Projet d’info n°2

Optimisation de procédés industriels

# Contexte et sujet :

Diam Portugal, secteur de l’industrie de transformation du liège (bouchons techniques), est sur le marché la 2ème plus grande entreprise mondiale du secteur, pionnière dans des solutions comme le TCA-Free (TriChloroAnisole). L’unité concernée est l’Usine de Fiães dans la Zone de Finition.

Contact :

Eric FEUNTEUN P93  
CEO (Directeur Général) DIAM BOUCHAGE, Mail: [FEUNTEUN@diam-bouchage.com](mailto:FEUNTEUN@diam-bouchage.com)

**Contexte opérationnel :**

* Nombre de machines de finition : 13
* Caractéristiques des machines : Potentiellement spécialisées ou avec des paramètres distincts
* Type de produit : Bouchons techniques en liège – produit final personnalisé selon le client
* Demande : Stable et prévisible (OF confirmées à l’avance)
* Défi actuel : Goulot d’étranglement en finition, besoin de maximiser l’efficacité, de minimiser les temps de changement de série et de respecter les délais

**Variables et contraintes à considérer :**

* Priorité selon la date de livraison
* Temps de changement de série entre différents types de produits
* Capacité ou profil technique des machines
* Regroupement des OF par client, type, taille, lavage, etc.
* Plages ou créneaux opérationnels disponibles
* Disponibilité de stock
* Localisation du stock
* Disponibilité des marquages
* Urgences éventuelles (OF « critiques »)
* Flexibilité de transfert entre machines (si applicable)

**Résultats attendus :**

* Outil prototype pratique et adapté à notre environnement  
  au choix une application simple, une application Web, une intégration à l’ERP (Enterprise resource planning)
* Logique de séquencement paramétrable
* Simulation de scénarios de planification
* Possibilité d’exporter la séquence vers les systèmes actuels

**Ressources disponibles :**

* Données historiques et actuelles des OF
* Équipe technique disponible pour des ateliers et validations
* Possibilité de tester des prototypes en environnement réel
* Collaboration active avec les équipes de planification et de production

Etapes de fabrication :

* Tamiser et purifié par le procédé diamant
* Mélangé avec un liant et micro sphère
* Remplir des moules
* Passage au four
* Repos 4 jours
* Ponçage
* Rogner
* Façonnage
* Contrôles
* Trieuse optique
* Marquage
* Satiné
* Conditionné
* Réalisation si besoin arrondi pour les spiritueux

# Résolution mathématique :

\*\* PROBLEME \*\*  
  
Il s’agit ici de traiter d’un problème décisionnel posé comme un problème dit d’ "optimisation combinatoire”.  
Le cahier des charges est assez bref mais il semble en effet que le problème se pose ainsi:  
  
Étant donné un ensemble de tâches  (les ordre de fabrications OF) et des machines de finition, il va falloir décider, pour chaque OF:  
- sur quelle(s) machine(s) l'exécuter  
- et à quel moment  
de manière à minimiser, par exemple, la somme des retards de livraison, et en tenant compte de différentes contraintes qui, typiquement, vont interdire d’exécuter les tâches toutes ensemble sur la même machine.  
C’est ce qu’on appelle un "problème d’ordonnancement “ où il s’agit d’affecter les machines et le temps aux tâches, et ce n’est pas évident a priori de calculer la meilleure solution.  
  
Pour le résoudre, il va déjà falloir le formaliser sur papier. À partir de là vous pourriez essayer d'écrire un algorithme “heuristique” capable de construire une bonne solution (à défaut de la meilleure). Mais je vous propose plutôt (car une semaine ça va passer vite) d'appeler directement un solveur d’optimisation mathématique. Dans ce cas, il vous faudra décrire votre modèle mathématique et l’implémenter dans le format du solveur pour simplement appelé l’algorithme de résolution pré-implémenté.  
  
\*\* MODELE \*\*  
  
En l’occurence ici, on peut probablement modéliser ce problème sous la forme d’un programme linéaire en nombres entiers (PLNE) et appeler le solveur Gurobi pour le résoudre.  
Comme vous avez vu en cours d’optimisation, il s’agit de décrire une solution du problème comme un vecteur de variables, l’objectif à optimiser comme une fonction (linéaire) de ces variables, et les contraintes comme des inégalités (linéaires ou affines) qui restreignent l’espace de définition de ces variables.  
  
Mais à la différence du cours, ici les variables représentent des décisions d’affectation (tâches/machines et tâches/temps), discrètes donc, que l’on va modéliser comme des variables à valeurs entières (0,1,2,3,..) ou binaires 0/1:  
Par exemple, pour modéliser l’assertion "la tâche numéro i est en cours d’éxécution sur la machine numéro m au pas de temps numéro t",on peut définir une variable binaire indexée sur le triplet (i,m,t) avec  x\_{i,m,t}  = 1 si l'assertion est vraie, et  x\_{i,m,t}  = 0 sinon. D’autres modèles sont possibles:  
par exemple, on peut définir une variable binaire y\_{im}= 0 ou 1 (pour l'affectation de la machine) et une variable à valeur entière s\_i = t (pour l'affectation de la date de début de la tâche).  
  
Le choix du modèle va dépendre du type de contraintes que vous aurez à exprimer. Par exemple, dans le premier modèle, la relation suivante exprime la contrainte "la machine m ne peut pas exécuter plus de 5 tâches différentes simultanément au temps t":  
  
Somme\_(tâches i) x\_{imt} <= 5  
  
C’est bien une inégalité qui est linéaire en chacune des variables.  
  
Il y a aussi des contraintes implicites (qui paraissent évidentes mais qui doivent être définies explicitement dans le modèle mathématique). Par exemple, la contrainte “une tâche ne démarre qu’une seule fois”  est implicite dans le second modèle (car la variable s\_i ne prend qu’une seule valeur t dans la solution) mais elle doit être explicitée dans le premier modèle  
Pour l’exemple, elle s’écrirait ainsi en définissant de nouvelles variables binaires s\_{imt} =1 si i débute au temps t sur la machine m, =0 sinon:  
s\_{imt} >= x\_{im(t+1)} - x\_{imt} pour tout (i, m, t)  
Somme\_(m) somme\_(t) s\_{imt} == 1 pour tout i  
  
  
Parce que les variables sont discrètes, le problème devient non-convexe et on utilise alors des algorithmes (par ex: branch-and-bound) très différents de ceux que vous avez vu en cours.  
En revanche, avec des variables binaires et des contraintes linéaires, il est possible de modéliser des conditions logiques ou des relations nonlinéaires.

# To do :

La première chose à faire va être d’interroger le partenaire industriel pour prendre connaissance du fonctionnement de l’atelier et donc des contraintes du problème (Peut-on exécuter plusieurs tâches sur une machine ? Ou une tâche sur plusieurs machines ? Dans quelle mesure peut on cumuler les tâches (en fonction de la taille ?), comment peut-on discrétiser l'horizon de planification ? y a t il un temps de préparation des machines avant une tâche ? Etc)  
  
Il y a des chances pour que le partenaire oublie de spécifier certaines contraintes (notamment celles qui sont implicites pour lui) ou évoque au contraire des aspect qui n’interviennent pas dans le modèle.  
En pratique, vous allez construire votre modèle en ajoutant les contraintes progressivement. C’est souvent en regardant la solution que le partenaire voit les contraintes manquantes. Vous pouvez aussi étudier les données qui vous seront fournies pour détecter des contraintes non spécifiées.  
En ajoutant des nouvelles contraintes, vous aurez besoin de définir de nouvelles variables, qu’il faudra bien penser à lier avec les variables de décision.   
  
L’idée c’est que dans la semaine vous réussissiez à intégrer un maximum de contraintes, et de réaliser un prototype pour l’entreprise à partir de ce modèle et du solveur Gurobi.  
Ca ne correspondra pas à ce que demande l’entreprise qui veut une “application simple” mais le fait est que ce n’est pas un “problème simple”, et je doute qu’en une semaine on peut imaginer aller jusqu’au déploiement. Par la suite, une fois que le modèle sera construit, on pourra remplacer Gurobi par un solveur de PLNE gratuit (celui d’excel par exemple mais qui est beaucoup moins performant) ou réfléchir à un algorithme “maison”, mais dans le cadre de ce projet, écrire et faire tourner le modèle, même incomplet, ce sera déjà très bien.  
  
Ne négligez pas non plus le temps nécessaire à la mise en forme des données (en input du modèle) et la mise en forme des solutions en output: peut-être 1 ou 2 d’entre vous pourraient se consacrer à cette tâche pendant que les autres démarrent la modélisation (toujours sur papier ! Avant d’implémenter/tester contrainte par contrainte).

# Questions :

Comment le séquençage est-il géré aujourd’hui ?

Est-il possible de parler à des gens de l’usine qui décide du séquençage ? (cf collab active équipes de planification)

ERP de l’entreprise ? (Enterprise resource planning)

Equipe technique dispo = équipe informatique entreprise ?

Tester en environnement réel ?

Peut-on exécuter plusieurs tâches sur une machine ?

Ou une tâche sur plusieurs machines ?

Dans quelle mesure peut-on cumuler les tâches (en fonction de la taille ?), comment peut-on discrétiser l'horizon de planification ? y a t il un temps de préparation des machines avant une tâche ? Etc)

# Documentation :

Gurobi met à disposition des exemples implémentés en python sous Jupyter Notebook et Google Colab:  
<https://www.gurobi.com/jupyter_models/>  
Je vous propose de commencer par lire et faire tourner le tutoriel:  
<https://www.gurobi.com/jupyter_models/introduction-to-mathematical-optimization-modeling/>  
  
Vous pouvez même modifier ce code pour implémenter/exécuter votre propre modèle (et l’enregistrer sous GitHub ou google drive)  
Vous pouvez aussi partir de codes de problèmes d’ordonnancement plus élaborés, comme:  
<https://www.gurobi.com/jupyter_models/airline-planning-after-flight-disruption/>  
Ou mieux encore:  
<https://www.gurobi.com/jupyter_models/technician-routing-and-scheduling-problem/>  
(Dans ce dernier, les variables d’ordonnancement t\_j correspondent aux s\_i du second modèle ci-dessus)  
  
Les slides de mon cours de programmation linéaire en nombres entiers sont disponibles ici:  
<https://sofdem.github.io/teach/ose/demassey-milp-psl23.pdf>  
Ici des exemples d’implémentation de problèmes et comment modéliser des conditions logiques ou des relations non-linéaires au moyen de variables binaires:  
<https://sofdem.github.io/teach/ose/demassey-milpnotes-ose24.pdf>