OVERVIEW OF CORTES: A CONSTRAINT BASED APPROACH TO PRODUCTION PLANNING, SCHEDULING AND CONTROL

Mark S. Fox and Katia P. Sycara

Center for Integrated Manufacturing Decision Systems
The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

ABSTRACT

We present an overview of CORTES, an integrated framework for production planning, scheduling and control (PSC). CORTES's approach to PSC problems departs from others in its underlying assumptions: *Generality Assumption*: There exists a single approach that can optimize decision making across a wide variety of PSC problems. *Flexibility Assumption*: The same approach can be used for both planning, predictive scheduling and reactive control. *Uncertainty Assumption*: In order to provide the appropriate level of precision in PSC, reasoning about uncertainty must be an integral part of the PSC approach. *Scale Assumption*: Large PSC problems, that contain thousands of activities, resources and constraints, must be solved in a qualitatively different manner than small PSC problems. CORTES uses Constrained Heuristic Search to make PSC decisions. In this paper, we describe CORTES, its architecture, problem solving method, and functions including modeling, planning, scheduling, distributed scheduling, dispatching, and uncertainty management.

1.0 INTRODUCTION

This paper provides an overview of CORTES, a distributed system for production planning, scheduling and control (PSC). The CORTES system is composed of an integrated set of modules distributed across many workstations and connected by a communication network. The overall architecture is shown in Figure 1-1.

CORTES represents a departure from previous approaches to solving the PSC problem in the hypotheses it explores:

1. There exists a single approach that can optimize decision making across a wide variety of PSC problems. Previously, PSC approaches were tailored to the particular production environment, with the "common wisdom" being that there does not exist a single approach, short of enumeration, that applies to all PSC problems. We believe that there does exist a single approach that may be generally applied to PSC problems, that also provides very good results and is computationally efficient.

2. The same approach can be used for both planning, predictive scheduling and reactive control. Traditionally, planning, scheduling and control approaches have tended to be separate and unrelated in approach. For example, MRP tends to be used for planning, scheduling approaches can range from dispatch approaches to knowledge based scheduling, and control tends to be ignored.

3. In order to provide the appropriate level of precision in PSC, reasoning about uncertainty must be an integral part of the PSC approach. Production environments contain a plethora of stochastic events that increase the uncertainty with which a schedule may be executed. For example, unexpected events such as personnel not showing up for work, machine failures, failure of resources, etc, may quickly invalidate a schedule. Consequently, PSC approachs must take a pro-active approach in mitigating the

[1]This research has been supported, in part, by the Defense Advance Research Projects Agency under contract #F30602-88-C-0001, and in part by grants from McDonnell Aircraft Company and Digital Equipment Corporation.

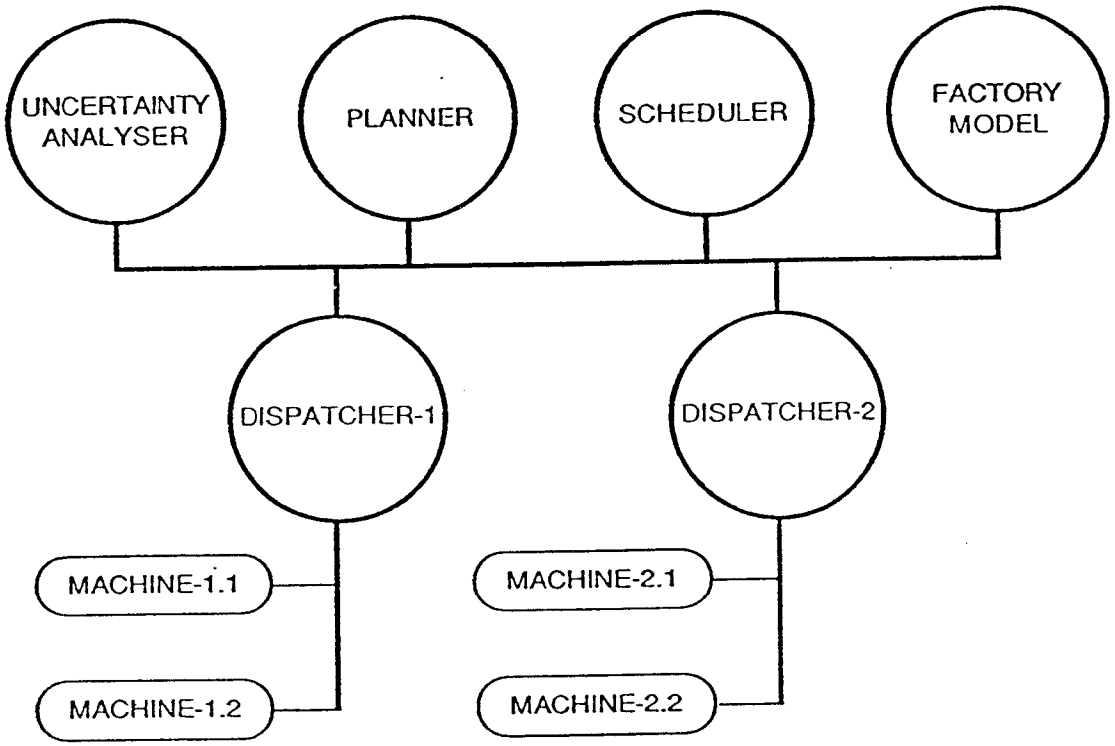

Figure 1-1: The CORTES Architecture

effects of uncertainty.

4. Large PSC problems, that contain thousands of activities, resources and constraints, must be solved in a qualitatively different manner than small PSC problems. The point is that in large PSCs, the aggregate behavior of the system be optimized, as opposed to any individual entity or job. Optimizing each decision is computationally expensive. Instead, many decisions must be made at the aggregate level using statistical summaries of underlying requirements.

CORTES is evolutionary in its approach in that it can be viewed as a continuation of the line of constraint directed scheduling systems developed at Carnegie Mellon University [8, 24, 7]. It departs from the approach of these previous systems in its use of Constrained Heuristic Search (CHS) as its underlying problem solving paradigm [9].

In the reminder of the paper, we first review the Constrained Heuristic Search (CHS) problem solving paradigm. We then describe the functionality of each of the modules in figure 1-1 and how this is provided using CHS.

2.0 CONSTRAINED HEURISTIC SEARCH

Constrained Heuristic Search (CHS)[2] is the problem solving method that lies at the core of CORTES. It is our belief that CHS is powerful enough that it can be applied across a variety of PSC problems, and that it can be used for planning, scheduling and control.

CHS views problem solving as a constraint optimization activity. CHS combines the process of constraint satisfaction (CSP) [16] with heuristic search (HS). CHS retains heuristic search's synthetic capabilities and extends

[2]This section is a composed of excerpts from [9].

it by adding the structural characteristics of constraint satisfaction techniques. In particular, our model adds to the definition of a problem space [18], composed of states, operators and an evaluation function, by refining a state to include:

1. **Problem Topology:** Provides a structural characterization of a problem.
2. **Problem Textures:** Provide measures of a problem topology that allows search to be focused in a way that reduces backtracking.
3. **Problem Objective:** Defines an objective function for rating alternative solutions that satisfy a goal description.

This model allows us to (1) view problem solving as constraint optimization, thus taking advantage of these techniques, (2), incorporate the synthetic capabilities of heuristic search, thus allowing the dynamic modification of the constraint model, and (3) extend constraint satisfaction to the larger class of optimization problems. In the following, problem topology and textures are defined.

2.1 Problem Topology

We define problem topology as a graph G, composed of vertices V and edges E:

$V = N \cup C \cup S$

where

N is a set of variables $\{n_1, n_2, ..., n_m\}$
C is a set of constraints $\{c_1, c_2, ..., c_n\}$
S is a set of satisfiability specifications $\{s_1, s_2, ..., s_o\}$

Each variable in N may be a vector of variables whose domains may be finite/infinite and continuous/discrete. Constraints are n-ary predicates over variables vertices. A satisfiability specification vertex groups constraints into sets of type AND, OR, or XOR. An XOR satisfaction set denotes that only one constraint in the set must be satisfied. Edges link constraint vertices to variable vertices, and satisfiability specifications to constraints.

We distinguish between two types of problem topologies:

Definition 1: A *completely structured problem* is one in which all non-redundant vertices and edges are known a priori.

This is true of all CSP formulations.

Definition 2: A *partially structured problem* is one in which not all non-redundant vertices and edges are known prior to problem solving.

This definition tends to be true of problems in which synthesis is performed resulting in new variables and constraints (e.g. the generation of new subgoals during the planning process).

Operators in CHS have many roles: refining the problem by adding new variable and constraint vertices, reducing the number of solutions by reducing the domains of variables (e.g., assigning a value to a variable vertex), or reformulating the problem by relaxing constraints or omitting constraints and/or variables.

2.2 Problem Textures

Focus of attention in search is concerned with the ability of the search algorithm to opportunistically decide where the next decision is to be made [3]. In CHS, for search to be well focused, that is to decide where in the problem topology an operator is to be applied, there must be features of the topology that differentiate one subgraph from another, and these features must be related to the goals of the problem. We have identified and are experimenting with seven such features that we call problem *textures* [20]. Below we define these textures for CHSs where all solutions are equally preferred, i.e., the Problem Objective rates all solutions to the constraints equally acceptable.

(Variable) Value Goodness: the probability that the assignment of that value to the variable leads to an overall

solution to the CHS (i.e. to a fully consistent set of assignments). This texture is related to the value ordering heuristics [12] which look for the least constraining values. Value ordering heuristics are meant to reduce the chance of backtracking. In the case of discrete variables, the goodness of a value is the ratio of complete assignments that are solutions to the CHS and have that value for the variable over the total number of possible assignments.

Constraint Tightness: Constraint tightness refers to the contention between one constraint or a subset of constraints with all the other problem constraints. Consider a CHS A and a subset C' of constraints in A. Let B be the CHS obtained by omitting C's constraints in A. The constraint tightness induced by C' on A is defined as the probability that a solution to B is not a solution to A. In the case of discrete variables, this is the ratio of solutions to B that are not solutions to A over the total number of solutions to B.

Variable Tightness with respect to a set of constraints: Again consider a CHS A, a subset C' of constraints, and the CHS B obtained by omitting C' in A. A variable V's tightness with respect to the set of constraints C' is defined as the probability that the value of V in a solution to B does not violate C'. In the case of discrete variables, this is simply the ratio of solutions to B in which V's value violates C' (i.e. at least one of the constraints in C') over the total number of solutions to B.

Constraint Reliance: This measures the the importance of satisfying a particular constraint. Consider a constraint c_i. We defined CHS B as being CHS A - $\{c_i\}$. Given that constraints can be disjunctively defined, the reliance of CHS A on a constraint c_i is the probability that a solution to CHS B is not a solution to A. In the case of discrete variables, constraint reliance is defined as the ratio of the number of solutions to CHS B that are not a solution to CHS A to the number of solutions to CHS B. The larger the value, the greater the reliance the problem has on satisfying the particular constraint.

Variable Tightness: Consider a variable v in a CHS A. Let C' be the set of constraints involving v and B be the CHS obtained by omitting C' in A. v's tightness with respect to C' is simply called v's tightness. Hence the tightness of a variable is the probability that an assignment consistent with all the problem constraints that do not involve that variable does not result in a solution. Alternatively one can define *variable looseness* as the probability that an assignment that has been checked for consistency with all the problem constraints, except those involving that variable, results in a fully consistent assignment. Notice that if one uses a variable instantiation order where v is the last variable, v's tightness is the *backtracking probability*. Variable looseness/tightness can be identified with variable ordering heuristics [12, 11] which instantiate variables in order of decreasing tightness.

These textures generalize the notion of constraint satisfiability or looseness defined by [17] and apply to both CHSs (and CSPs) with discrete and continuous variables. Notice that, unless one knows all the CHS's solutions, the textures that we have just defined have to be approximated. Textures may sometime be evaluated analytically [20]. A brute force method to evaluate any texture measure consists in the use of Monte Carlo techniques. Such techniques may however be very costly. In general, for a given CHS, some textures are easier to approximate than others, and some are also more useful than others. Usually the texture measures that contain the most information are also the ones that are the most difficult to evaluate. Hence there is a tradeoff. Each domain may have its own approximation for a texture measure.

We have extended these textures to take into account the Problem Objective where the objective is expressed as a sum of functions of one variable, using Bayesian probabilities to approximate the likelihood that a variable results in an optimal value [20].

2.3 CHS Problem Solver

The CHS model of problem solving is a combination of constraint satisfaction and heuristic search. Search is performed in the problem space where each state contains a problem topology. The problem solving model we propose contains the following elements:

- An initial state is defined composed of a problem topology, i.e., the PSC activity, time and capacity constraint graph,
- Constraint propagation is performed within the state,

- Texture measures and the problem objective are evaluated for the state's topology,
- Operators are matched against the state's topology, and
- A variable node/operator pair is selected and the operator is applied, i.e., a resource or start time is assigned to an activity.

The application of an operator results in either adding structure to the topology, further restricting the domain of a variable, or reformulating the problem (e.g., relaxation).

It is our belief, which is supported by experimentation, that this approach is powerful enough to solve a variety of PSC problems. Domains in which it has been applied include, job shop scheduling [21], cell scheduling and transportation planning [23]. Secondly, the opportunism inherent in the approach, allows the approach to be applied to both predictive planning and scheduling, and reactive control.

3.0 FACTORY MODEL

Tha factory floor module represents the shop floor environment. The module includes a manager agent that communicates with the other agent modules in the CORTES system in order to (a) supply needed information to the other module agents, and (b) update the factory model when appropriate information changes as a result of processing by the other agents, or as a result of unexpected events on the floor.

CORTES is being tested on a variety of PSC problems taken from the Operations Research and Artificial Intelligence literature. In addition, it is being tested on the CARMEMCO test data. The Carnegie Mellon Engineering and Manufacturing Company (CARMEMCO) is an imaginary company whose purpose is to provide a testbed for research into engineering and manufacturing decision systems at CIMDS[3]. CARMEMCO grew out of a need to provide an integrated knowledge base that could serve the research needs of our projects in design for manufacturability, facility layout, production planning, production scheduling, project management, and intelligent system interfaces. A long range goal for CARMEMCO is to bootstrap the implementation of an on-campus factory that will serve as a development- and test-bed for research projects in robotics and automation of manufacturing.

The criteria used for selecting the products CARMEMCO would produce were:

- The domain should provide a testbed for primarily mechanical design, with the opportunity for electrical and electronic design.
- The domain should involve description and manipulation of 3D objects that have interesting but not too detailed design features.
- Part components should be both fabricated and assembled so that planning and scheduling research could explore both.
- Components should be made out of a variety of materials. Some components should have material options.
- The domain should require a wide variety of resources and processes that provide complex challenges for process planning, facility layout, and scheduling systems.
- Components could actually be fabricated at CMU or purchased externally.
- Students and faculty at CMU should provide a market for the product.

Desk lamps were selected with these criteria in mind. Lamp components fit the design criterion quite well, as many are relatively simple, but all have at least a few interesting and unique features. For example, some arm components are straightforward hollow cylinders, while some base and head components are irregular polygons in 3D. With respect to the materials criterion, lamp components can be metal, hard plastic, soft plastic, wire, and foam. Some components can actually be either metal or plastic. With respect to the material and process variety criteria, lamp manufacturing requires purchasing, fabrication, assembly, subcontract fabrication, subcontract assembly, non-destructive testing, packing, and distribution, as well as front end marketing and sales operations. The resources for these processes are large in number and type, as well as diverse in their operational and maintenance needs. Parts

[3]Center for Integrated Manufacturing Decision Systems.

can be produced either in batches or on an individual basis.

CORTES and CARMEMCO use the same representation. The main conceptual primitives in the CORTES representation are *activities*, *resources*, *production units*, *states*, and *constraints* [6]. These primitives provide an extensible framework that can be used to represent the relevant aspects of manufacturing environments. In addition, these primitives are represented at various levels of conceptual abstraction depending on the granularity of knowledge. For example, an *operation* is a specialization of an activity. An important component of the representational framework is the *relations* that connect the primitives and their instantiations. The main types of relations are temporal and causal. The concepts presented in this section are represented as *schemata* [14, 22]. A schema encapsulates information and has an identifying name. A schema has a set of slots that describe attributes of the schema. *Meta* information can be attached to a slot, a slot value or schema. Meta information is information *about* the slot, slot value, or schema and is independent of the meaning represented by these entities.

3.1 Activities

Activities are the subject of scheduling decisions. The specification of activities includes the temporal and causal relations that connect them as well as their organization as *aggregate activities* into larger constructs. An activity is *elaborated* into an aggregate activity (an activity network) whose activities are *part-of* the aggregate activity. For example, a milling-operation has an elaboration *milling-operation-network*, which in turn has two activities, *milling-setup* and *milling-run*. The primitive relation *part-of* connects the activity network to its component activities. Thus, the *elaboration-of* relation separates an activity from its detailed description and facilitates the representation of multiple elaborations of the same activity at different levels of abstraction.

In the manufacturing environment, activities are typically of two types: (a) operations associated with the manufacturing processes to produce a part/product, and (b) supporting activities, such as maintenance and repair. Temporal and causal relations organize operations into production plans that indicate which operations need to succeed each other, which ones can be executed in parallel and which resources each operation needs for its execution.

Another way to aggregate operations is in terms of an *order* which indicates how many parts need to be produced for the order's fulfillment. An operation is the basic unit of action in a scheduling environment. It defines a transformation of the world from one state to another so that at the end a part (an order) is produced.

The primitive relation *has-subactivity* denotes the set of activities that comprise the prototype aggregate activity. The primitive relation *subactivity-of* relates a subactivity back to the aggregate activity to which it belongs. These relations are used to describe process abstractions.

In order to be performed, activities require one or more resources. For example, in order to perform a *cutting operation*, the required resources are a *machine* on whose machine head a *cutting tool* can be affixed, the *part* on which the cutting operation is to be performed, a *fixture* that immobilizes the part on the machine's bed, and an *operator* to operate the machine and to load and unload the part and fixture. The resource requirements of activities are expressed by the primitive relations *required-resource*, associated with an activity, and *resource-for* associated with a resource. In our model, it is assumed that all resources required by an activity, are required for the whole duration of the activity. An additional assumption that we make is that 100% of each resource is required for the performance of the associated activity. This is a simplifying assumption that might not be true in actual manufacturing environments. For example, depending on the duration of a particular cutting operation, an operator might be able to operate (e.g., load another part, fixture it and start an operation) another machine during the cutting operation.

3.2 States

A state is the collection of conditions under which an activity can be performed, or the collection of new conditions produced by the activity. An activity is connected to its precondition and postcondition states by *causal relations*. The primitive pair of inverse relations *enables* and *enabled-by* specify the connection between an activity and its enabling state. In a similar fashion, the pair of inverse primitive relations, *cause* and *caused-by* defines the

connection between an activity and its resulting set of conditions.

The abstraction of state information is performed using the same operators (conjunction and disjunction) as for activity information, resulting in *aggregate states* that are related to their component states via the *has-sub-state* relation. In turn, the component states are related back to the aggregate state via the *sub-state-of* relation. An aggregate state that is a disjunct is true if any of its sub-states is true. An aggregate state that is a conjunct should have all its sub-states true in order to be true. States, as well as activities, persist for particular time intervals. The relation *has-time-interval* is associated with each state and activity in a scheduling system.

3.3 Resources

In this section, we present the hierarchical representation of resources to enable reasoning at different levels of precision in allocating a resource. Viewing resources at various aggregate levels impacts the level of granularity of capacity definitions. In general, the capacity of a resource is the number of items that the resource can process simultaneously. The term "item" is a general term and can refer to different units on which the resource is operating. For example, a machine operator has capacity 1 and (usually) operates one machine; a milling machine can be configured to cut 3 identical parts at the same time (capacity 3).

We differentiate resources into groupings of various types. The two basic groupings that we distinguish are *stationary* and *mobile* resources. Stationary resources are the ones that have a fixed location. In a manufacturing organization, examples of stationary resources are machines and workstations. Examples of mobile resources are human operators and fixtures that can be loaded on different machines. The *current location* attribute holds location information for movable resources. In some situations, a stationary resource can have moving parts. For example, a milling machine can have a moving head that can take more than one position. Hence, in the representation of such a resource, the *current position* of the machine head has to be noted. Associated with fixtures are *load* and *unload* operations that require the fixture and the position on which it is to be loaded as resources. The *status* of a resource indicates whether it is available or not.

3.4 Production Units

Production units are the entities which are transformed by operations during the manufacturing process. In this work, we are interested in modeling production units from the standpoint of scheduling. The central concept in modeling production units is the *part*. The manufacture of a part enters the manufacturing system through a work-order that specifies the QUANTITY of the part ordered, the DUE-DATE of the order, the SERIAL-NUMBER and the order's PRIORITY.

One central characteristic of a part is the production process through which it is manufactured. An abstracted representation of the process is the *process-plan*. Production plans are usually prototypical in that at the time of their construction they have no information (neither can they anticipate) factory floor configurations and scheduling constraints. The scheduler instantiates the prototypical process plan for filling a work order to reflect the realities of the factory floor at scheduling time. Another characteristic of a part is the required material that is used to produce the part. This is related to inventory concerns of availability of material. Currently, we assume that the required material is present when needed for the production of the part.

3.5 Constraints

In general, there are five types of constraints that a scheduler should take into consideration.

- Physical constraints. Physical constraints include, number of machines, fixtures, setup and run times for each operation.
- Organizational constraints. Examples of organizational constraints include meeting due dates, reducing Work in Process, increase machine utilization, and enhance throughput.
- Preferential constraints. Examples of preferential constraints include preference for using a particular

machine for an operation (perhaps because of its speed or accuracy), or using a particular human operator (perhaps because of his skill).

- Enablement constraints. These refer to constraints, the fulfillment of which creates a state that enables the execution of an activity. For example, a process plan embodies enablement constraints.
- Availability constraints. These constraints refer to the availability of particular resources at scheduling time. For example, a machine may become unavailable because of breakdown, the assignment of a third shift makes extra resources available for scheduling.

In the model, we treat explicitly two types of constraints, *required constraints* and *preferential constraints* [4]. The degree of satisfaction of a preferential constraint is expressed by a *utility function* ranging between 0 and 1. A value of 0 utility is non-admissible; a value of 1 is optimal. Variables can be constrained by more than one constraint. The utility value associated with a variable is calculated by taking the weighted sum (with constraint importance as the weight) of the utilities of all the constraints that affect the variable.

Constraints differ in *importance*. A particular constraint could have different importance depending on the context in which it is applied. The importance of a constraint is specified by a value between 0 and 1. An importance of 0 implies that the constraint should not be considered, and 1 signifies maximum importance. The actual level of importance is relative to the importance of the other constraints under consideration. The measure of importance of a constraint may be viewed as a weight that can be combined with a constraint's utility value to form a weighted combination of utilities. Constraints also differ in *relevance*. Depending on the context, a constraint may be more relevant than others.

3.6 Representing temporal relations

In our model, the relations among variables that we consider are primarily temporal relations. In particular, Allen's 13 temporal relations defined over time intervals [1] are represented. The temporal relations express the precedence relations among manufacturing activities in the process plan.

Whereas Allen's temporal relations can be used to express the occurrence of events that are *relative* to one another in a temporal sense, in scheduling there also arises the need to represent the occurence of events in an *absolute* temporal sense. For example, the fact that operation1 has to precede operation2 can be expressed by "operation1 before operation2". On the other hand, the reservation of resource1 (required by operation1) for the time interval [t1, t2] is the expression of en event in an absolute temporal sense. Since, in scheduling, one is interested in the persistence of facts over time, we have chosen the *time interval* as the basic time primitive object.

4.0 SCHEDULER

The detailed scheduler is an activity-based scheduler [21], where the activities are the operations that must be scheduled according to a process plan that specifies a partial ordering among these operations. Each operation requires one or several resources for each of which there may be one or several alternatives. Scheduling is viewed as a constrained heuristic search problem whose solution is a schedule that satisfies the many technological, temporal, organizational, and preference constraints that are imposed both by the characteristics of the job shop itself and the environment.

The scheduler models a problem as a constraint graph, where there are two types of nodes: activities and resources. An activity is an 4-tuple defining its start time, duration, and resources it is to use. With each activity, we associate utility functions that map each possible start time and each possible resource alternatives onto a utility value (i.e. preference). These utilities [5, 20] arise from global organizational goals such as reducing order tardiness (i.e. meeting due dates), reducing order earliness (i.e. finished good inventory), reducing order flowtime (i.e. in-process inventory), using accurate machines, performing some activities during some shifts rather than others, etc. A resource is a 3-tuple defining its total capacity, available capacity over time, and the activities that are scheduled to use it.

We distinguish between two types of constraints: activity temporal constraints and capacity constraints. The activity temporal constraints together with the order release dates and latest acceptable completion dates restrict the set of acceptable start times of each activity. The capacity constraints restrict the number of activities that a resource can be allocated to at any moment in time to the capacity of that resource. For the sake of simplicity, we only consider resources with unary capacity in this paper. Typically the limited capacity of the resources induces interactions between orders competing for the possession of the same resource at the same time.

The schedule is built incrementally by iteratively selecting an activity and assigning a start time and resource(s) to it, propagating temporal and capacity constraints and checking for constraint violations. If constraint violations are detected the system backtracks. Search is focused via a set of *variable* and *value* ordering heuristics so as to minimize backtracking and optimize schedule quality.

The variable ordering heuristic assigns a *criticality measure* to each unscheduled activity; *the activity with the highest criticality is scheduled first.* The value ordering heuristic attempts to leave enough options open to the activities that have not yet been scheduled in order to reduce the chances of backtracking. This is done by assigning a *goodness* measure to each possible reservation of the activity to be scheduled. Both activity criticality and value goodness are composed of *texture measures.* The next two paragraphs briefly describe both of these measures[4].

A critical activity is one whose resource requirements are likely to conflict with the resource requirements of other activities. [20, 21] describes a technique to identify such activities. The technique starts by building for each unscheduled activity and for each appropriate time interval a probabilistic *activity demand* that denotes the probability that the activity will require a resource at that time interval. Clearly activities with many possible start times and resource reservations tend to have smaller demands at any moment in time, while activities with fewer possible reservations tend to have higher ones. In a second step, the activity demands for each resource are aggregated over time to form a demand profile for a resource. The demand profile expresses likely contention for the resource over time. The percentage contribution of an activity's demand to the aggregate demand for a resource over a highly contended-for time interval is the *activity reliance.*

To choose the next activity to schedule, the scheduler focuses on the resource/time interval with the highest aggregate demand. The activity with the the highest reliance on the resource is picked to be scheduled next, since it is the activity that is most likely to be involved in contention for the resource.

The particular start time assigned to the chosen activity is picked using either of two strategies:

1. **A Least Constraining Value Ordering Strategy (LCV):** This heuristic attempts to select the reservation that is the least likely to prevent other activities to be scheduled.
2. **A "Greedy" Value Ordering Strategy (GV):** At the other extreme, a reservation can be chosen that maximizes the preference of the activity for the resource/time interval.

Experimental results have demonstrated the effectiveness of this approach for problems where resource contention is an issue [21].

5.0 DISTRIBUTED SCHEDULING

As part of the CORTES project, we are investigating how to manage scheduling when distributed across multiple schedulers [26]. In particular, we are investigating how schedulers, which possess their own resources, coordinate their decisions when they require resources possessed by others. The distributed CORTES system consists of several schedulers, each of which is given a set of orders to schedule. The orders may require resources that are also needed by other schedulers. Each resource in the system is monitored by a scheduler, and conversely, each scheduler is a monitor for some set of resources. The concept of monitoring schedulers is introduced in order to facilitate detection of capacity conflicts. Each resource monitor keeps track of the reservations that have been made for the resource it monitors. Capacity constraints are detected by the monitor when a scheduler requests a reservation for the monitored resource and for a time interval that has already been reserved. In addition, monitoring

[4]For a more complete description of these measures, the reader is referred to [20, 21].

schedulers achieve more efficient inter-scheduler communication to inform schedulers of new reservations and changes in expected resource needs. They may be viewed as communication hubs for collecting and dispersing information.

Due to the size of the scheduling problem, we distinguish between coordination at the strategic level versus the tactical level. Our approach assumes that each scheduler develops schedules using Constrained Heuristic Search. At the strategic level, the coordination of large numbers of activities requiring resources outside of a particular scheduler is performed by communicating statistical summaries of aggregate demand textures. These demands are used to bias a scheduler's reservations so that it does not require another's scheduler's resources during a period of high demand. At the tactical level, particular scheduling decisions are made and, if needed, negotiation takes place.

Demand profiles are aggregated periodically to compute textures that allow schedulers to form expectations about the resource demands of other schedulers. Because of communication overhead, the demand profile information is restricted. Subsets of the schedulers communicate only demand profiles for the resources that they share, although reservations on the non-shared resources may impact scheduling decisions on the shared ones. Since several schedulers are scheduling asynchronously, and the communicated demand profiles are only those of the subset of shared resources, there is higher uncertainty in the system. This uncertainty also varies in an inversely proportional manner with the frequency at which the demand profiles are communicated. Moreover, the cost of backtracking is greater, since if a scheduler backtracks, the change in scheduling reservations may ripple through to the other schedulers and cause them to change their reservations.

We have implemented a distributed scheduling testbed and are currently experimenting with distributed scheduling protocols both at the strategic and tactical levels.

6.0 PLANNING

We are currently investigating the integration of planning with scheduling[5]. In previous planners, planning has been an end unto itself. Any feasible plan is considered a success, with only very inflexible criteria for plan quality, such as minimizing the total number of actions. In the context of the CORTES project, the planner will be producing process plans to be used by the scheduler. The quality of these plans is defined by the quality of the schedules that the scheduler can produce using them. Thus, there is a strong need for a constraint language to use in communicating with the scheduler to determine what sorts of plans would be good.

Current state-of-the-art planners are constraint-directed, domain-independent, hierarchical, nonlinear, and support replanning [27]. We intend to include these capabilities, and extend them where appropriate.

Planners already exist that use constraints on planning variables to increase the power of their representation and to reduce arbitrary decisions that can lead to unnecessary backtracking. In addition to making wider use of constraints, we will make this planner be truly constraint-*directed* by developing measures of criticality for goal ordering and operator selection. This will provide a domain-independent representation for the domain-dependent heuristics that focus attention in the search for a plan.

The planner will always support planning at different levels of abstraction, and the re-use of plans in support of reactive planning.

7.0 UNCERTAINTY ANALYZER

Uncertainty is a fact of life in most job shop scheduling environments. Sources of uncertainty include: Demand change (seasonal, forecast error, cancel orders, expedition), Inventory Policy (raw material arrival pattern, safety stock policy) Machine failure, Change of time duration (transit, set-up, processing), Yield, and Quality (Tool

[5]See [10] for more details.

and shop loads. The general result is that type-2 bounds give sufficient protection against uncertainty in processing time with less investment in the planned cost (planned cost= planned tardiness+planned work-in-process+ planned lateness)[7]. For a more detailed description of the experiments, see [2].

8.0 INTELLIGENT NETWORKING

Achieving manufacturing efficiency requires that the many groups that comprise a manufacturing enterprise, such as design, planning, production, distribution, field service, accounting, sales and marketing, cooperate in order to achieve their common goal. Cooperation can take many forms:

- Communication of information relevant to one or more groups' tasks. For example, sales informing marketing of customer requirements, or production informing the controller of production performances.
- Feedback on the performance of a group's task. For example, field service informing design and manufacturing of the operating performance of a new product.
- Monitoring and controlling activities. For example, controlling the execution of operations on the factory floor.
- Assignment of new tasks. For example, a new product manager signing up production facilities to produce a new product.

An intelligent network is viewed as the "nervous system" of the enterprise, enabling the functions described above. It is more than a network protocol (e.g., MAP) in that it operates and participates at the application level. The following describes the capabilities provided by the Intelligent Network:

- **Information routing:** given a representation for information to be placed on the network and a representation of the goals and information needs of groups on the network, the information routing capability is able to provide the following routing capabilities:
  - Static routing: transferring information to groups where the sender and the receivers are pre-defined.
  - Dynamic routing: transfering information to groups which appear to be interested in the information. This is accomplished by matching a group's goals and information needs to the information packet.
  - Retrospective routing: reviewing old information packets to see if they match new goals and information requirements specified by a group.
- **Closed loop system:** Often, the communication of information results in some activity, which the initiator of the communication may be interested in. The INP will support the providing of feedback in two modes:
  - Pre-define feedback: operationalizes pre-defined information flows between groups in the organization. For example, production providing feedback to sales on the receipt of orders.
  - Novel feedback: Providing feedback for new and novel messages.
- **Command and control:** Given a model of the firm which includes personnel, departments, resources, goals, constraints, authority and responsibility relations, the INP will support these lines of authority and responsibility in the assignment, execution and monitoring of goals and activities. In particular, it will manage the distribution of information and the performance of tasks.
- **Dynamic task distribution:** Supporting the creation of new organizational groups and decomposition, assignment and integration of new goals and tasks, contracting and negotiation are examples of techniques to be supported.

[7]To protect against uncertainty the planned operation duration is longer, more planned work-in-process exists and orders are planned to arrive late. Hence, the more protection we design into the bounds, the higher the planned cost.

wear, precision). Uncertainty increases as the planning horizon is extended, and its the amount and sources of uncertainty change over time.

The presence of uncertainty means that it is very unlikely that a detailed predictive schedule that assigns precise start and finishing times on resources for activities is going to be adhered to. This characteristic imposes two requirements on schedulers: (a) A scheduler should be able to represent and reason about degrees of uncertainty, and (b) a scheduler should be able to react to unexpected events on the factory floor. The inability of a scheduler to reason about uncertainty almost always results in a schedule being invalid at the time it is released to the production floor.

CORTES manages uncertainty in three stages. In the first stage, the Uncertainty Analysis module monitors and records the stochastic events. It develops over time a model of the sources and characteristics of uncertainty. Once a valid model is constructed, the Uncertainty Analysis module passes the information to the Scheduler. In the second stage, the scheduler uses the uncertainty models to reduce the precision of its schedules. Precision can be reduced by increasing the durations of activities, overlapping activity temporal intervals, or assigning activities to resource aggregates rather than to specific resources. In the third stage, the Dispatcher control module, is able to react more flexibly to stochastic events by taking advantage of the imprecision inserted in the schedule by the scheduler; it can start an activity earlier or later or assign an activity to another resource in an aggregate (i.e., work center). The Dispatcher's task is to dispatch jobs to machines and monitor machine and job execution status. The Dispatcher notes deviations from the schedule and resource unavailability and communicates this information to the Scheduler, Uncertainty Analyzer and factory floor.

Two approaches have typically been utilized to address the problem of temporal uncertainty. One approach is based on the idea of dividing the time horizon into time zones using progressively coarser time units to describe events in the future. For example, a time unit of one hour may be used to project a schedule over a one week horizon; a time unit of a day mey be used to project a schedule from a one-week to a one-month horizon and so on. Although this approach recognizes the fact that events that are further int the future are less accurately predictable, it has been criticized [15] as suffering from the presence of discontinuous boundaries between time zones and the difficulty of handling orders whose processing crosses a zone boundary. A second approach to handling uncertainty is the use of probability distributions to describe schedule parameters. This approach has the disadvantage [15] that probability is concerned with the combination and manipulation of independent random variables whereas many of the probabilistically described scheduling parameters are not independent (e.g., processing times of different jobs on a particular machine could depend on some characteristic of the machine)[6].

The CORTES uncertainty analyzer represents uncertainty in terms of fuzzy logic [28, 13, 19]. The present version [2] focuses on uncertainty concerning machine failures. The mean time between failure and mean duration of the failure are assumed known. It is also assumed that once a machine is fixed after a failure, processing resumes at the point of interruption with no rework necessary. In other words, machine failure causes a variation in processing time only and not in scheduling order. The time between machine failures and the failure duration are used to express uncertainty in processing time. Instead of being random variables of known distribution, the duration of failure and time between failures may be only approximately known. This approximate information on the procesing time bounds is expressed in terms of fuzzy numbers of *Type-1*, where a real number that is approximately known is expressed as a confidence interval of upper and lower bounds. Fuzzy bound values may be ther result of subjectively known processing characteristics described by a shop operator, or of known distributions described by shop statistics. An extension of type-1 fuzzy representation of uncertainty in operation duration is *Type-2* representation where the lower and upper bounds of a confidence interval, instead of being ordinary numbers are fuzzy numbers that themselves have intervals of confidence.

Uncertainty bounds work in a similar manner as earliest start time/latest start time and earliest finish/latest finish time. The bounds can be viewed as slack to protect against uncertainty. The mean processing time is reserved for the operation and the slack time is reserved for protection against uncertainty. Once an operation is ready for processing, a dispatcher should follow the schedule within the prescribed bounds. We ran experiments [2] to compare various cost measures, such as tardiness, work-in-process, and delayed orders, under various processing duration representation schemes (type-2 fuzzy representation, fixed processing time given in the process plan, mean processing time considering machine failure duration and time between failures) and under different cost structures

[6]Handling variable dependence through the use of conditional or joint probability distributions poses severe estimation problems.

9.0 CONCLUSION

In this paper, we have given an overview of the CORTES integrated framework for production planning, scheduling and control (PSC) system. CORTES's approach to PSC problems departs from others in its underlying assumptions:

1. Generality Assumption: There exists a single approach that can optimize decision making across a wide variety of PSC problems.
2. Flexibility Assumption: The same approach can be used for both planning, predictive scheduling and reactive control.
3. Uncertainty Assumption: In order to provide the appropriate level of precision in PSC, reasoning about uncertainty must be an integral part of the PSC approach.
4. Scale Assumption: Large PSC problems, that contain thousands of activities, resources and constraints, must be solved in a qualitatively different manner than small PSC problems.

The CORTES projects is investigating all four assumptions in parallel. We have experimental data across a variety of PSC problems that supports the generality assumption. The flexibility assumption is currently being tested by our integration of PSC functions. The uncertainty assumption is supported by the ease with which we have adapted CHS to account for uncertainty. The Scale assmumption is still been tested.

REFERENCES

1. Allen, J.F. "Towards a General Theory of Action and Time". *Artificial Intelligence 23*, 2 (1988), 123-154.

2. Chiang, W-Y., and Fox, M.S. Protection Against Uncertainty In a Deterministic Schedule. Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management, May, 1990. Submitted for publication.

3. Erman, L.D., Hayes-Roth, F., Lesser, V.R., and Reddy, D.R. "The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty". *ACM Computing Surveys 12*, 2 (1980), 213-253.

4. M. Fox. *Constraint-Directed Search: A Case Study of Job-Shop Scheduling*. Ph.D. Th., Department of Computer Science, Carnegie-Mellon University, Pittsburgh (Pennsylvania, U.S.A.), 1983.

5. Fox, M.S. Constraint-Directed Search: A Case Study of Job-Shop Scheduling. Carnegie-Mellon University, 1983. CMU-RI-TR-85-7, Intelligent Systems Laboratory, The Robotics Institute, Pittsburgh,PA.

6. Fox, M., and Sycara. K. Knowledge-Based Logistics Planning and its Application in Manufacturing and Strategic Planning. Tech. Rept. First Interim Report, submitted to RADC, CMU Robotics Institute, December, 1988.

7. Fox, M.S. "Constraint Guided Scheduling: A Short History of Scheduling Research at CMU". *Computers and Industry* (1990). To Appear.

8. Fox, M.S., and Smith, S. "ISIS: A Knoweldge-Based System for Factory Scheduling". *International Journal of Expert Systems 1*, 1 (1984), 25-49.

9. Fox, M.S., Sadeh, N., and Baycan, C. Constrained Heuristic Search. Proceedings of the International Joint Conference on Artificial Intelligence, 1989, pp. 309-316.

10. Frederking, R.E., and Chase, L.L. Planning in a CIM Environment: Research Towards a Constraint-Directed Planner. Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management, May, 1990. Submitted for publication.

11. Freuder, E.C. "A Sufficient Condition for Backtrack-free Search". *Journal of the ACM 29*, 1 (1982), 24-32.

12. Haralick, R.M., and Elliott, G.L. "Increasing Tree Search Efficiency for Constraint Satisfaction Problems". *Artificial Intelligence 14*, 3 (1980), 263-313.

The design of the Intelligent Network is divided into five layers [25]. The *Network Layer* provides for the definition of the network architecture. At this level the nodes are named and declared to be part of the network. Message sending between nodes is supported along with synchronization primitives.

The *Data Layer* provides for queries and responses to occur between nodes in a formal query language patterned after SQL.

The *Information Layer* provides "invisible" access to information spread throughout the Intelligent Network. The goal is to make information located anywhere in the Network locally accessible without having the programs executed locally know where in the network the information is located nor explicitly request its retrieval.

The *Organization Layer* provides automatic communication of information based upon the roles a node plays in the organization.

The *Market Layer* supports the distribution of tasks and the negotiation of change.

Events and information among the various CORTES modules is routed by the intelligent network. Translation tables located in each module translate between different representations. Modules communicate asynchronously through timestamped messages at the data layer. If information generated by one module is relevant to the problem solving of another, the information is automatically routed to the appropriate module at the information layer. If a module needs information that resides in another module's knowledge base, the information layer initiates search for the information by sending an appropriate request message to the module that possesses the information. Inter-module communication is shown in Figure 8-1.

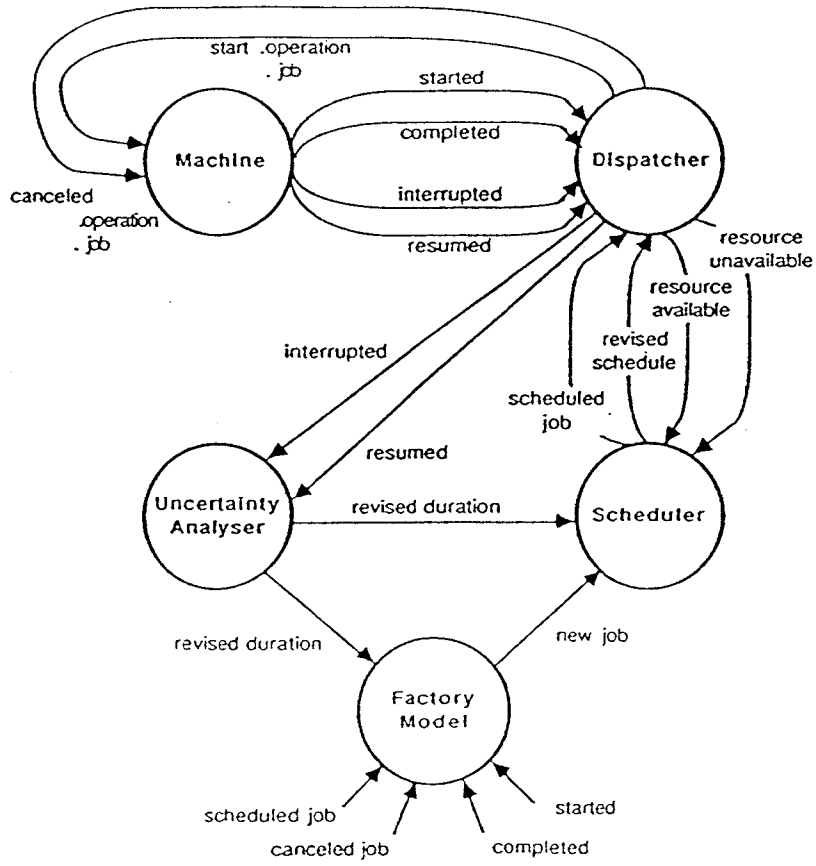

Figure 8-1: Inter-module communication in CORTES

13. Kaufmann, A. and Gupta, M.. *Introduction to Fuzzy Arithmetic: Theory and Applications.* Van Nostrand Reinhold, New York, N.Y., 1985.

14. *KnowledgeCraft Reference Manual.* Carnegie Group Inc., Pittsburgh, PA., 1986.

15. Kerr, R.M., and Walker, R.N. A Job Shop Scheduling System Based on Fuzzy Arithmetic. Proceedings of the 2nd Interational Conference on Expert systems and Leading Edge in Production and Operations Management, Hilton Head Island, S.C., May, 1989, pp. 433-450.

16. . "Consistency in Networks of Relations". *Artificial Intelligence 8*, 1 (1977), 99-118.

17. Nadel, B.A. The General Consistent Labeling (or Constraint Satisfaction) Problem. Tech. Rept. DCS-TR-170, Department of Computer Science, Laboratory for Computer Research, Rutgers University, New Brunswick, NJ 08903, 1986.

18. Newell, A., and Simon, H.A. "Computer Sciences as Empirical Inquiry: Symbols and Search". *Communications of the ACM 19*, 3 (1976), 113-126.

19. Prade, H. "Using fuzzy set theory in a scheduling problem". *Fuzzy Sets and Systems* 2, 2 (1979), 153-165.

20. Sadeh, N., and Fox, M.S. Preference Propagation in Temporal Constraints Graphs. Intelligent Systems Laboratory ,The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, 1988. CMU-RI-TR-89-2.

21. Sadeh, N., and Fox, M.S. Focusing Attention in an Activity-based Job-Shop Scheduler. Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management, May, 1990. Submitted for publication.

22. Sathi, A., Fox, M.S., and Greenberg, M. " Representation of Activity Knowledge for Project Management". *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-7*, 5 (September 1985), 531-552.

23. Sathi, N., Fox, M.S., Goyal, R., and Kott, A. CORAL: An Order Configuration and Resource Allocation System. Carnegie Group Inc., Five PPG Place, Pittsburgh PA 15219, 1990. In preparation.

24. Smith, S., Fox, M.S., and Ow, P.S. "Constructing and Maintaining Detailed Production Plans: Investigations into the Development of Knowledge-Based Factory Scheduling Systems". *AI Magazine 7*, 4 (Fall 1986), 45-61.

25. Sycara, K., and Marshall C. Towards an Architecture to Support Integration of Decision-Making in Manufacturing. Proceedings of the IJCAI-89 Workshop on Integrated Architectures for Manufacturing, Detroit, MI., 1989.

26. Sycara, K., Roth, S., Sadeh, N., and Fox, M.S. Distributing Production Control. Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management, May, 1990. Submitted for publication.

27. Wilkins, D.E.. *Practical Planning.* Morgan Kaufmann Publishers Inc., 1988.

28. Zadeh, L. "Fuzzy Sets". *Information and Control 8*, 338 (1985).