EURODECISION
OPERATIONAL RESEARCH

LP-TaskPlanner, un *framework* industriel, générique et flexible de génération de colonnes par résolution d'un plus court chemin contraint

Guillaume Pinot

ROADEF 2014

Sommaire

Sommaire

Introduction

Pourquoi LP-TaskPlanner ?

- Utilisation intensive de génération de colonnes par résolution d'un plus court chemin contraint.
- Besoin d’un outil séparant la modélisation de la résolution.

Qu’est-ce que LP-TaskPlanner

- Un modèle mathématique
  - Un problème maître de type programme linéaire
  - Un (ou des) sous-problème de type plus court chemin contraint dans un graphe fournissant des colonnes au problème maître
- Des algorithmes travaillant sur ce modèle mathématique

Sommaire

Description du problème exemple 1/2

- **Problème de planification RH : affecter des tâches à des employés**

- **Des tâches**
  - Fixées dans le temps (une date de début et une date de fin)
  - Chaque tâche doit être réalisée par 1 employé
  - Certaines ne peuvent être réalisées que par des experts

- **Des règles d'enchaînements**
  - Pour enchaîner 2 tâches, il faut qu'elles soient séparées de 2h maximum
  - S'il y a plus de 15 min entre 2 tâches, nous avons une pause

Description du problème exemple 2/2

Des employés

- 2 compétences : junior et senior
- Chaque employé ne peut faire qu'une tâche à la fois
- Les juniors sont payés 10€/h, les seniors 17€/h

Des règles de condition de travail

- Le temps de travail doit être de 9h maximum, 8h pour les juniors
- On ne peut travailler plus de 2h sans faire de pause
- La moyenne des temps de travail ne peut dépasser 7h
- Chaque employé sera payé au minimum 6h même s'il travaille moins.

Une fonction de coût

- Maximiser le nombre de tâches réalisées
- Minimiser la paye globale

Sommaire

Les ressources

Plusieurs types

- Les EnumRes : une valeur parmi un ensemble de valeur
- Les TaskSetRes : un sous ensemble d'un ensemble de valeur
- Les RealRes : un réel
- Les ListRealRes : une liste de réel

Les types des Enums

```
EnumType skill_type = sp.addEnumType({"junior", "senior"});
EnumType task_type = sp.addEnumType(vTask);
```

Les ressources

```
EnumExpr skill = sp.addEnumRes(skill_type);
RealExpr totalTime = sp.addRealRes();
ListRealExpr continuousTime = sp.addListRealRes();
TaskSetResIdx tasks = sp.addTaskSetRes(task_type);
```

Les expressions

Plusieurs types

- EnumExpr, RealExpr, ListRealExpr, BoolExpr, ListBoolExpr…
- BoolCst

Des opérateurs

- Calcul : +, -, *, max…
- Comparaison : ==, !=, <=, >=…
- Opération booléennes : Not, And, Or, All…
- …

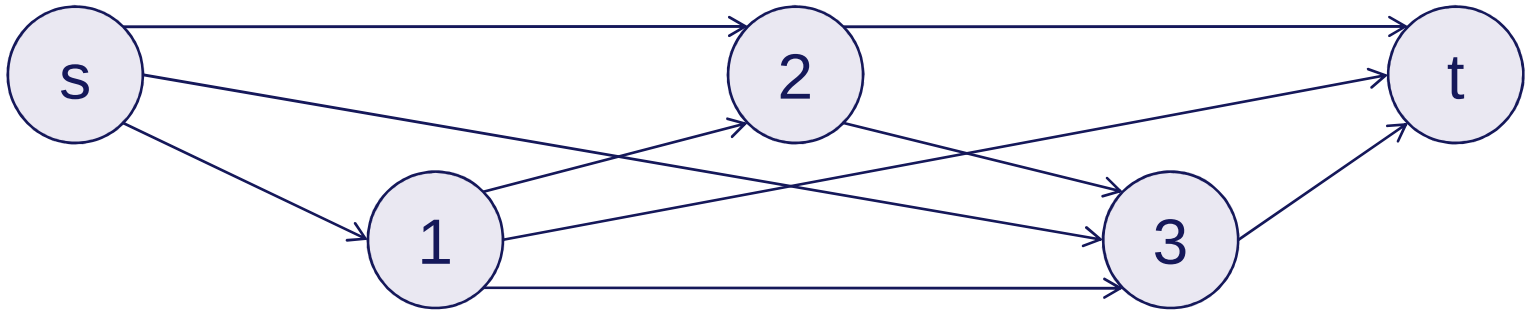
Notre problème

```
BoolExpr isJunior = skill == junior;
BoolExpr isSenior = skill == senior;
RealExpr paidTime = max(6., totalTime);
RealExpr bill = isJunior * (10 * paidTime) + isSenior * (17 * paidTime);
sp.addBoolCst(totalTime <= 9.);
sp.addBoolCst(!isJunior || totalTime <= 8.);
sp.addBoolCst(all(continuousTime <= 2.));
```

Le graphe

- **Des nœuds et des arcs dont 2 nœuds spéciaux : source et puits**

- **Pour notre cas**
  - Source s : début du service
  - Puits t : fin du service
  - 1 Nœud par tâche
  - Pour toute tâche i, un arc entre s et i, et un arc entre i et t
  - Pour tout couple de tâches enchaînables (i, j), un arc entre i et j



Les transitions

- **Les transitions modifient les ressources**
  - EnumRes : Diff, Eq, SubSet
  - RealRes : Add
  - ListRealRes : Add, AddElt
  - TaskSetRes : AddTask
  - …
- **Chaque nœud et chaque arc a une liste de transitions, réalisées dans l'ordre du chemin entre s et t**

- **Pour notre cas**
  - Sur s : [AddElt(continuousTime)]
  - Sur les nœuds des tâches i de durée d : [Add(continuousTime, d); Add(totalTime, d); AddTask(tasks, i), *Diff(skill, junior)*]
  - Sur les arcs de durée d
    - Pause : [AddElt(continuousTime); Add(totalTime, d)]
    - Sinon : [Add(continuousTime, d); Add(totalTime, d)]

Le problème maître

- Le poids des colonnes-chemins sont des RealExpr

- La fonction objectif

```
masterProblem.getObjective().setPathWeight(bill);
```

- La contrainte sur la moyenne

```
MasterConstraint maxMeanCtr;
maxMeanCtr.setPathWeight(totalTime - 7);
maxMeanCtr.setMax(0);
masterProblem.addConstraint(maxMeanCtr);
```

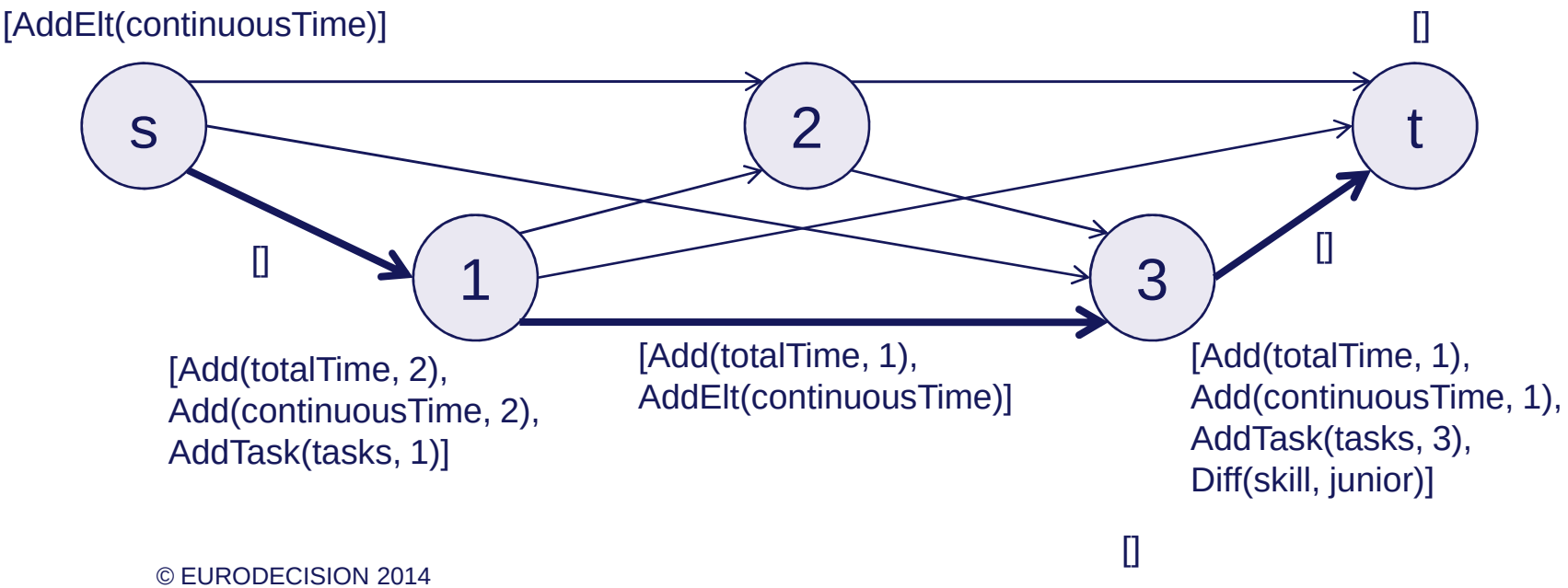
- Les contraintes de couverture de tâches i

```
MasterConstraint coverCtr;
coverCtr.setPathWeight(hasTask(taskRes, i));
coverCtr.setMin(1);
coverCtr.setMax(1);
coverCtr.setGoalMin(1e5);
masterProblem.addConstraint(coverCtr);
```

Sommaire

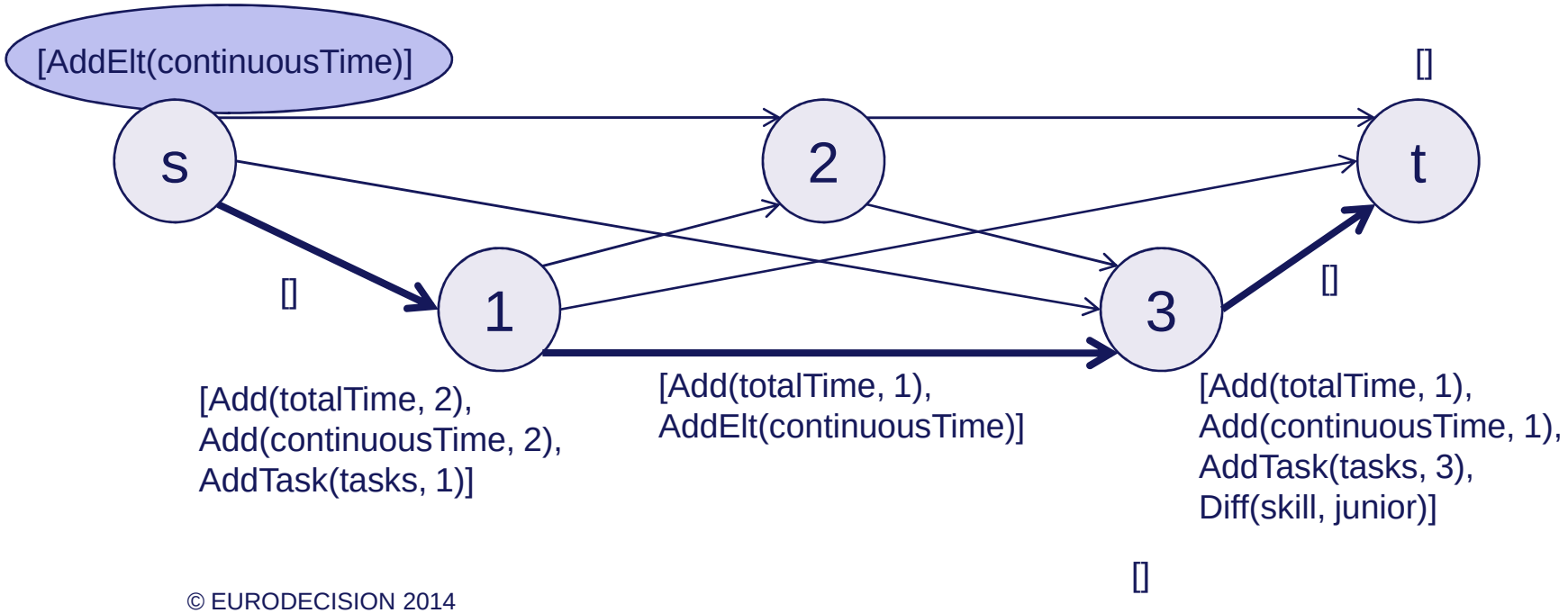
Colonne-chemin

skill ∈ {junior, senior}
totalTime = 0
continuousTime = []
tasks = {}



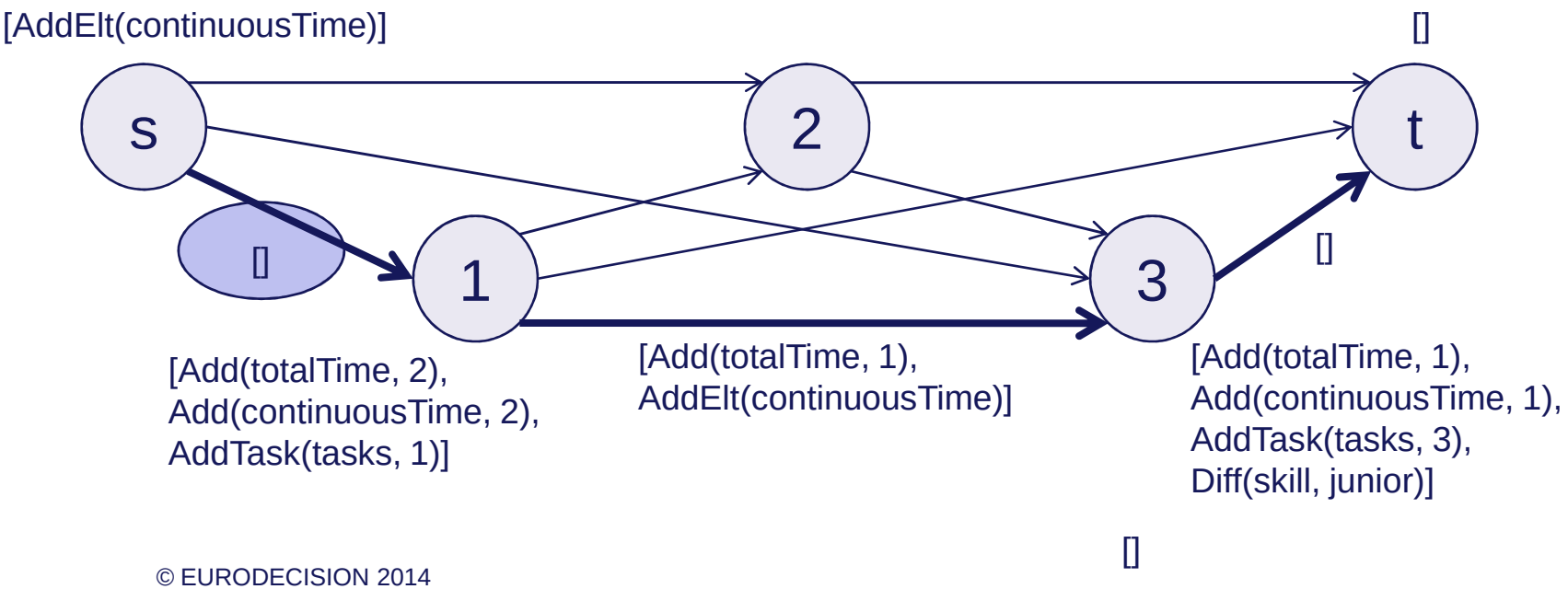
Colonne-chemin

skill ∈ {junior, senior}
totalTime = 0
continuousTime = [0]
tasks = {}



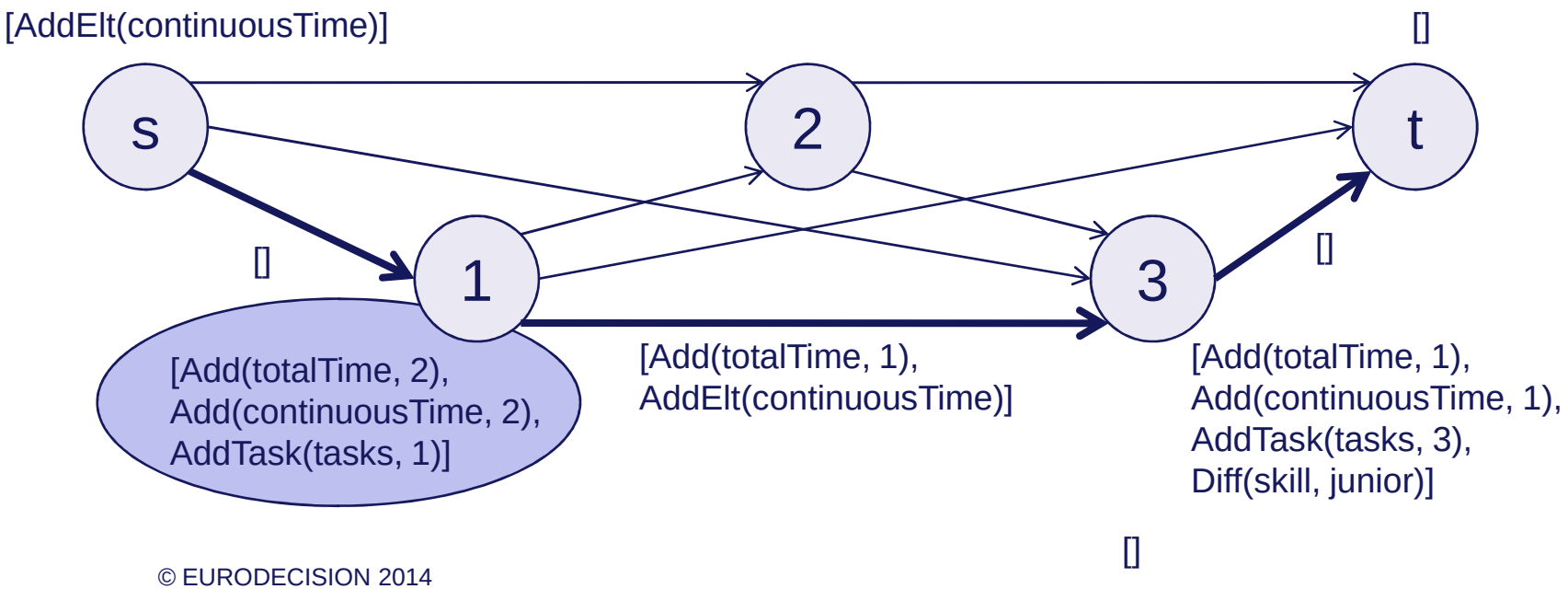
Colonne-chemin

skill ∈ {junior, senior}
totalTime = 0
continuousTime = [0]
tasks = {}



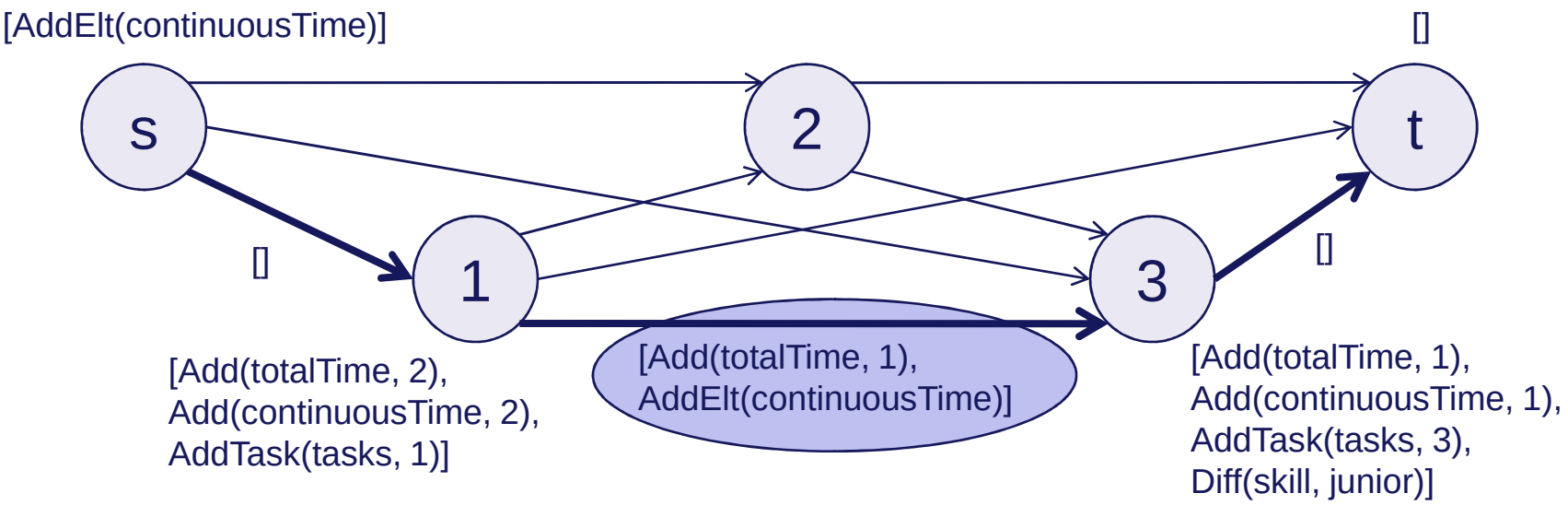
Colonne-chemin

skill ∈ {junior, senior}
totalTime = 2
continuousTime = [2]
tasks = {1}



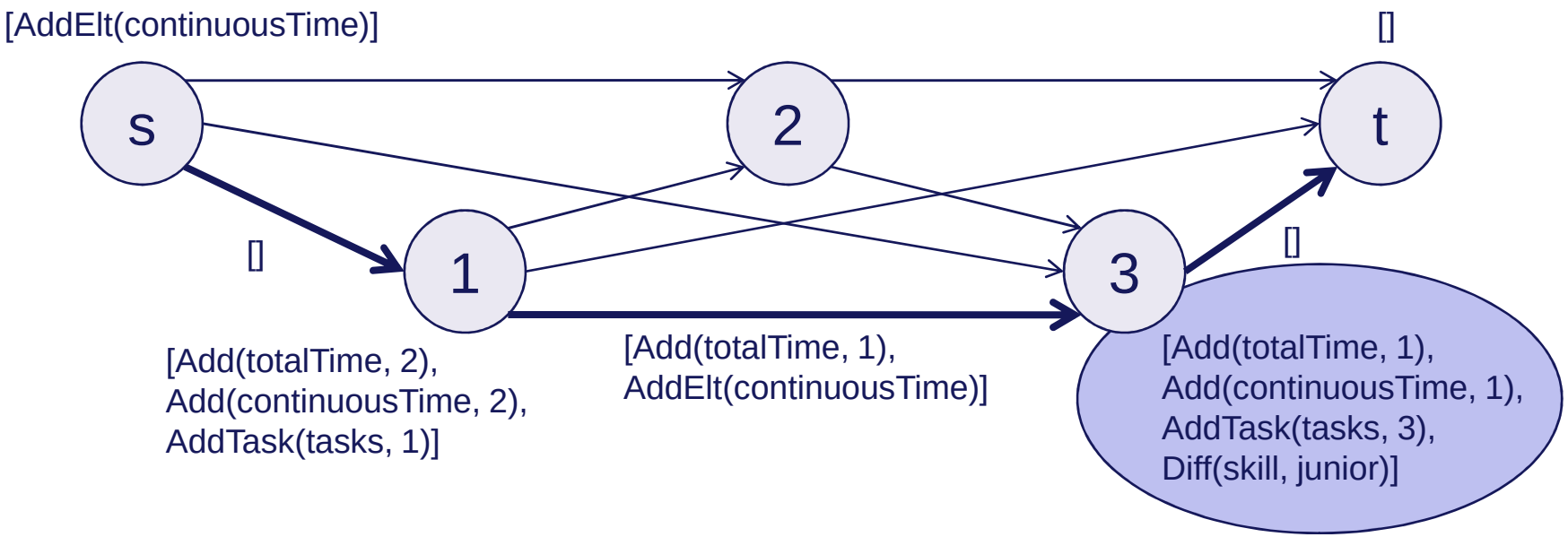
Colonne-chemin

skill ∈ {junior, senior}
totalTime = 3
continuousTime = [2, 0]
tasks = {1}



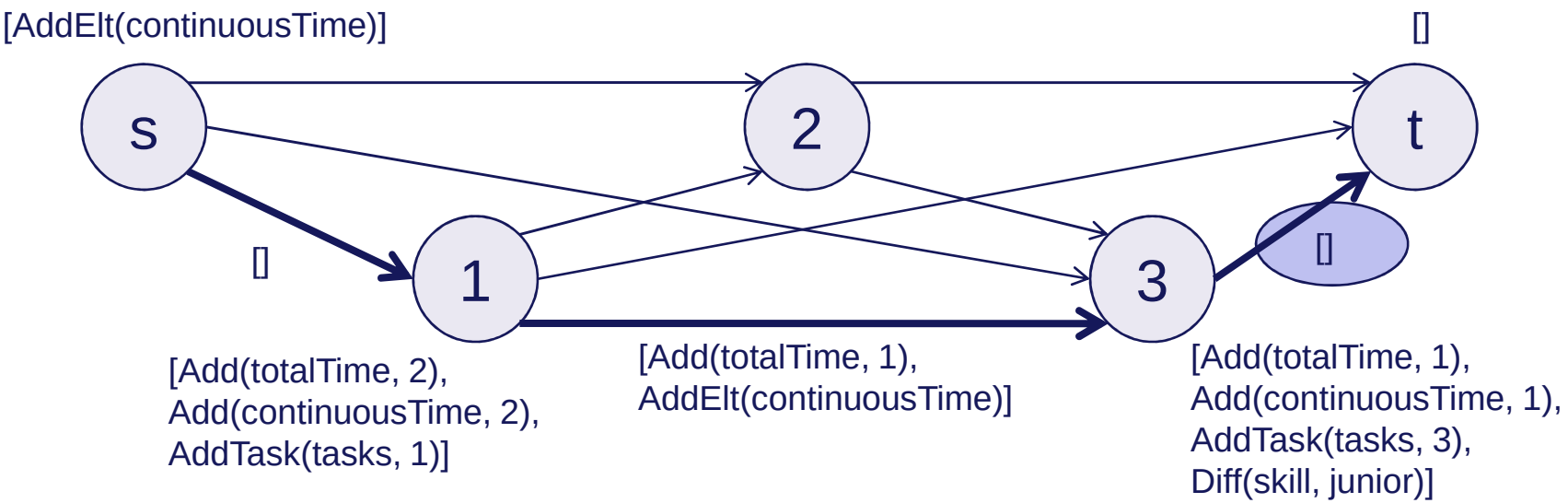
Colonne-chemin

skill ∈ {senior}
totalTime = 4
continuousTime = [2, 1]
tasks = {1, 3}



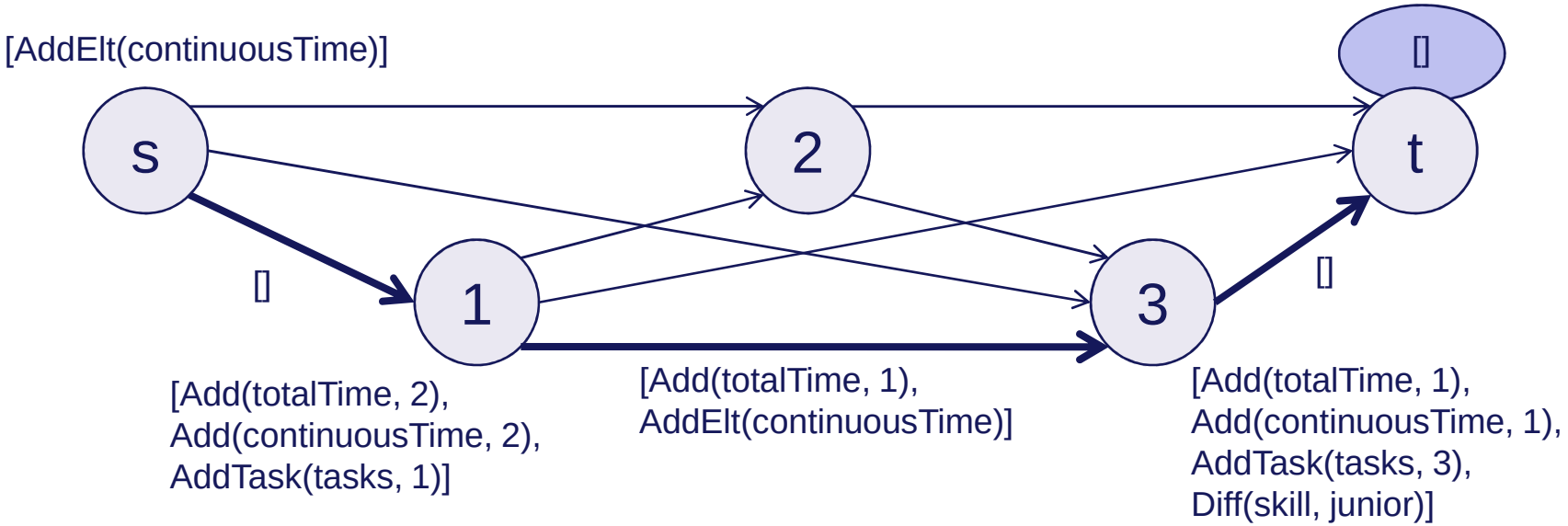
Colonne-chemin

skill ∈ {senior}
totalTime = 4
continuousTime = [2, 1]
tasks = {1, 3}



Colonne-chemin

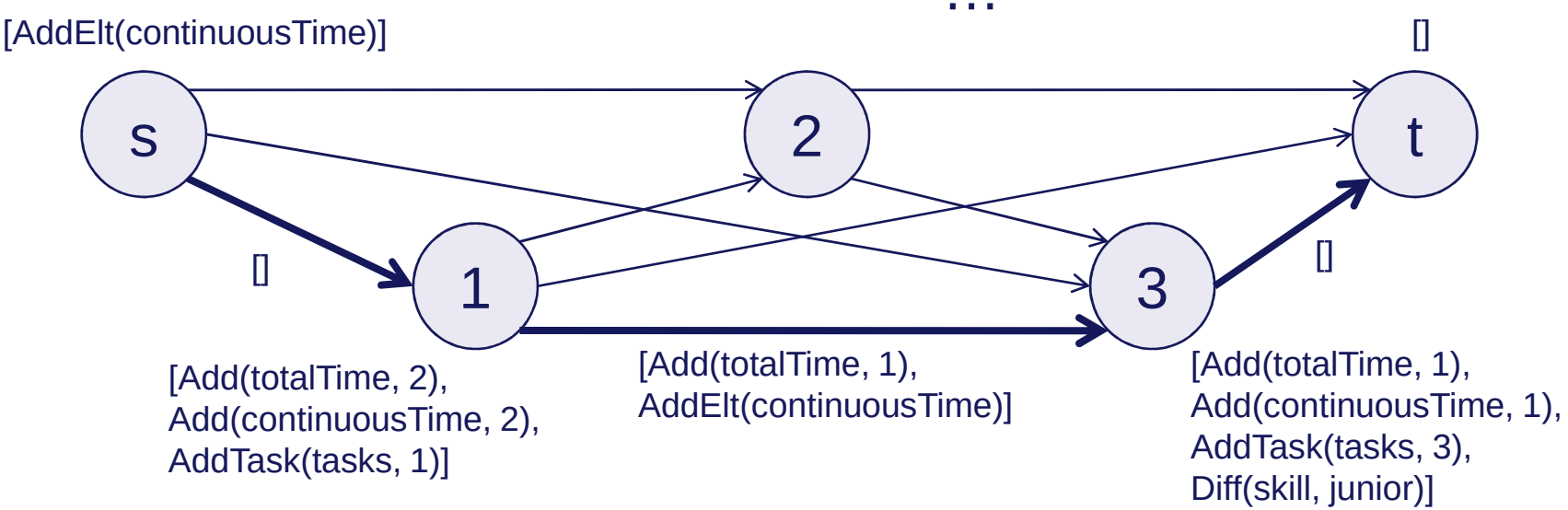
skill ∈ {senior}
totalTime = 4
continuousTime = [2, 1]
tasks = {1, 3}



Colonne-chemin

skill ∈ {senior}
totalTime = 4
continuousTime = [2, 1]
tasks = {1, 3}

skill = senior
constraints : OK
paidTime = 6, bill = 102
objWeight = 102
maxMeanWeight = -3
cover1Weight = 1
…

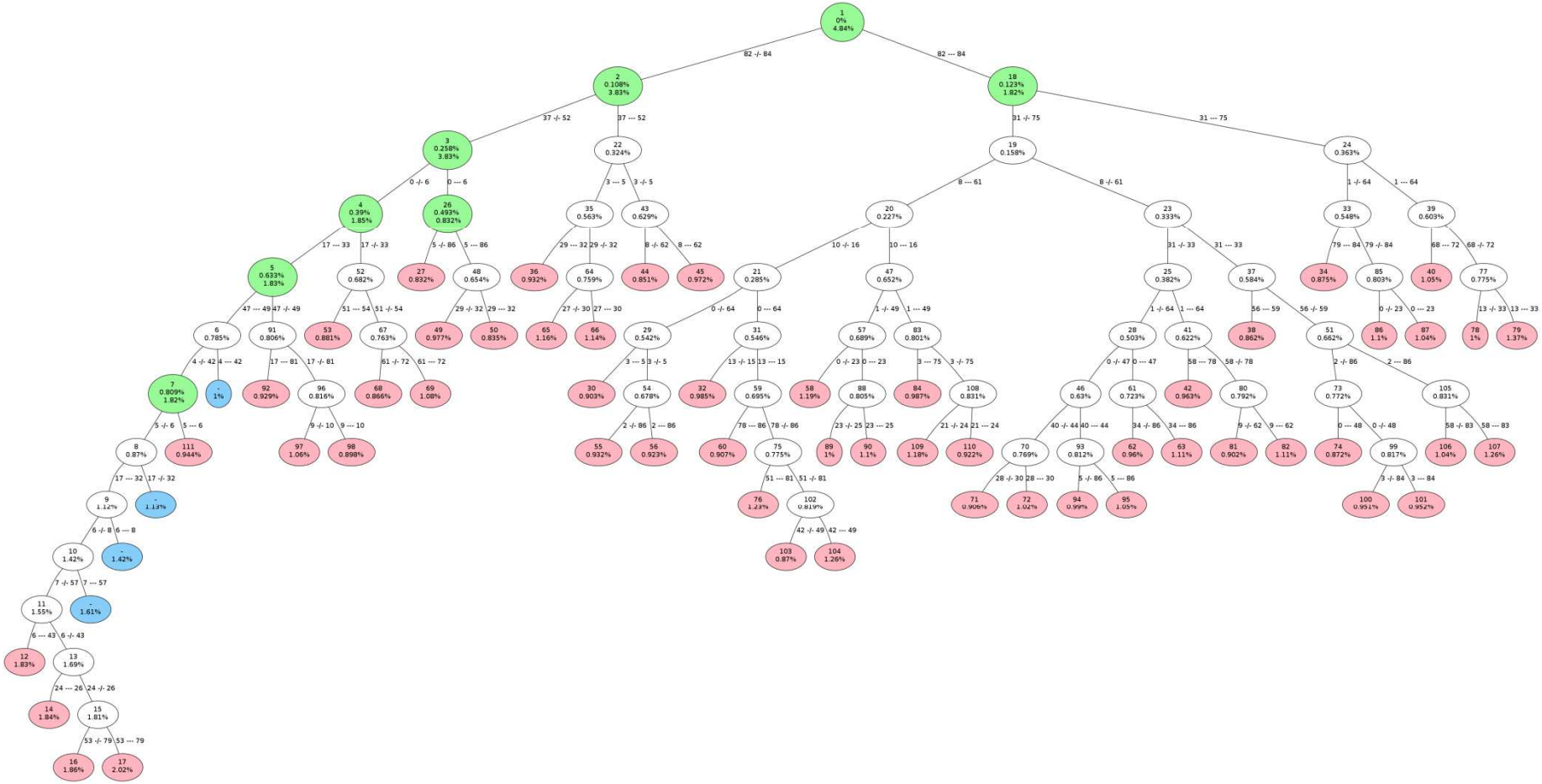


Sommaire

Résolution

On lance la résolution

```
solveWithColgen(tpProb);
```



Méthodes de résolution

Paramètres

- Choix du sous-problème (explicite, programmation dynamique…)
- Règle de branchement (Ryan & Foster…)
- Parcours d'arbre (profondeur, largeur…)
- Heuristiques primales
- Algorithme de séparation (cliques…)
- …

Technologies utilisées

- Calcul d'intervalles
- Génération de fonction de dominance
- Programmation dynamique
- Stabilisation
- Prétraitements
- …

Sommaire

Conclusion

- **Une génération de colonnes par résolution d'un plus court chemin contraint**
  - Séparation de la modélisation et de la résolution
  - Algorithmes de résolution génériques et paramétrables
  - Notre exemple en une centaine de lignes de C++
- **Utilisation actuelle**
  - LP-EasyDriver : habillage et graphicage en transport en commun (HK : 6 659 tâches, 1 350 790 arcs, 250 services)
  - CrewPairing (Projet de recherche CleanSky/Imagine) : génération de rotations PN implémentant EUOPS
- **Utilisations futures envisagées**
  - Résolution de problèmes de planification de RH
  - Tournées de véhicules

EURODECISION
OPERATIONAL RESEARCH

LP-TaskPlanner, un *framework* industriel, générique et flexible de génération de colonnes par résolution d'un plus court chemin contraint

Guillaume Pinot

ROADEF 2014