JCL

Borrower: WAU

Lending String: *UMR,AAA,TXA,IAY,RRR

Patron: <TN;856420> ;
<ODYSSEY;128.95.104.44/ILL>

Journal Title: The First Conference on Artificial Intelligence Applications ; Sheraton, Denver Tech Center, December 5-7, 1984 /

Volume: **Issue:**
Month/Year: 1984 **Pages:** 410-~~4~~ 416

Article Author:

Article Title: Planning with multiple resource constraints and an application to a naval planning problem

Imprint: Silver Spring, MD ; IEEE Computer Societ

ILL Number: 94186150

Call #: Q334 .C6 1984

Location: MST Second Floor NOT CHECKED OUT

Mail

Default

Charge
Maxcost: 80.00IFM

Shipping Address:
Suzzallo Library ILL
University of Washington
Box 352900
Seattle, WA 98195-2900

Fax:
Ariel: 128.95.217.16
Odyssey: 128.95.104.44
Email: interlib@u.washington.edu

PLANNING WITH MULTIPLE RESOURCE CONSTRAINTS AND AN APPLICATION TO A NAVAL PLANNING PROBLEM

A. Tate* - A.M. Whiter**

* Dept. AI, University of Edinburgh, Edinburgh, Scotland.
** Systems Designers, 105 Fleet Road, Fleet, Hants., England

ABSTRACT

This paper describes a multiple resource mechanism for a hierarchical, non-linear AI planner, NONLIN, which allows plans to be generated which satisfy resource limits and account for preferences in the resources used. This mechanism relies upon a local best then global best search strategy. Duration is treated as a special resource type. An application of NONLIN to a replenishment of naval vessels at sea planning problem is described.

INTRODUCTION

NONLIN[8] is a domain independent planning system that has been applied in several different problem areas including Block Stacking, House Building[7] Electricity Turbine Overhaul[2] and Robotic Manufacturing[9]. It generates non-linear plans at progessively greater levels of detail and can handle interactions between the components to produce a plan as a partially-ordered network of actions. Such planners are direct descendants of the NOAH planner[5]. The key features of NONLIN are described in the Appendix.

This paper describes the application of NONLIN to a Naval "Replenishment at Sea" problem and describes extensions to the NONLIN domain description language (Task Formalism) to extend the potential application area to those requiring reasoning about the use of limited availability multiple resources.

NONLIN[8] suffers from two major limitations. These are its inability to handle resource and time planning constraints. The DEVISER planner[11] has considered the addition of features to a NONLIN-like planner to handle time contraints on goals and externally time constrained events. The SIPE planner[12] allows for the specification of objects as resources upon which usage conflicts can be detected and corrected. This paper presents an enhanced version of NONLIN which is able to cope with many of the resource planning constraints required of practical applications. The features of this enhancement are:-

- Early finish plan duration is treated as a special kind of resource.
- An unlimited number of resource types is permitted.
- The total resource consumption for each resource type is evaluated incrementally as a plan is generated.
- Total Resource Consumption limits can be imposed for each resource type
- Plans are produced which try to take account of preferred relative consumption of the different resource types.

A NONLIN plan is a logically constrained partial ordering on actions. The duration information gathered in the plan indicate minimum duration requirements. It is possible to add further linearisation or to lengthen the period over which operations can be performed without effecting the plan's outcome (so long as other time critical or resource constraints are still valid). Hence, the balancing of the peak use of fixed resources is not accounted for by the extensions to NONLIN to be described below. It should be dealt with in a separate resource balancing phase or at execution scheduling time.

This paper is divided into three main sections. These deal respectively with changes to the application domain description langugage, Task Formalism[7], the change from the One-then-Best Search strategy to a Local Best then Global Best search strategy, and a practical application demonstrator which takes advantage of many of the features of

NONLIN including several of the new resource handling features.

MAINTAINING THE CUMULATIVE TOTALS FOR EACH RESOURCE TYPE

A new component is added to a representation of a node in a plan with name NODERESOURCE. It is a vector which length equal to the number of resources available. The resource vector at the last node in a plan, node(2) by convention in NONLIN, holds the cummulative use of each resource for the entire plan. NONLIN has been modified so that at each point where a critical path update function is invoked (which updates the early and late finish times as necessary for nodes in the plan) the cummulative totals for resources at the last node are maintained. Network propagation of values is not needed for this. It is sufficient to reduce the total by the resource usage level which was indicated at the higher level node being expanded and to increase it by the sum of the resource usage levels of the more detailed replacement nodes. No action is necessary when links are inserted or moved.

ACCOUNTING FOR RESOURCES IN THE PROBLEM DOMAIN DEFINITION

DECLARING RESOURCING TYPES

A Task Formalism statement called RESOURCETYPES allows the names of resource types to be declared together with the limits on consumption. Resources can be declared as "unlimited".

DURATION is a special resource always maintained by the planner. If it is not specified in a RESOURCETYPES description is is assumed to have unlimited availability. From this definition the system computes a total number of resources in use (not counting DURATION) to define the length of the resources vector for the NODERESOURCE component of each node. In addition a vector RESOURCENAME is declared with the resource names given (in order) as entries. An additional last entry is then added for DURATION. The RESOURCENAME vector is used whenever a name is to be checked as a resource and a vector subscript for it is to be found. The limit on each resource is stored in a corresponding vector called RESOURCELIMIT. "Unlimited" is stored as a very large integer. The DURATION limit is stored as a special entry.
Example:

```
RESOURCETYPES
   FUEL      10000
   DURATION  24
   WATER     UNLIMITED
```

DECLARING PREFERENCES

Specified preferences of planning objectives, e.g. minimising the consumption of a particular resource, are declared in the PREFER Task Formalism statement. User declared resource names, DURATION or CHOICETYPE can all be specified.

The PREFER statement will construct a best choice heuristic estimator from the given preference values. This new heuristic estimator incorporates the original NONLIN heuristic estimator which was based on different types of choice points. These were interaction correction ordering choices, operator choices, instantiation choices for objects, etc. There was a preference ordering on such choices. The relative importance of the choice point type can be indicated by declaring a preference value for CHOICETYPE.

If CHOICETYPE is not mentioned in the PREFER statement, it is assumed to be of the same importance as the most important resource specified.

The default heuristic estimator gives highest weight to the planner CHOICETYPE and "equal" weight to the overall plan DURATION (the DURATION weighting will only be really effective if a DURATION limit is specified). Here the default estimator is:-

```
PREFER 1 CHOICETYPE
       1 DURATION;
```

The action of the original NONLIN planner[7] can be simulated with:-

```
PREFER 1 CHOICTYPE;
```

Example

```
PREFER 1 DURATION       most important
       2 CHOICETYPE     next most important
       3 FUEL-SHIP-1 )
       3 FUEL-SHIP-2 ) treat all equally
       3 FUEL-SHIP-3 )
```

However, finer differentiation between levels is possible as an integer or floating point number may be given as the weighting.

DECLARING RESOURCE USAGE

It is possible to specify the level of resources used by an action or goal (or an estimate of this related to a high level abstraction). This is achieved by the USESRESOURCE statement. Each USESRESOURCE statement declares the usage of one or more resources associated with any action or goal.

Example:

```
USESRESOURCE    {move HMS-Bristol between
Task-Group-1 and Replenishment-Group}
  FUEL-BRISTOL = 1
  DURATION     = 15
END;
```

USESRESOURCE also allows the specificiaton of DURATIONS so a separate Task Formalism statement is not needed for that (as was the case in the earlier version of NONLIN[7]). It is also possible to override the "general" default values given in a USESRESOURCE statement for resource estimation for a particular use of an action or goal in certain schemas. In earlier versions of NONLIN this was only done for the DURATION and since there was a single resource it was mentioned at the point where an expansion component was being declared. For multiple resources this leads to a lot of clutter. Hence, a separate SCHEMA clause (at the same level as EXPANSION, CONDITIONS, ORDERINGS, etc) is used. This clause is called RESOURCES. It allows the resource usage of a particular type to be declared for a given node number in the schema expansion. The value given in USESRESOURCE (or else 0) is used for any pattern not over-ridden in this way in a particular schema.

Example:

```
RESOURCES
  FUEL    =  2 AT 2
  WATER   =  1 AT 4
```

CHANGES TO THE NONLIN SEARCH STRATEGY

THE PREVIOUS NONLIN SEARCH STRATEGY

The previous NONLIN search strategy utilised a One-then-Best backtracking scheme. This scheme focuses on the choice currently being made and tries to select one of the local choices which seems most promising. This continues while all is going well. However, if a failure occurs, the whole set of alternatives which have been generated (and ranked by a heuristic estimator) are re-considered to select the re-focusing point for the search.

Specific to NONLIN the local decision points are schema selection, linearisation selection, and instantiation selection. The first local legal choice is always selected. The remaining alternative choices together with the present state of the plan are added to the global choice points list, in a position which is determined by the heuristic estimator. The heuristic estimator is one of three values corresponding to the choice type (0.1 for schema choice; 0.5 for linearisation choice; 0.9 for instantiation choice). When a failure occurs a choice point is selected from the front of the global choice points list (the most recent of those having the smallest heuristic estimator) and is used to restart plan generation.

THE ENHANCED NONLIN HEURISTIC ESTIMATOR

The heuristic estimator ϵ is calculated thus:

$$\epsilon = \frac{r_1}{k_1} + \frac{r_2}{k_2} + \frac{r_3}{k_3} + \ldots + \frac{r_{choicetype}}{k_{choicetype}}$$

where

r_i is a normalised value (between 0 and 1) discussed below for each resource (including DURATION).

k_i is the number specified in the relevant preference definition. It should be lower for resources considered to be most important for the estimate.

$r_{choicetype}$ is 0.1 schema choice
0.5 linearisation choice
0.9 instantiation choice

$k_{choicetype}$ is either the specified preference value for CHOICETYPE or if no preference is declared it is the same as the smallest declared preference.

An ordering can be specified which tells the system to prefer solutions which use low quantities of a resource in preference to those which are low in other resources.

There needs to be some way of comparing the quantities of resources of the various types. Ideally we would like to reduce each to a value between 0 (for no resources used) and 1 for the limit of resource used.

Noderesource(Node(2)) <- cumulative value
Resourcelimit(i) at last node

However, UNLIMITED resources pose a problem unless you consider that it is not very useful to say that a resource is UNLIMITED and then use it to weight the relative merit of solutions. At least an upper estimate should be available if it really is wanted to take the resource into account for choice purpose. However, even if it is not, the very large value set as the "limit" for unlimited resources means that it may figure in the evaluation if desired but will tend to add very little to the value (will give little choice differentiation).

DURATION is a special case, computed as follows:

$$\frac{\text{EarlyFinish (Node(2))} \quad \text{Finish time <-for last node}}{\text{Duration Limit}}$$

again it is only helpful to use this when the DURATION has been limited.

UTILISING THE NEW HEURISTIC ESTIMATOR IN A LOCAL BEST THEN GLOBAL BEST SEARCH STRATEGY

Selection of a Local Best In NONLIN, local best decisions can only be determined for schema selection for node expansion and for linearisation selection for sub-goal conflict resolution. The effect of choosing different instantiations cannot be determined at a local level due to the lazy evaluated nature of the choice mechanism in NONLIN[8]. This type of local decision is therefore left unchanged in the new scheme, i.e. the first legal instantiation is chosen.

The list of schemas which can be used to expand a chosen node is reduced by eliminating those schemas which if used for the expansion would result in the violation of a non-DURATION type resource limit.

The effect that expanding a node has on the early finish time of the entire plan can only be determined by invoking the network propagation update procedure for the early and late finish times. This is felt to be a considerable overhead and is not therefore undertaken in the present version of NONLIN. However, if the parent node lies on a critical path then the difference between the parent's duration and each schema's expansion critical path length (which could be calculated in advance of its inclusion in the plan) can be used to determine a lower limit for the early finish time of the plan. If this value exceeds the duration limit then the appropriate schema is eliminated from the list of considered schemas.

For each of the schemas left, the heuristic estimator is evaluated using the early finish time of the plan before expansion, the new cummulative totals for the non-DURATION type resources resulting from the particular expansion, and the choice type value for a schema choice. The alternative schemas are formulated into choice points using the schema, the present state of the plan and the heuristic estimator and these are added to the global choice points list.

The process of choosing a best local linearisation requires an analysis of the different effects on the early finish time of the plan. This requires the network propagation of the early and late finish times for each alternative linearisation. Those linearisations that cause the finish time of the last node to exceed the duration limit are eliminated and the linearisation having the smallest early finish time is deemed to be the local best choice. For each of the remaining linearisations the heuristic estimator is evaluated and used to construct the choice points which are then added to the global choice points list.

Although it is not possible in NONLIN to determine the local best instantiation (due to lazy evaluation of this choice type) it is necessary to evaluate the heuristic estimator for the alternative instantiation choice points to ensure they take their correct place when added to the global choice points list.

Selection of a Global Best Whenever a local decision cannot be made because no legal decision points are available or when the local best choice fails, plan generation is restarted from the choice point in the global choice points list having the smallest heuristic estimator. If there is more than one choice point having this same value then the most recent is chosen to restart plan generation.

A NAVAL APPLICATION OF NONLIN

THE PROBLEM DOMAIN

The problem domain is the replenishment of naval vessels at sea. The problem has many degrees of freedom but relatively few successful solutions. It is also indefinitely extendable in terms of additional goals and constraints.

The present state of the problem domain is as follows:

Scenario

- several task groups and one replenishment group
- each group has a number of escorts
- each escort is individually named and has three characteristics:
  - its group
  - its type {MRADS (Air Defence Missile Ship)
    PDMS (Point Defence Missile Ship)
    FF (Anti Submarine Frigate)}
  - its replenishment state {unreplenished replenished}

Constraints to Movement

- a minimum number of escorts must be maintained at each group (different for each group).
- a minimum number of escorts of a given type must be maintained at each group (different for each type at each group).
- escorts must move in pairs (but not two MRADS types).

Resource Constraints

- each movement of an escort depletes the overall fuel usage by a predefined amount which is different for each escort type.
- a limit is imposed on the overall fuel usage.

Planning Problem

- to move escorts between groups and replenish certain escorts at the replenishment group so as to achieve the "replenishment" of one or more groups (in parallel or otherwise).
- a group is deemed to be replenished when there is at least a predefined number (the 'normal' number) of replenished escorts at the group.

THE TASK FORMALISM RESOURCE CONSTRUCTS

Figure 1 shows a sample schema for the problem. There are 16 schemas in all.

```
actschema     change-group-MRADS
pattern       {escort @*a with @*groupl to
               @*group2}
expansion     1 dummy
conditions    usewhen {escort @*a with
                @*groupl} at 1
              usewhen {escort @*a is a
                MRADS} at 1
resources     fuel = 15 at 1
effects       - {escort @*a with @*groupl}
              + {escort @*a with @*group2}
vars          a undef
              groupl undef
              group2 <:non @*groupl:>
end;
```

Figure 1: A simple example of an action schema having a resource consumption attribute

The declarations of resource type and resource preference are simply:

```
RESOURCETYPES  FUEL <limit>;

PREFER         1 FUEL;
```

Declaration of resource usage was dealt with in three almost identical action schemas which move a single escort from one group to another. The only difference between them was the amount of fuel used given the ship type.

AN EXAMPLE PLAN

The aim of the example plan is to replenish one task group (task-group-1) from the replenishment group. This is achieved when there are at least five replenished escorts at the group.

Initially there are five unreplenished escorts at the task group, two are MRADS type; two are PDMS type; and one is an FF type. There must always be at least three escorts (including at least one MRADS type and one PDMS type) and two replenished escorts (one MRADS type and one FF type) at the replenishment group. There must always be at least one MRADS type and one PDMS type escorts at the replenishment group.

Whenever an escort moves it uses up an amount of the total fuel available (15 for MRADS, 10 for PDMS, and 5 for FF).

The actions only version of this example plan is given in Figure 2.

No.	Action	Pre.	Succs.
1	{plan start}		3 4
2	{plan finish}	17 18	
3	{escort HMS1 from RGtoTG}	1	5
4	{escort HMS2 from RGtoTG}	1	5
5	{no action}	4 3	7 6
6	{escort HMS3 from TGtoRG}	5	8 9
7	{escort HMS4 from TGtoRG}	5	8 9
8	{replenish HMS5}	6 7	10 11
9	{replenish HMS4}	6 7	10 11
10	{escort HMS5 from RGtoTG}	8 9	12
11	{escort HMS4 from RGtoTG}	8 9	12
12	{no action}	10 11	13 14
13	{escort HMS6 from TGtoRG}	12	15 16
14	{escort HMS7 from TGtoRG}	12	15 16
15	{replenish HMS8}	13 14	17 18
16	{replenish HMS6}	13 14	17 18
17	{escort HMS8 from RGtoTG}	15 16	2
18	{escort HMS6 from RGtoTG}	15 16	2

FUEL USAGE = 100

Figure 2: The actions only version of the example plan

TG and RG are abbreviated constants for the Task Group and Replenishment Group respectively. HMS1, HMS2,... etc., are constants referring to specific escorts.

This plan took approximately 3 minutes of CPU time (in POP-11 under POPLOG on a VAX 11/780:) to complete and involved about 50 trial expansions. Plans which replenish two task groups of a similar size generally take about 30 minutes of CPU time to complete. These times may be acceptable for some applications. However, the purpose of this work was to provide a demonstration of the future potential of the technique.

CONCLUSIONS

It is envisaged that the mechanisms presented in this paper will cater for a large class of practical planning applications which involve resource handling. However there are still some serious limitations in this approach to resource handling. A major limitation is that at the abstract level of planning it is quite often impossible to provide the resource usage associated with any particular abstract action. This might result in a plan being developed which utilises several "cheap" actions whose total cost exceeds that of a single more "expensive" action which would have been more appropriate. This occurs when the single actions have been introduced in different expansions. It would be preferable to automatically compute averages of the various lower level resource usages for each higher level task.

A default "unknown" resource usage (instead of zero) together with a search strategy of depth first until a heuristic estimator can be evaluated then global best could be utilised to overcome this problem. Since a problem in NONLIN is the explosive growth of choicepoints and database contexts in a planning domain which involves many degrees of freedom but few successful solutions it may be impractical to implement such mechanisms in the original NONLIN. Dependency directed backtracking extensions to NONLIN[1] have been experimented with to overcome some of these problems. The resource handling mechanisms presented and proposed in this paper may well be a valuable contribution to a planner incorporating opportunistic[4] and dependency-directed[1,3,6] search mechanisms together with an efficient context-layered database.

REFERENCES

[1] Daniel L. and Tate, A (1982)
"A retrospective on the 'Planning: a joint AI/OR approach' project"
D.A.I. working paper no. 125 Edinburgh University.

[2] Daniel L. (1983)
"Planning and Operations Research" in "Artificial Intelligence : Tools, Techniques and Applications" Harper and Row, New York.

[3] Hayes, P.J. (1975)
"A representation for Robot Plans"
Advance papers of IJCAI-75, Tbilisi, USSR.

[4] Hayes-Roth, B. and Hayes-Roth, F. (1979) "A Cognitive Model of Planning",
Cognitive Science 1979 pp 275-310

[5] Sacerdoti, E.D. (1975)
"The Non-linear Nature of Plans"
Advance papers of IJCAI-75, Tbilisi, USSR.

[6] Stallman, R.M. and Sussman, G.J. (1977)
"Forward Reasoning and Dependency Directed Backtracking"
Artificial Intelligence Vol. 9 pp 135-196.

[7] Tate, A. (1976)
"Project Planning Using a Hierarchic Non-Linear Planner"
Department of Artificial Intelligence, Report 25, Edinburgh University.

[8] Tate, A. (1977)
"Generating Project Networks"
Proceedings of IJCAI-77, Boston, USA.

[9] Tate, A. (1984a)
"Planning and Condition Monitoring in a FMS"
Conference on the Development of Flexible Manufacturing Systems"
London, U.K.

[10] Tate, A. (1984b)
"Goal Structure - capturing the Intent of Plans"
European Conference on AI, Pisa, Italy.

[11] Vere, S. (1981)
"Planning in time : windows and durations for activities and goals"
NASA Jet Propulsion Lab. Technical Report.

[12] Wilkins, D.E. (1981)
"Representation in a Domain-Independent Planner"
IJCAI-83 pp 733-740, Karlsruhe, W. Germany.

APPENDIX

THE AI MECHANISMS OF NONLIN

Expansion Selection

Each node in an emerging plan has an associated pattern. If the pattern is not a primitive (no expansion or conditions) can generally be expanded in a number of alternative ways. These alternatives are defined within the Task Formalism in the form of schemas. The Task Formalism is the language in which the planning domain is defined. Schemas are templates which allow an expansion, resources, and/or effects to be associated with a given pattern. Generally each schema also includes a series of conditions which either must be satisfied before a schema can be used or impose constraints which are satisfied by adding links at some stage in the plan generation process.

Any pattern can have several alternative defining schemas. If it is required to expand such a node with such a pattern then the schema which if expanded does not violate any resource or duration limit and which also minimises the overall resource usage (evaluated by a heuristic estimator which accounts for preferred resource usages) is chosen for the expansion.

Node Expansion

A node is expanded so that its conditions and preceeding nodes are transferred to the starting node of the expansion and its effects and succeeding nodes are transferred to the end node of the expansion.

Adding and Choosing Choice Points

Whenever a decision is taken within the plan generation process the alternative decisions are added to a global choice points lists as choice points. A choice point contains the state of the plan generated together with the alternative decisions which could have been taken. Each choice point has associated with it a heuristic ordering parameter. This parameter incorporates the preferred decision type, i.e., alternative schemas then alternative links and finally alternative instantiations, and the heuristic estimator for overall resource usage. Whenever plan generation has to back-track to an alternative choice point the choice point with the "best" ordering parameter is invoked. This One-then-Best back-tracking strategy can therefore be thought of as a "local best" then "global best" decision strategy.

Logical Conflict Detection and Recovery

This is possibly the most complex of the mechanisms within NONLIN. The process of logical conflict detection is very dependent upon two data structures called the Table of Multiple Effects (TOME) and the Goal Structure (GOST) (Tate, 1984b). The TOME, which was first introduced by Sacerdoti, provides a simple means of determining the values of any pattern at any node quickly. The GOST which is an original feature of NONLIN, allows purposes of any particular effect at any node to be determined (if it has any). This allows interactions to be detected and allows corrections (suggested links) to be sensitive to the important effects of nodes (those with purposes). Unimportant effects are therefore ignored.