

# WELL ACTIVITY SCHEDULING - AN APPLICATION OF CONSTRAINT REASONING

G. Hasle<sup>1</sup>, R. C. Haut<sup>2</sup>, B. S. Johansen<sup>1</sup>, T. S. Ølberg<sup>2</sup>

<sup>1</sup>SINTEF Informatics, P.O. Box 124, N-0314 Oslo, Norway

<sup>2</sup>Saga Petroleum a.s., P.O. Box 117, N-4033 Forus, Norway

**Abstract:** The relatively low level of oil prices has in the past years drawn attention to cost reduction remedies in oil companies. One way of reducing costs is to improve coordination of activities and resources. The production of oil offshore involves a substantial number of activities in a set of wells. These activities have to be coordinated in order to obey various types of constraints and, at the same time, optimize on criteria such as the production level. *Well Activity Scheduling* (WAS) deals with the proactive coordination of these activities to form Well Activity Schedules, as well as the dynamic maintenance of these schedules over time.

In a cooperative project affiliated with the Esprit project TRUTH, Saga Petroleum a.s. and SINTEF Informatics have studied the WAS problem. In the project we have developed a constraint model, performed problem analysis, and developed a prototype system for WAS. The prototype system includes functionality for defining a particular WAS problem and generating a high-quality, feasible solution. The user may inspect and modify the generated schedule through interactive Gantt visualizers. Constraint violations are detected and presented to the user. The WAS prototype has been developed using a commercial constraint programming tool. The prototype is running on Unix/X11 and PC/Windows platforms.

Preliminary empirical investigations using the WAS prototype system on historical data indicates a high potential for improvement, although the current system is based on relatively straightforward heuristics and no automatic optimization facilities.

## INTRODUCTION

Oil production offshore provides a demanding environment for scheduling. The production of oil offshore involves a substantial number of activities in a set of wells. These activities have to be coordinated in order to obey various types of constraints and, at the same time, optimize on criteria such as the production level. *Well Activity Scheduling* (WAS) deals with the proactive coordination of these activities to form Well Activity Schedules, as well as the dynamic maintenance of these schedules over time. WAS is a distributed scheduling problem, involving constraints and criteria emerging from various departments within the oil company. This may require negotiation in the scheduling process. The main activity types to be scheduled are drilling, completion, perforation, and logging. There are two major (capacity 1) resources: the Drilling Rig and the Wireline Crane. For some activity types, these resources are interchangeable. For others, the Drilling Rig is the only possible resource. Activity durations are dependent on the sequence of activities because skidding of the drilling rig has to be included. The physical footprint of the drilling rig is the basis of idiosyncratic constraints.

Scheduling is a hard problem [5], involving combinatorial complexity, dynamics and uncertainty. WAS is a complex scheduling problem. The WAS characteristics call for a decision support approach with excellent user interface facilities including manual revision. An operational WAS decision support system will constitute a vehicle for keeping updated and integrated information on well activity schedules. In this way, human resources spent in scheduling may be reduced, and there will be more opportunities for exploring alternatives, e.g., through manual schedule revision. Through automatic optimization facilities one may further improve schedule quality.

In a typical WAS problem, several hundred activities must be scheduled over 4-5 years or more in 40-50 oil wells. At a reasonable level of granularity (and depending of type of well) there are 4 drilling phases, 2 completion phases, and perforation for each well. In addition there are logging activities that are performed regularly or based on status information. Activity durations are uncertain, and may vary according to season. As time passes, unexpected events occur and new information arrives. This calls for relatively frequent and substantial revisions of the schedule. Today, the WAS process is performed with little computer support and consumes considerable human resources.

The purpose of this paper is to describe the WAS problem and its characteristics, the constraint model we have developed, and our

constraint based scheduling approach. We shall present the prototype system and some preliminary results on historical data. In conclusion, we shall point to possible further developments and industrial impacts. First, we shall give a brief introduction to the field of Constraint Reasoning.

## **CONSTRAINT REASONING**

Constraint Reasoning (CR) is a relatively new field within computer science. It consists of generic methods for solving complex problems involving constraints. Such problems are abundant in industry and administration: planning, scheduling, resource allocation, control, and design. Constraint Reasoning is reasoning by active use of constraints. A set of methodologies has been developed, largely over the past two decades, and very significant results have been achieved. In recent years we have also seen the advent of industrial and academic tools based on these methodologies [3]. The Constraint Reasoning paradigm of computation inherits from the field of Artificial Intelligence the flexibility and versatility required to express very complex problems, and from Linear Programming and Operations Research the efficiency to solve them.

Constraint Reasoning is now often identified with issues relating to industrial competitiveness and efficiency. The class of constraint satisfaction problems covers almost any resource allocation task such as personnel scheduling in hospitals, job scheduling in factories, production planning, etc. CR methodologies have successfully automated the solution of complex combinatorial problems in many domains as diverse as planning, resource allocation, time management, personnel management, partitioning, and design.

In more detail, Constraint Reasoning deals with methodologies for representing constraints and objectives in tandem with methodologies for reasoning over such entities in order to arrive at solution to problems. An increasing number of researchers all over the world are now dealing with different aspects of constraint processing regarded as a general paradigm of computation. The constraint processing community is heterogeneous: Researchers from Logic Programming, Knowledge Representation, theoretical Computer Science, Operations Research and other related fields are investigating the use of constraint

processing methods, their theoretical foundations, as well as their applications to real-life problems. In the CR field there is a strong connection between theory and practice. Real-life applications (e.g., difficult or large-size scheduling problems) pose new challenges to effective constraint representation and resolution methods. Novel theoretical results in CR move the borderline between effectively solvable and unsolvable problems even more substantially than the development of more powerful computers.

Broadly speaking, Constraint Reasoning addresses two general problems: constraint satisfaction (CSP) and constrained optimization (COP) [13]. A CSP involves finding values for variables, subject to constraints on acceptable combinations of values. A COP may be regarded as a CSP with the addition of one or more goal criteria that discriminate between feasible solutions. Many important real world problems (e.g., in design, Operations Research) are naturally cast as CSPs or COPs. For certain types of CSP and COP, there exist well-established, and very effective techniques (e.g., the Simplex algorithm for solving Linear Programs). CR generally attacks problems that go beyond the capabilities of conventional techniques. In particular, CSP/COP formulates discrete, combinatorial problems for which there are no direct methods. They must be solved by weak methods (i.e., search). Moreover, the computational complexity of these problems forces one to apply non-complete, heuristic search. In CR (as opposed to parts of Operations Research), one focuses more on finding satisfactory solutions to realistic problem formulations than finding optimal solutions to idealized problem formulations.

CR also emphasizes the need for rich representations of constraints in order to support mixed-initiative problem solving processes between humans and computers rather than black-box solutions. Lately, considerable focus has been put on *reactive* problem solving capabilities, i.e., the ability to rapidly maintain a previously generated solution that has become infeasible or inferior due to unexpected events in the environment, and techniques for *interactive* constraint reasoning in which the computer and the human user cooperates tightly during the resolution process.

The current research topics in CR very much address issues related with real-life applications. They include representational adequacy for practical problems (e.g., representing constraints with priorities, hardness, relaxation alternatives) as well as methods for effective and efficient resolution of constraints. Moreover, there is considerable

research and development on frameworks that combine declarative semantics with efficient, modifiable resolution methods. In practical terms, these frameworks allow a very rapid (re)formulation of practical problems (which is needed to develop and maintain cost-efficient tools) as well as a flexible repertoire of efficient algorithms that combine generality with domain-specificity. Domain specificity (in the form of domain heuristics) is generally needed to achieve efficiency in real-life applications. For more detail, we refer to [1,2, 9].

Besides being at the core of theoretical computer science, CSPs and COPs are abundant in complex, practical decision making areas such as design, planning, resource allocation, scheduling, and control. The quality of decision making processes in these areas are critical to productivity, quality and efficiency in industries (manufacturing, transportation and services) as well as administration. Hence, effective methods for representation and resolution of CSPs and COPs, combined with efficient systems engineering methods and techniques, may contribute substantially to a healthy economy. There are also application areas where the economical effects are more indirect. In the area of modelling and simulation, CR may provide enhanced functionalities and accuracy, e.g., what-if functionalities for macro-economic simulation models.

- We may list the following important application areas for CR:
- manufacturing design
- architectural design
- electronics design
- manufacturing management
- transportation management
- personnel management
- project management
- process modelling, monitoring and control
- economic modelling

Generally speaking, automation through computer systems in these areas are particularly challenging. Automation is highly desirable due to the complexity which overwhelms humans and makes it necessary to enhance their rationality through computers. At the same time, formal

methods do have limited value due to the presence of informal knowledge. The solution is generally mixed-initiative decision-support systems with real-time response.

## THE WELL ACTIVITY SCHEDULING PROBLEM

Well Drilling, Completion, and Intervention Scheduling (we shall often use the abbreviation *Well Activity Scheduling, or WAS*) is the process of creating and maintaining detailed schedules for drilling, completion and intervention activities in oil wells. The creation and maintenance of feasible and high quality *Well Activity Schedules* is important to ensure optimal coordination of well activities.

Schedules must adhere to a number of physical, security, resource capacity, and temporal constraints. The major constraints in WAS are:

- sequencing and timing constraints between activities for each well
- sequencing and timing constraints between wells
- security constraints related with drilling in reservoir
- security constraints related with possible interference problems between wells
- security constraints related with the movement (skidding) of the drilling rig
- security constraints related with protection against falling objects ('dropped object protection').
- constraints on resource allocation
- constraints on the number of parallel activities
- constraints on the number of activities per year (e.g., routine data acquisition)
- physical constraints (e.g., originating from the geometry of the drilling rig)
- resource capacity constraints (e.g., the process unit capacity)
- logistics constraints (e.g., transportation capacity, capacity for accommodation)

A high quality Well Activity Schedule must contain flexibility. Unexpected events will occur during well activities. New information from the reservoir, the well, or the process unit may render the current schedule infeasible or sub-optimal. Several

alternative remedies may be relevant in a given situation. Hence, it is important not only to be able to generate high quality schedules, but also to have the capability of rapidly revising the schedule to explore the effects of alternative revisions. In the schedule revision process both economy and flexibility must be maintained.

A goal for WAS is to minimize the time until full production, i.e., the time until total production reaches the processing capacity limit. This “time until full production” is partially determined by the sequencing of well drilling and completion activities and the assignment of *template slots* to wells. When the processing capacity has been reached, a typical goal is to coordinate the remaining well activities in such a way that the total production never falls below the processing capacity. Moreover, it is important to minimize the time until all wells are completed. This will reduce platform and inventory costs, and enhance the flexibility to optimize on the net present value of the oil production. Coordination of activities towards the end of the field lifetime will also give benefits.

A more indirect goal is to maximize the utilization of the Drilling Rig, as the Drilling Rig is a critical resource. This is done by minimizing the time when the Rig is not in use, minimizing the time when the Rig is used for activities that may be performed by other resources (e.g. the Wireline Crane), and minimizing the Rig movement time.

From the above sketch of the problem, it is clear that WAS is a complex, dynamic, multi-criterion, constrained optimization problem where the optimization criteria are partially in conflict. The complexity and dynamics motivate the introduction of a decision-support system with functionality for automatic schedule generation and revision.

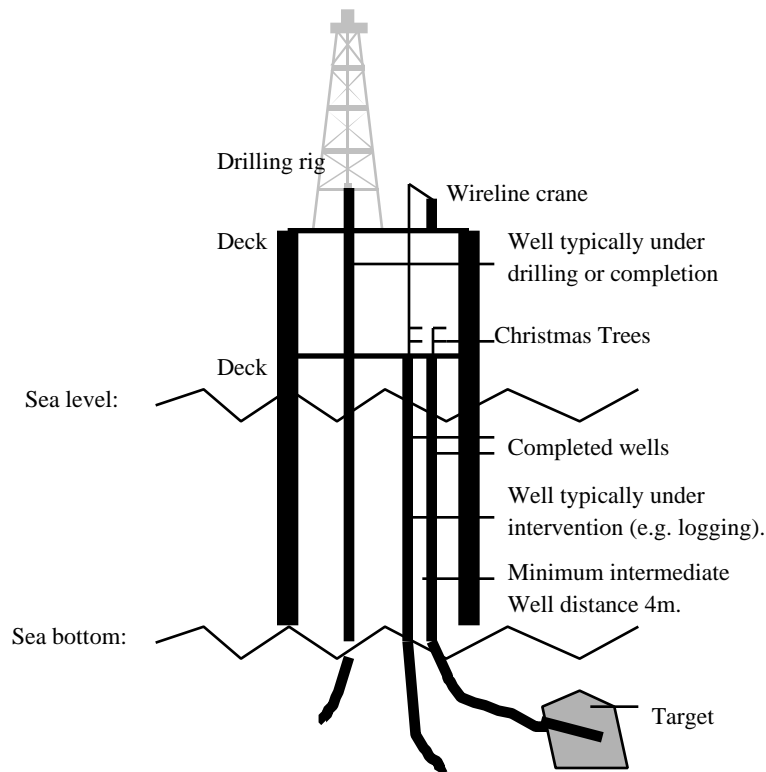
The aim of the present pilot project has been to demonstrate the feasibility of such a system through the implementation of a Well Activity Scheduler Prototype. In the project, as documented in the following description, a subset of the constraints mentioned above are considered. This subset covers the central aspects of the Well Activity Scheduling Problem. Our claim is that the realization of a WAS system based on a reduced constraint model strongly indicates the feasibility of a fully adequate Well Activity Scheduling kernel for decision support. A successful WAS implementation is also depending on the successful integration of a WAS kernel with existing systems.

## **THE CONSTRAINT MODEL**

We shall now describe the main entities of the prototype constraint model for the WAS problem. To this end we shall now give a fairly high level description of activities, resources, and constraints.

## The oil platform

A simplified drawing of an oil platform is shown in Figure 1.



**Fig. 1.** Simplified drawing of Oil Platform. Each well is either a Producer, an Injector, a Pilot hole or an Observer.

## Well activities

For each well, all drilling, completion, and logging activities are represented. In addition, we have modelled maintenance activities. The drilling and completion activities is naturally divided into *phases*, where each phase consists of a set of activities that must be executed consecutively on the same resource, and therefore can be treated as one entity.

- **DRILLING.** Each drilling activity must be performed with the Drilling Rig. Drilling is subdivided into 4 phases based on case settings. Drilling may be temporarily halted upon the completion of each phase. Total duration is approximately 60-80 days.
- **COMPLETION.** Each completion activity is subdivided into 2 phases, where the activities in phase 1 may be performed only by the Drilling Rig, but the activities in phase 2 is preferably performed by the Wireline Crane, but may be performed by the Drilling Rig. Completion phase 2 must follow immediately after completion phase 1. Total duration is approximately 8+3 days.
- **PERFORATION.** First phase perforation may be modelled as a part of the completion activity. Second phase perforation is similar to, and may be adequately modelled as a logging activity.
- **LOGGING.** Either Routine Data Acquisition, which is performed regularly, or Performance Based Activities, which are scheduled according to the status of the well (e.g., 20%, 50%, and 80% watercuts). Preferably performed by the Wireline Crane, but may be performed by the Drilling Rig. Typical duration 2-3 days.
- **GAUGING.** (Pressure build up (PBU) & static bottom hole pressure (static BHP)). In the first phase, a gauging instrument is inserted into a closed well. In the second phase the pressure in the well is measured. In the third phase, the gauging instrument is retrieved. For each well, PBU or Static BHP is performed annually, preferably during maintenance. Duration: PBU 2 days, Static BHP minimum 1 week.
- **MAINTENANCE.** All wells are closed. Maintenance is performed annually and (in the North Sea) typically starts in September. May be coordinated with maintenance on other platforms. Duration: 2-10 weeks.

## Resources

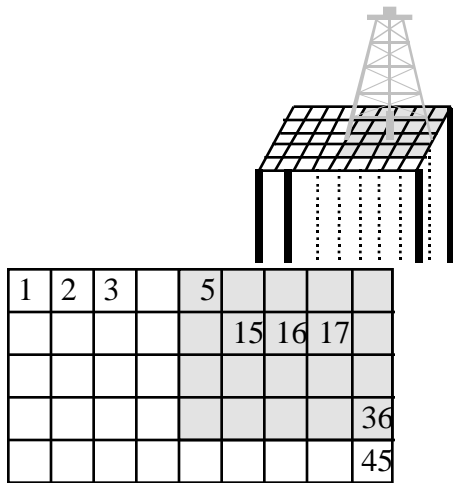
Well activities are performed using several kinds of *resources*. The resources considered in the prototype are:

- **THE DRILLING RIG.** The drilling rig is the most constrained resource and may perform only one activity at a time. A high quality schedule must ensure a good

utilization of this resource for the most critical activities, which are drilling and completion.

- **THE WIRELINE CRANE.** The Wireline Crane may also perform only one activity at a time. It is used for phase 2 completion, logging activities, and gauging.
- **THE TEMPLATE WITH SLOTS.** Each well is assigned to a specific *slot* in a template. A typical number of slots is 45. Each slot may be regarded as a fixed capacity resource. Constraints originating from the geometry of the drilling rig makes slot availability for the Wireline Crane (and other resources) dependent on the current positioning of the Drilling Rig. The corresponding constraint is called the *footprint constraint*.

In Figure 2 it is assumed that the Rig is presently performing a drilling activity in slot 16. All slots colored with gray are not available to the Wireline Crane.



**Fig. 2.** The footprint constraint. The physical extension of the Drilling Rig makes all slots colored by gray inaccessible when the Drilling Rig is located at slot 16.

In the prototype constraint model, the following resources were not included: The Process Unit, the Workover Rig, the Coil Tubing Equipment, the Snubbing Unit, and Crew.

## Constraints

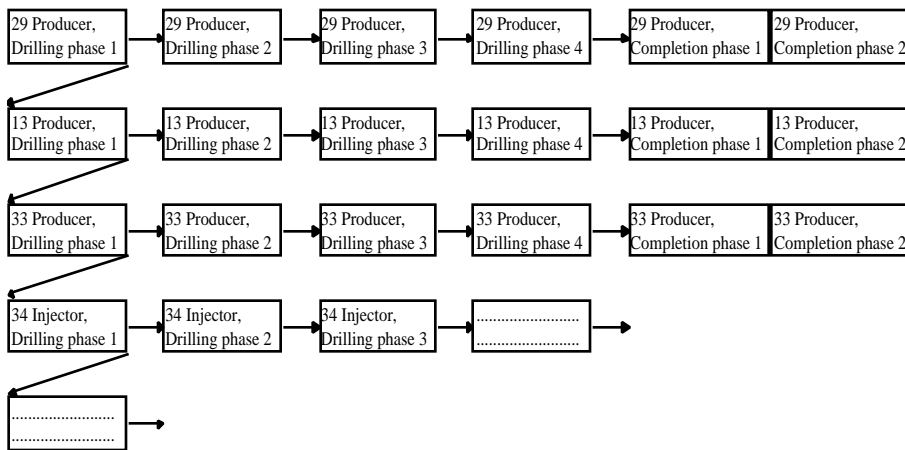
The following constraints were included in the prototype constraint model:

- *Technological precedence constraints.* The overall sequence of drilling and completion activities for each well is as follows: Drilling phase 1 -> Drilling phase 2 -> Drilling phase 3 -> Drilling phase 4 -> Completion phase 1 -> Completion phase 2. We refer to Figure 3 for illustration.
- *Well sequence constraints.* From a *Field Development Plan*, a *well sequence* is established<sup>1</sup>. The well sequence is defined to specify precedences between Phase 1 Drilling activities. (See Figure 3 for illustration). There are alternative (and more constraining) ways of interpreting the Well sequence constraints.
- *Footprint constraints.* The footprint constraint is an idiosyncratic constraint type for the WAS problem. The footprint constraint expresses a rather complex time and resource relationship between the activities in a given well, and activities in slots that are covered by the footprint of the slot to which this well is allocated. It is possible to model footprint constraints by using auxiliary resources and “artificial” capacities for the Rig and the Crane capacity 1 resources.
- *Sequence-dependent setup.* A certain (fixed) time is required to move the Rig from a given slot to another.<sup>2</sup>
- *Temporal coupling.* For all producer wells, one Gauging activity is scheduled in connection with Maintenance.

---

1. Also, a spider plot and a slot allocation is made from the Field development plan by a directional drilling company.

2. This is a simplification. Related with the skidding of the drilling rig between slots there is a non-trivial path optimization problem. Depending on the path, security constraints may necessitate closing of certain wells.



**Fig. 3.** Temporal precedences. Logging activities following Completion phase 2 are not shown.

## THE SCHEDULING PROCESS

In constraint-based scheduling, a critical issue is to encode the given problem as a CSP [7]. The WAS prototype constraint model was implemented using a commercial constraint solver, namely Ilog Solver with the Ilog Schedule predefined objects for resources and activities. The basic resolution process consists of an initial pruning of the search space through domain reduction via arc consistency techniques, and a subsequent search phase, where commitments are made in an opportunistic fashion [6, 8]. The consequences of these commitments are used to filter domains through constraint propagation.

The temporal constraints illustrated in Figure 3 are posted to the constraint solver and propagated prior to search. In a similar way, time window constraints on the Logging activities are posted and propagated. Each time a resource commitment is made, the corresponding resource availability is updated. Rig movement time and the insertion of Gauging activities for Producer wells are taken care of during search for a feasible schedule. The Footprint constraint is modelled using auxiliary resources.

Our selected scheduling strategy requires the ability to revise earlier commitments made during search. Such non-monotonicity is not supported by the tools used<sup>3</sup>. Special treatment takes care of situations when resource commitments are remade.

## THE PROTOTYPE WELL ACTIVITY SCHEDULER

Based on the description of the WAS problem, a prototype Well Activity Scheduler, called WAS 0.1, has been developed, based on the problem model described above. The prototype solves central aspects of the WAS problem:

- Activities are modelled at an adequate level of abstraction
- The most important resources, i.e., the Drilling Rig and the Wireline Crane are modelled
- The most important constraints, e.g., well sequence, maintenance timing, coupling of gauging and maintenance activities, resource capacity, and footprint constraints are modelled.
- The WAS 0.1 scheduling strategy alluded to above is able to generate a feasible, high quality schedule for real life cases, taking all operations and constraints into consideration.
- WAS 0.1 contains functionality for manual revision of schedules by moving operations in time or between resources.
- Constraint violations during manual revision are detected and presented to the user.

### User input

For each well, duration estimates for the four drilling phases, the two completion phases, and the logging activities must be given. For each logging activity, early and late start (relative to finished completion) must be given. Figure 4 shows the user interface for defining a new well. After all wells have been defined, a well sequence must be specified. This is done in an intuitive way. To conclude the definition of a particular WAS problem, the footprint diagram must be specified. The

---

3. Although support for non-monotonicity is not a part of Ilog Solver and Schedule, these tools have the flexibility to implement non-monotonicity, but at the programmers own risk.

interface is shown in figure 5. A black rectangle at row  $i$  and column  $j$ , indicates that the Wireline Crane may not access slot  $j$  when the Drilling Rig is located at slot  $i$ .

**Create new Well**

Well number:

TYPE OF WELL:

	Early start	Late start	Duration
Drilling phase 1:	<input type="text"/>	<input type="text"/>	<input type="text" value="5.0"/>
Drilling phase 2:	<input type="text"/>	<input type="text"/>	<input type="text" value="11.9"/>
Drilling phase 3:	<input type="text"/>	<input type="text"/>	<input type="text" value="15.6"/>
Drilling phase 4:	<input type="text"/>	<input type="text"/>	<input type="text" value="21.9"/>
Completion phase 1:	<input type="text"/>	<input type="text"/>	<input type="text" value="10.0"/>
Completion phase 2:	<input type="text"/>	<input type="text"/>	<input type="text" value="2.5"/>

Logging activities:	Name	Early start	Late start	Duration
	perforation_p1	12	16	2
	PLT_1	25	30	3
	PLT_2	50	60	4
		0	0	0

(Start and end given relative to finished completion for logging activities)  
 (One 'Kjoering' activity prior to maintenance, and one 'Trekking' activity after maintenance, automatically scheduled for each producer each year)

**Fig. 4** Specifying time estimates.

**Erreur! Impossible de lire ni d'afficher le fichier.**

**Fig. 5** Footprint diagram. for the well activities.

## Generating, displaying and editing a schedule

Was 0.1 is able to generate a feasible, but possibly not optimal schedule. We refer to Sections 6 and 7 for a description of results from an initial, empirical investigation and discussions of further work on schedule optimization.

A schedule is displayed through two types of Gantt-diagrams. A Resource Gantt focuses shows scheduled activities for all resources (presently the Drilling Rig and



## **Implementation**

The program was implemented in a modular, object-oriented, fashion, using C++ and the ILOG tools Solver, Schedule, and Views. These are C++ libraries facilitating the implementation of search problems, scheduling and powerful graphical user interfaces. We found the tools easy to use and efficient, both in terms of systems development time, and system performance. We perceived the basic non-monotonic property of ILOG Solver to be restrictive in the implementation of scheduling strategies for the WAS problem.

## **EMPIRICAL INVESTIGATION**

In the prototype project, we approached empirical assessment of the performance of the prototype through scheduling on the basis of one set of historical data.

The assessment of the WAS prototype was performed according to three major criteria:

- the efficiency and ease of use of the user interface
- the completeness of functionality
- the quality of schedules generated (in terms of makespan)

By comparison with the actual schedule executed and known characteristics of the manual scheduling process, the assessment also indicated the potential for improvement by introducing a WAS scheduling tool, both in terms of improvements of the scheduling process, and improvements of schedule quality.

The criteria were assessed by running the WAS prototype as a standalone system with historical data. A potential end user from Saga Petroleum performed the specification of a particular WAS problem before running the automatic schedule generation facility. Subsequently, the schedule was improved by the Saga end user through manual editing. The assessment was performed in January '95 at the Saga Petroleum premises at Forus. A more detailed description of the assessment will now follow. We are aware of inherent methodological problems in our 'historical data' approach to assessment caused by the lack of tracking information on the unexpected events that may have influenced the scheduling process.

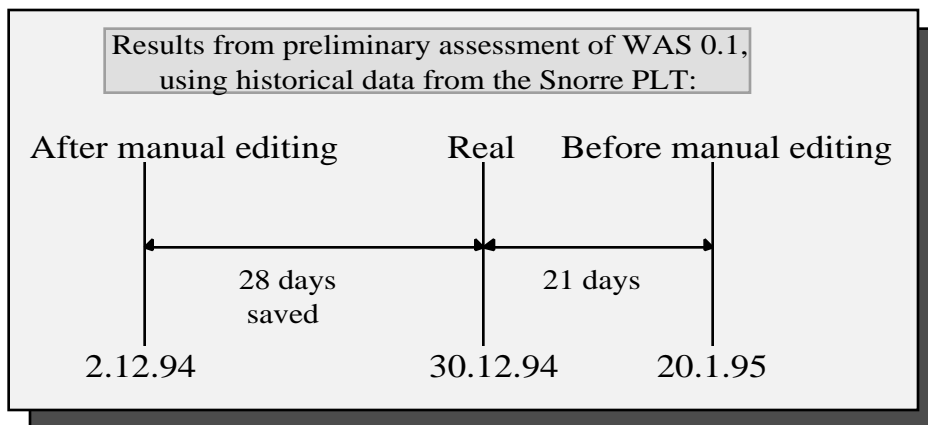
Historical Snorre PLT data in the form of well sequence, activity durations, and rig footprint was entered in the system as the problem definition. Drilling, completion, and logging activities from the first 17 wells were specified, corresponding to a

schedule horizon of approximately 2 \_ years. For logging activities, a time window for execution was given, relative to the completion of the completion activity for the well in question. The WAS prototype generated an initial, feasible schedule in about 3 minutes. The Saga end user then spent 2 hours of manual editing, using the graphical interface.

The above procedure revealed a few weaknesses in the user interface, as well as novel requirements for functionality. The overall Saga end user impression of the prototype was very positive.

The response time of 3 minutes for automatic generation of an initial, feasible schedule was deemed satisfactory, also

The makespan of the automatically generated initial schedule was 21 days longer than the makespan of the real schedule. This is not acceptable for a final system, and points to the need for development of stronger WAS scheduling strategies with the utilization of non-monotonic search. After 2 hours of manual editing, the Saga user was able to reduce the makespan by 49 days, thus achieving an improvement of 28 days over the real schedule. We refer to Figure 7 for illustration.



**Fig.7** Results from using WAS 0.1 on the Snorre PLT.

In figure 7, '**Before manual editing**' indicates the end of the initial schedule generated by WAS 0.1 and '**After manual editing**' indicates the end of the schedule after it has been manually edited by the expert in Saga. '**Real**' indicates when the 17 wells were actually completed.

As mentioned above, it must be emphasized that in this kind of “post mortem” schedule generation, uncertainties (e.g., in activity duration) are removed from the problem. It is therefore easier to make a tight schedule, as it is not necessary to make room for unexpected events. Some slack was probably introduced in the manually generated schedule as a proactive remedy for contingencies, in order to avoid major schedule disruptions. However, it was estimated by the Saga end user that 50% of the makespan improvement was real. A saving of 14 days of platform time implies a large cost reduction.

Note that the above savings resulted from manual schedule editing. No automatic optimization facilities were used. Based on the typical complexity and size of WAS problems we draw the conclusion that it is highly probable that the provision of automatic optimization facilities will result in substantial schedule quality improvements, as well as a sizable reduction in response time.

The prototype response time was deemed satisfactory by the Saga user, also in the context dynamic schedule revisions (reactive scheduling) and ‘what-if’ analyses. For predictive scheduling, a higher response time is tolerable. In case, the optimization algorithm ought to have an “anytime” characteristic, i.e., the optimization process may be stopped at any time and still produce a high quality schedule.

A typical manpower effort for weekly maintenance of well activity schedules is 5 man-hours. Through the introduction of a WAS system, this effort may, in our view, easily be reduced by 80%. However, WAS is not a monolithic problem. It involves exchange of information and negotiation between several organizational units with varying goals. In this context, we envisage potential savings from the implementation of a WAS system resulting from the presence of a common source of information, better consistency and security, and a vehicle for rapid “what-if” simulation capabilities. The latter type of improvements are of course dependent on a successful integration of a WAS system with relevant information systems, and a corresponding successful change of work-flow and organization.

## **FURTHER WORK**

Substantial improvement of the prototype are clearly needed in order to develop an operational WAS system. Identified requirements of improvement include:

- refinement of the WAS problem model
- better WAS scheduling strategies
- functions for schedule optimization

- simulation capabilities
- facilities for cost analyses
- user interface improvements
- development of interfaces to relevant information systems
- improved generality and re-configurability

The goal of an operational system is to provide a decision support that, when combined with human knowledge and expertise, will produce better schedules faster.

- **Problem model** regards an extension of the types of constraints and resources that may be modelled, and a richer model of each type of constraint.
- **Scheduling strategies.** It is necessary to develop better strategies, both for initial schedule generation, and dynamic schedule revision.
- **Optimization** facilities must be added. Relevant optimization criteria include rig utilization, makespan, level of water and gas production, and the utilization of processing capacity. The importance of different criteria should be user modifiable. Recent Iterative Improvement Techniques [4, 10, 12] with meta heuristics seem to be very well suited for overall improvement of well activity schedules. Complete backtrack search procedures [11] may be used locally in order to obtain a better schedule around critical 'anchor points', e.g., before and after a maintenance period.
- **Simulation.** At an operational level, it is desirable to evaluate several alternative courses of action. At a strategic and tactical level, it will be useful to the effects of adding/removing resources and/or adding/removing constraints. Examples are evaluation of the effect of adding an auxiliary rig, and sharing of resources between platforms.
- **Cost analysis.** Capabilities for short term and long term analyses are needed. The Well Activity Scheduler will contain information (e.g. production volumes, crew schedules) that will enable cost analyses that will be useful, e.g., in budgeting.
- **User interface.** It might be desirable to include functionality permitting the user to: Visualize and edit temporal activity precedence (e.g. showing precedence constraints as arrows that may be moved, added or deleted, possibly including a time interval indicating a temporal constraint on the interval between the activities. Warnings should be given when constraints are violated when a schedule is edited.) Temporarily overrule constraints when editing a schedule. Use default values for time estimates when creating new activities. Specify activities at several levels, thus giving an opportunity to specify activity hierarchies.

- **Integration** with other software systems is required in order obtain a well-working system with no redundancy and easy access of required data.
- **Generality and re-configurability.** Substantial benefits, and a wider potential for system exploitation and dissemination, may be achieved through increased flexibility of the problem model, as well as enhanced generality and rapid re-configurability. Improved flexibility will result from the specification of alternative well sequences and alternative slot allocations.

## References

1. Bouzoubaa, M., B. Neveu and G. Hasle (1995), Latif: A Solver for Constraints in a Hierarchical System. In *Proceedings CP'95 - First International Conference of Principles and Practice of Constraint Programming - Workshop on Over-Constrained Problems*, Cassis, France.
2. Bouzoubaa, M., B. Neveu and G. Hasle (1996), Houria III: Solver for Hierarchical System, Planning of Lexicographic Weight Sum Better Graph for Equational Constraints. Accepted for *The Fifth INFORMS Computer Science Technical Section Conference on Computer Science and Operations Research*, Dallas, Texas.
3. Cras, J-Y. (1993), *A review of Industrial Constraint Solving Tools*. AI Intelligence, Oxford UK, ISBN 1 898804 00 1.
4. Dorn, J. (1995), "Iterative Improvement Methods for Knowledge-Based Scheduling". *AI Communications* **8**, 1, pp. 20-35.
5. French, S. (1982), *Sequencing and Scheduling - An introduction of the Mathematics of the Job-Shop*. Ellis Horwood, Chichester. ISBN 0-13-806365-6.
6. Hasle, G. (1995), *Information Technology in Manufacturing Management - from scientific shoveling to focal-point opportunistic scheduling*. PhD Thesis, Research Report 205, Department of Informatics, University of Oslo, ISBN 82-7368-116-5.
7. Hasle, G., R. Haut, B. S. Johansen and T. S. Ølberg, Well Activity Scheduling - An Application of Constraint Reasoning. To appear in *Artificial Intelligence in the Petroleum Industry: Symbolic and Computational Applications II*, Editions Technip, 1996.
8. Hasle, G., G. Kelleher and J.E. Spragg (1995), Encoding the Pirelli Tyre Scheduling Problem as a CSP. In *Proceedings Intelligent Manufacturing Systems Workshop IJCAI'95*, Montreal, Canada.

9. Hasle, G. and S.F. Smith, (1995), Directing and Opportunistic Scheduler: An Empirical Investigation on Reactive Scenarios. Chapter 1 in *Artificial Intelligence in Reactive Scheduling*, (Edited by Kerr, R. and Szelke, E.). Chapman & Hall, ISBN 0412 72900-8.
10. Hasle, G., O.H. Wiig and J. Komorowski (1995), Reactive Scheduling Using Constraint Logic Programming - Investigating Constraint models of the Pirelli Tyre Scheduling Problem. In *Proceedings KBRS'95 - The Third IFIP WG5.7 Workshop on Knowledge-Based Reactive Scheduling*, Seattle, WA, USA.
11. Misund, G., B. S. Johansen, G. Hasle and J. Haukland, (1995), Integration of Geographical Information Technology and Constraint Reasoning - A Promising Approach to Forest Management. In *Proceedings ScanGIS'95*, Trondheim, Norway.
12. Nilsen, P.K. (1995), *Techniques for the Constraint Satisfaction Problem - A study on a tyre scheduling problem*. MSc. Thesis, Department of Informatics, University of Oslo.
13. Reeves, C. R. (Editor) (1993), *Modern Heuristic Techniques for Combinatorial Problems*. Blackwell Scientific Publications, London. ISBN 0-470-22079-1.
14. Tsang, E. (1993), *Foundations of Constraint Satisfaction*. Academic Press, London. ISBN 0-12-701610-4.