A Planning Algorithm for Distributed Manufacturing

Tibor Gyires and Balakrishnan Muthuswamy

Applied Computer Science Department
College of Applied Science & Technology
Illinois State University, Normal IL 61761

Abstract

Distributed manufacturing (DM), downsizing, and outsourcing are becoming increasingly popular. Processes in manufacturing a product may be distributed at different (geographical) sites. A production plan in such environments consists of various subplans, team coordination schedules, and decision making on task-team assignments. We present a Distributed Artificial Intelligence (DAI) approach that enables people and computers to work cooperatively as teams in decision making. Our model is a set of loosely coupled, autonomous, communicating problem solver nodes representing a functional unit such as design, production, testing, and accounting, that participate in planning, group problem solving and negotiation. A heuristic algorithm assigns subplans to sites to optimize costs.

1. Introduction

Distributed manufacturing (DM), downsizing, and outsourcing are becoming increasingly popular. In distributed manufacturing the plants are geographically distributed. Downsizing is the division of an organization into smaller autonomous units, and outsourcing is buying services (such as information processing) from external contractors.

The move to form a free-trade zone in North America, in Europe, and amongst Pacific-rim countries has the potential to provide today's organizations with more and better options for locating manufacturing sites. In the next few decades we should expect an increase in the extent of distributed manufacturing. Additionally, future organizations will have increased environmental turbulence due to factors such as: instability in growth rate, randomness in the surrounding market demand, and obsolescence rate [Hube87]. These shifting trends in manufacturing environments portend the nature of information systems required to support such organizations. Future organizational information systems are likely to rely heavily on the capabilities and efficiency of distributed computing technology. Product and process planning in such environments needs a dynamic planning information system to support frequent and faster decision making.

In a DM environment sub-parts (or sub-processes) of a product may be produced at different plants. Factors that influence relative advantages of plants include uniqueness, infrastructure, differences in labor quality, production costs, transportation costs, political stability, and State regulations. The success of an organization in a competitive market is dependent on how it plans its manufacturing distribution considering these cost factors. The objective of DM is to reduce costs.

Planning and coordinating the distribution of operations in a DM environment is an important research problem. Information processing to support DM involves linking the computing facilities of several manufacturing plants. An important function of such systems is to support planning and management of the manufacturing process. This task is especially difficult in organizations where production jobs may be assigned to plants on a real time basis, using dynamic data through a networked computer integrated manufacturing system.

This paper presents a Distributed Artificial Intelligence (DAI) approach to provide decision support for planning the manufacturing-processes of a product in a distributed manufacturing environment. We call our model Computer Aided Manufacturing Planning System (CAMPS). CAMPS is a distributed system consisting of sites connected by links. At each

0-8186-3135-X/93 $03.00 © 1993 IEEE

site a Local Planning System (LPS) supports local manufacturing activities. The links denote communication channels between sites.

We assume that a plan to manufacture a product consists of various modular subplans. In an environment that has redundant production capabilities, subplans (or sub-processes) may be assigned to manufacturing sites in several alternative ways. Each of these alternatives has an associated cost. It is desirable that the assignment of subplans to sites is done in a manner that minimizes costs. The goal of CAMPS is to provide real time support for distribution of subplans to manufacturing sites with dynamic decision making needs. We present a DAI algorithm for this purpose, called the δ-algorithm.

The advantages of using CAMPS over traditional planning methods such as liner programming, integer programming, and other operations research approaches is that the approach presented here is useful in dynamic distributed manufacturing environments with several geographically distributed computing sites. Operations Research solutions generally are: a) batch oriented, that is, static and not adaptable to changing environments, b) not appropriate for situations where there are many alternatives to choose from (combinatorial explosion), and c) do not have the capability to learn (improve over time). The CAMPS approach is appropriate in environments where assumptions of operations research approaches (such as linearity) may not be valid. In such cases, a heuristic approach to minimizing operational costs is an alternate method that better approximates the dynamics of a distributed planning environment. The algorithm presented in this paper uses heuristics about the application domain to overcome these limitations and is appropriate for planning in a DM environment.

In section 2 we briefly review some past works in areas related to this paper. Section 3 presents the CAMPS model and a planning algorithm. Conclusions and areas of future research are discussed in section 4.

2. Literature review

The CAMPS project was motivated by several previous works in the areas of planning and DAI. There has been a lot of research attention in the area of DAI during recent years. Most of these works are not independent of one other. For example, a planning methodology may use concepts from works classified as negotiation systems, distributed problem solving networks, or multiagent design. Due to page limitations our review of literature is brief. We present an overview of five DAI problem solving methods pertinent to this paper, and a few example systems. The more interested reader may refer to recent sources in this area (for example, Bond88, Gass89, Ras89, Ras90, Ras91, and SMC92).

Blackboard systems

These systems provide a central data structure called a blackboard, which is divided into regions or levels. Processes called knowledge sources read and write some levels supervised by a control system [Haye85]. Key issues include indexing schemes for retrieving knowledge, synchronization, and conflict resolution among knowledge sources.

Distributed Problem Solving Networks (DPSN)

A DPSN is a set of independent autonomous problem solver nodes that communicate with each other. Each node has the knowledge and intelligence needed to solve only sub-problems of a complex problem and to communicate the solutions of the sub-problems. Problem solver nodes work together to solve complex problem by individually solving the sub-problems and combining their sub-problem solutions into an overall solution.

Multi-agent planning

In a framework developed by [Geor83], a node acts as a group planner and the other nodes send this node all pertinent information. The planning node forms and distributes a multi-agent plan. Coordination is controlled by a global plan specifying all actions and interactions between nodes.

Functionally Accurate/Cooperative Systems

In these systems, each node has incomplete input data while exchanging the intermediate result of its processing with other nodes to construct a complete solution. Nodes cooperate to iteratively eliminate erroneous intermediate results and to converge to a complete and consistent solution [Less81].

Partial Global Planning (PGP)

Durfee and Lesser [Durf87] use the idea of Partial Global Plans (PGPs) in which global plans are built from individual node plans that describe the node's goals and long-term schedule of activities. The nodes exchange information to identify cases where their goals and plans interact based on their view of group activity.

Other approaches

Pan et al., describe a system to manage operations in a geographically dispersed manufacturing enterprise [Pan91]. Their approach is based on the vision of human agents interacting with a large number of computerized intelligent agents which perform some task functions and invoke the services of other agents when necessary. The enterprise operations are divided into elementary tasks and assigned to an agent for execution.

The Polymer system [Crof87] has been designed to support cooperative activities such as making decisions, communicating, creating, updating, and retrieving information in a distributed environment like in an office. The characteristics of these activities are that they are loosely structured, can have many exceptions, and change frequently. Polymer uses an integrated representation of cooperative activities between human and computerized agents. The planner supports sharing of responsibilities between several agents for achieving goals of a plan. If an agent does something unexpected that causes a problem in the plan, the Polymer system initiates a negotiation between agents to resolve the problem.

A multistage negotiation paradigm for planning permits an agent in a distributed problem solving system to acquire enough knowledge to reason about the impact of local activity on nonlocal state and to modify its behavior accordingly. The paradigm presented in [CONR86] is demonstrated in monitoring and control of a complex telecommunications system. The key element of the multistage negotiation is the ability to detect subgoal interactions in a distributed environment and reason about their impact.

A general theory of planning is described in [Morg87]. Agents who operate in a complex environment have to develop plans based on incomplete knowledge. In order to accomplish a goal agents (human or computer) should be able to get the information that they need to perform specific actions. In such situations a successful agent must incorporate actions into his plan to obtain the required information. These plans become more specific as the agent proceeds through his plan.

3. The CAMPS model

CAMPS may be considered as a naturally distributed AI system, a distributed problem solving network, where users, expertise, data, and control are distributed in time and/or space. CAMPS supports cooperation among intelligent agents working on tasks that involve agent interactions such as group decision making, planning, and problem solving. CAMPS is a distributed information system that is modeled as a set of loosely coupled, autonomous, communicating sites that participate in planning, group problem solving, and negotiation activities. Loosely coupled means that the agents do not necessarily belong to the same organization. They can be part of other enterprises, suppliers, subcontractors, or customers. In this paper, we focus on a specific component of CAMPS - an algorithm within the CAMPS planner module. In this section we present a description of the concepts, terms, and notations used in this paper.

CAMPS network

The structure of CAMPS is a network of sites and links. Figure 1 is an example CAMPS network.

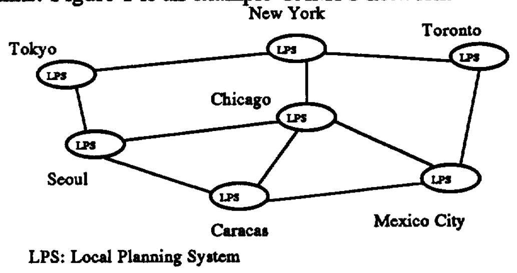

Figure 1: A network of distributed manufacturing plants

Depending on the nature of distribution of the organizational functions, a site may specialize in specific manufacturing functions such as design, inventory control, scheduling, accounting, marketing, prototyping, production, testing, and purchasing. A schedule for manufacturing a product in such an environment includes the distribution of activities amongst the sites. The objective is to obtain the least cost schedule to manufacture a product or set of products.

Plans and subplans

We assume that a **plan** for producing a product is comprised of several activities and processes distributed over the CAMPS network. Distribution of activities is needed because sites are limited in their resources and therefore in the activities that they can perform.

We define a plan as a set of all elemental activities required to achieve the goals of that plan. These activities may be modularized into subplans in a manner that makes sense to the organization. Thus a plan is comprised of several **subplans**. To execute a plan all its subplans need to be executed.

We assume that there is a need for real-time reallocation of plans and resources necessitating communication between sites/LPSs. Also, we assume functional redundancy amongst the manufacturing sites - the same function may be provided by more than one site. Furthermore, we assume that the LPSs have information about the sequence of subplans and so do not model this explicitly. It is immaterial to the approach of the paper whether activities of a plan are fully automated or involve human interaction for decision making, in fact we assume human interaction by planning experts. The execution of a LPS plan could be done by a single person, a group, an automated production unit, a workcell, or a combination of these.

Planning process

The planning process consists of initially assigning a plan P to a site that is capable of managing the plan. P is then expanded into first level subplans P1, P2, ... , Pn, and each subplan is assigned to a qualifying site (qualifying implies that the site has the resources to manage the subplan). Subsequently, a subplan may be further expanded into more subplans and assigned to sites. A site may be associated with several plans and subplans.

The planning process at each site consists of four activities:

1) creating initial plans,
2) formulating subplans,
3) forming partial global plans, and
4) coordinating the execution of subplans and if necessary reformulating subplans.

Let $\mathcal{S}$ = {S} be the set of sites. For example, consider the plan P in Figure 2. Let P be assigned to a site $S_1 \in \mathcal{S}$. S_1 decomposes P into subplans P_1, P_2, and P3. Assuming that P_1, P_2, and P_3 are not performed locally, S_1 tries to find other sites S_2, S_3, and S_4 (S_2 may be the same as S_3 or S_4), from $\mathcal{S}$ to perform P_1, P_2, and P_3. S_1 coordinates and integrates the results of P_1, P_2, and P_3.

P-trees

We represent a plan by a **P-tree** (plan tree) denoted $\mathcal{P}$ = {P, L_A}, where the set P is comprised of the (initial) plan and its subplans. Each P is viewed as an aggregation of several independently executable subplans. L_A is the set of AND-links that connect a plan to its subplans. The root of a P-tree is the initial plan. To execute a plan all its subplans need to be executed. For example, in Figure 2, to execute a plan P, a site needs to execute subplans P_1, P_2, and P_3. Likewise, to implement P_1, P_2, and P_3, P_4, P_5, P_6, P_7, and P_8 need to be executed.

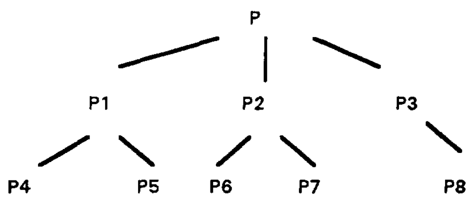

Figure 2: A plan tree (P-tree)

PS-trees and vertices

The assignment of a plan (or subplan) P to a qualifying site S is indicated by V(P,S). In the CAMPS network, sites have functional redundancy and resource limitations. Therefore, a subplan (or plan) may be alternatively assigned to any one of several qualifying sites.

The set of all alternative subplan-site assignments is represented by an AND/OR tree called **PS-tree** (Plan-Site tree) denoted by $\mathcal{V}$ = {{V(P,S)}, $L_A U L_O$}. $\mathcal{V}$ consists of the set of vertices {V(P,S)} and the set of links L_A U L_O that connect pairs of vertices. The initial plan P is in the root of $\mathcal{V}$. Each vertex of a PS-tree represents a subplan-site[1] assignment. L_A is the set of AND links of the P-tree $\mathcal{P}$. L_O is the set of OR links that connect a parent vertex with its child vertices. Only one of these OR child vertices need be

[1] For clarity, we refer to nodes of a P-tree as nodes, that of a PS-tree as vertices, and the nodes of CAMPS network as sites.

executed. Vertices in a PS-tree corresponding to elementary activities without any further subplans are called terminal vertices. Figure 3 depicts an example PS-tree corresponding to the P-tree of Figure 2.

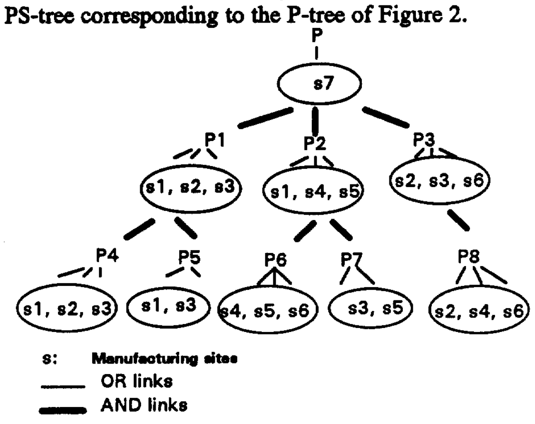

Figure 3: An example PS-tree corresponding to the P-tree of Figure 2.

A vertex $V(P_i, [])$ of a PS-tree is called a null-vertex if for the plan P_i there is no qualifying site available to perform P_i. In this sense, a null-vertex may be considered a dummy vertex to identify subplans that are not assigned to any site. The absence of a null-vertex in a PS-tree implies that all subplans of that tree are assigned to qualifying sites.

Solved-vertex

The concept of a solved-vertex is similar to that of [Nils71]. A vertex V(P,S) of a PS-tree is said to be solved if it and all its descendent vertices are non-null.

Solution-tree

A solution-tree for a plan P is represented by a subtree of the corresponding PS-tree, if that subtree has at least one non-null vertex for each subplan (including the root) of the PS-tree. Obviously, there may be several solution-trees for a plan P.

Communication cost

In a CAMPS environment the planning process requires communications between the LPSs. Since this could be a substantial portion of the CAMPS system, we model this cost separately. The transmission cost of sending a subplan P_i and the results of the related activities between two sites is denoted by $C_{P,Pi}$. $C_{P,Pi}$ is zero if P and P_i are executed at the same site.

Mis-match factor

The mis-match factor denoted by $R.S(P,S_i)$ is a measure of the quality and efficiency of site S_i in executing the assignment $V(P,S_i)$, as calculated by its parent site S. A high degree of mis-match between a plan or subplan's resource needs and the capabilities of S_i indicates increased manufacturing costs. Small (close to zero) values of $R.S(P,S_i)$ indicate high quality and efficiency, and high values indicate that site S_i is a poor performer of P. Initial values of $R.S(P,S_i)$ are provided by a local planning expert based on historical experience/data. The mis-match factor may be computed periodically with a human expert's input. (See Appendix A).

Subplan cost

Each V(P,S) has an associated implementation cost I_{PS}. This cost is also provided by a local planning expert. I_{PS} includes costs for implementation activities such as problem analysis, systems design, subplan decomposition, integration of subplans, execution of subplans, coordination between sites, and any rearrangements in production/assembly lines to execute a plan.

Solution tree cost

For a $V(P,S) \in \mathcal{V}$ we recursively define the cost h(V(P,S)) of a subtree of a solution tree rooted at V(P,S) as follows:

$h(V(P,S)) = I_{PS}$, if P is a leaf node of $\mathcal{P}$,

$$h(V(P,S)) = \Sigma_i \; [C_{P,Pi} + h(V(P_i,S_i)) + + R.S(P_i,S_i) + I_{PS}], \tag{1}$$

if V(P,S) is connected by some L_A to $V(P_i,S_i)$, $i = 1,2, \ldots, n$.

$$h(V(P,S)) = C_{P,Pi} + h(V(Pi,Sj)) + R.S(P_i,S_j) + I_{PS},$$

if V(P,S) is connected by some L_O to $V(P_i,Sj)$, $i = 1,2, \ldots ,n$, $j = 1,2, ., k_i$.

Heuristic function

We define a heuristic function $h^*(V(P,S))$ for vertex V(P,S) similar to that of the A^* algorithm [Hart68]. $h^*(V(P,S))$ is an estimate of the cost of the least cost solution tree rooted at vertex V(P,S). This heuristic function is used in the δ-algorithm in Section

3.2 as an evaluation function for vertex expansions, and in guiding the search for the optimal solution tree. $h^*(V(P,S))$ is calculated using a formula similar to (1) with $h(V(P,S))$ replaced by $h^*(V(P,S))$.

The inclusion of the mismatch-factor in the cost function can be justified by everyday examples, e.g., a company winning a bid. Assuming, that its estimated cost was far less than the real cost, (it won the bid): its mis-match factor increases. It has to pay the difference between the two values, or at least a portion of it. Hence its cost, calculated after the plan performance is higher and is reflected in the higher value of its mis-match factor. In the opposite case, when an estimate is close to the actual value, its mis-match factor decreases. If h^* is greater than h, the mis-match factor also increases, hence the cost is also increases, because the company takes the risk not to win the bid. It can be thought as a penalty for taking the risk of loosing the bid.

For a plan P, let $h^{*m}(V(P,S))$ be the minimum of all the estimates computed by all qualifying sites S_1, S_2,.., S_m, that is,
$h^{*m}(V(P,S)) = \min_i\{h^*(V(P,S_i))\}$, $i=1,2,..,m$.

δ-environment

For a plan P, F(P) is defined as:
$F(P) = \{S_i : |h^{*m}(V(P,S)) - h^*(V(P,S_i))| < \delta, \delta>0, i=1,2,..,m\}$. The sites in F are said to be in the δ-environment of the least-cost solution tree with root V(P,S). This is similar to the approach used in the A^*_ε algorithm [Pear84].

Plan set up time

A basic objective of CAMPS is to support dynamic plan modifications in a real time environment. This assumes a flexible manufacturing set up at the sites. However, each site may have differing set up times to execute a plan assignment. An estimate of the set up time of a site S for plan P is denoted by t(P,S), and is obtained from a local planning expert.

Successor operator

Let Γ be the successor operator that can be applied to a vertex V(P,S) to obtain all of the OR successors of that vertex, defined as follows: Multicast the description of V(P,S)'s subplans (i.e., P's subplans) to k sites that have the k least mis-match factors and are able to execute the subplans. These sites estimate one or more subplan costs, and those that can satisfy the requirements answer with the estimated cost of the subplans. Assign the responding sites to V(P,S)'s subplans obtaining the OR successor vertices of V(P,S) for each subplan.

Least-cost solution-tree

Using the cost function given above for solution-trees, it is desirable to obtain the solution-tree/s with the least cost. Denote the solution tree with the least cost by $V_{opt}(P,S)$. An algorithm to find a solution tree with the least cost or near to the least cost is presented in section 3.2.

3.1 Information components of a site

In order to support the planning methodology of this paper, the sites of the CAMPs network need four kinds of information processing modules. These are:
a) Knowledge base,
b) Planner module,
c) Communication and execution subsystems.

a) Knowledge base

The knowledge base at each site has information about: resources of sites, local plans (the plans within the site), remote plans (information about plans of other sites), site plans, partial global plans, constraints on forming vertices (such as $V(P_5, S_8)$ is disallowed), and planning rules of the planner module.

Local and global resources

Each LPS knowledge base has details regarding the resources available at the site, limitations of the resources, a directory of each plan/subplan of the site, and a table of the resources need to execute a plan or subplan. Each LPS has information about plans at other sites, the resources (subplans) available at the site, the limitations of the resources, and the resources available on a contingency basis.

Local plans

The concepts of local plans, node plans, and partial global plans used in this paper is borrowed from [Durf81]. Each site has detailed information about its plans and subplans. Such information may include data about subplan priorities, time and cost estimates of actions, and other such details. For example, a production site develops plans on detailed costs/capacity, production scheduling, inventory, assembly line balancing, critical operations etc., while the plans of an accounting site might include product pricing, cost allocations, capital budgeting, and cost control methods. Such information would include a description of the plan and its subplans, and the resources required to integrate it.

Remote plans

Sites have information on local plans of other remote sites and relationships between remote plans.

Site plans

A site plan is a higher level (less detailed) abstraction of a local plan, which may be transmitted to other sites interested in that information. The assumption is that execution of a subplan at a site requires detailed information contained in a local plan; whereas the details are not required at other sites that request information about the local plan. Each local plan has an associated site plan.

Partial Global Plan (PGP)

A PGP specifies site plans that match plans of other sites, concurrent and sequential activities, and goals of other sites working in related parts of a problem. In this approach, goals are not always handed down, but emerge from group communications. The sites constantly inform each other about their goals and refine their common goals by engaging in negotiation. Sites may have different assumptions, priorities, and evaluation criteria depending on local factors. Conflicts between sites need to be resolved by an overall system planning.

b) Planner module

The objective of the CAMPS planner module is to intelligently coordinate the plans of sites to formulate subplan-site assignments that constitute an optimal plan. The function of each LPS is to assign sites for all subplans considering the local goals, major activities, limitations on expected time and cost, expected result quality, and any local/internal priorities.

LPSs periodically transmit their plans to other sites that may be interested in cooperating with it on some plans. Such exchange may involve communicating the actual details of local plans and/or some higher level abstraction of the same. After each such exchange (or periodically) the planner evaluates the plans of other sites and builds PGPs which are also exchanged periodically with other LPSs.

For example, when a new product is to be introduced, an accounting, production, and marketing site from a PGP from their local plans based on factors such as expected price, cost scheduling, and current inventory of the new product. Initially each site may have a conflicting view about the new product. After identifying PGPs by matching local goals they can determine if their local goals/plans are part of a larger goal. The planner in each site recognizes the interleaving activities from the local plans, and reorders the site's activities to resolve conflicts and to reach agreement.

The planner also manages the search for knowledge and data not available locally. If site S does not have the resources to perform a subplan it consults its global directory to identify sites capable of implementing the subplan. It responds to messages originating from other sites, and updates the local knowledge base if new information is acquired. It compares the actual cost and the planned cost, and recomputes the mismatch factor values of other sites involved in cooperative planning and execution of activities. It integrates the results from other sites, formulates the final plan and transmits this to the execution subsystem. A heuristic algorithm, the δ-algorithm, used by the planner is presented in section 3.2.

c) Communication and execution subsystems

The communications model of CAMPS draws heavily on the concept of Partial Global Plans described in [Durf88]. The coordination structure of CAMPS is a pattern of decision making and communication among a set of sites that develop and execute plans in order to achieve goals. This is similar to that presented by Baligh [Bali86]. A site's communications subsystem is responsible for communicating about site plans. The sites communicate with each other to achieve a common goal. The role of the communication system is to ensure error free communication between the sites.

The execution subsystem has the necessary mapping information to map a solution plan into actual commands to control the machines on the manufacturing work floor.

3.2. A search algorithm to find a least cost solution-tree

The goal of our search algorithm is to find a solution tree with the least cost, that is, find the set of sites that can execute a plan with minimal cost, and they have the capability to adapt to the new production requirements in the shortest period of time. The δ-algorithm given below is an extension of Pearl's A^*_ε heuristic search procedure, and uses the method of vertex generation [Pear84].

Steps in the CAMPS algorithm

Assume that the initial plan P has been assigned to site S. We introduce two lists: L and L*. L is the list vertices generated by Γ and waiting for expansion; whenever Γ is applied to a vertex, the successors generated are put in L. L* is the list of vertices that were already expanded.

1. Put the vertex V(P,S) on the list L.

2. If L is empty or has only null-vertices, STOP and report 'No solution has been found'.

3. Remove a vertex V(Y,Z) from L and place it on L* such that site $Z \in F$, **and** Z sent the smallest time estimate t(Y,Z).

4. If V(Y,Z) is a terminal vertex, STOP with the solution tree constructed by designating the arcs from V(Y,Z) to V(P,S). Label the vertex V(Y,Z) as 'tentatively solved'.

   Otherwise apply the successor operator Γ for V(Y,Z) to generate the successors of V(Y,Z). Designate the arcs from every successor V(Y',Z') to V(Y,Z). For every successor V(Y',Z') of V(Y,Z):

   a. If V(Y',Z') is not already on L or L*, calculate h* (V(Y,Z)), and send it to the parent of V(Y,Z). Each parent does the same: it recalculates the cost estimate and sends it to its parent, etc., to the root V(P,S).

   b. If V(Y',Z') is already on L or L*, designate the arcs along the path where Z is still element of F, and has the smallest time estimate.

   c. If the arc of V(Y',Z') was re-designated and was on L*, then put it back on L.

5. Go to step 2.

After building a solution tree, the actual execution of the plans can start. After the execution of a plan P by a site S, the actual cost h(V(P,S)) of the solution tree/subtree can exactly be determined. The mis-match factor of S is reevaluated by S's parent in the PS-tree based on the difference between the estimated cost h^*(V(P,S)) and the actual cost h(V(P,S)) of the previous j plan execution as it will be described in appendix A.

Criteria to find a solution tree

Assume that the arc costs in the PS - tree are the communication costs, then the algorithm always finds a solution tree that has a cost in the δ environment of the least cost, if the following requirements are met:

a) $h^*(V(S,Q)) \leq h(V(S,Q))$ for all vertices in the PS - tree, and
b) all arc costs are greater than 0.

The proof is similar to that of A^*_ε in [Pearl, 1984].

The sites in F still should follow the same policy as in A^*, that is, they try to compute h^* as a close lower bound on h. Unlike h^*, t may overestimate the time required to adapt to the new production requirements.

3.3 Interesting features of the δ-algorithm

a) Our algorithm is different from the A* algorithm in that we use two heuristic functions, cost heuristics and set up time heuristics, to guide the search.

b) In the algorithm, the closer the lower bound on h that is used for h*, the more focused the search will be toward finding the estimated least cost solution tree, therefore the communications between sites are reduced.

c) The search algorithm selects that site in F (site in the δ-environment of the least cost) with the smallest set up time. The calculation of the mis-match factor ensures that the plans are assigned to those sites in F, for which past records indicate that on the average of the last j plans, their cost estimates have been very close to the actual cost. This implies that there is a high chance for the selected sites to be not only in the δ environment of the estimated least cost solution tree, but also in the δ environment of the actual least cost solution tree.

4. Conclusion and suggestions for future research

CAMPS is modeled as a set of loosely coupled, autonomous sites that communicate with each other to achieve a common goal. The goal is to assign plans/subplans to sites that are the most capable of executing the plan with the lowest cost and they have the capability to adapt to the production requirements in the shortest period of time. A search algorithm is presented for this purpose. This algorithm uses implementation costs, communication costs, plan-site mismatch factors, and site set up times, to obtain a solution. The practicality of our algorithm needs to be demonstrated using an appropriate simulation experiment involving manufacturing sites and plans. We are in the process of conducting such a study.

Appendix A

Calculation of the mis-match factor R and the parameter k

(Available from authors upon request).

References

Appl8 Applegate, L.M., B.R. Konsynski, and J.F. Nunamaker, "Model Management Systems: Design for Decision Support," Decision Support Systems, 2(1), 1986, pp. 81-91.

Bali86 Baligh, H.H., "Decision Rules and Transactions, Organizations and Markets," Management Science, 32, 1986, pp. 1480-1491.

Bond88 Bond, A.H., and Les Gasser, "Readings in Distributed Artificial Intelligence," Morgan Kaufmann Pub. Inc., 1988.

Bonc80 Bonczek, R., C. Holsapple, and A. Whinston, "The Evolving Roles of Models in Decision Support Systems," Decision Sciences, 11(3), 1980, pp. 337-356.

Conr86 Susan E. Conry, Robert A. Meyer, and Victor R. Lesser, "Multistage Negotiation in Distributed Planning," Technical Report 86-87, Department of computer and Information science, University of Massachusetts, Amherst, MA, December 1986.

Crof87 W. Bruce Croft and Lawrence S. Lefkowitz, "Knowledge-Based Support of Cooperative Activities," Proceedings of the 21st Annual Hawaii International Conference on System Sciences, vol. III, 1988, pp 312-318.

Date86 Date, C.J., An Introduction to Database Systems, Vol 1, Addison Wesley, Reading, Mass, 1986.

Druc88 Drucker, Peter, "The Coming of the New Organization", Harvard Business Review, Jan-Feb. 1988, pp. 45-53.

Durf88 Durfee, H. Edmund, Coordination of Distributed Problem Solvers, Kluwer Academic Publishers, 1988.

Eck89 Eck, R., M. Goul, A. Philippakis, and S. Richards, "Group Operating Systems for Decision Factories of the Future: An Extended Relational GDSS Architecture", Proceedings of the 22nd Annual Hawaii Conference on System Sciences, 1989, pp. 280-290.

Elam87 Elam, J.J., and B. Konsynski, "Using Artificial Intelligence Techniques to Enhance the Capabilities of Model Management Systems", Decision Sciences, 18(3), 1987, pp. 487-501.

Gass89 Gasser, L., and M. N. Huhns, (Editors), Distributed Artificial Intelligence, Morgan Kaufmann Publishers, 1989.

Geof87 Geoffrion, A.M., "An Introduction to Structured Modeling", Management Science, 33(5), 1987, pp. 547-588.

Goul87 Goul, M. and F. Tonge, "Project IPMA: Applying Decision System Design Principles to Building Expert-Based Systems", Decision Sciences, 1987, 18(3), pp. 448-467.

Gyir91 Tibor Gyires, "A Heuristic Algorithm for Distributed Control in Manufacturing

Systems," Applied Intelligence: The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem Solving Technologies, Vol. 1, No. 2, 1991, pp.145–155.

Hart68 Hart, P.E., Nilsson, N. J., and Raphael, B., A formal basis for the heuristic determination of minimum cost paths, IEEE Trans. Systems Science and Cybernetics, 4(2), pp.100-107, 1068.

Hube87 Huber, G.P. and Daft, R.l., "The Information Environment of Organizations", in F.M. Jablin, L.L. Putnam, K.H. Roberts and L.W. Porter, Handbook of Organizational Communication: An Interdisciplinary Perspective, Sage Publications, Newbury Park, pp. 130-164, 1987.

Kers86 Kerschberg, L., (Editor), Expert Database Systems, Proceedings of The First International Workshop, Benjamin Cummings Publishing Co., 1986.

Malo86 Malone, T.W., Modeling Coordination in Organizations and Markets, Management Science, 33(10), pp.1317-1332, 1987.

Morg87 Leora Morgnstern, "Knowledge Preconditions for Actions and Plans" Proceedings of the 1987 International Joint Conference on Artificial Intelligence, pp. 867-874, 1987.

Nils71 Nilsson, N.J., Problem-Solving Methods in Artificial Intelligence, McGrawHill Book company, 1971.

Pan91 Pan, J. Y. C. and Tenenbaum J, M., An Intelligent Agent Framework for Enterprise Integration, IEEE Transactions on Systems, Man, and Cybernetics, VOL. 21, NO. 6, November/December, pp. 1391-1408, 1991.

Pear84 Pearl, J., Heuristics, Addison-Wesley Publishing Company, pp. 63-64, 1984.

Ras89 Zbigniew W. Ras, (Ed.) "Methodologies for Intelligent Systems, 4," Proceedings of the Fourth International Symposium, ISMIS, North Holland, 1989.

Ras90 Zbigniew W. Ras, Maria Zemakova, and Mary L. Emrich, (Eds.) "Methodologies for Intelligent Systems, 5," Proceedings of the Fifth International Symposium, ISMIS, North Holland, 1990.

Ras91 Z. W. Ras, and M. Zemakova, (Eds.),"Methodologies for Intelligent Systems," Proceedings of the 6th International Symposium, ISMIS, Springer-Verlag, Berlin Heidelberg, 1991.

Ston87 Stonebraker, M., J. Anton, and E. Hanson, "Extending a Database System with Procedures", ACM Transactions on Database Systems, 1987, 12(3), 350–367.

Turb86 Turban, E., and P.R. Watkins, "Integrating Expert Systems and Decision Support Systems", MIS Quarterly, June 1986, 121–136.

Turb90 Turban, E., Decision Support and Expert Systems: Management Support Systems, 2nd Edition, Macmillan Publishing Co., 1990.

SMC92 IEEE Transactions on Systems, Man, and Cybernetics, VOL. 21, NO. 6, November/December

Wied86 Wiederhold, G., R.L. Blum, and M. Walker, "An Integration of Knowledge and Data Representation", On Knowledge Base Management Systems, M.L. Brodie and J. Mylopoulos (Eds.), Springer–Verlag, New York, 1986, 431–449.