# Scheduling in the Real World: Lessons Learnt*

**Roman van der Krogt** and **James Little** and **Helmut Simonis**
Cork Constraint Computation Centre
Department of Computer Science, University College Cork, Cork, Ireland
{roman | j.little | h.simonis}@4c.ucc.ie

**Abstract**

Developing scheduling systems for industry, whether these are prototype systems to explore whether a certain technique can be applied in a certain new area, or production systems that will support the day-to-day running of a factory, is quite different from developing research systems aimed at the vast library of benchmark problems. In particular, we have found that real-world problems do not come with a clear set of requirements such as the benchmark ones and seldom match an existing research problem due to additional constraints on the solution posed by the particular production environment.

This paper will present some lessons (issues and recommendations) we learnt from our interaction with industry in developing scheduling systems, hopefully providing some insight into the peculiarities of real-world scheduling for people about to apply scheduling (or planning, or any other AI or OR technique) in practice.

## Introduction

Scheduling has been an active area of research for several decades. Much of that work has been directly applied in practice, or has at least helped in developing real-world applications, see e.g. (Bellone, Chamard, and Fischler 1995; LePape 1995; Bixby, Burda, and Miller 2006). The success of scheduling applications developed by companies such as Ilog, Cosytec and IBM is further proof of this. However, many people in academia have had no interaction with industry yet. A large number of them will eventually, and it is for the benefit of those people that, in this paper, we want to share some of our experiences in developing scheduling applications for industry.[1]

There are many reasons why one may seek interaction with industry. First of all, industry interaction allows one to validate a piece of research in practice. Something that works well in a lab environment may or may not work in practice, and the ultimate test is one involving real-world data. Industry interaction may also open up new research agendas: there are a lot of interesting open problems out there that may inspire new lines of research. Finally, certain funding bodies are more likely to fund project proposals that have industry involvement as this shows that the problem being attacked has relevance. Thus, engaging with industry has all sorts of benefits.

Developing scheduling systems for industry is, however, quite different from developing a purely research-driven system. One of the first major obstacles is the fact that whereas the research community has a well established meaning for what a "scheduling problem" constitutes, this meaning is often far from what the schedulers in a factory mean when they talk about their "scheduling problem." Whereas the researchers are speaking about sequencing issues, the schedulers are conferring about day-to-day challenges. This is perhaps best illustrated by the field studies of (McKay, Safayeni, and Buzacott 1995; Wiers 1997). In these studies, schedulers were found not take the current situation for granted; instead, they endeavoured to influence the amount and allocation of short-term capacity, the immediate shipping patterns, and the technical characteristics of machines (such as a machine's tooling and fixtures). The scheduler employed a large number of heuristics to anticipate possible problems and take precautionary measures. Hence, the scheduler's task and role turned out to be a problem anticipator and solver instead of a simple sequencer or dispatcher.

A direct result of this difficulty is that real-world scheduling problems seldom match the problems studied in the re-

*The authors gratefully acknowledge support from the Science Foundation Ireland under Grant numbers 05/IN/I886 and 08/RFP/CMS1711 and from Enterprise Ireland under Grant number CFTD/06/209.


[1] Helmut Simonis has been working on constraint programming for over 20 years, much of which was spent on applying the technology in practice. Helmut was member of the CHIP project at ECRC in Munich, co-founder and technical director for COSYTEC in Orsay, and a principal research fellow at Imperial College London, as well as working for start-up companies Parc Technologies and CrossCore before joining the Cork Constraint Computation Centre (4C).
James Little is 4C's external liason officer. His role is to identify research opportunities with industry, support grant applications, manage the resulting projects, and to ensure that IP is identified, protected and exploited. Before joining 4C, James has worked with Ilog and was technical director at AI International and was involved in several EU projects at Burnel University.
Finally, Roman van der Krogt has worked on applied scheduling projects with Bausch & Lomb, Alcatel-Lucent, Intel and a number of SMEs over the past four years, all while at 4C.

search community. Usually, there are all sorts of "artificial" constraints that a factory wants to meet because of commitments to clients, the particular production environment or regulations. Moreover, where the benchmark problems are well defined with precise optimisation criteria, the real-world problems are fuzzy, requiring extensive discussions between the user and the researcher to tease out the exact problem, and have multiple optimisation criteria, which are often contradictory, and with only very subjective weights for the different performance indicators. We believe that a Constraint Programming (CP) approach is particularly well suited to handle such issues, as new constraints can be added without affecting the existing ones. All of the systems that we mention in this paper have been built on CP technology.

In this paper, we present some of the lessons that we have learnt over the years. They will be a mix of pitfalls to avoid, and recommendations for things that do work out well. Some of these will be fairly obvious, others might not be. We will illustrate the points we make by describing situations that we encountered in practice. Since the emphasis of this paper is on the lessons, rather than the individual systems or projects we drew the lessons from, we will not describe those systems in detail, but we do include references to other papers that fully describe the systems.

## Scheduling≠Scheduling, or Nobody Runs a Job Shop

The scheduling research is driven by a set of standard problems. These problems include *single machine scheduling*, and *scheduling with parallel machines*, as well as the different *"$x$ shop"* problems such as *job shop* and *open shop*. Over the years, a number of common variants have been established: pre-emptive scheduling; scheduling with release times and/or deadlines; and scheduling with sequence dependent setup times, to name a few. Overviews of these standard problems and their variants can be found in the books by Pinedo (2002) and Leung (2004).

Unfortunately, industrial problems seldom fully match the research problems. Usually, there are all sorts of practical constraints that a factory wants to meet for reasons that are almost irrelevant to the scheduling community (preferences due to perceived reliability of workers could be one, for example). This fact was already acknowledged in one of the first books on scheduling:

> *The solution is to extract a problem that ignores the possibility of such changes and considers only the questions of sequence. Such a problem is unrealistic in the sense that it does not exactly represent any individual real situation, but rather gains in generality, since it approximates many situations. The results obtained from the study of this abstract idealised model do not represent a solution to any real sequencing problem; they represent information that should be available along with judgement and data on other aspects of the real problem. — Conway, Maxwell, and Miller (1967)*

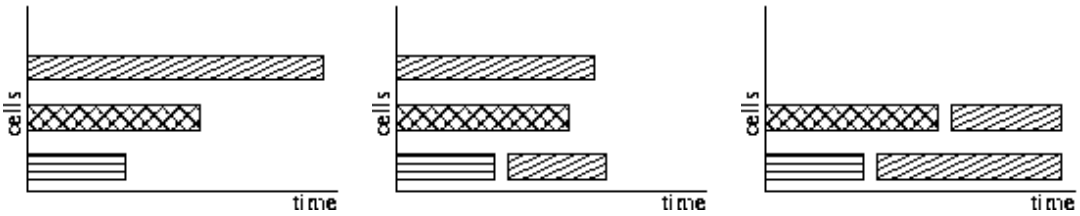
After more than 40 years, the situation remains much the same. This means that when one undertakes a scheduling project in the real world, most of the available research has no immediate value. Even if the particular environment is similar to one of the standard settings (say, a job shop), the tiny differences will likely violate the strict assumptions that the latest-and-greatest job shop algorithm makes. Thus, one is usually forced to come up with new algorithms or heuristics to solve the problem efficiently.

One particular example in this category is in the manufacturing of contact lenses (van der Krogt and Little 2006). This is a two-step process, where each order (the production of a batch of lenses of a particular type) consists of an activity to produce the moulds for the lenses, followed by the casting of the actual lenses using the moulds. This could be conceived as a job shop problem. However, as the moulds have to stabilise first and then degrade over time, there is a specific time window that has to be observed between the first and the second activities. The standard job shop algorithms can give no guarantees of this nature, and so we had to come up with a specialised search algorithm, inspired by the existing ones.

## Companies Don't Know What They Want to Optimise

Another big difference between theory and practice is the fact that in theory there is a well defined optimisation criterion, which is usually a single aspect of the problem such as makespan. In the real world, however, there is no such thing. Usually, the users want to optimise a variety of aspects, i.e. a multi-criterion optimisation problem, and often some of the criteria will be contradictory (e.g. a wish to minimise makespan and the number of machines used). Thus, we require a way to weigh the different aspects against each other: is the improvement in makespan large enough to warrant the use of an additional machine? Unfortunately, the user will not be able to tell you how to score the different criteria. Also, the criteria that are identified by the customer might not be the actual criteria. It may be too complicated for a scheduler to optimise a certain performance indicator $P$ directly. However, they may know from experience that optimising for another aspect $Q$ leads to a reasonable performance on the primary indicator. When describing the process, they may then tell you that $Q$ is an important factor, forgetting that the reason for that is actually $P$, which for an automated system may be equally hard to optimise.

What has helped us in some of our projects, is to show the user a number of small schedules and ask which one they prefer. For example, in one of our projects (van der Krogt et al. 2007), the user wanted to a tool to support the scheduling of teams producing mobile network equipment. For a variety of reasons, including efficiency and quality control, the factory wanted to move from manufacturing using a traditional production line to a "cellular manufacturing" setup. In cellular manufacturing (Irani 1999), separate teams produce the different products or product groups that a factory outputs. The schedule tool would have to decide how many cells to run on a given day, and which product group or groups each of the cells would handle. Ideally, each cell would be able to focus on a single or just a few product ranges for a given shift, in order to achieve the maximum benefit.




(a) Schedule A (b) Schedule B (c) Schedule C

Figure 1: Different schedules for different optimisation criteria

Clearly, there exists a trade-off between running multiple types of products through the same cell, and scheduling additional teams in order to lower the number of different product groups on each cell. For example, consider the three schedules in Figure 1 for the production of products from three different product groups, in different quantities (as expressed by the length of each block). These schedules represent different choices that can be made with respect to this trade-off. In schedule A, we use three cells, each producing the products from a single group. Notice how this is an unbalanced schedule, in that some cells finish before others. A much more level schedule may be obtained by moving some of the products between the cells (schedule B). However, now we have one cell that has to handle two product groups during this schedule. We may even obtain the same makespan as in schedule A, but using one fewer cell, if we accept that both cells now handle two products, as depicted in schedule C. In this case, the company decided that during the first few months after the transition to the new setup, they wanted schedules of type A to allow all teams to get used to the new situation, but move to schedules of type C once people were accustomed to the new situation.

Another issue that one may encounter is the following. Very often, different people in the organisation have very different objectives they want optimised. Depending on who is running the project, one group may grab the definition of the objective and force their point of view, or better, people realise that they have different objectives and use the tool to come up with a common understanding of what is possible and by which they can reach a compromise position. This also means that improving a single cost function by a few percent is basically meaningless in practice, so that much of the OR benchmarking is not justified in the "real world."

A good illustration of this issue is given by the ATLAS system (Simonis 2001), where the scheduling tool became an important part of the daily production meeting. It allowed people to present different scenarios and obtain an overall compromise. A tool to manage multiple planning scenarios (as opposed to the unique, actual production schedule) is thus very useful, as is a tool to show differences between two solutions (using a differential Gantt chart). In order for the different groups to accept a solution, the different KPI measurements for each schedule should also be presented.

In other cases it is much harder to get the stakeholders to cooperate to find a solution, e.g. if they are on different sides of the gap between supply and demand. We encountered an example of this when investigating a new materials management system for a hospital, aimed at minimising the number of deliveries to hospital wards, while guaranteeing agreed upon service levels (Little and Coughlan 2008). This project was initiated by the materials management group, who were forced to take a critical look at their operations when their main depot moved off-site. However, to be accepted by the wards throughout the hospital, the views of the different stakeholders had to be incorporated into the model: The materials management group are looking to provide service level agreements as cost-effectively as possible. The medical staff are looking for as high a service level as possible across all their products and the confidence of not running out. Finally, the department managers require visibility of the cost for the stock being held and distribution. Since the project was initiated by the materials management group, it was hard to convince the other stakeholders that their objectives were taken into account as well in the model. In general, it can be quite hard not to get involved in issues between different departments, especially if the funding situation seems to favour one group, or when there is suspicion about the project and a particular group has not has prior buy-in.

## Prototypes Are Worth Their Weight in Gold

When undertaking a research project at 4C, one of the initial steps we take is to produce a prototype model that captures the basic problem. The main reason for this is knowledge acquisition. One the one hand, this forces us to formally write down what we have understood to be the problem, exposing aspects that we need to get more information on. By describing to the user how the problem is modelled, discussing which aspects are taken into account and which are left out, it can be confirmed that the problem is fully understood. Moreover, it allows the user to confirm that they have explained all details of a problem. It is quite common for the user to point out errors with the schedules produced by an early prototype, that are the result of them simply forgetting to mention a particular constraint. Also, it is often very enlightening to compare the schedules produced by the tool with the schedules that were used before. As discussed in more detail in the section below titled "A scheduling tool is no silver bullet", the current schedules may not be satisfying all constraints. Hence, they may give a good indication as to which constraints are hard, and which are actually

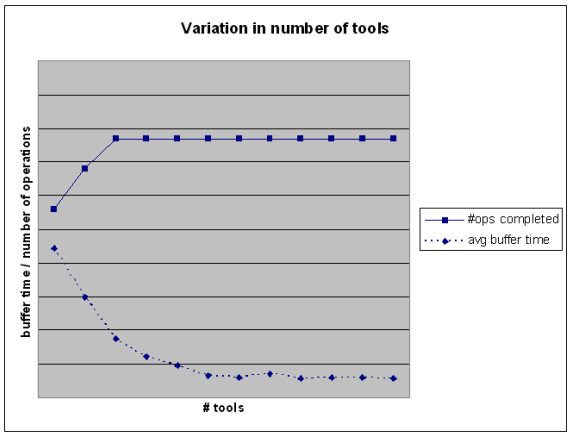



Figure 2: Changing the number of tools and its effect on throughput and buffer waiting times. The throughput is capped at the number of operations included in the data set. (Axes are without scale at request of the company.)

(strong) preferences. Besides knowledge acquisition, prototypes also serve as a demonstration of what the outcome of a project could be, provoking discussions on what the user might want the final tool to be.

For example, in the contact lense problem that we described earlier, the initial problem statement was focused around reducing the amount of buffer stock that has to be kept between the two processes. However, in discussing the issues we identified in the initial model, it turned out that they were interested in a much broader range of potential changes to the factory: additional machines, changing their rates of production, as well as the effect of changes in their demands. By incorporating these variables into the model, it made the final product much more useful to them.

This rapid exploration of what-if scenarios is an additional important benefit that automated systems have over manual scheduling. Demonstrating this ability to the user makes them more aware of the possibilities of a tool. The manual scheduler does not have the time available to look into such what-ifs, as (s)he is busy running the factory. Once a model is made, however, it is quite simple to change some parameters, such as the introduction of additional work or machines. For example, when we did a prototype model of implant scheduling at Intel (van der Krogt 2007), we showed the user the graph of Figure 2. It shows (for a particular data set) the effect on buffer waiting times and throughput (i.e. the number of completed operations) of adding additional tools, or removing some. This type of information, obtained through multiple what-ifs, showed for the first time new important patterns to the production manager.

## Pictures Are Worth a Thousand Spreadsheet Cells

From our experience, people very often shy away from graphical tools, only in order to reduce the development cost. Although there may indeed be a considerable cost involved in incorporating an interactive user interface, we believe that the benefits of such an interface far outweigh the costs. In our opinion, an interactive Gantt chart is highly valuable for the user in evaluating a schedule, or exploring alternatives. To allow this, some form capability is needed to freeze/unfreeze, either globally or per machine, and to fix schedule parts that must not be changed, or should be rescheduled. The interaction helps the user understand the produced schedules better, and allows adjustments to be made to the schedules if necessary. It also improves the confidence the user has in the tool.

As a particular example, during the development of the MOSES feed mill application (Simonis 2007), some visualisation was introduced to help understand the solver behaviour. This feature was not in the original specification, but its presence improved the quality of interaction during the development. When the end-user saw it running one day, they really liked the tool, as it helped them understand how a schedule is produced, and seeing how the strategy builds the schedule task by task. It was later included in the production version at the strong request of the company.

A good interface may bring more benefits to the company than simply a deep understanding of the scheduling system's behaviour. When the customer uses the tool to schedule services for a third person, a good interface can be used to sell their services. We carried out a project with a company that supplies solutions to the timber harvesting industry (Nugent et al. 2008). This company develops advanced simulation software that lets forest owners and buyers get more accurate measurements of the type of logs that a certain forest can provide. By having such detailed information for a number of forests, and the demand by sawmills in the region, we developed a prototype system that matches supply and demand in the most efficient manner. Figure 3 shows a visual front end based on Google Maps technology that we developed to present the solutions. The company used this front-end actively in demonstrating to potential customers what benefits the ability to accurately scan the forests could bring.

Finally, the structure of the factory, (machines, conveyors, bins, storage) may or may not change rapidly depending on the industry type and the overall level of capital expenditure. In any case, a simple graphical tool to "draw" the factory is much preferable to a purely database driven description of the environment. A simple tool may suffice for this purpose. As an example, consider the interface to the contact lens system, as shown in Figure 4. The underlying model is parametrised for the number of machines of the different types, their capacities, etc. A simple front-end in Visio allows the user to drag-and-drop new machines (or remove them by dragging them off the screen), and set their parameters. When the user gives the command to run the scheduler, a script iterates through the machines and sets up the model with the right parameters. Deploying a scheduling application through a graphical tool commonly found on the manager's desk supports its acceptance, while at the same time not investing in developing a new interface from the basics. In all of this, the application needs to be easily integrated within whatever system the user wants, as is discussed in more detail in the next section.

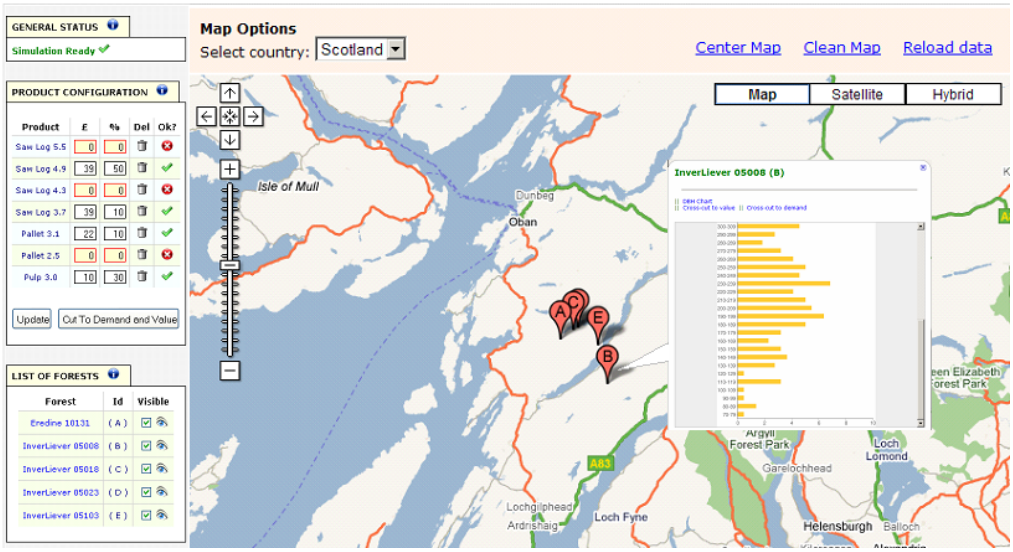



Figure 3: Graphical interface to the timber harvesting scheduling system based on Google Maps

## Integration Works Both Ways

When the outcome of a project is a production system, it is important that the scheduling tool is integrated with the order management/fulfilment system (such as an ERP or MRP package). This prevents the double entry of data, which is not only additional work, but may also introduce errors. The interface between the scheduling system and the order management system should be simple and open, and well documented. Quite often the commercial system will be replaced (because the company is sold, or merged with another division) before the production scheduling system. In that case having a clearly defined, documented interface helps a lot to keep the scheduling system alive.

One of the issues that may arise here, is the correctness of the data in the order management system. When human schedulers extract the current state of the factory and the work that is to be scheduled from the order management system, they will recognise incorrect or missing data, and use their knowledge to make up for this. The scheduling system cannot do this, however, and some means of validating the input is needed. It may also require the people entering the data into the management system to be more accurate. For example, one of the problems we faced with the cellular manufacturing scheduling system was the inaccuracy of the work-in-progress (WIP) that was recorded in the work floor management system. People would often forget to scan the piece of equipment they were working on, so that it would seem that their station was available, and that the product was sitting in a buffer waiting to be processed. Improving the accuracy of the WIP data proved to be a key issue in the success of the tool.

One should also recognise the fact that very often, people have built their own set of tools around the "official" system. Often, these are macros built in a spreadsheet application that help the user perform common tasks on some of the data. Providing these users with a way to keep using their existing tools will make sure not to alienate them from the project.

## A Scheduling Tool Is No Silver Bullet

Another important point that we want to make is that when a tool is intended to replace a manual schedule, the results are rarely much better than those obtained by the experienced human scheduler. This somewhat disappointing conclusion is derived from the fact that the production process has been tailored to suit the capability of the human schedulers. A good scheduling system to support the scheduler, however, does make him or her much more efficient, producing schedules in less time. This frees up some time that the scheduler can use for other purposes. Also, the automated tool is much better than the schedule made by the boss of the human scheduler, in case he is on holidays or sick leave. Hence, a scheduling system can make a company less dependent upon the knowledge of a single expert. This is especially true in SMEs where duplication of resources is difficult to justify financially.

A scheduling tool may also be very helpful when a major reorganisation of the factory invalidates the built up knowledge of the experts. In the cellular manufacturing case we discussed in an earlier section, a new way of working meant that the schedulers had no experience of how best to organise the work in the new working practices. Since the order in which products were started is so much more important in the new situation (a wrong order may starve certain cells, while others are faced with a buffer full of products waiting to be processed). By having a scheduling tool to support them, they were greatly helped in the transition to the new way of working.

Another reason why human schedules are often better than automated ones, is that humans may take some liberty when it comes to satisfying the constraints. In some cases, the fact that the automated schedule is *guaranteed* to satisfy

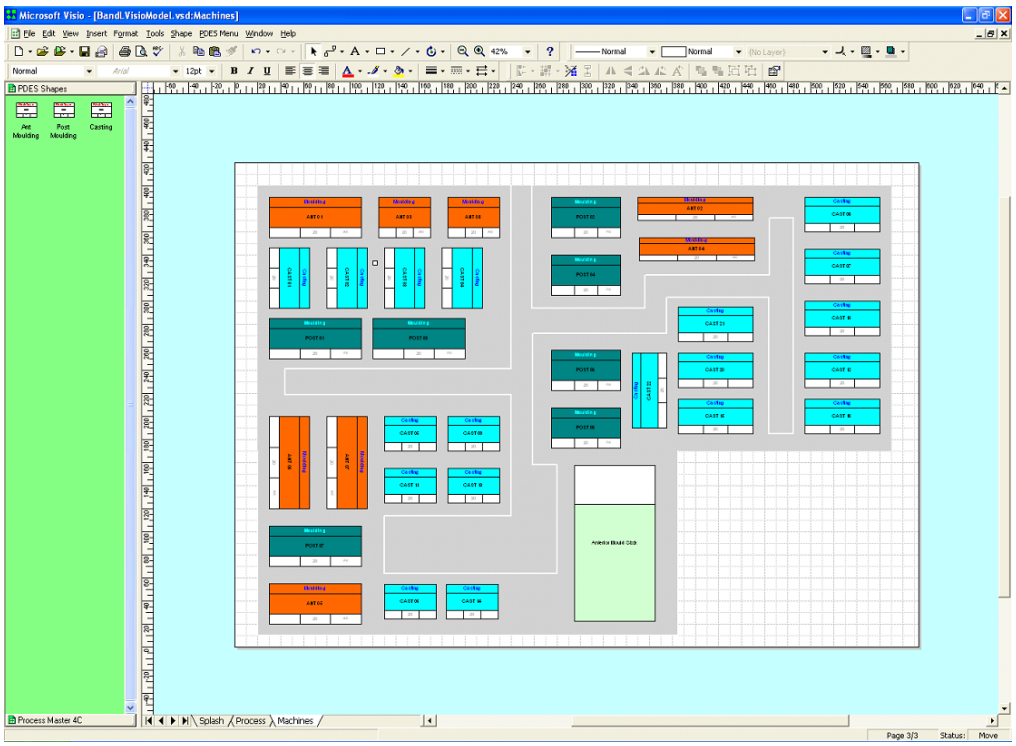



Figure 4: Graphical description of a fictitious production facility

all constraints is an important benefit in itself. For example, the TACT system (Simonis, Charlier, and Kay 2000) is used to schedule the transportation of poultry from farms to food processors. The human schedulers would often overload the trucks in order to obtain a more efficient schedule. If inspected, such trucks are liable to heavy fines both from an animal welfare perspective as well as from a road safety perspective. The schedules produced by the TACT system, on the other hand, will satisfy the constraints, preventing such fines. It is important to note, however, that one sometimes has to break some constraints (e.g. when assigning overtime) in order to find (efficient) solutions. To allow for this, the TACT systems allows the scheduler to turn off certain constraints to see what the impact is of relaxing those.

## Technology Transfer Are the Hardest Words

Often, projects with industry will result in an acceptable prototype or demonstration system. However, it may be the case that the outcome of a project is envisioned to be a tool that needs to be deployed, or the prototype may prove so successful that the company wants to deploy it. One of the biggest challenges in such cases, is to successfully transfer the technology to the company. The scheduling tool is an advanced piece of software, which may need maintenance to reflect changing work practices, product ranges, equipment, etc. This requires specialised knowledge of the model and the scheduling strategy, that the company may not have in-house. Whereas companies can provide support contracts for such cases, this is difficult to provide in a university context. How to deal with this situation will vary from project to project, but our advice would be to start thinking about how to deal with the issue early.

Successful projects have led to start-ups or spin-offs. A final point we want to make here, is that there exists a flawed perception within industry that small start-ups are unlikely to be able to maintain a piece of software over a longer period, whereas a large company can. The reality is often quite different, however. There is a variety of reasons for this. Firstly, a large company can more easily decide to leave a market, focusing on more profitable aspects of its business. A small company is usually only active in one field, however, and cannot make this switch. Secondly, unless a support contract brings in new revenue in terms of new projects, such a contract may be conceived as not worth the effort for a larger company. With many sources of revenue, a relatively small support contract is not that important. It is however, for a smaller company. Hence, the quality of support is likely to be better. Finally, there is of course the fear that a small company may be unsuccessful and go out of business altogether. However, if it develops a successful scheduling system that the customer still wants supported in 5 or 10 years time, they have clearly demonstrated their competency and will be viable. Unfortunately, from our experience in small start-ups, we know that it may be hard to convince people of this fact.

## Conclusions

In this paper, we listed a number of issues that one encounters when developing scheduling systems for industry, be it

prototype or production systems. We also proposed a number of recommendations. These issues and recommendations can be summarised as follows:

- Real world scheduling problems pose different problems than the research benchmarks, due to some artificial constraints on the solution posed by the particular production environment. Thus, custom algorithms are required, but these can often be derived from existing research.
- Constraint Programming provides a good framework for building real world scheduling systems.
- Real world scheduling problems are fuzzy. The details should be carefully elicited from the user. A prototype system may help to formally describe the problem and discover discrepancies and areas that are still not adequately described.
- Real world scheduling problems usually involve multiple optimisation criteria, with the user not able to tell you the relative weights of these criteria. They will be able to tell which of a number of schedules they prefer, however, which can be used to elicit this information.
- Real world scheduling problems may involve multiple stakeholders. It is important to discuss with the different stakeholders and represent their views in the model. The weights of these views can be determined by the stakeholders forming a common understanding of the problem through running the model in different scenarios. Only then will the project be deemed a success.
- A reasonably sophisticated user interfaces makes any scheduling tool much more useful to the user. Being able to interact with the tool, exploring different alternatives, will help the user in understanding the schedules that the system proposes, even why that schedule is suggested and in turn increase the confidence in the system.
- Integration with the existing order management software is imperative. We cannot expect the scheduling system to impose additional, disproportionate overheads on the company's IT structure. The acceptance of a system is often how little it disturbs the current setup.
- The benefits of a scheduling system do not necessarily come solely from the fact that more efficient schedules are derived. The guarantee that all constraints are satisfied may be highly valued in itself and act to support the manual scheduler in better decision making.
- Technology transfer is an important issue, that should tackled at an early stage. It is hard for a research group to provide adequate support into the future. If the scheduling application is important enough, the company must make provision to support it throughout the project timescale – although this may mean the researcher spending time with the company.

Despite the difficulties that we have listed, developing scheduling systems for industry is both challenging and rewarding. The difficult problems are rarely technical – they are social, managerial, but which the researcher should become aware of. Knowing that a factory employs a system that you helped design is a very satisfactory notion. We sincerely hope that this paper encourages people to apply their research in practice, and helps them avoid some of the pitfalls we mentioned.

## References


Bellone, J.; Chamard, A.; and Fischler, A. 1995. Constraint logic programming decision support systems for planning and scheduling aircraft manufacturing at Dassault Aviation. In *Proceedings of the Third International Conference on the Practical Applications of Prolog*, 111–113.

Bixby, R.; Burda, R.; and Miller, D. 2006. Short-interval detailed production scheduling in 300mm semiconductor manufacturing using mixed integer and constraint programming. In *The 17th Annual SEMI/IEEE Advanced Semiconductor Manufacturing Conference (ASMC-2006)*, 148– 154.

Conway, R.; Maxwell, W.; and Miller, L. 1967. *Theory of Scheduling*. Reading, MA: Addison-Wesley.

Irani, S. 1999. *Handbook of Cellular Manufacturing Systems*. New York, NY: John Wiley & Sons.

LePape, C. 1995. An application of constraint programming to a specific production scheduling problem. *Belgian Journal of Operations Research, Statistics and Computer Science*.

Leung, J. Y.-T. 2004. *Handbook of Scheduling: Algorithms, Models and Performance Analysis*. Boca Raton, FL: CRC Press.

Little, J., and Coughlan, B. 2008. Optimal inventory policy within hospital space constraints. *Health Care Management Science* 11(2):177–183.

McKay, K.; Safayeni, E.; and Buzacott, J. 1995. Common sense realities of planning and scheduling in printed circuit board production. *International Journal of Production Research* 33(6):1587–1603.

Nugent, C.; Curran, D.; Prestwich, S.; and Little, J. 2008. A hybrid evolutionary approach to forest management. In *Proceedings of the 19th Irish Conference on Artificial Intelligence and Cognitive Science*.

Pinedo, M. 2002. *Scheduling: Theory, Algorithms, and Systems*. New Jersey: Prentice-Hall.

Simonis, H.; Charlier, P.; and Kay, P. 2000. Constraint handling in an integrated transportation problem. *IEEE Intelligent Systems* 15(1):26–32.

Simonis, H. 2001. Building industrial applications with constraint programming. In *Constraints in Computational Logics*. 271–309.

Simonis, H. 2007. Models for global constraint applications. *Constraints* 12(1):63–92.

van der Krogt, R., and Little, J. 2006. The PDES workbench. In *Proceedings of the Thirteenth International Conference on Concurrent Engineering: Research and Applications*, 619–626.

van der Krogt, R.; Little, J.; Pulliam, K.; Hanhilammi, S.; and Jin, Y. 2007. Scheduling for cellular manufacturing. In *Proceedings of the Thirteenth International Conference on Principles and Practice of Constraint Programming (CP-07)*, 105–117.

van der Krogt, R. 2007. Scheduling implant operations using constraint-based scheduling (abstract). In *Online Proceedings of the Third Workshop on Simulation in Manufacturing, Services and Logistics*.

Wiers, V. 1997. *Human-Computer Interaction in Production Scheduling: Analysis and Design of Decision Support Systems for Production Scheduling Tasks*. Ph.D. Dissertation, Eindhoven University of Technology, Eindhoven, The Netherlands.