PROVIDING INTELLIGENT ASSISTANCE IN DISTRIBUTED OFFICE ENVIRONMENTS[1]

Sergei Nirenburg
Colgate University

Victor Lesser
University of Massachusetts

Abstract. We argue that a task-centered, an agent-centered and a cognition-oriented perspective are all needed for providing intelligent assistance in distributed office environments. We present the architecture for a system called OFFICE that combines these three perspectives. We illustrate this architecture through an example.

1. Introduction.

In this paper we describe OFFICE, a system that provides intelligent assistance in the office environment. A schematic diagram of the type of system we are proposing is shown in Figure 1.

In this diagram the office worker operating together with his/her workstation constitute one node in the office problem solving network. The initiative in such a problem-solving environment is mixed: it can be originated by the office worker performing a low-level task or specifying a high-level goal to be accomplished or the office system OFFICE requesting the worker to perform a task. Thus, we see OFFICE as an intelligent assistant to the office worker.

We argue that a task-centered, an agent-centered and a cognition-oriented perspective are all needed for providing intelligent assistance in distributed office environments. We need knowledge from each of these perspectives in order to support not only effective local interaction between OFFICE and the office worker, but also to coordinate cooperative problem solving among the nodes in the system. Coordinating problem solving is an especially difficult task, given the semi-autonomous nature of processing at each node; the bandwidth of the communication channel (which makes it not feasible for nodes to have a complete global view of problem solving in the network); the diversity of the types of knowledge necessary for coordinating and scheduling office activities; and the necessity to provide guidance to the office worker about how to prioritize his own tasks so that they are coherent with the goals of the whole system.

We see the coordination problem as breaking down into a number of subproblems, which include managing resources; equalizing workload distribution; managing goal conflicts; maintaining a proper level of redundancy in task execution and especially in information flow; analyzing dependencies in the sets of goals, plans and events, etc. Automation of any of the above tasks clearly involves manipulation of many types of knowledge, both domain and control.

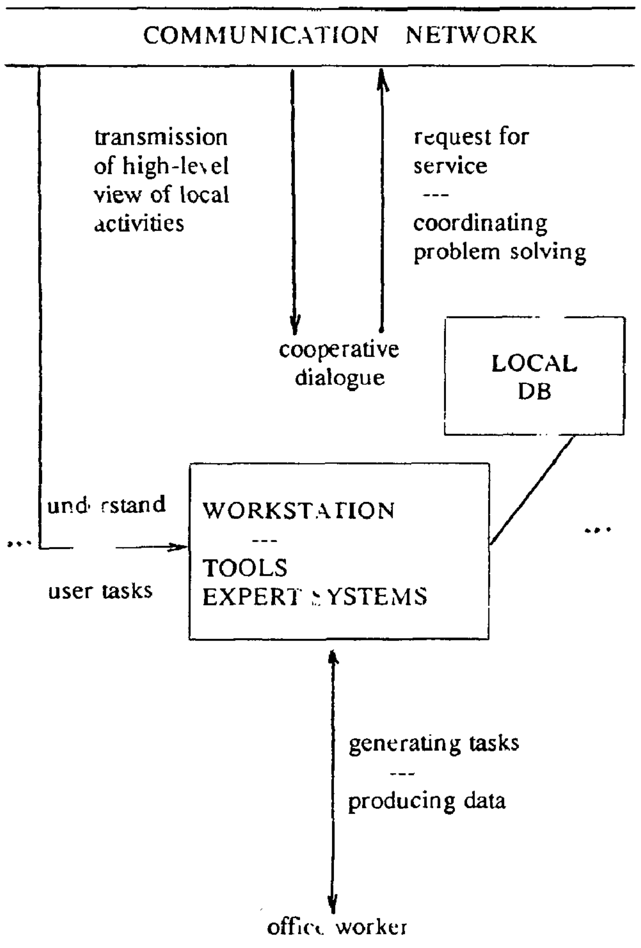
COMMUNICATION NETWORK
transmission of high-level view of local activities
request for service --- coordinating problem solving
cooperative dialogue
LOCAL DB
understand user tasks
WORKSTATION --- TOOLS EXPERT SYSTEMS
generating tasks --- producing data
office worker

Figure 1. A node in a network of cooperative office workstations.

[1] This work was supported by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Force Base, NY 13441-5700, and the Air Force Office of Scientific Research, Bolling Air Force Base, D.C. 20332 under contract number F30602-

To illustrate the problem of local scheduling that takes into account global coherence, consider an office consisting of an executive, E, and his/her secretary, S. Suppose, E is dictating letters to S, and the telephone rings. S answers, and the call appears to be about a very important shipment, and S is asked to provide some information about it. The scheduling choice here is between continuing with the letters (task T1) and performing the request that came over the phone (task T2). We want our system to consider a number of factors here, including the relative importance of the tasks (say, a number of people may be idle in the company because of the lack of raw materials that are to be shipped), the time limitations (suppose, the information is needed before the end of the business day, and it's already 4 p.m.; also, the estimated time of finding the requested information), personal characteristics of S and E, etc. If the secretary were scheduling purely locally, he/she may prefer to schedule T2, but knowing that E will be detained by her doing so, S may prefer T1 based on global coherence considerations. S's knowledge about personal characteristics of E can also be a factor: if E is very conscious of his/her status and importance, then the decision of scheduling T1 is even more strengthened; if not, and if S has the characteristic of being assertive, T2 may be preferred, after an explanation to E.

In what follows we, first, trace the project's genesis from three research projects in connected fields and discuss its functionality. Second, we describe how an office can be modelled in a distributed computer system such as OFFICE and describe its architecture and the basic processing cycle. Finally, we give an example of OFFICE operation where we concentrate on its reasoning capabilities.

The Task-Oriented Perspective.

Our initial effort in developing an expert system in the office domain is the task support system POISE (Croft et al., 1983). POISE has been designed to support office workers in their problem solving activities through the use of plan recognition and planning. In the plan recognition mode the system obtains messages about certain atomic events (such as tool invocations) and tries to determine into which of typical tasks known to the system this event fits. In this manner POISE is able to monitor the activities in an office, predict future activity and detect errors. If, as a result of the monitoring, the system understands the user's task, it can in principle take over its completion. This task completion mode is integrated with the planning mode of operation. In the planning mode POISE is supplied with a typical tasks and its parameters and tries to execute as much of it as possible, based on its knowledge of the task structure and the status of domain objects in a semantic database.

POISE's knowledge takes the form of an hierarchy of typical tasks. Each task is represented by a precondition statement that defines the necessary conditions for its execution; a goal statement that specifies the intended effect of the task; the sequence of subtasks needed to be performed in order to accomplish the task and the constraints among the parameters of the subtasks and those of the task. See Figure 2 for an example.

```
PROC   Purchase-items (Purchasing Amount Items Vendor)
DESC   Procedure for purchasing items with non-state funds.
IS     Receive-purchase-request
       ! (Process-purchase-order | Process-purchase-requisition)
       ! Complete-purchase

COND
       Process-purchase-order.Amount        = Receive-purchase-request.Amount
       OR
       Process-purchase-requisition.Amount  = Receive-purchase-request.Amount
       Process-purchase-order.Items         = Receive-purchase-request.Items
       OR
       Process-purchase-requisition.Items   = Receive-purchase-request.Items
       Process-purchase-order.Vendor        = Receive-purchase-request.Vendor
       OR
       Process-purchase-requisition.Vendor  = Receive-purchase-request.Vendor
       Process-purchase-order.Amount        = Complete-purchase.Amount
       OR
       Process-purchase-requisition.Amount  = Complete-purchase.Amount
       Process-purchase-order.Items         = Complete-purchase.Items
       OR
       Process-purchase-requisition.Items   = Complete-purchase.Items
       Process-purchase-order.Vendor        = Complete-purchase.Vendor
       OR
       Process-purchase-requisition.Vendor  = Complete-purchase.Vendor

WITH   Purchaser   = Receive-purchase-request.Purchaser
       Amount      = Receive-purchase-request.Amount
       Items       = Receive-purchase-request.Items
       Vendor      = Receive-purchase-request.Vendor
```

Figure 2. A plan in POISE

POISE plans are structured so that they in principle allow concurrent execution of subtasks of a task. Straightforward transformation of POISE into a distributed system cannot, however, be performed. Since POISE does not have a developed agent-oriented perspective, there is no way in it to express a fact such as 'requests made by the manager of the office have priority over those made by other workers' or the fact that even though certain workers are better at doing certain types of jobs, if they are not available to do a job of this type, then other workers have to be assigned this responsibility. There is also no way of talking about seemingly independent tasks being actually parts of a cooperative problem solving situation. this includes the considerations of arbitration of competing claims for limited resources.

POISE does not distinguish or reason about the agents' roles and the objects in plans. Thus, for instance, it does not have the possibility to understand that an unusual event happened if it gets the message that the president of a company typed a letter (and not a secretary). Therefore it cannot infer that the secretary may have a day off or that a goal must be instantiated of changing workload distribution among the employees.

Another deficiency of POISE is that the plan recognition and planning architectures are not designed for being distributed and assume a global blackboard and a single locus of control. POISE gives us some ideas about what an intelligent assistant could be but its architecture is not appropriate for use in a distributed environment and it lacks a distributed agent-oriented perspective.

The Distributed Agent-Oriented Perspective.

One of the research areas where we can look for ideas of how to implement the distributed agent-oriented perspective is the field of distributed AI. One of the current approaches there is the study of functionally accurate, cooperative (FA/C) distributed problem solving (Lesser and Corkill, 1983; Corkill, 1982; Durfee et al., 1984, 1985). With this approach, a problem is solved in cooperation by a set of semi-autonomous processing nodes (agents) that may have inconsistent and incomplete local databases. each node independently generates tentative partial solutions, communicates them through a network to other nodes, receives messages (partial solutions, goals, plans and facts) from other nodes, and modifies its processing in accordance with new input. The experience of this work has shown that the control problem is difficult; that the network communication is both difficult and computationally expensive; most importantly, it was found that the key to global coherence is having sophisticated agents who can reason about their own view of processing as well as the views of other agents. They have developed a system in which each node is guided by a high-level strategic plan for cooperation among the nodes in the network. This plan, which is a form of metalevel control, is represented as a network organizational structure that specifies in a general way the information and control relationships among the nodes. Examples of this information include static priorities among local tasks, to whom and what information to communicate and how to prioritize tasks that have been requested by other nodes versus those that were locally generated.

Other work by Smith and Davis (1981) has focused on the knowledge and the protocols necessary for nodes to decide in a distributed way how to allocate subtasks to other nodes. This involves a two-way bidding protocol in which the contractors (taking on the task perspective) and bidders (taking on an agent perspective) communicate to determine the best task allocation.

The work by Lesser et al. focuses on how to do local scheduling given a static task allocation that may redundantly allocate tasks among nodes, while Smith and Davis focus on dynamic task allocation. The office domain requires an integration of both approaches together with augmenting the knowledge used by both approaches for scheduling. The office domain also presents challenges to both approaches because of the tighter and more complex interactions among agents that exists in this domain, compared to the distributed interpretation domain from which both of the above approaches evolved.

The Cognition-Oriented Perspective.

The distributed problem solving approaches described above concentrated on the architecture of the network and the nodes, with the view of organizing the control structure. The types of knowledge necessary for control and communication in OFFICE are studied in the field of cognitive agency research (e.g. Georgeff, 1984, Moore, 1985, but mainly Nirenburg et al., 1985, 1986). The view of the world in this field is that cognitive agents are immersed in a world which is non-monotonic, in the sense that changes in the world can be introduced not only because of the activities of a single agent but also through uncontrolled external events. Agents are capable of a variety of cognitive tasks. They can perceive objects and events in the world. They possess a set of goal types and means of achieving goals of these types: plans. They perform goal and plan generation, selection and execution in complex situations in which many goals and plans coexist and compete for the attention of the agent's conscious processor.

The study of the knowledge that underlies the reasons for particular choices of goals and plans by an agent (in other words, reasons for scheduling and communication decisions) is the central theme of this approach. This knowledge is claimed to involve such factors as personality traits, and physical and mental states of the agent, in addition to the knowledge about the domain situation and the typical tasks and goals. Our approach is to use all the types of knowledge discussed in the cognitive agency approach within the architectural framework inspired by the distributed AI research.

2. An Architecture for a Distributed Office System.

We present here, through an example, an architecture for an intelligent assistance system that integrates the task-, agent- and cognition-oriented perspectives.

2.1. Representing an office.

An office is modelled as a network whose nodes are interpreted as office workers and edges, as communication channels. Every node in the network is a complete problem solver that consists of an office worker and his/her workstation. Following POISE, OFFICE deals with typical activities in a university-based research project (RP), namely: purchasing equipment, hiring and travel. The types of agents in the RP office include Principal Investigator (PI), Research Associate (RA), Graduate Student (GS), Secretary (S), Vendor (V) and Accountant (A). A typical instance of a project may involve 1 PI, 2 RA's, 6 GS's, 1 S, 3 V's (e.g., DEC, Symbolics and TI) and and 2 A's (say, one in Accounts Receivable and one in Personnel).

Figure 3 shows the communication channels for the RP office.

$$\begin{tikzcd}
 & \mathrm{GS} & \\
\mathrm{PI} \arrow[ur] \arrow[rr] \arrow[dr] & & \mathrm{RA} \arrow[ul] \arrow[dl] \\
 & \mathrm{S} \arrow[dl] \arrow[dr, leftrightarrow] & \\
\mathrm{V} & & \mathrm{A}
\end{tikzcd}$$

Figure 3. The network of processing nodes in a model of an RP office. The arrows illustrate authority relationships (see below).

Every node in the office network is aware of its responsibilities to carry out parts of certain plans. They also know who or where from they can and should seek information that is necessary for them to perform their tasks (recall that information about the typical agents for all types of tasks is among the knowledge that every agent possesses).

At any moment t each agent in OFFICE has an agenda of current goals or, more precisely, of current goal instances, as illustrated in (1),

$$\left\{ PU^5_{\sigma_1}, PU^6_{\sigma_1}, HI^1_{\sigma_3}, TR^2_{\sigma_4} \right\}_t \qquad (1)$$

where PU^i, HI^j and TR^k stand for instances of goal types *Purchase, Hire* and *Travel*, and σ_i designate subsets of network nodes that are working cooperatively on particular goals. Intuitively, at any given moment the office workers are pursuing a number of goals, working in teams. Note that some such goals can be in conflict or can compete for resources. Therefore, the agents must have means of resolving these conflicts.

The architecture of an agent in OFFICE is illustrated in Figure 4. A frame-based representation is used for objects, goals, plans and actions, including messages. Plans are represented in extended EDL (cf. Nirenburg et al., 1985). An agent has knowledge about the goals it is typically responsible for as well as about plans that are typically used to accomplish these goals. (If node A has a goal G on its agenda, then A is *responsible* for achieving G.) It also has the knowledge about the current state of its goal agenda, as well as a subset of the contents of other agents' agendas. Scheduling knowledge used by the agent to select goals and plans for processing is represented as a set of condition-action rules. The agent also is aware of the authority relationships in the office, illustrated in Figure 3, that are part of the agent's scheduling knowledge.

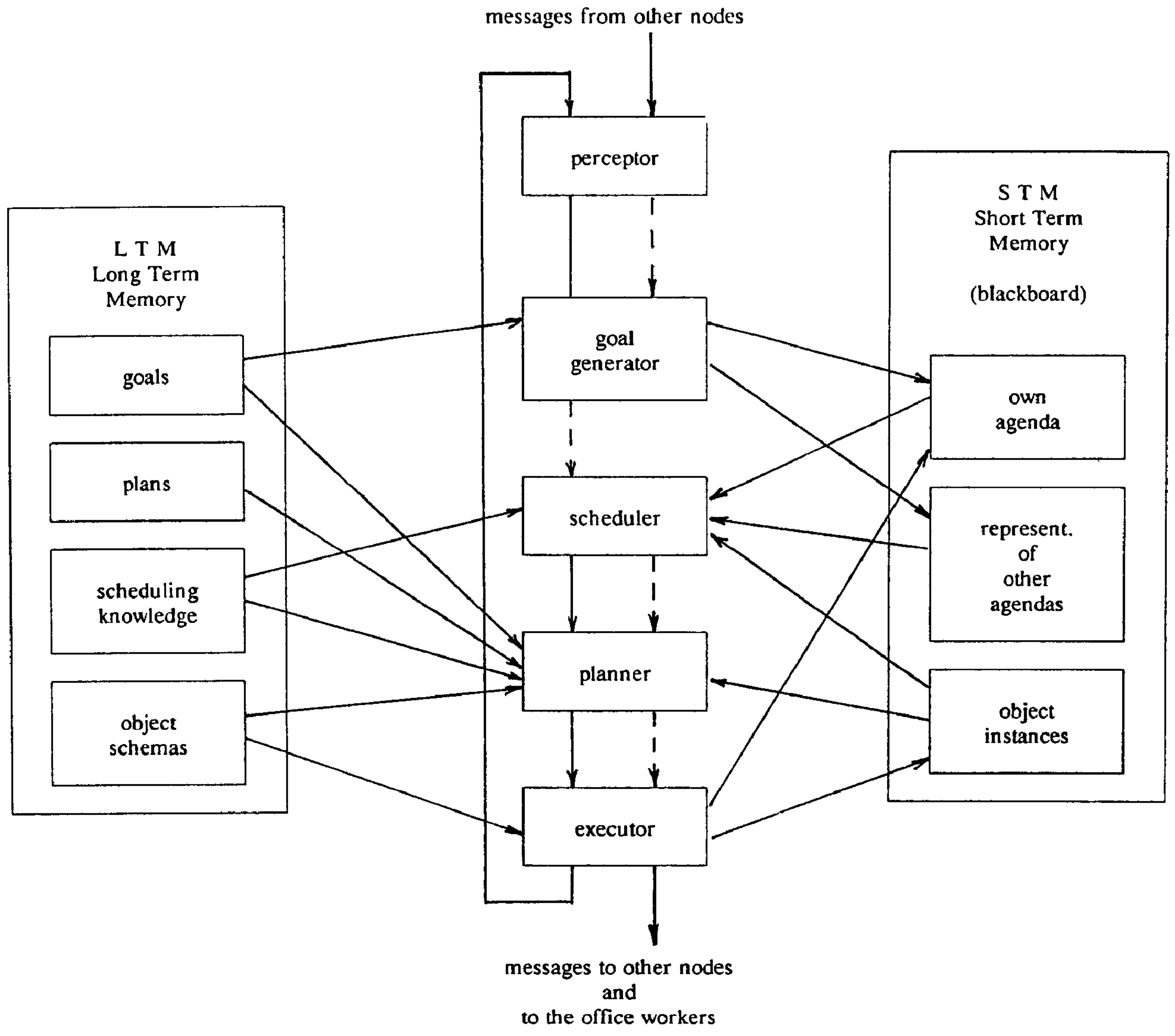

Figure 4. Node Architecture.
——> flow of data
- - - -> flow of control

Representations of two top-level goal types in OFFICE are given in Figure 5. A number ofOFFICE plans are illustrated in Figure 6.

```
(goal HIRE-PERSON
  (Typical-Responsible-Agents RP PI)
   (Typical-Plan Hire-plan)
   (time-scale days)
   (importance 2)
   (beneficiary Research-Project)
   (Supergoals  Conformity-between-Workers-and-Work-Amount
             Use-All-Resources-Available)
   (Trigger (or (sum of expenditures is less than available funds)
            (there are less workers than needed to do work)))
)

(goal PURCHASE
  (Typical-Responsible-Agents  PI)
  (Typical-Plan Purchase-plan)
  (time-scale days)
  (importance 1)
  (beneficiary (PI S RA GS))              ;any member of RP
  (Supergoals  Get-Equipment
            Use-All-Resources-Available)
 (Trigger  (and (there are funds available)
             (the beneficiary's resources are incomplete,
              compared to the typical resources allocated
              to this role-holder)))
)
```

Figure 5. The goals HIRE-PERSON and PURCHASE.

```
(Purchase-plan
   (icon PU)
   (With ((Agent RP.member)
          (Object POBJ) ;is not specified at the moment of
                        ;pl in instantiation
          (Amount int)  ; - " -
       (    ...))
   (is ((specify-item-to-buy (agent = RP.member
                          object = item
                          approx-price = int))
      (make-document (agent = RP.member
                   doc-type = purchase-request
                   object = item))
      (communicate (agent = RP.member
                 destination = Secretary
                 object = purshase-request))
      (plan-selector ((process-purchase-order (agent = Secretary
                                        object = item))
                   (process-purchase-requisition (agent = Secretary
                                             object = item)))
         ;both plans are compound
      (complete-purchase (agent = Secretary
                      object = item))
)
   (preconditions  (Agent has money. Vendor has Object))
   (effects (Agent has less money.
           Vendor has more money.
           Agent has Object))
 )

 (Process-Purchase-Order
    (with (agent = secretary
          object = item
          destination = vendor
          price = int))
     (preconditions (approx-price < $250 ))
     (is (make-document (agent = secretary
                      doc-type = purchase-order
                      object = (item vendor)))
        (communicate (agent = secretary
                    object = purchase-order
                    destination = vendor)))
)

(Complete-Purchase
  (with (agent= secretary
        object = item
        source = vendor))
  (is (# (communicate (agent = vendor
                    destination = secretary
                    object = item))
        (communicate (agent = vendor
                    destination = secretary
                    object = bill)))
     (check-goods (agent = secretary
                object = item))
     (plan-selector ((pay-for-goods (agent = secretary
                                 destination = vendor
                                 object = item
                                 amount = bill.amount)
                  (cancel-goods  (agent = secretary
                                 destination = vendor
                                 object = item
                                 amount = bill.amount))))
)

(make-document              ;a primitive plan
 (with (agent = person
       doc-type = purchase-request | bid-request | purchase-order |
              disbursement-form | item-rejection-form | cv | offer ...
       destination = person | organization
       object = (item, price ...)  ;parameters that are mentioned in
                              ;the document
 )
 (is primitive)
 (effects      )  ;the document exists
)

(check-goods
  (with (agent = person
        object = item))
  (is primitive)
 )

(Communicate
 (with (agent = person
      destination = person
      object = message
      type = assertion | question | order
      instrument = medium))
              ;medium is a list of phone, mail, csnet, etc.
 (is primitive)
 (action-for-primitive))


(Get-Info
  (with (agent = person
      object = message
      was-invoked-by = person3
      instrument medium))
 (is (plan-selector (ask-track (agent = person1
                          destination = person2
                          was-invoked-by = person3
                          instrument = medium))
                find-track (agent = person
                          object = message)))
(effects (communicate (agent = person1
                    destination person3
                    object = message
                    instrument = medium) ))
)
```

Figure 6. A sample of OFFICE plans.

Local and Global Scheduling Knowledge

A special part of the knowledge in OFFICE is the knowledge about scheduling and prioritizing activities by the nodes in the network. A part of the scheduling knowledge is static, that is, is considered true irrespective of the circumstances in which the scheduling takes place. The other portion of the scheduling knowledge is dynamic in that it takes into account the presence of other goals on the node's agenda and the suggests the ways of dealing with goal conflict.

The static part of an agent's scheduling knowledge includes the authority and responsibility structure of the office and the profiles of actual workers in specific roles within the organization. The latter includes both the workers' stated attitudes and preferences with respect to the types of jobs they are performing and their personality profiles, as understood by the current agent, on the basis of which the above attitudes and preferences can be inferred.

The dynamic part of this knowledge includes a snapshot of problem solving activities from the current agent's perspective; a representation of time and other resources; and a set of operational rules that contribute to the task of scheduling. In this paper we will present these rules as a set of scheduling heuristics, bypassing, for the sake of clarity and understandability the actual formalism in which they are expressed. The scheduling heuristics are as follows:

1. Static priorities are stated for all the types of top-level goals. The instances of goals with higher static priorities will be preferred. Thus, for instance, *Purchasing* can be declared more important than *Hiring*.
2. The more time a goal spends on the agenda, the higher the priority it acquires.
3. The less time the accomplishment of a goal will take (as estimated by an agent), the higher the priority it acquires.
4. The smaller the effort needed for the accomplishment of a goal (as estimated by an agent), the higher the priority it acquires. This rule measures effort in terms of both the amount of energy exertion on the part of the agents and the number of intermediate steps (plans) still estimated as needed to accomplish the goal.
5. If a precondition for a plan selected to achieve a goal is false, the goal's priority goes down; however, for specific types of preconditions and nodes a new goal of satisfying this precondition can be established.
6. The higher the authority of the node responsible for a goal, the higher the priority it acquires.
7. Beliefs about the agendas of other network nodes weigh less in the decision process than the contents of own agenda. For example, if the level of authority responsible for a goal G is *inferred* by a node then it will increase the priority of G to a lesser degree than in the case when the authority level was explicitly obtained as input.
8. If the accomplishment of a goal satisfies preconditions for the execution of a plan (or a number of plans) leading to the achievement of other goals (on any of the goal agendas in the network), the priority of the goal is considered higher.

The influence of prioritizing rules based on the above scheduling heuristics is calibrated to produce a general dynamic priority for every goal on a node's agenda.

2.2. How Do the Agents Operate?

A cycle of processing by each agent involves a consecutive invocation of the perceptor, the goal generator, the scheduler, the planner and the executor (cf. Figure 4).

The perceptor

obtains as input (either through the network or from the office worker) *messages* about changes in the world that were received since the previous time cycle (changes are various new states, including results of actions performed by agents in the system).

Input messages are classified according to their *speech act* character. Messages can be either assertions or requests. Assertions can be definitions, opinions, facts, promises, threats and advice. Requests can be questions (request-info) or commands (request-action). Commands are orders, suggestions or pleas. This classification is needed to improve the understanding capabilities of the system (as compared, e.g., with POISE). Also, it allows a clear way of setting goals for the nodes in the network.

Next, the perceptor 'understands' these actions in terms of *plans* they are parts of and, correspondingly, in terms of what was the *goal* that the agent of that action pursued. This step embodies the *plan recognition* activity of the system, since, in the general case, it must understand plans of others in order to perform its own plan production.

The goal generator

updates the agenda of the node's goals due to new inputs. Thus, the arrival of the following input:

```
(message-14
 (instance-of message)
 (speech-act order)
 (sender PI-1)
 (receiver Secretary-33)
 (proposition (communicate Secretary-33
                          Vendor-101
                          'what is the price of desk-22?'
                           Phone))
```

will lead to the generation of the low-level goal instance 'Get-Info-34' that will be fulfilled when the secretary knows the price of the desk. The plan selection for reaching this goal also is specified in the message: using the telephone. 'Get-Info-34' is added to secretary's agenda of goals to accomplish.

There are thus two kinds of sources of goals for every node. One source is the state of the (office) world (if there are more workers than workstations, the goal of purchasing equipment will be generated and put on the office head's agenda). The other source, as in the above example, is messages (requests and orders) from other nodes.

The **scheduler**

selects a goal to pursue from among a number of candidate goals on the agenda. It applies condition-action rules designed on the basis of the above scheduling heuristics and evaluates the current local state of problem solving from the current agent's perspective. After the scheduler finishes operation, one goal from the node's agenda is selected for processing, and control is passed to the planner.

The **planner**

has the task of providing a plan for the achievement of the goal scheduled by the scheduler. If the agent knows of a *canned plan* that typically leads from the current state to the goal state, the planner simply passes the plan to the executor (see below). If more than one plan can be used to achieve a given goal, the planner selects one of them, based on the scheduling rules. The same heuristics that are used for scheduling goals are also used for plan selection. This is in itself a scheduling heuristic.

The knowledge needed by the planner includes the list of plan types, the list of plans that are believed by the node to be instrumental in achieving the goal selected by the scheduler, and the for competing plans.

The **executor**

is called after the planner selects a plan for achieving the current goal.[1] It performs the following sequence of steps:

a) creates an instance of the chosen plan (if such an instance does not already exist) and lists it under the corresponding goal on the agenda.

b) checks preconditions of the plan; if preconditions do not hold (the plan is not immediately applicable) then sets precondition states to be (sub)goal states; puts them on the goal agenda (note that one of preconditions is 'to have values for all non-optional parameters') else expands the agenda tree by substituting the current plan by the sequence of its component plans.

c) if the first subplan in this sequence has the current node as its agent, it is processed by the executor; if another role in the office is the agent of a subplan, the execution of the current plan is interrupted and a value of its 'status' slot is set to 'suspended' and a corresponding message is issued to the agent of the next subplan.

d) if the plan is 'primitive' the actions specified in it are performed. Then the executor checks whether the plan is completed; if yes, the executor reports this, through the communication channels, to the node responsible for supergoal of the goal which the current plan helped achieve. In this way responsibility relationships are both statically and dynamically introduced into the system.

[1] This is a simplification. In reality, planning and execution steps can be interleaved.

3. An Example Run of OFFICE.

We will consider 2 top-level goals: Purchase and Hiring. The processing will be traced from the standpoint of one specific network node, that of Secretary (S). At the beginning of the run S already has a *nonempty* agenda of plans and goals. It also has a representation of agendas of other nodes in the network. This representation may contain mistakes, because it is mainly a result of plan understanding activities of the node. The contents of S's agenda and S's belief about the agendas of a sample of other nodes at the beginning of our manual trace are given in Figure 7.

```
                    -----------------
                    S's own agenda:
                    -----------------

AGENDA ITEM 1:
 Purchase-plan3 (object = terminal)
   communicate (agent = S, destination = PI, object =
                [communicate (agent = V1, object = terminal,
                              destination = S)
                 communicate (agent = V1, object = bill,
                              destination = S)])
   check-goods (agent = PI, object = terminal16)
   plan-selector (agent = S, object =
                [pay-for-goods (agent = S, destination = V1,
                                object = bill)
                 cancel-goods (agent = S, destination = V1,
                                object = (terminal16 bill)])

AGENDA ITEM 2:
 process-purchase-order5 (object = book)
   make-document (agent = S, document-type = purchase-order,
                  object =book, destination = V2)
   communicate (agent = S,object = purchase-order, destination = V2)

              -------------------------------------
              Secretary's beliefs about PI's agenda:
              -------------------------------------

AGENDA ITEM 1:
 Purchase-plan3 (object = terminal16)
   complete-purchase (agent = PI, object = terminal16)

AGENDA ITEM 2:
 Hiring-plan2 (RA)
   evaluate (agent = PI, object = candidate3)
   make-document (agent = S, object = offer, destination = candidates)
   communicate (agent = S, object = offer, destination = candidates)
   select (agent = candidate, object = accept/rej)
   make-doc (agent = candidate, object = accept/rej)
   communicate  (agent = candidate, object = accept/rej)
   plan-selector (agent = S, object =
                [acceptance-track rejection-track])
```

S's representation of RA1's agenda:

```
AGENDA ITEM 1:
  PU1 (object = book11)
    process-purchase-order (agent = S, object = book11)
    complete-purchase (agent = S, object = book11)
```

An agenda item consists of the name of a goal and the names of those of the plans selected for its accomplishment that are not yet (completely) executed, with the bindings for their parameters. Plan names are printed in **bold**. Plan names with numbers appended represent plan instances. The above agendas say that the secretary has the plans to facilitate the purchase of a terminal and to facilitate purchasing of a book asked for by a research associate (Purchase-plan1); S believes PI has plans to hire a research associate (Hiring-plan2) and to facilitate the purchase of a terminal (Purchase-plan3). S also believes that RA1 has the plan of purchasing a book (Purchase-plan1). PI is responsible for both g on its agenda; S is co-responsible for the Purchase-plan3. In contrast, S is responsible only for a subplan of the top-level plan Purchase-plan1. RA1 is responsible for Purchase-plan1.

Figure 7. Sample Contents of the Agendas of an Agent.

Now let us trace the operation of OFFICE through a number of time slices starting with the above state, observing the decision S makes and the changes to its agenda due to new inputs.

------ time slice 1 ------

Suppose, there is a message posted on the secretary S's blackboard : message19 from research associate RA2, of type *order*, that asks to get a price for a desk from vendor V by phone. This message is received by S and a new goal, GET-INFO11, is generated and put on its agenda. S also updates its representation of RA2's agenda by adding there the (inferred) plan of buying a desk. Note that the inferred Purchasing goal is not on S's agenda; therefore, S is not responsible for it.

Next, the scheduler must choose one of the 3 goals on the agenda (PU3 P-P-O5 and GET-INFO11) for immediate processing.

In our example the Get-Information goal will be chosen. This happens because the Purchasing goal is out of contention since it is in the stage of waiting for ordered goods (terminal) to come (Scheduling Heuristic 5). The choice is, therefore, between the Process-Purchase-Order and the Get-Information. P-P-O has, of course, been on agenda for a longer time (Scheduling Heuristic 2), but GET-INFO can be performed by just placing a phone call, while P-P-O requires typing out a form (Scheduling Heuristic 4). There is no rush on the book order, so the goal that can potentially be achieved sooner (Scheduling Heuristic 3) is selected (Scheduling Heuristics 2 and 3 prevail in this case over Scheduling Heuristic 4).

Next, a plan **get-info** is found for achieving the chosen goal; this plan is instantiated and the executor runs its first subplan: **communicate15** (agent = S, object = message34, proposition = message19.proposition, destination = V2, type = question, instrument = phone). As a result of that subplan, the vendor is informed about the question.

------ time slice 2 -----

New inputs: a) Message20: a terminal and a bill arrived from vendor V1 b) Message 21: the price for the book arrived from V2.

The messages are perceived and understood as the execution of specific plans traced on S's agenda: a) refers to the two *communicate* plans that are objects of the next component of the plan chosen for the Purchase3 goal instance; b) is the response to message19 above.

The above messages do not lead to the generation of any new goals. The scheduler now has the following choice: PU3, P-P-O5 and GET-INFO11. P-P-O5 has the same status as at the previous cycle. PU3 is now at a point where the PI must be told that preconditions are met for the execution of the *check-goods* plan (because the terminal arrived). Only one action remains to be performed in GET-INFO11, and that is to relay the information obtained from V2 to RA2.

At this point GET-INFO11 is chosen for the following reasons. S knows that PI is currently in a meeting with a candidate for hiring. Even though the importance of the *check-goods* plan is relatively high (Scheduling Heuristic 1), it cannot be performed at this point (the presence of PI is necessary) and is therefore rated low. GET-INFO11 is closer to completion than the other goals. In accordance with Scheduling Heuristic 3, it is selected, and S sends the plan (**communicate** agent= S, Destination = RA2, Object = Message21.proposition) to the executor.

After this plan is executed, the entire tree for GET-INFO11 is deleted from the agenda.

4. Summary and Status.

We hope we have shown that in order to provide assistance in distributed office environments we need to integrate the agent-centered, the task-centered and the cognition-oriented perspectives. It is important to carefully choose the task and delineate the world corresponding to it. It is equally important to provide an architecture that can support sophisticated scheduling activities by nodes in a distributed problem solving network. At the same time one should try to explore the sources of real-world knowledge that is used as the basis for scheduling. In addition to the observable world situation the scheduling algorithm must have access to the knowledge about the internal states of the processors, or, in other words, the 'personality profile' of the agents to whom the system provides assistance.

The node-level knowledge and processors have been implemented in Zetalisp on a Symbolics 3600 Lisp Machine. We are currently developing the network level of the system.

References

Corkill, D.D., 1982. A Framework for Organizational Self-Design in Distributed Problem Solving Networks. Ph.D. Dissertation, University of Massachusetts, Amherst. (Available as COINS Technical Report 82-33.)

Corkill, D.D. and V.R. Lesser, 1983. The use of meta-level control for coordination in a distributed problem solving network. Proceedings of 8th IJCAI, 748 - 756.

Croft, B.W., L.S.Lefkowitz, V.R. Lesser and K.E. Huff, 1983. POISE: an intelligent interface for profession-based systems. Proceedings of the Conference on Artificial Intelligence, Oakland University, Rochester, MI.

Durfee, E.H., D.D. Corkill and V.R. Lesser, 1984. Distributing a distributed problem solving network simulator. COINS Internal Memo, University of Massachusetts.

Durfee, E.H., D.D. Corkill and V.R. Lesser, 1985. Increasing coherence in a distributed problem solving network. Proceedings of Ninth IJCAI, Los Angeles, August 1985, 1025 - 1030.

Georgeff, M., 1984. A theory of action for multiagent planning. Proceedings of AAAI-84, 121 - 125.

Lesser, V.R., D.D.Corkill, 1981. Functionally accurate, cooperative distributive systems. *IEEE Transactions on Man, Systems and Cybernetics*, SMC-11, 81-96.

Lesser, V.R., D.D.Corkill, 1983. The distributive vehicle monitoring testbed: a tool for investigating distributed problem solving networks. *AI Magazine*, 4, 15-33.

Nirenburg, I., S. Nirenburg and J. Reynolds, 1985. POPLAR: Toward a Testbed for Cognitive Modelling. Technical Report COSC7, Colgate University.

Nirenburg, S., I.Nirenburg and J. Reynolds, 1986. Studying the Cognitive Agent. Technical Report COSC9, Colgate University.

Smith, R.G. and R. Davis, 1981. Frameworks for cooperation in distributed problem solving. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-11, 61-70.