Recherche tabou et planification de rencontres sportives

Tabu Search and sports league scheduling

Jean-Philippe Hamiez[1)] et Jin-Kao Hao[2)]

1) LGI^{2}P / EMA - EERIE - Parc Scientifique George BESSE - 30035 NÎMES CEDEX 01
email : hamiez@site-eerie.ema.fr

2) LERIA / Université d'Angers - 2 bd. Lavoisier - 49045 ANGERS CEDEX 01
email : Jin-Kao.Hao@univ-angers.fr

Résumé

Nous présentons dans cet article une approche tabou pour un type précis de problèmes de planification de tournois sportifs. Cette approche est fondée sur une formalisation du problème en CSP. Des tests sont effectués sur des problèmes jusqu'à 24 équipes représentant 276 variables avec 276 valeurs par variable[1]. Les expérimentations montrent que cette approche dépasse nettement certaines approches connues et fait partie des méthodes les plus intéressantes pour ce type de problèmes.

Mots clef

Planification de rencontres sportives, satisfaction de contraintes, méthodes heuristiques, recherche tabou.

Abstract

We present in this paper a Tabu approach for a version of the Sports League Scheduling problem. This approach is based on a formulation of the problem as a CSP. Tests are carried out on problems going up to 24 teams representing 231 variables with 231 value per variable[2]. Experimental results show that this approach outperforms some known methods and is among the most interesting methods for this type of problems.

Keywords

Sport Scheduling, Contraint Satisfaction, Heuristic Methods, Tabu Search.

1 Introduction

De nombreuses fédérations sportives (de football, hockey ou basket-ball par exemple) sont confrontées à des problèmes de planification de matches lors de tournois. Les règles qui régissent ces tournois sont autant de contraintes à prendre en compte avec les exigences des équipes, les souhaits des sponsors, etc. : minimisation des distances de déplacement [2][4], un seul match par équipe et par jour, indisponibilité de certains stades à certaines dates, nombre minimal de jours entre le match aller et le match retour correspondant, etc. La création de calendrier respectant ces conditions est donc un problème très difficile.

Les approches utilisées pour résoudre certains de ces problèmes sont nombreuses, avec des performances variables : programmation mathématique avec la programmation linéaire en nombres entiers [10][14], programmation par contraintes [12], recherche locale (recuit simulé [20], tabou [23], recherches hybrides [5]).

Nous considérons dans cet article un problème particulier de planification de tournois sportifs (SLSP, pour *Sports League Scheduling Problem*) décrit par K. McAloon, C. Tretkoff et G. Wetzel dans [13]. Après avoir testé, avec des résultats limités, une première approche par la programmation linéaire en nombres entiers avec contraintes de cardinalité, ils ont expérimenté l'approche de la programmation par contraintes. Cette approche a produit de meilleurs résultats. Enfin, ils proposent une troisième approche avec un algorithme de recherche locale basique aussi performant qu'ILOG Solver mais plus rapide.

J. C. Régin propose deux approches avec la programmation par contraintes [17][18]. La première, utilisant des algorithmes de filtrage puissants [3][15][16], fournit déjà de meilleurs résultats que ceux de McAloon et al., aussi bien en terme de rapidité d'exécution que de robustesse puisqu'il résout des problèmes de taille plus importante. Dans sa seconde approche, il transforme le problème en un problème équivalent en ajoutant une contrainte d'implication. Avec une nouvelle heuristique et des algorithmes de filtrages spécifiques à chaque type de contraintes, ses résultats sont encore meilleurs et très impressionnants.

L'objectif de cette étude consiste à proposer une approche de recherche locale avancée fondée sur la méthode tabou pour le SLSP. Les références initiales de cette étude sont les résultats présentés dans [13] en 1997 et ceux dans [17] en janvier 1998. Nous montrons que la recherche tabou [11] est une approche intéressante par sa simplicité et performante car les premiers résultats rejoignent les résultats de référence.

1. Nous présentons brièvement en Annexe 1 une autre stratégie qui nous a permis d'étendre les tests jusqu'à 40 équipes.
2. We briefly present in Annexe 1 another strategy which allowed us to go up to 40 teams.

L'article s'organise de la manière suivante : nous commençons par définir formellement le problème traité puis nous le modélisons sous forme d'un problème de satisfaction de contraintes (CSP) [21]. Nous poursuivons en présentant la méthode tabou utilisée et les résultats obtenus en les comparant avec ceux d'une forme simple de descente [1] et d'un algorithme tabou de base. Nous terminons en discutant des observations effectuées pendant ce travail puis concluons.

2 Description du problème

Nous baserons le reste du document sur les contraintes et définitions suivantes, les mêmes que dans [13] :

- un tournoi regroupe un ensemble T d'équipes (|T| pair) ;
- chaque équipe rencontre toutes les autres exactement une fois. La notion de matches aller/retour n'est pas traité dans ce problème ;
- le tournoi se déroule sur |T|-1 semaines. Chaque équipe joue une et une seule fois chaque semaine ;
- la programmation d'une rencontre se déroule sur un terrain, à une certaine date. Nous emploierons le terme générique de période, au nombre de |T|/2. A chaque période de chaque semaine correspond un match. Aucune équipe ne peut jouer plus de deux fois sur une période.

Le problème est de déterminer une programmation qui respecte l'ensemble de ces contraintes.

Le Tableau 1 montre un exemple de planification vérifiant l'ensemble des contraintes pour |T|=8 équipes numérotées de 0 à 7 (il y a donc 7 semaines et 4 périodes).

	Période 0	Période 1	Période 2	Période 3
Semaine 0	0 vs 1	2 vs 3	4 vs 5	6 vs 7
Semaine 1	0 vs 2	1 vs 7	3 vs 5	4 vs 6
Semaine 2	4 vs 7	0 vs 3	1 vs 6	2 vs 5
Semaine 3	3 vs 6	5 vs 7	0 vs 4	1 vs 2
Semaine 4	3 vs 7	1 vs 4	2 vs 6	0 vs 5
Semaine 5	1 vs 5	0 vs 6	2 vs 7	3 vs 4
Semaine 6	2 vs 4	5 vs 6	0 vs 7	1 vs 3

Tableau 1 - *Exemple de planification pour 8 équipes.*

Comme dans le tableau précédent, une configuration peut être représentée sous la forme d'un tableau à deux dimensions avec les semaines en lignes et les périodes en colonnes.

Chaque colonne vérifie la contrainte de cardinalité pour laquelle aucune équipe n'apparaît plus de deux fois. Sur chaque ligne chaque équipe apparaît exactement une fois. Ce qui est équivalent à dire que toutes les équipes d'une ligne sont différentes. Enfin, il existe une contrainte globale sur tout le tableau : chaque match n'y apparaît qu'une seule fois, i. e. tous les matches sont différents.

3 Formulation

Pour modéliser le SLSP, nous suivons l'approche de la programmation par contraintes. Nous considérons donc ce problème comme un problème de satisfaction de contraintes.

3.1 Problème de satisfaction de contraintes

Un problème de satisfaction de contraintes [21] est défini par un triplet (X, D, C) avec :

- X, ensemble fini de n variables : $X = \{X_1, ... , X_n\}$;
- une collection finie D de domaines associés aux variables : $D = \{D_1, ... , D_n\}$. Chaque domaine D_i représentant l'ensemble fini des valeurs possibles de la variable X_i ;
- un ensemble fini C de p contraintes : $C = \{C_1, ... , C_p\}$. Chaque contrainte est définie sur un ensemble de variables et spécifie quelles combinaisons de valeurs sont compatibles pour celles-ci.

Le problème consiste donc à trouver toutes les valeurs à assigner aux variables telles qu'elles satisfassent toutes les contraintes ; on parle dans ce cas de configuration consistante. L'ensemble de toutes les configurations possibles (consistantes ou non) est le produit cartésien $D_1 \times ... \times D_n$.

Résoudre un problème de satisfaction de contraintes consiste donc à déterminer une instanciation particulière (si elle existe) parmi un large éventail d'instanciations possibles.

Ce modèle est suffisamment général et puissant qu'il peut s'appliquer entre autres à des problèmes aussi connus que la k-coloration et la satisfiabilité.

3.2 Représentation

Nous utiliserons les notations suivantes pour formuler le SLSP en un problème de satisfaction de contraintes :

- P : ensemble des périodes, |P| = |T|/2 ;
- W : ensemble des semaines, |W| = |T|-1 ;
- t_n : équipe numéro n, $t_n \in T$, $1 \le n \le |T|$;
- $x(t_m, t_n)$: planification du match opposant les équipes t_m et t_n. Les valeurs d'une variable de ce type sont de la forme $(p_{m,n}, w_{m,n})$, indiquant respectivement la période et la semaine de planification du match, $p_{m,n} \in P$, $1 \le p_{m,n} \le |P|$, $w_{m,n} \in W$, $1 \le w_{m,n} \le |W|$.

L'ensemble des variables est alors $X = \{x(t_m, t_n), 1 \le m < n \le |T|\}$. Tous les domaines sont égaux à $D = \{(p_{m,n}, w_{m,n}), p_{m,n} \in P, 1 \le p_{m,n} \le |P|, w_{m,n} \in W, 1 \le w_{m,n} \le |W|\}$; $\forall x \in$

X, $D_x = D$. L'ensemble C des contraintes contient les trois différents types suivants de contraintes :

- unicité de toutes les équipes sur toutes les semaines. Pour chaque équipe t_n, $t_n \in T$, $1 \leq n \leq |T|$, on impose la contrainte :

  $\text{SEMAINE}(t_n) \Leftrightarrow w_{m,n} \neq w_{q,n}$, $\forall (m, q) \in [1, |T|]^2$, $m \neq n$, $q \neq n$ et $m \neq q$;

- chaque équipe ne doit pas jouer plus de deux fois sur une période. Pour chaque équipe t_n, $t_n \in T$, $1 \leq n \leq |T|$ et chaque période, on impose la contrainte :

  $\text{PERIODE}(t_n) \Leftrightarrow |\{p_{m,n} = p_{q,n}, (m, q) \in [1, |T|]^2, m \neq n, q \neq n \text{ et } m \neq q\}| \leq 1$.

- tous les matches du tournoi doivent être différents. Pour chaque match $< t_m, t_n >$, $(t_m, t_n) \in T^2$, $t_m \neq t_n$, on impose la contrainte[3] :

  $\text{ALLDIFF}(< t_m, t_n >) \Leftrightarrow < t_m, t_n > \neq < t_q, t_r >$, $\forall (t_q, t_r) \in T^2$, $t_q \neq t_r$.

4 Recherche tabou

La recherche tabou est une méthode de recherche locale avancée qui fait appel à un ensemble de règles et de mécanismes généraux pour guider la recherche de manière intelligente. Nous renvoyons le lecteur à [11] pour une description formelle de la méthode. Nous définissons maintenant les composantes de notre recherche tabou.

4.1 Espace de recherche - Configuration

Comme le représente clairement le Tableau 1, une configuration est une affectation complète des éléments de $D = \{(p_{m,n}, w_{m,n}), p_{m,n} \in P, 1 \leq p_{m,n} \leq |P|, w_{m,n} \in W, 1 \leq w_{m,n} \leq |W|\}$ aux variables de $X = \{x(t_m, t_n), 1 \leq m < n \leq |T|\}$: une configuration est donc un tableau de taille $|W| \times |P|$ dont les éléments sont des couples d'entiers (m, n), $1 \leq m < n \leq |T|$. Pour $|T| = 22$, cela représente un problème contenant 231 variables avec 231 valeurs par variable.

Il y a $|T|/2*(|T|-1)$ matches à planifier. Une solution peut être interprétée comme une permutation de ces matches. La taille de l'espace de recherche est donc $(|T|/2*(|T|-1))!$ En d'autres termes, la taille de l'espace de recherche croît d'un facteur du carré de $|T|/2$.

4.2 Voisinage

Soit s une configuration de l'espace de recherche S, le voisinage $N : S \rightarrow 2^s$ est une application telle que pour tout $s \in S$, $s' \in N(s)$ si et seulement si s et s' ne diffèrent que par les valeurs d'un seul couple de variables dont une au moins est en conflit[4].

Une configuration voisine de s peut donc être obtenue par le simple échange des valeurs courantes de deux variables quelconques, dont l'une au moins est en conflit dans s. Un mouvement peut donc être caractérisé par le couple $< x(t_m, t_n), x(t_q, t_r) >$. Concrètement, cela revient à échanger deux matches. Il est très clair que la taille du voisinage évolue tout au long de la recherche, en fonction du nombre de conflits.

4.3 Fonction de coût

Pour comparer, en terme de qualité, deux configurations s et s' de S, on définit une fonction de coût qui est en fait une relation d'ordre sur S. Nous allons appliquer des pénalités à deux des contraintes.

Soit $\text{Occ}P_s(p, t_n)$, le nombre d'occurrences de l'équipe t_n sur la période p dans la configuration s. Pour une contrainte du type PERIODE (une équipe peut jouer au plus deux fois sur une période), la pénalité $f_P(s)$ sera le nombre total de fois où les équipes jouent en trop sur les périodes, soit :

$$f_P(s) = \sum_{p=1}^{|P|} \sum_{n=1}^{|T|} \chi_P(s, n, p),$$

$$\chi_P(s, n, p) = \begin{cases} 0 & \text{si } \text{Occ}P_S(p, t_n) \leq 2 \\ \text{Occ}P_S(p, t_n) - 2 & \text{sinon} \end{cases}$$

De façon similaire, $\text{Occ}W_s(w, t_n)$, désigne le nombre d'occurrences de l'équipe t_n sur la semaine w dans la configuration s. La pénalité $f_W(s)$ d'une contrainte du type SEMAINE (une équipe ne peut jouer qu'une seule fois par semaine) sera donc :

$$f_W(s) = \sum_{w=1}^{|W|} \sum_{n=1}^{|T|} \chi_W(s, n, w),$$

$$\chi_W(s, n, w) = \begin{cases} 0 & \text{si } \text{Occ}W_S(w, t_n) \leq 1 \\ \text{Occ}W_S(w, t_n) - 1 & \text{sinon} \end{cases}$$

On remarquera que, dans les deux cas, la pénalité est exactement le nombre minimum de variables à changer dans s pour satisfaire les contraintes.
La fonction de coût est alors définie par la somme pondérée des pénalités des deux contraintes PERIODE et SEMAINE de C :

$$\forall s \in S, f(s) = \text{poids}(W) * f_W(s) + \text{poids}(P) * f_P(s),$$

où poids(P) et poids(W) sont les fonctions de poids respectives des deux contraintes PERIODE et SEMAINE de C, poids : $C \rightarrow R$. Elles sont déterminées de manière empirique[5].

3. Cette contrainte est toujours vérifiée dans l'algorithme que nous proposons.
4. Une variable est dite en conflit si elle est impliquée dans une contrainte non satisfaite.
5. Il semblerait que les meilleures valeurs soient 1 pour PERIODE et 2 pour SEMAINE.

Résoudre le SLSP revient donc à rechercher une configuration de coût nul.

4.4 Choix d'un meilleur voisin

A chaque itération l'algorithme examine la totalité du voisinage pour effectuer un mouvement. Il est donc impératif de pouvoir évaluer rapidement les coûts des voisins d'une configuration. Nous utilisons pour cela une technique s'inspirant de [9].

Soit δ une matrice de |X|*|X| cases. Chaque case $\delta[x(t_m, t_n), x(t_q, t_r)]$ représente la variation de la fonction coût qui devra être appliquée si le mouvement (échange des matches $< t_m, t_n >$ et $< t_q, t_r >$) correspondant à la case de la matrice est effectué. Ainsi, le coût de $s' \in N(s)$ est obtenu immédiatement par la simple somme du coût de s et de la valeur de la case correspondante de δ (en O(1)).

Pour obtenir le meilleur voisin, il suffit de parcourir un sous-ensemble de la matrice δ en temps O(|N(s)|). Après chaque mouvement, la matrice δ est mise à jour en temps $O(|X|\sqrt{|X|})$ dans le pire des cas.

4.5 Liste tabou

Un élément fondamental de la recherche tabou est l'utilisation d'une mémoire flexible, à court terme, qui garde une certaine trace des dernières opérations passées. On peut y stocker des informations pertinentes à certaines étapes de la recherche pour en profiter ultérieurement. Cette liste permet d'empêcher les blocages dans les minima locaux en interdisant de passer à nouveau sur des configurations de l'espace de recherche précédemment visitées.

Dans cette optique, nous considérons les entrées de notre liste tabou comme des couples de matches. En effet, pour passer d'une configuration actuelle s à une configuration s', le voisinage que nous avons choisi oblige à échanger deux matches (dont l'un au moins en conflit). Une fois l'échange effectué, nous allons l'interdire -il sera donc tabou- pendant un certain nombre de mouvements.

La durée de cette interdiction, notons la k, appelée longueur ou teneur tabou, est l'un des paramètres les plus importants et aussi l'un des plus difficiles à déterminer. De plus, elle est liée à la nature du problème. Si sa valeur est trop élevée alors des configurations non visitées seront injustement inaccessibles et la capacité de la méthode à exploiter le voisinage sera réduite. Inversement, si sa valeur est trop faible alors la méthode risque fortement d'être bloquée dans un minimum local. La longueur tabou permet donc d'éviter tous les cycles de longueur inférieure ou égale à k.

Après avoir testé différentes manières de fixer la longueur de la liste tabou (aléatoire et bornée, statique...), nous avons opté pour une longueur dynamique pondérée, à pas aléatoire [7], prenant en compte une évaluation de la fonction de coût où les poids des deux types de contraintes sont tous deux égaux à 1. Ce qui revient à considérer le nombre total de conflits.

A une configuration $s \in S$ on associera la teneur tabou k_s définie par :

$$k_s = \alpha * f_1(s) + \text{rand}(g),$$
$$f_1(s) = f_W(s) + f_P(s),$$

Équation 1 - *Longueur dynamique de la liste tabou*.

rand(g), le pas aléatoire, renvoie un nombre entier aléatoire choisi uniformément dans [1..g], α pondère le nombre de conflits dans s.

Afin d'implémenter la liste tabou, nous utilisons une matrice |X|*|X| dont chaque case correspond à un mouvement et dont son contenu est affecté avec le nombre actuel d'itérations plus la longueur k de la liste tabou. Avec cette structure de données, une simple comparaison avec l'itération courante suffit pour savoir si un échange est tabou ou non.

Rappelons avant de poursuivre que si un mouvement identifié comme tabou conduit à une configuration meilleure que celles précédemment rencontrées alors son statut tabou est levé. C'est le principe d'aspiration le plus simple. Il rejoint néanmoins notre objectif de minimisation de la fonction de coût.

4.6 Diversification – Intensification

L'intensification et la diversification [11] sont des techniques intéressantes pour améliorer la puissance de la méthode tabou.

L'intensification mémorise des propriétés favorables des meilleures configurations afin de les utiliser ultérieurement. Ainsi, elle permet d'explorer des régions de l'espace de recherche proches de celles caractérisants des configurations à priori de bonne qualité visitées antérieurement. La diversification cherche à diriger la recherche vers des configurations inexplorées. Elle permet alors de générer des configurations qui diffèrent de manière significative de celles rencontrées auparavant.

L'algorithme que nous proposons comprend une intensification très simple : après un certain nombre de mouvements sans amélioration de la meilleure fonction de coût nous retournons vers la meilleure configuration passée. Ici encore, la difficulté est de déterminer formellement à partir de quand la recherche doit revenir au meilleur optimum. Un intervalle étroit risque de couper court une recherche éventuellement prometteuse ou, effet contraire mais espéré, accélérer la résolution. Inversement, un délai plus long peut aussi bien aboutir que ralentir la résolution. Les valeurs de ce paramètre sont fonctions du nombre |T| d'équipes et ont été déterminées de façon empirique.

Nous effectuons une diversification immédiatement après une intensification en enlevant tous les statuts tabous de manière à autoriser les mouvements qui étaient interdits

avant diversification. Nous redirigeons donc la recherche locale vers des régions de l'espace de recherche non encore explorées.

5 Résultats expérimentaux

Cette section présente les résultats expérimentaux de notre algorithme tabou[6] ainsi que ceux d'une descente[7] et d'un algorithme tabou simple[8] pour comparaison. Pour cela, il est nécessaire de présenter les conditions d'expérimentation et les critères de comparaisons.

5.1 Conditions de tests

Comme nous l'avons déjà fait remarquer, le réglage des paramètres de la méthode tabou est une étape indispensable extrêmement importante qui conditionne la qualité des résultats obtenus.

Pour essayer d'obtenir les meilleures valeurs possibles, un protocole de réglage a été suivi pour les paramètres de l'algorithme et pour chaque nombre |T| d'équipes. Pour chaque paramètre, on détermine d'abord grossièrement un intervalle de valeurs intéressantes. On recherche ensuite dans cet intervalle la valeur donnant le meilleur résultat, avec un nombre limité de mouvements. Cette valeur est alors retenue comme étant la meilleure valeur du paramètre pour le nombre |T| d'équipes concerné et est utilisée pour les tests finaux.

Le Tableau 2 donne pour chaque paramètre un intervalle de valeurs intéressantes. α et g servent à régler la longueur de la liste tabou (cf. Équation 1). ID (pour Intensification / Diversification) spécifie l'intervalle, en terme de nombre de mouvements, après lequel une opération d'intensification / diversification est effectuée. Enfin, MAXMV correspond au nombre maximal de mouvements (échanges de matches) autorisés ; l'algorithme s'arrêtant bien évidemment dès qu'un coût nul est atteint.

Paramètre	Min.	Max.
α	0.1	1
g	6	100
ID	50	500
MAXMV	50	50000

Tableau 2 - *Intervalles de valeurs des paramètres de notre algorithme.*

Soulignons le fait que, pour $|T| \leq 10$, notre algorithme n'a pas besoin d'intensifier/diversifier pour arriver à une solution. Ceci explique la même valeur minimale de 50 pour ID et MAXMV.

Rappelons que la configuration initiale est générée de manière aléatoire. On commence par créer tous les matches (au nombre de $|T|*(|T|-1)/2$), en respectant par construction la contrainte ALLDIFF. Puis, à chaque couple (w, p), $w \in W$, $p \in P$, nous affectons aléatoirement un match de T.

5.2 Critères de comparaisons

Nous avons retenu trois principaux critères pour effectuer l'étude comparative. Ces critères sont par ordre décroissant d'importance :

- **Nombre |T| d'équipes** : la robustesse d'un algorithme ; plus un algorithme résout un problème avec un nombre |T| élevé, plus il est efficace ;
- **Nombre de mouvement** : l'effort nécessaire à un algorithme pour trouver une solution, critère indépendant de la machine utilisée ;
- **Temps d'exécution** : le temps CPU utilisé par un algorithme pour réaliser un nombre de mouvements donné ; ce critère est dépendant de la machine et des options d'optimisation de compilation utilisées.

5.3 Résultats

Nous présentons dans cette partie les résultats comparatifs de la descente, du tabou de base et de notre algorithme tabou. Pour obtenir ces résultats, chacun des trois algorithmes est exécuté avec les mêmes configurations initiales pour 10 germes différents.

Tous les tests ont été réalisés sur station Sparc Ultra 1 à 143 MHz avec 256 Mo de mémoire. Les trois algorithmes sont implémentés en C et compilés avec le compilateur CC et l'option –O5.

Le Tableau 3 présente les résultats obtenus en terme de robustesse pour |T| dans [6..22]. Les colonnes 3 – 4, 5 – 6 et 7 – 8 indiquent respectivement les résultats d'une simple descente (Descente), d'une méthode tabou de base (Tabou-B) et de notre version évoluée de recherche tabou (Tabou-E). Pour chaque algorithme, on donne le nombre de succès (f = 0) et le nombre d'échecs (f > 0) pour 10 initialisations différentes de la configuration initiale.

|T|	MAXMV	Descente		Tabou-B		Tabou-E	
		f = 0	f > 0	f = 0	f > 0	f = 0	f > 0
6	50	4	6	9	1	10	-
8	50	-	10	9	1	10	-
10	200	-	10	9	1	10	-
12	400	-	10	4	6	10	-
14	2000	-	10	2	8	10	-
16	40000	-	10	5	5	10	-
18	50000	-	10	-	10	4	6
20	50000	-	10	-	10	6	4
22	50000	-	10	-	10	2	8

Tableau 3 - *Nombre de solutions trouvées pour 10 initialisations différentes.*

6. Nous présentons en Annexe 1 une autre stratégie et nos résultats les plus récents, jusqu'à 40 équipes.
7. Avec choix du meilleur mouvement dans le même voisinage que notre algorithme tabou et un tableau δ.
8. Sans intensification/diversification, ni pondération des contraintes, mais avec la même teneur tabou et le même voisinage que notre algorithme tabou et un tableau δ.

Le premier constat est que la descente -dans sa forme la plus simple- n'est pas aussi adaptée au SLSP que les méthodes tabous. En effet, elle ne résout que |T| = 6. Les expérimentations avec un nombre plus élevé d'exécutions n'ont rien donné ; la descente a échoué sur 50 initialisations différentes de la configuration initiale, en autorisant 10000 mouvements pour |T| = 12.

Tabou-B est une amélioration considérable de la descente car elle résout apparemment facilement le SLSP jusque |T|=16. Il a suffit d'ajouter simplement une structure interdisant les k derniers mouvements effectués pour arriver à un premier résultat intéressant.

Tabou-E, version évoluée de Tabou-B (avec pondération des contraintes et intensification/diversification), est plus robuste car elle résout plus de configurations initiales et qu'elle permet de résoudre le SLSP jusque |T|=22.

Le Tableau 4 présente les temps CPU moyens (en seconde) et les nombres moyens de mouvements nécessaires (#MV) à notre version de tabou pour résoudre le SLSP. Les temps sont donnés à titre indicatif car ils changent selon la machine utilisée et les structures de données employées. Nous donnons également le nombre de solutions trouvées (#R) pour #E configurations initiales différentes.

|T|	#R / #E	#MV	Temps (s)
6	10 / 10	18.8	0.03
8	10 / 10	25.1	0.12
10	10 / 10	98	0.84
12	10 / 10	268.3	3.74
14	10 / 10	561.4	11.54
16	10 / 10	14010.3	386.38
18	4 / 10	25992.5	955.49
20	6 / 10	26370	1031.77
22	2 / 10	34145.5	1649.34

Tableau 4 – *Performances de Tabou-E.*

Tabou-E n'a pas encore trouvé de solution pour |T|=24 bien que nous autorisions un nombre relativement élevé de mouvements (800000). La meilleure fonction de coût atteinte est de 1, après 18845 mouvements[9].

5.4 Comparaisons / McAloon & Régin

Nous présentons maintenant les résultats des travaux décrits dans [13] et ceux de [17][18] afin de comparer leurs résultats sur le SLSP avec notre algorithme de recherche tabou Tabou-E.

McAloon et al. [13] rappellent que la programmation linéaire en nombres entiers[10] [6] n'a pas été capable de résoudre le SLSP pour |T| = 14, même en évitant les symétries. La méthode fonctionne néanmoins pour |T| ≤ 12. Ils proposent alors un algorithme de programmation par contraintes développé sous ILOG Solver qui résout l'instance |T| = 14 équipes en 45 minutes sur une station UltraSparc. Ce résultat est le premier obtenu avec la programmation par contraintes. Ils testent également un algorithme de programmation logique sous contraintes qui échoue à |T| = 14. Enfin, avec un algorithme basique de recherche locale, ils résolvent |T| = 14 en 10 minutes.

Aux vues de ces résultats, notre algorithme est plus performant que ces trois méthodes, aussi bien en terme de robustesse que de rapidité, puisque qu'il traite des instances allant jusqu'à |T| = 22 et résout |T| = 14 en moins de 12 secondes en moyenne.

|T|	#Echecs	Temps (s)
4[11]	2	0.01
6	12	0.03
8	32	0.08
10	417	0.8
12	41	0.2
14	3514	9.2
16	1112	4.2
18	8756	36
20	72095	338
22	6172672	10h
24	6391470	12h

Tableau 5 – *Résultats de la première méthode de Régin.*

Une deuxième approche de programmation par contraintes est développée en janvier 1998 dans [17]. C'était notre référence de départ. J. C. Régin y propose une première amélioration importante en utilisant des techniques avancées (dont l'élimination des symétries) et nouvelles (plusieurs algorithmes[12] de filtrage très puissants basés sur le principe de consistance d'arc [3][15][16]). Nous rappelons ses résultats dans le Tableau 5. Les colonnes 1, 2 et 3 précisent respectivement le nombre d'équipes, le nombre de retours arrière et le temps d'exécution en secondes.

Nos résultats rejoignent assez ces résultats de référence en terme de robustesse, même si pour l'instant Tabou-E n'a pas trouvé de solution pour |T| = 24. Les temps de résolution sont souvent comparables compte tenu des différences des machines utilisées. Remarquons néanmoins que pour 22 équipes Tabou-E est très rapide.

Très récemment, en mai 1999, une troisième approche de programmation par contraintes est présentée dans [18]. Cette approche utilise une heuristique d'instanciation des matches basée sur un théorème d'Euler pour les carrés latins. Le SLSP y est transformé en un problème équivalent en ajoutant une semaine fictive qui permet de déduire certaines inconsistances plus rapidement. Cette version évoluée comprenant des mécanismes très poussés obtient des résultats impressionnants. Nous les rapportons dans le tableau suivant :

9. Avec une nouvelle approche, présentée en Annexe 1, nous proposons une solution pour 24 équipes en 1 heure / 4 161 278 mouvements en moyenne.
10. Expérimentations réalisées avec Cplex, un logiciel commercial de programmation linéaire en nombres entiers.
11. Notons que pour |T| = 4, il n'existe pas de solution. Une simple énumération exhaustive suffit à le prouver.
12. Un par type de contraintes.

|T|	#Echecs	Temps (s)
8	10	0.01
10	24	0.06
12	58	0.2
14	21	0.2
16	182	0.6
18	263	0.9
20	226	1.2
22	157	1.2
24	2702	10.5
26	5683	26.4
28	32324	316
30	11895	138
40	2834754	6h

Tabou-E est ici largement dépassé par les performances de la deuxième méthode qui est devenue une nouvelle référence pour les études futures[13].

6 Discussion

Nous discutons ici de l'influence des paramètres de notre algorithme et du contenu de la liste tabou.

6.1 Influence des paramètres

Il est bien connu que les paramètres jouent un rôle primordial pour la performance des algorithmes de recherche locale. Ce point s'est confirmé lors de nos études.

Nous avons donné des intervalles de valeurs pour régler les paramètres de Tabou-E (cf. Tableau 2). Ces chiffres sont purement indicatifs et sont très variables. En effet, les résultats de nos expérimentations montrent clairement qu'il est très difficile de trouver des paramètres optimaux avec la seule donnée de |T|. Pour un |T| fixé il existe plusieurs réglages des paramètres possibles suivant la configuration initiale choisie aléatoirement. Ainsi, considérant une configuration initiale particulière s_0 pour 14 équipes et un nombre maximal de 2000 mouvements, Tabou-E ne trouve pas de solution avec α=0.4 ; α=0.6 trouve une solution en 1523 mouvements. Enfin, α=0.9 fournit une solution beaucoup plus rapidement, en 220 mouvements !

Ces observations, valables pour certains des autres paramètres, nous amènent à penser que la phase d'initialisation de la configuration de départ pourrait jouer un rôle important dans la rapidité et l'efficacité de la recherche[14]. Il est même fortement probable qu'un réglage quasi optimal des paramètres prendrait en compte certaines des propriétés des configurations. Nous pensons également que les paramètres ne demeurent pas quasi optimaux pendant toute la recherche et qu'il serait éventuellement intéressant de les faire évoluer à certaines étapes de la recherche.

6.2 Influence du contenu tabou

La puissance de la méthode tabou repose essentiellement sur une certaine notion d'historique qui empêche la recherche de retourner vers des configurations passées et qui permet de sortir des optima locaux.

Une des difficultés de la méthode est alors de déterminer l'information la plus intéressante à stocker dans la mémoire tabou.

Nous avons essayé trois contenus tabous différents pour Tabou-B, la méthode tabou de base et pour Tabou-E, notre version évoluée de Tabou-B. Dans un premier temps, nous considérons simplement un couple de matches comme tabou (v1). Ensuite, nous déclarons tabou tout échange avec un seul des deux matches (v2). Enfin, nous qualifions de tabou tout échange avec l'un ou l'autre des deux matches (v3).

L'idée principale est ici de faire apparaître le caractère plus ou moins contraignant d'un état tabou. En effet, on peut parler d'un certain ordre dans les trois versions de statut tabou définis précédemment : v1 < v2 < v3. v2 (respectivement v3) interdit le passage à plus de configurations voisines de la configuration courante que v1 (respectivement v2).

Les quelques expériences effectuées donnent des résultats qui suivent l'ordre précédemment défini. Tabou-B et Tabou-E sont moins performants avec v2 et v3 ; Tabou-E donnant néanmoins toujours de meilleurs résultats que Tabou-B. Cette observation nous conforte dans l'intérêt d'utiliser une pondération des contraintes et un processus d'intensification / diversification, seules différences entre Tabou-B et Tabou-E.

7 Conclusion

Dans ce travail, nous avons effectué une étude expérimentale pour un problème précis de la planification de rencontres sportives que nous avons modélisé sous forme de CSP. Nous avons présenté un algorithme de recherche tabou dont l'efficacité repose sur les éléments suivants :

- une configuration et un voisinage simples ;
- une structure de données accélérant la recherche ;
- une longueur tabou dynamique ;
- une pondération des contraintes ;
- un mécanisme d'intensification / diversification.

Les résultats empiriques des expérimentations ont montré que Tabou-E est plus robuste que la programmation linéaire en nombres entiers qui se limite à $|T| \leq 12$ alors que notre version de recherche tabou résout le SLSP jusqu'à 22 équipes. Les résultats de Tabou-E rejoignent ceux de la première méthode de programmation par contraintes présentée dans [17], qui était la référence initiale de cette étude. Pour 22 équipes, notre algorithme

13. La nouvelle approche que nous décrivons en Annexe 1 rejoint ces résultats.
14. C'est en suivant cette idée que nous avons développé une nouvelle stratégie plus robuste capable de résoudre le SLSP jusqu'à 40 équipes (cf. Annexe 1).

fournit de plus un temps moyen de résolution considérablement meilleur. Néanmoins, ces résultats sont très loin derrière les tout récents résultats présentés dans [18]. (Notons que notre dernière approche présentée en Annexe 1 rejoint ces résultats.)

Ce travail a montré que des structures de données performantes sont indispensables pour obtenir une bonne efficacité de l'algorithme. Enfin, il confirme que le réglage des paramètres est crucial pour l'obtention de solutions et qu'un réglage optimal reste une tâche importante.

Une étude plus approfondie sur les structures des configurations, d'autres voisinages et leurs liens avec les différents paramètres de la recherche tabou reste à établir pour espérer parvenir à un algorithme plus robuste.

Il reste à savoir si ces résultats peuvent être généralisés à d'autres problèmes de planification de rencontres sportives, par exemple aux problèmes de minimisation du nombre total de breaks[15] (ainsi nommés dans [22]). La recherche tabou a déjà montré son efficacité pour certains types de problèmes de planification de rencontres sportives, avec des contraintes de différents types. Une recherche tabou hybride [5] est ainsi proposé pour le problème de la National Hockey League, montrant l'efficacité d'une alliance recherche tabou / algorithmes génétiques ; le mélange habile d'éléments des deux méthodes donnant de meilleurs résultats que ceux de chacune des deux méthodes prise séparément.
Les travaux déjà effectués sur l'application de la méthode tabou à des problèmes de planification de rencontres sportives nous confortent dans l'idée qu'elle constitue une approche intéressante pour la résolution pratique des problèmes et des applications de contraintes et qu'elle mérite d'être étudiée davantage. Une voie de recherche possible, et qui plus est intéressante, pourrait consister en une hybridation des techniques avancées héritées de la programmation par contraintes et d'un algorithme évolué de recherche tabou.

Remerciements

Les auteurs tiennent à remercier J. C. Régin pour leur avoir transmis ses derniers résultats ainsi que les relecteurs de cet article pour leurs remarques pertinentes.

Références

[1] D. H. Ackley, A connectionist machine for genetic hillclimbing, *Kluwer Academic Publishers*, 1987.

[2] J. C. Bean et J. R. Birge, Reducing travelling costs and player fatigue in the National Basketball Association, *Interfaces* 10, pp. 98-102, 1980.

[3] C. Bessière et J. C. Régin, Arc consistency for general constraint networks : preliminary results, *IJCAI* (1), pp. 398-404, 1997.

[4] R. T. Campbell et D. S. Chen, A minimun distance basketball scheduling problem, *Management Science in Sports*, R. E. Machol, S. P. Ladany et D. G. Morrison (Eds.), North-Holland, Amsterdam, pp. 15-25, 1976.

[5] D. Costa, An evolutionary tabu search algorithm and the NHL scheduling problem, *INFOR* 33 (3), pp. 161-178, 1995.

[6] CPLEX Optimization, Using the CPLEX©Callable Library, Tahoe, NV, Inc. 1995.

[7] R. Dorne et J. K. Hao, Tabu search for graph coloring, T-colorings and Set T-colorings, *Meta-heuristics : Advances and Trends in Local Search Paradigms for Optimization*, Chapitre 6, pp. 77-92, S. Voss et al. (Eds.), Kluwer Academic Publishers, 1998.

[8] S. Fiorini et R. J. Wilson, Edge-Colourings of graphs, *Research Notes in Mathematics* 16 (Pitman publishing, London), 1977.

[9] C. Fleurent et J. A. Ferland, Genetic and hybrid algorithms for graph coloring, G. Laporte, I. H. Osman et P. L. Hammer (Eds.), Special Issue of Annals of Operations Research, *Metaheuristics in Combinatorial Optimization*.

[10] C. Fleurent et J. A. Ferland, Allocating games for the NHL using integer programming, *Operations Research* 41 (4), pp. 649-654, 1993.

[11] F. Glover et M. Laguna, Tabu Search, Kluwer Academic Publishers, 1997.

[12] M. Henz, Scheduling a major college basketball conference – revisited, à paraître dans *Operations Research*, 1998.

[13] K. McAloon, C. Tretkoff et G. Wetzel, Sports league scheduling, *Proceedings of the 1997 ILOG Optimization Suite International Users' Conference*, 1997.

[14] G. L. Nemhauser et M. A. Trick, Scheduling a major college basketball conference, *Operations Research* 46 (1), 1998.

[15] J. C. Régin, A filtering algorithm for constraints of difference in CSPs, *AAAI*, Vol. 1, pp. 362-367, 1994.

[16] J. C. Régin, Generalized arc consistency for global cardinality constraint, *AAAI* / *IAAI*, Vol. 1, pp. 209-215, 1996.

15. Deux matches consécutifs à domicile (respectivement à l'extérieur), pour une équipe.

[17] J. C. Régin, Modeling and solving sports league scheduling with constraint programming, *1er Congrès de la Société Française de Recherche Opérationnelle et Aide à la Décision (ROAD F)*, Paris, France, 14-16 janvier 1998.

[18] J. C. Régin, Sports scheduling and constraint programming, *INFORMS*, Cincinnati, mai 1999.

[19] J. A. M. Schreuder, Constructing timetables for sport competitions, *Mathematical Programming Study* 13, pp. 58-67, 1980.

[20] B. J. Terril et R. J. Willis, Scheduling the Australian state cricket season using simulated annealing, *Journal of the Operational Research Society* 45, pp. 276-280, 1994.

[21] E. Tsang, Foundations of Constraint Satisfaction, *Academic Press*, 1993.

[22] D. de Werra, Geography, games and graphs, *Discrete Applied Mathematics* 2, pp. 327-337, 1980.

[23] M. Wright, Timetabling county cricket fixtures using a form of tabu search, *Journal of the Operational Research Society* 45, pp. 758-770, 1994.

Annexe 1

Nous présentons brièvement ici une autre stratégie pour le SLSP, toujours basée sur un algorithme tabou, qui utilise une technique de réduction de l'espace de recherche. L'étape de réduction utilise deux résultats de la théorie des graphes [8][19] pour considérer les contraintes *SEMAINE* et *ALLDIFF* comme des invariants dans le processus de la recherche locale : ces contraintes sont vérifiées par construction de la configuration initiale au départ de l'algorithme et le restent pendant toute la résolution.

La recherche taboue tente ensuite de découvrir un tournoi valide dans un espace de recherche et un voisinage réduits où les configurations non valides ne contiennent que des conflits engendrés par la contrainte du type *PERIODE*.

Pour définir notre configuration initiale, nous construisons un graphe complet à |T|-1 sommets où chaque sommet est une des |T|-1 premières équipes. A chaque arête correspond alors un unique match ; ce qui satisfait la contrainte *ALLDIFF*. Ce graphe est représenté sous la forme d'un polygone régulier. Chaque ensemble d'arêtes parallèles dans le polygone nous donne (|T|/2)-1 matches pour chaque semaine. Le dernier match de chaque semaine est trivialement celui qui oppose les équipes manquantes. Cette construction permet alors de satisfaire la contrainte *SEMAINE* car les arêtes parallèles n'ont pas de sommets en commun.
Pendant la recherche locale les seuls mouvements autorisés sont restreints aux matches d'une même semaine afin de conserver l'invariant *SEMAINE*.

Nous donnons dans le tableau suivant les meilleurs résultats connus[16] et les résultats moyens de notre nouvelle approche, en précisant, pour notre approche, le nombre total d'essais avec le nombre d'échecs entre parenthèses[17].

|T|	Meilleurs résultats connus		Résultats de notre nouvelle approche		
	Temps (s)	#Echecs	Essais	Temps	#Mvts.
16	0.6	182	10(2)	2.3 s	5 171
18	0.9	263	10(3)	1.6 s	2 890
20	1.2	226	10(0)	16 min	1 443 069
22	1.2	157	10(1)	18 min	1 360 879
24	10.5	2 702	10(2)	1 h	4 161 278
26	26.4	5 683	10(5)	1 h	4 097 359
28	316	32 324	5(3)	35 min	1 763 616
30	138	11 895	5(4)	1 h	3 034 073
32	-	-	5(4)	2.5 h	6 387 470
34	-	-	5(3)	1 h	2 336 530
36	-	-	5(4)	4.5 h	9 307 524
40	6 heures	2 834 754	5(4)	2 min	68 746

L'annexe suivante donne la solution obtenue pour 40 équipes.

16. Nous ne disposons pas des meilleurs résultats connus pour 32 à 38 équipes.
17. Le temps moyen pour |T| = 40 est à interpréter avec précaution.

Annexe 2 (exemple de planification valide pour 40 équipes)

Périodes	Semaines 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
1	1 2	2 3	3 4	4 5	5 6	6 7	7 8	8 9	9 10	10 11	11 12	32 40	27 39	14 15	15 16	16 17	17 18	18 19	19 20	20 21	21 22	22 23	23 24	24 25	25 26	1 13	27 28	28 29	29 30	30 31	12 40	32 33	33 34	34 35	35 36	36 37	37 38	38 39	14 26
2	3 39	1 4	2 5	3 6	19 31	5 8	6 9	22 34	8 11	9 12	10 13	11 14	12 15	13 16	14 17	15 18	7 28	38 40	18 21	10 31	20 23	21 24	22 25	23 26	24 27	25 28	26 29	27 30	4 16	29 32	30 33	7 19	32 35	33 36	34 37	17 40	36 39	1 37	2 38
3	8 34	5 39	10 36	11 37	3 8	4 9	23 31	6 11	7 12	4 17	9 14	2 23	7 20	12 17	13 18	14 19	15 20	16 21	13 26	1 40	19 24	16 29	21 26	22 27	15 36	24 29	25 30	22 35	27 32	28 33	25 38	30 35	10 18	2 28	33 38	34 39	5 31	6 32	3 37
4	5 37	6 38	7 39	1 8	25 40	3 10	4 11	5 12	6 13	7 14	8 15	9 16	10 17	11 18	12 19	13 20	14 21	15 22	16 23	17 24	18 25	19 26	20 27	21 28	22 29	23 30	24 31	9 40	26 33	27 34	28 35	29 36	30 37	31 38	32 39	1 33	2 34	3 35	4 36
5	9 33	7 37	8 38	9 39	1 10	2 11	3 12	4 13	5 14	6 15	7 16	8 17	30 36	10 19	11 20	12 21	13 22	14 23	15 24	16 25	17 26	18 27	19 28	20 29	6 40	22 31	23 32	24 33	25 34	26 35	27 36	28 37	29 38	30 39	1 31	2 32	18 40	4 34	5 35
6	7 35	8 36	9 37	10 38	11 39	1 12	2 13	3 14	4 15	5 16	6 17	7 18	8 19	9 20	10 21	11 22	12 23	13 24	14 25	15 26	2 40	17 28	18 29	19 30	20 31	21 32	22 33	23 34	24 35	25 36	26 37	27 38	28 39	1 29	16 40	3 31	4 32	5 33	6 34
7	4 38	9 35	1 6	2 7	12 38	13 39	27 40	2 15	3 16	8 13	5 18	6 19	11 16	8 21	9 22	10 23	11 24	12 25	17 22	14 27	15 28	20 25	17 30	18 31	19 32	20 33	21 34	26 31	23 36	24 37	29 34	26 39	14 40	32 37	3 29	4 30	1 35	10 28	7 33
8	18 24	10 34	11 35	12 36	13 37	14 38	24 30	1 16	2 17	30 40	4 19	5 20	33 40	7 22	8 23	33 39	10 25	11 26	12 27	13 28	14 29	6 39	16 31	17 32	3 9	19 34	20 35	21 36	22 37	23 38	9 15	1 25	2 26	3 27	4 28	5 29	15 21	7 31	8 32
9	10 32	11 33	12 34	17 31	14 36	26 40	16 38	21 35	1 18	23 37	3 20	25 39	5 22	2 27	7 24	36 40	9 26	6 31	11 28	12 29	13 30	10 35	15 32	16 33	17 34	14 39	1 15	20 37	21 38	4 18	5 19	2 24	3 25	8 22	9 23	6 28	7 29	8 30	13 27
10	11 31	12 32	13 33	14 34	15 35	16 36	17 37	18 38	19 39	1 20	2 21	3 22	4 23	34 40	6 25	7 26	8 27	9 28	10 29	11 30	12 31	13 32	14 33	5 40	16 35	17 36	18 37	19 38	20 39	1 21	2 22	3 23	4 24	5 25	6 26	7 27	8 28	9 29	10 30
11	12 30	13 31	14 32	15 33	16 34	17 35	18 36	28 40	20 38	21 39	1 22	10 15	3 24	4 25	5 26	6 27	4 31	8 29	9 30	19 22	11 32	12 33	10 37	14 35	23 28	16 37	17 38	18 39	1 19	2 20	3 21	13 40	5 23	6 24	7 25	8 26	9 27	2 36	11 29
12	13 29	14 30	15 31	16 32	17 33	18 34	19 35	20 36	21 37	22 38	31 40	1 24	2 25	3 26	4 27	5 28	6 29	7 30	39 40	9 32	10 33	11 34	12 35	13 36	14 37	15 38	16 39	1 17	2 18	3 19	4 20	5 21	6 22	7 23	8 24	9 25	10 26	11 27	12 28
13	14 28	15 29	16 30	13 35	18 32	19 33	20 34	17 39	22 36	2 19	24 38	4 21	1 26	6 23	3 28	4 29	5 30	10 27	7 32	8 33	9 34	14 31	11 36	12 37	13 38	18 35	8 40	2 16	3 17	22 39	1 23	6 20	7 21	15 40	5 27	10 24	11 25	12 26	9 31
14	15 27	22 40	17 29	18 30	4 7	20 32	21 33	19 37	23 35	24 36	25 37	26 38	13 14	1 28	2 29	3 30	16 19	5 32	6 33	7 34	8 35	9 36	4 40	11 38	12 39	26 27	2 14	3 15	28 31	5 17	6 18	31 34	8 20	9 21	10 22	11 23	12 24	13 25	1 39
15	16 26	17 27	18 28	24 40	20 30	21 31	22 32	23 33	24 34	25 35	26 36	27 37	28 38	29 39	1 30	2 31	3 32	4 33	5 34	6 35	7 36	8 37	9 38	10 39	1 11	2 12	3 13	4 14	5 15	6 16	7 17	8 18	9 19	10 20	11 21	12 22	13 23	19 40	15 25
16	17 25	18 26	19 27	20 28	21 29	22 30	5 10	24 32	25 33	26 34	27 35	28 36	29 37	30 38	31 39	1 32	2 33	3 34	4 35	18 23	6 37	3 40	8 39	1 9	2 10	7 40	4 12	5 13	6 14	7 15	8 16	9 17	31 36	11 19	12 20	13 21	14 22	15 23	16 24
17	6 36	19 25	23 40	21 27	22 28	23 29	15 39	25 31	26 32	27 33	28 34	29 35	9 18	31 37	32 38	9 24	1 34	2 35	3 36	4 37	5 38	15 30	1 7	2 8	21 30	4 10	5 11	6 12	7 13	8 14	24 39	10 16	11 17	12 18	13 19	14 20	3 33	16 22	20 40
18	19 23	20 24	21 25	22 26	23 27	24 28	25 29	26 30	27 31	28 32	29 33	30 34	31 35	32 36	35 40	34 38	37 40	1 36	2 37	3 38	4 39	1 5	2 6	3 7	4 8	5 9	6 10	7 11	8 12	9 13	10 14	11 15	12 16	13 17	14 18	15 19	16 20	17 21	18 22
19	20 22	21 23	22 24	23 25	24 26	25 27	26 28	27 29	28 30	29 31	30 32	31 33	32 34	33 35	34 36	35 37	36 38	37 39	1 38	2 39	1 3	2 4	3 5	4 6	5 7	6 8	7 9	8 10	10 40	11 40	11 13	12 14	13 15	14 16	15 17	16 18	17 19	18 20	19 21

	20	21 40	16 28	20 26	19 29	2 9	15 37	1 14	7 10	29 40	3 18	23 39	12 13	6 21	5 24	33 37	8 25	35 39	17 20	8 31	5 36	16 27	7 38	13 34	15 34	18 33	3 11	19 36	25 32	9 11	10 12	31 32	4 22	1 27	4 26	2 30	35 38	6 30	14 24	17 23