Université de Montpellier

Faculté des Scicences

30 Place E.Bataillon, 34095 Montpellier

Année 2023-2024

Rapport de Projet

Projet CSP – Génération de benchmark et évaluation de méthode

par

Romain GALLERNE & Loris BENAITIER

Encadrant de TP : Mme Marie-Laure Mugnier

Responsable du module : M. Michel Leclère

1 Construction des jeux d'essais

1.1 Paramètres de générations

Une première étape du projet a été d'identifier le rôle de chaque paramètre de la fonction urbcsp afin de comprendre en quoi celui-ci impactait les CSP générés. De part les formules de calcul de la densité et de la dureté, on a respectivement :

— **Densité** : $2c/(n^2-n)$

— **Dureté** : $(d^2-t)/d^2$

Avec c, le nombre de contraintes, n le nombre de variables, d la taille du domaine et t le nombre de tuples.

On déduit rapidement de ces formules (que l'on multipliera par 100 pour les exprimer en pourcentages) que :

— Le nombre de variable fait **baisser** la densité

— Le nombre de contraintes fait **augmenter** la densité

— La taille du domaine fait **augmenter** la dureté

— Le nombre de tuples fait **baisser** la dureté

Afin d'obtenir des résultats pertinent, il est ici important de fixer certains paramètres. Premièrement, le nombre de réseaux qui n'a pas d'importance sur la densité ou la dureté mais permet, lorsque il est plus élevé d'obtenir des résultats plus fiable au prix d'un temps de calcul exponentiellement plus élevé. Nous fixerons celui-ci à **15** pour tout nos essais. Le nombre de variables sera lui aussi fixer à **40**, de même que la taille du domaine à **25**.

Ainsi, les seuls paramètres qui seront variant sont le nombre de contraintes et le nombre de tuples. Ceux-ci étant suffisant pour contrôler la densité et la dureté. Lorsque le nombre de contraintes augmentera, c'est la densité qui augmentera et lorsque le nombre de tuples diminuera, c'est la dureté qui augmentera.

```
for c in 100 200 300
do
    echo "Generation des fichiers pour c=$c"
    for ((t=100;t<=600;t+=5));
    do
    mkdir -p CSP/"nbContraintes-$c"
    ./urbcsp 40 25 $c $t 15 > CSP/"nbContraintes-$c"/"csp-40-25-$c-$t-15.txt"
    done
done
echo "Fichiers CSP generes"
```

Script .sh utilisé pour générés les différents fichiers CSP.

1.2 Caractéristiques des CSP

Comme expliqué dans la partie précédente, nous ne ferons varier ici que le nombre de tuples et le nombre de contraintes. Le nombre de contraintes c variera selon des valeurs présélectionnés permettant de couvrir différentes densités à des fins de comparaison. c prendra donc respectivement les valeurs **100**, **200** et **300**. Ces valeurs permettront de couvrir les densités : **13%**, **26%** et **38%**.

Pour chaque densité prévu, nous étudierons l'effet de l'évolution de la dureté. Pour cela nous prévoyons de faire varier le nombre de tuples de **600** à **100** avec un pas de 5 ; passant ainsi la dureté de **4%** à **84%**, la dureté variant ainsi d'un peu moins d'1% à chaque pas. Il n'est pas utile d'aller de 0 à 100%, car le taux de satisfaction atteint rapidement des valeurs extrêmes. On cherche principalement à observer la transition de phase.

1.3 Calcul du Taux de satisfaction et du Temps

Afin de générer des statistiques et des courbes pertinentes, il nous faut tout d'abord pouvoir calculer efficacement deux paramètres : le taux de satisfaction d'un réseau et le temps.

Taux de Satisfaction

Pour calculer le taux de satisfaction, nous allons compter le nombre de réseaux ayant été satisfaits en moins de 10s puis le diviser par le nombre de réseaux total d'un CSP (ici 15). Nous choisissons d'imposer un délais ou *"TimeOut"* de 10 secondes afin que chaque réseau ne prennent pas un temps trop long (voir infini) à être parcouru. Ce délais à été mis en place grâce à la méthode *limitTime()*. Par satisfait, on entend que le système a pu trouver au moins une solution. Ce taux de satisfaction est enfin multiplier par 100 afin d'obtenir une pourcentage. On s'attend à ce que celui-ci s'approche de 100 pour des réseaux à faible dureté puis qu'il atteigne presque 0 pour des réseaux à haute dureté.

Calcul du temps

Le calcul du temps est un peu plus complexe à mettre en place, notamment car celui-ci nécessite de prendre en compte le caractère imprévisible de la machine virtuel de Java (JVM). Nous réaliserons donc 5 essais pour chaque réseau où nous mesurerons le temps de résolution (<=10s donc). Afin d'éviter au maximum les valeurs "parasites" de la machine virtuel capricieuse, nous retirerons à chaque fois les deux valeurs les plus extrême (la plus grande et la plus petite) avant de faire la moyenne des trois restante. Cette moyenne constituera le temps d'exécution considéré pour ce réseau. La mesure du temps est ici effectuer avec les méthodes java appropriés (getCurrentThreadUserTime() et getCurrentThreadCpuTime()). Ces deux valeurs sont mesurés avant l'exécution de la fonction de taux puis on soustrait le temps utilisateur au temps CPU avant de faire la manipulation décrite plus haut.

Calcul du "TimeOut"

Comme indiqué plus haut, on définit un délais de 10s. Afin de mesurer la quantité de délais atteints (qu'on considère comme des réseau non-satisfait) on incrémente un compteur à chaque délais dépassé. Ce compteur est incrémenté à chaque passage d'un réseau (pour rappel, chaque réseau est testé 5 fois) afin, encore une fois, d'éviter l'aléatoire de la JVM. Le nombre de TimeOut présenté est donc la somme des TimeOut pour les 5 essais.

2 Mesures Effectués

2.1 Résultats obtenus

Taux de réseaux satisfaits et temps d'exécution

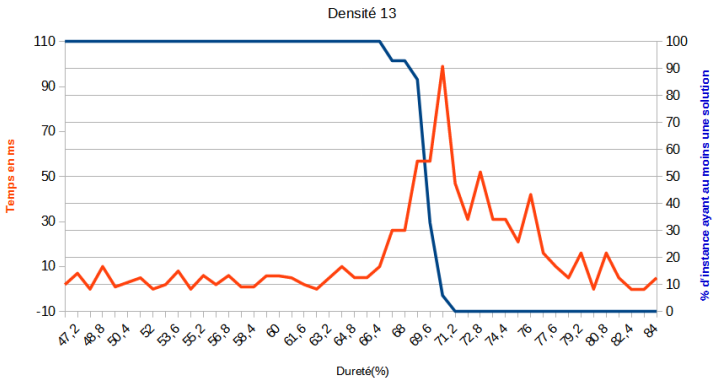
Ci-dessous les courbes obtenus sur lesquelles sont représentés le taux de réseaux ayant au moins une solution ainsi que le temps d'exécution (en ms) en fonction de la dureté. Celle-ci est calculée, comme indiqué dans la partie précédente, en faisant décroître le nombre de tuples. Pour des données chiffrés plus détaillés concernant ces résultats, le lecteur est invité à se référer au fichier de résultats transmis dans le rendu.

Prise en compte des TimeOuts

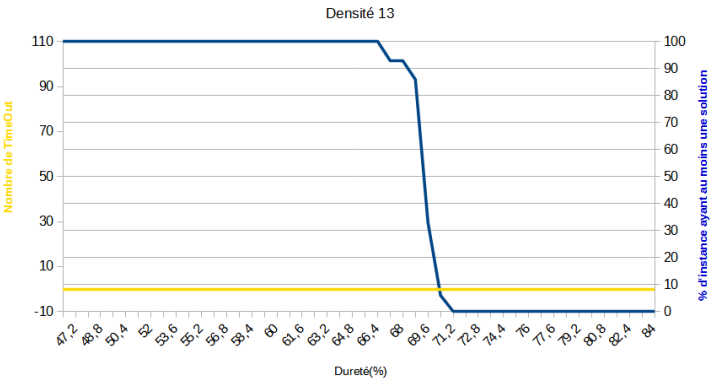
Sur une seconde courbe générée, il sera possible d'observer, là aussi pour chaque niveau de densité, le nombre de TimeOut enregistrés (multiplié par 5) en corrélation avec la transition de phase du taux de réseaux satisfaits, toujours en fonction du nombre de tuples. Ce paramètres va nous permettre de mesurer la difficulté de résolution des problèmes. Un grand nombre de TimeOut traduira ainsi la présence de problèmes pour lesquels il est compliqué et coûteux de trouver une solution (ou de trouver qu'il n'y a pas de solution).

Courbes de résultats

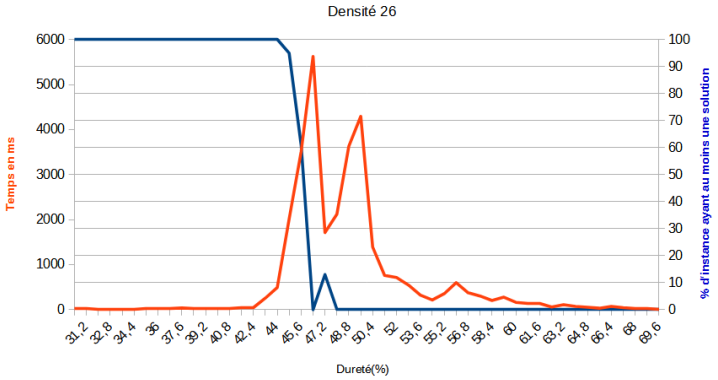
Diagrame du Taux de réseaux satisfaits et du temps d'exécution moyen en fonction de la dureté



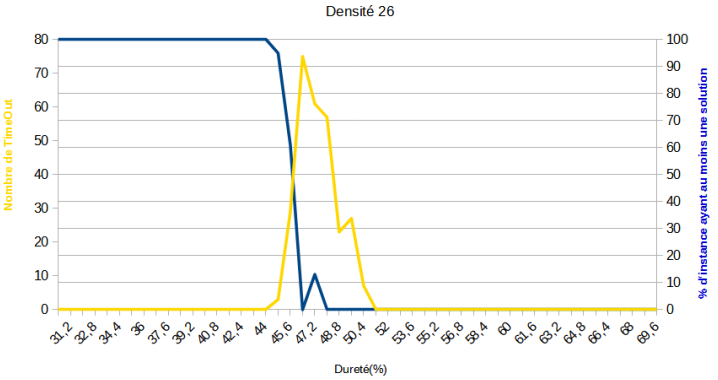
Diagrame du Taux de réseaux satisfaits et du nombre de TimeOut en fonction de la dureté



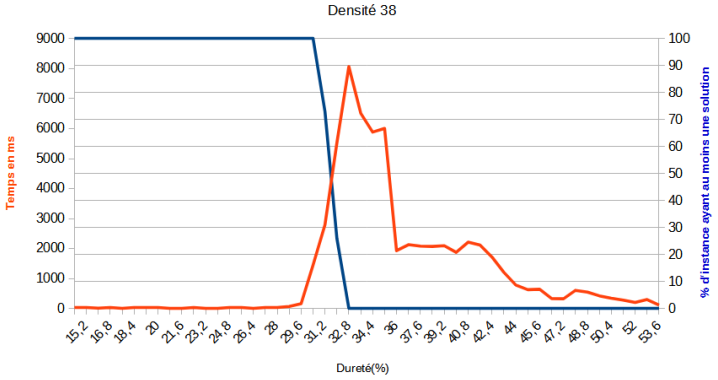
Diagrame du Taux de réseaux satisfaits et du temps d'exécution moyen en fonction de la dureté



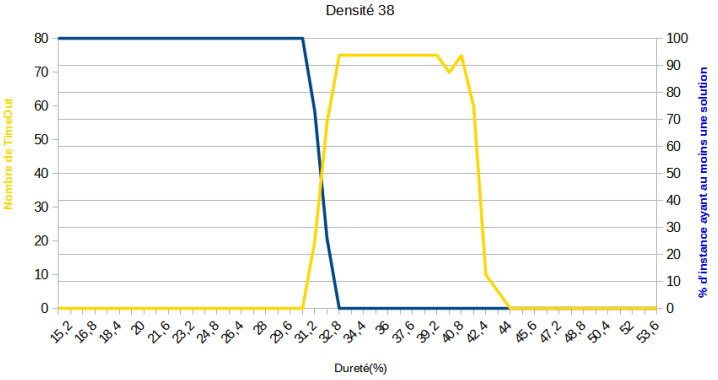
Diagrame du Taux de réseaux satisfaits et du nombre de TimeOut en fonction de la dureté



Diagrame du Taux de réseaux satisfaits et du temps d'exécution moyen en fonction de la dureté



Diagrame du Taux de réseaux satisfaits et du nombre de TimeOut en fonction de la dureté



2.2 Analyse des résultats

Plusieurs conclusions sont à tirés des résultats obtenus *(Voir Courbes de résultats en page 4)*. Premièrement, cela était à prévoir mais la transition de phase n'a pas lieu au même niveau de dureté pour chaque densité. En effet, en densité 13, celle-ci se produit entre 66% et 72% de dureté contre 41% à 49% pour la densité 26 et 29% à 33% pour la densité 38. Ceci est tout à fait logique car plus les réseaux sont denses, plus le problème est difficile à résoudre, la transition de phase se produit donc plus tôt et on arrive plus rapidement à des problèmes sans solutions avec une densité plus élevé.

Un second phénomène à observer est la corrélation entre le temps de calcul moyen et la transition de phase. En effet le pic de temps de calcul se situe généralement au milieu de la transition de phase. Ceci est normal car c'est à ce moment que nous étudions des problèmes "intéressants" c'est à dire ni trop faciles ni trop difficiles à résoudre. On remarque cependant que le temps de calcul a tendance à rester élevé pendant environ 10% de dureté après la transition de phase. Ici, les problème sont insolubles et difficiles à résoudre, cependant il n'est pas encore facile de s'apercevoir que ceux-ci sont trop difficiles, ce qui explique ce temps de calcul élevé. Sans surprise, le temps de calcul est très bas pour des problèmes très faciles comme pour des problèmes bien trop difficiles dont on arrive facilement à prouver l'insatisfiabilité.

Enfin, le nombre de TimeOut est particulièrement corrélé au temps d'exécution moyen. Il est cependant intéressant de relevé que pour une densité faible, le programme n'a jamais atteint de TimeOut et en a toujours été loin (maximum de 90ms en moyenne). La résolution d'un réseau d'une si faible densité est donc une opération très rapide pour laquelle la mise en place d'une telle sécurité n'est pas nécessaire.