Francis BISSON (06 794 819)
Kenny CÔTÉ (06 836 427)
Pierre-Luc ROGER (06 801 883)

PLANIFICATION DE TÂCHES DANS MS PROJECT

IFT702 – Planification en intelligence artificielle

Présenté à
M. Froduald KABANZA
Département d'informatique
Université de Sherbrooke

2009-04-20

Table des matières

Partie 1

Présentation du problème

Depuis déjà plusieurs années, les gestionnaires de projets se servent d'outils informatiques afin d'ordonner dans le temps une séquence d'assignations de travail. Dans le milieu des gestionnaires de projets, le terme *assignation* signifie l'association à un temps précis d'un employé (ou *ressource*) à une tâche à accomplir dans le cadre du projet. Non seulement cette tâche doit tomber dans le champ de compétence de la ressource, mais le moment auquel elle lui est assignée doit aussi correspondre à son horaire de travail.

1.1 L'outil de planification MS Project

Dans le cadre de notre projet, nous nous sommes intéressés à l'outil informatique Microsoft Office Project 2007, un des plus utilisés sur le marché. Ce logiciel de planification permet à un gestionnaire d'établir les tâches à accomplir pour la réalisation de son projet ainsi que les ressources à sa disposition. Il y est possible de personnaliser entièrement l'horaire de travail de chaque ressource, en plus de pouvoir déterminer des contraintes de précédence entre les tâches et une date limite pour l'accomplissement de chacune.

Quatre types de contraintes de précédence peuvent s'appliquer entre deux tâches : *finish-to-start* impose que la première tâche soit terminée avant que la seconde ne soit commencée, *start-to-finish* étant son contraire sémantique ; *start-to-start* et *finish-to-finish* contraignent les deux tâches à commencer ou terminer, respectivement, au même moment.

1.2 La problématique

Une fois qu'il a déterminé tous ces facteurs, l'utilisateur de MS Project peut visualiser son projet à l'aide d'un diagramme de Gantt généré automatiquement par le programme. Ce type de diagramme représente les tâches de façon séquentielle dans le temps, tout en affichant les contraintes de précédence et de succession entre elles.

Cependant, l'algorithme utilisé par MS Project pour générer son diagramme de Gantt ne donne pas nécessairement une planification valide, et encore moins optimale. En effet, il ne fait qu'enchaîner les tâches selon leur contraintes de précédence, leur date de début et leur date limite de fin. Une ressource peut ainsi être assignée sur plusieurs tâches au même moment, chacune obtenant le montant total d'heures consacrées par la ressource, ce qui rend invalide la planification obtenue.

1.3 L'approche

Afin de résoudre ce problème et obtenir un agencement valide d'assignations de ressources aux tâches avec Microsoft Project, nous avons réalisé une application Windows qui prend en entrée un fichier .mpp contenant déjà les ressources et les tâches à accomplir d'un projet, accompagnées de toutes leurs contraintes.

Le programme utilise ensuite l'approche des *Constraint Satisfaction Problems* (CSP) pour représenter le problème de planification en tant qu'un ensemble de variables et de contraintes. Enfin, il utilise un algorithme de satisfaction de contraintes pour obtenir l'ensemble d'assignations final avec lequel le fichier Microsoft Project sera mis à jour. En ouvrant par la suite le fichier .mpp dans MS Project, le gestionnaire de projets aura alors sous la main un diagramme de Gantt supérieur à celui généré automatiquement par le programme.

1.4 L'organisation du rapport

Les prochaines pages expliqueront tout d'abord comment utiliser notre application, puis détailleront les techniques utilisées pour implanter l'approche CSP dans le cadre d'une planification de projet Microsoft Project. Ensuite, nous présenterons quelques résultats d'assignations obtenues par l'utilisation de notre programme sur deux projets de complexité différente, avant d'analyser la validité et l'optimalité de chacun. Pour terminer, nous discuterons des avenues possibles pour perfectionner notre application et lui donner un fini plus professionnel.

Partie 2

Démarrage de l'application

2.1 Prérequis

Le logiciel MS Project doit tout d'abord être installé sur le poste où l'application doit être démarrée. La version 2007 de Microsoft Project a été utilisée lors du développement du projet ; nous ne pouvons donc pas garantir son fonctionnement avec d'autres versions du logiciel.

La plateforme d'exécution .NET Framework 3.5 de Microsoft doit être installée afin de pouvoir exécuter le projet. Un programme d'installation est fournis sur le CD-ROM sous le nom dotNetFx35setup.exe dans le dossier *bin* et peut aussi être téléchargé depuis plusieurs sites Internet.

2.2 Étapes de l'exécution

1. Lancez le fichier exécutable MSProjectScheduler.exe.
2. Cliquez sur le bouton *Ouvrir*.
3. Sélectionnez un fichier Microsoft Project d'extension .mpp contenant déjà les ressources et les tâches à accomplir.
4. Cliquez sur le bouton *Ouvrir* ; l'application calculera alors les assignations et mettra à jour le fichier .mpp.
5. Cliquez sur le bouton *Ouvrir* pour sélectionner un autre fichier .mpp, ou quittez l'application MSProjectScheduler.
6. Ouvrir le fichier .mpp mis à jour dans Microsoft Project pour visualiser le résultat de la planification.

Partie 3

Stratégie d'implémentation

3.1 Intégration avec MS Project

Afin de réaliser ce projet, il était essentiel de pouvoir accéder aux structures de données contenues dans un fichier de projet .mpp. Cet accès est grandement simplifié par l'utilisation de la librairie Microsoft.Office.Interop.MSProject de la plateforme .NET Framework, permettant l'interopérabilité entre les produits Microsoft. Avec cette librairie, il est donc possible d'appeler une méthode pour lire un fichier .mpp et d'obtenir un objet correspondant à l'arborescence du projet. Cependant, il n'est possible de s'en servir qu'en utilisant des langages de programmation de Microsoft ; aussi avons-nous programmé le projet en C# pour bénéficier de la librairie.

3.2 Algorithme de satisfaction de contraintes

L'algorithme de satisfaction de contraintes que nous avons implanté dans notre application est le *backtrack-search* [2]. Il s'agit d'un algorithme de recherche semblable à la recherche en profondeur dans un graphe. L'algorithme prend en entrée un tuple $\Omega = (V, D, C)$ et une assignation partielle σ (consistante par rapport aux contraintes C) contenant des couples (v_i, d_j) avec $v_i \in V$, une variable et $d_j \in D_i$, une valeur dans le domaine de la variable v_i.

Lors d'un appel récursif à *backtrack-search*, une variable parmi toutes celles qui n'ont pas encore été assignées est choisie. Le domaine de valeurs possibles de cette variable est ensuite parcouru jusqu'à ce qu'une valeur consistante, c'est-à-dire une valeur qui respecte toutes les contraintes, soit trouvée. Cette valeur est assignée à la variable sélectionnée et la fonction est appelée récursivement, afin de trouver une assignation valide pour la variable suivante. Si, pour une variable donnée, aucune assignation consistante n'est trouvée, l'algorithme retourne sur ses pas (d'où le nom *backtrack*) et une nouvelle valeur est sélectionnée pour la variable précédente.

L'algorithme se termine soit avec une assignation consistante de valeurs pour chaque variable du problème CSP, ou en déterminant qu'il est impossible d'obtenir une assignation complète respectant toutes les contraintes.

3.3 Modélisation du problème

Notre problématique, quoiqu'adaptée à l'utilisation d'une approche CSP, requiert tout de même plusieurs ajustements afin que nous soyons capables de modéliser notre problème en variables et en contraintes utilisables par un algorithme de type *backtrack-search*.

3.3.1 Variables

Tout d'abord, il est important de noter qu'un projet comporte une dimension additionnelle qui n'est pas abordée dans l'approche CSP de base : le temps. En effet, en plus d'avoir des ressources pouvant être affectées à certaines tâches, le moment et l'ordre dans lesquelles ces assignations sont faites comportent aussi des contraintes. Puisqu'une tâche peut être accomplie par l'effort conjoint de ressources différentes et parfois au même moment, il est impensable de modéliser une tâche comme une variable ; une seule valeur doit être choisie parmi le domaine d'une variable dans un problème de satisfaction de contraintes.

La solution s'imposant alors est la modélisation des ressources comme des variables ; toutes les tâches que la ressource peut accomplir composent le domaine de valeurs possibles. Nous avons pris la décision de regrouper le temps avec les ressources pour former l'ensemble de nos variables, parce que nous disposons déjà *a priori* des horaires de travail des ressources dans le fichier .mpp. Par exemple, si une ressouce X (possédant l'ID 1 dans MS Project) travaillait du lundi au mardi en journée, une ressource Y (possédant l'ID 2 dans MS Project) travaillait le soir du mercredi au vendredi et que le projet devait être accompli la semaine du 13 avril 2009, nous aurions comme variables :

- 1 ;13/04/2009 ;am
- 1 ;13/04/2009 ;pm
- 1 ;14/04/2009 ;am
- 1 ;14/04/2009 ;pm
- 2 ;15/04/2009 ;evening
- 2 ;16/04/2009 ;evening
- 2 ;17/04/2009 ;evening

Pour terminer, chacune de ces variables aurait comme domaine l'ensemble des tâches que X (ou Y) peut accomplir.

3.3.2 Contraintes

Au lieu d'être sur les variables, les contraintes agissent sur les valeurs des variables, c'est-à-dire les tâches à accomplir dans un projet. Cette façon de procéder offre un avantage simple, soit la réduction du nombre de contraintes dans le problème CSP. En effet, une représentation classique des contraintes nous forcerait à créer une contrainte entre chaque paire de variables « ressource – date – plage horaire ». Un nombre aussi grand de contraintes à valider à chaque itération de l'algorithme *backtrack-search* induit un temps de calcul non-négligeable.

3.3.3 *Backtrack-search*

Une première modification, qui est en fait une optimisation, que nous avons apporté à l'algorithme *backtrack-search* original consiste à ignorer, lors du parcours des valeurs du domaine d'une variable, les tâches qui sont déjà complétées. Ces tâches sont celles pour lesquelles les ressources ont investi au total un nombre d'heures égal à la durée estimée de ces tâches. Cette technique permet de détecter préemptivement une inconsistance dans l'assignation des ressources aux tâches.

Nous avons aussi modifié la condition de fin de l'algorithme afin de tenir compte de la complétion des tâches. Lorsque toutes les tâches sont complétées, c'est-à-dire que les ressources ont investi une quantité suffisante d'heures sur chacune des tâches, l'algorithme termine en retournant la solution. Les variables qui n'ont pas encore obtenu de valeur sont simplement ignorées ; ces plages horaires correspondent aux périodes où les ressources ne travaillent pas.

Partie 4

Expérimentations et résultats

Pour effectuer nos tests, nous avons utilisé une mise en situation du mémoire de Guy Brault [1], puisque nous avions déjà accès à la solution. Pour des raisons de validations de code, nous avons élaboré une version simplifiée de la mise en situation normale.

4.1 Mise en situation simplifiée

Task Name	Duration	Start	Finish	Predecessors	Resource Names
Project	**3 days**	**Tue 09-03-31**	**Thu 09-04-02**		
Analyse	**1,5 days**	**Tue 09-03-31**	**Wed 09-04-01**		
T4	12 hrs	Tue 09-03-31	Wed 09-04-01		Resource1
T5	4 hrs	Tue 09-03-31	Tue 09-03-31		Resource1
T6	12 hrs	Tue 09-03-31	Wed 09-04-01		Resource1
Conception	**2,5 days**	**Tue 09-03-31**	**Thu 09-04-02**		
T7	8 hrs	Wed 09-04-01	Thu 09-04-02	3	Resource2
T8	16 hrs	Tue 09-03-31	Thu 09-04-02	4	Resource2
T9	12 hrs	Wed 09-04-01	Thu 09-04-02	5	Resource2

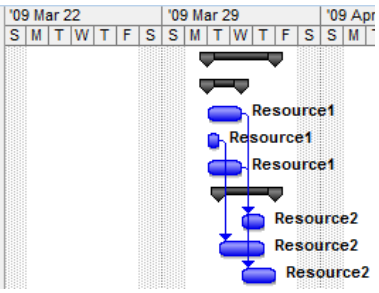


FIG. 4.1 – Diagramme de Gantt de la mise en situation simplifiée avant planification

La première mise en situation contient deux ressources qui doivent se séparer le travail de six tâches. La première ressource peut effectuer les trois premières tâches alors que la deuxième ressource doit s'occuper des trois dernières. Chacune des tâches de la deuxième ressource a une contrainte de précédence sur une tâche de la première ressource. Par exemple, la tâche T_7 doit être effectuée lorsque la tâche T_4 est terminée, ce qui veut dire que toutes les contraintes sont de type *finish-to-start*.

Notre application résoud ce problème en 27 itérations de *backtrack-search*, et présente comme résultat l'accomplissement de chaque tâche de la première ressource une à la suite de l'autre. La fin

Task Name	Duration	Start	Finish	Predecessors	Resource Names
Project	**5,5 days**	**Tue 09-03-31**	**Tue 09-04-07**		
Analyse	**3,5 days**	**Tue 09-03-31**	**Fri 09-04-03**		
T4	12 hrs	Tue 09-03-31	Wed 09-04-01		Resource1
T5	4 hrs	Wed 09-04-01	Wed 09-04-01		Resource1
T6	12 hrs	Thu 09-04-02	Fri 09-04-03		Resource1
Conception	**4 days**	**Wed 09-04-01**	**Tue 09-04-07**		
T7	8 hrs	Wed 09-04-01	Thu 09-04-02	3	Resource2
T8	16 hrs	Thu 09-04-02	Mon 09-04-06	4	Resource2
T9	12 hrs	Mon 09-04-06	Tue 09-04-07	5	Resource2

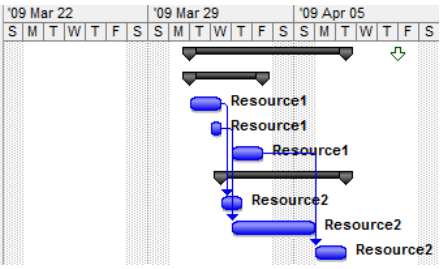


FIG. 4.2 – Diagramme de Gantt de la mise en situation simplifiée après planification

de chacune des tâches de la ressource 1 est suivie par le commencement de la tâche de la ressource 2 qui avait pour contrainte que cette première soit terminée. Le projet complet est accompli dans une période de 6 jours.

4.2 Mise en situation normale

La seconde mise en situation est plus complexe, car elle implique quatre ressources devant s'occuper de douze tâches. Comme dans la mise en situation simplifiée, nous retrouvons des contraintes de type *finish-to-start* entre chaque tâche d'une ressource et une tâche d'une autre ressource. La tâche T_4 doit donc être terminée avant de pouvoir commencer la tâche T_8, qui elle-même doit précéder la tâche T_{12} et ainsi de suite.

Task Name	Duration	Start	Finish	Predecessors	Resource Names
Project	**8 days**	**Tue 09-03-24**	**Thu 09-04-02**		
Analyse	**1,5 days**	**Tue 09-03-24**	**Wed 09-03-25**		
T4	12 hrs	Tue 09-03-24	Wed 09-03-25		Resource1
T5	4 hrs	Tue 09-03-24	Tue 09-03-24		Resource1
T6	12 hrs	Tue 09-03-24	Wed 09-03-25		Resource1
Conception	**2,5 days**	**Tue 09-03-24**	**Thu 09-03-26**		
T8	8 hrs	Wed 09-03-25	Thu 09-03-26	3	Resource2
T9	16 hrs	Tue 09-03-24	Thu 09-03-26	4	Resource2
T10	12 hrs	Wed 09-03-25	Thu 09-03-26	5	Resource2
Implantation	**2,5 days**	**Thu 09-03-26**	**Mon 09-03-30**		
T12	8 hrs	Thu 09-03-26	Fri 09-03-27	7	Resource3
T13	20 hrs	Thu 09-03-26	Mon 09-03-30	8	Resource3
T14	16 hrs	Fri 09-03-27	Mon 09-03-30	9	Resource3
Tests	**4,5 days**	**Fri 09-03-27**	**Thu 09-04-02**		
T16	36 hrs	Fri 09-03-27	Thu 09-04-02	11	Resource4
T17	20 hrs	Tue 09-03-31	Thu 09-04-02	12	Resource4
T18	16 hrs	Tue 09-03-31	Wed 09-04-01	13	Resource4

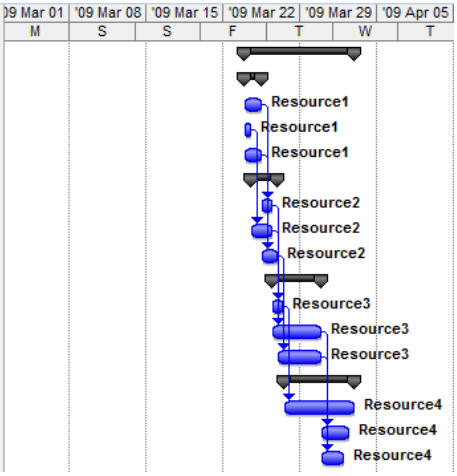
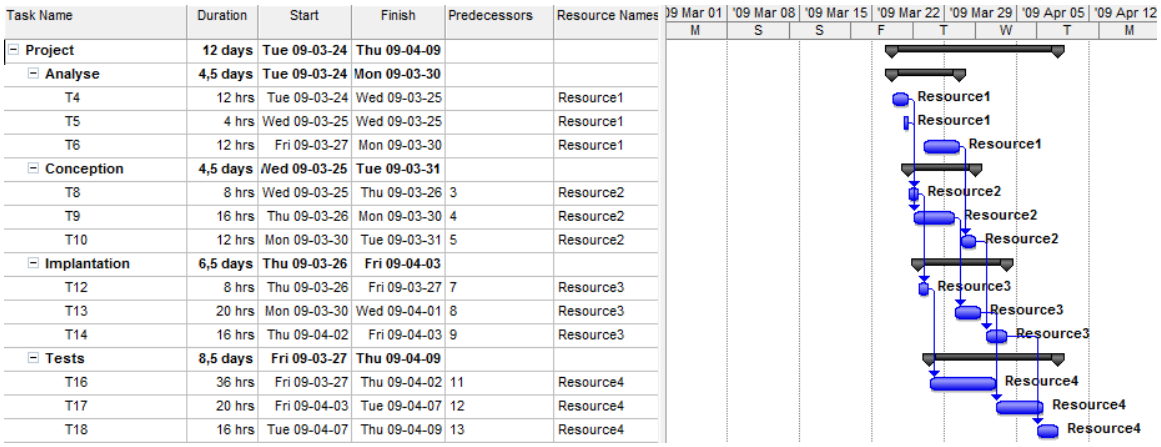


FIG. 4.3 – Diagramme de Gantt de la mise en situation normale avant planification

Notre application exécute l'algorithme *backtrack-search* 108 fois avant qu'un ensemble final d'assignations soit trouvé. Le résultat démontre les ressources travaillant sur une seule tâche à la fois, et ne commençant pas une tâche avant qu'une autre ayant une contrainte de précédence envers



Task Name	Duration	Start	Finish	Predecessors	Resource Names
Project	**12 days**	**Tue 09-03-24**	**Thu 09-04-09**		
Analyse	**4,5 days**	**Tue 09-03-24**	**Mon 09-03-30**		
T4	12 hrs	Tue 09-03-24	Wed 09-03-25		Resource1
T5	4 hrs	Wed 09-03-25	Wed 09-03-25		Resource1
T6	12 hrs	Fri 09-03-27	Mon 09-03-30		Resource1
Conception	**4,5 days**	**Wed 09-03-25**	**Tue 09-03-31**		
T8	8 hrs	Wed 09-03-25	Thu 09-03-26	3	Resource2
T9	16 hrs	Thu 09-03-26	Mon 09-03-30	4	Resource2
T10	12 hrs	Mon 09-03-30	Tue 09-03-31	5	Resource2
Implantation	**6,5 days**	**Thu 09-03-26**	**Fri 09-04-03**		
T12	8 hrs	Thu 09-03-26	Fri 09-03-27	7	Resource3
T13	20 hrs	Mon 09-03-30	Wed 09-04-01	8	Resource3
T14	16 hrs	Thu 09-04-02	Fri 09-04-03	9	Resource3
Tests	**8,5 days**	**Fri 09-03-27**	**Thu 09-04-09**		
T16	36 hrs	Fri 09-03-27	Thu 09-04-02	11	Resource4
T17	20 hrs	Fri 09-04-03	Tue 09-04-07	12	Resource4
T18	16 hrs	Tue 09-04-07	Thu 09-04-09	13	Resource4

FIG. 4.4 – Diagramme de Gantt de la mise en situation normale après planification

celle-ci ne soit terminée. La durée totale du projet est de 12 jours.

Partie 5

Analyse des résultats

5.1 Mise en situation simplifiée

En analysant nos résultats pour la mise en situation simplifiée, les assignations résultantes sont valides : aucune ressource n'investit du temps sur deux tâches en même temps et les contraintes *finish-to-start* sont maintenant respectées. La solution n'est cependant pas optimale ; pour cela il serait nécessaire que la tâche T_5 soit effectuée par la ressource 1 en premier afin que la ressource 2 puisse commencer ses tâches le plus tôt possible.

5.2 Mise en situation normale

En se penchant sur les résultats de la situation normale, nous continuons de remarquer que les ressources investissent du temps sur une seule tâche à la fois et que les contraintes de précédence entre les tâches sont respectées. Ceci nous permet de conclure que la solution trouvée par notre algorithme est valide.

Cependant, plusieurs éléments font en sorte que la solution n'est pas optimale. Premièrement, nous pouvons remarquer que la tâche T_6 aurait pu être amorcée par la ressource 1 une journée plus tôt sans affecter la validité de la solution. Ceci aurait mené de fil en aiguille à une réalisation plus rapide des tâches T_{10} et T_{14}, qui s'ammorcent aussi un peu plus tard que le minimum possible. La tâche T_{18}, cependant, n'aurait pas pu commencer plus tôt, compte tenu des tâches précédentes dont la ressource 4 doit s'acquitter.

De plus, notre implémentation de *backtrack-search* ne tente pas d'assigner les tâches aux ressources dans un ordre qui minimiserait la date de fin de toutes les tâches d'une ressource, d'une partie du projet ou du projet au complet. En priorisant l'accomplissement des tâches les plus courtes

en premier ou celles qui ont le plus de contrainte de précédences avec d'autres par exemple, nous pourrions ainsi obtenir un ensemble d'assignations plus près de la solution optimale.

Partie 6

Conclusion

Nous avons démontré, grâce à ce projet, qu'une technique de planification en intelligence artificielle peut être appliquée à la gestion de projet afin de générer automatiquement des assignations de ressources à des tâches. L'exécution rapide de l'algorithme ainsi que son intégration au logiciel MS Project, un des plus utilisés sur le marché, rend cette application très intéressante.

Nous avons aussi réussi à appliquer une technique de satisfaction de contraintes sur un problème beaucoup plus complexe que ceux présentés dans la littérature académique comme le coloriage de cartes ou le problème des *n-queens*. La modélisation de la problématique n'a cependant pas été une tâche simple. En effet, le simple ajout de la dimension du temps apporte une grande complexité au problème. La représentation des variables et les contraintes formant un problème CSP a donc été pensée en fonction du temps qui évolue au cours de la planification.

6.1 Lacunes et améliorations futures

Malgré le bon fonctionnement de l'application et l'atteinte de notre objectif principal, soit la planification automatique de ressources et de tâches dans le cadre de la gestion de projet, certaines lacunes peuvent être relevées dans notre approche ou dans notre implémentation.

Bien que l'algorithme trouve toujours une solution valide (si elle existe), rien ne garantit son optimalité. L'algorithme *backtrack-search* que nous avons implémenté retourne la première solution consistante par rapport aux contraintes qui est trouvée. Des heuristiques permettant d'ordonner les tâches plus prioritaires en premier, par exemple, pourraient conduire à une meilleure solution.

La qualité d'une solution pourrait également être améliorée en optant pour une plus fine granularité des plages horaire. Dans l'état actuel de notre implémentation, une journée de travail est divisée en 3 blocs de 4 heures : le matin (am), l'après-midi (pm) et le soir ($evening$). Une subdivision

plus fine des plages horaire, par exemple en blocs d'une heure, permettrait d'éviter l'inutilisation d'une ressource lorsqu'une tâche se termine avant la fin d'une tranche de temps.

Lorsque l'assignation des ressources aux tâches est impossible à réaliser dans les délais donnés, l'algorithme ne retourne aucune solution. Il serait intéressant de plutôt retourner la meilleure assignation possible jusqu'à la date d'échéance du projet. Mieux encore, l'algorithme pourrait dépasser la date limite et identifier automatiquement la durée minimale du projet. Cependant, une telle méthode nécessiterait l'ajout dynamique de variables au problème, puisque les variables sont des tuples « ressource – date – plage horaire ».

L'interface graphique de notre application gagnerait à être peaufinée. Nous pourrions par exemple ajouter des réglages ajustables par l'utilisateur, comme les types de tâches exécutables par chaque ressource. Ultimement, le planificateur pourrait être intégré directement à MS Project par le biais d'une barre d'outils.

Il est à noter que deux planifications successives du même fichier produiront des résultats différents. Cela se produit parce que nous utilisons l'assignation précédente de tâches pour déterminer quelle ressource peut effectuer quel type de tâche. Puisque l'assignation sera modifiée suite à une planification, les ressources ne pourront qu'exécuter les tâches auxquelles elles étaient assignées précédemment. Prenons par exemple une tâche T_1 pouvant être exécutée par deux ressources, X et Y. Supposons que les deux ressources sont assignées à la tâche T_1 dans la première assignation et que la planification assigne uniquement la ressource X à la tâche T_1 et que la ressource Y est inutilisée (ou assignée à une autre tâche). Lors de la deuxième planification, seule la ressource X sera considérée comme pouvant exécuter la tâche T_1. L'interface graphique améliorée que nous venons tout juste d'évoquer permettrait de régler ce problème de façon élégante et conviviale.

Un algorithme de recherche différent pourrait être implémenté pour accélérer la planification. L'algorithme *AC-3* (*arc consistency*) trouve généralement une solution beaucoup plus rapidement que l'algorithme *backtrack-search* de base [3]. Les bases de cet algorithme ont déjà implémentées dans le code fourni sur le CD-ROM (classe *ArcConsistency*) ; le travail restant à effectuer devrait être relativement trivial.

Enfin, une dernière amélioration possible à cette application serait d'effectuer une planification partielle d'un projet. Par exemple, un gestionnaire pourrait décider, après la moitié de l'avancement d'un projet, de revoir l'assignation des ressources ; les tâches déjà exécutées ne devraient pas être prises en compte par le planificateur. Deux solutions se présentent. La première consiste simplement à ignorer la création des variables pour les dates antérieures à la date actuelle (ou à une date donnée par l'utilisateur). La deuxième méthode consiste à exploiter le fonctionnement des algorithmes CSP en ne donnant qu'une seule valeur aux variables qui ne doivent pas être changées. L'algorithme de recherche se chargera automatiquement d'assigner l'unique valeur possible à ces variables. Cette seconde méthode, plus flexible, permettrait également de forcer certains comportements sur le planificateur.

Bibliographie

[1] Guy Brault : Génération automatique de plans en gestion de projets. Mémoire de D.E.A., Université de Sherbrooke, Sherbrooke, Québec, Canada, 2003.

[2] Malik Ghallab, Dana Nau et Paolo Traverso : *Automated Planning : Theory and Pracice*. Morgan Kaufmann, San Francisco, California, 2004.

[3] Stuart Russell et Peter Norvig : *Artificial Intelligence : A Modern Approach*. Prentice Hall, Upper Saddle River, New Jersey, 2e édition, 2003.

[4] David R. Wilkins : *Getting Started with LaTeX*. 2e édition, 1995.