IxTeT : un système de planification hiérarchique gérant le temps et les ressources

Philippe Laborie
LAAS/CNRS
7, Avenue du Colonel Roche
31077 Toulouse Cedex, France
email: laborie@laas.fr

Résumé - Cet article décrit IxTeT , un système de génération de plans d'actions d'une grande expressivité qui permet de résoudre une large gamme de problèmes réalistes. Le formalisme repose sur une logique temporelle réifiée permettant de représenter le monde au travers d'un ensemble d'attributs d'état valués dont la valeur varie au cours du temps et d'un ensemble de ressources utilisables par les opérateurs. La planification s'effectue par une résolution en concurrence des différents défauts du plan partiel courant (sous-buts pendants, menaces sur les protections, conflits de ressource) en suivant une stratégie de moindre engagement. Chaque type de défaut est analysé par un module spécialisé. Une approche originale de hiérarchisation de la recherche (hiérarchie dynamique) à été implémentée. La riche expressivité du formalisme et l'exigence de complétude du système sont supportées par une algorithmique de contrôle sophistiquée qui permet d'atteindre des performances satisfaisantes pour des problèmes réalistes.

Mots clés - planification temporelle, gestion de ressources, hiérarchisation de la recherche.

Introduction

Dans l'état de l'art en planification, deux types de travaux de recherche peuvent être distingués: d'une part des travaux, essentiellement basés sur le formalisme STRIPS, visant à caractériser la complexité du processus de planification, à proposer différents algorithmes de parcours de l'espace de recherche et à établir certaines propriétés de ces algorithmes (complétude, systématicité, ...) [Chapman, 1987] [McAllester et Rosenblitt, 1991] [Knoblock et al., 1991] et d'autre part, des travaux visant à enrichir le formalisme STRIPS pour pouvoir résoudre de manière performante des problèmes se posant en pratique et faisant, par exemple, intervenir des contraintes temporelles ou des contraintes de ressources [Wilkins, 1988][Currie et Tate, 1991].

Cet article décrit un système de génération de plan d'actions basé sur le formalisme IxTeT développé au LAAS/CNRS depuis environ 8 ans sous la direction de Malik Ghallab. Il s'agit donc principalement d'une synthèse. La contribution personnelle de l'auteur se situe essentiellement au niveau de la gestion de ressources et de la hiérarchisation de la recherche.

L'un des objectifs principaux du système de planification IxTeT est de pouvoir résoudre une large gamme de problèmes réalistes tout en assurant certaines propriétés de l'algorithme de recherche (en particulier sa complétude). L'approche emprunte donc un certain nombre d'idées aux travaux ci-dessus (ainsi le mécanisme d'explication par liens causaux [McAllester et Rosenblitt, 1991], la propriété de monotonicité de la hiérarchie d'abstraction [Knoblock et al., 1991], la notion de défaut d'un plan partiel [Currie et Tate, 1991]), idées qui sont intégrées et étendues à une représentation et à des d'algorithmes originaux aussi bien en ce qui concerne la gestion du temps et des ressources qu'en ce qui concerne la hiérarchie d'abstraction utilisée par le système.

La première section de cet article présente le formalisme IxTeT qui permet d'exprimer toutes les connaissances nécessaires au système de planification. La section suivante décrit le mécanisme de planification en insistant particulièrement sur les principales originalités de l'approche (détermination d'une tâche pour résoudre un sous-but, gestion des ressources, stratégie de moindre engagement, hiérarchie d'abstraction dynamique). Enfin, deux exemples illustrant la variété des problèmes exprimables et résoluble avec IxTeT sont présentés dans la dernière section.

1 Le formalisme IxTeT

1.1 Représentation du monde

A un instant donné, l'état courant du monde est décrit par un ensemble d'*attributs d'état valués* et un ensemble d'*attributs de ressources*. Notre représentation du monde est supposée **complète et exhaustive**: toutes les connaissances relatives à l'état du monde à un instant donné sont entièrement explicitées. Nous supposons d'autre part que nos opérateurs de planification (tâches) sont **déterministes**.

Les attributs d'état

Un **attribut d'état** permet de décrire une caractéristique non-cumulative du monde à un instant donné; par exemple, la position d'un robot $robot1$ dans un lieu donné $piece1$ est représentée par l'attribut $position(robot1) : piece1$. Par hypothèse, un attribut d'état ne peut prendre qu'une valeur à un instant donné.

Un des intérêt principaux à l'utilisation d'attributs valués plutôt que des relations binaires est qu'ils permettent d'exprimer de manière tout à fait naturelle des contraintes telle que la non-ubiquïté sans avoir pour cela besoin de définir de règles particulières.

Les domaines possibles pour les arguments d'un attribut d'état sont explicitement déclarés; ainsi dans le cas ci-dessus on aura:

```
attribute position(?robot){
```

```
  ?robot in {robot1, robot2};
  ?value in {piece1, piece2, couloir};
 }
```

Selon leur dynamicité plusieurs types d'attributs d'états sont distingués:

- les attributs **rigides** sont ceux dont la valeur ne change pas au cours du temps et qui souvent, expriment une relation structurelle entre les arguments, par exemple $lieu(machine1) : piece1$;
- les attributs **flexibles** dont la valeur peut changer au cours du temps; parmi ceux-ci on distingue:
  - les attributs **contrôlables** qui sont ceux dont la valeur pourra être modifiée suite à des actions du plan, par exemple $position(robot1) : piece1$
  - les attributs **contingents** dont les changements de valeur ne peuvent être contrôlés par l'agent qui planifie, par exemple $lumiere\_naturelle(lieu1) : nuit$

Les attributs de ressource

Une **ressource** est définie comme toute substance ou ensemble d'objets dont le coût ou la quantité disponible induit des contraintes sur les actions qui l'utilisent.

Pour une action donnée, plusieurs ressources peuvent être considérées comme équivalentes. Nous dirons que deux ressources sont de même *type* ssi il existe au moins une action qui peut utiliser indifféremment l'une ou l'autre de ces ressources. Les ressources d'un type donné sont explicitement déclarées; par exemple:

```
resource puissance_electrique(?batterie){
 ?batterie in {batterie1, batterie2};
 capacity(batterie1)=50;
 capacity(batterie2)=60;
}
```

Un **attribut de ressource** permet de spécifier une certaine quantité de ressource utilisée par une action à un instant donné; par exemple $puissance\_electrique(batterie1) : 10$.

1.2 Représentation des contraintes temporelles

Pour des raisons de complexité algorithmique, le **gestionnaire de contraintes temporelles** d'IxTeT repose sur une représentation à base d'**instants**. Sont gérées à la fois:

- des contraintes temporelles symboliques (i.e. contraintes de précédence: $t < t'$, de simultanéité: $t = t'$); et
- des contraintes numériques exprimées sous la forme d'un intervalle de réels bornant la distance temporelle entre deux instants (du type $t - t'$ *in* $[d_{min}, d_{max}]$). Ces contraintes numériques doivent être gérées différemment selon qu'elles expriment une contrainte contrôlable (i.e. l'agent planifiant peut choisir une valeur dans l'intervalle $[d_{min}, d_{max}]$ pour la durée entre t et t') ou bien une contrainte temporelle contingente (dans ce cas, l'intervalle représente une incertitude sur la durée entre t et t' incontrôlable par l'agent qui planifie) [Ghallab et Vidal, 1995b].

La complexité de la propagation de contraintes symboliques, dans le cas où le réseau de contraintes temporelles est purement symbolique est, en moyenne, en $O(n)$ si n est le nombre d'instants [Ghallab et Mounir-Alaoui, 1989]. Celle d'une contrainte numérique est en $O(n^2)$ [Ghallab et Vidal, 1995b]. Dans le cas où le réseau contient à la fois des contraintes symboliques et des contraintes numériques, les contraintes sont gérées dans deux réseaux différents et des algorithmes spécifiques permettent d'assurer la cohérence entre les deux réseaux [Ghallab et Vidal, 1995a]. Ceci permet d'obtenir d'excellentes performances lorsque le nombre d'instants numériques (i.e. ceux sur lesquels portent au moins une contrainte numérique) est faible devant le nombre d'instants symboliques.

Les disjonctions de contraintes temporelles ne sont pas directement gérées à ce niveau mais, comme nous le verrons plus bas, au niveau de l'algorithmique de contrôle.

1.3 Représentation des contraintes sur les variables atemporelles

Toutes les variables sont considérées comme ayant un domaine de valeurs possibles discret et fini. Le **gestionnaire de contraintes sur les variables** permet d'exprimer les contraintes suivantes:

- appartenance à un domaine: $?x \ in \ \{X_1, X_2, ..., X_n\}$
- égalité: $?x = ?x'$
- inégalité: $?x \neq ?x'$
- dépendance:
  $?x \ in \ \{X_1, X_2, ..., X_k\} \Rightarrow ?y \ in \ \{Y_1, Y_2, ..., Y_l\}$

Le rôle du gestionnaire de contraintes sur les variables est de vérifier la cohérence globale du réseau[1], de propager ces contraintes et de répondre à des requêtes (par exemple: deux variables peuvent-elles s'unifier ? peuvent-elles prendre simultanément des valeurs différentes ?, ...). Dans notre implémentation du système, les contraintes d'égalité permettent de définir des classes d'équivalences de variables; lorsqu'une contrainte d'égalité entre deux variables est posée, les deux classes d'équivalences sont fusionnées, le domaine de la nouvelle classe d'équivalence devient l'intersection des deux anciens domaines et les autres contraintes sont propagées. Les contraintes d'inégalités ou de dépendance entre classes d'équivalences de variables sont propagées par des techniques classiques de consistance d'arc.

1.4 Représentation de la persistance et du changement

IxTeT repose sur une logique réifiée où les attributs d'état flexibles sont qualifiés temporellement par les prédicats *event* et *hold*.

Une assertion $hold(att(x_1, ..., x_n) : v, (t_1, t_2))$ exprime la persistance de la valeur de l'attribut $att(x_1, ..., x_n)$ à v sur l'intervalle temporel $[t_1, t_2]$. Nous supposons que tous les attributs d'état sont homogènes c'est à dire que la valeur d'un attribut sur un intervalle temporel est héritée sur tous les sous-intervalles. A noter que t_1 et t_2 sont des instants (variables temporelles) du réseau de contraintes temporelles (c.f. §1.2) et $x_1, ..., x_n$ des variables atemporelles qui peuvent supporter des contraintes (c.f. §1.3).

Un événement $event(att(x_1, ..., x_n) : (v_1, v_2), t)$ représente un changement instantané de valeur à l'instant t pour l'attribut $att(x_1, ..., x_n)$ de l'ancienne valeur v_1 à la nouvelle valeur v_2. Les événements permettent d'exprimer des changements discrets du monde; nous ne représentons pas les changements continus comme cela est fait dans [Penberthy et Weld, 1994].

1.5 Représentation de l'utilisation de ressource

Comme pour les attributs d'états, l'utilisation de ressource est représentée par une réification temporelle des attributs de ressources par les prédicats *use*, *consume* et *produce*.

Nous distinguons en effet trois modes d'utilisation possibles d'une ressource par une action:

[1] Pour les deux derniers types de contraintes, la vérification de la cohérence globale du réseau est un problème NP-complet aussi nous contentons nous dans ce cas d'une cohérence locale ce qui ne remet toutefois pas en cause la complétude du système de planification.

- la *consommation* d'une certaine quantité q de ressource à un instant donné t: après exécution de l'action la ressource devient moins disponible qu'elle ne l'était avant: $consume(res(x_1, ..., x_n) : q, t)$ (ex: consommation de carburant);
- la *production* de ressource à un instant donné: $produce(res(x_1, ..., x_n) : q, t)$ (ex: production de carburant par l'action "faire le plein");
- l'*emprunt* de ressource pour lequel la capacité de la ressource n'est affectée que sur un intervalle temporel donné: $use(res(x_1, ..., x_n) : q, (t_1, t_2))$ (ex: utilisation d'un outil pour une action).

Comme nous le verrons par la suite, pour des raisons algorithmiques, les ressources sont gérées au niveau des intervalles temporels plutôt que des instants. Dans ce cadre là, on peut constater que la consommation de ressource à un instant t n'est autre que son emprunt sur un intervalle $[t, +\infty]$ et que de manière similaire (mais un peu moins intuitive), la production de ressource à un instant t peut être représentée par une augmentation de la capacité (atemporelle) de la ressource jointe à un emprunt de celle-ci sur l'intervalle $[-\infty, t]$ (cf figure 1). Ceci permet, au niveau de l'algorithmique de la gestion de ressource, de ne faire intervenir que des prédicats d'emprunt sur des intervalles temporels (use).

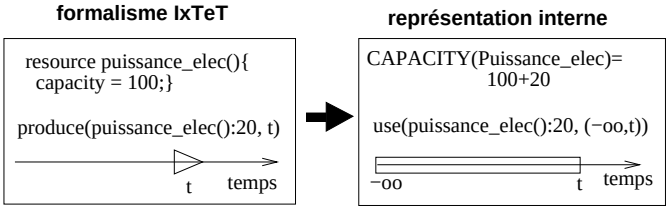

Figure 1: *représentation interne d'une production de ressource*

Nous faisons l'hypothèse que les quantités q utilisées sont des constantes réelles. Dans IxTeT, on ne distingue pas, au niveau de la représentation, la typologie classique des ressources (ressources non-partageables, partageables, uniquement consommables, productibles, ...) telle qu'elle est définie par exemple dans [Drabble, 1995]. Le mode d'utilisation d'une ressource n'est pas directement attaché à celle-ci mais dépend de l'action qui l'utilise. Ceci est justifié par le fait qu'une même ressource peut, selon l'action pour laquelle elle est utilisée être considérée par exemple comme ressource partageable ou bien comme ressource productible. Cette représentation des ressources permet une grande expressivité pour formuler des problèmes réalistes, qu'il s'agisse aussi bien de problèmes de synthèse de plan d'action que de problèmes plus classiques d'ordonnancement [Laborie et Ghallab, 1995]. Comme nous le verrons plus loin, cette représentation riche ne constitue un obstacle ni pour les performances du système, ni pour sa complétude.

1.6 Le formalisme de tâche

Dans IxTeT, les schémas d'opérateurs de planification sont appelés **tâches**. Une hiérarchie de tâches est définie où chaque tâche est composée:

- d'un ensemble de sous-tâches;
- d'un ensemble d'événements permettant de décrire les changements du monde déclenché par l'exécution de la tâche;
- d'un ensemble d'assertions sur des attributs d'états permettant d'exprimer certaines conditions à l'exécution de la tâche ainsi que certaines protections nécessaires à son bon déroulement;
- d'un ensemble de propositions d'utilisation de ressources spécifiant quelles ressources sont utilisées par la tâche et de quelle manière;
- d'un ensemble de contraintes temporelles et de contraintes sur les variables liant les différentes variables (temporelles ou non) de la tâche.

Les sous-tâches auxquelles fait référence une tâche donnée sont spécifiées avec la même syntaxe. Cette représentation hiérarchique ne doit pas être récursive; de plus il s'agit uniquement d'une hiérarchie de représentation visant à simplifier l'écriture des tâches par l'opérateur. Cette hiérarchie n'est pas, à l'heure actuelle, liée à la hiérarchie de contrôle qui sera discutée plus loin. Nous supposons d'autre part que les tâches sont des opérateurs déterministes et que tous les changements apporté par l'exécution d'une tâche sont décrits dans celle-ci.

Voici un exemple de tâche élémentaire (ne contenant pas de sous-tâche) dans un monde où il s'agit de planifier, pour un robot, des activités de maintenance dans un laboratoire (livraison du courrier, surveillance du bon fonctionnement des machines: imprimantes, fax, photocopieuses, ...). Le robot est limité quant aux objets qu'il peut transporter simultanément.

```
resource place_sur_robot(){ capacity = 3;}
resource papier_sur_robot(){ capacity = 0;}

task alimente_en_papier(?machine) (start,end){
  variable ?salle;
  lieu(?machine,?salle);
  hold    (position(robot):?salle, (start,end));
  event   (etat_machine(?machine):(ko,ok), end);
  consume (papier_sur_robot():1, end);
  produce (place_sur_robot():1, end);
  (end - start) in [00:01:00,00:02:00];}
```

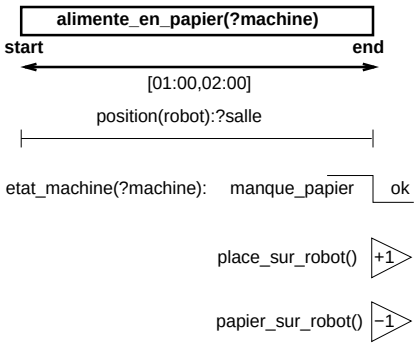

Figure 2: *La tâche* `alimente_en_papier`

La tâche consiste à remplir le bac à papier d'une machine ($?machine$) qui est dans une salle donnée ($?salle$). La contrainte liant une machine donnée au lieu où elle est située est représentée par l'attribut rigide *lieu* qui est codé sous la forme de contraintes sur les variables du type: $?machine\ in\ \{fax1\} \Rightarrow ?salle\ in\ \{Salle3\}$; .... L'assertion impose que le robot reste dans la salle $?salle$ pendant toute la durée de la tâche. L'événement décrit le changement d'état de la machine à l'instant de fin de la tâche; à noter que la première valeur d'un événement (ici $etat\_machine(?machine) : ko$) exprime aussi une condition à l'exécution de la tâche. Enfin, l'exécution de la tâche va entraîner la consommation d'une rame de papier parmi celles que portait le robot et, ce faisant, la libération d'un espace libre supplémentaire sur la plateforme du robot et sa durée s'étalera sur 1 ou 2 mn.

Le problème initial $\mathcal{P}_{init}$ est une tâche particulière qui décrit le scénario du problème, c'est à dire:

- la valeur initiale pour l'ensemble des instances des attributs d'état;

- les changements attendus au niveau de certains attributs d'état contingents qui ne seront pas contrôlé par le système;
- les quantités de ressources disponibles et le profil de disponibilité des ressources; et enfin
- les buts qui doivent être résolus.

IxTeT permet une flexibilité totale quant à l'expression de contraintes temporelles entre ces différents éléments. Voici un exemple très simplifié de problème pour le monde décrit ci-dessus:

```
task probleme1() (start,end){
  timepoint t_fax1, t_but;
  explained event (position(robot):(?,salle1), start);
  explained event (etat_machine(fax1):(?,ok), start);
  explained event (etat_machine(fax1):(ok,ko), t_fax1);
  hold (etat_machine(fax1):ok, (t_but,end));
  (t_fax1 - start) = 10:00:00;
  (t_but  - start) = 10:10:00;}
```

Initialement, le robot est en salle $salle1$ et le fax $fax1$ est en état de fonctionnement. On sait qu'à 10h, il manquera de papier et qu'il faudra le remettre en état de fonctionnement avant 10h10. Initialement, le robot n'a pas de rame de papier sur lui: il devra donc d'abord passer préalablement au magasin.

2 La synthèse d'un plan d'action

Le système de planification explore un arbre de recherche dont la racine est le problème initial, les noeuds sont des plans partiels et les feuilles des plan solutions. Dans cette partie, Nous allons présenter l'algorithme général de recherche d'une solution et les modules plus spécifiques d'analyse d'un plan partiel qui sont utilisés par cet algorithme puis, nous nous attarderons sur la manière dont sont effectués les choix non-déterministes de l'algorithme général.

2.1 Critère de plan solution et Algorithme général

A un noeud donné de l'arbre de recherche correspond un plan partiel courant $\mathcal{P}$. Sur ce plan partiel nous distinguons 3 types de **défaut** qui vont nécessiter un raffinement ultérieur de ce plan partiel: les propositions non-expliquées (ou sous-buts), les contraintes sur des persistances possiblement invalides (menaces sur des protections) et les conflits de ressource potentiels.

Définition 1 (proposition non-expliquée)
Dans un plan partiel $\mathcal{P}$, une proposition temporelle $hold(att(x_1,...,x_n) : v,(t,t^*))$ ou $event(att(x_1,...,x_n) : (v,v^*),t)$ sera dite expliquée *ssi:*

1. *il existe un* événement établisseur $event(att(x'_1,...,x'_n) : (v_*,v'),t')$ *tel que nécessairement $(t' \leq t)$, $(v' = v)$, $(x'_1 = x_1)$, ... et $(x'_n = x_n)$; et*
2. *si $(t' < t)$, il existe une assertion (*lien causal*) $hold(att(x_1,...,x_n) : v,(t',t))$ entre l'événement établisseur et la proposition.*

Une menace est une proposition temporelle (événement ou assertion) qui risque de violer une persistance imposée (assertion). Plus précisément:

Définition 2 (menace sur une protection)
Une menace est:

1. *soit un couple $< e, cl >$ où $e = event(att(x'_1,...,x'_n) : (v_*,v^*),t')$ et $cl = hold(att(x_1,...,x_n) : v,(t_1,t_2))$ tel qu'aucune des contraintes suivantes ne peut être déduites du plan partiel $\mathcal{P}$: $\{(t' < t_1);(t' = t_1$ et $v = v^*);(t_2 < t');(t' = t_2$ et $v = v_*);(x_1 \neq x'_1);...;(x_n \neq x'_n)\}$; ou*
2. *soit un couple $< h, cl >$ où $h = hold(att(x'_1,...,x'_n) : v',(t_*,t^*))$ et $cl = hold(att(x_1,...,x_n) : v,(t_1,t_2))$ tel qu'aucune des contraintes suivantes ne peut être déduites du plan partiel $\mathcal{P}$: $\{(t^* < t_1);(t_2 < t_*);(x_1 \neq x'_1);...;(x_n \neq x'_n);(x_1 = x'_1$ et ... et $x_n = x'_n$ et $v = v')\}$*

Un conflit de ressource est un ensemble de propositions d'utilisation de ressources qui risque de causer une surconsommation de la ressource étant donné sa capacité maximale.

Définition 3 (conflit de ressource)
Un conflit de ressource est un ensemble de propositions d'utilisation de ressources d'un même type res $\{u_1,u_2,...,u_k\}$ tel que (si $u_i = use(res(x_{i,1},...,x_{i,n}) : q_i,(t_i^-,t_i^+))$):

1. *aucune des contraintes $\{(t_i^+ < t_j^-)_{(i \neq j)}\}$ ne peut être déduite de $\mathcal{P}$ (i.e., les propositions peuvent s'intersecter globalement dans le temps);*
2. *il existe un n-uplet de constantes $(X_1,...,X_n)$ tel que la conjonction de contraintes sur les variables $(x_{i,j} = X_j)$ pour $(i,j) \in [1,k] \times [1,n]$ n'est pas incohérente avec $\mathcal{P}$; et*
3. *$\sum_{j=1}^{k} q_i > Q(res(X_1,...,X_n))$ où Q désigne la capacité maximale des ressources.*

A chaque défaut est associé une disjonction de **résolvantes** qui correspond aux différentes possibilités pour la résolution de ce défaut; Ces résolvantes consistent essentiellement en des insertions de tâches ou de contraintes sur le plan partiel courant.

Un plan partiel $\mathcal{P}$ sera dit **plan solution** ssi:

1. le réseau des contraintes temporelles est consistant;
2. le réseau des contraintes sur les variables atemporelles est consistant; et
3. il ne contient plus aucun défaut.

L'architecture du système de planification est présentée sur la figure 3. Un module de contrôle déroule l'algorithme général non-déterministe ci-dessous en faisant appel à trois modules d'analyse du plan, chacun étant chargé d'analyser un type particulier de défaut et de calculer les résolvantes associées à chacun de ces derniers.

Algorithme $planifier(\mathcal{P})$

1. **terminaison:** si $\mathcal{P}$ ne contient plus de défaut, retourner P
2. **analyse:** appel des 3 modules d'analyse:
   $DEFAUTS = sous\_buts(\mathcal{P}) \cup menaces(\mathcal{P}) \cup conflits\_ressources(\mathcal{P})$
3. **sélection**[2]**d'un défaut:** $\Phi \in DEFAUTS$;
4. **choix d'une résolvante:** $\rho \in resolvantes(\Phi)$;
5. **appel récursif:** $planifier(insertion(\mathcal{P},\rho))$

2.2 Les modules d'analyse du plan partiel

Analyse des sous-buts
Le mécanisme d'explication de sous-buts utilisé dans le planificateur IxTeT repose sur l'utilisation de liens causaux. C'est une extension de l'approche décrite dans [McAllester et Rosenblitt, 1991] au cadre temporel.

Étant donné une proposition non-expliquée, on distingue deux types de résolvantes: l'explication par un

[2]Suivant la terminologie définie dans [Weld, 1995] nous employons le mot *choix* pour désigner un choix non-déterministe et par conséquent un point possible de retour-arrière par opposition à *sélection* qui correspond à un choix heuristique sur lequel un retour-arrière ne sera pas nécessaire.

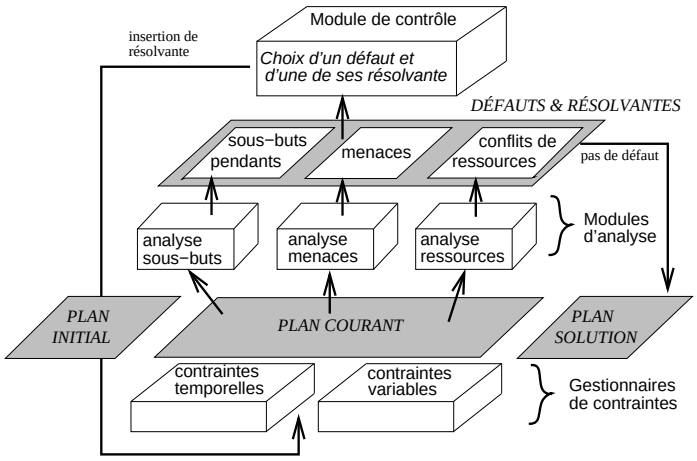

Figure 3: *Architecture générale du système*

événement déjà présent dans le plan partiel ou l'insertion d'une nouvelle tâche pour réaliser cette explication (c.f. figure 4).

Pour un sous-but donné, l'étape d'analyse construit toutes les résolvantes envisageables, c'est à dire:

- l'ensemble des événements du plan susceptibles d'expliquer le sous-but; et
- l'ensemble des événements appartenant à un modèle de tâche et qui seraient susceptibles d'expliquer le sous-but après insertion de la tâche dans le plan.

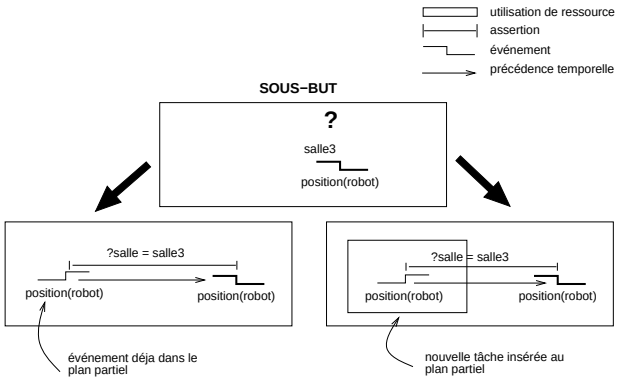

Figure 4: *Sous-but et deux de ses résolvantes*

Le choix de la "bonne tâche" pour expliquer un sous-but est un point très sensible du processus de planification. Dans IxTeT , une procédure particulière nommée *faisabilité* [Ghallab et Laruelle, 1994] permet d'estimer avec finesse l'intérêt relatif des différentes tâches pour résoudre un sous-but en effectuant un "look-ahead" qui permet d'analyser la difficulté des nouveaux sous-buts introduits par l'insertion de cette tâche. On peut considérer cette procédure comme une étape de planification approximative avec une profondeur bornée qui ignore les conflits possibles entre les différentes tâches. Le résultat de la faisabilité est une pondération des tâches utiles à la résolution d'un sous-but donné.

Si p est la profondeur de l'arbre ET/OU de faisabilité (paramètre de contrôle donné par l'opérateur), la complexité au pire cas de la faisabilité pour analyser l'ensemble des sous-buts du plan partiel est en $O(n.b^p.s^{p-1})$ si n désigne la taille du plan partiel (nombre d'assertions et d'événements), b le nombre maximal de tâches susceptibles d'expliquer un sous-but donné et s le nombre maximal de sous-buts supplémentaires apportés par l'insertion de la tâche (conditions de la tâche).

Analyse des menaces
Le module d'analyse des menaces sur les protections examine tous les couples $< e, cl >$ et $< h, cl >$ où e est un événement et h et cl sont des assertions. Dans les cas où ce couple est conflictuel, un défaut est calculé et ses résolvantes consistent en des contraintes temporelles ou des contraintes sur les variables (c.f. figure 5).

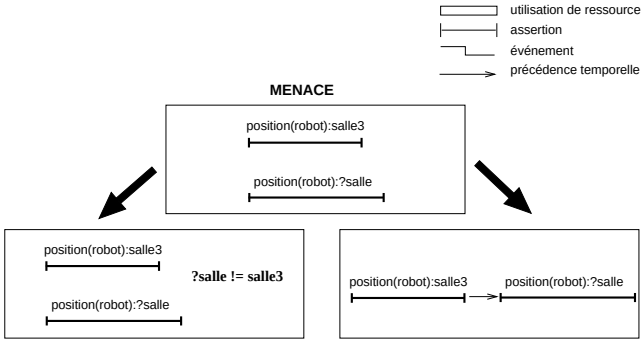

Figure 5: *Menace et deux de ses résolvantes*

La complexité de l'analyse complète des menaces sur un plan partiel est en $O(n^2)$.

Analyse des conflits de ressources potentiels
Nous nous intéressons ici uniquement aux types de ressources complètement agglomérés pour lesquels on ne différencie aucune sous-entité particulière (i.e. l'attribut de ressource n'a aucun argument). L'analyse des ressources individualisées et non-partageables s'effectue grâce à un algorithme du même type que celui de l'analyse des menaces et de même complexité quadratique que nous ne détaillerons pas ici.

La recherche des conflits de ressources sur un plan partiel utilise une représentation restreinte des contraintes temporelles du plan sous la forme de **graphe d'intersections possibles (g.i.p.)**.

Définition 4 (relation d'intersection possible)
Un sous-ensemble V de propositions d'emprunt de ressources (use) sera dit en intersection possible s'il existe au moins une instanciation des contraintes temporelles pour laquelle les propositions de V s'intersectent globalement dans le temps.

Un g.i.p. (fig. 6) est un graphe G=(U,E) non-orienté où:

- l'ensemble U des sommets est l'ensemble des propositions d'emprunt de ressources du plan partiel; et
- l'ensemble E des arêtes est l'ensemble des paires $\{u_i, u_j\}$ de propositions en intersection possible.

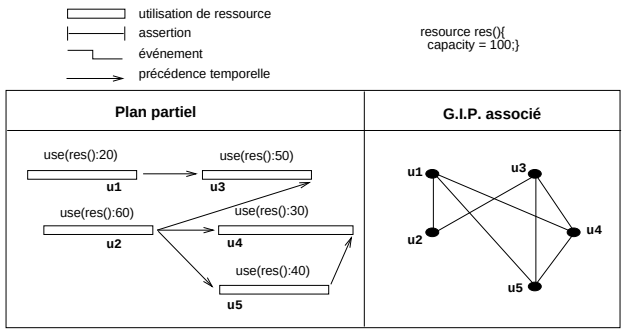

Figure 6: *Exemple de g.i.p.*

On montre alors le résultat suivant:

Théorème 1
V est un sous-ensemble de propositions de U en intersection possible ssi l'ensemble des propositions de V s'intersectent possiblement deux à deux.

Ce théorème est fondamental car il permet de formuler le problème de la recherche d'ensemble de propositions d'emprunt de ressources pouvant globalement s'intersecter dans le temps comme un problème de recherche de cliques sur un graphe particulier: le g.i.p.

De manière générale, le problème de la recherche de cliques particulières sur un graphe quelconque est un problème difficile. La propriété suivante permet toutefois de raccrocher notre problème à un corpus de plus en plus étudié de la théorie des graphes:

Propriété 1
Les graphes d'intersection possible sont des **graphes faiblement triangulés**, *c'est à dire qu'ils ne contiennent pas de cycle de longueur supérieure ou égale à 5 sans corde* [3].

Divers résultats en théorie des graphes montrent que certains problèmes de recherche de cliques particulières sur les graphes faiblement triangulés (clique maximale, clique maximale pondérée) sont de complexité polynomiale [Hayward et al., 1989].

Dans notre approche, nous recherchons les **ensembles critiques minimaux** (e.c.m.) qui sont exactement les conflits de ressources minimaux au sens de l'inclusion ensembliste (par exemple, sur la figure 6, le conflit de ressource $\{u_3, u_4, u_5\}$ est minimal puisque qu'aucun de ses sous-ensembles strict n'est surconsommateur de la ressource dont la capacité maximale est 100). Nous avons développé un algorithme, plus particulièrement détaillé dans [Laborie, 1994], basé sur la recherche de cliques maximales sur les graphes triangulés [Gavril, 1972] et qui permet de trouver de manière relativement efficace les e.c.m. de taille minimale du plan partiel, c'est à dire ceux qui sont les plus prioritaires dans le cadre d'une stratégie de moindre engagement (par exemple, sur la figure 6, il n'y a qu'un seul e.c.m. de taille minimale: $\{u_2, u_3\}$).

La complexité au pire cas de l'algorithme pour la recherche de l'ensemble des e.c.m. de taille minimale est en $O(n^2.k.\alpha^{k-1})$ si α désigne le degré de parallélisme maximal du plan (nombre maximal de proposition d'emprunt d'une même ressource partageable pouvant avoir lieu simultanément) et k la taille des e.c.m. de taille minimale (en pratique, k dépasse rarement 3 ou 4).

Après la recherche des e.c.m. de taille minimale, la deuxième phase de l'analyse des ressources consiste à calculer les résolvantes pour un e.c.m. donné (c.f. figure 7). Ces résolvantes sont de deux types:

1. l'insertion d'une contrainte de précédence temporelle entre un couple de propositions de l'e.c.m. ; et
2. dans le cas où il existe des tâches capables de produire cette ressource, l'insertion d'une telle tâche avant l'une quelconque des propositions de l'e.c.m.

Une étape de réduction des contraintes temporelles permet d'éliminer certaines des résolvantes surcontraignantes sans remettre en cause la complétude de la recherche [Laborie, 1994]. Ainsi, dans l'exemple ci-dessus, la résolvante $u_4 < u_3$ peut être éliminée car elle contient la contrainte $u_5 < u_3$.

2.3 Contrôle de la planification et gestion des choix non-déterministes

L'algorithme général présente deux points de décisions essentiels quant au parcours correct de l'espace de recherche et aux performances du système: la sélection d'un défaut et le choix d'une de ses résolvantes.

Pour un défaut donné, le choix d'une de ses résolvantes s'effectue selon une stratégie de **moindre engagement**. La sélection d'un défaut est basée sur une partition de

[3] une corde est une arête joignant deux sommets non-consécutifs dans un cycle

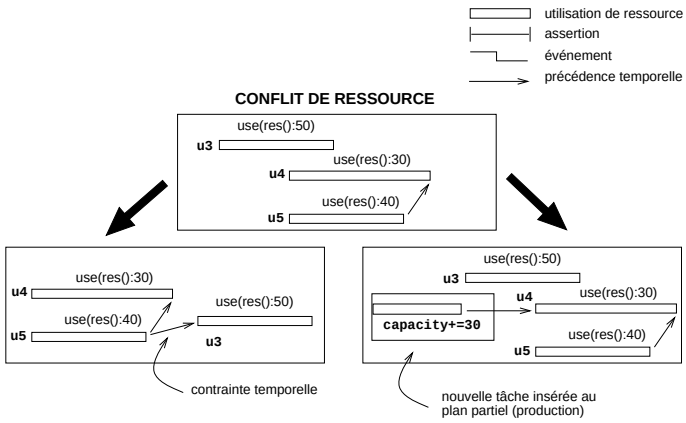

Figure 7: *Conflit de ressource et deux de ses résolvantes*

ceux-ci en différents **niveaux d'abstraction** associée à la notion d'**opportunité** de la résolution d'un défaut dans un contexte donné.

La stratégie de moindre engagement

A un noeud donné de l'arbre de recherche, un plan partiel $\mathcal{P}$ représente un ensemble (en général infini) d'instances possibles $INST(\mathcal{P})$: toutes les instanciations des instants et des variables compatibles avec les contraintes posées. L'insertion d'une contrainte ρ (sous la forme d'une résolvante) sur le plan partiel va éliminer un certain nombre d'instances possibles de sorte que $INST(insertion(\mathcal{P}, \rho)) \subset INST(\mathcal{P})$. Si l'on veut conserver un plan partiel le moins contraignant possible afin de pouvoir, ultérieurement dans la recherche, faire face à un éventail plus large de situations sans avoir à revenir en arrière, on a intérêt à choisir d'insérer en priorité les résolvantes qui éliminent le minimum d'instances représentées par ce plan partiel. L'engagement lié à l'insertion d'une résolvante ρ sur un plan partiel $\mathcal{P}$ est calculé comme une estimation du taux d'instances de $\mathcal{P}$ éliminé par l'insertion de ρ:

$$engagt(\mathcal{P}, \rho)) = 1 - \frac{card(INST(insertion(\mathcal{P}, \{\rho\})))}{card(INST(\mathcal{P}))}$$

Il est clair que $engagt(\mathcal{P}, \rho)) = 0$ ssi la résolvante ρ est redondante par rapport aux contraintes déjà présentes dans $\mathcal{P}$ et $engagt(\mathcal{P}, \rho)) = 1$ ssi ρ est inconsistante avec les contraintes déjà présentes dans $\mathcal{P}$.

Pour une contrainte ρ donnée, l'engagement lié à son insertion dans le plan partiel est estimé de manière locale en fonction du domaine courant des variables sur lesquelles porte la contrainte.

Ainsi pour une contrainte temporelle entre deux instants t et t', si sur $\mathcal{P}$ la propagation des contraintes numériques impose $(t' - t) \in [d_{min}, d_{max}]$, on estimera l'engagement lié à l'insertion de la contrainte $(t' < t)$ sur $\mathcal{P}$ en supposant, dans $INST(\mathcal{P})$ une répartition uniforme de la valeur de $(t' - t)$ sur $[d_{min}, d_{max}]$, ce qui donne:

$$engagt(\mathcal{P}, (t' < t)) = \frac{min(d_{max}, 0) - min(d_{min}, 0)}{d_{max} - d_{min}}$$

L'engagement lié à l'insertion de contraintes sur les variables ou de liens causaux est estimé par des fonctions du même type. L'engagement lié à l'insertion d'une tâche est plus délicat à quantifier; c'est là un des rôles de la procédure de faisabilité présentée plus haut (§2.2).

Étant donné un défaut, le système choisit toujours d'insérer en priorité sa résolvante qui minimise l'engagement.

Une recherche hiérarchisée

L'approche choisie dans IxTeT pour supporter une expressivité extrêmement riche est d'avancer très prudemment à chaque noeud de l'arbre de recherche; ceci explique l'analyse du plan relativement sophistiquée qui a été présentée plus haut (§2.2). Dans des problèmes de taille réaliste, toutefois, on ne peux pas se permettre d'examiner, à un noeud de l'arbre de recherche, l'ensemble de tous les défauts du plan partiel. Une approche de structuration et de hiérarchisation de l'espace de recherche à été implémentée; cette approche est davantage détaillée dans [Garcia et Laborie, 1995].

Il s'agit de gérer une hiérarchie d'abstraction, c'est à dire une affectation d'un niveau d'abstraction $niv(att)$ à chacun des attributs du domaine att (qu'il s'agisse d'attributs d'états ou d'attributs de ressources) de manière à ce que la propriété suivante soit vérifiée:

Propriété 2 (monotonicité)
Quelque soit le plan partiel, la résolution d'un défaut sur un attribut de niveau d'abstraction i ne crée jamais de nouveau défaut sur un attribut de niveau d'abstraction j avec $j > i$.

L'intérêt de cette propriété à été clairement démontré dans le cadre du formalisme STRIPS [Knoblock, 1990] [Yang et Tenenberg, 1990]; une fois trouvée une hiérarchie d'abstraction qui vérifie la propriété de monotonicité, celle-ci est exploitée en ne gérant, à un niveau d'abstraction donné, i, que les défauts portant sur des attributs de niveau i et en descendant au niveau d'abstraction $i-1$ dès qu'il n'existe plus de défaut au niveau i.

Dans le planificateur IxTeT , à la manière d'Alpine [Knoblock, 1994], une hiérarchie d'abstraction vérifiant la propriété de monotonicité est générée hors-ligne en analysant la syntaxe des tâches du domaine et en exploitant l'idée que les seuls défauts dont la résolution peut entraîner l'apparition de nouveaux défauts sont les sous-buts non-expliqués lorsqu'ils sont résolus par une insertion de tâche.

On peut alors aisément se convaincre que toutes hiérarchies vérifiant la propriété ci-dessous vérifient aussi la propriété de monotonicité:

Propriété 3 (treillis d'abstraction)
Quelle que soit la tâche T du domaine, si e est un événement ou une proposition de production de ressource qui peut justifier l'insertion de T dans un plan partiel (effet principal) et si p est une proposition temporelle quelconque (événement, assertion, production, consommation ou emprunt de ressource), $niv(att_p) < niv(att_e)$.

Le balayage hors-ligne de l'ensemble des tâches permet de générer un ordre partiel sur les attributs, une hiérarchie partielle comme celle de la figure 8 pour l'exemple du robot chargé de la maintenance d'un laboratoire.

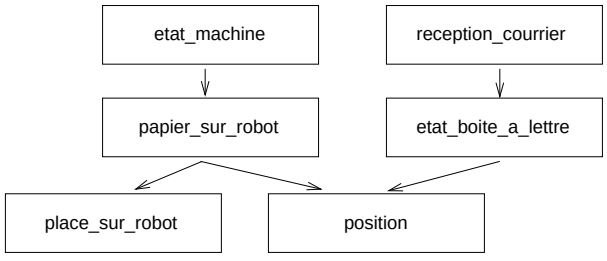

Figure 8: *Exemple de treillis d'abstraction*

Une des principales originalités de l'approche hiérarchique implémentée dans IxTeT est que, contrairement à ce qui est fait classiquement dans les autres planificateurs, le système ne s'engage pas hors-ligne à linéariser le treillis d'abstraction de façon à obtenir un ordre total sur niveaux des attributs. Au contraire, cette linéarisation à lieu en cours de planification, de manière dynamique en fonction du contexte. Lorsqu'un attribut ne présente plus de défaut dans le plan partiel courant, il est considéré comme résolu et tous les défauts qui attendaient sa résolution sont maintenant considérés. Cette approche permet en particulier d'avancer plus vite le long de certaines branches parallèles du treillis d'abstraction dont la résolution est "facile" et au contraire d'attendre d'avoir un contexte plus complet avant de résoudre certain autres défauts. Un exemple d'avancement dans le treillis en cours de planification est donné sur la figure 9.

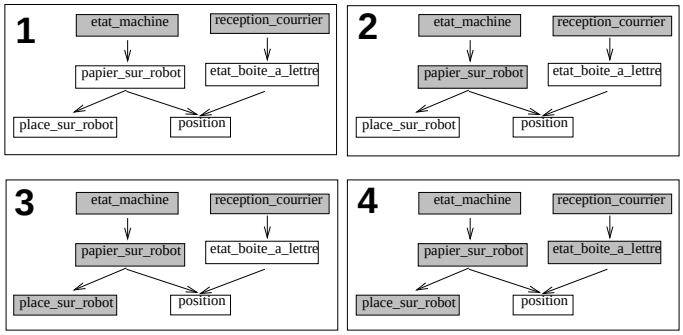

Figure 9: *Exploitation du treillis en cours de planification*

La hiérarchisation de l'espace de recherche permet d'améliorer les performances du système pour deux raisons:

1. seulement un sous-ensemble des défauts est examiné à chaque noeud de l'arbre de recherche: les défauts qui sont les "plus abstraits" au sens de l'ordre partiel sur les attributs; et
2. la structuration de l'espace de recherche permet de résoudre en priorité les défauts les plus significatifs et d'éviter de s'engager dans des choix qui relèvent plus du détail et qui pourraient être remis en cause par la suite; ceci permet de réduire les causes de retour-arrière.

Une recherche opportuniste

A un niveau d'abstraction donné, en cours de planification, il reste toutefois une sélection à faire quant au défaut à résoudre. Nous effectuons cette sélection de manière opportuniste en essayant de minimiser les chances de retour-arrière. Une des originalités essentielles du système est que, à un noeud donné de l'arbre de recherche, les défauts sont sélectionnés indépendamment de leur type (résolution de sous-but, de menace ou de conflit de ressource). Il n'y a pas, comme dans SNLP [McAllester et Rosenblitt, 1991], d'ordre posé a priori en fonction du type des défauts (SNLP gère toujours les menaces en priorité avant de gérer l'explication des sous-buts): dans IxTeT , les défauts sont gérés, quelle que soit leur origine, en fonction de l'opportunité de leur résolution. Pour un défaut donné Φ, un critère $K(\Phi)$ permet d'estimer la facilité qu'il y aura à faire un choix parmi les différentes résolvantes de Φ dans le cadre la stratégie de moindre engagement; nous sélectionnons en priorité les défauts qui maximisent cette facilité. Si le défaut Φ admet les résolvantes $\{\rho_1, ..., \rho_k\}$ et si ρ_{min} est la meilleure résolvante, $K(\Phi)$ est estime dans quelle mesure l'engagement lié à l'insertion ρ_{min} est largement intéressant par rapport aux autres résolvantes de Φ:

$$\frac{1}{K(\phi)} = \sum_{i=1}^{k} \frac{1}{1 + engagt(\mathcal{P}, \rho_i) - engagt(\mathcal{P}, \rho_{min})}$$

Il est à noter que l'on a toujours $0 < K(\phi) \leq 1$. $K(\phi) = 1$ ssi ϕ possède une seule résolvante, on dit dans ce cas que Φ est un défaut **déterministe** dans la mesure ou il n'y aura aucun choix à effectuer. Les défauts déterministes sont bien entendu résolus en priorité. De manière générale, le critère K permet de sélectionner en priorité les défauts qui ont peu de résolvantes, ceci étant modulé par la valeur de l'engagement lié à chacune des résolvantes.

Une recherche complète
L'arbre de recherche global est exploré grâce à un algorithme A_ϵ [Ghallab et Allard, 1983]. Notre objectif pour la recherche d'une solution est double: trouver une solution en un minimum de temps de calcul et trouver une solution de "bonne" qualité. Ces deux points ne sont toutefois pas incompatibles: dans le cadre d'une stratégie de moindre engagement (visant à améliorer les performances du système), les résolvantes choisies en priorité sont celles qui surcontraignent le moins le plan partiel et, ce faisant, on aboutit à une solution peu contrainte et très souple dans le sens où elle représente une très grande variété d'instances possibles, et ceci correspond bien, en général, à la conception intuitive d'un "bon" plan.

L'estimation d'un noeud (f) correspond à la somme entre l'engagement lié à l'insertion de toutes les résolvantes depuis le noeud racine (g) et une estimation heuristique (h) de l'engagement minimal qu'il resterait à effectuer sur le plan partiel pour arriver à une solution. Cette heuristique h est calculée en cumulant l'engagement lié aux meilleures résolvantes des défauts non-résolus du plan partiel. Si les défauts étaient indépendants les uns des autres, h serait une estimation exacte. En pratique, les défauts sont largement interdépendants et h n'est donc qu'une estimation; d'autre part, du fait des interactions positives entre les résolvantes (une même résolvante peut très bien résoudre plusieurs défauts), h n'est pas nécessairement une heuristique minorante. On constate toutefois, de manière empirique, que cette heuristique est relativement efficace pour guider l'exploration de l'arbre de recherche.

Une caractéristique essentielle de notre système est sa **complétude**: s'il existe une solution au problème posé, alors le planificateur trouvera un plan solution et s'il n'en existe pas, il devra parcourir tout l'arbre de recherche avant de donner une réponse négative. A noter toutefois que, du fait de l'utilisation de variables (temporelles ou atemporelles) à domaine discret, l'arbre de recherche est de taille finie.

3 Exemples

Grâce à sa riche expressivité, le planificateur IxTeT s'adresse à une très large gamme d'applications allant des problèmes classiques d'ordonnancement (job-shop) jusqu'au problèmes de synthèse de plan d'actions (en robotique autonome par exemple).

3.1 Synthèse de plan avec contraintes de ressources

Nous donnons sur la figure 10 un court exemple de plan solution dans le domaine du robot chargé de la maintenance d'un laboratoire. Il s'agit d'un problème de synthèse de plan: le problème initial ne spécifie que les buts à atteindre, ici, le fait que le courrier doit être délivré dans certaines salles dès que le facteur sera passé et que certaines machines devront être alimentées en papier. Les barres horizontales représentent les différentes tâches planifiées avec une longueur proportionnelle à leur durée moyenne. Les flèches représentent les contraintes temporelles de précédence. Pour ce qui est de la gestion de ressources, on remarquera la présence de la tâche *charge_papier* qui permet au robot de faire le plein de papier dans une salle particulière afin de pouvoir ultérieurement alimenter les machines. La synthèse de ce plan demande environ 1s de calcul sur une station Sparc10.

Le planificateur IxTeT a été testé et validé dans différents autres domaines comme la planification d'activités sur un chantier de construction ou pour des travaux de finition d'une pièce, pour la gestion d'expériences de biologie embarquées à bord d'une station spatiale, pour la planification des activités d'un robot d'exploration planétaire, etc ...

3.2 Ordonnancement avec gammes alternatives

Soit le problème suivant où il s'agit de produire cinq éléments A, B, C, D et E à l'aide de trois machines M1, M2 et M3 non-partageables. Chaque élément peut être produit de deux manières différentes selon l'ordre dans lequel il utilise les machines. Suivant la nature de l'opération à effectuer, la durée pendant laquelle une machine est utilisée est variable.

élément A: M2(d=3), M1(d=3), M3(d=6) ou
M2(d=3), M3(d=6), M1(d=3)
élément B: M2(d=2), M1(d=5), M2(d=2), M3(d=7) ou
M2(d=2), M3(d=7), M2(d=2), M1(d=5)
élément C: M1(d=7), M3(d=5), M2(d=3) ou
M3(d=5), M1(d=7), M2(d=3)
élément D: M2(d=4), M3(d=6), M1(d=7), M2(d=4) ou
M2(d=4), M3(d=6), M2(d=4), M1(d=7)
élément E: M2(d=6), M3(d=2) ou
M3(d=2), M2(d=6)

Ce problème est facilement modélisable avec IxTeT en définissant une tâche par gamme alternative dont l'effet est la production de l'élément et en spécifiant comme buts que tous les éléments doivent être produits. On montre alors, en contraignant peu à peu la durée maximale du plan, que le plan donné sur la figure 11 (généré par IxTeT en 3s environ) est celui qui minimise la durée totale avec une durée minimale de 26; pour une durée strictement inférieure à 26, le planificateur explore tout l'arbre de recherche en 30s environ sans trouver de solution.

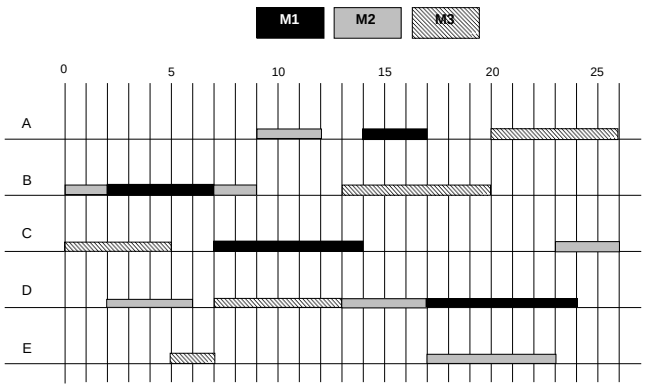

Figure 11: *Exemple de plan solution*

Conclusion et perspectives

L'expressivité du formalisme IxTeT et l'approche algorithmique qui le supporte permettent au système décrit dans cet article de résoudre un grand éventail de problèmes de planification s'étendant entre les problèmes d'ordonnancement et ceux de synthèse de plan. Plusieurs prolongements de ces travaux sont en cours que l'on peut classer en trois axes principaux:

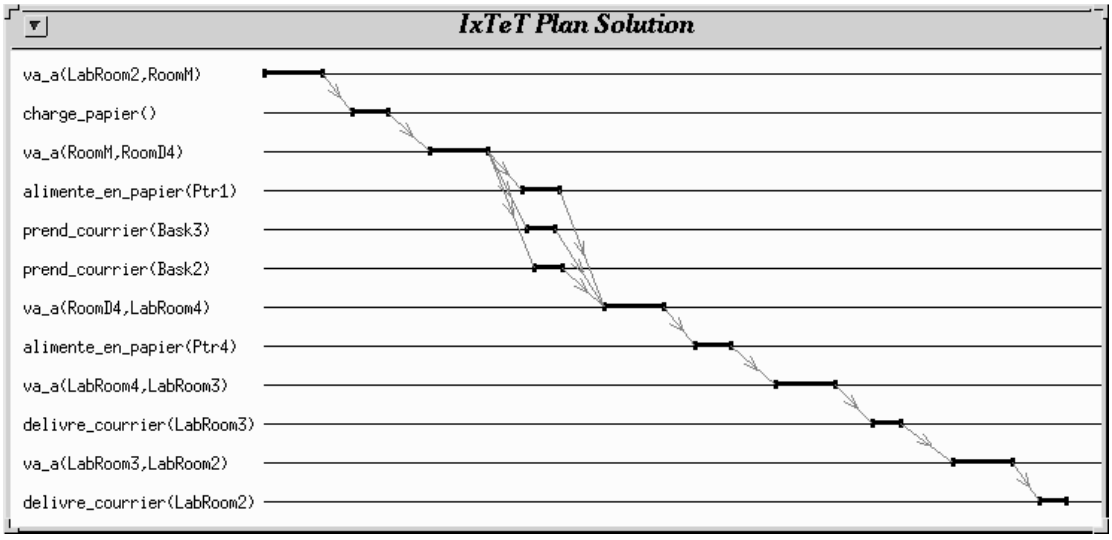

Figure 10: *Exemple de plan solution*

1. **améliorer les performances du système et la qualité de la solution produite:** le premier point passe par l'utilisation d'algorithmes spécifiques pour gérer certains sous-problèmes du problème général (par exemple lorsque l'on peut se ramener à la recherche d'un chemin sur un graphe d'états) et le second par la recherche d'un compromis entre la stratégie de moindre engagement et la qualité de la solution trouvée lorsque ces deux objectifs divergent;
2. **ouvrir le système sur son environnement** en intégrant le planificateur à un système de contrôle d'exécution du plan et en introduisant la possibilité d'une planification incrémentale; et
3. **enrichir l'expressivité du formalisme**, en introduisant par exemple des axiomes du domaine pour compléter la description du scénario initial ou les modèles de tâches.

Bibliographie

Chapman, D. (1987). Planning for conjunctive goals. *Artificial Intelligence*, 32:333–377.

Currie, K. et Tate, A. (1991). O-plan: the open planning architecture. *Artificial Intelligence*, 52:49–86.

Drabble, B. (1995). O-plan project second year demonstration: Reasonning with resources. Technical Report ARPA-RL/O-Plan/TR/19, Artificial Intelligence Application Institute, University of Edimburg.

Garcia, F. et Laborie, P. (1995). Hierarchisation of the search space in temporal planning. Proceedings EWSP-95.

Gavril, F. (1972). Algorithms for maximum coloring, maximum clique, minimum covering by cliques and maximum independant set of a chordal graph. *SIAM J. Comput.*, 1:180–187.

Ghallab, M. et Allard, D. (1983). A_ε: an efficient near admissible heuristic search algorithm. Dans *Proceedings IJCAI-83*.

Ghallab, M. et Laruelle, H. (1994). Representation and Control in Ixtet, a Temporal Planner. Dans *Proceedings AIPS-94*, pages 61–67.

Ghallab, M. et Mounir-Alaoui, A. (1989). Managing Efficiently Temporal Relations Through Indexed Spanning Trees. Dans *Proceedings IJCAI-89*.

Ghallab, M. et Vidal, T. (1995a). Focusing on a Subgraph for Managing Efficiently Numerical Temporal Constraints. Dans *Proceedings FLAIRS-95 (to appear)*.

Ghallab, M. et Vidal, T. (1995b). Temporal constraint in planning: Free or not free ? Dans *FLAIRS-95 Workshop on Constraint Reasonning (to appear)*.

Hayward, R., Hoang, C., et Maffray, F. (1989). Optimizing Weakly Triangulated Graphs. *Graphs and Combinatorics*, 5(4):339–350.

Knoblock, C. (1990). Learning abstraction hierarchies for problem solving. Dans *Proceedings AAAI-90*, pages 923–928.

Knoblock, C. (1994). Automatically generating abstractions for planning. *Artificial Intelligence*, 68:243–302.

Knoblock, C., Tenenberg, J., et Yang, Q. (1991). Characterizing Abstraction Hierarchies for Planning. Dans *Proceedings AAAI-91*, pages 692–697.

Laborie, P. (1994). Planifier avec des contraintes de ressources. Technical Report 94077, LAAS-CNRS, Toulouse (France).

Laborie, P. et Ghallab, M. (1995). Planning with Sharable Resource Constraints. Dans *Proceedings IJCAI-95*, pages 1643–1649.

McAllester, D. et Rosenblitt, D. (1991). Systematic nonlinear planning. Dans *Proceedings AAAI-91*, pages 634–639.

Penberthy, J. et Weld, D. (1994). Temporal planning with continual change. Dans *Proceedings AAAI-94*.

Weld, D. (1995). An introduction to least commitment planning. *AI Magazine*, pages 27–61.

Wilkins, D. (1988). *Practical Planning*. Morgan Kaufmann, San-Mateo, CA.

Yang, Q. et Tenenberg, J. (1990). ABTWEAK : Abstracting a Nonlinear Least Commitment Planner. Dans *Proceedings AAAI-90*, volume 1, pages 204–209.