A Spreadsheet Implementation of Aspiration/Reservation Based Decision Support

Jacek Korycki
Clemson University, Mathematical Sciences,
Clemson, South Carolina, USA

Włodzimierz Ogryczak †
Universite Libre de Bruxelles, Mathematiques de la Gestion,
Brussels, Belgium

Abstract

The paper presents a spreadsheet implementation of the aspiration/reservation based decision support. This multi–criteria interactive technique, originating from the reference point method, brings together strict optimization rules with an interactive process based on commonly accepted goal programming control parameters (aspiration levels). Moreover, the interactive process is completely open, like typical *what–if* analysis, thus making the method very well suited for a spreadsheet implementation. The purpose of this paper is twofold. First, we show how easily the method can be implemented as an additional analysis mode in a standard spreadsheet with an optimization capability. Second, we conduct computational experiments to compare the implemented technique with the standard goal programming approaches.

† on leave from Warsaw University, Mathematics and Informatics, Warsaw, Poland
thanks to the grant of the Brussels–Capital Region.

During the past decade electronic spreadsheet software, introduced originally to simplify accounting work, has become a standard analysis tool for business decision makers. Contemporary spreadsheets provide tools for easy implementation of most of the functions of decision support systems (DeSanctis and Gallupe, 1987). First of all, they support data storage functions in a tabular form as well as in the relational database model (e.g., Lotus, 1990). For business decision making, the former is very important as most data has a natural form of rectangular tables and its storage in a relational database is usually inefficient (Choobineh, 1991). On the other hand, the relational database functions are very useful in the process of translation of selected solution (decisions) into implementable business procedures. There is no doubt that contemporary spreadsheets armed with a variety of graphical presentation tools provide the complete user interface for decision support.

The standard spreadsheet analysis tool is *what–if* analysis. With this approach the user tries various values for the cells representing the decision variables (in a hunt–and–peck manner) and then analyzes the updated spreadsheet display in the quest for the most satisfactory choices. Simulation type *what–if* analysis is not efficient enough to tackle more complicated quantitative models. Therefore, some spreadsheets offer the so–called *goal–seeking* analysis (e.g., Gray, 1988). With this capability the user may designate a preferred target value for some outcome cell of the spreadsheet model and invoke the command to find the appropriate value of the specified input cell (decision variable). Thus, *goal–seeking* may be considered as equation solving or a straightforward inverse simulation. Hence, the natural next step is the use of optimization tools to look for the best values of decision variables which correspond to optimal value of the specified outcome cell. Beginning from the pioneering implementations in the mid 1980s (c.f., Sharda, 1985) linear programming, or more generally optimization, has become available in spreadsheets as add–in software (Frontline Systems, 1991; Lindo Systems, 1992) or even built–in functions (Borland International, 1991). However, to meet expectations of the decision makers, optimization techniques should not be restricted to single–objective analysis. Since spreadsheet models allow the user to observe many outcomes with *what–if* analysis, one would expect a capability of the multi–criteria optimization.

Multi–criteria optimization, unlike the single–objective case, cannot offer a unique answer and therefore it cannot be implemented as a simple solver. It requires usually an interactive process to identify the most preferred efficient (Pareto–optimal) solution. Spreadsheets provide an ideal environment for an interactive analysis (Troutt et al., 1991). Certainly, not all multi–criteria techniques match a specific style of the spreadsheet interactive analysis. However, many techniques originated from goal programming or other reference point approaches are well suited for interaction using a spreadsheet.

In this paper we present a spreadsheet implementation of the aspiration/reservation based decision support (ARBDS). This multi–criteria interactive technique, originating from the reference point method (Wierzbicki, 1982), joins the strict optimization rules with an interactive process based on commonly accepted goal programming control parameters (aspiration levels). Moreover, the interactive process is completely open, as in typical *what–if* analysis, thus making the method very well suited for a spreadsheet implementation. The purpose of this paper is twofold. First, we show how easily the ARBDS method can be implemented as an additional analysis mode in a standard spreadsheet with an optimization capability. Second, we make computational experiments to compare

paper may be viewed as a complement to the theoretical paper by Ogryczak and Zaroda (1992). The paper is organized as follows. The methodology used in our implementation is discussed in Section 2. Next, in Section 3, more technical aspects of the implementation are described. In Section 4, a hypothetical ARBDS interactive session on a sample multi–criteria decision problem is presented. Finally, in Section 5, we compare performances of the ARBDS technique and goal programming approaches on the sample problem and on randomly generated test problems.

2 Methodology

Consider a decision problem defined as an optimization problem with k objective functions. For simplification of the formal presentation we assume, without loss of generality, that all the objective functions are to be minimized. The problem can be formulated then as follows

$$\begin{aligned} \text{minimize} \quad & \mathbf{F}(\mathbf{x}) && (1) \\ \text{subject to} \quad & \mathbf{x} \in Q && (2) \end{aligned}$$

where

$\mathbf{F} = (F_1, \ldots, F_k)$ — represents a vector of k objective functions,
Q — denotes the feasible set of the problem,
$\mathbf{x}$ — is a vector of decision variables.

Consider further an achievement vector $\mathbf{q} = \mathbf{F}(\mathbf{x})$ which measures achievement of decision $\mathbf{x}$ with respect to the specified set of k objectives $F_1, \ldots, F_k$. Let $Y = \{\mathbf{q} = \mathbf{F}(\mathbf{x}) : \mathbf{x} \in Q\}$ denote the set of all the attainable achievement vectors. It is clear that an achievement vector is better than another if all of its individual achievements are better or at least one individual achievement is better whereas no other one is worse. Such a relation is called domination of achievement vectors. Unfortunately, there usually does not exist an achievement vector that dominates all the other vectors with respect to all the criteria. Thus in terms of strict mathematical relations, we cannot distinguish the best achievement vector. The nondominated vectors are incomparable on the basis of the specified set of objective functions. The feasible solutions (decisions) that generate nondominated achievement vectors are called efficient or Pareto-optimal solutions to the multi–criteria problem.

It seems to be clear that the solution of multi–criteria optimization problems should simply depend on identification of the efficient solutions. However, even a finite characterization of the efficient set for a real–life problem is usually so large that it cannot be considered a solution to the decision problem. So, there arises a need for further analysis, or rather decision support, to help the decision maker (DM) to select one efficient solution for implementation. Of course, the original objective functions do not allow one to select any efficient solution as better than any other one. Therefore, this analysis depends usually on additional information about the DM's preferences. The DM, working interactively with a decision support system (DSS), specifies the preferences in terms of some control parameters, and the DSS at each interactive step provides one efficient solution that meets the current preferences. Such a DSS can be used for analysis of decision

control parameters the DM can identify every nondominated achievement vector.

Goal programming (GP), originally proposed by Charnes and Cooper (1961), seems to be a convenient generating technique for a DSS. It is, in fact, commonly used in real–life applications (White, 1990). Goal programming requires one to transform objectives into goals by specification of an aspiration level for each objective. An optimal solution is then the one that minimizes deviations from the aspiration levels. Various measures of multidimensional deviations have been proposed. They are expressed as achievement functions. Depending on the type of the achievement function we distinguish (compare Ignizio, 1982): minisum (weighted) GP, minimax (fuzzy) GP, lexicographic (preemptive) GP. If a GP model is used as the basis of a DSS, the aspiration levels can be changed during an interactive analysis as the DM preferences evolve. One of the most important advantages of the interactive GP approach is that it does not (necessarily) require the DM to be consistent and coherent in his/her preferences. Thus the interactive goal programming, as a natural extension of *goal–seeking* analysis, seems to be the most appropriate multi–criteria technique for a spreadsheet based DSS.

Goal programming, unfortunately, does not satisfy the efficiency (Pareto–optimality) principle. Simply, the GP approach does not suggest decisions that optimize the objective functions. It only yields decisions that have outcomes closest to the specified aspiration levels. This flaw of goal programming has led to the development of the quasisatisficing approach. This approach deals with the so–called scalarizing achievement function which, when optimized, generates efficient decisions relative to the specified aspiration levels. The best formalization of the quasisatisficing approach to multi–criteria optimization was proposed and developed mainly by Wierzbicki (1982) as the reference point method. The reference point method was later extended to permit additional information from the DM and, eventually, led to efficient implementations of the so–called aspiration/reservation based decision support (ARBDS) approach with many successful applications (Lewandowski and Wierzbicki, 1989).

The ARBDS approach is an interactive technique. The basic concept of the interactive scheme is as follows. The DM specifies requirements in terms of aspiration and reservation levels for several objective functions. Depending on the specified aspiration and reservation levels, a special scalarizing achievement function is built which while being minimized generates an efficient solution to the problem. The computed efficient solution is presented to the DM as the current solution in a form that allows comparison with the previous ones and modification of the aspiration and reservation levels if necessary. While building the scalarizing achievement function the following properties of the preference model are assumed:

P1. For any individual outcome $F_i(\mathbf{x})$ $(i = 1, 2, \ldots, k)$ less is preferred to more (minimization);

P2. A solution with all individual outcomes $F_i(\mathbf{x})$ satisfying the corresponding reservation levels is preferred to any solution with at least one individual outcome greater than its reservation level;

P3. Provided that all the reservation levels are satisfied, a solution with all individual outcomes $F_i(\mathbf{x})$ equal to the corresponding aspiration levels is preferred to any

Ogryczak and Lahoda (1992) have shown that the implementation techniques of goal programming can be used to model the ARBDS approach. Namely, they have shown how employing lexicographic and minimax GP with appropriate weights one receives a GP achievement function that satisfies all the requirements for the scalarizing achievement function used in the ARBDS approach. The ARBDS approach uses the reservation levels as the second vector of control parameters. The reservation levels can be introduced into the GP model by the following goals (Ogryczak and Lahoda, 1992):

$$F_i(\mathbf{x}) + d_i^- - d_i^a - d_i^r = a_i \qquad \text{for } i = 1, 2, \ldots, k \tag{3}$$

$$d_i^- \geq 0, \quad 0 \leq d_i^a \leq r_i - a_i, \quad d_i^r \geq 0 \quad \text{for } i = 1, 2, \ldots, k \tag{4}$$

$$d_i^- d_i^a = 0, \quad (r_i - a_i - d_i^a) d_i^r = 0 \qquad \text{for } i = 1, 2, \ldots, k \tag{5}$$

where

a_i and r_i — denote aspiration and reservation level for the i-th objective, respectively;

d_i^-, d_i^a, d_i^r — are nonnegative state variables which measure deviations of the current value of the i-th objective function from the corresponding aspiration and reservation levels:

- d_i^- — negative deviation from the aspiration level,
- d_i^a — positive deviation from the aspiration level within the interval between the aspiration and the reservation level,
- d_i^r — positive deviation from the reservation level.

The goals (3)–(5) differ from the typical ones only through the splitting of the positive deviation d_i^+ into a sum of two deviations d_i^a and d_i^r where the first one is limited to the interval between the aspiration and reservation levels, and the second can be positive only if $d_i^a = r_i - a_i$. It is assumed that the aspiration and reservation levels satisfy the natural inequality $a_i < r_i$ $(i = 1, 2. \ldots, k)$.
Finally, the following lexicographic GP problem is formed:

$$\begin{aligned} \text{RGP:} \quad & \text{lexmin } \mathbf{g}(\mathbf{d}^-, \mathbf{d}^a, \mathbf{d}^r) = [g_1(\mathbf{d}^-, \mathbf{d}^a, \mathbf{d}^r), g_2(\mathbf{d}^-, \mathbf{d}^a, \mathbf{d}^r)] \\ & \text{subject to (3)–(5) and } \mathbf{x} \in Q \end{aligned}$$

where

$$g_1(\mathbf{d}^-, \mathbf{d}^a, \mathbf{d}^r) = \max_{i=1,\ldots,k} \{(-\beta d_i^- + d_i^a + \gamma d_i^r)/(r_i - a_i)\} \tag{6}$$

$$g_2(\mathbf{d}^-, \mathbf{d}^a, \mathbf{d}^r) = \sum_{i=1}^{k} (-\beta d_i^- + d_i^a + \gamma d_i^r)/(r_i - a_i) \tag{7}$$

β and γ are arbitrarily defined parameters satisfying inequalities $0 < \beta \leq 1 \leq \gamma$. Due to (4) and (5), parameters β and γ do not affect the minimization of (6) and they can be skipped (replaced with 1) in the formula for g_1. Nevertheless, we keep them in formula (6) to have both the achievement functions g_1 and g_2 based on the same individual quantities, which simplifies the implementation.

nonconvex or discrete problems) complying with the rules of the ARBDS approach (i.e., complying with the preference model defined by properties P1 to P3). Moreover, for parameters β and γ satisfying strict inequalities $0 < \beta < 1 < \gamma$, the requirements (5) can be simply omitted in the constraints of the problem RGP since they are guaranteed by the optimization (Ogryczak and Lahoda, 1992, Proposition 6). In our implementation, we use $\beta = 0.1$ and $\gamma = 10$ as the default values.

3 Implementation

We have implemented our system in Lotus 1-2-3 with What'sBest! (Lindo Systems, 1992) as an optimization add–in. The implementation augments the set of variables of the original decision model by deviation variables d_i^-, d_i^a and d_i^r, adds new constraints of type (3) and (4), and performs two optimization runs. The first run minimizes the objective (6), and the second one — the objective (7) while keeping the value of (6) fixed on the former, optimal level. Of course, an auxiliary variable choosing the maximum in (6) is introduced. This way, two consecutive linear programs are obtained. A special template implementing constraints (3)–(4) and achievement functions (6)–(7) has been prepared. Since the additional goal constraints (3)–(4) use only values of the objective functions from the original model, the template is connected with the original model only through a few cells related to the objective functions $F_i(\mathbf{x})$ in (3). In order to use the ARBDS analysis capability a user only needs to copy the template into a worksheet containing the decision model and specify the objective functions under the analysis. The add–in optimization software (Lindo Systems, 1992) allows linear and mixed integer multi–criteria models to be analyzed. However, since ARBDS applies also to nonlinear problems, it could be implemented for such problems, provided that the add–in software supports nonlinear optimization.

Essentially, most of the ARBDS template is hidden from the user. The user works with only one table prepared as a control panel. The control panel supports the entire interactive analysis process as well as the assignment of the ARBDS tool to the original model (by specification of the objective functions). The control panel has the form of a simple table as in Table 1. Rows of the control panel correspond to several objective functions. They all have the same structure of six columns. The first four columns have to be filled out prior to the interactive analysis to assign the template for the current decision model. The first column (Objective) contains a text cell to be filled with name of the objective function. The second column (Max/Min) is used to specify the objective type. It contains a text cell to be filled with max or min depending on the objective function type. The third column (Activity) allows the user to deactivate some objective functions during the interactive analysis. The fourth column (Value) is designed to show the current value of the objective function. It contains a formula cell which at the beginning has to be filled with the cell address of the objective function in the original decision model. This operation assigns the control panel to the decision model.

Text cell	Text cell	Numeric cell	Formula cell	Numeric cell	Numeric cell
...	...	...	...	...	...
Text cell	Text cell	Numeric cell	Formula cell	Numeric cell	Numeric cell

Table 1. ARBDS Control Panel

Most of our implementation design lies in data structures prepared for the largest potential number of criteria, containing definitions of variables, formulas and cells to store some necessary constants. We have prepared a set of formulas which represent constraints (3)–(4) for the largest possible number of criteria k. This set consists of k disjoint subsets, each referring to a particular criterion $i = 1, 2, \ldots, k$. We have also prepared two cells containing formulas representing objectives (6) and (7), respectively. In order to adapt these structures to the particular decision model we need a mechanism to exclude unnecessary cells associated with inactive criteria, according to the 0/1 switches in the third column of the control panel. Fortunately, this can be done while calling What'sBest! add–in which enables us to exclude certain spreadsheet ranges of cells from the optimization problem. Note that our additional structures are independent of any particular decision model and therefore they are of general use. The only connection with the decision model is present in constraints (3) which are linked to the rest of the model through the addresses of objective functions in the fourth column of the control panel.

The second part of the implementation is a spreadsheet macro. Its main task is to exclude unnecessary ranges of cells, as described above, preparing linear programs for each of two optimization runs and to call the What'sBest! add–in. This task is accomplished with the standard Lotus 1-2-3 commands, like copying, erasing, protecting cells, naming ranges etc. For the first optimization run the cell containing objective (7) is excluded. After the first optimization the auxiliary variable choosing the maximum in (6) is fixed at its current optimal value whereas objective (7) is activated for the second optimization run. The macro is provided in Appendix.

Having assigned the control panel to the decision model one can start the ARBDS interactive analysis as the model will be automatically extended by constraints (3)–(4) built in the template. The user controls the interactive analysis editing aspiration and/or reservation levels, i.e., using the fifth (Aspiration level) and sixth (Reservation level) columns of the panel. These columns contain numeric cells to be filled out with appropriate data. One may also deactivate or activate some objective functions in the third column. Whenever the ARBDS macro is executed, the add–in optimization software is called to solve the corresponding RGP problem and to generate the corresponding efficient solution. Current achievements (values) of the objective functions appear in the fourth column (Value) of the panel whereas current values of the original decision variables (adjustable cells in the What'sBest! terminology) are placed in the corresponding cells of the original model. The DM can perform an open interactive analysis without necessity of any coherent scenario, just like *what–if* analysis. Standard spreadsheet data storage and graphic capabilities allow the DM to store and compare several efficient solutions in his/her most convenient way.

Optimization of each objective function separately is frequently used as the first step of the multi–criteria analysis. It generates the so–called pay–off matrix (Steuer, 1986). Therefore, we have provided in our implementation an additional macro to perform all

containing values of all the objective functions (rows) obtained while solving several single–objective problems (columns) and thereby it helps the DM to understand the conflict between different objectives. Due to the regularization technique used while computing the pay–off matrix, each generated single–objective optimal solution is also an efficient solution of the original multi–criteria problem. The pay–off matrix provides the DM with two reference vectors: the utopia vector and the nadir vector. The utopia vector represents the best values of each objective considered separately, and the nadir vector expresses the worst values of each objective noticed during several single–objective optimizations. Usually, the utopia vector is not attainable, i.e. there is no feasible solution with such objective values. Coefficients of the nadir vector cannot be considered as the worst values of the objectives over the entire efficient (Pareto–optimal) set. They usually estimate these values but they express only the worst values of each objective noticed during optimization of the other objective functions.

4 Sample problem

In order to show an outline of the basic multi–criteria analysis performed with our ARBDS spreadsheet template we use a small sample decision problem. The problem is based on the textbook financial planning model (Shogan, 1988, Section 4.4) but we take into account the multi–criteria nature of the problem whereas the original model was formed as a single–objective linear program. One needs to decide how to invest for 6 months an amount of "idle" cash. There are several investment opportunities characterized by various maturity horizons, interest rates and risk indices. At the time of maturity of some investment its principal and interests are available for immediate reinvestment in other opportunities available at that time. It creates a dynamic network of cash flow which can be described with linear balance equations (compare Shogan, 1988, for details). The spreadsheet modeling technique (compare, Lindo Systems, 1990) allows us to keep the decision model in the lucid form of a rectangular table of decision variables (amounts invested) with rows corresponding to several investment opportunities and columns corresponding to several months, whereas the balance equations are hidden in some additional formulas.

The goal is to develop an investment scenario that maximizes the profit at the end of the 6 month period, minimizes the investment risk, and maximizes the investment mobility. The profit is simply defined as the difference between the final cash outflow and the initial amount of the invested funds. To minimize the investment risk we define for each month the total risk index of invested funds as the sum of current investments weighted with the corresponding risk indices. The objective function (to be minimized) is then the maximum, over all 6 months, of the total risk indices. As a measure of the investment mobility we consider the amount of funds available for an early withdrawal. For this purpose we calculate for each month two quantities: amount of funds available to withdraw in 1 month and amount of funds available to withdraw in 2 months. This allows us to form two corresponding objective functions (to be maximized) expressing the minimum, over all 6 months, of funds available for an early withdrawal in 1 or 2 months, respectively. Thus the decision problem under consideration is finally modeled as the multi–criteria linear program with four objective functions. Hereafter we call the four

Encoding the corresponding indices we get the pay–off matrix as well as the utopia and nadir vectors, as presented in Table 2. We find out that the objective values vary significantly depending on the selected single–objective optimization. Moreover, we recognize a strong conflict between Profit and all the other objectives.

	Optimized objective					
Objective	Profit	Risk	EW–1	EW–2	Utopia	Nadir
Profit	12.36	9.35	9.35	10.80	12.36	9.35
Risk	9.54	1.08	1.08	4.00	1.08	9.54
EW–1	0.00	101.50	101.50	0.00	101.50	0.00
EW–2	0.00	101.50	101.50	103.50	103.50	0.00

Table 2. Pay-off matrix

	Iteration 1			Iteration 2			Iteration 3			Iteration 4		
Obj.	Asp.	Res.	Sol.	Asp.	Res.	Sol.	Asp.	Res.	Sol.	Asp.	Res.	Sol.
Profit	12.36	9.35	10.86	12.36	9.35	10.51	12.36	11.00	10.76	12.36	11.00	11.04
Risk	1.08	9.54	5.26	3.00	4.00	3.62	3.00	4.00	4.19	3.00	5.00	4.94
EW–1	101.50	0.00	51.15	50.00	30.00	37.65	50.00	30.00	26.26	40.00	20.00	20.56
EW–2	103.50	0.00	52.17	50.00	30.00	65.48	50.00	30.00	53.49	50.00	30.00	30.59

Table 3. ARBDS analysis

The interactive analysis is summarized in Table 3. At the beginning we use the utopia vector as the aspiration levels and the nadir vector as the reservation levels, which results in the so–called neutral solution. Analyzing the neutral solution, we find the level of Risk as unsatisfactory whereas the outcomes of both the mobility criteria as better than we expected. Therefore, in Iteration 2, we change the aspiration and reservation levels for those criteria. In effect, we get an efficient solution with a smaller Profit and very good other achievements. Note that the value of EW–2 is even better than the corresponding aspiration level as it could be improved without worsening the other criteria. It just differentiates the ARBDS approach from standard goal programming. We want to find a solution with a higher Profit. Therefore, in Iteration 3, we increase the corresponding reservation level to 11. As a result we get an efficient solution with all the achievements worse than the corresponding reservation levels. Thus we reconsider our requirements and decide to relax the requirements on the risk and one month early withdrawal. It allows us to get, in Iteration 4, an efficient solution with all the achievements satisfying the specified requirements. We decide to accept it as a final solution. The selection of the final solution depends, certainly, on the DM's preferences. Our sample ARBDS session shows, however, how easily the DM can learn the decision problem and control the analysis of the efficient frontier, working in a spreadsheet *what–if* style.

5 ARBDS versus Goal Programming

In the ARBDS approach, reservation levels are used to define, implicitly, weights $w_i = 1/(r_i - a_i)$. One may consider the standard GP approach with such defined weights as an even simpler technique to be implemented within a spreadsheet. In this section, we

Moreover, we allow the reference levels to be used as the second targets thus defining goal programming approach with penalty functions (Romero, 1991, Chpt. 6). In terms of model (1)–(5), the minimized GP achievement functions, we consider, take the following forms

$$g(\mathbf{d}^-, \mathbf{d}^a, \mathbf{d}^r) = \sum_{i=1}^{k} (d_i^a + \gamma d_i^r)/(r_i - a_i) \tag{8}$$

$$g(\mathbf{d}^-, \mathbf{d}^a, \mathbf{d}^r) = \sum_{i=1}^{k} (\beta d_i^- + d_i^a + \gamma d_i^r)/(r_i - a_i) \tag{9}$$

$$g(\mathbf{d}^-, \mathbf{d}^a, \mathbf{d}^r) = \max_{i=1,\ldots,k} \{(d_i^a + \gamma d_i^r)/(r_i - a_i)\} \tag{10}$$

$$g(\mathbf{d}^-, \mathbf{d}^a, \mathbf{d}^r) = \max_{i=1,\ldots,k} \{(\beta d_i^- + d_i^a + \gamma d_i^r)/(r_i - a_i)\} \tag{11}$$

Formula (8) represents the one–sided minisum achievement function minimizing only positive deviations whereas two–sided formula (9) takes into account also negative deviations. In the case of $\gamma = 1$, the standard positive deviations are considered with the corresponding weights equal to $1/(r_i - a_i)$. For $\gamma > 1$ the formulas define double–target achievement functions with additional penalties for deviations exceeding the corresponding reservation levels. Formulas (10) and (11) define similar minimax achievement functions. Note that GP achievement functions (11) and (9) differ from functions (6) and (7) used in the ARBDS approach due to the sign of the final weights assigned to negative deviations d_i^-.

Table 4 summarizes results for the standard minisum GP approach (8) with $\gamma = 1$ to the sample problem from the previous section. Iterations 1 to 4 correspond to those from Table 3 in the sense that the same aspiration and reservation levels have been used. One can easily notice much worse controllability of the interactive process by the aspiration and reservation levels. In the first iteration, using the utopia vector as aspiration levels and the nadir vector as reservation levels, instead of some compromise solution we get the solution corresponding to the single–objective optimization of Risk or EW–1 (compare Table 2). In Iteration 3, despite the change of the reservation level for Profit, we get again the solution from Iteration 2 with the value of Profit below the reservation level whereas all the other criteria reach or exceed their aspiration levels. Similar properties has the solution of Iteration 4. In order to get a solution with a higher Profit, in Iteration 5, we raise the corresponding reservation level to 12. It results in a solution with Profit=11.04 (like in Iteration 4 of ARBDS but much below the current reservation level) and the value of Risk exceeding its reservation level whereas the mobility criteria reach their aspiration levels.

	Iter. 1	Iter. 2	Iter. 3	Iter. 4	Iteration 5		
Obj.	Sol.	Sol.	Sol.	Sol.	Asp.	Res.	Sol.
Profit	9.35	10.24	10.24	10.28	12.36	12.00	11.04
Risk	1.08	3.00	3.00	3.00	3.00	5.00	5.60
EW–1	101.50	50.00	50.00	40.00	40.00	20.00	40.00
EW–2	101.50	78.32	78.32	93.71	50.00	30.00	50.00

Table 4. Minisum GP analysis for the sample problem

minisum GP approach (with $\gamma = 10$) to our sample problem, for the same five iterations as in Table 4. One may notice an improvement in the controllability by the reservation levels in the sense that the solutions are forced to satisfy all the reservation levels if it is possible. However, apart from an improvement in Iteration 4, all main difficulties with controllability pointed out in Table 4 remain valid in Table 5. In particular, we still have not got any compromise solution in Iteration 1, and even the solution from Iteration 4 does not treat the aspiration and reservation levels in such an equal manner as that in Table 3.

Obj.	Iter. 1 Sol.	Iter. 2 Sol.	Iter. 3 Sol.	Iter. 4 Sol.	Iter. 5 Sol.
Profit	9.35	10.24	10.62	11.00	11.62
Risk	1.08	3.00	4.00	5.00	7.17
EW–1	101.50	50.00	41.20	23.55	20.00
EW–2	101.50	78.32	50.00	50.00	30.00

Table 5. Double–target minisum GP analysis for the sample problem

A level of controllability similar to that of the ARBDS approach can be reached with the double–target minimax GP approach. Its achievement function (10) differs from the first level ARBDS achievement function (6) only due to not using negative weights for deviations d_i^-. However, these negative weights and the use of the second level achievement function (7) are crucial to guarantee efficiency of a generated solution (Ogryczak and Lahoda, 1992). The standard GP techniques violate the property P1 and therefore they do not comply with the efficiency principle. We address this issue with analysis of randomly generated test problems. We have generated 20 bicriteria integer problems defined directly in the two–dimensional objective space (i.e. $y_i = F_i(\mathbf{x}) = x_i$). For each problem we have randomly generated 10 sets of aspiration and reservation levels. All the feasible solutions have been included in the square defined by vertices $(0,0)$, $(0,100)$, $(100,100)$ and $(100,0)$. Moreover, there have been randomly generated two inequalities cutting off vertex $(0,0)$; one from vertex $(0,100)$ and the second from vertex $(100,0)$. Aspiration vectors have been generated as integer points in the triangle defined by vertices $(0,0)$, $(0,99)$ and $(99,0)$ thus all them have defined reasonable aspiration levels for goal programming approaches. In fact, 87% of the generated aspiration vectors have been unattainable. The reference levels have been randomly generated as integers satisfying inequalities $a_i + 1 \leq r_i \leq 100$. It results in 44% of the reservation vectors being unattainable. All the random numbers have been generated as uniformly distributed.

Tables 6 and 7 summarize performances of various achievement functions on the randomly generated test problems. We include in this comparison four GP achievement functions (8)–(11), achievement functions (6) and (7) from the ARBDS approach considered separately and, finally, the ARBDS technique based on the lexicographic minimization of (6) and (7). The corresponding rows of the tables report: percent of the generated solutions that have failed the efficiency requirement, percent of the generated solutions that have failed to reach their reservation levels (i.e. $y_1 > r_1$ or $y_2 > r_2$), percent of the generated solutions that have failed to reach their aspiration levels (i.e. $y_1 > a_1$ or $y_2 > a_2$), and the average percent of overlapped (duplicated) solutions generated. One may easily notice that all the GP achievement functions have generated 11–30% nonefficient solutions

technique only function (7) has generated only efficient solutions. However, function (7), which corresponds to the standard weighting approach (Steuer, 1986), has presented very poor controllability by aspiration and reservation levels. All the approaches based on the minimax achievement functions, including the ARBDS technique, have reached all the attainable aspiration and reservation levels. Note that even "flattened" version of the ARBDS technique (with $\beta = \gamma = 1$) have satisfied properties P1–P3. Introducing of parameters $\beta = 0.1$ and $\gamma = 10$ (Table 7) have improved performances of the GP achievement functions but still many solutions have been left nonefficient.

	Achievement functions with $\beta = \gamma = 1$						
	(8)	(9)	(10)	(11)	(6)	(7)	(6)–(7)
Not reached efficiency	15%	30%	17%	30%	20%	0%	0%
Not reached reservations	50%	50%	44%	44%	44%	82%	44%
Not reached aspirations	87%	87%	87%	87%	87%	98%	87%
Overlapped solutions	27%	21%	24%	18%	20%	66%	26%

Table 6. Statistics for randomly generated MIP problems

	Achievement functions with $\beta = 0.1$ and $\gamma = 10$						
	(8)	(9)	(10)	(11)	(6)	(7)	(6)–(7)
Not reached efficiency	11%	26%	13%	21%	15%	0%	0%
Not reached reservations	44%	44%	44%	44%	44%	44%	44%
Not reached aspirations	87%	87%	87%	87%	87%	90%	87%
Overlapped solutions	19%	14%	23%	17%	19%	27%	25%

Table 7. Statistics for randomly generated MIP problems

While implementing GP approaches to our sample problem from Section 4, we noticed a serious modeling difficulty which seems to be overlooked in goal programming methodology (Romero, 1991). It is related to minimax criteria in the original optimization problem (1)–(2). In many decision problems, an objective of type $\max_{j \in J} x_j$ needs to be minimized. It is usually converted into a conventional linear programming form by introducing a variable z to represent the objective value and additional inequalities $x_j \leq z$ for $j \in J$ (Williams, 1993). The inequalities guarantee that z is no less than any x_j ($j \in J$) and by minimizing z it is driven down to the maximum of x_j. Note that the standard GP approach with nonnegative weights violates the property P1 and therefore, when applied to z as an objective function, it may arise that z is no longer equal to $\max_{j \in J} x_j$. For instance, $x_1 = 0.4$, $x_2 = 0.6$, $z = 0.7$ and $d^- = d^+ = 0$ define an optimal solution to the GP problem

$$\min\{w^- d^- + w^+ d^+ : \quad z + d^- - d^+ = 0.7, \quad x_1 \leq z, \quad x_2 \leq z, \quad x_1 + x_2 = 1\}$$

for any nonnegative weights w^- and w^+. In this case, neither z expresses $\max\{x_1, x_2\}$ nor d^- expresses the corresponding negative deviation. Thus, unlike to the ARBDS approach, the standard GP approach cannot be applied to multi–criteria optimization problem (1)–(2) as an external tool without an in–depth analysis and possible rebuilding of the original optimization model.

Spreadsheets allow the decision maker to observe many outcomes during an analysis performed in an interactive way. Thus they seem to be an ideal environment for implementation of interactive multi–criteria techniques for decision support. It is therefore natural to consider an interactive multi–criteria optimization mode as an additional spreadsheet capability (Troutt et al., 1991). Unfortunately, there is a remarkable lack of such tools even as experimental implementations. Certainly, not all multi–criteria techniques match a specific style of the spreadsheet interactive analysis. However, many techniques originated from goal programming or reference point approaches are appropriate for the spreadsheet interaction.

In this paper we have presented a spreadsheet implementation of the ARBDS method. This multi–criteria interactive technique, originating from the reference point method (Wierzbicki, 1982) joins the strict optimization rules with an interactive process based upon commonly accepted goal programming control parameters. Moreover, the interactive process is completely open, as in typical *what–if* analysis, thus making the method very well suited for the spreadsheet implementation. Our work confirms that the goal programming model of the ARBDS method (Ogryczak and Lahoda, 1992) can easily be implemented as an additional analysis mode in a standard spreadsheet with an optimization capability. Even our experimental template implementation preserves the independence of the interactive optimization tool from the original model formulation. The implementation is used to analyze applicability of the bicriteria L_1 Markovitz's portfolio optimization model (Konno and Yamazaki, 1991) to the Polish stock market. Initial experience shows that the ARBDS interaction, which is very similar to the spreadsheet *what–if* and *goal–seeking* analyses, is easily understood by spreadsheet users.

One may consider the standard GP approaches as simpler techniques to be implemented within a spreadsheet. However, our experience shows that even GP approaches with penalty functions fail to generate efficient solutions and provide worse controllability of the interactive analysis. Moreover, as pointed out in Section 5, the GP achievement functions can not always be implemented independently of the original model formulation.

References

Borland International (1991). *Quattro Pro: Version 3.0*, Scotts Valey, Borland International.

Charnes, A. and Cooper, W.W. (1961). *Management Models and Industrial Applications of Linear Programming*, New York, John Wiley & Sons.

Choobineh, J. (1991). "SQLMP: a data sublanguage for representation and formulation of linear mathematical models", *ORSA Journal on Computing*, **3**, 358–375.

DeSanctis, G. and Gallupe, R.B. (1987). "A foundation for the study of group decision support systems", *Management Science*, **33**, 589–609.

Frontline Systems (1991). *What–if Solver*, Incline Village, Frontline Systems.

Gray, P. (1988). *Guide to IFPS/Personal: The Interactive Financial Planning System for Personal Computers*, New York, McGraw–Hill.

Englewood Cliffs, Prentice–Hall.

Konno, H. and Yamazaki, H. (1991). "Mean–absolute deviation portfolio optimization model and its application to Tokyo stock market", *Management Science*, **37**, 519–531.

Lewandowski, A. and Wierzbicki, A.P., (Eds.) (1989). *Aspiration Based Decision Support Systems — Theory, Software and Applications (Lect. Notes in Ecs. & Math. Sys. 331)*, Berlin, Springer.

Lindo Systems (1990). *A BestCase! Scenario Library: Financial Management*, Chicago, Lindo Systems Inc.

Lindo Systems (1992). *What'sBest!: Release 1.77*, Chicago, Lindo Systems Inc.

Lotus Development Corporation (1990). *1-2-3: Release 3*, Cambridge, Lotus Development Corporation.

Ogryczak, W. and Lahoda, S. (1992). "Aspiration/reservation–based decision support — a step beyond goal programming", *Journal of Multi–Criteria Decision Analysis*, **1**, 101–117.

Romero, C. (1991). *Handbook of Critical Issues in Goal Programming*, Oxford, Pergamon Press.

Sharda, R. (1985). "Optimization using spreadsheets on a microcomputer", *Annals of Operations Research*, **5**, 599–612.

Shogan, A.W. (1988). *Management Science*, Englewood Cliffs, Prentice–Hall.

Steuer, R.E. (1986). *Multiple Criteria Optimization — Theory, Computation & Applications*, New York, John Wiley & Sons.

Troutt, M.D., Tadisima, S.K. and Clinton, R.J. (1991). "Interactive optimization aspects of electronic spreadsheet models for design and planning", *Journal of Operational Research Society*, **42**, 349–355.

White, D.J. (1990). "A bibliography on the applications of mathematical programming multiple–objective methods", *Journal of Operational Research Society*, **41**, 669–691.

Wierzbicki, A.P. (1982). "A mathematical basis for satisficing decision making", *Mathematical Modelling*, **3**, 391–405.

Wierzbicki, A.P. (1986). "On completeness and constructiveness of parametric characterizations to vector optimization problems", *OR Spektrum*, **8**, 73–87.

Williams, H.P. (1993). *Model Building in Mathematical Programming — Third Edition*, New York, John Wiley & Sons.

```
\v    /rncwbfree1~c28..j28~
      /rncwbfree2~c30..j30~
      /rncwbfree3~c32..j32~
      /rncwbfree4~c34..j34~
      /rncwbfree5~c36..j36~
      /cd24..h24~d26..h26~
      {if c8=0}{branch crit2}
      /rndwbfree1~
      {if @exact(b8,"max")}/cd25..d25~d26..d26~
      {let d22,(1-2*@exact(b8,"max"))*e8}
      {let d23,(1-2*@exact(b8,"max"))*f8}
      {let l28,1/@min(l26,@max(k26,(d23-d22)))}
      {let k28,k27*l28}
      {let m28,l27*l28}
crit2 {if c10=0}{branch crit3}
      /rndwbfree2~
      {if @exact(b10,"max")}/ce25..e25~e26..e26~
      {let e22,(1-2*@exact(b10,"max"))*e10}
      {let e23,(1-2*@exact(b10,"max"))*f10}
      {let l30,1/@min(l26,@max(k26,(e23-e22)))}
      {let k30,k27*l30}
      {let m30,l27*l30}
crit3 {if c12=0}{branch crit4}

      ...

crit5 {if c16=0}{branch addr}
      /rndwbfree5~
      {if @exact(b16,"max")}/ch25..h25~h26..h26~
      {let h22,(1-2*@exact(b16,"max"))*e16}
      {let h23,(1-2*@exact(b16,"max"))*f16}
      {let l36,1/@min(l26,@max(k26,(h23-h22)))}
      {let k36,k27*l36}
      {let m36,l27*l36}
addr  /ruj26..j26~
      /rncwbmin~j25..j25~
      /rnc\0~start~
      /aiwb~F2
start /rnd\0~
      /rpj26..j26~
      {let j26,j26+l23*@abs(j26)+l24}
      /rndwbmin~
      /rncwbmin~j27..j27~
      /aiwb~F2
```