Cooperative Agents based Multicriteria Decision Aid

Ins Ben Jaafar & Khaled Ghdira
Department of computer science
High Institute of management
41, rue de la libert, 2000 cit Bouchoucha.
TUNISIE

Abstract: - Decision making consists in choosing, on the basis of various criteria, an appropriate subset of actions among a set of alternatives. The most research related to this field proceeds by aggregating all the criteria in a single objective function and rank alternatives according to this scalar measure. Unfortunately, this is often quite inadequate because it risks to alter the final decision. To discard this drawback, outranking based centralized methods have been proposed. However, they remain insufficient because they don't match with the logical distribution of criteria. That is why we propose a distributed approach which finds out the best compromise between all criteria by considering them as cooperative agents. The underlying foundations are detailed and illustrated via both an example and experimentation.

Key-Words: - Multicriteria optimization, decision making, cooperative agents, multi-agent systems.

1 Introduction

Specialists in multicriteria decision aid aim at giving the decision-maker some tools, in order to deal efficiently with decision problems where several-often contradictory- points of views must be taken into account. The first family of tools consists in aggregating the different points of view into a unique function which must subsequently be optimized. Unfortunately this is very quite inadequate, because the criteria generally are incommensurable. They measure different properties that cannot be related to each other directly and cannot be combined into a single function. Indeed such scheme reflects neither real optimization nor the expected result from all criteria points of view[3,13]. The second family aims firstly to build a relation, called an outranking relation, which represents the decision-maker's strongly established preferences and secondly to exploit the outranking relation in order to help the decision-maker solve his problem[3,13]. Furthermore preference modeling and defining the set of decisions are indispensable and complex steps which have been at the center of some research. The third and most recent family proposes methods which alternate calculation steps and dialogue steps with the decision-maker in order to select appropriate and pertinent information on his preferences [13]. Despite the varieties of research in this field, the obtained results remain insufficient. We think that is may be due to the fact that all these methods are centralized. However, the notion of multicriteria decision problem is logically distributed. That is why we propose to explore distributed artificial intelligence and more specifically the multi agent systems that focus on cooperative and distributed problem solving techniques [2,4,5].

The main objective of this paper is to introduce a Multi Agent model for dealing with multicriteria decision-aid, called Cooperative Agents based Multicriteria Decision-Aid "CAMDA" where each agent is responsible for a single criterion. Thus, the criteria are separately optimized without any "scalarization" form. The entities, called Criterion Agents, cooperate and negotiate in order to find out a sub-set of solutions, *compromise sub-set*.

CAMDA is detailed in the second section, illustrated through an example in the third section and experimented in the fourth one. The last section contains concluding remarks and future work.

2 The foundations of CAMDA model

CAMDA consists of agents that cooperate to find out a *compromise-set* of solutions rather than *Pareto optimal* solutions because Pareto optimality alone is not always adequate for pin-pointing the final decision. The set of Pareto optimal choices is often very large and even after eliminating all the alternatives that are not Pareto optimal, the user is usually left with a large number of alternatives to choose from[3]. In the other hand, the compromise set consists of solutions obtained by means of cooperation and negotiation between all criteria. This set is limited, it's size is controlled by the user. Before detailing CAMDA, let us recall some useful definitions.

Definition1: *A multicriteria decision problem is a situation in which, having defined a set A of actions and a consistent family F of criteria on A, one wishes to:*

$$\min_{x \in A} F(x) = \begin{vmatrix} f_1(x) \\ f_2(x) \\ . \\ \\ \\ . \\ f_n(x) \end{vmatrix} \quad n \geq 2 \qquad \text{(MOP)}$$

Where A denotes the feasible set of design alternatives, or the design space[10].

Definition2: *The vector $F(\bar{U})$ is said to dominate another vector $F(\check{U})$, denoted $F(\bar{U}) < F(\check{U})$, if and only if $f_i(\bar{U}) \leq f_i(\check{U})$ for all $i \in \{1,2,...n\}$ and $f_j(\bar{U}) < f_j(\check{U})$ for some j in $\{1,2,...n\}$. A point $U^* \in A$ is said to be Pareto optimal or an efficient point for (MOP) if and only if there does not exist $U \in$ satisfying $F(U) < F(U^*)$. The vector $F(U^*)$ is then called non- dominated or non inferior.*

The above definition qualifies Pareto optimality in the global sense[10,13]. The definition of local Pareto optimality is similar and can be found in [3].

2.1 The architecture of CAMDA model

CAMDA benefits from the multi-agent techniques[2,4,5]that have opened a natural and efficient way to solve diverse problems in terms of cooperation, conflict and concurrence within a society of agents. Each agent is an autonomous entity that is asynchronously able to acquire information from its environment and/or from other agents, to reason on the basis of this information and to act consequently. Within the large domain of multi-agent systems, our model consists of cognitive agents whose number is often limited (here equal to the total number of criteria). The multi-agent architecture consists of Criterion agents cooperating in order to select a sub-set of solutions corresponding to the best compromise from the global satisfaction points of view. However, this class is not sufficient: an Interface is necessary between this society of agents and the user essentially to detect that the problem is solved. Consequently, a second and last class, called Interface, was created. It contains a single individual (figure1).

Consider a Criterion agent Cr_i. It's static knowledge corresponds to the criterion to be optimized f_i (figure1), the set of actions A and the cost of each action with regards to f_i and its ideal action valuation denoted id_i. Note that the ideal action corresponds to the best action according to f_i. Its dynamic knowledge consists of its current satisfaction level denoted sl_i expressing the satisfactory utility level for f_i, its current frequency denoted fCr_i expressing how much time the set of action A has been modified, and its anti-ideal solution valuation denoted aid_i which corresponds to the worst action according to f_i. Note that criterion acquaintances consist of the set of all the other Criterion agents denoted Aq_i.

The Interface Agent has as acquaintances all the Criterion agents. Its static knowledge consists of the set of actions A, whereas its dynamic one consists of the Criterion frequencies and the final result.

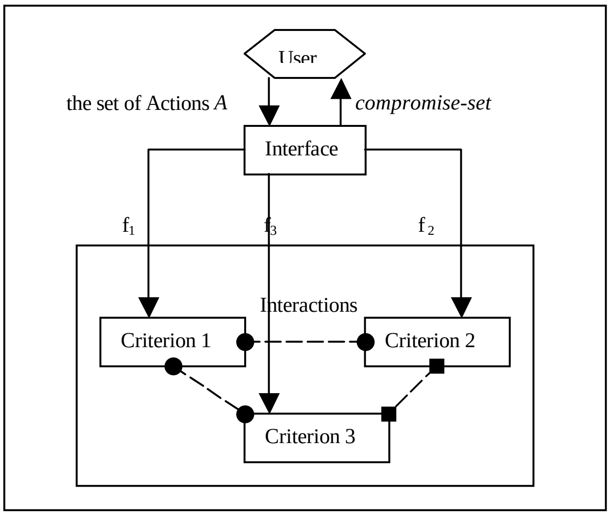

Fig.1. Example of CAMDA architecture with 3 criteria

2.2 The dynamic of CAMDA model

It consists of two phases: dynamic Initialization and Cooperation.

At the first phase, the Interface agent creates n Criterion agents and associates to each one of them a criterion f_i (figure1). Then, each Criterion Cr_i computes, on the one hand, the cost of each action x belonging to the set of actions, A, and on the other hand, both its ideal id_i and anti-ideal aid_i costs as follows:

$aid_i = Min(f_i(x))$ (for maximization), $x \in A$

$id_i = Max(f_i(x))$ (for maximization), $x \in A$

In the second phase, each Criterion agent Cr_i dynamically computes its satisfaction level sl_i,

$sl_i = aid_i + (idi \quad aidi)*\varepsilon$

ε is an adjustment parameter in]0,1[used to gradually increase sl_i (algorithm1. line5) and consequently to reduce the search space namely the set of actions A_i into A_i (algorithm1. line6). If A_i is empty, i.e. the search space has been too much compressed, then Cr_i backtracks and tries to expand it by modifying sl_i as follows:

$sl_i = sl_i-((id_i - sl_i)* \varepsilon./2)$

This process is repeated until A_i is not empty (algorithm1. lines8-10). Hence, each Criterion agent Cr_i extracts a sub-set A_i from A_i. Each solution in A_i has a cost greater than sl_i (in the case of maximization that we adopt here) or lower than sl_i (in the case of minimization). Then A_i is sent to the others criteria Cr_j to be checked (algorithm1. line11).

When receiving A_i, each Criterion Cr_j selects from A_i the sub-set of solutions A_i/f_j which cost f_j is in [aid_j, id_j] and then answers Cr_i by sending it A_i/f_j (algorithm1. line24). As the expected answers are received, Cr_i computes the intersection of the sub-sets A_i/f_j j=1..| Aq_i| (algorithme1. line26). Once Cr_i has received all the answers, it activates the Adjustment process. Three cases may occur:

- *$0\neq$ card (In)>k*, where *k* is a value predefined by the user and corresponds to the maximum size of the compromise set.

In this case, the agent Cr_i chooses the agent Cr_{j0} having the maximal acceptance ratio corresponding to the size of A_i/f_{j0} divided by fCr_{j0} and asks it to compress its current action set. The aim is to satisfy the condition "*card(In)≤ k*". When more than one agent has the same acceptance ratio, Cr_i chooses randomly one of them. While performing a *"Reduce-sub-space*" message (algorithme1. lines 3-11), Cr_{j0} ignores all the future similar incoming messages. Similarly, whenever a Criterion agent Cr_i sends its reduced set of action A_i to be checked, it performs neither *"Reduce-sub-space"* nor *"Expand-sub-space"* (algorithme1. lines 3,12) message until it receives all the expected answers.

Note that *"sendMsg(receiver, sender, "message)"* defines the message "message" transmitted by "sender" to "receiver".

- *$0\neq$ card (In) ≤k*

Given that, the process has reached a sufficiently good compromise namely the best equilibrium state relative to the predefined *k* value. Thus, Cr_i communicates the final result to the Interface that informs the user and interrupts the search process.

- *0= card (In)*

Given that, the agent Cr_i chooses the agent Cr_{j0} having the maximal refusal ratio corresponding to the size of (A_i - A_i/f_{j0}) divided by fCr_{j0} and asks it to expand its current sub-space. When more than one agent have the same refusal ratio, Cr_i chooses randomly one of them. While performing a *"Expand-sub-space"* message, (algorithm1. lines12-19), Cr_{j0} ignores all the future similar incoming messages. Whenever a criterion Cr_j is requested to modify its current set of action A_i, it increments its modification frequency fCr_j and communicates it to the other acquaintances.

Note that the less is *k* , the best is the quality of the *compromise-set*.

3. Illustrative example

Consider a consumer who wants to buy a television set and, after a first selection, retains eight models and then evaluates them by tacking into account the price, quality of picture (PQ), quality of sound (SQ) and maintenance contract (MC) (V.G.= good, G.= good, A.= average, N.G. = not good). (V.G was somewhat arbitrarily replaced by 2, G. by 1, A. by and N. G by -1) [13] .

The data are represented by the table1 where we have modified the prices by changing signs, thereby getting a maximization problem.

Model	- Price	P. Q	S.Q	MC
T1	-1300	V.G/2	V.G/2	A/0
T2	-1200	V.G/2	V.G/2	G/1
T3	-1150	V.G/2	G/1	G/1
T4	-1000	G/1	G/1	N.G/-1
T5	-950	G/1	G/1	A/0
T6	-950	A/0	G/1	N.G/-1
T7	-900	G/1	A/0	N.G/-1
T8	-900	A/0	A/0	A/0

Table1: Data example

```
While not equilibrium state do
    if mailBox not empty then m← getMsg (m)
     case m
      reduce-sub-space (In) :
       aid_i ← generate-anti-ideal-cost (In)
       sl_i ← aid_i + (id_i - aid_i ) * ε
       A _i ← modification-set (In, sl_i )
       f_Cri ← f_Cri + 1
       while (  _i is empty) do
          sl_i ← sl_i   ((id_i -sl_i) * ε/2)
             _i ← modification-set (In, sl_i )
       sendMsg (Aq_i, Cr_i, "request-checking-          _i)")
    expand-sub-space :
        sl_i ← sl_i   ((id_i -sl_i) * ε/2)
           _i ← modification-set ( A_i , sl_i)
        f_Cri ← f_Cri + 1
        while (  _i is empty) do
            sl_i ← sl_i   ((id_i -sl_i) * ε/2)
               _i ← modification-set ( A_i , sl_i)
       sendMsg (Aq_i, Cr_i, "request-checking-          _i)")
    request-checking-solutions(   _i):
       _i / f_j0 ← ∅
     for each solution x_i ∈   _i do
       if f_j(x_i) ∈ [sl_j , id_j ] then  _i / f_j ←  _i / f_j ∪ {x_i}
     sendMsg (Cr_i, Cr_j , "answer-checking-          _i / f_j )")
   answers -checking-solutions(  _i / f_j):
  In ← In ∩   _i / f_j
  count-answers
   if nbr-answers = |Aq_i| then Adjustment (In)
```

Algorithm 1: Behavior of Criterion agent Cr_i

The dominance relation introduced in the definition2 can be represented by the accompanying graph of figure2:

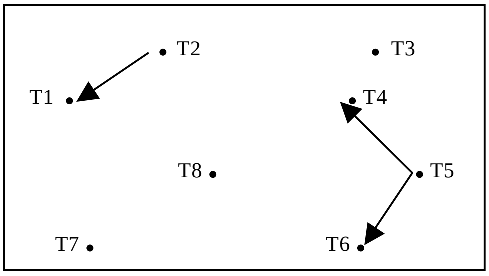

Fig. 2. The dominance relation

Note that if there is an indifference threshold of 50 on the prices, one could be tamped to say that T_2 dominates T_3.

The efficient actions are T_2, T_3, T_5, T_7 and T_8.

The *compromise-set* of actions generated by the CAMDA is presented through the different steps of table2. We take as numerical values ε=20% and K=5.

	aid_i	id_i	sl_i	$_i$
Cr_1	-1300	-900	-1220	{T_2, T_3, T_4, T_5, T_6, T_7, T_8}
Cr_2	0	2	0.4	{T_1, T_2, T_3, T_4, T_5, T_7}
Cr_3	0	2	0.4	{T_1, T_2, T_3, T_4, T_5, T_6}
Cr_4	-1	1	-0.6	{T_1, T_2, T_3, T_5, T_8}
In	--	--	--	{ T_2, T_3, T_5}

Table2: Different steps of CAMDA dynamic

For simplification, the interactions between the different Criterion agents have been omitted namely successive solutions checking. For each agent we have presented only the set of actions A_i satisfying its level sl_i and the global intersection *(In)*. Thus, each Criterion agent Cr_i extracts a sub-set A_i from A_i. Then A_i is sent to the other criteria Cr_j to be checked and this via the message *"request-checking-solutions (A_i)"*. When receiving A_1 for example, each Criterion Cr_j selects from A_1 the sub-set of solutions A_1/f_j which cost f_j is in [aid_j, id_j]. So, Cr_2 determines A_1/f_2={T_2, T_3, T_4, T_5, T_7} and send it to Cr_1 agent via the message *"answer-checking-solutions(A_i/f_2)"*. Similarly for the other criteria agents, the following sub-set of solutions A_1/f_3={T_2, T_3, T_4, T_5, T_7}, A_1/f_4={T_2, T_3, T_5, T_8} are sent to Cr_1. As the expected answers are received, Cr_1 computes the intersection *(In)* of the sub-sets A_1/f_j j=1..|Aq_i|. The size of the intersection set *(In)* is lower than *K*, so the decision process stops and the *compromise-set* of actions{T_2, T_3, T_5} is obtained.

4 Experiments

4.1 Experiment design

The experiments are based on randomly generated binary CSMOPs (Constraint Satisfaction and Multicriteria Optimization Problems)[1]. Constraint Satisfaction and Multicriteria Optimization Problem CSMOP is inspired from Constraint Satisfaction Problems *"CSP"* [6,7,8,11] that only focus on constraint satisfaction. It's a formalism that consists of variables associated with their domains, constraints involving subsets of variables and a set of *c* functions where each function is a performance criterion mapping every solution to a numerical value. The generation is guided by classical CSP parameters[9,12]: number of variables (n), domain size (d), constraint density *p* (a number between 0 and 100% indicating the ratio between the number of the problem effective constraints to the number of all possible constraints, i.e., a complete constraint graph), constraint tightness *q* (a number between 0 and 100% indicating the ratio between the number of couples of values forbidden (not allowed) by the constraint to the size of the domain cross product) and the number of performance criterion (*ncr*).

As numerical values, we use n = 10, d = 6, *p* (resp. *q*) varying from 50% to 65% (resp. 35% to 50%)by steps of 5% to both keep the solutions number reasonable and to guarantee consistent problems. The adjustment parameter *ε* varies in

{10%,20%,30%,40%,50%} whereas *ncr* in {4,6,8}. Regarding *k*, it is set to 10%k_0 where k_0 is equal to the initial size of the set actions *A*.

4.2 Evaluation parameters

CAMDA is assessed through two families of measures : efficiency and quality. In terms of efficiency, we measure the run time requested to reach an equilibrium state. Concerning the quality, it is quantified as follows:

Suppose that the *compromise-set* is denoted by *Cs*

$\forall x_i \in Cs$, $d_i = \sum_{j=1..c} |id_j \quad f_j(x_i)|/id_j$, where id_j is the ideal cost and $f_j(x_i)$ is the cost of x_i according to the Criterion *j*.

Let us define the quality by *d* such that $d= inf_{i=1..|Cs|}(d_i)$.

So, the lower is *d*, the better is the quality. This procedure is used just to assess our model and to yield objectively best solutions.

Thus, we have generated 240 examples corresponding each one to the configuration (p, q, ncr, ε). Due to the non deterministic character of our model, we have performed each example 10 times and then we have taken the average for all parameters.

4.3 Experimental results

Run time versus both tightness and Criterion number (figure3)

As the tightness decreases, the run time increases. This phenomenon can be explained by the fact that the number of solutions grows at low tightness and then the decision process spends too much effort to reach the equilibrium state.

For example, for (p=0.6,q=0.35 and ncr=4) the solution number is equal to 198 whereas it is equal to 15 for (p=0.6,q=0.5 and ncr =4) .

Run time versus both density and Criterion number (figure4)

As the density increases, the run time decreases. In fact, the more the density is, the less the solutions number is and then the equilibrium state is quickly reached. For example, for (p=0.5, q=0.5, nc =4) the solution number is equal to 40 whereas it is equal to 10 for(p=0.6, q=0.5, ncr =4). Notice that for a given couple (p,q), the maximal run time often increases with the Criterion number. Figure 4 presents an almost linear tendency whereas figure 3 gives heterogeneous results. These ones may be explained by the inter-Criteria conflict effect.

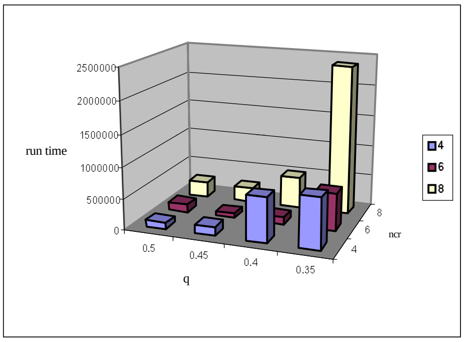

Fig.3. Run time versus both tightness and Criterion number

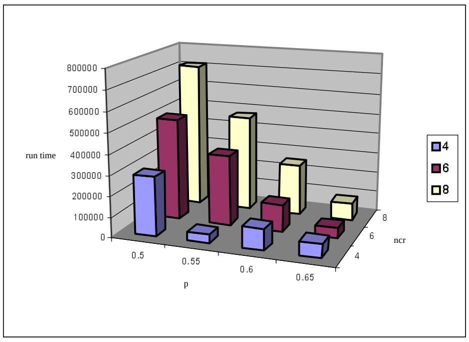

Fig.4. Run time versus both density and Criterion number

Which adjustment parameter for which inter-Criteria conflict?

As before mentioned, the inter-Criteria conflict seems to be important in the decision phase. That is why we perform the following experiments. In fact the aim is to affect the right adjustment parameter ε to the right inter-Criteria conflict parameter.

Let us call this later conflict rate τ and define it by the conflictive criterion number divided by the total number of criteria.

So, let us vary τ in{0.2, 0.4, 0.6, 0.8} and ε in {10%, 20%, 30%, 40%}. Table3 reports the sum of all Criterion frequencies denoted *F*, ($F=\Sigma_{i=1..ncr} fCr_i$), whereas the table4 reports the sum of all Criterion backtracks denoted *B*, ($B=\Sigma_{i=1..ncr}$ backtracks).

ε↓ τ→	0.2	0.4	0.6	0.8
10%	43	11.4	10	12
20%	18	10	10	19
30%	10	16.2	27	32
40%	15.75	31.8	41	24

Table3. The sum of all criterion frequencies F

ε↓ τ→	0.2	0.4	0.6	0.8
10%	20	0.4	0	1
20%	2.4	0	0	9
30%	00	6.2	17	20
40%	05.75	21.8	30	17

Table4. The sum of all criterion backtracks B

Note that *F* and *B* reflect the optimization effort. So, the greater they are and the worst the run time is. Consequently, we select, from Table 3 and 4, for each τ the best ε, i.e., the ε that provides the minimal values of *B*, *F* and *d*.

-Slightly inter-criteria conflict

When τ =0.2, the Criteria are slightly conflictive. Table 3 and 4 shows that *F* and *B* decreases as ε increases. This can be explained by the fact that when *ε=10% or 20%*, the intersection is too dense and must be compressed until it satisfies the condition *"card(In)≤ k"* which increases *F* and *B*. So, it is more appropriate to choose *ε=30% or 40%*. Concerning *d*, the experimental results show that it doesn't change.

-Highly inter-criteria conflict

When τ takes value in {0.4,0.6,0.8}, the Criteria are more conflictive then before. Table 3 and 4 shows that *F* and *B* decrease as ε decreases. This can be explained by the fact that when ε =10% or 20% the solutions corresponding to the different criteria are located in dispersed regions as the criteria are very conflictive. Consequently, one had better choose little ε values, otherwise the intersection may be empty which requires more frequencies and backtracks to both reach an nonempty intersection and the equilibrium state. Concerning *d*, the experimental results also show that it is almost the same.

These results show that the adjustment should be small when the criteria are conflictive, otherwise it should be large. Moreover, other experimental results performed show that the quality, i.e. *d* is as worst as τ increases.

5 Conclusion and future work

In this paper, we have developed an agent based model for dealing with multicriteria decision-aid. In this model each agent is responsible for a single criterion. Thus the criteria are separately optimized without any "scalarization" form by considering them as cooperative agents trying to reach their best equilibrium state which corresponds to the best compromise-set. The effectiveness of the model is demonstrated in the domain of Constraint Satisfaction and Optimization Problems and discussed on randomly generated examples. The experiments have shown, on the one hand, that as the tightness (density) decreases the optimization effort increases, and on the other hand, that the run time often increases with the criterion number. Moreover, they have provided that the quality gets worse as the inter-criteria conflict increases.

As far as our future work is concerned, other experiments will be performed. In addition, we shall extend our model to the dynamic aspect that concerns a restriction (a criterion is removed from the MOP) and/or a relaxation (a new criterion is imposed).

References:

[1] I.BenJaafar and K.Ghedira, Ajustement Multicritre Distribu, *In Proceedings of JFIADSMA'00*, 2000, p 71-84.

[2] S.H.Clearwater, B.A.Huberman and T.Hogg, Cooperative problem Solving, *In Huberman, B., editor, Computation: The Micro and the View*, 1992, p 33-70.

[3] I.Dasand J.E.Dennis, Normal boundary intersection, In *WCSMO-2, Proceedings of the Second World Congress of Structural and Multidisciplinary Optimization*, 1997, p 49-54.

[4] T. Hogg and P.C. Williams, Solving the really hard problems with cooperative search, *In hirsh, H. et al., editors, AAAI Spring symposium on IA and NP-Hard Problems*, 1993, p 78-84.

[5] B.A.Huberman, The performance of cooperative process, *Phisica D*, 1990, p38-47.

[6] P.Jgou, Contribution tude des problmes de satisfaction de contraintes: algorithmes de propagation et de rsolution, propagation des contraintes dans les rseaux dynamiques, *PHD-Thesis*, 1991.

[7] A.Mackworth, Consistency in networks of relations, *Artificial Intelligence*, vol.(8), 1977, p 99-118.

[8] U.Montanari, Networks of Constraints: fundamental properties and applications to picture processing, *Information Sciences* , vol.(7), 1974.

[9] D.Sabin and G. Freuder, Contradicting conventional wisdom in Constraint Satisfaction, *In proceeding of ECAI-94*, 1994, p 125-129.

[10] R.B. Statnikov and J.B. Matusov, Multicriteria Optimisation and Engineering, Chapman and Hall, New York, 1995.

[11] E.Tsang, Foundations of Constraint Satisfaction, *Academic Press*, 1993.

[12] E.Tsang and C.Voudouris, Constraint Satisfaction in Discrete Optimisation, *Unicom Siminar*, 1998.

[13] P.Vincke, Multicriteria Decision -Aid, JHON WILEY & SONS,1989.