ELSEVIER

Knowledge-Based Systems 10 (1997) 169–175

Knowledge-Based SYSTEMS

PIPES: A heuristic search model for pipeline schedule generation[1]

M. Sasikumar*, P. Ravi Prakash, Shailaja M. Patil, S. Ramani

National Centre for Software Technology, Gulmohar Cross Road 9, Juhu, Mumbai 400 049, India

Received 15 May 1997; accepted 29 May 1997

Abstract

Resource scheduling problems are increasingly being solved using AI techniques. Solving real-life versions of these problems demands the ability to model a range of constraints and, at the same time, to be flexible enough to make revisions to these constraints. In this paper, we outline the pipeline schedule generation problem, where the task is to generate a pumping schedule for a single-source multiple-destinations oil pipeline carrying multiple products. The schedule must take into account product availability and requirements while satisfying a wide variety of domain constraints, including tankage constraints, product sequencing constraints, quality control constraints, delivery constraints, etc. We describe an approach based on heuristic search, which we have adopted for solving it. The system has been successfully implemented and is in use. © 1997 Elsevier Science B.V.

Keywords: Heuristic search; Scheduling; Constraints

1. Introduction

Resource scheduling problems are being increasingly solved using AI techniques [1,2,11]. One of the earliest milestones in this area is the ISIS system [3]. The focus of this system was job shop scheduling. Another important class is distribution planning, where suitable media such as tankers, road vehicles, pipelines, etc. have to be used to plan the distribution of items to locations. In this paper, we describe such a distribution planning problem based on pipelines and discuss our approach for solving this problem.

Pipelines provide a convenient mechanism for transporting oil from one place to another. Though the capital investment is higher, the operating costs are much lower compared with other modes such as rail, road, etc. The losses are lower and reliability higher. Hence for locations to which a steady input of oil is required, pipelines provide an attractive medium of transport. And for the same reason, the oil industry would like to use this medium to the maximum efficiency possible. The proposed system was formulated in this context to improve the utilisation of pipelines for distribution of petroleum products.

Pipelines come in a range of complexity. The simplest would be one with one source location, one destination, and one type of product to be pumped. Planning the schedule of pumping for such lines is a trivial matter. There are such pipelines being used for crude oil movement from coastal ports to inland refineries. At the next level of complexity, we can have multiple destinations. A more realistic pipeline would also handle multiple types of oil. This paper addresses the scheduling problem for this class of pipelines. Still more complex pipelines will also cater for multiple source locations. Our model does not currently handle this type. Moreover, such pipelines are relatively few.

Note that, unlike in the case of vehicles, there is no physical separation between different items as they move in a pipeline. Therefore, mixing and consequent contamination of some part of the items is inevitable. One consequence of this is that pumping small quantities of products is not economical. This results in significant constraints on the minimum amount of each of the items that can be pumped. The possible sequences in which different items can be pumped into the pipeline are also restricted.

Apart from the constraints, the pumping must, obviously, take into account product availability at the source (typically this is a refinery) and the consumption of different products at the destinations. The storage available at the source and destination is limited. Another source of complexity is the time scale involved. Since the pipelines, in general, cover hundreds of kilometers in length, an item that is pumped may take up to 5 days to reach the destination. There are restrictions, derived from experience, on the amount of any item that can be pumped. This limits the

[1] Revised of paper from K.S.R. Anjaneyulu, M. Sasikumar and S. Ramani (Eds.). Knowledge-based Computer Systems: Research and Applications. © Narosa Publishing, 1997.

* E-mail: sasi@saathi.ncst.ernet.in

PII S0950-7051(97)00026-9

size of a parcel, by imposing a lower bound as well as an upper bound. Considering such size restrictions versus the pumping rates possible, pumping of an item may take from a few hours to a few days. Hence it is essential to anticipate requirements sufficiently early and plan the pumping.

All these aspects make the planning of the pumping schedule for a pipeline a fairly complex activity, requiring a substantial amount of domain expertise. The requirement for our system was to consider all relevant parameters and generate a good monthly schedule, which is feasible. Characteristics of a good schedule are outlined in Section 2. Preparing such a schedule used to take a couple of days with a number of experts from the refineries, destination locations, and pipeline operations getting together and drawing up the schedule. Each person would be concerned with the implications, with respect to his domain of concern, of the resulting schedule and would demand changes, if he could not meet the schedule.

In the next section, we describe the scope and inputs of the system in detail. Section 3 will discuss our approach to the problem. Section 4 gives a summary of the implementation and the outputs generated. Section 5 concludes the paper.

2. The pipeline scheduling problem

The task of the system is to generate a ''good'' pumping schedule for a period of about one month from a given date, based on the given supply/demand scenario of the different locations as well as the constraints mentioned earlier. As we mentioned earlier, the pipeline model assumes a single source and one or more destinations. The system must provide for the pumping of multiple products.

A good schedule must ensure that:

- There is no containment at refineries (i.e., products must be pumped out of the refinery before the tanks for the product fill up). A containment can lead to the refinery cutting down production.
- There is no dryout at consumer locations (i.e., products must be made available at the locations before the tanks become empty). A shortage of product can affect the local industry.
- The pipeline is utilised to the maximum extent possible. Ideally, there should be no pipeline shutdowns. And as far as possible delivery to all locations must be maximised.
- Above all, the schedule must be implementable. Shortcuts (such as ignoring some constraints) which lead to infeasibility must be avoided. Note that by this specification, there may be more than one such good schedule for a given set of requirements. This was acceptable.

The environment in the oil industry changes very often and quite substantially. Hence, it is important to ensure that much of the domain information in the system can be modified easily. Another important aspect is that, as in an expert system, we are modelling a system to solve a problem which requires a substantial amount of domain knowledge and expertise. Thus we are to gather much of the knowledge from human scheduling experts; not surprisingly, one encounters knowledge-acquisition bottlenecks here. It turned out that we only had 40–50% of the major constraints relevant to the task (major from an implementation point of view) when the requirements were approved and work on the system started. Thus, it was important that the system be flexible enough to extend by addition or modification of constraints as the development progressed.

The major inputs (including parameters and the specific constraints involving them) to the system and their relevance are briefly outlined below.[2] We will take a specific pipeline, running from Mathura to Jalandhar, as an example.

- *Locations:* Mathura, Delhi, Ambala and Jalandhar. Mathura, having an oil refinery, is the source location; the others are all destinations. Overall the line covers a distance of about 500 kilometers.
- *Products:* This line is used to pump high speed diesel (HSD), kerosene oil (SKO), motor spirit (MS) and aviation turbine fuel (ATF). There are sequencing and batch size constraints on all these individually as well as on possible combinations. In particular, SKO is often used as a plug product to separate other products in line. Hence, there will be a small quantity of SKO in between every pair of other products in the line. A typical sequence of pumping would be: ...SKO, ATF, SKO, HSD, SKO, MS, SKO, HSD, SKO,... Because of this use as plug, SKO has to be available for pumping any product through the line. Hence, SKO is always pumped in small quantities. In certain months, when SKO availability is poor, running the line smoothly can be a difficult task. Even with the SKO plug, ATF is never pumped after an MS batch (i.e., the sequence ...MS, SKO, ATF,... is not acceptable). Batch size constraints dictate that quantities chosen for pumping cannot be arbitrary. For example, the size of an HSD batch[3] is always a multiple of 7. Depending on the products pumped before, the current product's batch size may be constrained. Before an ATF batch, the SKO plug must be at least 6; before and after MS, it will be at least 3; and for all other cases, it must be at least 2. If an MS batch meant for more than one location is scheduled and an HSD batch is taken up next, the HSD batch must be at least 45 in size. Many of these constraints may appear strange or arbitrary, but they are guidelines arising out of quality control requirements or thumb-rules obtained through years of experience. These constraints were by far the most fragile and hard to model in the system.
- *Refinery production at Mathura:* All products are not available uniformly throughout the month. In particular,

[2] Some of the figures quoted in the paper have been changed without affecting the issues or overall pattern, to retain confidentiality of the data.
[3] All quantities in this paper are in 1000 kl.

products such as ATF and MS are produced only during specific periods within a month. Assumption of uniform availability can severely affect the feasibility of the generated schedule. The model must be capable of working with varying availability. For example, ATF production could be at the rate of one unit per day from 1st to 6th of a month, then no production until 14th, then again one unit per day, etc.

- *Demands at destinations:* This is more or less uniform over a month. However, festival demands, holidays and the like can provide significant deviations. Again, it was expected that the system would provide for specification of such constraints, as and when they arise.
- *Other inputs/outputs:* Part of the production at Mathura is distributed via road and rail. The destination locations also function as secondary loading centres by road/rail to other areas. Products delivered to these locations by the pipeline are (partially) collected at these loading centres for transport by these means. Rail/road can also be an auxiliary source of input to some of these locations. For example, if pipeline pumping cannot meet the demand at a location, a rail parcel can be planned to supplement the pipeline input. Normally, the extent of these rail/road movements will be decided before the pipeline scheduling is done and hence will be available as an input to the system.

Table 1 shows the various components in the supply–demand specification at a specific location. Separate tables are used for different locations. The columns indicate the products and the rows the various relevant parameters. The information includes the storage capacity (indicated by gross and operating tankage), opening stock and the estimated closing stock, in addition to the various modes of input and output. This table is generated by a higher-level planning team and provides indication of the type and extent of oil movement required by various modes from various locations. For example, the figure indicates 1.4 units of MS being brought in by rail to the location and 8.9 units being taken out via road transport. In addition, there is a local demand of about 10 units. Almost all of the local and road requirement in this location is to be met by pipeline transfer of MS. The table shows that this is estimated to be about 19 units.

A schedule for this problem should decide the sequence of products to be pumped for the period specified, along with the quantity and how it should be distributed to the destination locations. For example:

Product	Quantity	Distribution		
		Delhi	Ambala	Jalandhar
SKO	2.0	0.6	0.6	0.8
HSD	45.0	10.0	15.0	20.0
SKO	6.0	1.5	1.5	3.0
ATF	7.0	7.0	0.0	0.0(full to Delhi)

3. Our approach

We used a knowledge-based heuristic search approach to solve this problem. The rationale for this is manifold.

The conventional approach to scheduling problems is using a numeric model [4]. However, there is a general feeling [5,6] that numeric models have many disadvantages for these types of problems. The most critical is the inability to tolerate changes in system specification. Constraints and evaluation functions may change from time to time. For numeric models, these may result in significant change at the model level itself. It is essential that the system provides adequate visibility of domain knowledge and allows flexibility in changing it. Therefore, a symbolic model was preferred. Another factor in favour of this was that, in such complex domains, feasibility of the resulting schedule is more important than its optimality. The former may be compromised, if constraints are ignored or simplified. A

Table 1
Supply–demand balance at a specific location, October 1994

	ATF	MS	SKO	HSD
Gross tankage	7.7	10.5	11.1	52.4
Effective operating tankage	6.2	8.4	8.9	41.9
Effective opening stock	6.0	4.8	2.9	17.4
Production	0.0	0.0	0.0	0.0
Rail (in)	5.3	1.4	1.4	65.0
Road (in)	0.0	0.0	0.0	0.0
Pipeline (in)	0.0	18.9	12.7	37.7
Total availability	11.3	25.1	17.0	120.1
Local consumption	0.0	10.2	7.3	48.2
BG rail (out)	0.0	0.0	0.0	0.0
Pipeline (out)	0.0	0.0	0.0	0.0
Road (out)	4.4	8.9	5.5	47.9
Total offtake	4.4	19.1	12.8	96.1
Effective closing stock	6.9	6.0	4.2	24.0

Pumping Sequences

S1: [sko,atf,sko,hsd], S2: [sko,ms], S3: [sko,hsd]

Pumping patterns

Pat-id	Mathura	Delhi	Ambala	Jalandhar	Pat-id	Mathura	Delhi	Ambala	Jalandhar
p1	-0.6	0.2	0.1	0.3	p2	-0.6	0.1	0.2	0.3
p3	-0.6	0.6	0.0	0.0	p4	-0.6	0.0	0.2	0.4
p5	-0.6	0.2	0.0	0.4	p6	-0.6	0.2	0.2	0.2

....

Fig. 1. Pumping sequences and pumping patterns.

third factor is the ability to track down causes of undesirable behaviour. Therefore, resource scheduling is increasingly being viewed as a challenging area for AI applications [3,1,7,8].

Though the problem can be viewed as a candidate for an expert system approach, it does not fit easily into a shallow model of reasoning followed in conventional expert system shells. An implementation of this problem using an expert system shell was attempted, but it had a very limited success. The reason was essentially that the simple forward chaining rule based implementation provided by the shell assumes that no deep searches need to be made. Decisions must be taken irrevocably by just examining the current state. We found that the ability to look ahead a few moves would be critical for this problem. Theoretical optimality requires an exhaustive search, but a combination of heuristics and suitable evaluation functions has been able to yield good feasible schedules with a look-ahead level of about 4. A look-ahead level of 4 was found to yield the best compromise between the quality of schedule and the resource (time and system memory) requirements.

We had earlier worked on a few other scheduling projects where we used a heuristic search model [9]. We have also explored the characteristics of these problems in general [6]. We obtained very good results using this approach for those problems and were confident of its applicability for this problem as well.

3.1. State space search

For this problem, a ‘‘move’’ was defined as choosing the next batch to pump. Scheduling was done in temporal order. Each move consisted of four components: the product to be pumped, the quantity to be pumped, how the product is to be distributed among the destinations, and the pumping sequence being followed.

At the generation time itself, minimal checking is done to ensure that the quantity fits the product batch-size constraint and the tankage constraints.

The pumping sequence is a guideline that human experts have formulated to ensure sequencing constraints. Some of these sequences are long and have evolved to represent typical operating patterns. We have pruned this, retaining only the product sequencing aspects. Using the manual sequence completely was found to interfere with the system’s look-ahead ability. The look-ahead assumes that there will be no serious impact of the current decision beyond the look-ahead level. When we fix a long sequence for pumping, we are commiting the products for the pumping of the next five to ten batches. Our look-ahead level is normally smaller than this. Therefore, we are violating the assumption mentioned above and end up taking decisions which we are unable to satisfy. In our system, there are three pumping sequences, marked S1, S2 and S3 in Fig. 1.

While, theoretically, an intermediate location can draw any quantity from zero to maximum pumping rate from the pipeline, for operational reasons intermediate deliveries are given in certain standard patterns. If a product is needed only at Delhi, the entire flow may be diverted to Delhi, shutting of the rest of the line. Similarly, if Delhi does not want a product, it may bypass and operate as if Delhi is not in the line. We formulated about 10 such possible patterns which are called pumping patterns, marked p1, p2, etc. in Fig. 1. Thus a move containing the above information defines the current batch completely.

Our state representation captures the scenario of all the locations and the pipeline, just before a new batch is pumped (see Fig. 2). This includes the projected inventory map for all products at all locations, the current content of the pipeline (called linefill), and the current time.

The root of the state-space is the situation at the start of the schedule period, with no moves planned. At each node, child nodes corresponding to different possible moves are generated.

3.2. Beam search

We use a fixed-width search (beam search) [10] to explore this search space. At each node, all the children are generated. The list is pruned by applying the domain constraints to them and the resulting nodes are evaluated using a heuristic function. A fixed number of these (called the beam size) are expanded further for about four levels in a similar manner. The branch which looks best at that depth is chosen as the next move. The beam size is also kept as a parameter that can be changed, if necessary. This was found to depend strongly on the search space model—for the

```
current_time - 12,
batch_number - 268,
current_pumping_sequence - [atf,sko,hsd],
cost_so_far - 1661.53,
linefill -[delhi-[t(268,sko,6.0,p1),t(267,ms,3.4,p1), t(266,sko,3,p1),
                 t(265,atf,5.0,p3)], ... jalandhar-[t(263,hsd,11.2,p1),
                 t(262,sko,1.4,p1)]],
transactions - [t(265,delhi,atf,2.0,7.66667,11.0,p3), .....
                  t(268,mathura,sko,-6.0,1,12,p1)],
pending_requirements - [pr(mathura,atf,2.0,45.2,45.2), .....
                            pr(jalandhar,hsd,3.7,108.1,108.1)],
graphs - [locgraph(delhi,atf,[0@7.5,1-100@ -0.3,1-100@ -2.0,1@2.0],..),
          locgraph(jalandhar,ms, ....)]
```

Fig. 2. State structure.

model described, we found a beam size of about 8–10 to yield good results.

3.3. Modelling the constraints

As the description so far indicates, there are a wide range of constraints that the system has to cater for in drawing up the schedule. Most of these are mandatory constraints. These were incorporated into a constraint-checking phase after the move generation. This removes moves which violate any of these constraints.

There are some constraints which are preferential—for example, provide products to all locations, minimise shutdowns, etc. A penalty function was designed for these and incorporated into the heuristic evaluation function, thereby allowing the system to pick moves violating these constraints, if there are no better options available.

Some of the constraints were hard to model directly. An example is the requirement of SKO being available for use as a plug. The difficulty here is that because of this requirement, SKO decisions have an impact much beyond the look-ahead level. For example, a shutdown of the line may be recommended towards the end of the month, because adequate plug product is not available. It is possible that this could be avoided by choosing a smaller SKO batch size much earlier in the month.

We implemented this requirement by using a minimum stock level concept, called the plug-limit. Violation of the plug-limit for such products are also penalised in the evaluation function. This ensures that higher batch sizes for products such as SKO will be used only if necessary or if the stock levels are quite good.

3.4. Evaluation function

The evaluation function uses a weighted sum of a number of factors in assessing the schedule so far. These factors include:

1. the cost of the schedule so far (using the same function for each of the moves constituting the schedule);
2. the predicted stock level at all locations for all products as per the current schedule;
3. how much of the requirements of the locations have been satisfied;
4. penalty for any shutdown in the pipeline;
5. penalty for violating plug-limits (as described in the previous subsection).

4. Implementation and performance

The system was implemented on a 486 system running DOS. The program was developed using Quintus Prolog. Since the interface offered by Quintus on DOS was poor, we used Turbo Prolog to develop user-friendly interfaces for entry of supply–demand information, linefill, etc. Turbo Prolog programs were converted to stand-alone executables and invoked from Quintus Prolog. The actual scheduling process was done completely in Quintus Prolog.

Much of the data and constraints were captured as Prolog clauses, providing ease of modification. For the dynamic data, i.e., data which change very frequently (including supply–demand information, linefill, etc.), the Turbo Prolog interface hid this representation from the users, offering them a friendlier interface. For other data, it was agreed that only someone with some familiarity with Prolog would modify the files. Given the readable and symbolic style of clausal forms, this was not expected to be difficult.

The development of the system to handle the Mathura–Jalandhar pipeline took about one year. The first prototype was completed in about 6 months. The initial knowledge gathering was done by our attending their scheduling meetings, and interacting with the staff doing the scheduling and management of the pipeline. During the subsequent 6 months, the system was tested against real data by running the program with the input provided by the users. The outputs were analysed by the users and feedback, in terms of undesirable/illegal moves, was given to us. Often this would be traced to a data error or sometimes to a constraint/parameter that we were unaware of, which would then be incorporated into the system.

The system has been handed over to the users at their premises. A monthly schedule preparation takes about one hour. This enables 4–5 runs in a day, changing different constraints or parameters—for example, what if the rail input to Jalandhar is increased?

One of the advantages of automated systems is the amount of detailed reports they can generate, providing

MJPL PUMPING DELIVERY SCHEDULE (October 1994)

BATCH	DT	Mathura Quantity				DT	Delhi Quantity			DT	Jalandhar Quantity		
		MS	ATF	SKO	HSD		MS	ATF	SKO HSD		MS	ATF	SKO HSD
a-265(p3)						1		7.0					
s-266(p1)						1			0.9	6			1.4
m-267(p1)						2	1.1			6	1.5		
s-268(p1)	1			6.0		2			1.9	7			2.7
a-269(p3)	1		10.0			2		10.0					
s-270(p3)	2			5.0		3			5.0				
h-271(p5)	2				42.0	3			13.3	7			28.7
s-272(p4)	5			6.0						9			3.7
m-273(p1)	5	21.0				6	6.6			10	9.4		
PLAN (TKLs)		42.0	31.0	45.0	154.0		10.4	38.0	10.7 48.8		11.0	0.0	22.5 83.7
					[272.0]				[107.9]				[117.1]
SPM (TKLs)		67.9	60.9	72.0	259.2		11.5	60.9	20.9 96.2		30.1	0.0	34.7 117.4
					[460.0]				[189.5]				[182.1]
Stk 1(TKLs)		40.3	11.6	16.7	33.9		10.8	7.6	15.2 21.4		11.2	10.9	7.7 23.0
Stk 31(TKLs)		62.6	35.2	46.6	131.4		10.0	2.6	15.6 15.7		1.2	14.8	1.3 4.5

(Ambala dropped from the figure due to space constraints)

Fig. 3. Schedule.

different perspectives on the resulting scenario. The Pipes system's main output is the monthly schedule identifying the product, quantity and manner of distribution of the product at different locations. The listing also shows the date of pumping and date of receipt of the share at the different locations. A sample of this is shown in Fig. 3. The output also includes the total pumping product-wise and a comparison against the recommended quantities (see Section 2). Variations from the recommended figures are quite common, since this does not take into account many detailed parameters and constraints.

In addition to the pumping schedule, the system also produces a detailed description of the product availability (inventory) at different locations throughout the month, assuming the generated schedule is followed completely. Fig. 4 shows the graph for HSD at Delhi before and after the scheduled movements take place. This helps users to quickly identify shortcomings (shortage of products, containments, etc.).

5. Conclusions

In this paper, we have presented a model for applying heuristic search techniques for solving a type of resource scheduling problem. The system has been implemented

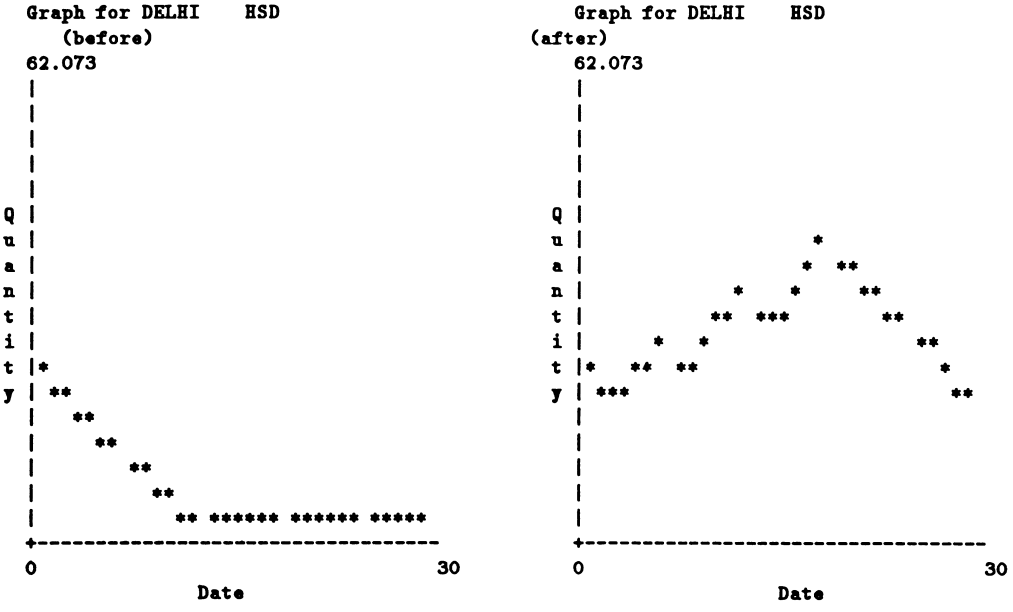

Fig. 4. Graphs of the inventory before and after the scheduled movements take place.

successfully and the users have found the schedules generated to be acceptable. Even during the development itself, we revised some of the model components significantly, so making changes has not been difficult in most cases.

We have kept most of the domain information (static as well as dynamic) as user visible data files. For the developers or, to a large extent, for the end-users to change these is not difficult.

The aim during development was to make the model as general as possible, in order to be applicable to a wide range of pipelines. Subject to the assumption of a single source, we feel the system can be easily modified to handle other pipelines also. Attempts are currently being made to relax this assumption.

One of the difficult issues in these kinds of systems is data validation. We have no way of doing any serious checking on the inputs or parameters as they are entered. On the other hand, most of the parameters are not random; they are decided and generated through some higher-level planning process. Therefore, in most cases, one expects to obtain a reasonable schedule, if the data is correctly entered. Small mistakes in data can cause the system to go completely out of gear. Once, during the testing phase, the system generated massive shutdowns in the schedule. This was quickly traced to a mistake in the entry of the HSD production figure at Mathura. Since HSD formed the primary component, volume-wise, in the pipeline pumping, lack of HSD naturally resulted in shutdowns. When the correct figure was entered, this problem was resolved. Providing higher-level data validation mechanisms to detect such mistakes is another area for further work.

Choosing good models to capture the various aspects of a domain relevant to the solution of the problem for using an AI-based approach is a hard task. Implementors usually follow an empirical approach, based on prior experience. This is particularly true in the case of modelling domain constraints. Part of the reason is the significant variation in the domain model from instance to instance. We had examined this issue in the context of vehicle scheduling problems [6]. One of our future projects is to explore this issue further.

Acknowledgements

We would like to acknowledge the help and co-operation we have received from the staff of Indian Oil Corporation for this project, particularly, Shri P. Banerjee, Shri P.S. Krishnan, Shri V. Sundar and Shri J. Rajaraman. It has been a pleasure working with them. We would also like to thank our colleague Dr K.S.R. Anjaneyulu and the anonymous referees for their comments on this paper.

References

[1] K. Kempf, C. Le Pape, S.F. Smith, B.R. Fox, Issues in the design of AI-based schedulers: A workshop report, AI Magazine 11 (5) (1991) 37–46.
[2] K.P. Sycara, S.F. Roth, N. Sadeh, M.S. Fox, Resource allocation in distributed factory scheduling, IEEE Expert 6 (1) (1991) 29–40.
[3] M.S. Fox, Constraint-directed Search: A Case Study of Job Shop Scheduling, Morgan Kaufmann, San Mateo, CA, 1987.
[4] L.D. Bodin, B.L. Golden, A.A. Assad, M.O. Ball, Routing and scheduling of vehicles and crews: The state of the art, Computers and Operations Research 10 (1983) 69–211.
[5] V. Dhar, N. Ranganathan, Integer Programming vs Expert Systems: An Experimental Comparison, Communications of the ACM 3 (33) (1990) 323–336.
[6] M. Sasikumar, A framework for solving vehicle scheduling problems using AI techniques, MSc(Engg) thesis, Indian Institute of Science, Bangalore.
[7] J. Hendler, T. Austin, D. Mark, AI planning: Systems and techniques, AI Magazine 2 (11) (1990) 61–77.
[8] M. Zweben, M.S. Fox, Intelligent Scheduling, Morgan Kaufmann, San Mateo, CA, 1994.
[9] M. Sasikumar, V. Kripa Sunder, R. Chandrasekar, S. Ramani, A heuristic scheduling system for oil tankers, Technical Report, National Centre for Software Technology, Mumbai, 1994.
[10] R.E. Korf, Planning as search, Artificial Intelligence (33) (1987) 65–88.
[11] C. Le Pape, Solving scheduling problems with constraint propagation and a blackboard system, Information Technology (Journal of the Singapore Computer Society) 5 (2) (1993) 19–26.