PLANNING THE DISCHARGING AND LOADING OF CONTAINER SHIPS: A KNOWLEDGE-BASED APPROACH

Tat-Leong CHEW, Andrew GILL, Joo-Hong LIM

Information Technology Institute,
National Computer Board,
71, Science Park Drive, Singapore 0511,
Republic of Singapore
• Tel: 7720469 • Fax: 7795966 • email: tatleong@itivax.bitnet

Abstract

The Port of Singapore is one of the busiest and most efficient ports in the world. Fast and efficient ship planning which is the generation of schedules for container ship discharging and loading is an important determinant in ensuring the competitiveness of the port. This is a complex task of generating schedules to meet, as far as possible, planning guidelines such as optimizing equipment use while respecting numerous constraints. We describe in this paper an interactive intelligent planning tool which assists human planners in planning such schedules.

Introduction

This report describes the Ship Planning System (SPS), a fielded application at the Port of Singapore's container terminal which schedules container ship discharging and loading operations.

The planning tool includes a comprehensive graphics interface which reduces the workload of the human planner by taking over more mundane ship planning tasks such as tracking multiple plans (or scenarios) explored for a container ship or performing constraint verification during planning. In addition, it incorporates an automated planner capable of generating its own schedules, completing partial schedules generated by human planners and generating partial schedules to a point where human intervention is required.

The need for human intervention stems from the fact that, as an operational system, not only must the automated planner perform at close to expert level in terms of speed and quality, it must also be capable of deferring to a human counterpart, in an interactive fashion, decisions that it cannot make.

Human intervention requires that when automated planning fails to complete a schedule, the state of the plan is intelligible to the human planner so that he can quickly apply remedial measures. This implies a design approach that allows the automated planner to reason in a similar fashion to human planners by the use of heuristics and reasoning at multiple levels of abstraction.

We have therefore adopted a knowledge-based design approach where the emphasis is placed on explicitly representing the objects manipulated and on the reasoning used by a human planner in his task.

This approach has been used successfully in our system.

The domain

When a container ship arrives at the port, containers have to be discharged and loaded. The chief pieces of equipment necessary are cranes and transtainers. Container operations at the ship are handled by cranes which move along the length of the wharf. Containers, during their stay in the port are stacked in a storage area, known as the yard where they are handled by transtainers. Trucks carry containers between transtainers and cranes (Fig. 1).

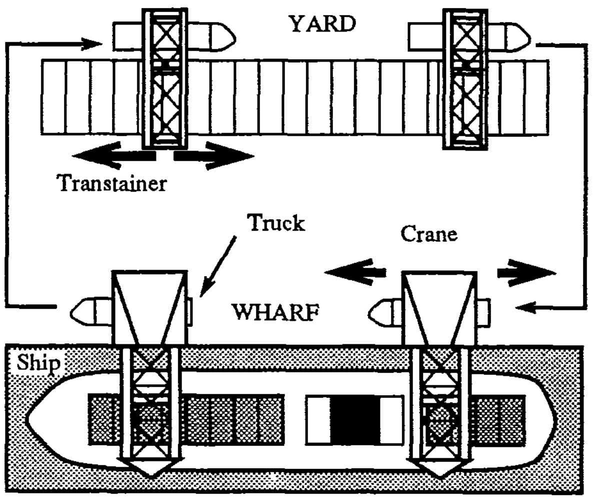

Figure 1. Simplified schematic of the yard and a ship.

The task of human planners is to generate schedules respecting numerous constraints such as:

- precedences which exist between container operations because of the way containers are stacked on board ship
- crane/transtainer clearances
- ship trim and stability
- special precedence constraints resulting from the need to move transit containers from one part of the ship to another.
- timing constraints resulting from containers connecting with other container ships.

At the same time the schedule must comply as far as possible with *planning guidelines* such as:

- minimizing the time needed to complete operations
- minimizing crane and transtainer movement
- minimizing the breaking up of logical groupings of operations.

The process of ship planning involves the following steps:

- compilation of planning information into a stowage summary (Fig. 2)
- allocating (or *splitting*) containers amongst the cranes resulting in a *split*
- simulation (or *sequencing*) of the operations while verifying constraints.

During splitting and sequencing, any constraint violation encountered is resolved by relaxing either the constraint or another conflicting constraint or guideline.

Usually, many attempts at splitting and sequencing are required before a workable plan is produced.

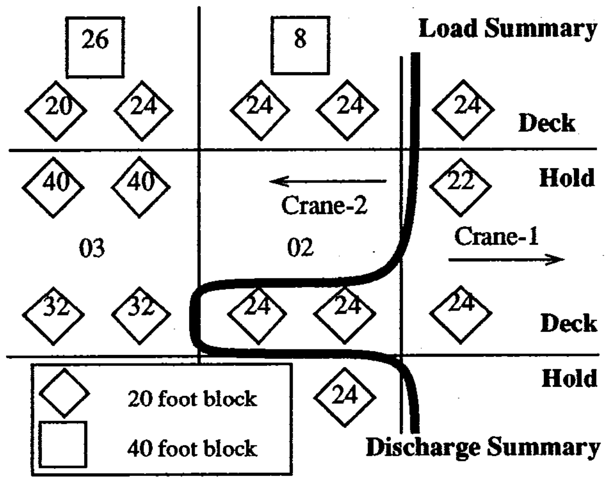

Figure 2. A simplified example of a stowage summary. The bold line partitions (or *splits*) the blocks of containers between, in this case, two cranes

Main design considerations

Based on results acquired from a 3-month knowledge acquisition period with human planners followed by a prototyping phase of another 4 months, the main tasks identified in the design of a final delivery system were:

- to design an architecture to support interactive planning;
- to develop a representation of physical objects manipulated by human planners during planning such as cranes and containers;
- to model and automate the planning process. [Chew, Gill & Lui 1989].

An interactive approach to automated planning

As mentioned in the introduction, human intervention is a necessary component to the planning process.

One reason is that the human planner is eventually accountable for any compromises in a plan and must be able to guide the automated planner in arbitration between conflicting constraints. In many cases, only a human can accomplish this role. He may have to decide whether a particular constraint violation can be ignored if it is found to jeopardize the quality of the plan severely.

Another reason is that it is impossible to enumerate ad hoc constraints arising from unforeseen operational considerations which often occur.

Finally, given the complexity of the domain, the initial delivery system cannot be expected to perform at expert level at all times. In fact, a significant amount of refinement by the human planner is expected. The delivery of a 'black box' whose workings are comprehensible only by its designers and which therefore requires their intervention in the event of an unsatisfactory plan would have condemned the planning tool to failure.

Representation for ship planning

Object oriented techniques were used to facilitate the implementation of an extensive graphic user interface and the representation of the complex real world objects manipulated by human planners during planning. These techniques help manage program complexity through high-level data abstractions while providing for a high degree of code reusability.

Objects represented in SPS include:

- objects representing the physical structure of the ship such as bays, superstructures, etc.
- objects used to construct a schedule such as cranes, containers and blocks of containers
- constraints imposed on objects which have to be verified as they are scheduled such as precedence or crane

clearance.

Our representation of blocks enables the system to model the capability of human planners of scheduling large blocks of containers and resorting to scheduling with smaller blocks when a constraint violation is to be resolved. This is solved in our system by organizing blocks into a tree hierarchy (Fig.3).

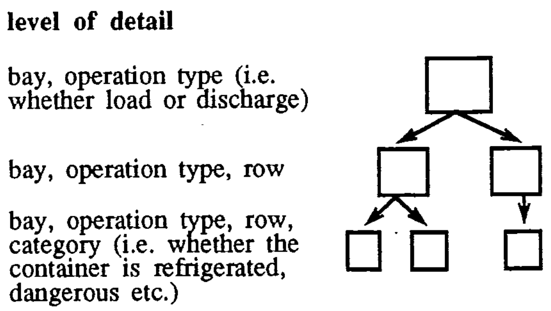
level of detail
bay, operation type (i.e. whether load or discharge)
bay, operation type, row
bay, operation type, row, category (i.e. whether the container is refrigerated, dangerous etc.)

Figure 3. The block hierarchy.

Each level of the hierarchy corresponds to a set of container characteristics. Each block at a given level is formed by containers having the same values of these characteristics. Containers at the next lower level are formed by breaking blocks of the level into sub-blocks by the addition of supplementary container characteristics. The lowest level of blocks (leaf blocks) consist often of one container. All constraints in our system are indexed by these blocks thus significantly limiting the constraints considered during verification. Since the leaf blocks of a non-leaf block can potentially be worked in any order, the verification can return *violated*, *not violated* and/or *possibly violated* constraints. A possibly violated constraint is an indication to the planner (human or otherwise) that planning ought to be done at a lower level.

Finally, the ease of indexing procedures by data structures provided by object oriented programming techniques is exploited in our system to enable a high degree of maintainability. In effect, a significant amount of the automated planner's code involves the implementation of numerous human planner heuristics for handling constraint violations. Instead of implementing these heuristics in the form of a few monolithic procedures, each handling violations for different types of constraints, the code for the implementation of these heuristics are factored out into manageable chunks indexed by each type of constraint. The addition of a new constraint requires only the addition of new code implementing a required set of behaviors (or methods in object oriented terminology) rather than the modification of existing code. This framework allows quick exploration of the use of the different heuristics in solving constraint violations.

Modeling of the planning process

We describe in this section our model of problem solving as observed in human planning. This model is made up of three hierarchical levels: interface, splitter and sequencer (Fig.4). The latter two levels form the automated planner.

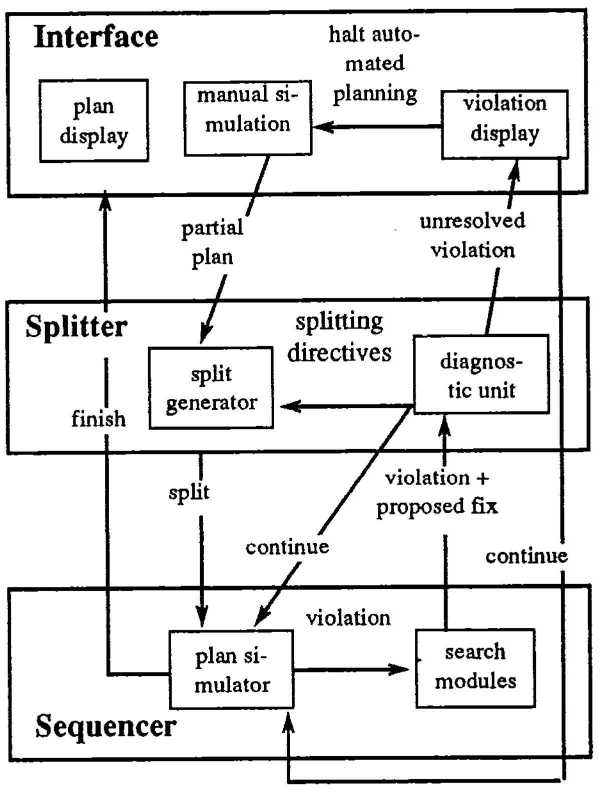

Figure 4. Architecture of the planning tool

Interface

The interface's main screen is an electronic equivalent of the pen and paper version previously used by human planners. It allows the user to manually schedule the ship at varying levels of detail from almost container level to bigger block sizes. It compiles and presents plan characteristics of all plans explored by the user. At any one time, the interface displays the current plan on which the user can manually schedule operations; any constraint violation is highlighted during this process. In this case, the human planner then overrides or corrects the constraint violation by modifying the existing schedule.

Sequencer

The sequencer accepts as input a split and schedules it to discover any constraint violations. It incorporates a series of modules which search for alternative schedules in the event of a constraint violation. Each module incorporates heuristics for localizing the search to specific cranes and blocks. Solutions found (none if no module can be applied) together with the constraint violation are returned to the splitter which decides to pursue or abandon the plan.

Splitter

The splitter consists of:

- a split generator which allocates containers to cranes
- a diagnosis unit which handles constraint violations detected when this split is scheduled by the sequencer
- a mechanism for noticing plan failure during automated planning.

The split generator implements human planner heuristics in the form of established procedures and practices for allocating containers to cranes based on an analysis of the distribution of containers in a ship. In addition, the split generator receives a set of splitting directives. These are used to force relaxation of planning guidelines during split generation.

Initially, when a partially scheduled plan is submitted to the splitter by the human planner, there are no splitting directives and only planning guidelines are used, resulting in 'ideal' splits in which planning guidelines are closely adhered to. Such splits are rarely workable and cause constraint violations when scheduled by the sequencer. These violations are reported to the diagnostic unit.

The diagnostic unit consists of a repertoire of constraint violation fixing methods. Each method implements a set of heuristics for correcting a specific type of constraint violation such as crane clearance or tight connection. The output of each method is a set of additional splitting directives to be used in the next cycle of split generation.

Plan failure is noticed by the splitter when no method can be found by the diagnostic unit to resolve a constraint violation or when the same constraint violation is encountered twice by the diagnostic unit. In such cases, the constraint violation is reported to the human planner.

Relation to other work

One of the major constraints of the project was to deliver an operational system within two years. Given the short period of time available, one of the options was to study the use of AI work in planning and especially, domain independent planners based on NOAH [Sacerdoti 1975] and ABSTRIPS [Nilsson 1980] in the hope that they could eventually be applied to our problem. However, current research ([Swartout 1988], [Chapman 1987]) indicates that a significant amount of reformulation and rigorous reconstruction of such work is still needed. Until this is done, we feel that it will be difficult to apply this work to the solution of our problem within our time frame.

Our problem solving approach resembles the approaches taken in the work on MOLGEN [Stefik 1981] and ISIS [Fox 1986] in that constraints are explicitly represented and manipulated.

We found that indexing constraints by the objects that they are associated with was a useful technique in limiting the amount of computation. Similarly, indexing the problem solving procedures by the constraints was most natural. However, we have not formulated problem solving behavior in our system as a constraint directed search since human planner heuristics are best represented as procedures.

User interface

All user interface software in SPS is based on an object oriented library of graphics routines which simplify code maintenance and enforce interface uniformity throughout the system. This library was developed in Objective-C[1], a hybrid object oriented C, giving a flavor of Smalltalk-80 to standard C. We also developed a package which elegantly allows the transparent use of these libraries in Lisp based applications. The package parses the graphic library source files and defines Flavors and Flavor-methods encapsulating interface code dealing with type translation details between Objective-C and Lisp and vice-versa. Such an approach ensures:

- user interface consistency across all Lisp and Objective-C based applications
- programmers only need learn one model of graphical user interface
- high graphic performance in the Lisp based applications since the underlying language is C and the interface layer is negligible in overhead
- centralized maintenance of only one set of libraries.

Criteria for success

One of the main criteria that gauges the success of SPS is the system's ability to reduce the current eight-hour clos-

[1] Objective-C is a trademark of the Stepstone Corporation.

ing time. This closing time is the cutoff time by which all containers to be loaded on a ship are to be declared in order to give sufficient planning time to the human planners. To exporters, a reduction in closing time is considered as an improvement in the port's services. Much of the closing time in the current procedure is spent in planning the discharging and loading operations and in data transcription between human and computer and vice-versa. The aim of SPS is to reduce the planning time by 50% and the data transcription to a negligible amount. This will then allow a shorter closing time.

In addition, the volume of container traffic is increasing at a rate which will soon outstrip the capacity of skilled planners using the current methods. A shorter planning time enabled by the use of SPS will enable human planners to cope with the increased work-load.

Payoff so far

The introduction of SPS represents a major change in the established working style and habits of the planners. The system is being phased into their work in a gradual manner. Nevertheless results so far are very encouraging. Version 1 of SPS entered operation in June 1988. This version incorporated all the user interface tools but only parts of the automated planner, notably the sequencer and not the splitter. By November 1988 it was assisting in the planning of 21% of the container ships calling at the port (130 out of 600 in November 1988). These are mostly the simpler cases where the human planners have gained sufficient confidence in the planning tool. In these cases SPS has enabled the planning time to be reduced to less than 50% of that previously needed, allowing the planners more time on complex cases. Almost all of the computer generated schedules were used without amendment in operations. The current plan is to increase the proportion of cases planned with SPS by 10% per month and to have the system plan progressively more complex cases.

Version 2 of SPS which incorporates the fully automated planner will be delivered for user testing by end January 1989 after which it will be phased into operational use.

A side-benefit of the development effort has been the formalizing of the planning methods used by human planners. This allows the study of these methods for the purpose of improvement.

In addition, SPS ensures a minimum plan quality which the human planner can try to improve upon by exploring modifications to the basic plan using the intelligent graphic user interface provided.

Costs

The project has a budget of US$1.5 million. So far, 19 person years of development effort has been spent. The project team has grown to 13 members. The team includes an experienced ship planner who has been involved full-time since the inception of the project.

Conclusion

SPS is of strategic importance to the Port of Singapore in its attempt to maintain its competitiveness and to meet the challenge of increasing container shipping volume in the 1990s. The first half year of fielding has yielded very encouraging results. Not only is it becoming a key component in the overall port automation effort, it is also a show-case in illustrating how artificial intelligence can be applied in an area of strategic economic value to Singapore.

From a technical viewpoint, SPS constitutes an innovative use of interactive hierarchical planning and object oriented techniques in a complex real world problem.

Acknowledgments

We wish to acknowledge the contributions of the SPS project team members and ship planners of the Port of Singapore Authority.

References

[Chew, Gill & Lui 1989] T.L. Chew, A. Gill and E. Lui, Invited paper for *Sun Technology* Winter 1989.

[Sacerdoti 1975] E.D.Sacerdoti, "A Structure for Plans and Behavior", SRI Technical Note 109, August 1975.

[Nilsson 1980] N.J. Nilsson, Principles of Artificial Intelligence, Morgan Kaufmann Publishers Inc., 1980, pp. 350 - 357.

[Swartout 1988] W. Swartout (editor), Workshop report of "DARPA Santa Cruz Workshop on Planning", AI Magazine, vol. 9, no. 2, Summer 1988, pp. 119 - 130.

[Chapman 1987] D. Chapman, "Planning for Conjunctive Goals", Artificial Intelligence 32, 1987, pp. 333 - 377.

[Stefik 1981] M. Stefik, "Planning with Constraints (MOLGEN: Part 1)", Artificial Intelligence 16, 1981, pp. 111 - 140.

[Stefik 1981] M. Stefik, "Planning and Metaplanning (MOLGEN: Part 2)", Artificial Intelligence 16, 1981, pp. 141 - 169.

[Fox 1986] M.S. Fox, "Observations on the role of constraints in problem solving" in Proceedings Sixth Canadian Conference on Artificial Intelligence, May 1986, pp. 172 - 187.