18 PREFERENCE BASED SCHEDULING FOR AN HMS ENVIRONMENT

S. Misbah Deen[1] and Rashid Jayousi[2]
[1]*DAKE Group, Computer Science Department,*
University of Keele, Keele, Staffs. ST5 5BG, ENGLAND.
deen@cs.keele.ac.uk
[2] *Computer Science Department,AlQuds University, Jerusalem, ISRAEL*
rjayousi@science.alquds.edu

This paper presents a model for preference-based multi-agent scheduling suitable for Holonic Manufacturing Systems in which holons can cooperate in producing a satisfactory global schedule. The goodness of the scheduling model has been verified by a theoretical behaviour model and confirmed by simulation, using a number of Assembler holons as the scheduler agents of manufacturing tasks. The result of this study, which we found to be satisfactory, has been presented in the paper.

1. INTRODUCTION

In Holonic Manufacturing Systems (HMS) holons (interpreted here as agents) cooperate together to manufacture products. We may assume a coordinator holon has a global task (joint task) to create a product with the help of a set of Assembler holons as cohorts, each cohort scheduling its local component of the global task in an environment where some of the scheduling slots (e.g. time-slots) of each cohort will have been already occupied by previously allocated tasks from other coordinators. From the perspective of the coordinators, the ideal situation is when all the required Assemblers are idle, so that the waiting time in between the Assemblers can be reduced to zero, while from the perspective of an Assembler there should not have any idle time at all, so that it does not lose say financially. Thus contentions are inevitable, there is no global optimum, only negotiated compromises.

Recognising this reality, we have developed a model based on user-defined preferences, which are expressed on resources used in task scheduling, such as machines, time or labour. It is not possible to meet all the preferences due to contention, and therefore we can define the *best solution* as the solution that meets as many preferences as theoretically possible, while satisfying all the constraints. However, since a the theoretical best is not really practicable, we opt for a *good* solution that lies within the upper and lower bounds of theoretical predictions.

In this paper we address the problem of how to derive a preference-based solution in presence of contention, where awarding preferences to the solution of

one task can only be done by depriving/removing preferences from that of another. Deprivation and particularly removal of preferences creates a high non-linearity leading to non-convergence. We propose a general model that produces a good solution in finite time (section 2), backed by a theoretical performance model (section 3) and a simulation study that verifies the correctness of the theoretical model (section 4). This work extends that published in [2].

Preferences are being used in many applications including document ordering, learning and storage in an Web-based environment [1], electronic commerce [4], product design [5], agent-based routing [6], distributed meeting scheduling [7, 10], advanced information retrieval [8], fuzzy ranking [11] and cooperative decision-making [12]. Preferences are used in most of these papers as a simple ranking system, and most of the applications occasionally require human intervention at some point. Also most applications do not guarantee the convergence of distributed computation. None of these papers mentions explicitly the cascading effect or a mechanism for dealing with rescheduling. Numerous researchers use agent technology to resolve the manufacturing scheduling problem. Shen and Norrie [9] give an overview of recent projects, and how they deal with the scheduling.

Our multi-agent approach is based on what we call Cooperating Knowledge-Based Systems (CKBS), in which holons cooperate together in solving a global task through a Cooperation Block (CB) where one holon acts as the coordinator and the others as cohorts. This is an engineering paradigm as opposed to the mentallistic paradigm of distributed AI/Multi-agent systems (DAI/MAS) [3], but it blends ideas from both distributed databases and DAI/MAS.

2. THE SCHEDULING MODEL

We assume a (global) task T, subdivided to lower-level tasks, to be referred to as subtasks (T_1 ... T_n), each subtask T_i having preferences on the resources (strictly speaking resource instances) used for their scheduling. Dependencies among subtasks, including precedents constraints can lead to a heterarchical structure. The task T is executed by the coordination, while each subtask T_i is allocated to an agent (Assembler) A_i. It is often impossible to satisfy all the preferences due the following reasons: (i) contention with the preferences of other agents, (ii) processing cost and (iii) intractability leading to non-convergence.

We assume each subtask to specify preference values on the resources required for the allocation of each subtask. A coordinator can specify very high preference values for its subtasks greedily. To control this greed, we use a market based cost model. The coordinator must state how much it is prepared to pay to achieve its preference. A task then is expressed as follows:

$$T:: [(G_1 .... G_m), (P_1 .... P_n), (V_1 .... V_n) (O_1 .... O_n)]$$

where (G_1 .... G_m) are the set of the task constraints, typically task dependencies, (P_1 .... P_n) are a set of (preferred) resource instances, such as end-times, (V_1 .... V_n) are the corresponding preference values on these resource instances, and (O_1 .... O_n) are the corresponding offer prices – the prices the coordinator is prepared to pay to get these preferred resource instances. In other words, the coordinator is prepared to pay price O_i for resource (instance) P_i with preference value V_i. The offer price has to be

checked against the actual cost (cost price) of the requested allocation as discussed below.

Associated with the allocation of each task T_i is a cost C_T, which is meant to be covered by the offer price. The cost C_T is used to terminate a branch if the agreed cost is exceeded. Each coordinator can accumulate the payments it has received from other coordinators, and use this to buy preference values in the future. C_T can be less than the offer price, depending on the negotiation but no offer is accepted if it is less than C_T. Cost C_T has two components: initial (or basic) cost C_I, and a refinement cost C_R. Component C_I is the cost of finding an allocation for the task ignoring its preferences (on resources). Such an allocation might fortuitously satisfy some or even all preferences. If not, then further processing (called refinement on allocation) over many iterations may be needed to gain more preference values. In any iteration, this task may be reallocated to a preferred resource instance, removing another task from that resource instance. potentially recursively. This cost C_R is the estimated relocation cost of those dislocated tasks, and can be a cascaded cost due to task dependencies. It is expressed as follows:

$$C_T = C_I + \sum_{i=1}^{i=m} C_R$$

where m is the number of refinement needed to find a solution. The basic algorithm is outlined in the following pseudo code, in which an initial total preference value V_t is obtained at first, which is subsequently improved by refinement iterations. At each iteration, multiple candidate allocations are evaluated, from which the one that provides the maximum preference value, say V_m, is selected within the allowed cost (i.e $O_I \leq C_T$). However, this improvement is accepted only if $(V_m - V_t) \geq \Phi$ where Φ is a preference cut-off value (see section 4). If this improvement is accepted, then V_m becomes the new V_t, and a new iteration begins. But if this gain does not exceed Φ, the execution is terminated. In practice, iterations are continued for several more times before termination. The actual process is more elaborate as can be gleamed from section 4.

```
For each coordinator agent:
Get an initial solution, say total preference value Vt
If (initial solution satisfies all the preference values)
        Accept the solution
else{
        Begin an iteration to maximise the preference values
        Seek new allocation by negotiating with other tasks.
        Find the new preference values and costs of all possible new solutions.
        Select from these solutions the one with Vm for (OI ≤ CT)
        If  (Vm – Vt) ≥ Φ,
                Accept this solution, and set Vt to Vm
                Proceed to the next iteration.
       else
                Accept the earlier value of  Vt and Terminate
    }
```

3. THEORETICAL DISTRIBUTION MODEL

Because the scheduling model described above is highly nonlinear, verification using the available mathematical techniques is difficult, as we cannot relate the arbitrary user values, such as the preference values, cost values and preference cut-off values, effectively by a mathematical formula However, we have been able to produce a behavioural model by examining the iterative allocation process and the associated incremental preference gain, as presented below. After each iteration I, the total remaining preference value that is yet to be satisfied is given by R_I. During the allocation process, described in section 2, the final global preference gain, G, gradually decreases per iteration as the iteration number I increases, eventually converging onto a minimal value for the total remaining preference value not achieved. We can express R_I for iteration I in an exponential form as:

$$R_I = A + Be^{-\lambda I}$$

where the constant A is the height of the plateau when $e^{-\lambda I}$ tends to zero. Hence $A = R_{min}$, which is the remaining minimum preference value at $e^{-\lambda I} = 0$, to be referred to as the residue R_{min}. Constant B is R_{max}, which is equal to $(R_I - R_{min})$, and constant λ gives the curvature of the distribution related to the rate of preference gain. Thus the equation can be written as follows:

$$R_I = R_{min} + R_{max} e^{-\lambda I}$$

The initial allocation is made ignoring preferences. This is the zeroth iteration (I = 0) when $R_0 = R_{min} + R_{max}$,. At the next iteration (I = 1), $R_I = R_{min} + R_{max} e^{-\lambda I}$. As I increases, $e^{-\lambda I} \rightarrow 0$ and $R_I \rightarrow R_{min}$.

Our extensive investigation to determine the factors that affect the value shows (not presented here) that predominant factor is the clustering of preferences on resource instances. For example if only one subtask can be allocated to a time-slot say 11 am on a machine. but three subtasks jostle for it, then only one of these can be satisfied even in theory, the other two contributing to preference losses.. We have have captured the clustering effect in our theoretical estimation of R_{min}. In our formulation, we use two parameters, the distribution density d, and the *Preference Reduction Function* ρ, as explained below for preferences on a single resource type, which can be imagined as the end-times for subtasks. If we have t subtasks, all having preferences over the same m (<t) resource instances (eg end-time slots), then the density d = t/m. This is the effect of clustering, which can only be resolved by allocating some of these t subtasks away from m, but still as close to their preferred resource instance as possible, so that the preference loss is minimised. The function ρ is used to determine this preference loss.

Given a subtask, we may assume that its preferred resource instances are placed in a convenient preference-value order. Thus, if the end times are the resources, then the time-slots are the resource instances, which would be in time order. Hence, if a preference value V has been attached to the instance k and if ρ is the percentage decrease for each slot away from k on either sides of k, then V will change to $V(1 - j\rho)$ for both the instances (k-j) and (k+j)). For example if a preference V is expressed for the hourly slot 10 am, then it will be $V(1 - \rho)$ for both 9 am and 11 am slots. Note $(1 - j\rho) = 0$ if $j\rho > 1$.

We assume we have t=2h subtasks with preference over m=2n resource instances, where t > n (see Figure 1). Each resource instance as a time-slot,

preference values decreases on either sides of the most preferred time-slot, this decrease is given by ρ, as discussed above. In order to find the preference loss formula we need to evaluate the average movement from slot 1 at the right side of the midpoint m_1 for the right half slots and hence half the subtasks (t/2).

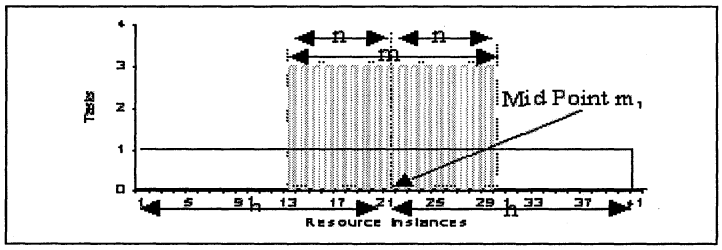

Figure 1- Preference Distribution

In Figure 1 above, more than three subtasks have been given for each slot in the central region, which must be declustered with one subtask per slot (indicated by the lower horizontal line above the X-axis). To do this we move subtasks away from the midpoint, we evaluate the shifts and then calculate the total preference loss, which will give us R_{min}..

The first d subtasks at slot 1 moves to slots 1,2,3,...d, with displacements [0, 1, 2, .., d-1], then the displaced d subtasks moves to slots d+1, d+2, .. 2d, with displacements [d-1, d, d+1, .. 2(d-1)], then the displaced d subtasks moves to slots 2d+1, 2d+2, .. 3d, with displacements [2(d-1), .. 3(d-1)], and so on. Generally speaking the displaced d subtasks moves to

slots (n-1)d+1, (n-1)d+2, .. nd, with displacements
[(n-1)(d-1), (n-1)d, (n-1)(d+1), .. n(d-1)].

The summation of the displacement of each sets of d subtasks:

[0, 1, 2, .., d-1] = (d-1)d/2
[d-1, d, d+1, .. 2(d-1)] = [(d-1)+2(d-1)]d/2= 3(d-1)d/2
[2(d-1), .. 3(d-1)] = [2(d-1)+3(d-1)]d/2= 5(d-1)d/2
.....
.....
[(n-1)(d-1), .. n(d-1)] = [(n-1)(d-1)+n(d-1)]d/2= (2n-1)(d-1)d/2

If we sum the right hand side we get

(1 + 3 + 5 + 7 + ... + (2n-1))(d-1)d/2
= [1 + 2n -1](n/2)(d-1)d/2 = nn(d-1)d/2

This also applies for the subtasks on the left half, hence the total displacement for the whole distribution is n n(d-1)d. To get the average movement per subtask we divide by t = 2nd, the average movement per subtask becomes:

nn(d-1)d/(2nd) = n(d-1)/2

If the preference loss per unit shift is ρ, then the preference loss per subtask is:

ρn (d-1)/2 Eqn 1 (Preference Loss Formulae)

Upper and Lower Bounds of Preference Loss

First assume that the subtasks do not have any precedent constraints and hence can be allocated freely. In that case we can evaluate an weighted average of the total preference loss as follows:

$$[P_1*L_1 + P_2*L_2 + ... P_q*L_q]/q$$

where L_i is the preference loss (in percentage) of Assembler A_i (calculated from the preference loss formula given above), and then P_i is the total preference of A_i. This is the minimal loss based on density d_i in each Assembler A_i and hence is the predicted lowest limit of R_{min}.

Assume that the t subtasks were distributed over s resource instances in iteration 0 due dependencies (including precedent constraints), where $s \geq t$. Therefore, each subtask will occupy on average s/t (≥ 1) slots rather than t/t (= 1) slot. In that case the density $d_i = t/m$ will change into a revised density $D_i = s/m$ for Assembler A_i. Its effect is the same as replacing d by D in Eqn 1, which will then yield a revised estimate of the preference loss that includes the effect of dependencies. This will give the upper limit of R_{min}. Therefore our simulation should yield a value for R_{min} that lies between these two limits.

4. SIMULATION STUDY

In this section we aim to present the results obtained from a simulation study for scheduling in a distributed manufacturing environment using the algorithm described earlier in section 2. We have implemented the scheduling model in a demonstrator using a Java platform.

For the simulation study we have used a set of coordinators CA_1, CA_2, .. CA_n, each CA_I with a (global) task T_i , each task T_i being further subdivided into subtasks: $T_{ij} \mid \{j = 1, 2, .. m\}$, one or more subtasks being allocated to a target agent (say an Assembler holon) A_k. In our implementation we have used up to n = 14, that is, up to 14 coordinators, up to six subtasks (m= 6) in each task and three target agents, each target agent sharing many subtasks of different coordinators. The subtasks have precedent subtasks and their allocation to the target agents can be conjectured as the allocation to Assembler agents in manufacturing. The preferred resource type used in our simulation is end-time slot of a subtask. Each subtask T_i has a preference value V_i and an offer price O_i that the task is willing to pay for preference satisfaction. Also the costs C_i and C_r, are set by the coordinator, and indicate the price that the task should pay for a preferred allocation (see section 2). Our objective is to find a schedule that satisfies as much preferences as possible using our proposed scheduling model.

According to our algorithm initially each subtask is allocated the earliest possible slot that satisfies precedence constraints. In the following iterations the coordinator will accept a negotiation for an exchange if its offer price $O \geq$ the cost C for the expected preference gain g. After negotiations, an actual exchange with preference gain g', where $g \geq g'$ is proposed. Note g' is $(V_m - V_t)$, in the algorithm in section 2. However we use a special preference cut-off value Φ such that if an iteration does not improve the preference gained by at least this Φ, then this gain is rejected, in

order to prevent too many insignificant gains and iterations. Thus the coordinator will accept it if $g' \geq \Phi$, but pay pro-rata to C^*g'/g. If the exchange is unsuccessful, the negotiation will continue for another possible exchange. If no improvement on preferences can be made, allocation is made according to the previous allocation. In our implementation the preference values and offer prices are assigned randomly using the random method available in Java Math package. We have conducted several hundreds of experiments with many permutations and combinations in order to verify the basic properties of our model and to verify the mathematical model described in section 3.

4.1 Verification of Basic Properties

In order to show that the solution converges to the same final value independent of the initial order of subtask processing, we carried out an experiment with 24 subtasks of all mixes but with non-conflicting end times slots (preferred resource). If these subtasks are allocated in the arrival order (which was the end time order) over three target agents without paying any attention to their preferences, 100% preference values will be automatically achieved. We then allocated these tasks in the reverse order without taking any preference into account. This yielded 30% preference gain. On this distribution we applied our model and re-allocated the subtasks, this time (iteration 1) taking preferences into account. This first iteration achieved 100% gain. We repeated this experiment with different initial order, and each case 100% gain was achieved at the first iteration. In these experiments the value of preference cut-off Φ was kept fixed at 5%.

These experiments confirmed that our model behaves, as we expected, and that it does lead to convergence. *A significant point is that this model produces results which are independent of initial allocations (i.e the order of subtask processing), so difficult to achieve in machine scheduling.*

In the next set of experiments we investigated if higher offer prices by some coordinators can distort the results significantly. We have used six coordinators, each task T_i of the coordinator CA_i having m number of subtasks, m varying from 3 to 6. Initially all subtasks are allocated on the first available (time) slots in the (global) task order $T_1, T_2, \ldots, T_6$, at a given offer price, but without considering the

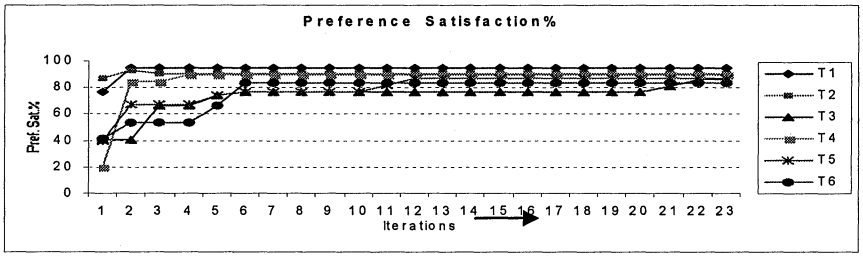

Figure 2 - Preference satisfaction of tasks

preferences. Then a series of iterations was carried out with the same Φ value of 5%. At each iteration, the offer price of one of the coordinators was raised by 5% and its subtasks reallocated, taking only its preferences into consideration.

The results presented in Figure 3 which shows the changes in the preferences satisfied during the allocation of each task (T_1, T_2, ..., T_6) in which only preferences of that task was taken into account. So our conclusion is that higher offer prices do not make any significant change, and therefore our model produces stable preference gain. Next we have examined if the preference gain and the cost of one task is affected by increasing offer prices of the other coordinators. We have conducted this experiment for each task T_i, but selected arbitrarily to show it for T_3 in Figure 3, which was typical for all other tasks.

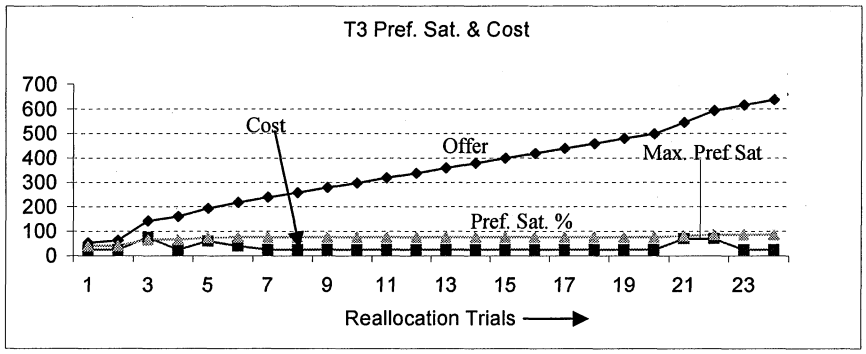

Figure 3 - T_3 preference satisfaction offer & cost variation

We can observe from Figure 3 that the preference gain and cost for T_3, against the accumulated offer prices of the other coordinators. Evidently the increase in their offer prices did not affect the preference gain of T_3 in any significant way. This result is typical of all T_i's.

4.2 Verification of Our Model

These second sets of experiments were carried out to verify that the preference gain over iterations obeys the theoretical model. These results are presented below for two categories:

Observance of the exponential rule at a variable Φ value
Observance of the upper and lower bounds

4.2.1 Variable Φ Value.

In this study we used 14 tasks (coordinators) and 54 subtasks distributed over 3 target agents, and carried out three experiments for different Φ (2%, 5%, 10%), the same set of preference values, and the same preferred end-time slots of each subtask. As shown in Figure 4, the fits on the results of the iterations for preference gain confirm the exponential pattern predicted by the theory.

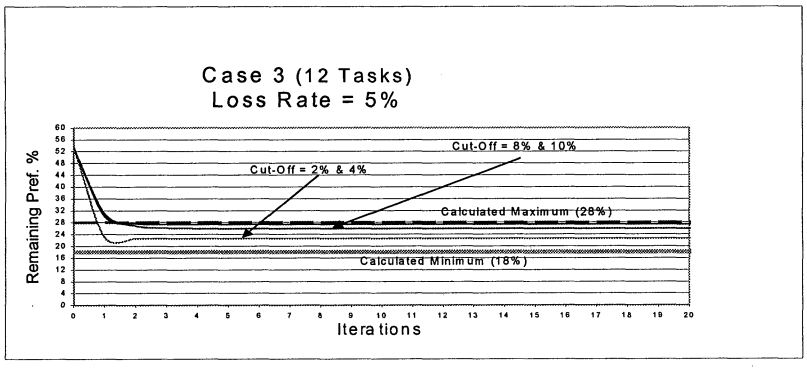

Figure 4 - T_3 preference satisfaction offer & cost variation

4.2.2 Boundaries

The objective of this experiment is to compare the predicted results from the theoretical model with the obtained results from the implementation. In this experiment we used the results from the previous experiments as well as new cases. In some experiments we changed the value of the preference loss rate (explained in the previous section). We used different number of tasks with different number of coordinators, 12, 24, 29, 54 tasks and 4, 6, 7 and 14 coordinators, respectively. For the purpose of this paper we show a summery of part of the is shown in Table 1.

Table 1 - Results Summary

Case	Number Of Tasks	Pref. Loss Rate	Calculated Results		Experimental Results
			Minimum	Maximum	
1	12	5	4.9	11.6	8.5
2	12	10	9.8	23.3	16.7
3	28	5	9.3	27.5	22.8
4	54	5	29.2	58.6	50.8
5	54	5	18.9	28	25.6

Clearly the experimental results on preference loss lie within the upper and the lower bounds of the theoretical predictions.

5. CONCLUSION

In this paper we have presented a cooperative scheduling approach based on user-defined preferences that can be applied in HMS applications. In this context, we have described a distributed allocation technique and a theoretical model to assess its correctness, which we have verified by conducting a simulation study. We have used a cost-based negotiation approach to ensure that the system can converge to a good solution within the upper and lower bounds of our theoretical prediction. The

allocation is independent of initial allocation, it converges, and furthermore the convergence can be achieved faster for crude allocation, should it be desired. We see the potential of applying this approach in HMS and other agent-based cooperative scheduling applications, including distributed project managements. Finally although not part of the HMS project, this work is a spin-off from it. We are indebted to our HMS partners for the various ideas and discussions from which this work has indirectly benefited.

Acknowledgements

The authors are indebted to the partners of the Holonic Manufacturing Systems (IMS/HMS) project for providing inspiration behind this work. One of the authors (RJ) is indebted to Al-Quds University (Jerusalem) for funding this research.

6. REFERENCES

1. K. Burke and K. Aytes: Preference for Procedural Ordering in Distributed Groups..., Proc. of the 34th Hawaii Int. Conference on System Sciences (HICSS-34), IEEE Computer Press (2001)
2. S.M. Deen and R Jayousi: Preference Based Task Allocation in Holonic Manufacturing, the 13th International Conference on Database and Expert Systems Applications (DEXA'02), published by the IEEE Computer Society (2002) 573-577
3. S. M. Deen and C. A. Johnson: Formalizing an Engineering Approach to Cooperating Knowledge-Based Systems. TKDE 15(1) (2003) 103-117
4. K.H. Joo, T. Kinoshita and N. Shiratori: Agent-based Grocery Shopping System Based on User's Preference, Proc of the 7th Int. Conf. on Parallel and Distributed Systems (ICPADS'00 Workshop: Flexible Networking and Cooperative Distributed Agents, IEEE (2000) 499-505
5. T. Keinonen: Expected Usability and Product Preference, Proceedings of DIS'97 Conference Designing Interactive Systems, August, Amsterdam. ACM (1997) 197-204
6. S. Rogers, C. Fiechter, and P. Langley: An Adaptive Interactive Agent For Route Advice, Proceedings of the 3rd Int. Conference on Autonomous Agents, Seattle: ACM Press (1999) 198-205
7. S. Sen, T. Haynes and N.Arora: Satisfying user preferences while negotiating meetings, International Journal of Human-Computer Studies, Academic Press, London (1997)
8. Y. W. Seo and B. T. Zhang: A Reinforcement Learning Agent for Personalized Information Filtering, Proceedings. of International Conference on Intelligent User Interface(2000) 248-251
9. W. Shen, D.H. Norrie,: Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey, An extended HTML version of the paper published in Knowledge and Information Systems (KAIS), an International Journal, 1(2), (1999), 129-156.
10. T. Shintani, T. Ito, and K Sycara: "Multiple Negotiations among Agents for a Distributed Meeting Scheduler", Proc. of the 4th Int. Conference on Multi Agent Systems (ICMAS'2000), poster
11. J. Wang: Ranking Engineering Design Concepts Using a Fuzzy Outranking Preference Model", Fuzzy Sets and Systems 119 (2001) 161-170
12. S. T. C. WONG: "Preference-Based Decision Making for Cooperative Knowledge-Based Systems", ACM Transactions on Information Systems, Vol. 12, No. 4 (1994) 407-435