Review of State of the Art in Planning and Scheduling as Related to COORDINATORS Project: Part I

Adele Howe

December 22, 2005

1 Summary of Program Concept

The COORDINATORS program emphasizes distributed intelligent, cooperative problem solving to a degree not previously demanded of the research community. Automated agents must recognize important change, assess its ramifications on its local view of activity, request or propagate information to other agents as necessary to reduce uncertainty and accommodate change, and re-schedule activity or select contingencies to stave off plan failure. Additionally, the agents must manage their own limited resources (computational and informational), make decisions within the allowed rule bounds and learn models of appropriate interaction with their human users. All of these tasks must be accomplished within an intrinsically distributed environment with interaction and knowledge circumscribed by formal organization such as chain of command and need-to-know.

The BAA set out five primary technical areas: distributed activity coordination, context-dependent coordination autonomy, machine learning, organizational reasoning and meta-cognition. The BAA also identified four key hard research problems: distributed coordination over large dynamic structures, coordination of multiple role units, learning appropriate decision making autonomy with sparse data, adapting activity in real time in response to change and incorporating military decision policies in coordinated decision making.

In this document, we will review research in planning and scheduling that bears directly on the hard research problems in COORDINATORS. The focus will be on identifying what most taxes the current state of the art, what has been done previously to address those issues and how the contractors' research fits in the field as a whole.

1.1 Issues that Tax the State of the Art

The COORDINATORS vision goes well beyond the reach of current state of the art. Most of the core capabilities required (e.g., reasoning about temporal changes in tasks, identifying alternative resources, representing distributed task interdependencies, etc.) have been previously developed to a limited extent. This is reflected in the proposals from the contractors, who are working from an existing method base and integrating components and extending capabilities. However, the capabilities have been studied either in a centralized framework, independent of other capabilities, or in a small scale application with simplified characteristics (e.g., limited or

no dynamics, certain and complete information, etc.) Lack of centralized representation and embedding the capabilities are not simple addenda.

Although the BAA does not use the term, a key capability is scheduling: allocation of resources to tasks during specific time frames. Change Evaluation requires identifying how change affects the ongoing schedule. Task Analysis requires determining whether task timing can be adjusted without violating schedule constraints (task dependencies), resources can be substituted or alternative contingencies selected for tasks. Meta-cognition performs scheduling of the agent's computational resources.

State of the art scheduling approaches can handle large numbers (1000s) of tasks with well defined inter-dependencies (as in task A must proceed task B or a resource cannot be used for more than eight hours continuously). Multi-capacity resources are manageable, but the capacities are assumed to be fixed. Indirect interaction through resource contention or cascaded contention effects are commonly handled, but otherwise it is assumed that all other interactions that must be maintained are known. Task durations and time windows are assumed to be known and may be incorporated in evaluations of the proposed schedule. A variety of schedule evaluation metrics have been explored; techniques tend to be accommodating of changing priorities or metrics, again as long as they are explicit and measurable. Representation and display technologies have also been formulated to allow humans access to and some control over schedules. Typically, schedule generation and execution are completely de-coupled. Incremental scheduling techniques handle unexpected events so as to minimize perturbations and computation.

However, many issues intrinsic to COORDINATORS have received little attention in the literature and so pose significant challenges to the contractors, in particular:

- no centralized detailed schedule,
- extremely dynamic operating environment,
- accounting for and responding to a wide range of execution failures,
- incomplete information due to uncertain state of unfolding schedule and institutional restrictions on available information,
- near real time response, and
- consideration of human factors (appropriate autonomy, complex objectives and information overload on the human).

The most challenging issue is probably the distributed nature, which requires actively seeking information and confirming proposed changes with other agents.

Task analysis considers contingency selection as an alternative to modifying time slots or resource allocation as in pure scheduling. Action selection can either be viewed as a limited horizon form of planning or as a control problem (as in the reinforcement learning approaches). Each view incurs different assumptions and limitations; neither tends to work well on large state and/or action spaces. Both require considerable knowledge either explicit about the domain (for planning) or many examples (for reinforcement learning). The challenges posed by

COORDINATORS to the state of the art in action selection include those listed for scheduling plus:

- incompletely specified effects of actions and
- much larger scale (number of actions and state space).

More has been done on distributed planning/action selection within the agents community (than with scheduling), probably because of the common encoding of causal structures within plans that assist in determining what can be changed. However, the techniques for building and managing the plan structures mean that they tend to be more vulnerable to exponential scale-up of computational costs with increasing problem size.

2 State of the Art Approaches to Scheduling

Most research examines traditional manufacturing/commercial scheduling applications, e.g., job shop, vehicle routing and personnel scheduling. Solution approaches divide into ones that guarantee optimality and those that don't[1]. Guarantees of optimality typically are provided by methods developed by the Operations Research community, such as branch and bound and constraint propagation. Satisficing methods (those that produce solutions that are not necessarily optimal) are heuristic and search based.

Optimality comes at the cost of high computation and practical limits on problem size and/or complexity, as well as a stiff required level of accuracy and precision in the information available about the schedule. However, the distributed nature of the scheduling in COORDINATORS means that individual agent's schedules may be small enough to be solved by optimal methods (see section 4 for such an example).

By their nature, heuristic schedulers tend to be more forgiving of inaccurate data and more accommodating of dynamism. For example, the planners/schedulers developed at NASA (i.e., ASPEN [13] and CASPER [12]) consider execution as well as off-line scheduling and have been effectively deployed on missions to deep space and Mars, clearly uncertain environments.

This section reviews a range of approaches (especially somewhat less well known ones) that may be applicable to the issues in the COORDINATORS problems. Where possible, the strengths and weaknesses of the methods are assessed.

2.1 Scheduling Under Uncertainty

Scheduling under uncertainty is intrinsically related to dynamic scheduling. Uncertainty arises either because everything is not known in advance (and so may prove problematic later) or because what is known may change. Depending on the domain, the changes may be delays on activities, variations on activity durations or resource breakdowns as in a workflow model [11]. More generally, they may be addition or deletion of constraints, addition or deletion of tasks, the restriction or relaxation of time windows or the addition, deletion or reduction of resources. Presumably, any of the events may occur within the context of COORDINATORS.

[1] See `http://scom.hud.ac.uk/planet/repository/schedulers.html` for a listing of many research schedulers and their approaches.

Uncertainty means that the schedule must be monitored during execution. Lack of progress on tasks may result in an unacceptable decrease in surety (probability that a task will finish during its required window). In constraint programming contexts, downstream effects are immediately detected through propagation when constraints are updated. In grid computing, surety is monitored by requesting progress updates on each task in a job, as well as the critical path through the schedule; should surety drop or the critical path composition change, the scheduler can update time/cost estimates on different resources and then use CPM and PERT to re-compute the downstream allocation [49]. Such methods are feasible because the number of tasks are relatively small in this application; the primary overhead is in soliciting information (bids) from the resources for each task in a job when change of resource is required. In JSP where the activity durations are thought to be the primary source of uncertainty (machine breakdowns are the second source), one can use a simulation to play out effects possible during execution and then set a threshold for forcing re-scheduling based on violations of actual activity durations against anticipated distributions [8].

Uncertainty may also lead to *opportunities* for schedule improvement. As with detrimental change, opportunities (e.g., taking less than anticipated time to complete, resources that become available) may also trigger changes to on-going schedules. Opportunities can be anticipated and kept in a "opportunity library" with the plan; the library can be searched during plan monitoring to identify plan modifications [29]. The success of such an approach depends on the degree to which opportunities can be anticipated and the speed at which the new plan fragments can be spliced in.

In addition to standard metrics of quality of the schedule, schedules in uncertain/dynamic environments must also be stable and robust. Stability (also called "continuity" [11]) measures the difference between the original schedule and a schedule that is adjusted to accommodate changes; stable schedules minimize the difference [39] and are often desirable when schedules involve people [11]. As Davenport and Beck [14] observe, "A schedule therefore is not simply an internal recipe for a set of activities but also a basis for communication and coordination with external entities. The external dependencies make the management of uncertainty even more critical as unexpected events that are not reacted to and contained may have an impact that far out-weighs their original importance."

Robustness trades-off quality and vulnerability to change. Robust schedules tend to be less vulnerable to change but possibly at the expense of *predicted* quality measures [2].

The predictive or proactive approach to handling dynamism is to build a robust schedule originally [14]. The reactive approach is to repair a schedule with violated constraints. Because COORDINATORS assumes the original schedule is obtained elsewhere, this document will focus on reactive approaches.

Robustness and stability can be incorporated at schedule generation and repair time by operationalizing the definition as an evaluation measure [11]. At repair time, stability can be varied by imposing hard constraints on existing commitments to prevent them from being modified during schedule repair [11].

For dynamic JSP, Wu et al. [55] developed an approach that identifies critical decisions

[2]Interestingly, many researchers conjecture that robust schedules are lower quality, but I could find no study that showed the relationship between predicted and actual quality in dynamic domains.

(those that dictate global performance) and uses graph theory to decompose the global problem into ordered sequences of these critical decisions (solving the ordered assignment problem). The disjunctive graph formed by the task constraints becomes a series of subgraphs. Each subgraph is solved with a branch and bound algorithm. Decisions not fixed in the partial orders are made on-line in response to actual conditions. They tested their method in Monte Carlo simulations using dynamic dispatching rules to resolve the open decisions and found their approach to be quite effective on 5x10 JSPs from the standard benchmark.

Jensen [28] exploited co-evolution to generate flexible schedules which can cope with machine breakdowns in JSP. One population is used to construct the schedule with the best worst case performance. A second population constructs the worst breakdown scenarios to be used in assessing schedule performance. The best schedules then are the ones that best address the possible breakdowns. A hillclimber is used to reschedule given the robust schedule found by the GA. The approach was tried on 10 problems from the OR library varying in size and found to be quite effective, although potentially expensive to produce the schedules (taking minutes to produce schedules as opposed to seconds for the state of the art for the static version of these problems).

A common approach to schedule repair is domain tailored heuristics; such approaches produce solutions quickly. The heuristics make local repairs to the schedule to promote stability; several heuristic modifications may be proposed with the repair selected based on a trade-off of quality and stability [39]. Heuristics used in iterative repair include max-flexibility [32], conflict directed activity changes [45] and min-conflicts [37]. The heuristics are fast to compute and have been quite effective in a variety of domains, e.g., manufacturing scheduling, space applications and logistics. They have been less successful in applications with many way interaction effects as the heuristics tend to be myopic.

If one has access to historical information about likely perturbations, then approaches can learn or anticipate repairs in advance. An artificial immune system evolves a set of partial schedules off-line to respond to likely events that have occurred at run-time [22]. Preliminary results of this approach on machine scheduling problems have been promising, making for quick response to the most likely events. While not guaranteeing optimality or complete coverage of events, such an approach may also support a distributed environment if these contingencies could be calculated by and distributed to the impacted agents either at the same time as the original schedule or as an update during slow periods. Although not explored in the papers on the topic, the immune system approach could be interfaced to a simulator (as in [8]) rather than to historical data to produce the partial schedules that accommodate contingencies.

The literature is silent on uncertainty due to inaccuracies, mistakes or missing items in the data. Yet, it is likely to be an issue in this application. For example, during a recent visit to Air Mobility Command, Air Force personnel who handle logistics planning/scheduling stated that their data about what-is-where are often missing or is incorrect. The schedulers remarked that a pallet may suddenly show up thousands of miles away from where it had last been recorded without having been shown as being on a plane's manifest. It seems likely that such an issue will have to be dealt with should COORDINATORS technology be deployed. Current heuristic repairs are unlikely to have been designed for such situations, and solutions produced by optimization methods are likely to be extremely brittle.

2.2 Other Search Based Approaches to Scheduling

A variety of evolutionary and local search algorithms have been developed for scheduling applications. In general, the results have been, at best, mixed (which makes them more or less on par with any other solution). Other than those mentioned in earlier sections, some notable algorithms are Local Search (including Tabu Search) and Genetic Algorithms.

Local Search (LS) refers to a large family of gradient descent algorithms that use local information to make iterative repairs to complete solutions. Tabu Search (TS) adds limited memory to LS to avoid cycling and to force search into potentially more productive regions; the state of the art for JSP, based on performance on established benchmarks, is a carefully tuned TS algorithm that uses a neighborhood operator restricted to a subset of feasible solutions, two forms of memory and clever initialization [38].

Another favorable attribute of LS is that it can be easily adapted to incremental repair. Because it works with complete solutions and can accept tailored neighborhood operators, it can initialize with the current solution and fix the neighborhood to not change tasks in progress. The iterative nature of the search favors stable solutions.

A type of LS (called Attenuated Leap Local Search), Squeaky Wheel Optimization (SWO) and the Genitor GA have done well on the Air Force Satellite Control Network (AFSCN) communication problem [3]. This application includes a discrete optimization function and plenty of bottlenecks that cause cascade effects; the combination produces many large plateaus in the search space. The most successful algorithms are those that make multiple simultaneous changes with the number of changes reducing as the best solution is approached. SWO uses a greedy strategy to initialize the solution, then iteratively improves it by identifying trouble makers in the solution and increasing their priority, which has the effect of moving all of them simultaneously to better positions in the schedule potentially bumping other tasks [30].

GAs[3] have done well on a variety of scheduling applications, e.g., warehouse order management [47], Multiple Resource Constrained Project Scheduling problem [46], nurse scheduling [1] and the previously mentioned AFSCN. Generally, the algorithms have done well when the representation and operators were suited to fast modifications and evaluation of solutions. They require relatively little knowledge of what constitutes good solutions to be used.

The advantages of these knowledge poor search strategies is that they are easy to implement in the base case, it is easy to change goal criteria (unlike heuristics in which the goals may be implicitly coded), and they may be adapted to re-scheduling because all of them perform better when their initial solutions are close to the optimum. They can, however, take a long time to find the optimum or may miss it altogether and they do not take advantage of domain knowledge, generally.

2.3 Optimal Scheduling and CSP

Commercial scheduling systems use constraint programming as one of their core techniques. For example, ILOG produces arguably the best known commercial scheduling system. Its system description lists four categories of algorithms in its scheduler: edge finding for resource

[3]Rumor has it that GAs have been incorporated into several well known commercial schedulers including ILOG's.

constraints, timetabling for capacity constraints, sequencing and special purpose algorithms for applications such as personnel scheduling [27].

Edge finding and timetabling define constraint equations that reduce the values of bounding variables for tasks (such as resource capacity usage and end time and duration upper bounds) [33]. These equations are used to propagate restrictions on variable values for activities across the n activities and r resources. Each of these methods impose restrictions on the kinds of applications that can be handled (e.g., unary and discrete resources, cannot be oversubscribed, must reason about absolute points in time) and may not model the kinds of constraints in a given application.

A few constraint propagation algorithms have recently been proposed to deal with more complex resources, dynamism and relative time reasoning. The Energy Precedence algorithm allows new activities and resource constraints to be added and time windows kept loose by constructing a new time propagation constraint rule [33]. The Balance algorithm works on reservoir resources (ones that can increase and decrease capacity over time) without incurring exponential increases in computation as do some previous approaches [33]. Both the energy precedence and balance algorithm propagate constraints over precedence graphs to support reasoning about relative times. These algorithms have been demonstrated on some scheduling problems and proposed to be used in HTN or POP planners. Focacci et al. (2000) define new constraints (alternative resource and path optimization) to the propagation algorithm on precedence graphs to handle problems with setup times and alternative resources; the enhanced algorithm has found good solutions to JSP and open shop problems with up to 300 activities (computation times could be up to 240 seconds).

In dynamic scheduling, *unimodular probing* is a strategy for minimizing schedule perturbation (sum of absolute value of differences in start times) overall [48]. Unimodular probing interleaves phases of linear and constraint programming to identify minimal modifications to a schedule to accommodate activity, resource and temporal changes on-line. In the first phase (resource feasibility), linear programming orders temporal variables to resolve resource contention. In the second phase (temporal optimization), constraint programming finds optimal values for the temporal variables, given the orderings found in the first phase. The authors experiment with different constraint programming techniques including Constraint Backtracking, Probe Backtracking and a variant on Probe Backtracking that uses LP to prune states due to violation of bounds and to guide value selection. The technique has been applied to a large commercial aircraft scheduling problem.

3 State of the Art Approaches to Action Selection

The traditional delineation of planning and scheduling is that planners decide what to do and schedulers decide when/where to do it. Action selection is essentially the first step from scheduling towards planning – given a set of options for a given time slot (usually impending), pick an action to execute. This problem is embedded in on-line plan execution and usually requires a plan with explicit or implicit contingencies.

This section reviews some of the more promising techniques for solving the execution time action selection problem as in COORDINATORS. Although many researchers approach the

problem in terms of building plans that are robust to incompleteness of information (i.e., "conformant planning" [51]), we are ignoring that because of assuming the plan generation is not under our control.

3.1 Decisions with Imperfect Knowledge

The research in planning under uncertainty falls mostly into two categories: POMDPs and contingent planning. Partially Observable Markov Decision Processes (POMDPs) exploit a Markov process model, but extend it to handle domains in which the exact state of the world may be unknown (or even unknowable). POMDPs were first developed in the control theory community in the early 1960's as a means of controlling environments in which the true state of a stochastic, dynamic system may be unknowable. Within AI, they have been used mostly for robust robotic navigation (going back to [10, 43]), although the reinforcement learning community has been exploring the connections to Q-learning for somewhat longer. POMDPs define a policy (contingency plan) that maps the belief state (usually a vector of probability distributions over the possible states of the world) to values for possible actions.

POMDPs have the advantage that they smoothly encompass a variety of sources of uncertainty (e.g., effects of actions, noise in sensors, poor mapping from sensor data to state, missing information, etc.) and that the resulting courses of action are provably optimal, given the information that is available.

Unfortunately, the primary drawback for POMDPs has been keeping the computation manageable. Computation is an issue in both the off-line search (constructing the POMDP and so the optimal strategy) as well as the on-line search (collecting and mapping the sensor information to monitor the belief state) [42]. The computation (both time and space) can increase exponentially with the size of the state space, which is why relatively small grid worlds were fairly popular. Typically, responses to this problem have traded off accuracy against computation tractability via sampling, projection and special data structures like density trees [42].

Recent research has investigated techniques for increasing the state space that can be accommodated. For example, point-based algorithms approximate the value iteration, while bounding the error; recent enhancements can solve problems with 10^5 states to very close to optimal within 10^4 seconds [52].

An alternative to approximation is compression. Value-Directed compression [41] retains the same structure, which allows it to be solved similarly, but has scaled up to 33 million states on a synthetic network management problem (three actions are possible to maintain a network of up to 25 computers). Another approach exploits Exponential family Principal Components Analysis (PCA) to produce even more dramatic compressions (solving a real robotic navigation application – traversing an entire retirement facility); the difference with other compression work is that it requires different algorithms to execute.

Another approach to limiting the search space for POMDPs, but achieving on-line action selection, is to restrict the space to only branch points [6]. Benazera proposes that at execution time plans can be abstracted into models where states are the choice points for contingencies, intermediate actions/states are not incorporated. Because computing all possible contingencies is intractable, most plans include only a few, if any contingencies, that represent the most likely events; consequently, this strategy reduces the POMDP to a manageable size and then supports

robust probabilistic reasoning about outcomes. He also uses Monte Carlo techniques with a simulator to approximate the probability and utility values needed by the POMDP to produce anytime computation of the models.

Contingent planners adopt a variety of approaches. Binary Decision Diagrams (BDDs) coupled with symbolic model checking is a promising approach to handling sensing and partial observability within a somewhat traditional planning framework [7], but results for contingent planning so far have been shown on fairly small navigation problems. Similarly, knowledge based search approaches have significantly extended the capabilities of traditional planners, but are solving the entire planning problem (i.e., generating the plan from scratch) and are solving relatively small problems (e.g., [40]).

NASA's planner/schedulers typically operate in environments that are poorly anticipated by the designers. NASA researchers typically deviate from approaches advocated in the literature; in a recent analysis, they reviewed approaches from literature on planning with uncertainty and summarized why they don't work for NASA problems in [9]. In short, most techniques assume discrete action sequences and fairly small search spaces. One of the most successful of NASA's systems is CASPER (Continuous Activity Scheduling Planning Execution and Replanning) [44, 12]. CASPER manages a rolling plan that is kept flexible as long as possible with details filled out as needed. Plans are modified to improve quality or to repair problems in an iterative optimization framework where "experts" get to propose changes, which are ranked based on expected cost and benefit. Five classes of experts are supported to make different modifications: local activity variable, activity/goal counts, resource/state variables, resource/state change counts and state durations. The experts identify possible changes and gauge their cost/benefits.

3.2 Resource Bounded, Time/Quality Trade-Offs

Reasoning about resources and action selection quality is an area that has until recently received relatively little attention. Although scheduling considers alternative measures of schedule quality, planning is difficult enough that researchers have by and large ignored application specific quality measures concentrating simply on minimizing the number of actions.

Zilberstein has cast the problem in terms of anytime algorithms [56] and contract algorithms [57]. Such algorithms balance computation time with result quality when temporal deadlines are restrictive and make informed estimates of resource allocation. Anytime algorithms require decision making that can always return some solution; contract algorithms require a good model of how much time is required to return some solution and how additional time influences quality.

Schwarzfischer [50] identifies quality as a local property and schedule deadlines/utility as a contextual property. He also relates deadlines to anytime problems in real-time scheduling. In OR, similar work has been done on the time/cost trade-off problem in which durations can be shortened at the expense of activity cost [31]. Some work has been done attempting to combine the time/cost trade-off problem with resource constraints [53, 26], and Fulkerson [17] shows that such a problem can be solved in polynomial time when there is only a single resource. Wang [54] presents a proof of NP-completeness for the case of multiple resources.

Portfolios provide a more general framework. The basic idea is that a suite of algorithms can be combined so as to provide performance superior to any individual one. A meta-reasoner

selects which search algorithms to apply to attempt to solve a problem and then allocates computational resources (almost always time, but in the heterogeneous grid literature, specific platforms) to the selected algorithms. In its original formulation [25], the portfolio allocated the fraction of total CPU time to n different Las Vegas algorithms (these algorithms could be distinctly different or could be multiple runs of the same stochastic algorithm); the portfolio terminated as soon as one of the algorithms solved the problem instance. Huberman et al. [25] used an economics framework and models of run time distributions of algorithms on problem instances to compute the best portfolio, given a trade-off between expected cost and risk (standard deviation of cost).

Gomes and Selman [19, 18] advanced a similar idea with a focus on the role of restarts (starting the same stochastic algorithm repeatedly either in sequence or in parallel). They explicitly compute the value of multiple versions (restarts) of the same algorithm as part of the portfolio and show that in some cases multiple copies of the same algorithm may be preferable to a mix of algorithms.

These approaches assume the run-time distributions for algorithms on problem instances have been collected off-line. Alternatively, the model construction can happen on-line through a probing phase [5]. During probing, all algorithms are run for the same amount of time (a small fraction of the available CPU time for solving the problem); the quality of solutions returned directs the selection, using simple rules, of which algorithm to continue to run for the time remaining. They found they could achieve performance comparable to (i.e., not statistically different than) the best pure technique for 100 JSP problems, which include randomly generated as well as both job and machine correlated problem instances.

The Bus meta-planner [24] used a round robin scheme to allocate time to different planning systems based on a trade-off between a model of expected success and expected cost; for each planning system, its model was a simple linear regression of easily extractable problem features based on considerable off-line experience in planning. Bus outperformed any single pure system on a set of benchmark planning problems.

Each of these approaches requires only shallow knowledge of the algorithms and problems. However, they ignore the considerable knowledge that is being compiled about algorithm performance and encapsulated in algorithm and application models.

A variety of approaches have been applied to learning models of performance to inform problem solver design and algorithm selection (e.g., [20, 36, 15, 2, 35, 21, 34]). These approaches differ in the machine learning technique used (e.g., reinforcement learning, Bayesian networks, decision trees, statistical regression) and on the algorithms and applications modeled (e.g., satellite communication scheduling, combinatorial auctions, sorting, SAT), but all use relatively easily extracted features of the problems and solution progress and all rely on models generated off-line from problem instances to adapt on-line.

Most approaches also leverage statistical run-time distribution information. For example, Horvitz et al. [23] constructed Bayesian models to predict the total run-time of an algorithm solving a class of problems. Models were constructed for each algorithm in each application domain based on many problem instances. Features, both application and algorithm specific as well as more general, were collected at search choice points during a probing phase (which they call the observation horizon). The models were extremely accurate and can be used as the basis for a dynamic restart policy. Guerri and Milano [21] focus on identifying features that

uncover *structure* of the problem instance. They extract features from the common problem space representation for combinatorial auctions (a specialized graph). These features can be used to very accurately predict algorithm performance on specific instances.

4 Relationship to the COORDINATORS Program

The literature provides a wealth of techniques that are applicable to COORDINATORS. None of them on their own can be inserted in a plug-and-play fashion. However, given the distributed nature of the COORDINATORS problem and the fact that full scale planning is not being considered, even some of the optimal approaches described in this document can be applied to selected sub-problems. In this section, I offer my opinion about which are most promising and which are unlikely to be successful.

Scheduling Following the military hierarchy of command, the scheduling problem is probably two different problems: command level and executor level. Each requires different approaches. The command level must monitor and coordinate a large number of activities, seen at a distance (with the delay and uncertainty about state entailed by the disconnect). The approaches to dynamic scheduling that exploit search are well suited to this, as a starting point. However, I know of no off-the-shelf techniques that will fully do the job. A good first approach is heuristics embedded within search frameworks.

The executor level will have a much narrower, but better informed view of activity. Because the amount of activity is constrained, optimal or near optimal techniques may be tractable. My understanding is that commercial CSP has been used by the Cornell group to solve scheduling sub-problems at this level. It doesn't surprise me that these techniques might work as they have been successfully applied in large, well-defined manufacturing scheduling for some time. The key is casting the problem into the framework. For integration, they will have to address two issues: translating representations from TAEMS and other sources into the CSP constraints and deciding how to accommodate uncertainty.

Much as I favor meta-heuristic approaches (e.g., local search, evolutionary algorithms) in my own research, they are better suited to applications in which the representation is simple/well defined and full solutions can be manipulated.

Action Selection is a bigger challenge in COORDINATORS. Control theory based approaches, such as POMDPS, require considerable computation and considerable knowledge about the problem currently at hand. The major stumbling block is defining the state space so that it can be tractable, which is more art than science. The result of computation is brittle and does not generalize well to other similar problems, and the output (recommendation) is sometimes difficult for a person to appreciate.

Although they are more knowledge intensive, heuristic contingent planning approaches are more likely to possess the characteristics most needed for COORDINATORS. The most general of these still have not been demonstrated on anything but toy problems. However, NASA's heuristic approaches based on critics have been shown to be effective in real applications. They are more ad hoc, but may fit well within the COORCINATORS framework.

References

[1] U. Aickelin and P. White. Building better nurse scheduling algorithms. *Annals of Operations Research*, 128:159–177, 2004.

[2] Luis Baptista and Joao P. Marques Silva. Using randomization and learning to solve hard real-world instances of satisfiability. In *Principles and Practice of Constraint Programming*, pages 489–494, 2000.

[3] L. Barbulescu, J.P. Watson, L.D. Whitley, and A.E. Howe. Scheduling space-ground communications for the air force satellite control network. *Journal of Scheduling*, 7(1):7–34, January 2004.

[4] R. Bartak. Constraint-based scheduling: An introduction for newcomers. In Kadar, Monostori, and More, editors, *Intelligent Manufacturing Systems 2003*, pages 69–74. IFAC Publications, Elsevier Science, 2003.

[5] J. Christopher Beck and Eugene C. Freuder. Simple rules for low-knowledge algorithm selection. In *Proceedings of the First International Conference on Integrati on of AI and OR Techniques in Constraint Programming for Combinatorial Optimisat ion Problems (CPAIOR'04)*. Springer-Verlag, 2004.

[6] E. Benazera. Alternatives to re-planning: Methods for plan re-evaluation at runtime. In *Proceedings of ICAPS 2005 Workshop on Plan Execution: A Reality Check*, Monterey, CA, USA, June 2005.

[7] P. Bertoli, A. Cimatti, M. Roveri, and P. Traverso. Planning in nondeterministic domains under partial observability via symbolic model checking. In *Proceedings of IJCAI-01*, Seattle, WA, USA, 2001.

[8] J. Bidot, P. Laborie, J.C. Beck, and T. Vidal. Using simulation for execution monitoring and on-line rescheduling with uncertain durations. In *Proceedings of ICAPS'03 Workshop on Plan Execution*, 2003.

[9] J. Bresina, R. Dearden, N. Meuleau, S. Ramakrishnan, D. Smith, and R. Washington. Planning under continuous time and uncertainty: A challenge for ai. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pages 77–84, 2002.

[10] A.R. Cassandra, L.P. Kaelbling, and M.L. Littman. Acting optimally in partially observable stochastic domains. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Seattle, WA, USA, 1994.

[11] A. Cesta and R. Rasconi. Execution monitoring and schedule revision for o-oscar: a preliminary report. In K. Brown C. Beck and G. Verfaillie, editors, *Proceedings of Online-2003 (Online Constraint Solving: Handling Change and Uncertainty)*, pages 9–23, Kinsale, Ireland, September 2003.

[12] S. Chien, R. Knight, A. Stechert, R. Sherwood, , and G. Rabideau. Using iterative repair to increase the responsiveness of planning and scheduling for autonomous spacecraft. In *Proceedings of International Conference on Artificial Intelligence Planning Systems (AIPS 2000)*, pages 300–307, Breckenridge, CO, USA, April 2000.

[13] S. Chien, G. Rabideau, R. Knight, R. Sherwood, B. Engelhardt, D. Mutz, T. Estlin, B. Smith, F. Fisher, T. Barrett, G. Stebbins, and D. Tran. ASPEN - automating space mission operations using automated planning and scheduling. In *SpaceOps 2000*, Toulouse, France, June 2000.

[14] A.J. Davenport and J.C. Beck. A survey of techniques for scheduling with uncertainty. http://www.eil.utoronto.ca/EIL/profiles/chris/zip/uncertainty-survey.ps.zip.

[15] Eugene Fink. How to solve it automatically: Selection among problem solving methods. In *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems*, pages 128–136, 1998.

[16] F. Focacci, P. Laborie, and W. Nuijten. Solving scheduling problems with setup times and alternative resources. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling*, pages 92–101, Breckenridge, CO, USA, April 2000.

[17] D. R. Fulkerson. A network flow computation for project cost curves. *Management Science*, 7(2):167–178, January 1961.

[18] C. Gomes and B. Selman. Practical aspects of algorithm portfolio design. In *Proc. of Third ILOG International Users Meeting*, 1997.

[19] Carla P. Gomes and Bart Selman. Algorithm portfolio design: Theory vs. practice. In *Proceedings of the Thirteenth Conference On Uncertainty in Artificial Intelligence (UAI-97)*, Linz, Austria., 1997. Morgan Kaufman.

[20] Jonathan Gratch and Steve Chien. Adaptive problem-solving for large-scale scheduling problems: A case study. *Journal of Artificial Intelligence Research*, 4:365–396, 1996.

[21] Alessio Guerri and Michela Milano. Learning techniques for automatic algorithm portfolio selection. In *Proceedings of the 16th biennial European Conference on Artificial Intelligence*, pages 475–479, Valencia, Spain, August 2004.

[22] E. Hart, P. Ross, and D. Corne. Evolutionary scheduling: A review. *Genetic Programming and Evolvable Machines*, 6:191–220, 2005.

[23] Eric Horvitz, Yongshao Ruan, Carla P. Gomes, Henry Kautz, Bart Selman, and David Maxwell Chickering. A bayesian approach to tackling hard computational problems. In *Proceedings of the Seventeenth Conference on Uncertainty and Artificial Intelligence*, pages 235–244, 2001.

[24] Adele E. Howe, Eric Dahlman, Christopher Hansen, Anneliese von Mayrhauser, and Michael Scheetz. Exploiting competitive planner performance. In *Proceedings of Fifth European Conference on Planning (ECP-99)*, Durham, England, September 1999.

[25] Bernardo A. Huberman, Rajan M. Lukose, and Tad Hogg. An economics approach to hard combinatorial problems. *Science*, 275:51–54, January 3 1997.

[26] O. Icmeli and S. Erenguc. The resource constrained time-cost tradeoff project scheduling problem with discounted cash flows. *Journal of Operations Management*, 14:255–275, 1996.

[27] ILOG. Three tiered approach to scheduling. http://www.ilog.com/products/scheduler/threetier.cfm, September 2006.

[28] M. Jensen. Finding worst-case flexible schedules using coevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*, pages 1144–1151, San Fransisco, CA, USA, 2001.

[29] David Johnstone and Steven Bradley. Opportunistic scheduling in a constraint-rich world. In *Proceedings of IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2005)*, San Francisco, CA, March 2005.

[30] David E. Joslin and David P. Clements. "squeaky wheel" optimization. In *Journal of Artificial Intelligence Research*, volume 10, pages 353–373, 1999.

[31] I. E. Kelley and M. R. Walker. *Critical Path Planning and Scheduting: An Introduction.* Mauchly Associates, Inc., Ambler, PA, 1959.

[32] L.A. Kramer and S.F. Smith. Maximizing flexibility: A retraction heuristic for oversubscribed scheduling problems. In *Proceedings of 18th International Joint Conference on Artificial Intelligence*, Acapulco, Mexico, August 2003.

[33] P. Laborie. Algorithms for propagating resource constraints in ai planning and scheduling: Existing approaches and new results. *Artificial Intelligence*, 143:151–188, 2003.

[34] Michail G. Lagoudakis and Michael L. Littman. Algorithm selection using reinforcement learning. In *Proc. 17th International Conf. on Machine Learning*, pages 511–518. Morgan Kaufmann, San Francisco, CA, 2000.

[35] Kevin Leyton-Brown, Eugene Nudelman, Galen Andrew, Jim McFadden, and Yoav Shoham. A portfolio approach to algorithm selection. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, Acapulco, Mexico, August 2003.

[36] Steven Minton. Automatically configuring constraint satisfaction programs: A case study. *Constraints*, 1(1/2):7–43, 1996.

[37] Steven Minton, Mark D. Johnston, Andrew B. Philips, and Philip Laird. Minimizing conflicts: A heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence,*, 58:161–205, 1992.

[38] E. Nowicki and C. Smutnicki. *Metaheuristic Optimization Via Memory and Evolution*, chapter Some New Ideas in TS for Job Shop Scheduling, pages 165–190. Kluwer Academic Publishers, 2005.

[39] D. Ouelhadj, P. Cowling, and S. Petrovic. Utility and stability measures for agent-based dynamic scheduling of steel continuous casting. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'2003)*, pages 175–180, Taipei, Taiwan, 2003.

[40] R. Petrick and F. Bacchus. Extending the knowledge-based approach to planning with incomplete information and sensing. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS-04)*, pages 2–11, 2004.

[41] P. Poupart and C. Boutilier. VDCBPI: an approximate scalable algorithm for large POMDPs. In *Advances in Neural Information Processing Systems 17 (NIPS)*, pages 1081–1088, Vancouver, CA, 2004.

[42] Pascal Poupart. Approximate value-directed belief state monitoring for partially observable markov decision processes. Department of cmputer science, University of British Columbia, November 2000.

[43] Simmons R and S. Koenig. Probabilistic robot navigation in partially observable environments. In *Proceedings of IJCAI 95*, Montreal, CA, July 1995.

[44] G. Rabideau, B. Englehardt, and S. Chien. Using generic preferences to incrementally improve plan quality. In *Proceedings of International Conference on Artificial Intelligence Planning Systems (AIPS 2000)*, pages 236–245, Breckenridge, CO, USA, April 2000.

[45] G. Rabideau, R. Knight, S. Chien, and A. Govindjee. Iterative repair planning for spacecraft operations using the aspen system. In *International Symposium on Artificial Intelligence Robotics and Autonomy in Space (ISAIRAS)*, Noordwijk, The Netherlands,, June 1999.

[46] E. Ramat, G. Venturini, C. Lente, and M. Simane. Solving the multiple resource constrained project scheduling problem with a hybrid genetic algorithm. In T. Back, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 489–496, 1997.

[47] S. Rana, A.E. Howe, L.D. Whitley, and K. Mathias. A genetic algorithm scheduler: Assessing the contribution of search, heuristics and the objective function. *Engineering Design and Automation Journal*, 3(2):107–118, 1997.

[48] Hani El Sakkout and Mark Wallace. Probe backtrack search for minimal perturbation in dynamic scheduling. *Constraints*, 5(4):359–388, October 2000.

[49] Neal Sample, Pedram Keyani, and Gio Wiederhold. Scheduling under uncertainty: Planning for the ubiquitous grid. In *Fifth International Conference on Coordination Models and Languages*, pages 300–316, April 2002.

[50] Thomas Schwarzfischer. Quality and utility - towards a generalization of deadline and anytime scheduling. In *ICAPS*, pages 277–286, 2003.

[51] D. Smith and D. Weld. Conformant graphplan. In *Proceedings of AAAI-98*, pages 889–896, 1998.

[52] T. Smith and R. Simmons. Point-based pomdp algorithms: Improved analysis and implementation. In *Proceedings of UAI 2005*, 2005.

[53] F. Talbot. Resource-constrained project scheduling with time-resource tradeoffs: the non preemptive case. *Management Science*, 28:1197–1210, 1982.

[54] Xiaofang Wang and Stephen Smith. Generating schedules to maximize quality. Technical report, Robotics Institute, Carnegie Mellon University, April 2004.

[55] S.D. Wu, E. Byeon, and R.H. Storer. A graph-theoretic decomposition of the job shop scheduling problem to achieve scheduling robustness. *Operations Research*, 47(1):113–124, 1999.

[56] Shlomo Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3):73–83, 1996.

[57] Shlomo Zilberstein, Francois Charpillet, and Philippe Chassaing. Optimal sequencing of contract algorithms. *Annals of Mathematics and Artificial Intelligence*, 39(1-2):1–18, 2003.