-code is shown as algorithm A.1 [2, 17]. An initial population of individuals (chromosomes) is generated at random, and will evolve over successive improved generations towards the global optimum . The individuals evolve through successive generations t (iterations) by means of genetic operators. More precisely, a new population $P(t+1)$ is formed by selecting the more fit individuals, whose members undergo reproduction by means of crossover and mutation. Usually, a gene has converged when 95 % of the population has the same value and the population converges when all the genes have converged [2].

[6] This numerical application is an adaptation of the Nishizaki and Sakawa's example, page 94 [42].

[7] The QP problem is solved by using the primitive 'FindMaximum' of the Mathematica package.

Algorithm A.1: simple genetic algorithm
begin /* initial random population */
  t:=0;
  generate initial P(t);
  evaluate fitness of P(t);
    while (NOT finished) **do**;
    begin /* new generation */
      for populationSize/2 **do**;
      begin /* reproductive cycle */
        select two individuals from P(t) for mating;
        recombine P(t) to yield offspring C(t);
        evaluate offspring's fitness;
        select P(t+1) from P(t) and C(t);
        t:=t+1;
      end
      if population has converged **then**
          finished := TRUE
    end
end

Binary Encoding and Fitness

Let the OP be simply the scalar function $\max\ f(x,y),\ x,y \in \mathbb{R}$, each variable may be represented by a 5-bit binary number[8]. An individual (or chromosome) contains two parameters (or genes) and consists of 10 binary digits, such as for (x,y)= $(8,10)_{10} \equiv (01000|01010)_2$. Let a simple OP with bounds be [39] $\max\ f(x) = x\sin(10\pi x) + 1,\ x \in [-1,2]$. Suppose that the required precision is six decimals. The range of x is $3 = 2 - (-1)$. The domain should be divised into at least 3×10^6 equal size subranges. Since we have $2097152 = 2^{21} \leq 3 \times 10^6 \leq 2^{22} = 4194304$, 22 bits are required as a binary vector. The lower and upper bounds of x are the 22-bit strings $(000\dots000)$ and $(111\dots111)$ respectively. The mapping from the 22-bits string $(< b_{21}, b_{20}, \dots, b_0 >)_2$ into a real number x in $[-1,2]$ requires two steps: firstly, convert the binary string to base 10

$$(< b_{21}, b_{20}, \dots, b_0 >)_2 = \Big(\sum_{i=0}^{21} b_i 2^i\Big)_{10} = x',$$

and secondly find the corresponding real number

$$x = -1 + x' \frac{3}{2^{22}-1}.$$

[8]Real-number encoding has been proposed to prevent some drawbacks of the binary encoding [9]: in industrial engineering optimization problems, 100 variables in the range $[-500, 500]$with a 6 digits precision would produce a binary solution vector of length 3000 and generate a serach space of 10^{1000}.

For example, $(1000101110110101000111) = x'$ represents 0.637197, since we have $x' = 2288967$ and $x = -1 + 2288967 \times \frac{3}{4194303} = .637197$. The fitness of individuals depends on the performance of the corresponding phenotypes (objective function). The Mathematica module for decoding the chromosome is shown in Figure **??**.

```
chromosome = {1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1}
{1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1}

 decodeBGA[chromosome_] :=
   Module[{lList, lchrom, values, phenotype, xinf, xsup},
    lchrom = Length[chromosome];
    pList = Flatten[Position[chromosome, 1]];
    values = Map[2^(lchrom - #) &, pList];
   decimal = Apply[Plus, values]; (*convert to decimal*)
   xinf = -1; xsup = 2; (*scaling  to the proper range*)
   phenotype = N[-1 + decimal ((xsup - xinf) / (2^lchrom - 1))];
   Return[phenotype];  ];

decodeBGA[chromo2]

0.637197
```

Figure A.1: Coding procedure

Genetic Operators

There are three types of operators for the reproduction phase: the selection operator of more fitted individuals, the crossover operator that creates new individuals by combining parts of strings of two individuals and the mutation that make one or more changes in a single individual string. Each gene (string) is selected with a probability proportional to its fitness value. The biased roulette-wheel mechanism consists in a wheel with N divisions, where the size is in proportion to the fitness value (see Figure **??**). The wheeel is spun N

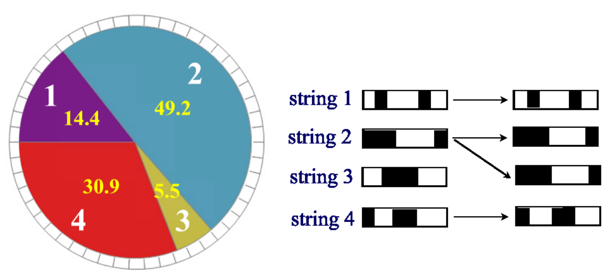

Figure A.2: Biased roulette-wheel

times each times chosing the individual indicated by the pointer. At a crossover single point, the chromosomes of two performant individuals (parents) are cut at some random position.The tail segments are then swapped over to create two new chromosomes (see Figure **??**). The Mathematica primitives [9]. The muta-

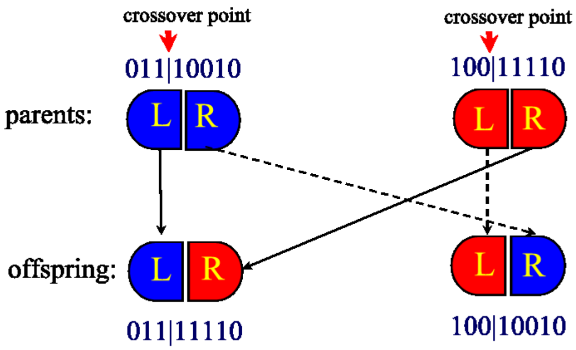

Figure A.3: Single point crossover

```
string1 = Table[Random[Integer], {stringLength}]; (* random pair of chromosomes*)
string2 = Table[Random[Integer], {stringLength}];
Print[string1, string2]

{1, 1, 1, 1, 0, 1}{0, 0, 0, 0, 1, 0}

doCrossover[{string1_, string2_}] := (* mating two chromosome*)
  Module[{stle, cut, temp1, temp2},
   stle = Length[string1];
   cut = Random[Integer, {1, stle - 1}]; (* cut point at random*)
   temp1 = Join[Take[string1, cut],  Drop[string2, cut]];
   temp2 = Join[Take[string2, cut],  Drop[string1, cut]];
  Return[{cut, temp1, temp2}] ];

doCrossover[{string1, string2}]

{4, {1, 1, 1, 1, 1, 0}, {0, 0, 0, 0, 0, 1}}
```

Figure A.4: Simple Crossover

tion operator alters one or more genes of the offspring. The crossover and mutation probabilities, denoted by p_c and p_m respectively, are key parameters of control besides the population size N.

Example

Let the OP be

$$\max f(x) = 1 + \cos(\pi x) + (3x \; mod \; 1), \;\; x \in [0, 1].$$

The exact solution is given by $x^* = .2739, f(x^*) = 1.5358$. The application uses the simple Mathematica notebook due to Bengtsson [4]. The population size is 32, the string length is 6 and the mutation rate is .002. The Fig. **??** shows the initial and the final eleventh generation of chromosomes.

```
string

{1, 0, 0, 0, 1, 1}

mutationRate = .002;
doMutation[string_] :=
  Module[{tempstring, i},
    tempstring = string;
   Do[If[Random[] < mutationRate,
      tempstring[[i]] = 1 - tempstring[[i]]],
    {i, stringLength}];
   Return[{tempstring}]
 ]

doMutation[string]

{{1, 0, 0, 0, 1, 1}}
```

Figure A.5: Mutation

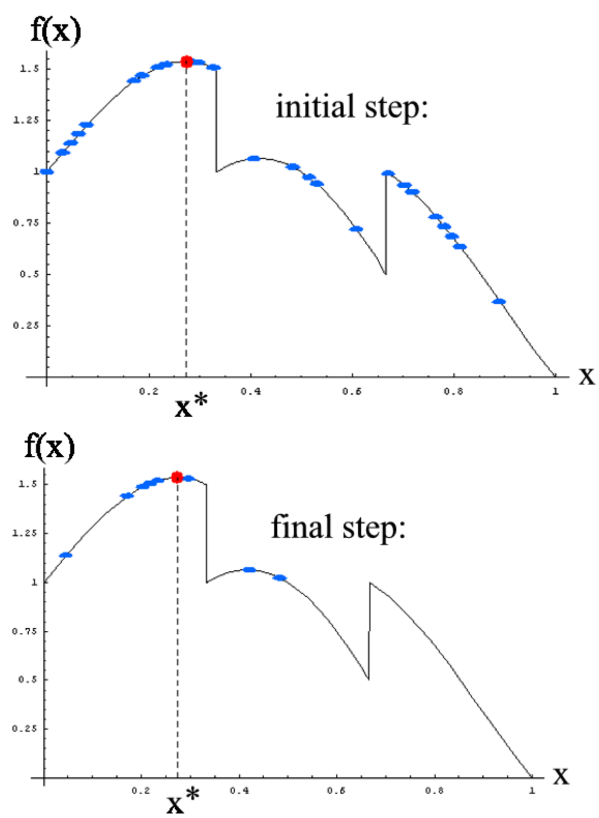

Figure A.6: Simple application of GAs

[9]The Mathematica primitives have been adapted frm Freeman [14] and Bengtsson [**?**] are shown in Fig. **??**

B GA-based Optimization Package GENOCOP III

The GENOCOP system [36, 36] retains a *floating point* representation: for a problem with n variables, the *i*-th chromosome in a permisible solution is coded as a *n*-dimensional vector. The GENOCOP system initialy a population of potential solutions. Two sub-population are considered: the first population P_s consists of search points satisfying *only* the linear constraints and the second population P_r consists of reference points satsfying *all* the constraints. The development in one population influences the evaluations of individuals in the second. The reference points are infeasible search points are "repaired" for evaluation. The feasibility of the points in P_s is maintained trough specific operators [10] Linear equations must be eliminated at the beginning to prevent from instabilities. The number of variables is reduced by substitutions. The nonlinear equation $h_k(\mathbf{x}) = 0, k = q+1, m$ are replaced by a pair of inequations $\varepsilon \leq h_k(x) \leq \varepsilon$, where the parameter ε defines the precision of the system.

GENOCOP is notably available from the University of North Carolina / College of Engeneering at Charlotte (USA) : ftp.uncc.edu/coe/evol (file "genocopIII.tar.Z") [11]. Different data and controlled parameters are defined in the input file, such as: the linear inequalities and ranges for the variables, the population size, the number of generations, a "0" for a minimization problem and a "1" otherwise, a "1" for a start from a single point (identical individuals) or "0" for a start from a random population, the probability of replacement, etc. Figs.B.1 to B.1 illustrate and give more details about the example of section 1.2. Fig.B.1 shows the feasible region $\mathcal{F}$. From the convexity of the feasible region, it follows that for each point $a \in \mathcal{F}$, there exists feasibles ranges, such as $[\underline{x}_a, \bar{x}_a]$ for a *fixed* $y(a)$, and $[\underline{y}_a, \bar{y}_a]$ for a fixed $x(a)$. The listing of the ouput file is shown in Fig.B.2. Fig.B.3 shows the solutions and errors for different number od generations. The exact solution is praticaly obtained after 100 iterations of the GA approach.

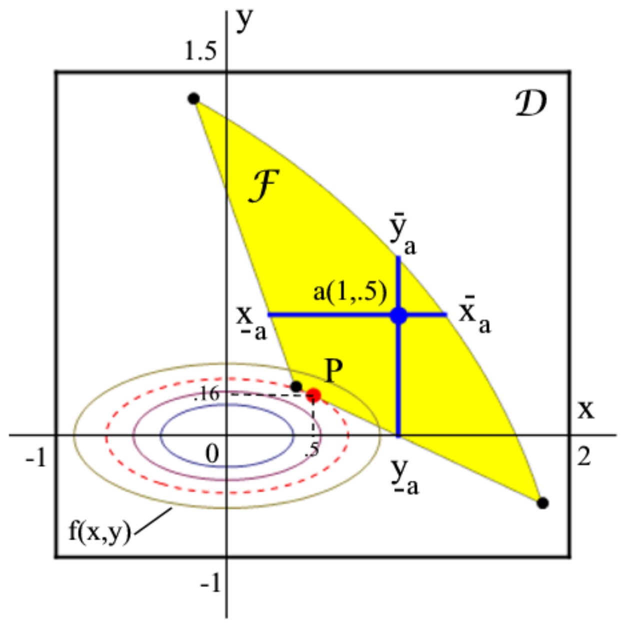

Figure B.1: Feasible region and solution

```
Sun Jul 25 10:25:40 2010


Linear inequalities :
 -2.00X1 - 1.00X2 <=  -1.00
 -1.00X1 - 3.00X2 <=  -1.00


Domains :
 -1.00   <=  X1   <=    2.00
 -0.50   <=  X2   <=    1.50


Test case number          : 5
Number of operators       : 4  4  4  4  4  4  4
Number of generations     : 10
Population size           : 70
Parameter B               : 6
Parameter Q               : 0.100000
Max/Min                   : 0
Probability of replacement : 0.150000
Initialization mode       : 0
Selection of refer. point  : 0
Number of NI constraints   : 1
Number of NE constraints   : 0


USING MULTIPLE POINT INITIAL POPULATION...




Generation#      Solution Value
      1              0.73862767
      2              0.68133080
      5              0.67284006
      7              0.61450130
      8              0.60458982
      9              0.56254035

Best solution was found at generation 10 (solution value = 0.56254035)


Best solution found:

 X[ 1] :             0.62503725
 X[ 2] :             0.13819019


Total run time : 30 seconds
```

Figure B.2: Listing of the output file

[10]Given a search point $\mathbf{s} \in P_s$, if $\mathbf{s}$ is fully feasible ($\mathbf{s} \in \mathcal{F}$), then $\texttt{eval}(\mathbf{s}) = f(\mathbf{s})$. Otherwise (i.e., $\mathbf{s}notnot \in \mathcal{F}$), the system selects one od the reference points in P_r, creates random points $\mathbf{z}$ such as $\mathbf{z} = a\mathbf{s} + (1-a)\mathbf{r}$, a being random numbers. For a fully feasible $\mathbf{z}$ we have $\texttt{eval}(\mathbf{s}) = \texttt{eval}(\mathbf{z}) = f(\mathbf{z})$. If $f(z)$ is greater that $f(r)$, then z replaces r as a new reference point (see [37] for more details).

[11]The implementation of the package has been adapted for this study, by using the compiler ACC 1.4 from Absoft $C/C++$.

generations	$\hat{x}$	$\hat{y}$	$\hat{x}-x^*$	$\hat{y}-y^*$
10	0.625037	0.13819	0.125037	−0.028476
20	0.501075	0.167209	0.001075	0.000543
30	0.487432	0.170932	−0.012566	0.004272
100	0.500091	0.166636	0.000091	0.00003

Figure B.3: Best and exact solutions

References:

[1] J.F. Bard. *Practical Bilevel Optimization: Algorithms and Applications*. Kluwer Academic Publishers, Boston–Dordrecht–London, 1998.

[2] D. Beasley, D.R. Bull, and R.R. Martin. An overview of genetic algorithms: Part 1, fundamentals. *University Computing*, 15(2):58–69, 1993. Available from: citeseerx.ist.psu.edu.

[3] D. Beasley, D.R. Bull, and R.R. Martin. An overview of genetic algorithms: Part 2, research topics. *University Computing*, 15(4):170–181, 1993. Available from: citeseerx.ist.psu.edu.

[4] M.G. Bengtsson. Genetic algorithms. Mathematica 2.2 notebook, National Defense Research Establishment, Linkoping, Sweden, 1993. http://library.wolfram.com/.../MathSource/569/.

[5] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK–New York–Melbourne–Cape Town, 2004.

[6] D. Bunnag and M. Sun. Genetic algorithm for constrained global optimization. *Applied Mathematics and Computation*, 171:604–636, 2005.

[7] R. Chelouah and P. Siarry. Genetic and Nelder-Mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions. *European Journal of Operational Research*, 148:335–348, 2003.

[8] R. Chelouah and P. Siarry. A hybrid method combining continuous tabu search and Nelder-Mead simplex algoriths for the global optimization of multiminima functions. *European Journal of Operational Research*, 161:636–654, 2005.

[9] Y.-W. Chen. An alternative approach to the matrix non-cooperative game with fuzzy multiple objectives. *Journal of the Chinese Institute of Industrial Engineers*, 19(5):9–16, 2002.

[10] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester–New York–Weinheim, 2001.

[11] N. Durand and J-M. Alliot. A combined Nelder-Mead simplex and genetic algorithm. Orlando, Florida, 1999. Genetic and Evolutionary Computation Conference. Available from: http://www.recherche.enac.fr/opti/papers/articles.

[12] K. Jayanthakumaran (Ed.). *Advanced Technologies*. In-teh, 2009. Available from: http://intechweb.org/book.php?id=225.

[13] L. Davis (Ed.). *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

[14] J. A. Freeman. *Simulating Neural Networks with Mathematica*. Addison-Wesley, Reading, Mass.–Menlo Park, Cal.–New York, 1994.

[15] C. Geiger and C. Kanzow. *Numerische Verfahren zur Lősung unrestringierter Optimierungsaufgaben*, chapter Das Nelder-Mead Verfahren. Springer-Lehrbuch. Springer Verlag, Heidelberg–Berlin, 1999. Available from: http://www.mathematik.uni-wuerzburg.de/ kanzow/books.NelMead.pdf.

[16] C. Geiger and C. Kanzow. *Theorie und Numerik restringierter Optimierungsaufgaben*. Springer Verlag, Berlin–Heidelberg–New York, 2000.

[17] M. Gen and R. Cheng. *Genetic Algorithms & Engineering Optimization*. John Wiley & Sons, New York–Chichester–Toronto, 2000.

[18] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Co, Reading,MA–Menlo Park, Cal.–Sydney–Tokyo, 1989.

[19] R.L. Haupt and S.E. Haupt. *Practical genetic Algorithms*. Wiley-InterScience, Hoboken, NJ., 2004.

[20] S.R. Hejazi, A. Memariani, G. Jahanshahloo, and M.M. Sepehri. Linear bilevel programming solution by genetic algorithm. *Computers & Operations Research*, 29:1913–1925, 2002.

[21] J.H. Holland. *Adaptation in Natural and Artificial Systems*. The MIT Press, Cambridge, Mass.–London, 1992.

[22] C. Jacob. *Illustrating Evolutionary Computation with Mathematica*. Academic Press, San Diego, CA, 2001.

[23] Y. Jun, L. Xiande, and H. Lu. Evolutionary game algorithm for continuous parameter optimization. *Information Processing Letters*, 91:211–219, 2004.

[24] A.A. Keller. Fuzzy multi-objective optimization modeling with mathematica. *WSEAS Transactions on Systems*, 8(3):368–378, 2009.

[25] A.A. Keller. Fuzzy multiobjective bimatrix game: introduction to the computational techniques. In N.E. Mastorakis, M. Demiralp, V. Mladenov, and Z. Boikovic, editors, *Recent Advances in System Theory & Scientific Computation*, Mathematics and Computers in Science and Engineering, pages 148–156, Moscow, Russia, August 2009. WSEAS, WSEAS Press.

[26] A.A. Keller. *Optimal economic stabilization policy under uncertainty*. In Jayanthakumaran [12], 2009. Available from: http://intechweb.org/book.php?id=225.

[27] C.T. Kelley. Detection and remediation of stagnation in the Nelder-Mead algorithm using a sufficient decrease condition. NSF DMS 9321938, North Carolina University, Raleigh, N.C, 1997.

[28] T.G. Kolda, R.M. Lewis, and V.J. Torczon. Optimization by direct search: new perspectives on some classical and modern methods. *SIAM Review*, 45(3):385–482, 2003.

[29] J.C. Lagarias, J.A. Reeds, M.H. Wright, and P.E. Wright. Convergence properties of the nelder-mead simplex method in low dimensions. *SIAM Journal of Optimization*, 9(1):112–147, 1998.

[30] D. Lammert. Direct search algorithms. Technical report, Mathematisches Institut, Universitắt zu Kőln, 2008. Available from: https://scai.fhg.de/fileadmin/ArbeitsgruppeTrottenberg/WS0809/seminar/Lammert.pdf.

[31] R.M. Lewis, V. Torczon, and M.W. Trosset. Direct search methods: then and now. *Journal of Computational and Applied Mathematics*, 124:191–207, 2000.

[32] M.A. Luersen and R. Le Riche. Globalized Nelder-Mead method for engineering optimization. *Computers and Structures*, 82:2251–2260, 2004.

[33] N.E. Mastorakis. Solving non-linear equations via genetic algorithms. pages 24–28, Lisbon, Portugal, June 16-18 2005. Proceedings of the 6th WSEAS Inter. Conf. on Evolutionary Computing.

[34] N.E. Mastorakis, I.F. Gonos, and M.N.S. Swamy. Stability of multidimentional systems using genetic algorithms. *IEEE Transactions on Circuits and Systems- 1: Fundamental Theory and Applications*, 50(7):962–965, 2003.

[35] N.E. Mastorakis and M.N.S. Swamy. Design of two-dimensional recursive filters using genetic algorithms. *IEEE Transactions on Circuits and Systems- 1: Fundamental Theory and Applications*, 50(5):634–639, 2003.

[36] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, Berlin–Heidelberg–New York, 1999.

[37] Z. Michalewicz and D.B. Fogel. *How to Solve It: Modern Heuristics, 2nd edition*. Springer, Berlin–Heidelberg–New York, 2004.

[38] Z. Michalewicz and G. Nazhiyath. Genocop iii: a co-evolutionary algorithm for numerical optimization problems with nonlinear constraints. volume 2, pages 647–651, Perth, November 29-December 1 1995. 2nd IEEE International Conference on Evomutionary Computation.

[39] Z. Michalewicz and M. Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, 1(4):1–32, 1996.

[40] M. Mitchell. *An Introduction to Genetic Algorithms*. The MIT Press, Cambridge, Mass.–London, 1998.

[41] J.A. Nelder and R. Mead. A simplex method for function minization. *Computer Journal*, 7(4):308–313, 1965.

[42] I. Nishizaki and M. Sakawa. *Fuzzy and Multiobjective Games for Conflict Resolution*. Physica-Verlag, Springer, Heidelberg, 2001.

[43] M. Sakawa. *Large Scale Interactive Fuzzy Multiobjective Programming*. Physica-Verlag, Springer, Heidelberg–New York, 2000.

[44] M. Sakawa. *Genetic Algorithms and Fuzzy Multiobjective Optimization*. Kluwer Academic Publishers, Boston–Dordrecht–London, 2002.

[45] M. Sakawa and I. Nishizaki. *Cooperative and Noncooperative Multi-Level Programming*. Operations research — Computer Science Interfaces. Springer Science+ Business Media, Dordrecht–Heidelberg–London–New York, 2009.

[46] K.-B. Sim and J.-Y. Kim. Solution of multiobjective optimization problems: coevolutionary algorithm based on evolutionary game theory. *Artificial Life and Robotics*, 8:174–185, 2004.

[47] V.J. Torczon. *Multi-directional search: a direct search algorithm for parallel machines*. PhD thesis, Rice University, Houston, Texas, 1989.

[48] F.H. Walters, S.L. Morgan, L.R. Parker Jr, and S.N. Deming. *Sequential Simplex Optimization*. CRC Press, Boca raton, 1991.

[49] G. Wang, Z. Wan, X. Wang, and Y. Lv. Genetic algorithm based on simplex method for solving linear-quadratic bilevel programming problem. *Computers and Mathematics with Applications*, 56:2550–2555, 2008.

[50] H.-J. Zimmermann. *Fuzzy Set Theory and Its Applications, 4th edition*. Kluwer Academic Publishers, Boston–Dordrecht–London, 2001.