Nº d'ordre : 06/2012 D /INF

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumediene
Faculté d'Électronique et d'Informatique

THÈSE

Présentée en vue de l'obtention du grade de DOCTORAT
EN : Informatique

Spécialité : Intelligence Artificielle et Bases de Données Avancées

Par: HADJIDJ Drifa ép. BELHI

Thème

Contribution à la Résolution du Problème d'Emploi du Temps par la Recherche Dispersée et la Recherche Locale Guidée

Soutenue publiquement, le 04 juillet 2012, devant le jury composé de :

M. MEZGHICHE Mohamed, Professeur à l'UMBB…… ……..…… Président
Mme DRIAS Habiba, Professeur à l'USTHB,…….....…Directrice de thèse
M. AIDER Méziane, Professeur à l'USTHB ……..……...Examinateur
M. KACEM Imed, Professeur à l'Université de Metz France ……...……Examinateur
M. AHMED_OUAMER Rachid, Maitre de conférences A à l'UMMTO.....Examinateur
Mme BOUGHACI Dalila, Maitre de conférences A à l'USTHB ……..Examinatrice
Mme TEBIBEL Thouraya, Maitre de conférences A à l'ESI ……...Examinatrice

A mes fils
Abdelhak et Ibrahim

Remerciements

Je tiens à remercier très chaleureusement et avec une profonde gratitude, ma directrice de thèse, Madame Habiba DRIAS, Professeur et Directrice du Laboratoire de Recherche en Intelligence Artificielle à l'Université des Sciences et de la Technologie Houari Boumediene, pour m'avoir donnée la possibilité de faire cette thèse, pour sa confiance, et ses remarques judicieuses.

Je remercie aussi Monsieur Mezghiche Mohamed, Professeur à l'université M'Hamed Bougara de Boumerdes, pour l'honneur qu'il me fait en acceptant de présider ce jury.

Je remercie également Monsieur Ahmed-Ouamer Rachid, Maitre de conférences A à l'UMMTO, Monsieur Aider Méziane, Professeur à l'USTHB, Madame Boughaci Dalila, Maitre de conférences A à l'USTHB, Monsieur Kacem Imed, Professeur à l'Université de Metz France, et Madame Tebibel Thouraya, Maitre de conférences A à l'ESI, pour avoir accepté de consacrer une partie de leur temps pour examiner ce travail et participer au jury.

Mes remerciements vont aussi à Monsieur le Doyen de la faculté des sciences de l'UMBB, Pr. Badari Kamel, à Monsieur le Chef du département d'informatique, Dr. Herzallah Abdelkrim et à Madame le Chef du département d'informatique de l'USTHB, Pr. Boukala Ioualalen Malika pour leurs encouragements.

Enfin, que mes collègues de l'USTHB et l'UMBB, mes amies Atika, Baya, Nacéra et Nadjia, ma famille et tous ceux qui m'ont aidée de prés ou de loin, trouvent ici mes sincères remerciements.

A toutes et à tous merci.

Résumé

Le problème d'emploi du temps consiste à affecter un ensemble d'événements à un nombre limité de périodes de temps, tout en respectant un ensemble de contraintes. Il est parmi les problèmes les plus influents, rencontrés dans la gestion des employés, des enseignements, des examens, des compétitions, des transports etc. Sa nature fortement combinatoire l'inclut dans la classe des problèmes NP-difficiles.

Dans cette présente thèse, nous décrivons un modèle pour un type de problèmes d'emploi du temps de l'université: le problème d'emploi du temps des examens. Ensuite, nous détaillons notre démarche d'investigation pour l'élaboration, pour la première fois (à notre connaissance), de deux nouvelles catégories d'approches de l'intelligence artificielle, pour résoudre ce problème. La première évolutionnaire, basée sur la Recherche Dispersée (Scatter search, SS), tandis que la seconde est de type local, basée sur la Recherche Locale Guidée (Guided Local Search, GLS).

Dans le cas de la catégorie d'approches basées sur la recherche dispersée, des stratégies de construction, de sélection et de combinaison, basées sur la diversité et la qualité sont proposées, pour construire et faire évoluer un ensemble de solutions initiales, afin d'en générer de nouvelles, de qualité meilleure. Les solutions finales, ainsi obtenues, offrent au décideur la possibilité de trouver un compromis entres ses objectifs, pour un choix pertinent.

Pour les approches basées sur la Recherche Locale Guidée, des stratégies de construction, de diversification et d'intensification sont proposées pour démarrer et guider le processus de recherche afin d'atteindre les solutions élites. Par la suite, la stratégie d'aspiration puis la stratégie des mouvements aléatoires et enfin la combinaison des deux sont intégrées aux approches de cette catégorie, afin de raffiner davantage les résultats.

Des expérimentations intensives menées sur des benchmarks publics, montrent que nos approches sont comparatives et compétitives dans bien des cas, avec plusieurs autres approches de l'état de l'art, parmi les plus récentes.

Mots clés : Emploi du temps, emploi du temps des examens, recherche dispersée, recherche local guidée, évolutionnaire, heuristique, métaheuristique, coloration de graphe.

Abstract

Timetabling problem consists in assigning a set of events to a limited number of time slots, while respecting a set of constraints. It is among the most influential problems encountered in managing employees, lessons, exams, competitions, transport etc. Its highly combinatorial nature includes it in the class of NP-hard problems.

In this work, we describe a model for a type of university timetabling problems: the examination timetabling problem. Then we present our investigation method for developing for the first time (to our knowledge), two new categories of artificial intelligence approaches, to solve it. The first one is based on the evolutionary Scatter Search, while the second is based on Guided Local Search.

In the case of Scatter Search approaches, specific construction, selection and combination strategies, based on diversity and quality, are proposed to create and evolve a set of initial solutions for generating new better ones. The final obtained solutions provide the decision maker the opportunity to find a compromise among its objectives, for a relevant choice.

For Guided Local Search approaches, construction, diversification and intensification strategies are proposed to start and guide the search process, to achieve elite solutions in the search space. Subsequently, aspiration strategy, then random move strategy and finally their combination, are added for more refined results.

Intensive experiments conducted on public benchmarks show that, our approaches are comparative and competitive in many cases, with the most recent from the state of the art.

Keywords: Timetabling, examination timetabling, Scatter Search, Guided Local Search, evolutionary, heuristics, meta-heuristic, graph coloring.

TABLE DES MATIERES

Liste des figures

Liste des Tables

Liste des Algorithmes

Introduction générale

L'aiguillon de la compétition a au cours de ces dernières années exacerbé le besoin d'organiser les méthodes de travail. Cet état de fait a conduit de plus en plus d'entreprises à améliorer non seulement les techniques et le matériel, mais aussi, leur gestion du potentiel humain. En effet, un grand défi des décideurs d'aujourd'hui et de demain est de savoir utiliser au mieux, toutes les ressources mises à leur disposition. Cette valorisation des ressources humaines nécessitent l'adoption de politiques permettant d'avoir, dans les meilleurs délais, des résultats de rentabilité, de sécurité et d'amélioration des conditions de travail.

Notre sujet de recherche s'inscrit dans cette optique et consiste à explorer deux nouvelles techniques de l'intelligence artificielle pour apporter une contribution à la résolution du problème d'emploi du temps : la Recherche Locale Guidée dite en anglais Guided Local Search (GLS) et la Recherche Dispersée dite Scatter Search (SS).

Le problème d'emploi du temps est un problème rencontré dans de nombreux domaines, tels que la gestion des employés de certains secteurs comme l'enseignement et la santé, la gestion du trafic aérien, ferroviaire, routier etc. Sa résolution consiste à trouver une planification ordonnançant des actions ou des événements, en leur affectant des ressources, en respectant un ensemble de contraintes et en répondant, au mieux, à un ensemble d'objectifs. Malgré les points communs aux différents problèmes d'emploi du temps, les différences existantes entre eux constituent un obstacle majeur à l'élaboration d'une méthode de résolution générale pour les traiter. Pour cela, généralement, les travaux de recherche se penchent sur des cas de problèmes d'emploi du temps particuliers, et la résolution est toujours spécifique aux cas considérés. Parmi les plus étudiés, nous avons ceux des transports, des employés, des conférences, des écoles et collèges, ainsi que ceux des enseignements et des examens dans les universités. L'élaboration manuelle des emplois du temps est considérablement lente et difficile, nécessitant de l'expérience, et la solution est rarement satisfaisante. Généralement, ils sont fortement combinatoires et appartiennent à la classe des problèmes NP-difficiles [karp, 1972]. Ils sont caractérisés par un volume de données important, des contraintes diverses et parfois antagonistes. Pour ces raisons ces problèmes ne sont pas encore bien établis d'une façon complète, et font l'objet de plusieurs travaux de recherche.

Dans ce travail, nous nous intéressons particulièrement, au problème d'emploi du temps des examens. Plusieurs variantes sont proposées dans la littérature. Elles dépendent généralement des institutions et des contraintes à respecter. Par ailleurs, plusieurs méthodes ont été proposées pour résoudre chacune de ces variantes. Parmi les premières, nous avons la programmation en nombre entiers, les heuristiques de graphes, et les flots dans les réseaux, (De Werra, 1985). Néanmoins, certaines de ces techniques sont impraticables, ou trop simples pour résoudre des problèmes complexes ou de grande taille. Elles n'ont atteint jusqu'à présent, que des résultats partiellement satisfaisants aux regards des objectifs escomptés. Pour pallier à cet inconvénient, des approches, basées sur les heuristiques séquentielles (Burke et al., 1998, 2004; Carter et al., 1996; De Werra, 1985) et la programmation logique par contraintes (Boizumault et al., 1996), ont été appliquées durant des années.

Plus récemment, plusieurs travaux de recherche ont adopté avec succès, des approches basées sur des métaheuristiques, telles que la Recherche Tabou, le Recuit Simulé, et les Algorithmes Génétiques (Kendall et Mohd Hussin, 2005; Thompson et Dowsland, 1998; White et al.,2004). Ces métaheuristiques permettent d'avoir en un temps raisonnable, comparées aux méthodes exactes, une ou plusieurs solutions (selon la méthode) proches de l'optimum. Généralement, elles démarrent avec une ou plusieurs solutions initiales générées aléatoirement ou a l'aide d'une heuristique, et utilisent des stratégies de recherche permettant d'éviter les optima locaux.

Ces dernières années, des travaux intensifs explorent de nouvelles approches pour le problème d'emploi du temps, afin de collecter le plus d'expériences possibles sur la question. Parmi les plus importantes, nous avons l*es Hyperheuristiques* (Burke et al., 2003, 2005, 2007; Kendall et Mohd Hussin, 2005), les *Méthodes hybrides* (Casey et Thompson, 2002; Merlot et al., 2003), le *raisonnement à base de connaissance* (Burke et al., 2006b; Petrovic et al., 2003), les *approches multiobjectives hybrides* (Côté et al., 2005), les *algorithmes de colonies de Fourmies* (Eley, 2006), les *méthodologies floues* (Asmuni et al., 2005), les *approches multicritères* (Burke et al., 2001), l*es approches de voisinage* (Abdullah et al., 2007a, 2007b; Abdullah et Burke, 2006), la *méthode du grand déluge* (Burke et Bykov, 2006a) etc. Néanmoins, la multitude des approches utilisées ainsi que l'absence d'une forme standard du problème font que la question reste encore ouverte. Voir les études de Schaerf 1999, Rhydian 2008, et Rong et al., 2009.

Dans cette thèse, nous proposons en premier lieu un modèle pour le problème d'emploi du temps des examens. Nous détaillons ensuite, notre démarche d'investigation pour élaborer, pour la première fois (à notre connaissance), deux nouvelles catégories d'approches de l'intelligence

artificielle pour le résoudre. La première est basée sur la Recherche Dispersée tandis que la seconde sur la Recherche Locale Guidée.

La Recherche Dispersée est une métaheuristique évolutionniste, basée population, proposée par Glover (1977) pour résoudre des problèmes d'optimisation combinatoire difficiles. Cette méthode, très peu utilisée à l'origine, a eu un regain d'intérêt suite aux publications de Glover (1994 et 1998). Néanmoins, le nombre de travaux basés sur cette approche reste encore, moins important que celui d'autres approches, telles que les Algorithmes Génétique, la Recherche Tabou etc.

La Recherche Locale Guidée, due à Voudouris (1997), est une autre métaheuristique élaborée à partir des travaux antérieurs sur Genet (Davenport, 1997). Genet est un algorithme inspiré des Réseaux de Neurones, basé sur la pondération des contraintes au niveau de la fonction objectif, pour échapper à l'optimum local (Voudouris, 1997). Depuis sa création, la Recherche Locale Guidée est utilisée avec succès pour divers problèmes d'optimisation combinatoire difficiles, suscitant ainsi, de plus en plus d'intérêt à l'adopter pour en affronter d'autres.

Pour la catégorie d'approches de la Recherche Dispersée, des stratégies de construction et de sélection basées sur la diversité et la qualité, ainsi que des opérateurs de combinaison, tous spécifiques au problème d'emploi du temps, sont proposés sous forme d'algorithmes pour construire et faire évoluer un ensemble de solutions afin d'en générer de nouvelles, de qualité meilleure. Les solutions ainsi obtenues, offrent au décideur la possibilité de trouver un compromis entres ses objectifs pour un choix pertinent. Pour la catégorie d'approches de la Recherche Locale Guidée, nous avons décrit à travers plusieurs algorithmes aussi, notre démarche pour concevoir de nouvelles stratégies de construction, de diversification et d'intensification, pour démarrer et guider le processus de recherche afin d'atteindre les solutions élites. Par la suite, nous avons intégré à nos approches avec cette méthode, une stratégie d'aspiration, puis une autre de mouvements aléatoires, et enfin la combinaison des deux, afin de raffiner davantage nos résultats. Des tests intensifs, menés sur des benchmarks publics, montrent que nos approches sont comparatives et compétitives avec plusieurs autres approches, parmi les plus récentes.

Cette thèse est organisée en six chapitres comme suit :

Dans le premier chapitre dédié à l'état de l'art, nous présentons une introduction au problème d'emploi du temps en général et de l'université en particulier, l'étude de sa complexité et sa

modélisation, une description des stratégies de résolution exactes et heuristiques pour les problèmes difficiles et enfin un résumé des travaux antérieurs.

Le deuxième et le troisième chapitre sont consacrés respectivement, à la description détaillée de la Recherche Dispersée et la Recherche Locale Guidée.

Dans le quatrième chapitre, nous présentons d'abord, le problème d'emploi du temps des examens considéré, ainsi que la modélisation que nous lui avons associée. Ensuite, nous décrivons en détail, nos différentes procédures de construction de diversification, de combinaison et de sélection, pour mettre au point les premières approches, basées sur la Recherche Dispersée. Ces approches permettront de créer des solutions initiales, puis d'en générer au cours d'un processus évolutif, de nouvelles, de plus en plus améliorées, afin de faciliter au décideur l'opération du choix pertinent en rapport avec ses objectifs.

Le cinquième chapitre décrit également, nos différentes stratégies de construction et de sélection, basées sur la diversité et la qualité, pour construire également, les premières approches basées sur la Recherche Locale Guidée. Ces dernières permettront de démarrer et de guider le processus de recherche afin d'atteindre les solutions élites. Une stratégie d'aspiration et une stratégie de mouvements aléatoires sont par la suite exploitées dans ce contexte, afin d'approfondir la recherche pour améliorer davantage, les résultats.

Dans Le sixième chapitre, des expérimentations intensives sur des benchmarks publics, détaillées illustrées et analysées, décrivent le cheminement de nos investigations pour mettre au point nos approches, basées sur la Recherche Dispersée et la Recherche Locale Guidée, pour le problème d'emploi du temps des examens. Ce chapitre s'achève par une étude comparative de nos résultats avec ceux obtenus par d'autres approches, parmi les plus récentes. Cette étude montre que nos approches sont comparatives et compétitives dans plusieurs cas.

Enfin, dans la conclusion générale, nous rappelons les résultats obtenus et nous proposons certaines perspectives sur la suite de notre travail.

Chapitre 1 Généralités sur Le problème d'emploi du temps

1.1 Introduction

Dans ce chapitre, nous présentons le contexte de notre étude. Nous commençons par des concepts sur les problèmes d'emploi du temps. Puis, nous décrivons ceux rencontrés dans le domaine de l'enseignement. Aux sections 1.4 et 1.5, nous présentons la complexité du problème et sa modélisation par le problème de coloration de graphe et la programmation mathématique. A la section 1.6, nous présentons les stratégies et heuristiques pour les problèmes difficiles et nous terminons enfin, à la section 1.7, avec une description des principales approches antérieures, proposées pour ce problème, avant de conclure.

1.2 Concepts de base

Le problème général d'emploi du temps consiste à affecter un ensemble d'événements à un nombre fini de périodes contiguës de temps (créneaux horaires), étant donné un ensemble de ressources, des contraintes impératives, et des objectifs à satisfaire autant que faire se peut.

La principale différence entre un problème d'ordonnancement et celui d'emploi du temps est que le souci majeur dans ce dernier, c'est quand est ce qu'un événement a lieu et non où a-t-il lieu. En d'autres termes, l'importance de l'attribution des ressources est restreinte pour le problème d'emploi du temps.

Dans le contexte du problème d'emploi du temps, lorsqu'on parle d'événement on sous-entend une activité à programmer. Une période représente un intervalle de temps auquel des événements peuvent être assignés. Une ressource représente par exemple, des salles ou un quelconque équipement exigé par les événements. Une contrainte est associée à une restriction, par exemple, quand et où des événements peuvent être programmés.

Les contraintes sont habituellement divisées en deux catégories :

- *Les contraintes impératives (fortes)* qui doivent être complètement satisfaites. Les emplois du temps vérifiant ces contraintes sont dits réalisables.
- *Les contraintes souhaitables (souples)* qui ne sont pas essentielles, mais dont la satisfaction est fortement souhaitée, afin de produire un emploi du temps de bonne qualité.

Un problème d'emploi du temps, dans le milieu de l'enseignement, assigne un nombre fini d'événements comme des cours, des examens, des conférences, des sessions de laboratoire etc., à un nombre limité de périodes, en réduisant au minimum, la violation de contraintes souhaitables tout en satisfaisant des contraintes impératives.

Les contraintes impératives typiques sont :

- Un événement doit être affecté à une et une seule période uniquement.
- Aucune ressource ne peut être allouée pour deux événements ou plus en même temps. Par exemple, dans le problème d'emploi du temps des examens, un étudiant ne doit pas être programmé à deux examens en même temps. Autrement, on dira que les deux examens en question sont en conflit.
- Pour chaque période, les ressources telles que les sièges doivent être en quantité suffisante.

Quelques contraintes souhaitables sont par exemple :

- Prévoir un jour d'intervalle pour des événements donnés.
- Un événement pourrait demander une période particulière.
- Certains événements pourraient demander une même période.

Dans la pratique, la qualité d'un emploi du temps est évaluée par la mesure de satisfaction des contraintes souhaitables; puisqu'il est habituellement impossible de les satisfaire entièrement toutes. En outre, ces contraintes souhaitables diffèrent d'un problème d'emploi du temps à un autre. Par exemple, contrairement aux cours, il est possible de programmer plusieurs examens dans une même salle, pour vu qu'ils ne soient pas en conflit.

Par ailleurs, bien qu'un algorithme puissant joue un rôle significatif dans un logiciel d'emploi du temps, il est aussi important de considérer d'autres points, tels que les interfaces, les formats de données, la compatibilité avec d'autres logiciels, la maintenance, la formation des utilisateurs etc.

Ces tâches nécessitent la collaboration d'équipes pluridisciplinaires, impliquant la recherche opérationnelle, l'intelligence artificielle, le développement de logiciel, les langages de programmation, les bases de données, les interfaces utilisateurs, et l'administration de l'université. En effet, toutes ces composantes ont un impact important et incontournable sur le problème d'emploi du temps.

Tous les problèmes d'emploi du temps partagent les mêmes caractéristiques de base, mais les différences existantes entres eux, rendent la tache de l'élaboration d'une solution générale, pratiquement difficile. Les travaux sur ce type de problèmes ont débuté les années 60 (Appleby et al., 1960; Csima et Gotlieb, 1964) et touchent depuis, plusieurs domaines, tels que le sport (Easton et al., 2004), le transport (Kwan, 2004), le collège (Abramson et al.,1999; Schaerf, 1995) et l'université (Burke et Newall, 1999 ; Burke et al., 2006b ; Burke et al., 1996; Thompson et Dowsland, 1998; Terashima Marin et al., 1999; white et Xie, 2004; Wren, 1996 ; Burke et al.,1996; Schaef, 1999). Cependant, comme ces problèmes deviennent de plus en plus complexes, ce champ important de recherche continue encore, de nos jours, à susciter l'intérêt de la communauté scientifique (Burke et Erben, 2001; Burke et Petrovic, 2002; Burke et al., 2006a , 2006b; Burke et al., 2007, 2009).

1.3 Problèmes d'emploi du temps dans l'enseignement

Étant donné, le nombre croissant d'étudiants, de ressources et de contraintes, l'élaboration d'emploi du temps dans les établissements de l'enseignement, est l'une des tâches difficiles, les plus importantes à effectuer, une ou plusieurs fois par an. Habituellement, ces institutions attribuent la tache de conception de leur emploi du temps à des membres particuliers du personnel, ou bien carrément, à un département dédié. Le procédé de conception est soit manuel, soit semi-automatisé ou bien entièrement automatisé.

1.3.1 Classification des problèmes d'emploi du temps dans l'enseignement

Les problèmes d'emploi du temps dans le domaine de l'enseignement sont repartis en trois classes principales (Schaerf, 1999): les emplois du temps des collèges, et ceux des cours et des examens à l'université.

Problème d'emploi du temps du collège

Le problème d'emploi du temps du collège, appelé souvent emploi du temps classe-professeur, concerne l'établissement du programme hebdomadaire, pour tous les cours du collège. Il est composé d'un ensemble de cours, des professeurs, des classes (ensemble d'élèves), et des périodes hebdomadaires prédéfinies. Ce problème consiste à affecter les cours à des périodes, de sorte qu'aucun professeur ou classe ne soit affecté à plus d'un cours, à une même période. D'autres contraintes sont également considérées, telles que les charges des professeurs, le temps de repos entre deux cours, les capacités, les endroits etc. Des exemples de recherches sur ce type de problèmes sont présentés dans (Abramson, 1991; De Werra, 1983).

Problème d'emploi du temps à l'université

Le problème d'emploi du temps à l'université comprend deux catégories principales: l'emploi du temps des examens et celui des cours.

- **Le problème d'emploi du temps des examens** affecte un ensemble d'examens à un nombre limité de périodes, de manière à ce que les étudiants puissent passer leurs examens, sans conflits.
- **Le problème d'emploi du temps des cours** consiste à assigner les cours à des périodes de temps et des salles, de sorte que les réunions entre les professeurs et les étudiants puissent avoir lieu, sans conflits.

Bien que ces deux problèmes paraissent à priori semblables, des différences fondamentales existent entre eux. Par exemple, contrairement au cours, le nombre de périodes d'examen peut varier. Plusieurs examens peuvent être assignés à une grande salle en même temps, alors que les cours doivent se dérouler chacun, dans une salle à part. Les conflits des cours ne sont pas aussi cruciaux que ceux des examens, puisque les étudiants peuvent avoir l'option de choisir un autre cours selon leurs convenances, alors qu'ils ne peuvent pas le faire dans le cas des examens.

Par ailleurs, l'emploi du temps des cours du collège et celui des cours à l'université peuvent aussi sembler identiques. Il n'en est rien, puisque des cours, à l'université, peuvent avoir des étudiants en commun, alors que le collège a habituellement des classes d'étudiants disjointes. En effet, dans certaines universités, les cours ont une gamme d'inscriptions plus variée, et les étudiants ont plus de liberté à choisir, à quels cours s'inscrire.

Il est enfin également important de noter, que l'emploi du temps des collèges et celui des cours à l'université sont construits sur une base hebdomadaire, alors que la session d'examens a un nombre de périodes de temps limité, quoique variable d'un établissement à un autre.

1.3.2 Problème d'emploi du temps des examens à l'université

Le problème d'emploi du temps des examens représente une activité administrative importante. C'est un problème difficile, impliquant les administrateurs, les enseignants et les étudiants. Beaucoup d'universités voient un nombre croissant d'inscriptions d'étudiants dans une plus grande variété de cours et ceci accentue, naturellement, le défi à relever pour résoudre ce problème.

Carter et Laporte (1996) ont défini le problème d'emploi du temps des examens comme étant : *l'attribution des examens à un nombre limité de périodes de temps disponibles, de sorte qu'il n'y ait aucun conflit.*

Quelques exemples de contraintes fortes pour l'emploi du temps des examens sont :

- Chaque examen doit être programmé, au plus, à une et une seule période.
- Il ne doit pas y avoir d'examens en conflit, programmés à la même période. Autrement dit, aucun étudiant ne doit être programmé à deux examens simultanément.
- Certains examens doivent être consécutifs, ou doivent avoir lieu dans un ordre spécifique.
- Les examens ayant le plus grand nombre d'étudiants, doivent être programmés plus tôt, dans l'emploi du temps, afin d'accorder plus de temps à l'inscription.

Quelques contraintes souhaitables communes, rapportées dans (Burke et al., 1996a), sont:

- Ne pas programmer un étudiant, à des examens de périodes consécutives.
- Ne pas programmer un étudiant, à plus d'un examen par jour.
- Espacer autant que possible, les examens pour chaque étudiant.
- Affecter certains examens particuliers, à des périodes particulières.
- Programmer les examens ayant le même nombre d'étudiants, à la même salle.
- Programmer les examens dans des salles proches du département approprié.
- Programmer les examens avec des questions en commun, à la même période.
- Programmer quelques examens, dans des salles particulières.
- Programmer un examen A, avant un examen B.

Il est cependant important de noter, que suivant les établissements, certaines contraintes souhaitables peuvent aller dans la catégorie des contraintes fortes et vice-versa. De même, plusieurs examens peuvent avoir lieu dans une même salle. Et certains examens peuvent être répartis dans plusieurs salles, faisant du nombre global de sièges par créneau horaire, la principale condition à respecter.

L'emploi du temps des examens est un problème complexe, vu le nombre croissant d'examens à programmer, le nombre limité de périodes et d'autres ressources, le nombre et la diversité des contraintes etc. Pour réussir une session d'examens, quatre étapes principales sont nécessaires :

Etape 1: Elle correspond à la collecte des informations essentielles, telles que le nombre de périodes, la liste des examens, la liste des étudiants inscrits aux examens, la liste des salles, la liste des surveillants et d'autres contraintes appropriées, déterminées par les départements.

Etape 2: Elle consiste à affecter des examens aux périodes disponibles, en évitant les conflits entres ceux ayant des étudiants en commun, en respectant absolument, les contraintes impératives et autant que faire se peut, les contraintes souhaitables. Ce problème référencé dans la littérature par le terme ‘‘Timetabling Problem’’ est modélisé généralement, comme un problème de coloration de graphe, augmenté d'un ensemble de contraintes. Il est dit «sans capacité » si la capacité des salles est non considérée et « avec capacité » si un nombre maximale de places d'examen par période est fixé (sans faire allusion aux salles).

Etape 3: Elle consiste à traiter le problème d'affectation des examens aux salles. Dans ce problème référencé dans la littérature par le terme ‘‘ Roomtabling’’, il s'agit d'affecter chaque examen à une salle, de sorte que l'effectif de l'examen ne dépasse pas sa capacité. Le problème se complique davantage, lorsqu'on répartit un examen sur plusieurs salles, ou bien lorsqu'on regroupe plusieurs examens dans une même salle. Généralement, il est NP-complet, de type bin-packing (Johnson 1974) où l'on essaye de ranger un ensemble d'articles de différentes tailles, dans un ensemble de boites, de capacités différentes à ne pas dépasser. Dans le cas du problème d'emploi du temps des examens, les articles correspondent aux examens, alors que les boites correspondent aux salles. Notons tout de même, que si l'on n'autorise qu'un seul examen par salle à une période donnée alors, l'affectation des examens, aux salles d'une manière optimale, peut

être résolue en un temps polynomial, à l'aide d'un algorithme glouton, tel que celui du couplage parfait.

Etape 4: Elle concerne l'affectation des surveillants aux salles (Awad et Chinneck, 1998). Ce problème, modélisé et traité par (Marti et al., 2000), veille à ce qu'aucun surveillant ne soit affecté à deux endroits en même temps. D'autres facteurs, comme les préférences des surveillants pour des périodes particulières et le nombre limité des périodes, augmentent la complexité du problème.

Enfin, le but final est de garantir, que tous les étudiants puissent passer n'importe quel examen auquel ils se sont inscrits, en veillant cependant, à maintenir une utilisation raisonnable des ressources (périodes, places d'examen ou salles etc.). Pour accomplir ceci, les contraintes impératives doivent être respectées absolument, et les contraintes souhaitables satisfaites, autant que faire se peut.

Naturellement, dans n'importe quelle situation du monde réel, il serait extrêmement rare, voir impossible, de satisfaire toutes les contraintes souhaitables. Par conséquent, une mesure utile de la qualité d'un emploi du temps peut être prise, comme étant le coût de violations de ces contraintes. La minimisation de ces violations constitue alors, un objectif à atteindre lors du développement des logiciels d'emploi du temps. La gestion d'une variété importante de contraintes est cependant, une tâche difficile, et chaque contrainte additionnelle peut augmenter davantage cette complexité. Par conséquent, dans la réalité, on recourt souvent à la relaxation de certaines d'entre elles. Cette observation pourrait motiver l'argument, que les systèmes semi-automatiques, interactifs sont préférables aux systèmes complètement automatiques. À cet égard, Schaerf (1999) a écrit, que beaucoup d'auteurs croient que le problème d'emploi du temps ne peut être complètement automatisé. Car d'une part, il est difficile d'exprimer toutes les conditions pour avoir « un bon » emplois du temps et d'autre part, parfois un système ne peut trouver la meilleure direction de recherche sans l'interaction humaine. Nous pouvons en conclusion dire, que le potentiel des méthodes automatisées est loin d'être entièrement exploré.

Notons cependant, que beaucoup de chercheurs ont proposé, pour résoudre des problèmes d'emploi du temps des examens, des approches heuristiques (Carter et Laporte, 1996), ou des approches méta-heuristiques, telles que la Recherche Tabou (white et Xie, 2001; Gaspero et Schaerf, 2001), le Recuit Simulé (Thompson et Dowsland, 1998), les algorithmes génétiques (Burke et al., 1994b, 1995; les algorithmes mémétiques (Burke et al., 1996) etc.

Un résumé sur les techniques antérieures, proposées pour traiter des problèmes d'emploi du temps des examens, seront discutés dans la dernière section de ce chapitre.

1.3.3 Problème d'emploi du temps des cours à l'université

Le problème d'emploi du temps des cours à l'université peut être définit comme suit :

« Un problème d'affectation multidimensionnel, dans lequel des étudiants ainsi que des professeurs (ou autre membres de la faculté) sont affectés aux cours, aux travaux dirigés ou aux travaux pratiques et les événements (différentes réunions entre les étudiants et les professeurs) sont affectés aux salles et aux périodes », (Carter et Laporte, 1998).

Dans le problème d'emploi du temps des cours, un ensemble de cours, composé chacun par un ensemble d'étudiants et un professeur, est programmé dans un nombre indiqué de salles et de périodes étalées sur une semaine. La définition de ce problème varie d'une institution à une autre. Néanmoins, certains concepts restent communs tels qu'un cours pouvant avoir une ou plusieurs séances par semaines et un cours pouvant être composé de plusieurs sections prises en charge par plusieurs enseignants. Quelques modèles combinatoires, basés sur le problème de coloration de graphe pour des problèmes de cours simples, peuvent être trouvés dans De Werra (1996b, 1997b). Tout comme l'emploi du temps des examens, celui des cours implique également des contraintes impératives et souhaitables.

Des exemples de contraintes fortes pour le problème d'emploi du temps des cours sont :

- La capacité d'une salle doit être suffisante, pour les étudiants d'un cours, à un créneau horaire particulier.
- Un cours ne peut être reparti dans plusieurs salles.
- La salle de cours doit satisfaire les spécificités exigées par le cours.
- Un étudiant ne doit pas être programmé dans deux cours à la fois, à la même période.

Quelques contraintes souhaitables relatives sont :

- Un étudiant ne devrait pas avoir un seul cours, par jour.
- Un étudiant ne devrait pas suivre plus de deux cours consécutifs, par jour.

- Un étudiant ne devrait pas être programmé à un cours, au dernier créneau horaire de la journée.

Carter et Laporte (1998) ont décomposé le problème de l'élaboration d'emploi du temps des cours, en cinq problèmes : le problème d'emploi du temps des cours, celui des classes-professeurs, celui de l'étudiant, celui du professeur et celui des salles. Des travaux sur les modèles associés sont rapportés dans (de Werra et al., 2002 ; Di Gaspero et al., 2003).

1.4 Complexité des problèmes d'emploi du temps

Le problème d'emploi du temps est un problème où l'on veut, tout en respectant une variété de contraintes, atteindre, autant que possible, un ensemble d'objectifs, parfois de nature antagoniste. C'est un problème difficile et la recherche pour mettre au point des approches efficaces pour le traiter, reste un challenge permanent pour les chercheurs. Avant d'aborder les approches de résolution, nous allons dans cette section, situer sa complexité.

Introduction à la théorie de la complexité des problèmes

Un problème combinatoire inclut les problèmes qui consistent à chercher des affectations, des arrangements, ou des permutations d'un ensemble discret d'objets, en respectant un ensemble de contraintes. Ces objets constituent généralement, les variables de décision du problème, pour lesquelles on cherche les valeurs satisfaisant les critères imposés.

L'ensemble des problèmes combinatoires est constitué par les problèmes d'optimisation, les problèmes de décision et les problèmes de rechercher. Un problème de recherche aspire à trouver une solution réalisable, sans tenir compte des contraintes souhaitables. Le problème d'optimisation se caractérise par la recherche d'une solution réalisable optimisant une fonction objectif. Le problème de décision consiste à trouver une solution satisfaisant toutes les contraintes, seulement dans ce cas, on accepte toutes les solutions dont la qualité est inférieure (supérieure) à un certain seuil.

Plus formellement, Un problème de recherche combinatoire est un ensemble de couples *(I, R)* où *I* est une instance du problème (un nombre fini de paramètres représentant les données du problème) et *R* une réponse à cette instance (Papadimitriou et Steiglitz, 1982). Dans un problème d'optimisation, on cherche une solution réalisable d'une instance optimisant la fonction objectif *f* du problème. Le problème de décision est défini pour qu'une instance du problème prenne

seulement la réponse « oui » ou « non ». Lorsque l'on fixe une borne à la fonction objectif d'un problème d'optimisation, le problème de décision correspondant consiste à répondre à la question: existe-t-il une solution réalisable S telle que *f(S) <= (>=) B ?* Par conséquent on peut dire qu'à chaque problème d'optimisation combinatoire correspond un problème de décision.

Par exemple, pour le problème d'emploi du temps des examens, on cherche une affection d'un ensemble d'examens à un ensemble de périodes, minimisant la violation d'un ensemble de contraintes souhaitables (souples), tout en respectant absolument, un ensemble de contraintes impératives (dures). Le problème de décision associé est : existe-t-il un emploi du temps ayant la valeur de la fonction objectif inférieure ou égale à une borne B?

Dans la pratique, il existe des problèmes d'optimisation plus faciles à résoudre que d'autres. Cette remarque a justifié de nombreux travaux sur la théorie de la complexité. Pour les algorithmes exacts, la théorie de la complexité est un outil utilisé pour analyser le comportement de l'algorithme. Un exemple de la complexité d'un algorithme est défini comme étant le nombre d'opérations de base pour l'exécuter. Si ce nombre est égale à $3n^2 + 4n$, n étant la taille des données du problème considéré, alors la complexité est en $O(n^2)$). L'algorithme est de complexité polynomiale et il est considéré comme efficace. Si la complexité est en $O(a^n)$ où a est une constante supérieure à 1, il est de complexité exponentiel, généralement, non pratique pour des instances de grandes taille.

Dans le domaine de la complexité des algorithmes, P dénote la classe de tous les problèmes de décision résolus par des algorithmes de complexité polynomiale, c'est à dire dont le temps d'exécution est en O (p(n)), n étant la taille des données d'une instance du problème.

NP (Non deterministic Polynomial) dénote la classe de tous les problèmes de décision pour lesquels il existe un algorithme vérifiant, en un temps polynomiale, si la réponse «oui» est juste. Autrement dit, c'est la classe des problèmes de décision solubles en un temps polynomiale par des algorithmes non déterministes (Garey et Johnson, 1979). En général, ces algorithmes se composent de deux parties : la première devine une solution, et la deuxième vérifie en un temps polynômial si cette solution résout le problème. Un algorithme non déterministe comprend des instructions de choix. Si on a le bon choix, le temps de résolution est polynomial. Par contre, si on énumère toutes les possibilités, l'algorithme devient déterministe et le temps de résolution devient

exponentiel, lorsque la taille du problème augmente. On peut donc, en déduire que les problèmes de la classe P constituent une sous-classe de la classe NP ($P \subset NP$).

Un problème de décision Π est dit NP-complet, si Π est dans la classe NP et tout problème de la classe NP lui est polynomialement réductible (CooK, 1971). Si un problème de décision est NP-complet alors le problème d'optimisation correspondant est appelé problème NP-difficile.

Pour démontrer qu'un problème est NP-complet, on doit montrer que tout problème de la classe NP dont le nombre est infini, lui est polynomialement réductible. En 1971, Cook a publié sa preuve révolutionnaire que le problème SAT est NP-Complet et que la réduction polynomiale est transitive. Par ailleurs, on note que la classe des problèmes NP-complets, pouvant être polynomialement réduits les uns aux autres, comprend plusieurs centaines de problèmes: le problème du Circuit Hamiltonian, le problème du voyageur de commerce, SAT etc.) (Garey et Johnson, 1979). Cependant, il n'y a encore aucun algorithme exact pour résoudre n'importe quel problème NP-complet. D'ailleurs, il n'y a aucune preuve de son existence ou non-existence. Habituellement, ces problèmes sont résolus par quelques algorithmes inexacts, (souvent nommés *heuristiques*), permettant de produire, en un temps raisonnable, des solutions proches des solutions optimales.

Complexité du problème d'emploi du temps des examens

Dans le cas du problème d'emploi du temps des examens, si l'on considère que nous avons 68 examens et 28 périodes, alors nous avons 28^{68} possibilités d'affecter les examens aux périodes. Le problème qu'on se pose, c'est de trouver un bon emploi du temps, dans ce vaste espace de recherche composé de solutions réalisables et non réalisables.

La solution (*optimum global*) pour un tel problème combinatoire peut être obtenue par un nombre fini d'étapes. Cependant, le nombre de ces étapes augmente exponentiellement, en fonction de la taille des données d'entrée du problème. Ainsi, des algorithmes énumérant complètement toutes les possibilités, peuvent être appliqués, uniquement aux problèmes de petite taille. Pour les problèmes de grande taille, la durée de calcul devient prohibitive. En outre, la large variété additionnelle des contraintes impératives et souhaitable, variables d'une institution à une autre, contribue également, à la difficulté de résolution de ce type de problème (De Werra et al, 2002).

En 1967, Welsh et Powell ont montré que le problème consistant à affecter *n* examens à *p* périodes, en évitant les conflits entre les examens, est équivalent au problème de *p*-coloration de graphe, consistant à colorer les sommets d'un graphe avec au plus *p* couleurs, de sorte que deux sommets adjacents reliés par une arête, n'aient pas la même couleur (Welsh et Powell, 1967). En 1972, Karp a prouvé que ce problème de coloration de graph est un problème NP-complet. En 1976 Even et al., ont démontré également, qu'une version basique du problème d'emploi du temps est NP-complet, en déterminant une réduction polynomiale du problème 3-SAT au problème considéré. Depuis, plusieurs travaux, ont prouvé ensuite, que les variantes de ce problème sont également, des problèmes NP-complets ou NP-difficiles, sauf pour quelques cas simples rares, sans contraintes. Voir (Cooper et Kingston, 1996; De Werra, et al., 1985, 2002).

1.5 Modèles

Au cours des dernières décennies, des questions importantes liées à la modélisation et à la complexité des problèmes d'emploi du temps ont été discutées dans la littérature. Cependant jusqu'ici, on n'a aucun modèle complet standard et universellement admis, pour tous les problèmes d'emploi du temps. Dans la suite, nous allons présenter deux méthodes de modélisation pour ce type de problèmes, la première est basée sur la théorie de graphe (De Werra, 1985, 1997) tandis que la seconde est basée sur la programmation mathématique (Leong et Yeong, 1987 ; Tripathy, 1984).

1.5.1 Modèle de coloration de graphique

Avant de présenter le modèle de coloration de graphe, nous commençons d'abord par donner quelques définitions essentielles.

Soit un graphe non orienté G = (*V, E) où* $V = \{v_1, \ldots, v_n\}$ est un ensemble de *n* sommets et *E* un ensemble fini d'arêtes. Une arête *e* de *E* est définie par une paire de sommets distincts v_i *et* v_j de *V*. On dit que v_i *et* v_j sont incidents à *e*, que v_i *et* v_j sont des extrémités de *e*, que v_i *et* v_j sont adjacents et on écrit $e = (v_i, v_j)$. On appelle degré d'un sommet v_i, le nombre d'arêtes incidentes à v_i. Un exemple de graphe non orienté ayant pour sommets $\{v_1, v_2, v_3, v_4, v_5\}$ est représenté par la Figure 1.1, où le sommet v_2 est de degré égal à 3.

Un graphe complet, ou clique à n sommets, noté K_n , est le graphe non orienté d'ordre n, dont deux sommets quelconques sont adjacents.

Etant donnée une partie *R* de *V*, on appelle sous-graphe *S* de *G* engendré par *R*, le graphe ayant pour ensemble de sommets, *R*. Le sous-graphe *S* est dit partiel, si une arête de *G*, donnant naissance à une arête de *S*, a ses deux extrémités dans *R*.

Le nombre chromatique χ d'un graphe est le nombre minimal de couleurs nécessaires pour colorer ses sommets, de sorte qu'aucune paire de sommets reliés par une arête, ne soit de la même couleur (Welsh et Powell, 1967).

Il est démontré que χ est plus grand ou égal à la plus grande clique dans le graphe (la clique est un sous-graphe dont les sommets sont reliés les uns aux autres. Cependant, il n'y a aucune méthode pour calculer exactement ce nombre, et par voie de conséquence, celui du problème de coloration qui est alors NP-complet (Karp, 1972).

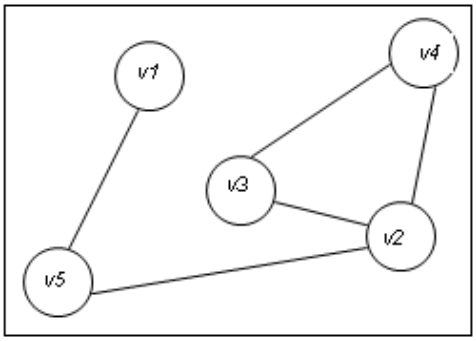


Figure 1. 1 Un graph non orienté *G* = (*V, E)*

Le problème de construction d'un emploi du temps, sans conflits, est analogue à la résolution du problème de *p*-coloration de graphe (Welsh et Powell, 1967, Christofides, 1971). Dans ce dernier, on utilise *p* couleurs différentes pour colorer les sommets du graphe, de sorte que deux sommets adjacents n'aient pas la même couleur. A noter que *p* peut être soit une donnée, soit un paramètre à minimiser.

Considérons à présent, le problème d'emploi du temps des examens où un ensemble d'examens doit être programmé, sans conflits, dans un nombre limité de périodes. Ce problème peut être vu comme un problème de coloration de graphe, où les sommets représentent les examens, les couleurs représentent les périodes et les arêtes représentent les conflits entre les examens. Notons cependant, que le modèle de base n'est pas pondéré. Néanmoins, des poids peuvent être associés, d'une part à chaque sommet, pour indiquer le nombre d'étudiants inscrits à l'examen

correspondant et d'autre part, à chaque arête, pour indiquera le nombre d'étudiants inscrits en même temps aux deux examens reliés par cette arête (examens en conflit). Le degré de chaque sommet correspond au nombre d'examens en conflit avec l'examen associé à ce sommet. Pour l'illustration, un problème simple d'emploi du temps, avec quatre examens, est représenté à la Figure 1.2.

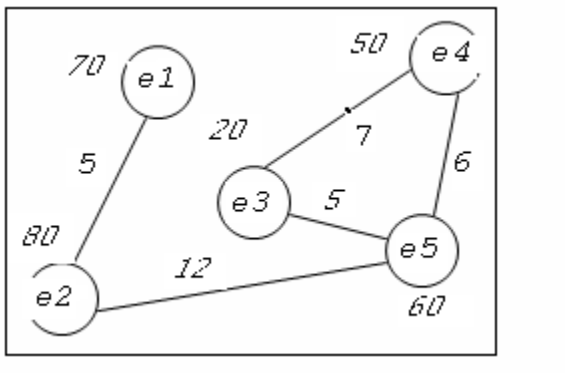


Figure 1. 2 Un exemple simple du problème d'emploi du temps des examens

Par exemple, le poids de e_1 est 70 parce que 70 étudiants y sont inscrits. L'arête reliant e_5 *à* e_4, est de poids égal a 6, parce qu'il y a 6 étudiants inscrits en même temps dans les deux examens.

Le problème de la Figure 1.3, comprend 5 examens : e_1 ,e_2 ,e_3 ,e_4 *et* e_5 avec quatre arêtes affichant les examens en conflit. On remarque qu'il n'existe pas de solution utilisant moins de 3 périodes (couleur), car les examens e_1 ,e_3 *et* e_4 sont en conflit et nécessitent d'être affectés à des périodes différentes. Par ailleurs, comme *e2* est en conflit avec *e1,* il ne peut être affecté qu'à la même période que e_3 ou e_4 (sans causer de conflits). Enfin, l'examen e_5, sans conflit avec les autres examens, peut être placé à n'importe laquelle des périodes : P_1 , P_2 *et* P_3. Dans l'exemple, il est affecté à P_1.

Il est à noter que ce modèle de coloration de graphe reste utilisable, même pour des instances de grande taille.

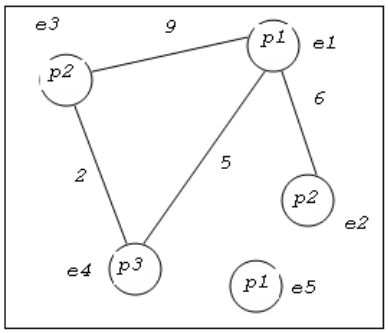


Figure 1. 3 Modèle de coloration de graphe pour un emploi du temps simple des examens

Pour plus de détails sur des études mettant en exergue la relation entre le problème de coloration de graphe et le problème d'emploi du temps, voir (Brelaz, 1979; De Werra 1985, 1997a; Carter, 1986; Burke et Ross, 1996; Burke et al., 2004a).

Notons ici, que le graphe représentant un problème d'emploi du temps des examens peut être également, représenté par une matrice d'adjacence sommet-sommet, booléenne et symétrique, $N \times N$, où N est le nombre d'examens, $C_{ij} = 1$ ($\forall\ i, j \in \{1,\ldots, N\}$), s'il y a un conflit entre les examens i et j ; et $C_{ij} = 0$ sinon. Voir un exemple de matrice de conflits dans la Table 1.1.

	1	2	3	4	5	6	. .	N
1	-	1	0	1	1	1		1
2	1	-	1	1	0	0		0
3	0	1	-	0	1	1		1
4	1	1	0	-	1	0		1
5	1	0	1	1	-	1		0
6	1	0	1	0	1	-		0
. .							-	
N	1	0	1	1	0	0		-

Table 1. 1 Matrice des conflits des examens

1.5.2 Modèle de programmation mathématique

En 1985 De Werra a présenté un modèle de programmation mathématique pour le problème d'emploi du temps des examens défini comme suit :
Etant données les notations et les hypothèses suivantes :

- *N :* Le nombre des examens.

- P : le nombre de périodes.
- w_{ip} : le coût de programmer l'examen *i* à la période *p.*
- t_{ip} = 1 si l'examen *i* est programmé à la période p et 0 autrement.
- C_{ij} = 1 si l'examen *i* est en conflit avec l'examen *j* et 0 autrement.
- Chaque examen doit être programmé une et une seule fois.
- Les examens en conflit ne doivent pas être programmés à la même période.

Le problème d'emploi du temps des examens correspondant est représenté, comme un problème d'optimisation avec une fonction objectif (1.1) à minimiser (Terashima-Marín, 1999) :

$$\min \sum_{n=1}^{N}\sum_{p=1}^{P} w_{np} t_{np} \qquad (1.1)$$

Sous les contraintes

$$\sum_{p=1}^{P} t_{np} = 1 \qquad \text{n=1,…N} \qquad (1.2)$$

$$\sum_{n=1}^{N}\sum_{m=1}^{N}\sum_{p=1}^{P} t_{np} t_{mp} C_{nm} = 0 \qquad (1.3)$$

Les contraintes (1.2) et (1.3) indiquent respectivement qu'un examen est programmé à une période, une et une seule fois et que les examens ayant des étudiants en commun ne doivent pas être programmés à la même période.

Une autre contrainte forte, pouvant être également considérée, est celle de la capacité totale des salles. Soit X_P cette capacité à la période *p*. La contrainte forte associée peut être représentée par l'expression 1.4 suivante :

$$\sum_{n=1}^{N} t_{np} \leq X_P \qquad p=1,\ldots\ldots P \qquad (1.4)$$

Notons que lorsque la fonction objectif n'est pas considérée, nous avons affaire à un problème de recherche d'un emploi du temps vérifiant les contrainte impératives, autrement dit, un problème de recherche de solutions réalisables.

1.6 Stratégies de résolution des problèmes NP-difficiles

Un problème d'optimisation combinatoire consiste à chercher à minimiser (maximiser) une fonction, avec ou sans contraintes impératives, sur un ensemble fini de possibilités. Un problème

d'optimisation est dit *NP-difficiles* si on ne connaît pas, pour le résoudre, un algorithme exact *polynomial* (dont le temps de calcul est en O(N^a), où *N* représente le nombre des paramètres inconnus du problème, et *a* une constante entière). Pour les grandes instances de ce type de problèmes NP-difficiles, le temps de traitement devient rapidement prohibitif.

L'un des exemples les plus caractéristiques du champ de l'optimisation difficile est le problème du voyageur de commerce, dans lequel on doit visiter un certain nombre de villes, avant de retourner au point de départ. La question est de déterminer l'itinéraire le plus court, traversant chaque ville une seule fois.

Etant donné l'importance pratique et la difficulté intrinsèque des problèmes NP-difficiles, de nombreuses méthodes de résolution ont été développées en recherche opérationnelle (RO) et en intelligence artificielle (IA). Parmi ces méthodes, nous citons la séparation évaluation (branch and bound), la recherche locale, le Recuit Simule, la Recherche Tabou, la programmation par les contraintes CSP, les heuristiques random walk, les algorithmes génétique, les approches évolutives de colonies de fourmis, les essaims particulaires etc. Ces méthodes peuvent être classées sommairement, en deux grandes catégories: les méthodes exactes ou méthodes complètes garantissant la complétude de la résolution et les méthodes heuristiques et metaheuristiques restituant, en un temps raisonnable, une ou plusieurs solutions proches de l'optimum.

1.6.1 Méthodes exactes

Le principe essentiel d'une méthode exacte consiste généralement, à énumérer souvent de manière implicite, l'ensemble des solutions de l'espace de recherche. Parmi les méthodes exactes, on trouve les techniques de séparation et évaluation progressive (SEP) et les algorithmes avec retour arrière, disposant de techniques permettant d'améliorer l'énumération des solutions. Les techniques de séparation et évaluation, ou recherche arborescente, sont des méthodes exactes, capables de restituer l'optimum, sans balayer tout l'espace de recherche représenté sous la forme d'un arbre. La fonction d'évaluation permet d'éliminer des branches de l'arbre pour localiser la solution optimale. Ces méthodes ont un temps d'exécution exponentiel, mais présentent une amélioration substantielle, comparée aux méthodes d'énumération totale de l'espace des solutions. Pour les cas des méthodes avec retour arrière, le principe est de construire une solution de manière itérative, où une fonction objectif permet de tester si la solution, en phase de construction, a des chances d'être de bonne qualité. Dés qu'un échec est détecté, on revient sur une décision déjà prise et on poursuit la construction. Cette méthode se base sur un parcours en profondeur ou en largeur de l'espace des solutions représenté sous la forme d'un arbre. Tout

comme les méthodes de séparation évaluation, la complexité des méthodes avec retour arrière est exponentielle.

Les méthodes exactes permettent de trouver des solutions optimales pour des problèmes de taille raisonnable. Néanmoins, elles rencontrent généralement, des difficultés face aux applications de taille importante où le temps de traitement peut devenir, rapidement, prohibitif.

1.6.2 Méthodes heuristiques et métaheuristiques

Les méthodes **heuristiques** et **métaheuristiques** constituent une alternative très intéressante, pour traiter les problèmes d'optimisation difficile de grande taille. Elles sont capables de trouver des solutions proches de l'optimum (bonne solutions), en un temps raisonnable. Cette classe de méthodes regroupe les *heuristiques*, souvent spécifiques au problème considéré, et les *métaheuristiques* basées sur un ensemble de stratégies et d'heuristiques. Certaines métaheuristiques sont basées sur la simulation d'un système physique comme la méthode du Recuit Simulé, ou d'un système biologique comme les algorithmes génétiques. D'autres, sont basées sur l'observation du comportement d'individus, telles que les colonies de fourmis et les essaims particulaires.

Une heuristique peut être définie, comme étant une stratégie limitant la recherche des solutions dans l'espace des configurations possibles, afin de résoudre un problème donné. Dans la pratique, la plupart des heuristiques sont conçues pour des problèmes spécifiques et ne peuvent être utilisées pour résoudre d'autres types de problèmes

En revanche, une **métaheuristique** est un ensemble de concepts, utilisés pour définir des méthodes heuristiques, pouvant être appliqués à une grande variété de problèmes d'optimisation discrets ou continus. Les métaheuristiques datent des années 80, et ne sont pas des méthodes figées. Leur efficacité dépend des paramètres utilisés, et du problème d'application, d'où l'intérêt des simulations informatisées. Elles sont généralement, exploitées pour des problèmes difficiles, pour lesquels les méthodes classiques échouent. Leur principe consiste à explorer l'espace de recherche en incorporant souvent un principe stochastique permettant de surmonter l'explosion combinatoire. Elles font parfois usage de l'expérience accumulée (durant la recherche de l'optimum), pour mieux guider la suite du processus de recherche. Ainsi, des stratégies d'apprentissage sont souvent, utilisées pour structurer l'information, afin de trouver efficacement des solutions optimales, ou presque optimales.

Comme les heuristiques, les métaheuristiques n'offrent généralement pas de garantie d'optimalité, bien qu'on ait pu démontrer par exemple, la convergence du Recuit Simulé. Cependant, ces preuves sont fondées sur des hypothèses rarement réalisables, limitant largement leur utilité pratique (Hao et al., 1999). Comme exemple, nous avons le cas du Recuit Simulé, pour lequel il a été démontré que sous certaines conditions de décroissance de la température, l'algorithme converge vers un optimum global, lorsque le nombre d'itérations tend vers l'infini (Aarts et al., 1997). Pour le cas de la Recherche Tabou, il n'existe pas de résultats théoriques garantissant sa convergence vers un optimum global. Le seul résultat théorique, connu à ce jour, concerne uniquement, une version probabiliste de la méthode, proche de celle du Recuit Simulé (Faigle et Kern, 1992).

Malgré tout, les métaheuristiques sont des techniques puissantes, pouvant être adaptées à plusieurs types de problèmes. Elles ont permis d'ailleurs, d'en résoudre en pratique, plusieurs. En effet, depuis une vingtaine d'années, des progrès importants ont été réalisés avec cette nouvelle génération de méthodes. Des solutions proches de l'optimum pour des problèmes d'optimisation de grande taille, qu'il était impossible de traiter auparavant, sont de plus en plus fréquentes. Les métaheuristiques peuvent être classées, selon leur principe de fonctionnement. Elles sont représentées essentiellement, par les approches *gloutonnes de construction*, comme la méthode de Lin et Kernighan (1973), les approches *de recherche locale* dites aussi *de voisinage*, comme le Recuit Simulé et la Recherche Tabou, les approches *évolutives*, comme les algorithmes génétiques, et enfin les approches *hybrides*.

1.6.2.1 Approches de construction

Le principe de ces approches se base sur la construction gloutonne des solutions, pas à pas. Initialement, la solution est vide, puis à chaque itération, une variable de décision de la solution reçoit une valeur de manière aléatoire, ou bien en utilisant une heuristique. La performance de ces approches est étroitement liée aux heuristiques utilisées pour la construction. Cependant, malgré leur facilité de mise en œuvre, elles engendrent souvent, des solutions médiocres.

1.6.2.2 Approches de voisinage

Avant d'aborder ces approches de voisinage, nous définissons un problème d'optimisation combinatoire comme étant un ensemble d'instances. A chaque instance du problème est associé :

- un ensemble discret de solutions X,
- un sous-ensemble S de X, représentant les solutions admissibles (réalisables),

- une fonction de coût *f* (fonction objectif, fonction pénalité), assignant à chaque solution *s* $\in S$ le nombre réel (ou entier) *f(s)*.

Résoudre une instance du problème, consiste à trouver une solution $s^* \in S$ optimisant la valeur de la fonction de coût *f*. La solution s^* est appelée une *solution optimale* ou un *optimum global*.

Definition (Papadimitriou et Steiglitz, 1982) Une *instance I* d'un problème de minimisation (maximisation) est un couple *(S, f)*, où $S \subseteq X$ est un ensemble fini de solutions admissibles, et *f* une fonction de coût (ou objectif) à minimiser (maximiser) $f : S \rightarrow R$. Le problème est de trouver, $s^* \in S$ tel que $f(s^*) \leq f(s)$ $(f(s^*) \geq f(s))$pour tout élément $s \in S$.

D'autres concepts importants pour les approches de voisinage sont également utilisés tels que la structure de voisinage, l'optimum local et l'optimum global, illustrés dans la Figure 1.4.

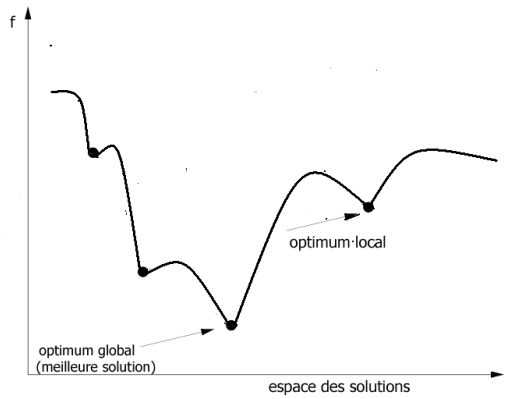


Figure 1. 4 Espace de recherche d'une fonction objectif à minimiser

Structure de voisinage: On appelle une structure de voisinage (ou tout simplement un voisinage) une fonction N, associant un sous-ensemble de *S* à toute solution $s \in$ S. Une solution $s' \in N(s)$ est dite *voisine* de s.

Optimum local : Une solution $s \in$ S est un *optimum local* relativement à la structure de voisinage N, si $f(s) \leq f(s')$ $\forall$ s' $\in N$(s) dans le cas d'une minimisation et si $f(s) \geq f(s')$ $\forall$ s' $\in N(s)$ dans le cas d'une maximisation.

Optimum global : Une solution $s \in$ S est un *Optimum global* si $f(s) \leq f(s')$ $\forall s' \in$ S dans le cas d'une minimisation, et si $f(s) \geq f(s')$ $\forall s' \in$ S dans le cas d'une maximisation.

Il existe plusieurs stratégies dépendantes du problème considéré, pour la génération d'un voisinage. Ces stratégies décrivent les étapes de passage d'une solution *s* vers les éléments du voisinage *N(s)*. De même, il existe plusieurs stratégies d'exploration du voisinage, définissant la

procédure de sélection d'une solution dans un voisinage (c'est-à-dire, précisant comment, la recherche passe d'une configuration de la solution *s* à une autre configuration *s'*, tel que $s' \in N(s)$).

Certaines approches de voisinage, améliorant une solution initiale en explorant son voisinage immédiat, présentent l'inconvénient de s'arrêter au premier optimum local trouvé. Comme nous le verrons plus loin, des approches de voisinage plus élaborées, appelées métaheuristiques, incluent souvent, une technique permettant d'éviter d'être piégé dans ces optima locaux. Elles utilisent une ou plusieurs stratégies pour explorer davantage l'espace des solutions, de façon à augmenter la probabilité de rencontrer l'optimum global. Parmi les plus connues, nous avons le Recuit Simulé (Kirkpatrick et al., 1983) et la méthode Tabou.

1.6.2.3 Approches évolutives

Les méthodes évolutives sont le fruit d'observations approfondies du comportement social des individus ou des communautés. Elles doivent leur nom à leur analogie, avec les mécanismes d'évolution des espèces vivantes. Contrairement aux méthodes de voisinage, un algorithme évolutif typique est composé de trois éléments essentiels :

- une *population* composée de plusieurs solutions initiales du problème considéré, générées aléatoirement ou en utilisant une heuristique,
- un *mécanisme d'évaluation* de chaque individu de la population,
- un *mécanisme d'évolution*, utilisant des opérateurs permettant l'élimination de certains individus et la production de nouveaux, à partir des individus sélectionnés.

Généralement, un algorithme évolutif suit les étapes suivantes:

1. Génération aléatoire ou heuristique, d'une population de solutions initiales.
2. Evaluation de chaque individu de la population.
3. Sélection d'une partie des individus.
4. Production de nouveaux individus, en combinant des individus sélectionnés.

Le processus formé par les étapes 2,3 et 4 est répété jusqu'à ce que la condition d'arrêt soit vérifiée (par exemple, lorsqu'un nombre maximum de générations est atteint). Selon l'analogie avec l'évolution naturelle, la qualité des individus de la population initiale devrait tendre à s'améliorer, au fur et à mesure. Dans cette seconde catégorie évolutive, on recense: les algorithmes génétiques,

les algorithmes de colonies de fourmis, l'optimisation par essaims particulaires**,** la Recherche Dispersée etc.

1.6.2.4 Hybridation de méthodes

L'hybridation de méthodes, dans ce contexte, permet de tirer profit des avantages de différentes métaheuristiques. C'est une tendance qui est observée dans de nombreux travaux ces dernières années.

Une des techniques les plus populaires d'hybridation concerne l'utilisation de métaheuristique de voisinage avec des métaheuristiques basées population. L'idée essentielle consiste à exploiter sur une population de solutions, la puissance de recherche des méthodes de voisinage, avec la puissance de combinaison des algorithmes évolutifs.

Une deuxième hybridation consiste à faire coopérer plusieurs métaheuristiques, à l'aide de processus parallèles communiquant entre eux pour échanger de l'information sur leurs résultats partiels.

Il est également possible d'hybrider d'autres types d'approches, par exemple une heuristique gloutonne avec une méthode de voisinage et un algorithme évolutif, ou bien un algorithme exact avec une métaheuristique etc.

L'approche de l'hybridation a permis de produire de très bons résultats, sur des *benchmarks* de problèmes de référence, tels que le problème du voyageur de commerce (Ulder et al., 1991) et celui de coloration de graphe (Galinier, et Hao, 1999). La plupart des applications réussies d'algorithmes génétiques, ou de colonies de fourmis, sont d'ailleurs, complétées par une phase de recherche locale, car c'est ce qui leur manquait à l'origine. A noter que ces deux types de recherche, globale et locale, peuvent s'exécuter de façon asynchrone, sur des processeurs différents.

1.6.3 Intensification et diversification

Les concepts d'intensification et de diversification ont été introduits par Glover (1997), dans son papier sur la Recherche Tabou. Par intensification, on sous-entend l'exploitation de l'information accumulée durant la recherche. C'est ce qui permet l'amélioration de la valeur d'une solution trouvée, dans un certain voisinage. Par diversification, on sous-entend au contraire, l'exploration de l'espace de recherche, encore inconnu. Elle est généralement obtenue par des processus

stochastiques. De manière générale, les métaheuristiques progressent itérativement et alternativement, entre les phases de diversification, d'intensification et d'apprentissage. Cependant, la diversification et l'intensification doivent être équilibrées, afin que l'exploration puisse rapidement identifier les régions de l'espace de recherche contenant des solutions de bonne qualité.

Les métaheuristiques, de type voisinage consacrent une bonne partie de leur temps à inspecter des régions limitées de l'espace de recherche. Elles utilisent néanmoins, une stratégie, afin d'introduire de la diversification dans ce schéma de fonctionnement, plutôt intensif et local à la base.

1.6.4 Conclusion

En conclusion, nous pouvons dire qu'une classe apparente des approches informatiques, pour résoudre les problèmes combinatoires difficiles, est essentiellement basée sur des algorithmes de recherche. L'idée fondamentale, derrière ces approches, est de produire et d'évaluer itérativement des solutions réalisables. Néanmoins, nous assistons ces dernières années à l'émergence d'une variété d'approches avec de nouveaux concepts, conduisant à dire, qu'il est peu probable qu'une classification stricte des heuristiques puisse être établie.

Dans la suite, nous allons faire un tour d'horizon sur un panel de méthodes, proposées dans la littérature pour contribuer à la résolution du problème d'emploi du temps des examens.

1.7 Travaux antérieurs

Le problème d'emploi du temps des examens est un problème difficile, auquel se confrontent toutes les universités, plusieurs fois par an. Les études de l'état de l'art sur ce problème ont débuté avec les travaux de Miles (1975) et Schmidt et Ströhlein (1979). Par la suite, De Werra (1985) a introduit des modèles mathématiques basés sur la théorie des graphes. En 1985, Carter a présenté une revue des premières recherches sur les applications pratiques d'emploi du temps dans les universités. Il a souligné également, qu'il n'y avait pas de données standards pour effectuer la comparaison entres les différentes méthodes proposées. En 1995, Kingston à présenté une bibliographie en ligne, avec plus de 1000 références. En 1996, Carter et Laporte ont mis à jour leur précédente étude et ont classé les méthodes en quatre types: les méthodes basées sur les cliques, les méthodes séquentielles, les méthodes métaheuristiques et les techniques de programmation par contraintes. Pour encourager la recherche sur les problèmes d'emploi du

temps, les auteurs ont aussi rendu public un ensemble de problèmes sur lesquels ils avaient effectué leurs tests. Depuis, ces problèmes de test sont repris par la communauté des chercheurs du problème d'emploi du temps. Nous en donnerons une description, dans le chapitre 4. En 1999, Schaerf a présenté une étude sur les emplois du temps des collèges et des universités. Sur la base des définitions des variantes de ces problèmes, des techniques de résolution, particulièrement de l'intelligence artificielle, ont été classées et réexaminées. Les orientations futures possibles sur les différentes techniques ont été également, présentées, dans (Burke et Petrovic, 2002; Petrovic et Burke, 2004; Rong, 2009 et Rhydian, 2008).

Dans la section suivante, nous essayons de classer une grande variété d'approches et de techniques de résolution pour ce problème. Cependant, comme un bon nombre de méthodes efficaces sont le résultat d'hybridations, cette classification est loin d'être stricte.

1.7.1 Méthodes exactes

Le principal avantage des méthodes exactes, c'est quelles sont conçues pour trouver une solution optimale. Toutefois, cela prendrait un temps prohibitif pour un problème NP difficile, comme celui d'emploi du temps des examens. En outre, il est souvent difficile de formuler mathématiquement ce problème de manière satisfaisante pour utiliser une méthode exacte, étant donné le nombre et la nature des objectifs et contraintes parfois, antagonistes. Par conséquent, les méthodes exactes, telles que la séparation évaluation (branch-and-bound) (Chen et Bushnell, 1996; Lawler et Wood, 1966), ou la programmation linéaire en nombre entiers (Sierksma et al., 1996) sont très peu adoptées pour traiter les problèmes d'emploi du temps, sans une relaxation de certaines contraintes. D'après certains travaux, la meilleure façon d'utiliser une méthode exacte, serait de la combiner avec des méthodes heuristiques ou métaheuristiques (Puchinger et Raidl, 2005).

1.7.2 Méthodes à base de contraintes

Généralement, dans une approche à base de contraintes, un problème est modélisé comme un ensemble de variables et domaines fini de valeurs possibles. La méthode affecte des valeurs aux variables, en tenant compte d'un certain nombre de contraintes.

Brailsford et al., (1999) donnent une introduction au problème de satisfaction de contraintes (CSP) et le définissent comme étant composé des éléments suivants :

- Un ensemble de variables $X = \{ x_1, \ldots, xn \}$.

• Un ensemble fini Di de valeurs possibles (domaine), pour chaque variable xi.

• Un ensemble de contraintes C= $\{c_1, \ldots\ldots, c_m\}$, pour restreindre les valeurs que les variables peuvent prendre simultanément.

Un problème de satisfaction de contraintes est résolu pour donner une solution réalisable, en attribuant à chaque variable une seule valeur de son domaine, satisfaisant chaque contrainte. Si une telle affectation n'existe pas, alors le problème est non satisfiable pour les contraintes considérées. En général, une approche, à base de contraintes seule, peut générer des solutions réalisables de façon efficace. Cependant, la plupart des ces approches de recherche n'ont pas la capacité de continuer à améliorer la qualité de la solution générée.

Plusieurs variantes à base de contraintes (Kang et white, 1992; Cheng et al. 1996) ont utilisé une approche logique de programmation par contraintes basée sur Prolog pour générer un emploi du temps d'examen. Boizumault et al. (1996) ont adopté la programmation logique par contraintes, en utilisant CHIP (Constraint Handling In Prolog). Brailsford et al. (1999) rapportent aussi, que les pures approches à base de contraintes ne peuvent rivaliser avec l'état de l'art des méthodes de recherche locale. Par conséquent, elles sont largement appliquées dans des approches d'hybridation, avec d'autres méthodes de recherche locale. Par exemple, White et Zhang (1998) et Merlot et al. (2003) ont proposé l'hybridation de la programmation par contraintes avec une Recherche Tabou et le Recuit Simulé, où la Programmation logique par contraintes est utilisée pour générer rapidement une solution de départ.

1.7.3 Techniques basées sur les heuristiques de coloration de graphes

La mise en exergue de la relation existante entre le problème de coloration de graphes et celui d'emploi du temps, due à Welsh et Powell (1967) et Garey et Johnson (1979), est a l'origine d'une quantité importante de travaux de recherches sur la coloration de graphe et le problème d'emploi du temps des examens (Brelaz, 1979; Carter et al., 1996 ; Carter et Johnson, 2001 et Burke et al., 1998). Dans ces travaux, les examens sont représentés par les sommets d'un graphe, alors que les contraintes impératives entre les examens sont représentées par les arêtes reliant des sommets. Le problème de coloration de graphe, consistant à attribuer aux sommets d'un graphe des couleurs de sorte que deux sommets adjacents n'aient pas la même couleur, correspond au problème d'attribution des périodes aux examens, en évitant les conflits entre examens tout en négligeant les contraintes souhaitables. Pour un problème d'emploi du temps, la qualité d'une solution réalisable est mesurée, par le degré de satisfactions de ces contraintes souhaitables.

Les premières approches, basées sur des heuristiques de coloration de graphe, sont dues à (Welsh et Powell, 1967; Foxley et Lockyer, 1968 ; De Werra, 1985, 1997). L'idée principale se base sur une stratégie permettant d'ordonner les examens, selon leurs degrés de difficulté à être affecté, un par un, à une période. Parmi ces approches, nous avons :

• *L'heuristique du plus grand degré (Largest Degree LD),* où les examens avec le plus grand nombre de conflits sont programmés en premier (Carter et al., 1996).
• *L'heuristique du degré de saturation (Saturation Degree SD),* considérée comme une stratégie de séquençage dynamique, où le choix du prochain examen à programmer est basé sur le plus petit nombre de périodes disponibles (Carter et al., 1996)[1].
• *L'heuristique du plus grand degré pondéré (Largest Weighted Degree LWD),* où la priorité est donnée à l'examen qui a le plus grand nombre de conflits pondérés (Carter et al., 1996)[2].

Cependant, il est important de noter que ces stratégies restent limitées, dans la mesure où les affectations de départ peuvent rapidement conduire à des situations, où des périodes valides ne sont plus disponibles pour poursuivre le processus de construction d'emploi du temps. Parmi les solutions proposées à ce problème, nous avons le retour arrière (backtrakting) (Carter et al., 1996) permettant de déloger, au cours du processus de construction, les examens en conflit, afin d'affecter les examens courants et poursuivre la construction.

1.7.4 Méthodes fondées sur les cliques

L'idée de la méthode des cliques (cluster) est proposée par Desroches et al.,(1978). White et Chan (1979) et white et Haddad (1983) décrivent ce type de techniques, comme des méthodes à trois phases. Dans la première phase, les examens sont regroupés dans des périodes de temps, pour construire un emploi du temps réalisable. Les examens groupés dans une même période ne doivent pas avoir de conflits entres eux (pas d'étudiants en commun). A la deuxième phase, des tentatives de réduction des conflits de second ordre, appelés aussi contraintes souhaitables, sont basées sur des permutations de périodes. La troisième phase est employée dans le but de raffiner davantage la qualité de la solution, en déplaçant les examens entre les périodes, moyennant une recherche locale.

[1] L'examen, ayant le plus petit nombre de périodes disponibles, sera programmé en premier.
[2] Chaque conflit est pondéré avec le nombre d'étudiants inscrits aux deux examens en même temps.

1.7.5 Techniques de recherche locale

Tout comme les algorithmes évolutionnaires, les techniques fondées sur la recherche locale, telles que la Recherche Tabou, le Recuit Simulé et leurs variantes, sont généralement considérées comme des métaheuristiques (Glover 1997). Les méthodes de recherche locale sont une famille de techniques générales, utilisées pour résoudre les problèmes d'optimisation combinatoire difficiles. Leur principe se base sur une procédure de recherche dans le voisinage d'une solution initiale, générée aléatoirement, ou a l'aide d'une heuristique. Les différentes structures et opérateurs de déplacement dans l'espace de recherche utilisés, font la distinction entres ces différentes techniques, où la recherche est guidée par une fonction objectif évaluant la qualité de la solution.

Depuis la dernière décennie, plusieurs méthodes de recherche locale sont largement appliquées à une variété de problèmes d'emploi du temps (Carter et Laporte, 1996). Leur performances et leur efficacité dépendent fortement, des paramètres et des propriétés de l'espace de recherche, tels que la convexité et la rugosité. Par conséquent, des connaissances sur le domaine de recherche et une expertise, dans l'opération d'ajustement des paramètres, constituent les éléments clés de la résolution de problèmes spécifiques difficiles. Dans la suite, nous présentons les plus importantes variantes de ces méthodes : la méthode de descente, le Recuit Simulé, et la Recherche Tabou.

1.7.5.1 Méthode de descente (Hill Climbing)

La méthode de descente est la plus simple méthode de recherche locale. Elle évolue itérativement, pour remplacer la solution courante par une autre du voisinage, de qualité meilleure.

En général, la performance de la méthode de descente, seule, est relativement faible. Car elle est facilement piégée dans un optimum local. Cependant, les méthodes d'hybridation qui l'utilisent, peuvent être efficaces. Par exemple, Merlot et al., (2003) ont utilisé une hybridation de la programmation par contraintes avec le Recuit Simulé et la méthode de descente, alors que Kendall et Hussin (2005) ont appliqué avec succès, l'hybridation de la descente avec une hyper-heuristique. Par ailleurs, il est à noter que cette méthode est au cœur de nombreuses métaheuristiques, telles que la Recherche Tabou, le Recuit Simulé et des algorithmes mémétiques.

1.7.5.2 Recherche Tabou

La Recherche Tabou est une approche itérative proposée par Fred Glover (1989). Glover et Laguna (1997) définissent la Recherche Tabou comme étant *une métaheuristique guidant une procédure heuristique de recherche locale, pour explorer l'espace au-delà de la solution optimum local.*

La Recherche Tabou utilise une condition d'arrêt spécifiée[3] pour arrêter le processus de recherche et restituer la meilleure solution obtenue. Elle démarre avec une solution initiale aléatoire ou heuristique. Puis, à chaque itération, la solution courante est remplacée par la meilleure solution du voisinage. L'exploration de l'espace de recherche se fait sans revisiter une liste de mouvements pris récemment et conservés dans une liste, dite liste taboue. Cette liste est construite et mise à jour au cours du processus de recherche, afin d'éviter de revenir sur des solutions déjà visitées. Les mouvements tabous sont alors, interdits pour un certain nombre d'itérations (la durée taboue). Ils peuvent toutefois être sélectionnés, en se basant sur un critère d'aspiration. L'un des critères d'aspiration, communément utilisé, consiste à changer le statut d'une solution taboue, si celle-ci génère la meilleure solution obtenue jusqu'ici. Sinon, le mouvement de recherche se déplace vers un autre voisinage, même si la solution obtenue est de moindre qualité que la solution courante.

Une étude complète de la Recherche Tabou est présentée dans Glover et Laguna (1997). Plusieurs travaux sur le problème d'emploi du temps, utilisant cette approche avec d'autre mécanismes, tels que, le concept hyper-heuristique, la modification des poids dans la fonction objectif etc, peuvent être consultés, dans (White et al., 2004; Di Gaspero et Schaerf, 2001; Schaerf, 1999 et Kendall et Hussin, 2005).

1.7.5.3 Recuit Simulé

La méthode du Recuit Simulé, introduite par Kirkpatrick et al. (1983), est basée sur les travaux bien antérieurs de Metropolis et al. (1953). Son principe s'inspire du processus naturel du recuit, consistant à chauffer un métal jusqu'à l'état liquide, puis à le refroidir lentement, en marquant des paliers de température pour obtenir un état cristallisé stable. Elle est utilisée dans la résolution des problèmes d'optimisation, où la fonction des coûts, à minimiser, correspond à l'énergie du métal. On démarre avec une solution initiale réalisable *s* de pénalité *f(s)* et une température *T*. on génère ensuite un mouvement pour obtenir une nouvelle solution *s'*, acceptée si sa pénalité *f(s')* est plus basse (c'est-à-dire de qualité meilleure). Autrement, elle est acceptée comme même, mais avec une probabilité dite de Boltzmann, égale à $\exp((f(s') - f(s))/T)$, pour un certain nombre d'itérations. La température est ensuite diminuée et le processus reprend jusqu'à un refroidissement proche de zéro.

[3] Un nombre maximal d'itérations, un temps imparti, une valeur de la fonction objectif de la solution etc.

Donc à température élevée, la probabilité est voisine à 1; ce qui conduit à accepter la plus part des configurations. A basse température, où la probabilité est proche de 0, la plupart des configurations, n'améliorant pas la pénalité, sont refusées. À température intermédiaire, une solution détériorant la pénalité est de temps en temps acceptée.

Donc L'idée principale, est de chercher une zone plus vaste de l'espace de recherche au début du processus, en acceptant de mauvaises solutions, avec une probabilité plus élevée. Cette probabilité diminuera progressivement, au cours des étapes suivantes. Les paramètres, tels que les températures initiale et finale ainsi que le facteur de refroidissement, doivent être ajustés de manière empirique.

Les travaux, les plus connus pour notre type de problèmes, intégrant le Recuit Simulé, sont dus à Thompson et Dowsland (1998), Merlot et al (2003) et Duong et Lam (2004). D'autres exemples également, sont décrits dans (Burke et al., 2003b, 2004b; Kendall, 2005.

1.7.5.4 Grand Déluge

L'algorithme du grand déluge proposé par Dueck (1993) est une procédure de recherche locale ayant certaines similitudes avec le Recuit Simulé. La différence entre elles, réside dans les lois d'acceptation des solutions détériorant la qualité de la pénalité de la solution. Cette approche est beaucoup moins dépendante des paramètres du Recuit Simulé (température initiale, taux de décroissance de la température, durée des paliers de température, critère d'arrêt du programme). Elle a juste besoin de deux paramètres: la durée du temps de calcul que l'utilisateur souhaite consacrer et une estimation de la qualité de la solution à atteindre. En dehors de l'acceptation d'une solution améliorant la pénalité, l'algorithme Grand Déluge accepte également une mauvaise solution, si sa pénalité est inférieure ou égale (pour le cas de la minimisation) à une certaine valeur limite supérieure donnée B, appelé Niveau. Généralement, ce niveau est initialement fixé à la valeur de la fonction objectif de la solution initiale. Au cours de la recherche, sa valeur est itérativement abaissée par une constante, dite taux décroissant.

L'algorithme du Grand Déluge est appliqué pour le problème d'emploi du temps des examens, dans (Burke et al., 2004; Burke et Newall, 2003a). Kendall et Hussin (2005) l'ont hybridé avec la Recherche Tabou dans une approche hyper-heuristique, alors que Petrovic et al. (2003) l'ont intégré dans une technique de raisonnement par cas. D'autres travaux peuvent être trouvés dans (Burke et al., 2001; Yang et Petrovic, 2005).

1.7.5.5 GRASP

GRASP (Resende et Ribeiro, 2003) est une métaheuristique composée de deux phases. La première construit une solution basée sur un algorithme probabiliste, glouton et adaptatif. Elle utilise des éléments de la solution, classés en fonction d'une stratégie, dans une liste dont la longueur est fixée de manière empirique. La seconde permet d'améliorer la solution construite à l'aide d'une procédure, telle que la méthode de descente, par exemple. Les deux paramètres de cette méthode sont la longueur de la liste des candidats et le nombre d'itérations autorisées. Une approche, utilisant la GRASP avec la Recherche Tabou, est proposée par Casey et Thompson (2002).

1.7.5.6 Recherche de voisinage variable (VNS)

Le Recuit Simulé et la Recherche Tabou utilisent généralement, une structure unique pour l'ensemble des voisinages de la recherche. Ils se concentrent le plus souvent, sur les paramètres ayant une influence sur l'acceptation des mouvements, plutôt que sur la structure du voisinage. Thompson et Dowsland (1998) ont étudié l'influence du choix de la structure du voisinage, sur la qualité des solutions obtenues. Ils sont arrivés à conclure, que la méthode de recherche n'est pas la seule à avoir un impact significatif sur les algorithmes ; la structure du voisinage joue également, un rôle non négligeable. Ahuja et al. (2001) ont de leur coté aussi, souligné l'importance de la structure du voisinage dans la recherche locale.

La Recherche par Voisinage Variable (variable neighborhood search, VNS) est proposée par Mladenović et Hansen (1997). À l'origine, la VNS est une méthode de descente, où les structures de voisinage sont variées régulièrement. Comme un optimum local dans une structure de voisinage n'est pas forcément un optimum local dans une autre structure de voisinage, le changement des structures de voisinage constitue un moyen pour échapper aux optima locaux rencontrés au cours du processus de recherche. Le critère de terminaison peut être par exemple, un nombre maximal d'itérations, un temps fixé ou un nombre maximal d'itérations sans amélioration.

L'application de la VNS au problème d'emploi du temps est étudiée par (Abdullah et al., 2006; Ahmadi et al.,2003). Qu et Burke (2005) ont intégré la VNS dans leur approche hyperheuristique et ont constaté, que la recherche locale itérée donne de bien meilleurs résultats que les méthodes VNS testées. Néanmoins, ces dernières étaient toujours compétitives.

1.7.6 Algorithmes basés population

Les Algorithmes évolutionnaires représentent une famille d'algorithmes basés population. Dans cette classe, les algorithmes génétiques, les algorithmes mémétiques, les algorithmes de colonies de fourmis et les algorithmes des essaims particulaires sont les plus étudiés pour le problème d'emploi du temps.

1.7.6.1 Algorithmes Génétiques

Les algorithmes génétiques (AG), dus à Holland (1975), simulent les processus d'évolution dans la nature, par la manipulation et l'évolution des populations de solutions, dans l'espace de recherche.

Cette méthode emploie sur une population de solutions individuelles, dénommées chromosomes, des opérateurs génétiques, comme la sélection, le croisement et la mutation. Son principe consiste à produire un certain nombre de générations, en vue d'améliorer la fonction objectif (souvent appelé fitness). L'opérateur de croisement est utilisé pour créer un ou plusieurs descendants, issus de deux parents. Les meilleures solutions sont ensuite choisies pour devenir des parents à leur tour. Un ensemble de paramètres et d'opérateurs dans les algorithmes génétiques, comme la taille de la population, le taux de croisement, le taux de mutation et le nombre de générations (Goldberg, 1989), doivent être ajustés de manière empirique. Ceci rend l'approche généralement, plus compliquée que celle des méthodes basées sur la recherche locale. En outre, la stratégie de recherche dans les algorithmes génétiques est fondamentalement différente de celle de la recherche locale, dans la mesure où, plusieurs solutions sont traitées à la fois, plutôt qu'une seule améliorée de manière itérative.

Des travaux utilisant les algorithmes génétiques pour le problème d'emploi du temps des examens sont décrits dans (Corne et al., 1994; Ross et al.,2003; Erben 2001; Burke et al., 1994, 1995; Terashima-Marin et al., 1999). En particulier, des hybridations d'algorithmes génétiques avec des méthodes de recherche locale, parfois appelées algorithmes mémétiques, ont conduit à plusieurs succès dans le domaine.

1.7.6.2 Algorithmes Mémétiques

Les algorithmes mémétiques sont une extension des algorithmes génétiques, en leur rajoutant une procédure de recherche locale. Cependant, il y a habituellement un prix à payer en termes de temps de calcul nécessaire. Plusieurs questions, relatives à la conception d'algorithmes mimétiques pour le problème d'emploi du temps sont présentées dans, (Burke et al.,1996; Burke

et Newall, 1999; Burke et Landa, 2004; Moscato, 1999), où les résultats expérimentaux montrent que la qualité des solutions est meilleure que celles restituées par les algorithmes génétiques, adoptés seuls.

1.7.6.2 Algorithmes de colonie de fourmis

L'algorithme de colonie de fourmis est une métaheuristique basée population, proposée par Dorigo et al., (1996) pour la résolution des problèmes d'optimisation difficiles. L'idée de base de cette technique s'inspire du comportement des fourmis, laissant derrière elles, une traînée de phéromone. En effet, lorsque celles-ci explorent leur environnement à la recherche de la nourriture, elles finissent généralement, par trouver le plus court chemin à la fourmilière. Les fourmis choisissent, avec une plus grande probabilité, les chemins contenant les plus fortes concentrations de phéromones. Elles communiquent par l'intermédiaire des modifications dynamiques de leur environnement (traces de phéromones) et finissent par construire une solution au problème, en se basant sur leur expérience collective. Ainsi, nous avons une collaboration et une auto organisation des fourmis peu intelligentes, pour avoir une organisation globale complexe. Des travaux, sur cette approche, sont décrits dans (Costa et Hertz, 1997; Dowsland et Thompson, 2005 et Naji Azimi, 2004).

1.7.6.3 Algorithmes des essaims particulaires

La méthode des essaims particulaires est une approche relativement nouvelle pour l'optimisation globale (Kennedy et Eberhart, 1995, 2001). Elle utilise la modification des mouvements des individus dans leur population appelée essaim, en s'appuyant sur le concept de collaboration et d'auto-organisation. L'algorithme commence par placer, généralement de manière aléatoire, chaque particule dans l'espace de recherche du problème. Puis, à chaque itération, chaque particule est déplacée en fonction de trois composantes : sa vitesse courante, sa meilleure solution et la meilleure solution obtenue dans son voisinage. Ainsi, grâce à des règles de déplacement dans l'espace des solutions, les particules convergent vers un optimum local. Des travaux utilisant cette approche sont décrits dans **(**Daniel Falko, 2005).

1.7.7 Techniques multicritères

Dans la majorité des algorithmes et approches conçus pour le problème d'emploi du temps, les coûts pondérés des violations des différentes contraintes sont additionnés, pour donner une mesure de la qualité des solutions. Cependant, dans les circonstances réelles, les contraintes sont souvent considérées comme des points de vue différents. La somme des coûts correspondant aux différentes contraintes ne peut donc pas toujours, rendre compte de la situation dans de tels cas.

Plusieurs techniques multicritères ont été étudiées récemment pour le problème d'emploi du temps. Par exemple, Pacquete et Fonseca (2001) ont proposé une approche évolutionnaire multi objectif utilisant le classement Pareto, afin de manipuler des contraintes différentes. Burke, Bykov et Petrovic (2001) on développé une approche dite programmation de compromis, afin de traiter aussi, plusieurs critères différents. Carrasco et Pato (2001) ont mis au point une approche où ils combinent le principe des algorithmes génétiques, avec la méthode multi-objectif. Pour d'autres travaux, voir (Burke et Petrovic, 2002; Petrovic et Burke, 2004; Petrovic et Bykov, 2003).

1.7.8 Méthodes Hyper-heuristiques

Dans les métaheuristiques, le réglage des paramètres pour de nouveaux problèmes est souvent difficile et requière une certaine compétence. Ce problème bien connu a conduit des chercheurs à s'intéresser, au développement de nouvelles méthodes dites Hyper-heuristiques, opérant à un niveau plus élevé de généralité. Les Hyper-heuristiques sont par conséquent, motivées par de telles observations. Et le terme peut être interprété comme une heuristique choisissant des heuristiques sur la base de mécanismes d'apprentissage. Autrement dit, l'espace de recherche d'heuristiques devient le centre d'intérêt, plutôt que l'espace de recherche de solutions, comme c'est le cas habituellement, avec la plupart des méta-heuristiques (Burke et al., 2003b, Ross, 2004). L'objectif est donc, de développer des approches plus générales, plutôt que des approches affinées pour des problèmes spécifiques, nécessitant souvent beaucoup d'efforts pour le réglage des paramètres. Comme exemple, Ross et al.,(2004) ont déduit de leurs travaux, qu'un algorithme génétique peut être employé avec succès dans la recherche d'un bon algorithme, plutôt que dans celle des solutions.

De nos jours, les hyper-heuristiques sont en passe de devenir des approches puissantes, relevant le niveau des systèmes de résolution des problèmes. Elles sont maintenant largement, explorées pour les problèmes d'emploi du temps et d'ordonnancement. Néanmoins, il est important de noter, que bien que les résultats expérimentaux montrent qu'en général ces approches sont en mesure de trouver de bonnes solutions, la taille de l'espace de recherche augmente de façon significative. Donc, le temps de calcul peut devenir un problème. Pour plus de détails voir (Ahmadi et al., 2003; Qu et Burke, 2005; Ross et al.,1998 ; Yang et Petrovic, 2005).

1.7.9 Systèmes multi-agents

Lin (2002) a développé un système multi-agents, où le problème d'emploi du temps est partitionné en sous problèmes traités chacun par un agent, avant de rassembler les résultats

obtenus en une solution globale. Les tests expérimentaux révèlent, que cette approche opère mieux sur les problèmes épars, comparé au problème denses.

1.7.10 Techniques à base de règles et raisonnement à base de cas

Le raisonnement à base de règles est souvent utilisé dans les systèmes de recherche à base de connaissance. Cependant, il est rarement utilisé comme une approche pratique pour résoudre les problèmes d'emploi du temps.

Un système fondé sur le raisonnement à base de règles résout les problèmes en simulant le processus manuel de résolution d'un spécialiste expérimenté. L'expertise de l'homme est modélisée comme un ensemble de "IF THEN" règles stockées dans la base de connaissances. Un moteur d'inférence doit être développé afin de déterminer comment et quand est ce qu'il faut appliquer ces règles prédéfinies. La plus grande limitation, dans un système à base de règle, est l'énorme effort nécessaire pour acquérir les connaissances de l'expert. Voir Guyette et al., 1994 ; Kong et Kwok, (1999).

Le raisonnement à base de cas CBR est une technique de l'IA soutenue par l'étude des sciences cognitives. Elle est motivée par les observations sur le comportement humain, utilisant l'expérience acquise pour résoudre des problèmes similaires, ou réutilisant cette expérience, avec quelques modifications, pour répondre à d'autres exigences (Kolodner et Kolodner, 1996).

Dans le CBR, les problèmes sont représentés par des cas définis par Kolodner et Leake (1996) comme des pièces contextuelles de la connaissance. Chaque cas représente une expérience qui enseigne une leçon fondamentale à la réalisation des objectifs du raisonneur. Un cas a habituellement deux parties: la description et la solution du problème. Les cas sont stockés dans une base de connaissances : la base des cas. D'après Aamodt et Plaza (1994), il existe quatre processus cycliques en CBR:

- La Récupération : les cas les plus similaires (en utilisant des mesures de similarité) doivent être extraits de la base des cas.
- La Réutilisation : la solution récupérée est utilisée pour résoudre de nouveaux problèmes.
- La Révision : la solution récupérée est révisés suivant les nouvelles exigences.
- La Conservation : la solution du nouveau problème est retenue dans la base des cas.

Des travaux, utilisant des approches à base de cas, sont rapportés dans (Burke et al., 2006b; Petrovic et al., 2003; Petrovic et Burke, 2004).

1.7.11 Méthodes Hybrides

L'approche hybride permet d'incorporer une idée d'une approche, dans une autre approche. Beaucoup de méthodes, déjà décrites, sont en fait des hybridations de méthodes. L'Hybridation s'est révélée très efficace pour le problème d'emploi du temps. Par exemple, Merlot et al., (2003) ont appliqué une méthode hybridant la programmation par contraintes, le Recuit Simulé et la méthode de descente. D'autres travaux sont également décrits, dans (Caramia et al., 2001; Burke et Newall, 2003a ; Burke et al., 2004b ; Côté et al., 2005; Thompson et Dowsland, 1998).

1.8 Conclusion

Dans ce chapitre dédié à l'état de l'art du problème d'emploi du temps en générale et celui des examens en particulier, nous avons présenté les principaux modèles adoptés, une étude de la complexité du problème, une description des stratégies de résolution des problèmes difficiles et enfin un tour d'horizon sur les principales approches proposées dans la littérature.

Dans le chapitre suivant, nous allons décrire une méthode évolutionniste : la Recherche Dispersée, dite en anglais, Scatter Search (SS).

Chapitre 2 Une métaheuristique évolutionniste: la Recherche Dispersée

2.1 Introduction

La Recherche Dispersée (Scatter Search, SS) est une métaheuristique évolutionniste, basée population, proposée par Glover (1977) pour résoudre des problèmes d'optimisation combinatoire NP-difficiles. Cette méthode, très peu utilisée à l'origine, a eu un regain d'intérêt, suite aux publications de Glover (19994 et 1998) et aux résultats intéressants, obtenus pour plusieurs problèmes difficiles. Parmi les travaux mettant en exergue le caractère générique indéniable de cette méthode, nous citons :

- le problème de mise en ordre linéaire (Campos et al., 2001).
- Le problème de clique maximale (Cavique et al., 2001).
- le problème du broadcasting dans les réseaux mobiles Ad-hoc (Luna et al., 2006).
- le problème de coloration de graphe (Hamiez et Hao, 2001).
- les problèmes SAT, Max-SAT et Max-W-SAT (Drias 2001; Drias et Khabzaoui, 2001; Drias et Azi, 2001; Boughaci et Drias, 2005; Boughaci et al., 2007; Boughaci et al., 2008).
- le problème de la recherche d'information (Drias et al., 2009).
- Le problème de la cryptanalyse (Garici et Drias, 2005).

Notons tout de même, qu'il est important de signaler que le nombre de travaux basés sur cette approche est encore bien inférieur à celui d'autres approches, telles que les algorithmes génétique, la Recherche Tabou etc. Néanmoins, les résultats prometteurs des travaux, effectués sur cette méthode que nous décrivons dans la suite, suscitent, de plus en plus d'intérêts à l'adopter pour d'autres problèmes d'optimisation difficiles. Pour plus de détails, voir (Glover et al., 2003).

2.2 Description générale de la Recherche Dispersée

Le concept général de la Recherche Dispersée suppose, que l'information utile sur la solution optimale est contenue dans une collection diversifiée de solutions élites. Que les stratégies de combinaison, pour générer de nouvelles solutions, incorporent des techniques de diversification et

d'intensification. Et que l'utilisation de plusieurs combinaisons de solutions de référence augmente l'opportunité d'exploiter l'information contenue, dans l'union des solutions élites.

La Recherche Dispersée utilise des stratégies de combinaison et de mise à jour, permettant de combiner les solutions d'un ensemble de référence, pour en générer de nouvelles. Elle peut être mise en œuvre, de différentes manières, et offre beaucoup d'alternatives pour exploiter ses idées fondamentales. De façon générale, elle comprend cinq méthodes que l'on peut esquisser comme suit :

Méthode de génération de diversification: pour produire une population de solutions initiales, diversifiées et de bonne qualité.

Méthode d'amélioration des solutions*:* pour améliorer la qualité des solutions obtenues par la *méthode de génération de diversification,* ou par la *méthode de combinaison de solutions*.

Méthode de construction et de mise à jour de l'ensemble de référence: pour construire, à partir de la population de solutions initiales et maintenir l'ensemble de référence *R* composé des *b1* meilleures solutions en termes de qualité et des *b2* meilleures solutions en termes de diversité.

Méthode de génération des sous-ensembles à combiner*:* pour générer, à partir de l'ensemble de référence *R*, des sous-ensembles de solutions avec deux éléments ou plus, afin de préparer la phase de combinaison.

Méthode de combinaison de solutions: pour combiner les solutions de chaque sous ensemble, généré par la méthode précédente, afin de produire une ou plusieurs nouvelles solutions.

Tout comme les algorithmes génétiques, la Recherche Dispersée est une approche évolutive basée population. Cependant, alors que la taille de la population de référence dans les AG est de 100, elle est en moyenne de 20, dans la SS. Dans les AG, les combinaisons ont lieu entre toutes les paires de solutions, alors que la SS peut combiner, de deux à 5 éléments. Les AG utilisent les operateurs aveugles, alors que la SS utilise des opérateurs intelligents, spécifiques aux problèmes considérés. Par ailleurs, pour éviter une convergence prématurée vers un optimum local, la population de la SS est composée de deux sous-ensembles, l'un formé par les meilleures solutions et l'autre par les solutions les plus diversifiées. La SS utilise également, des idées empruntées à la Recherche Tabou, par exemple, la mémoire adaptative permettant de contrôler la diversification et l'intensification, pour une meilleure exploration de l'espace de recherche, afin d'atteindre les régions prometteuses.

2.3 Algorithme de la Recherche Dispersée

La Recherche Dispersée fait évoluer une population de solutions et s'appuie sur le principe suivant :

Initialement une population de solutions diversifiées est générée et éventuellement améliorée par l'application d'une recherche locale. Ensuite, un ensemble de référence *R*, contenant les meilleures solutions en terme de qualité et les meilleures solutions en terme de diversité, est constitué. Les solutions de sous-ensembles de *R* sont alors combinées entre elles pour générer de nouvelles, à améliorer. Les opérations de combinaison, d'amélioration et de mise a jour de *R* sont itérées, jusqu'a ce qu'il n'y ait plus de nouvelles solutions. A ce stade, une partie de la population est régénérée et le processus recommence jusqu'à la satisfaction d'un critère d'arrêt, tel qu'un nombre d'itérations, un temps imparti, ou une valeur de la fonction objectif. Voir Algorithme 2.1.

Notons ici, que l'un des points importants de cette méthode est la mesure de la diversité des solutions, basée sur l'espace des solutions et non sur celui des objectifs.

Dans la Recherche Dispersée, les facteurs d'intensification sont présents dans l'application systématique d'une méthode de recherche locale, dans la combinaison répétée des solutions, ainsi que dans la mise à jour de *R*. L'utilisation des solutions les plus diversifiées, et le renouvellement de la population, constituent les facteurs de diversification.

Algorithme 2. 1 Algorithme de base de la Recherche Dispersée

```
Générer une population de solutions initiales variées, et diversifiées ;
Améliorer éventuellement les solutions initiales;
Construire un ensemble de référence R à partir de la population ;
Tantque (NbIter non atteint) Faire
        Construire les sous ensembles à combiner;
        Tantque (critère d'arrêt non atteint) Faire
                Appliquer la combinaison à chaque sous ensemble ;
                Améliorer la solution obtenue;
                Mettre à jour l'ensemble référence R;
        FinTantque
Renouveler la population, en gardant les meilleures solutions de R, et en générant le reste ;
FinTanque.
```

2.4 Génération de diversification

Dans la Recherche Dispersée, le but de la diversification est de produire une population de solutions diversifiées, afin de couvrir le plus possible, l'espace de recherche. Plusieurs générateurs

de diversification, très dépendants du problème considéré, ont été développés dans la littérature. Particulièrement, Glover (1998) a proposé deux types de générateurs : les générateurs de diversification pour les problèmes dont les solutions sont des vecteurs en 0/1, et les générateurs de diversification, pour les problèmes de permutation d'éléments.

2.4.1 Générateur de diversification pour les solutions en 0/1

L'idée de cette méthode, décrite dans Algorithme 2.2, est de générer un ensemble de solutions, à partir d'une solution donnée. Le principe consiste à modifier les valeurs des éléments, dont l'indice est associé à un facteur.

Un exemple de générateur de diversification de ce type consiste à considérer un vecteur X représentant une solution constituée de n éléments xi, pouvant prendre la valeur 0 ou 1. A partir de chaque solution X, on génère une collection de solutions associées à un entier h= 1,.., h*, avec h* ≤n-1 (il est recommandé de prendre h* ≤n /5).

Pour chaque valeur de h, deux solutions, X' et son complément X'', sont générées de la manière suivante :

X' [1] := 1- X [1].
X' [1+kh] := 1-X [1+kh] ; pour k=1,….,k* (k* le plus grand entier tel que k*<=n/h) .
X' [i] := 0 ailleurs.

Algorithme 2. 2 Générateur de diversification

```
//Soit X une solution de taille n.
Pour (h=1 à n/5 faire
        A partir de X, générer deux solution X' et X'' ;
        X' [1] := 1- X [1] ;
        Pour (k=1à n/h) faire
        X' [1+kh] := 1-X[1+kh] ;
Finpour
X'' complément de X ;
Finpour.
```

2.4.2 Générateur de diversifications pour les problèmes de permutation

Pour les problèmes de permutation, on considère une permutation P=(1,2,…,n) associée à une solution initiale. On définit la permutation *P(h :s)* tel que *1 ≤s ≤h* , par :
P(h :s)=(s, s+h, s+2h,…, s+rh), où *r* est le plus grand entier non négatif tel que *: s+rh ≤n.*

La permutation *P (h)* devient :

P (h)=(P(h :h), P(h :h-1), …, P(h :1)).

Il est recommandé de prendre h ϵ [1, n/2].

Exemple :

Soit *P* donnée par *P*= (1, 2,3,...,15), Pour h=5, on a *P*(5 :5)=(5,5+5,5+2*5)=(5,10,15),

P(5 :4)=(4,9,14), *P*(5 :3)=(3,8,13), *P*(5:2)=(2,7,12) , *P*(5:1)=(1,6,11).

La permutation est alors : *P*(5)=(5,10,15,4,9,14,3,8,13,2,7,12,1,6,11).

2.5 Amélioration des solutions

Dans la Recherche dispersée, la méthode d'amélioration (Glover 1998) est dans la plupart des cas, contexte-dépendante. Elle cherche à améliorer la solution courante, en la remplaçant par une autre du voisinage, de meilleure qualité. Cependant, comme le voisinage peut être très vaste, il est conseillé d'investir dans une méthode de voisinage adéquate, et de trouver un équilibre entre l'exploration du voisinage et la qualité d'amélioration de la solution.

2.6 Génération de l'ensemble de référence

Dans la Recherche Dispersée, une importance particulière est accordée à la construction de l'ensemble de référence *R*, à partir de la population initiale. Cet ensemble prépare le terrain au processus évolutif de recherche, pour échapper aux optima locaux et explorer l'espace de recherche, afin d'atteindre les régions prometteuses. Initialement, il est constitué à partir de la population de solutions initiales, par les *b1* meilleures solutions en termes de qualité de la fonction objectif et des *b2* solutions les plus diversifiées. L'évaluation de la diversification dans la Recherche Dispersée est basée sur la distance entre les solutions. L'expression de cette distance dépend de la nature du problème considéré. En général, la taille de *R*, *b=b1+b2*, représentant le dixième de celle de la population est moyenne (de l'ordre de 20), comparée à celle des algorithmes génétiques, généralement, dépassant les 100 individus (Glover, 1998).

2.7 Mise à jour de l'ensemble de référence

La mise à jour de l'ensemble de référence (Marti, 2006) est une opération très importante dans la Recherche Dispersée. Elle doit maintenir l'équilibre entre la qualité et la diversité des solutions de l'ensemble. Les solutions, résultantes de l'opération de combinaison suivie de l'amélioration, sont à chaque itération, considérées pour leur éventuelle adhésion à l'ensemble de référence *R*. les solutions de haute qualité sont sélectionnées en fonction de la valeur de leur fonction objectif,

alors que les solutions les plus diversifiées le sont en fonction de leurs distances. Plusieurs méthodes, de mise à jour de l'ensemble de référence, peuvent être envisagées: la méthode statique, la méthode dynamique, la méthode 2-Tier (Marti, 2006) et la méthode 3-Tier (Campos et al., 2001).

2.7.1 Mise à jour statique

Dans le cas de l'opération de mise à jour statique (Marti, 2006), chaque solution, générée par la méthode de combinaison, est raffinée par la méthode d'amélioration avant d'être mémorisée. Lorsque tous les sous- ensembles à combiner sont traités, la mise à jour de l'ensemble de référence *R* est activée. Ainsi, *R* sera constitué des meilleures solutions résultantes de l'union des anciennes solutions de *R*, avec celles récemment mémorisées.

2.7.2 Mise à jour dynamique

Contrairement à la mise jour statique, la mise à jour dynamique (Marti 2006) est activée après la génération (par la méthode de combinaison) et l'amélioration de chaque nouvelle solution. Cette dernière est ensuite considérée, pour faire part éventuellement, de l'ensemble de référence. Si c'est le cas, la nouvelle solution est tout de suite utilisée dans les prochaines combinaisons. Il est cependant important de noter, que cette méthode engendre une convergence agressive, vu qu'elle remplace, plus rapidement, les solutions de qualité inférieure, par des solutions meilleures, au moment de l'opération de combinaison. En outre, ceci pourrait également conduire à écarter, éventuellement, certaines combinaisons prometteuses.

2.7.3 Mise à jour 2-Tiers

Dans une mise en œuvre simple de la Recherche Dispersée, la mise à jour de l'ensemble de référence considère comme seule mesure de qualité, la valeur de la fonction objectif. La mise à jour 2-Tiers (Marti, 2006) introduit en plus, la mesure de la diversité. Ainsi, l'ensemble de référence *R* est composé de deux sous ensembles: le sous ensemble *R1* contenant les *b1* solutions de haute qualité et le sous ensemble *R2* contenant les *b2* solutions les plus diversifiées. Chaque nouvelle solution, générée par la combinaison de solutions de *R* et améliorée, est considérée pour remplacer une solution de *R1*, ou bien une solution de *R2*, selon qu'elle améliore la qualité ou bien la diversité.

2.7.4 Mise à jour 3-Tiers

La mise à jour 3-Tiers proposée par Campos et al. (2001), est une extension de la méthode 2-Tiers. La méthode 3-Tiers ajoute pour la mise à jour, une autre mesure, dite mesure du *meilleur générateur*. Une solution est dite *bonne génératrice*, si elle génère une solution de haute qualité, en utilisant la méthode de combinaison, suivie de l'amélioration. L'ensemble de référence *R* est ainsi, composé des deux sous-ensembles précédents, *R1* et *R2*, plus le sous ensemble *R3* constitué par les *b3* meilleures solutions génératrices. La mise à jour de *R1* et *R2* est identique à celle de 2-Tiers. Et d'une manière similaire, une nouvelle solution est considérée, pour remplacer une solution de *R3*, si elle améliore la mesure de la meilleure génératrice.

A noter également, que l'initialisation du sous ensemble *R3* est faite, avec les meilleures solutions de la population initiale *P*, non incluses dans *R1*. La mise à jour 3-Tiers est très utile, dans le cas où les solutions, de qualité particulièrement basse, produisent des solutions de haute qualité, en appliquant l'opérateur de combinaison suivi de l'amélioration.

2.8 Génération des sous ensembles

La méthode de génération des sous ensembles, proposée par Glover (1998), consiste à générer quatre types de sous-ensembles sans duplication. Ces sous-ensembles, désignés par 2-SubSets, 3-SubSets, 4-SubSets et 5-SubSets, sont décrits comme suit :

2-SubSets : contenant tous les sous-ensembles distincts ayant deux éléments. Ils sont au nombre de *b*(b-1)/2* sous-ensembles différents, *b* étant le cardinal de *R*.

3-SubSets : composé de sous-ensembles à trois éléments générés chacun, à partir d'un sous-ensemble de 2-SubSets, auxquels on rajoute la meilleure solution de l'ensemble de référence n'appartenant pas au sous-ensemble. Sa taille est de *(b-1) (b-2)/2*.

4-SubSets: composé de sous-ensembles à quatre éléments, issus chacun, d'un sous ensemble de 3-SubSets augmenté de la meilleure solution de l'ensemble de référence non incluse dans le sous-ensemble. Sa taille est de *(b-2) (b-3)/2*.

5-SubSets : Composé de sous-ensembles constitués des *i* meilleures solutions, pour *i*=5, jusqu'à *b*. Sa taille est de *(b-4)*.

Le nombre total de tous ces sous ensembles est de *(3b-7) b/2*.

2.9 Combinaison des solutions

Dans la Recherche Dispersée, la méthode de combinaison combine les solutions de chaque sous ensemble généré par la méthode précédente, afin d'en produire de nouvelles. Différentes méthode de combinaison sont proposées dans la littérature. Elles sont généralement contexte- dépendantes. Parmi les plus connues, nous avons:

- ***Les méthodes de combinaisons basées sur les votes,*** permettant de déterminer, à partir des solutions à combiner, selon une stratégie, l'élément suivant à insérer dans la solution à produire. Quelques exemples de ces méthodes sont décrits dans (Campos et al., 2001).
- ***Les méthodes des λ-filtre,*** permettant de créer une combinaison de solutions, en considérant un vecteur filtre appliqué à l'union des solutions de référence. Le vecteur filtre permet d'extraire des informations partielles, à partir des solutions à combiner. Voir (Cavique et al., 2001).
- ***Les méthodes de combinaison pour les problèmes non linéaires, à variables bornées,*** consistant à trouver des combinaisons linéaires de solutions de références. Voir (Campos et al., 2001).
- ***Les méthodes de combinaison des algorithmes génétiques,*** qui sont également, utilisées dans la conception des méthodes de combinaison, pour des approches de la Recherche Dispersée (Glover et al., 2003).
- ***Le chemin reliant (Path relinking),*** proposée comme une approche pour intégrer les stratégies d'intensification et de diversification dans le contexte de la Recherche Taboue, peut être aussi considéré ici, comme une extension des mécanismes de combinaison de la Recherche Dispersée. Le chemin reliant reste l'une des méthodes les plus générales, qui ne dépendent pas de la nature du problème. Elle repose, sur le concept de la trajectoire entre les solutions, dans l'espace du voisinage. Elle produit généralement, de nouvelles solutions partageant un sous-ensemble considérable de caractéristiques, contenues dans les solutions parentes. Plusieurs variantes de cette méthode sont proposées, dans (Glover, 1998 ; Glover et al., 2000).

2.10 Reconstitution de l'ensemble de référence

L'opération de reconstitution de l'ensemble de référence *R* peut être activée, lorsque les méthodes de combinaison et d'amélioration des solutions de *R*, ne permettent plus d'en avoir de nouvelles, de qualité meilleure (Marti 2006). Son principe consiste à suivre les étapes suivantes :

1. Initialiser la population initiale avec, les *b1 meilleures* solutions de *R*.
2. Faire appel au générateur de diversification, pour compléter la population *P*.
3. Construire *R*, avec les *b1* meilleures solutions de *P* en termes d'objectif.
4. Compléter l'ensemble *R* par les *b2* solutions les plus diversifiées de *P-R*.

2.11 Conclusion

Dans ce chapitre, nous avons vu, que la Recherche Dispersée, une métaheuristique évolutionnaire basée population, permet d'utiliser des stratégies intelligentes, pour capturer l'information des solutions courantes, afin d'en construire de nouvelles. Son processus de recherche évolutif, basé sur un équilibre de la qualité et la diversité des solutions, favorise l'exploration des régions prometteuses, afin d'atteindre les solutions élites. Ainsi, de part sa philosophie, la Recherche Dispersée se prête bien à la résolution des problèmes NP-difficiles, issus de domaines différents.

Dans le chapitre suivant, nous présentons une description détaillée d'une méthode de recherche locale robuste : La Recherche Locale Guidée, dite 'Guided Local Serach (GLS).

Chapitre 3: Une recherche locale robuste : la Recherche Locale Guidée

3.1 Introduction

Le problème de l'explosion combinatoire des méthodes complètes, utilisée pour la résolution des problèmes d'optimisation NP-difficiles, peut être contourné en sacrifiant cette complétude. Les méthodes de recherche locales (Burke et al., 2003a) sont parmi les premières méthodes utilisées. Nous présentons d'abord, leur principe, en mettant l'accent sur leurs inconvénients, qui étaient à l'origine de l'élaboration, entre autres, de la méthode de la Recherche Locale Guidée (Guided Local Search ,GLS) par Voudouris et Tsang (1997, 1999). Puis, nous passons à la description détaillée de cette dernière avant de conclure.

3.2 Recherche Locale

Nous avons vu dans le chapitre 1, que les méthodes de recherche locale se caractérisent par la structure de leurs voisinages et la stratégie de leur exploration. Parmi les plus utilisées nous avons :

La marche aléatoire (Random Walk): c'est la plus simple recherche locale. Elle sélectionne aléatoirement, une solution du voisinage *N(s)*. Cette méthode est souvent utilisée, en combinaison avec d'autres techniques, comme une stratégie de diversification.

La descente simple (Simple Descent, *basic local search* ou hill climbing) : c'est la recherche locale classique, où à chaque itération, une solution candidate s' est sélectionnée aléatoirement, dans le voisinage *N(s) de s.* Si *f (s')* est une amélioration de *f (s)* alors s' est acceptée, pour remplacer s.

La méthode de descente, décrite dans l'Algorithme 3.1 et illustrée par la Figure 3.1, démarre avec une solution *s* aléatoire ou bien *g*énérée à l'aide d'une heuristique spécifique au problème. Une solution *s'*, dans un voisinage de *s*, est ensuite choisie, si elle vérifie *f(s') < f(s)(pour un problème de minimisation*) pour remplacer *s*. Ce processus s'arrête lorsqu'il n'y a plus d'amélioration.

L'inconvénient majeur avec cette méthode de recherche, c'est qu'elle se bloque au premier optimum local rencontré. En outre, la solution obtenue dépend fortement, non seulement, de la solution initiale, mais aussi, de la structure de voisinage utilisée.

Algorithme 3. 1 Méthode de la *descente simple*

// Choisir une solution initiale $s \in S$.
Répéter
Choisir *s'* dans *N(s) ;*
Si *f(s')* < *f(s)* alors $s \leftarrow s'$;
Jusqu'à (avoir $f(s') \geq f(s)$, $\forall s' \in S$).

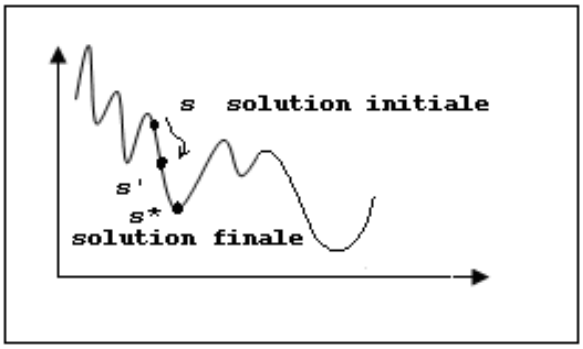


Figure 3. 1 Processus de la méthode de la descente

La meilleure descente (grande descente, Steepest Descent) : dans cette méthode, chaque solution du voisinage *N(s)* est évaluée. La solution s', ayant la meilleure évaluation de *f (s),* est retournée. Si *f (s')* améliore *f (s)* alors s' est acceptée pour remplacer s, sinon l'algorithme s'arrête. Voir Algorithme 3.2.

Notons que la méthode de la meilleure descente peut être très lente pour des voisinages vastes. La restriction des évaluations à un sous ensemble du voisinage *N(s)* choisi de manière aléatoire, ou bien en utilisant une stratégie, est souvent utilisée comme une solution à ce problème.

Algorithme 3. 2 Méthode de la grande *descente*

Choisir une solution initiale $s \in S$.
Répéter
Déterminer une solution s' qui minimise *f* dans *N(s) ;*
Si *f(s')* < *f(s)* **alors** poser $s \leftarrow s'$;
Finsi
Jusqu'à (*f(s')* ≥ *f(s)).*

Le principal défaut des méthodes de descente, malgré la simplicité de leur mise en œuvre, c'est qu'elles s'arrêtent au premier optimum local rencontré. En outre, un minimum local pour une structure de voisinage, ne l'est pas forcément pour une autre structure. Ainsi, la solution initiale, le choix de la structure de voisinage ainsi que sa taille, peuvent avoir un impact important sur l'efficacité d'une méthode de descente. Pour échapper à un optimum local, plusieurs métaheuristiques ont été proposées, telles que la méthode du Recuit Simulé (Kirkpatrick et al., 1983) et la Recherche Tabou (Di Gaspero et Schearf, 2001; Glover 1989; Thompson et Dowsland, 1998). Ces méthodes acceptent de temps à autre, une certaine dégradation des solutions trouvées, pour mieux explorer l'espace de recherche.

En 1997, Voudouris et Tsang ont proposé une nouvelle métaheuristique appelée Recherche Locale Guidée. Cette dernière se présente, comme une méthode de recherche locale élaborée, dans laquelle la fonction objectif varie durant le processus de recherche. Le but étant de rendre les optima locaux, déjà visités, moins attractifs afin d'explorer des régions prometteuses, Figure 3.2.

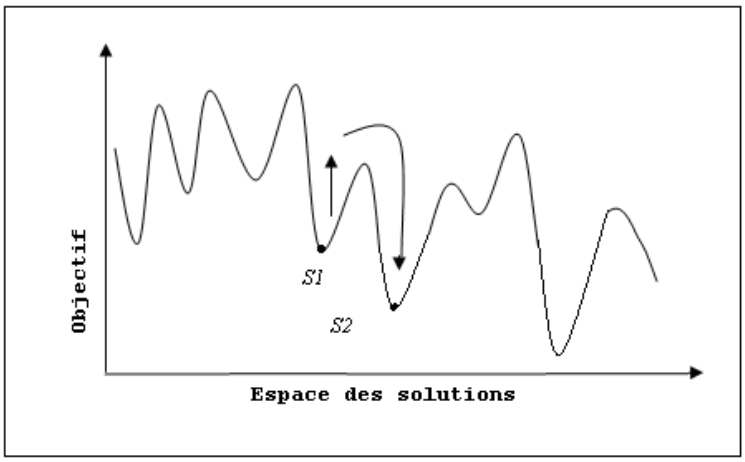


Figure 3. 2 L'idée de base de la GLS c'est d'échapper à l'optimum local

La Recherche Locale Guidée, utilisée pour trouver une solution optimale à la fonction objectif considérée, est élaborée à partir des travaux antérieurs sur une approche pour la satisfaction des contraintes nommée Genet (Davenport, 1997). Genet est un algorithme inspiré des réseaux de neurones, élaboré pour trouver une solution satisfaisant toutes les contraintes. Son principe se base sur une pondération des contraintes au niveau de la fonction objectif, permettant d'échapper à l'optimum local.

La Recherche Locale Guidée appartient à la famille des algorithmes utilisant des pénalités ou des pondérations pour modifier la fonction objectif à optimiser, afin d'échapper à l'optimum local. Cette famille inclut :

- La Recherche Tabou, basée pénalités, associant au niveau de la fonction objectif, une pénalité à chaque membre de la liste tabou (Fox, 1993).

- La Recherche Tabou basée sur la mémoire des fréquences, utilisant dans la fonction objectif des poids permettant de pénaliser des mouvements ou des attributs de solutions, s'ils apparaissent plus fréquemment (Glover et Laguna, 1997).

Contrairement à La Recherche Tabou qui se focalise sur les membres de la liste taboue, la GLS pénalise seulement les attributs apparaissant dans l'optimum local. Par ailleurs, dans la GLS, les attributs, appelés aussi caractéristiques, ne permettent pas seulement à la recherche Locale d'échapper à un optimum local, mais surtout de prospecter les régions où les attributs pénalisés n'apparaissent pas. Ainsi, les pénalités peuvent êtres vus comme des attributs tabous, faisant de la Recherche Locale Guidée une forme de Recherche Tabou.

La GLS est exploitée avec succès, pour divers problèmes NP-difficiles, tels que l'affectation de fréquences (Voudouris, 1997,1999), les problèmes d'ordonnancement de la main d'œuvre (Tsang et Voudouris, 1997), l'acheminement (Voudouris, 1997), le problème Max SAT (Mills et Tsang, 2000) et également, les problèmes classiques, tels que le problème du voyageur de commerce (TSP), et le problème d'affectation quadratique (Voudouris, 1997; Voudouris et Tsang, 1999).

3.3 Description de la Recherche Locale Guidée

Principalement, la GLS emploie les informations intrinsèques au problème et au processus, de recherche, de manière dynamique, afin de guider la recherche, vers les régions prometteuses. Pour cela, elle effectue des appels successifs, à une procédure de recherche locale. Lorsqu'un optimum local est atteint, la fonction objectif du problème est alors augmentée, avec un ensemble de pénalités, associées aux attributs indésirables présents dans cet optimum local. Ces attributs sont qualifiés d'indésirables, car normalement, non présents dans une solution optimale. Autrement dit, les attributs indésirables de l'optimum local sont pénalisés de manière à ce que cette solution devienne plus couteuse, donc moins attractive que ses voisines, où ces attributs sont absents.

Dans la GLS, chaque attribut a un coût et une pénalité. Le coût se rapporte à la fonction objectif originale, les pénalités initialisées à 0 au commencement sont mises à jour, au cours du processus de recherche, lorsqu' un minimum local est atteint.

3.3.1 Attributs et pénalités d'une solution

Le mécanisme utilisé par la GLS est essentiellement basé, sur les attributs représentant n'importe quelle caractéristique permettant de distinguer les solutions. Les attributs sont des propriétés qui ne sont pas, nécessairement présentes dans toutes les solutions. Leur choix dépend du problème considéré. Par exemple, Pour le Problème du Voyageur de Commerce PVC, on peut par exemple, associer un attribut à chaque arête du graphe correspondant. On dira qu'une tournée possède l'attribut *Ae* si l'arête *e* fait partie de la tournée (Voudouri, 1999). Pour le problème SAT, un attribut peut être le nombre de clauses non satisfaites (Mills et Tsang, 2000).

On fait correspondre à chaque attribut :

- **Un indicateur de la fonction** $I_i(s)\,(1)$ indiquant si l'attribut *i* est présent ou non dans la solution *s:*

$$I_i(s) = \begin{cases} 1 \text{ si l'attribut est present dans la solution} \\ 0 \quad \text{sinon} \end{cases} \tag{3.1}$$

- **Une pénalité** p_i , initialisée à 0, pour pénaliser les occurrences de l'attribut i. Dans un optimum local, les pénalités sont incrémentées (3.3) pour les attributs ayant l'utilité maximale, définie en fonction du coût et de la pénalité (3.2). On note ici, que plus un attribut est pénalisé, moins il le sera encore.

$$util(s^*, f_i) = I_i(s^*) \cdot \frac{c_i}{1 + p_i} \tag{3.2}$$

$$p_i = p_i + 1 \tag{3.3}$$

- **Un coût** c_i assigné à l'attribut *i*, présent dans *s*, pour exprimer l'importance de l'attribut relativement aux autres. Plus grande est sa valeur, plus grande est son utilité. Dans l'expression de l'utilité, le coût est équilibré par la valeur du paramètre pénalité, pour tenir compte de l'historique et empêcher que l'algorithme ne soit entièrement biaisé par la valeur du coût uniquement.

3.3.2 Fonction objectif augmentée

La GLS modifie la fonction objectif, pour échapper à l'optimum local atteint par la recherche locale. Elle incrémente les pénalités (initialisées à 0) des attributs d'utilité maximale, présents dans la solution. L'augmentation de la fonction objectif, avec ces pénalités, permet à la GLS de guider la recherche loin de l'optimum local; puisque, celui-ci aura une valeur de l'objectif

supérieure à celle des solutions de son voisinage, où ces attributs ne sont pas présents. Notons que si un attribut est pénalisé plusieurs fois, son utilité diminue de manière à réduire les chances de le pénaliser encore.

La fonction f est remplacée par la fonction augmentée *h* (3.4) définie comme suit:

$$h(s) = f(s) + \lambda \cdot \sum_{i=1}^{K} p_i \cdot I_i(s) \qquad (3.4)$$

h étant la fonction objectif augmentée, *s* la solution obtenue, *k* le nombre d'attributs, *f(s)* le coût de la solution s, I_i l'indicateur pour l'attribut *i*, p_i la pénalité de *i*, λ un facteur régulateur de l'intensification s'il est proche de 0, et de la diversification s'il est loin de 0.

L'article initial de Voudouris et Tsang, soumis en octobre 1995 et publié en 1999 (voudouris, 1999), présente une description de cette méthode. Pour une étude approfondie, voir la thèse de Voudouris (1997).

3.4 Algorithme de la Recherche Local Guidée

Le principe de l'algorithme de la GLS, décrit par l'Algorithme 3.3, consiste à générer une solution initiale de manière aléatoire, ou en utilisant une heuristique. On applique ensuite pour la fonction objectif *g* une recherché locale, jusqu'a atteindre un optimum local. Les attributs de cet optimum, d'utilité maximale, verront leurs pénalités (initialisées à 0) incrémentées de 1. La recherche locale est ensuite, relancée encore à nouveau, mais avec la fonction *h*. Ces étapes sont répétées successivement, dans un processus itératif, jusqu'a la satisfaction de la condition d'arrêt spécifiée (e.g. un nombre d'itérations maximal, un temps imparti ou bien une valeur particulière de la fonction objectif). A la fin du processus, la meilleure solution, trouvée par rapport à la fonction objectif originale *g*, est retournée, Figure 3.3.

Algorithme 3. 3 Algorithme basique de la Recherche Locale Guidée

Générer une solution initiale $s \in S$; poser s*:=s;
Initialiser les pénalités à 0 ;
Tantque (le critère d'arrêt non atteint) **faire**
    Augmenter la fonction objectif originale, *g*, avec les pénalités. Soit *h* cette fonction ;
    Appliquer une recherche locale avec *h* pour obtenir la solution *s' ;*
    Incrémenter les pénalités des attributs de la solution *s',* ayant les utilités maximales ;
    poser s=s';
FinTantque
Retourner la meilleure solution s* trouvée par rapport à la fonction originale *g.*

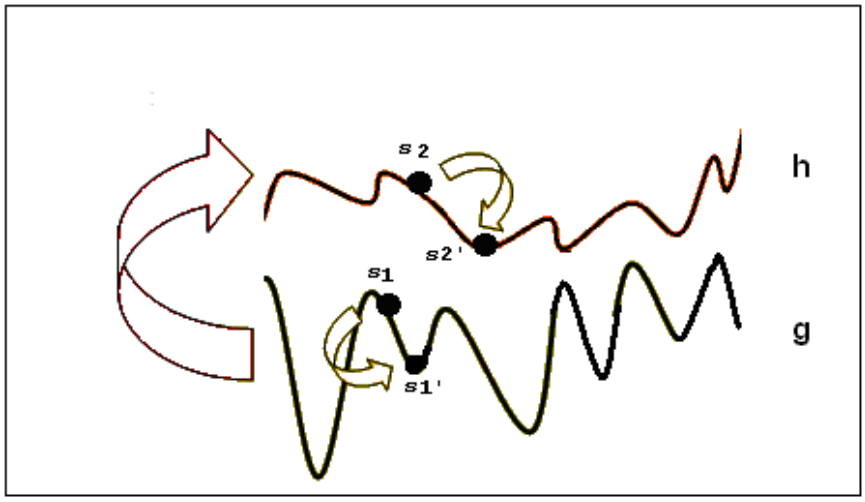


Figure 3. 3 Changement dynamique de la fonction objectif g.

Dans cet algorithme basique de la GLS, l'intensification est assurée par la recherche locale. Le calcul des expressions d'utilités, des différents attributs d'une solution, permet pour celles qui sont maximales, de pénaliser la fonction objectif. Ainsi, la recherche sera dirigée dans d'autres directions que celles qui semblaient prometteuses, lors des précédentes étapes. Cette mise à jour des pénalités remplit, par conséquent, les objectifs de diversification de la recherche.

3.5 Stratégies d'aspiration et mouvements aléatoires dans la Recherché Locale Guidée

Tout comme pour la Recherche Tabou, nous pouvons dans le cas de la Recherche Locale Guidée, envisager une stratégie d'aspiration et une stratégie de mouvements aléatoires pour tenter d'améliorer ses performances et explorer plus profondément l'espace de recherche (Mills, 2001).

3.5.1 Stratégie d'aspiration

La stratégie d'aspiration est un concept utilisé dans la Recherche Tabou (Glover, F. 1989). Il permet de débloquer une situation, où toutes les solutions du voisinage sont taboues. Avec cette stratégie, une solution taboue remplace la solution courante, afin de résoudre un problème généré par le statu tabou d'un attribut ou d'un mouvement qui nous éloigne d'une région, pourtant intéressante. Le plus utilisé des critères d'aspiration est le critère de la meilleure amélioration, où une nouvelle solution, de meilleure qualité que les solutions déjà obtenues, est atteinte par un mouvement tabou. Le statut tabou de ce mouvement est alors ignoré et le mouvement exécuté. D'autres critères d'aspirations peuvent êtres trouvés dans (Glover et Laguna, 1997).

Contrairement à la Recherche Tabou basée soit sur une liste d'attributs tabous, soit sur une liste de mouvements tabous, la Recherche Locale Guidée impose un ensemble de pénalités sur des attributs apparaissant dans un optimum local. Ainsi, le critère d'aspiration basé sur le meilleur mouvement d'amélioration, dans la Recherche Locale Guidée, peut être défini comme étant un mouvement qui a généré la meilleure solution jusque là, et qui n'aurait pas été choisi par la recherche locale utilisant la fonction objectif augmentée sans l'aspiration.

3.5.2 Mouvements aléatoires

La stratégie des mouvements aléatoires, permet dans un processus de recherche, d'effectuer, selon un critère donné, des mouvements dans le voisinage d'une solution, vers des solutions non nécessairement les meilleures. Celle inspirée des travaux de Selman et al.(1994) sur GWSAT et WalkSat, permet d'effectuer, avec une probabilité fixée, des mouvements dans le voisinage, vers des solutions non nécessairement les meilleures. Cette action réduit l'effet de l'opération de l'intensification, et favorise l'exploration de nouvelles régions de l'espace, afin d'éviter au processus de recherche d'être piégé dans un optimum local. Dans notre contexte plusieurs stratégies de mouvements aléatoires peuvent être envisagées, nous en décrirons deux, au chapitre5.

3.6 Conclusion

Dans ce chapitre, nous avons vu que le principal défaut des méthodes de recherche locale simples, malgré la simplicité de leur mise en œuvre, c'est qu'elles s'arrêtent au premier optimum local rencontré. En outre, un optimum local pour une structure de voisinage dans ces méthodes, ne l'est forcément pas pour une autre structure. De même, la solution initiale, le choix de la structure de voisinage, ainsi que sa taille, peuvent avoir un grand impact, sur l'efficacité d'une méthode de descente.

Pour échapper à un optimum local, plusieurs métaheuristiques ont été proposées. Parmi les plus récentes, nous avons la Recherche Locale Guidée (GLS). Cette dernière se présente, comme une méthode de recherche locale élaborée, où la fonction objectif varie durant le processus de recherche. Le but étant de rendre les optima locaux déjà visités, moins attractifs, afin d'explorer des régions prometteuses.

Par ailleurs, nous avons vu aussi, que tout comme pour les autres méthodes de recherche locales, il est intéressant de rajouter à la GLS des stratégies connues, comme l'aspiration et les

mouvements aléatoires afin de tenter d'améliorer ses performances et d'explorer plus profondément l'espace de recherche.

Dans les chapitres suivants, nous allons décrire nos travaux, ayant fait l'objet de publications, et communications, ou en soumission. Voir CIAA'05, JNAM'07, COSI'07, IAJIT 2008 vol. 5, ICMWI'2010 et IJAISC 2010 vol. 2.

Nous commençons dans la suite, par la description des différentes composantes des premières approches évolutionnaires basées sur la Recherche Dispersée, pour le problème d'emploi du temps des examens.

Chapitre 4: La Recherche Dispersée pour le problème d'emploi du temps des examens

4.1 Introduction

Dans ce chapitre, nous présentons en premier lieu, le problème étudié et le modèle mathématique que nous lui avons associé. Ensuite, nous donnons une description de nos différentes composantes pour concevoir les premières approches, (à notre connaissance), de la Recherche Dispersée, pour le problème d'emploi du temps des examens. Cette description, sera jalonnée de nombreux algorithmes permettant de suivre en détail, toutes les étapes de nos investigations.

4.2 Description du problème

Le problème d'emploi du temps des examens considéré peut être décrit comme étant, la donnée de *N* examens à affecter à un ensemble limité de *P* périodes contiguës. Chaque examen doit être programmé à une et une seule période, en évitant, absolument, les chevauchements entre les examens ayant des étudiants en commun et espaçant, autant que faire se peut, les examens pour chaque étudiant.

Dans notre étude, nous considérons les hypothèses de Carter et al., (1996) pour construire nos approches et leurs problèmes de test pour effectuer nos expérimentations.

Les hypothèses ainsi que les problèmes de test de Carter et al., (1996) disponibles via: *ftp://ftp.mie.utoronto.ca/pub/carter/tesprob/*, constituent, depuis leurs publications, une référence incontournable dans la communauté du problème d'emploi du temps, en permettant d'effectuer des études comparatives, entre les différentes approches de la littérature (Burke et al., 1998; Schaerf, 1999; Burke et al., (2004b).

Soient :

N : Le nombre des examens.

P : le nombre de périodes.

M : Le nombre des étudiants.

$C = (c_{ij})_{N \times N}$ La matrice des conflits où chaque élément noté c_{ij}, $i,j \in \{1,...,N\}$ représente le nombre des étudiants inscrits aux examens *i et j*.

On suppose comme Carter et al., (1996), qu'il n'y a pas de limite pour les places d'examens disponibles, et les pénalités de l'espacement des examens définies comme suit :

Lorsqu'un étudiant assiste à deux examens séparés par *s* périodes, la valeur de la pénalité (coûts), w_s correspondante est telle que :

$w_1 = 16,\ w_2 = 8,\ w_3 = 4,\ w_4 = 2,\ w_5 = 1$ et $w_s = 0$ si $s > 5$).

Il est clair qu'il existe une relation étroite, entre le modèle associé à un problème donné et la méthode de résolution à adopter. Dans notre cas, vues les méthodes considérées dans notre étude, nous avons modélisé notre problème, comme un problème d'optimisation combinatoire, où la fonction objectif, représentée par L'expression (4.1), fournit la pénalité moyenne par étudiant. Plus petite est sa valeur, meilleure est la qualité de la solution. L'expression (4.2) représente la pénalité entre deux examens programmés dans deux périodes différentes *q* et *p*. Les contraintes impératives sont représentées par les expressions (4.3) et (4.4). L'expression (4.3) exprime le fait que chaque examen doit être programmé à une et une seule période d'examen. L'expression (4.4) spécifie que les examens, ayant des étudiants en commun (des examens en conflits), ne doivent pas être programmés à la même période.

Il est important de noter, que dans la plupart des cas, il n'est pas toujours possible de construire une solution réalisable, c'est-à-dire où tous les examens sont affectés à des périodes valides. En outre, il est rapporté, que le désir excessif de vouloir satisfaire toutes les contraintes fortes, peut conduire à des résultats médiocres (Ross, 1996). Nous décidons par conséquent, d'affecter à une période extra, la période *(P+1)*, chaque examen ne pouvant être affecté à une période valide, et ce, en lui attribuant une pénalité plus élevée (Burke et al., 2004; Burke et Newal, 2004). Ainsi nous encouragerons progressivement, l'affectation de tous les examens, de manière à obtenir une solution réalisable utilisant *p* périodes.

La formulation mathématique de notre problème se présente alors, comme suit :

Minimiser

$$g(t) = (\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \sum_{p=1}^{P} \sum_{q=1}^{P} W(p,q) t_{ip} t_{jq} c_{ij} + \sum_{i=1}^{N} 5000\, t_{i(P+1)}) / M. \quad (4.1)$$

$$W(p,q) = 2^{5-|q-p|} \text{ si } 1 \le |q-p| \le 5 \text{ ; 0 sinon.} \quad (4.2)$$

Sous les contraintes $\sum_{p=1}^{P+1} t_{ip} = 1 \quad \forall i \in \{1,.....N\}$ (4.3)

$$\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \sum_{p=1}^{P} t_{ip} t_{jp} c_{ij} = 0. \quad (4.4)$$

$$t_{ip} \in \{0,1\} \qquad i = 1..N, \quad p = 1..P+1$$

Où c_{ij} est un entier non négatif indiquant le nombre des étudiants participants aux examens *i* et *j* , $t = (t_{11},.....,t_{n(P+1)})$ une matrice binaire des variables de décision.

$$t_{ip} = \begin{cases} 1 & \text{si et seulement si l'examen i est affecté à la période p} \\ 0 & \text{sinon} \end{cases}$$

L'ensemble des solutions réalisables, S se présente donc, comme suit:

$$\text{S} = \{\ t_{ip} \in \{0,1\}; i = 1..N, p = 1..P+1 \mid \sum_{p=1}^{P+1} t_{ip} = 1, i = 1..N, \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \sum_{p=1}^{P} t_{ip} t_{jp} c_{ij} = 0\ \}.$$

La valeur 5000 est une constante choisie, pour pénaliser fortement, tout examen affecté à la période P+1. Pour l'illustration, la Table 4.1. Présente un exemple de matrice de conflits *C*, pour un nombre d'examens *N=10*.

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
e_1	0	5	0	7	1	0	5	7	7	0
e_2	5	0	7	10	5	5	0	0	3	2
e_3	0	7	0	0	10	2	0	6	7	0
e_4	7	10	0	0	3	1	6	4	0	0
e_5	1	5	10	3	0	3	0	1	0	0
e_6	0	5	2	1	3	0	0	0	4	3
e_7	5	0	0	6	0	0	0	4	0	0
e_8	7	0	6	4	1	0	4	0	5	3
e_9	7	3	7	0	0	4	0	5	0	0
e_{10}	0	2	0	0	0	3	0	3	0	0

Table 4. 1 Un exemple de la matrice de conflits *C*

Si on suppose que le nombre de périodes est $P= 6$, alors, une solution réalisable peut être définie par le vecteur V=(1,2,1,3,4,5,2,5,4,3) où chaque $V(i)$ correspond à la période de l'examen i. Une des représentations de cette solution est donnée dans la Table 4.2. Les pénalités entre les examens sont alors, celles reportées dans la. Table 4.3. A titre d'exemple, la pénalité entre les examens e_2 et e_3 est égale à 16.

Examination	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
Période	P_1	P_2	P_1	P_3	P_4	P_5	P_2	P_5	P_4	P_3

Table 4. 2 Affectation des examens aux périodes

.

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
e_1	-	16	0	8	4	2	16	2	4	8
e_2	-	-	16	16	8	2	0	2	8	16
e_3	-	-	-	8	4	2	16	2	4	8
e_4	-	-	-	-	16	8	16	8	16	0
e_5	-	-	-	-	-	16	8	16	0	16
e_6	-	-	-	-	-	-	4	0	16	8
e_7	-	-	-	-	-	-	-	4	8	16
e_8	-	-	-	-	-	-	-	-	16	8
e_9	-	-	-	-	-	-	-	-	-	16
e_{10}	-	-	-		-		-	-	-	-

Table 4. 3 La matrice des pénalités entre les examens de la solution

4.3 Approches de la Recherche Dispersée pour le problème d'emploi du temps des examens

Le principe de la Recherche Dispersée consiste à démarrer avec une population de bonnes solutions initiales, variées et dispersées. Puis, un ensemble de référence est construit avec les *b1* meilleures solutions de la population en termes de qualité de la fonction objectif, et les *b2* solutions, les plus espacées les unes des autres. Une stratégie de combinaison de ces solutions, suivie d'une méthode d'amélioration puis d'une méthode de mise à jour de l'ensemble de référence, sont ensuite répétées dans un processus itératif et évolutif, afin d'atteindre des solutions élites.

Tel que nous l'avons déjà vu précédemment, la Recherche Dispersée peut être implémentée de différentes manières. Dans cette section, nous décrivons dans un ordre similaire à celui du chapitre 2, les différentes méthodes utilisées dans la construction de nos approches pour le problème d'emploi du temps des examens.

4.3.1 Méthode de Génération de diversification

Rappelons qu'à chaque solution d'un problème d'emploi du temps, correspond la valeur de sa fonction objectif (fonction pénalité), représentant le coût de violation des contraintes souhaitables. Plus petite est sa valeur, meilleure est la qualité de la solution. La méthode de génération de diversification est utilisée pour créer une population de solutions initiales variées, dispersées et de bonne qualité, afin de donner à l'approche durant l'exploration de l'espace de recherche, le maximum d'opportunités pour atteindre l'objectif à optimiser. Il existe dans la littérature plusieurs méthodes de génération de diversification, généralement, très dépendantes du contexte d'étude.

Afin d'élaborer une méthode ad hoc de génération de solutions initiales variées, dispersées et de bonne qualité, nous avons dans notre contexte, étudié pour la Recherche Dispersée plusieurs variantes où les heuristiques de coloration de graphe jouent un rôle prépondérant.

4.3.1.1 Méthode de génération basée sur une heuristique de construction

Cette variante de génération est utilisée pour une étude comparative de quatre heuristiques de construction de solutions initiales. L'objectif étant de déterminer la meilleure d'entre elles, afin de l'utiliser dans la suite du travail. Les heuristiques choisies pour l'étude sont ***l'heuristique random*** (RD), ainsi que trois autres heuristiques proposées à l'origine, pour le problème de coloration de graphe et adaptées à notre contexte : ***L'heuristique du plus grand degré*** **(***Largest Degree heuristic* (LD)), ***L'heuristique du plus grand degré pondéré*** **(***Largest Weighted Degree heuristic* (LWD), *et* ***L'heuristiques du degré de saturation*** (*Saturation Degree heuristic* (SD)*) (Brelaz, 1979;* Burke et al., 1998; Carter et al., 1996).

L'heuristique du plus grand degré **(LD)**: range les examens dans un ordre décroissant, en fonction du nombre de conflits qu'ils ont avec les autres. L'objectif étant de programmer les examens qui ont le plus de conflits d'abord.

L'heuristique du plus grand degré pondéré **(LWD):** range les examens dans un ordre décroissant, en fonction du nombre de conflits qu'ils ont avec les autres, pondérés par le nombre d'étudiants en commun. LWD donne la priorité aux examens, où des groupes importants d'étudiants ont des conflits similaires.

L'heuristique du degré de saturation **(SD)**: range les examens non encore programmés, dans un ordre croissant, en fonction du nombre courant de périodes disponibles. Ainsi, à chaque étape de la construction de la solution, la priorité des examens change dynamiquement avec la situation rencontrée.

Dans la suite, l'Algorithme 4.1 décrit la génération de solutions initiales, basée sur l'une des heuristiques RD, LWD, LD et SD.

Par ailleurs, pour couvrir une large part de l'espace de recherche, nous introduisons dans le cas des heuristiques LWD, LD et SD, une technique de randomisation, afin de générer des solutions différentes. Pour cela, nous combinons une heuristique de coloration de graphe avec le processus de sélection par tournoi (Burke et al., 1998; Carter et al., 1996). La génération d'une solution initiale démarre par la sélection aléatoire, d'un sous-ensemble (tournoi) d'examens qui sera ordonné, en utilisant l'heuristique choisie. Ensuite, dans l'ordre, chaque examen du tournoi est affecté à une période valide (sans violation des contraintes impératives) ayant la plus faible pénalité. Ces étapes sont répétées dans un processus itératif jusqu'à l'épuisement de tous les examens.

Remarque

Pour toutes nos méthodes, il est important de noter les points suivants :

- Pour traiter les cas ex aequo dans les stratégies LWD et SD, nous employons la stratégie *LD.*
- Dans le cas où nous avons le choix entre plusieurs périodes, alors l'examen est programmé à la première période rencontrée.
- Si aucune période n'est valide, l'examen est affecté à la période, $p+1$, associée à la liste des examens non programmés.

Une autre manière pour maintenir la diversité dans une population générée par cette méthode, est de considérer, de temps en temps, quelques affectations (e.g. c=5% du nombre total des examens) d'examens de la liste totale, aux périodes qui n'ont pas été fréquemment employées. Ainsi, nous créons une solution partielle, en affectant aux périodes qui n'ont pas été fréquemment employées, les $c \times N$ premiers examens de l'ensemble total (ordonné dans le cas des heuristiques, SD, LWD et LD). Les $N - (c \times N)$ examens restants sont ensuite, traités à tour de rôle, en respectant l'ordre de l'heuristique considérée. Cette diversification de la population est activée, toutes les dix constructions de solutions réalisables.

Nota: Dans le chapitre 6, nous étudierons une approche de la Recherche Dispersée, basée sur la SD, désignée par SD-SS.

Algorithme 4. 1 Génération de diversification basée sur une heuristique

---

{Pour générer un emploi du temps.}
Répéter
Construire aléatoirement à partir de la liste d'examens, un tournoi d'examens de taille T ;
{T= un pourcentage de la taille totale des examens; par exemple 20% de la taille totale}
Ordonner le tournoi, seulement dans les cas des stratégies LWD, LD ou SD;
Pour chaque examen *e* du tournoi choisi dans l'ordre **faire**
Affecter *e* à une période valide causant la plus faible pénalité ;
Fait
Jusqu'à (l'épuisement des examens).
{Pour traiter les cas ex aequo dans LWD et SD, utiliser la stratégie LD. S'il y a un cas d'ex aequo entre les périodes, alors affecter l'examen e à la première période rencontrée. Si aucune période n'est valide, alors ajouter l'examen à la liste des examens non programmés.}

4.3.1.2 Méthode de génération basée sur l'hybridation de LWD et SD

Dans cette variante décrite dans l'Algorithme 4.2, que nous désignerons par LWD-SD, nous combinons les heuristiques de coloration de graphe LWD et SD en attribuant la priorité aux examens ayant les plus grands degrés de conflits pondérés[4] avec les autres (car plus difficiles à programmer). Par ailleurs, pour éviter d'avoir une population de solutions similaires, nous introduisons la notion de fréquence de l'exploitation des périodes, pour une diversification des solutions.

Ainsi, pour générer une solution de la population initiale, tout en considérant la remarque précédente, nous affectons de manière aléatoire, $c \times N$ examens à des périodes qui n'ont pas été fréquemment utilisées. Les $N - (c \times N)$ examens restants sont ordonnés avec la stratégie LWD. Puis, dans l'ordre, les M (pourcentage empirique) premiers examens sont affectés chacun à une période valide ayant la pénalité minimale. Les examens restants sont ensuite, ordonnés suivant la stratégie SD, et affectés aussi dans l'ordre, chacun à une période valide ayant la pénalité minimale.

Par ailleurs, une manière pour maintenir la diversité dans une population générée par cette méthode, consiste à créer, de temps en temps, une solution en utilisant la première variante basée

[4] Chaque conflit est pondéré par le nombre d'étudiants impliqués dans les deux examens en conflit.

sur la SD. Cette diversification de la population est activée toutes les dix constructions de solutions réalisables.

Dans le chapitre 6, nous étudierons une approche de la Recherche Dispersée basée sur cette méthode désignée par LWDSD-SS.

Algorithme 4. 2 LWD-SD : Génération de diversification basée sur LWD et SD

{Pour générer un emploi du temps}

Affecter à partir de la liste des examens d'ordre aléatoire, $c \times N$ examens aux périodes qui n'ont pas été fréquemment employées; *{ex. c = 0.05}*

Ordonner le reste des examens en utilisant la stratégie LWD;

Affecter dans l'ordre les *M* premiers examens, chacun à une période valide ayant la plus faible pénalité ; *{M étant un pourcentage empirique}*

Avec la stratégie (SD), ordonner les examens restants, puis les affecter dans l'ordre, chacun à une période valide ayant la plus faible pénalité;

{Pour traiter les cas ex aequo avec les stratégies LWD et SD, utiliser la stratégie LD. S'il y a un cas d'ex aequo entre les périodes, alors programmer l'examen à la première période rencontrée. Si aucune période n'est valide, alors ajouter l'examen à la liste des examens non programmés.}

4.3.1.3 Méthode de génération basée sur GRASP et SD

Dans cette variante décrite par l'Algorithme 4.4, que nous désignerons par G-SD, nous combinons l'heuristique de coloration de graphe SD avec la procédure Grasp caractérisée par sa fonction adaptative.

Grasp est introduite en 1989, par Feo et Resende et exploité avec succès dans plusieurs travaux sur des problèmes d'optimisation combinatoire (Binato et al., 2001 ; Festa et Resende, 2002; Mavridou et al., 1998). C'est une procédure itérative gloutonne, adaptative et randomisée où chaque itération se compose de deux phases : une phase de construction et une phase d'amélioration basée sur une recherche locale. A l'issue du processus itératif, la meilleure solution globale constituera le résultat final (Feo et Resende, 1989; Festa et Resende, 2002; Resende et Pitsoulis, 2002).

1- La phase de construction : au cours de cette phase, une solution réalisable est construite itérativement pas à pas, à partir d'une liste d'attributs, dite Restricted Candidate List, (RCL). La RCL, constituée par des fragments d'une solution réalisable, est ordonnée suivant une heuristique

choisie et mise à jour, après l'affectation de chacun de ses attributs. Cette mise à jour dynamique est basée sur une procédure adaptative prenant en considération les changements antérieurs. La RCL peut être gérée selon deux procédures :

a. La liste est alimentée avec les T meilleurs candidats de l'ensemble des éléments rangés selon une heuristique adaptée au problème, et le choix de l'un de ses éléments pour construire la solution, est aléatoire.

b. La liste est alimentée avec les T candidats sélectionnés aléatoirement à partir de l'ensemble des éléments, et le choix de l'un de ses éléments pour construire la solution est sélectionné en fonction d'une heuristique spécifique et adaptée au problème considéré.

L'Algorithme 4.3 décrit les étapes de la méthode Grasp dans sa forme basique. Dans notre étude, nous avons opté pour la procédure b). Notons que l'étape de mise à jour de la RCL est un des points clés de cette méthode, et que la taille de la RCL est un paramètre important. Elle peut être fixée pour tout l'algorithme, ou bien variée de manière adaptative. Si sa taille vaut 1 alors on est ramené à un algorithme constructif, purement aléatoire. Dans le cas où elle est égale au nombre total d'examens, l'algorithme revient à appliquer au problème, l'heuristique.

Algorithme 4. 3 Algorithme basique de GRASP

```
f*←∞ ;
Répéter
   s ← ∅; {s une solution vide}
   Répéter
       Construire RCL à partir de l'ensemble des attributs d'une solution;
       Tantque (RCL est non vide )
                  x← select(RCL);
                s ←  s ∪ {x};
                Miseajour(RCL);
       FinTantque
   Jusqu'à (s complétée)
   RechercheLocale(s) ;
   Si f(s) < f* alors s* ← s et f*← f(s);
   Finsi
Jusqu'à (critère d'arrêt atteint) ;
Retourner s*.
```

2 - La phase d'amélioration : étant donné que la solution produite par la phase précédente n'est pas nécessairement la meilleure dans son voisinage, cette phase consiste à remplacer itérativement la solution courante par une autre du voisinage, de qualité meilleure, jusqu'à ce qu'il n'y ait plus d'amélioration.

Notons que, dans notre méthode Grasp, l'intensification est obtenue, non seulement par la recherche locale, mais également, par la mise à jour de la liste des attributs permettant de contrôler l'intensification et par voie de conséquence, la convergence de la méthode. La diversification est assurée par contre, par l'algorithme glouton. En effet, ce dernier peut commencer à construire une bonne solution, puis s'´eloigner rapidement à chaque étape d'une bonne solution très proche, a cause du choix aveugle des éléments pour la RCL. Généralement, la méthode GRASP est classée moins performante que les autres métaheuristiques à cause de l'absence de mémoire permettant d'éviter l'exploration des régions déjà visitées.

Le principe de la G-SD, que nous proposons ici, est semblable à celui de la variante 1, utilisant l'heuristique SD, décrite par l'Algorithme 4.1. Cependant, la liste RCL est mise à jour, de manière adaptative, après l'affectation de chacun de ses examens à une période valide, afin de tenir compte, de manière dynamique, des états de la solution, en phase de construction.

Comme dans la variante1, une autre manière pour maintenir la diversité dans une population générée par cette méthode, est de considérer, de temps en temps, quelques affectations d'examens aux périodes qui n'ont pas été fréquemment employées.

Ainsi, nous affectons aux périodes qui n'ont pas été fréquemment employées, $c \times N$ examens de l'ensemble total, ordonné avec la SD. Les examens restants sont ensuite, traités à tour de rôle, en utilisant cette fois-ci, un ordre adaptatif de la SD, c'est-à-dire que les examens restants sont réordonnés avec la SD, à l'issue du le traitement de chaque examen. Cette diversification de la population est activée, après toutes les dix constructions de solutions réalisables.

Nota: Dans le chapitre 6, nous décrirons l'étude d'une approche de la Recherche Dispersée basée sur cette méthode désignée par G-SD-SS.

Algorithme 4. 4 G-SD: Génération de diversification utilisant une GRASP et SD

{Pour générer un emploi du temps.}
Répéter
Construire aléatoirement, à partir de la liste d'examens, une liste (RCL) de taille *T;*
{T : un pourcentage de la taille totale des examens, par exemple 20% de la taille totale}
Ordonner la RCL avec la stratégie SD;
Pour chaque examen *e* de la liste, choisi dans l'ordre **faire**
Affecter *e* à une période valide causant la plus faible pénalité;
Mettre à jour l'ordre des examens restants dans RCL, en utilisant SD;
Fait
Jusqu'à (l'épuisement des examens).
{Pour traiter les cas ex aequo avec la SD, utiliser la stratégie LD. S'il y a un cas d'ex aequo entre les périodes, alors affecter l'examen e à la première période rencontrée. Si aucune période n'est valide, alors ajouter l'examen à la liste des examens non programmés.}

4.3.1.4 Méthode de génération utilisant une GRASP basée sur LWD et SD

Afin de combiner les avantages des deux variantes précédentes, nous avons basé celle-ci, désignée par G-LWD-SD et décrite dans l'Algorithme 4.5, sur le même principe que la variante LWD-SD, auquel nous avons introduit l'exploitation adaptative de la liste RCL.

Pour générer une solution initiale de la population, nous affectons $c \times N$ examens (e.g. c = 0.05) à des périodes qui n'ont pas été fréquemment utilisées. Les examens restants sont ensuite, ordonnés avec la stratégie LWD. Puis, dans l'ordre, les *M* (pourcentage empirique) premiers examens sont affectés chacun à la RCL. Celle-ci est alors, ordonnée et mise à jour avec la SD, après l'affectation de chacun de ses examens à une période valide de pénalité minimale. Les examens restants sont ensuite, affectés à la RCL et gérés de la même manière adaptative que les *M* examens précédents.

Par ailleurs, comme dans la variante LWD-SD, une autre manière de maintenir la diversité dans une population générée par cette méthode, est de créer de temps en temps, une solution en utilisant la première variante basée sur la SD, mais gérée de manière adaptative. Cette diversification de la population est activée toutes les dix constructions de solutions réalisables.

Nota: Dans ce chapitre 6, nous étudierons une approche de la Recherche Dispersée basée sur cette méthode, désignée par G-LWDSD-SS.

Algorithme 4. 5 G-LWD-SD: Génération de diversification utilisant une GRASP, LWD et SD

{Pour générer un emploi du temps}

Affecter aléatoirement c×N examens aux périodes qui n'ont pas été fréquemment employées*;*
{ex. c = 0.05}

Ordonner le reste des examens ($N – (c \times N)$) en utilisant la stratégie LWD;

Affecter à la RCL les *M* premiers examens; *{(M étant un pourcentage empirique, ex. 25%}*

En utilisant la stratégie (SD), ordonner et mettre à jour cette liste après l'affectation de chacun de ses examens à une période valide de pénalité minimale;

Affecter le reste des examens à la RCL;

En utilisant la stratégie (SD), ordonner et mettre à jour cette liste après l'affectation de chacun de ses examens à une période valide de pénalité minimale;

{Pour traiter les cas ex aequo avec les stratégies LWD et SD, utiliser la stratégie LD. S'il y a un cas d'ex aequo entre les périodes, alors programmer l'examen à la première période rencontrée. Si aucune période n'est valide, alors ajouter l'examen à la liste des examens non programmés.}

4.3.2 Méthode d'amélioration

La méthode d'amélioration peut être appliquée à une solution initiale, ou bien à une solution générée par la méthode de combinaison. Le but étant d'améliorer la valeur de sa fonction objectif, en utilisant une structure de voisinage adéquate et une stratégie d'exploration du voisinage efficiente. Plusieurs techniques d'amélioration existent dans la littérature. Elles varient de la plus simple comme la recherche locale, aux plus sophistiquées comme la Recherche Tabou, le Recuit Simulé, le voisinage variable etc. Cependant, comme la méthode d'amélioration ne représente qu'une composante dans nos approches, notre choix s'est porté délibérément sur la méthode de descente, afin de réduire le temps d'exécution.

Notre méthode d'amélioration, décrite dans l'Algorithme 4.6, explore itérativement le voisinage jusqu'à la satisfaction d'un critère d'arrêt, tel qu'un nombre d'itérations maximal, un temps imparti ou bien une valeur donnée de la fonction objectif. Le voisinage utilisé est composé de l'ensemble des solutions produites à partir de la solution courante, en appliquant une série de mouvements d'examens, vers des périodes améliorant la fonction objectif.

Notons ici, que pour nos approches, cette méthode n'explore pas les régions sans amélioration immédiate tels que les plateaux.

Algorithme 4. 6 Méthode d'amélioration

Répéter
Pour chaque période *p* d'emploi du temps choisie aléatoirement **faire**
    Pour chaque examen *e* dans la période *p* **faire**
    Calculer la pénalité d'affecter *e* dans chaque autre période valide ;
    Si (la meilleure période a une pénalité inférieure à celle de la période *p*) **alors** affecter *e* à cette période.
    Finsi
    Fait
Fait
    Ordonner les examens non programmés à l'aide de la stratégie *SD ;*
    Essayer d'affecter chaque examen non encore programmé à une période *valide ;*
    {Pour traiter les cas ex aequo dans SD, utiliser la stratégie LD.S'il y a un cas d'ex aequo entre les périodes, alors programmer l'examen à la première période rencontrée.};
Jusqu'à (une amélioration nulle).

4.3.3 Méthode de construction de l'ensemble des solutions de référence

La méthode de construction de l'ensemble des solutions de référence est utilisée, afin de préparer les sous-ensembles nécessaires au processus de combinaison pour la production de nouvelles générations de solutions. Elle démarre par l'affectation des *b1* meilleures solutions de la population *P*, à l'ensemble de référence *R.* Ensuite, pour chaque solution de *P-R*, nous calculons le minimum des distances aux solutions de *R*. la solution avec le minimum maximal est alors ajoutée à *R* et supprimée de *P*, et les distances minimales mises à jour. Notons que la mesure de la diversité, utilisée entre deux solutions *t* et *t'*, peut être définie comme étant le nombre des examens ayant des périodes différentes. Par conséquent, la distance (4.5) entre deux solutions *t* et *t'*, utilisée pour mesurer cette diversité dans l'ensemble de référence *R*, ne se traduit pas sous la forme d'une différence des valeurs de leurs fonctions objectif, mais s'exprime comme suit :

$$\partial(t,t') = \sum_{i=1}^{P}\sum_{j=1}^{E}\left|t_{ij} - t'_{ij}\right| \qquad (4.5)$$

Ainsi, l'ensemble de référence *R* de cardinalité *b* est constitué de deux sous ensembles de la population initiale *P* : l'ensemble *R1* composé des *b1* meilleures solutions et l'ensemble *R2* composé des *b2* solutions, les plus distantes les unes des autres. Voir l'Algorithme 4.7.

Algorithme 4. 7 Construction de l'ensemble de référence *R*

{Pour construire l'ensemble de référencé R composé des b1 meilleures solutions et des b2 solutions les plus diversifiées.}

Affecter les *b1* meilleures solutions de la population *P* à *R* et les supprimer de *P ;*
Pour chaque solution de *P- R* **faire**
Calculer le minimum des distances aux solutions de *R* ;
Fait
Pour i=1 à *b2* **faire**
    Ajouter la solution ayant le minimum maximal à *R* et la supprimer de *P ;*
    Mettre à jour les distances ;
Fait

4.3.4 Méthode de génération des sous-ensembles

Cette méthode permet de créer, sans répétition, différentes collections de sous-ensembles de solutions, à partir de l'ensemble *R.* Ces sous ensembles seront utilisés entièrement, ou partiellement, par la méthode de combinaison, afin de générer de nouvelles solutions, et faire évoluer le processus de recherche.

Variante 1

Dans cette variante décrite par l'Algorithme 4.8, nous avons limité la génération au sous-ensemble 2-SubSets construit avec les paires d'éléments de l'ensemble de référence *R*. Ces paires sont au nombre de *b(b-1)/2*, où *b* représente le cardinal de *R*.

Algorithme 4. 8 Générateur du sous ensemble 2-SubSets

2-SubSets= ∅ ;
Pour i= 1 à b-1 **faire**
Pour j= i+1 à b **faire**
2-SubSets = 2-SubSets ∪{(R[i], R[j])};
Fait
Fait

Variante 2

Dans cette variante, nous générons les éléments des sous-ensembles 2-SubSets et 3-SubSets. Les éléments de ce dernier, au nombre de (b-1)(b-2)/2 , sont générés chacun, à partir d'un élément de 2-SubSets auquel nous rajoutons la meilleure solution de l'ensemble de référence, non présente dans l'élément. Voir l'Algorithme 4.9 de génération des éléments du sous-ensemble 3-SubSets.

Algorithme 4. 9 Générateur du sous ensemble 3-SubSets

```
3-SubSets= ∅ ;
Pour i= 2 à b-1 faire
Pour j= i+1 à b faire
3-SubSets = 3-SubSets ∪{(R[1],  R[i], R[j] )};
Fait
Fait
```

Il est à noter que les autres types de sous-ensembles, proposés par Glover, ne sont pas pris en considération et ce, afin de réduire le temps de traitement des approches proposées.

4.3.5 Méthode de combinaison

Tout comme la méthode de génération et d''amélioration, la méthode de combinaison est contexte dépendante. Ainsi, contrairement aux algorithmes génétiques où la combinaison est aveugle, nous adaptons dans notre cas, quelques heuristiques de coloration de graphe, afin de générer de nouvelles solutions variées, diversifiées et de bonne qualité. Dans ce travail, deux méthodes de combinaison sont proposées :

Variante 1

Cette variante de la méthode de combinaison, décrite par l'Algorithme 4.10, permet de produire une nouvelle solution, à partir de la combinaison de chaque paire du sous-ensemble 2-SubSets. Chaque solution produite ainsi, est ensuite soumise à la méthode d'amélioration avant d'être considérée pour l'adhésion à l'ensemble *R*. La combinaison, l'amélioration et la mise à jour sont itérées, jusqu'à la satisfaction du critère d'arrêt. On dira que ce critère est atteint, lorsqu'on ne peut plus avoir de changement dans l'ensemble *R* et que tous ses éléments sont traités, ou bien lorsqu'on atteint un nombre d'itérations fixé de manière empirique. A ce stade, la population est reconstruite avec la méthode de génération diversification, en conservant les *b1* meilleures solutions de *R*.

Dans notre contexte, chaque paire d'emplois du temps du sous-ensemble 2-SubSets est combinée, afin de produire un nouvel emploi du temps fils. Au départ, tous les examens non programmés des deux emplois du temps pères sont insérés dans la liste des examens non programmés du fils. Nous considérons ensuite, les périodes, dans l'ordre. Tous les examens programmés à une période, dans les deux parents en même temps, sont également, programmés à cette même période, dans l'emploi du temps fils. Les examens qui sont programmés à une période, dans l'un des parents seulement, sont ajoutés à la liste des examens non programmés du fils. Nous essayons

par la suite, d'affecter les examens non programmés d'emploi du temps fils en utilisant la stratégie SD pour son caractère adaptatif dynamique, ou bien en adoptant la procédure suivante:

- Si les deux parents ont des examens non programmés, nous considérons que le problème principal c'est de programmer tous les examens à un nombre limité de périodes. Par conséquent, nous employons ***l'heuristique du plus grand degré pondéré*** (LWD), où la priorité la plus élevée est accordée à l'examen ayant le plus grand nombre de conflits pondérés (chaque conflit est pondéré par le nombre d'étudiants impliqués dans les deux examens en conflit).
- Si l'un des parents, ou bien les deux, n'ont pas d'examens non programmés, nous considérons que le problème principal est de réduire le nombre d'examens ayant des étudiants en commun, programmés à des périodes très rapprochées et nous employons l'heuristique du minimum de conflits avec la période précédente (*the Least Conflicting with Previous Period heuristic (LCPP)).* Cette dernière permet de ranger les examens à affecter dans l'emploi du temps fils, dans un ordre croissant, en fonction du nombre de conflits qu'ils ont avec les examens déjà programmés, à la période précédente du fils (Burke et Newall, 2003; Burke et al.,1998).
- Les examens, ne pouvant être programmés, sont affectés à la liste des examens non programmés pour être reconsidérés à la période suivante.
- Pour traiter les cas ex aequo dans LWD ou LCPP, nous utilisons la stratégie LD. S'il y a un cas d'ex aequo entre les périodes, alors l'examen est programmé à la première période rencontrée. Si aucune période n'est valide, alors l'examen est rajouté à la liste des examens non programmés pour être considéré à nouveau, à la période suivante.

Variante 2

Cette variante combine les éléments des sous-ensembles 2-SubSets et 3-SubSets, avec le même principe que celui de la variante1 en utilisant trois emplois du temps au lieu de deux.

Algorithme 4. 10 Méthode de combinaison des solutions (variante 1)

{Combinaison d'une paire d'emplois du temps du sous-ensemble 2-SubSets.}
{Per : numéro de période.
NbMaxPer : Nombre maximal des périodes du problème d'emploi du temps à *traiter.}*

Pour chaque paire d'emplois du temps du sous-ensemble 2-SubSets **faire**

Insérer dans la liste des examens non programmés de l'emploi du temps fils, tous les examens non programmés des deux emplois du temps pères;

Pour Per=1 à Per =NbMaxPer **faire**

Affecter les examens, des deux emplois du temps pères, programmés en même temps dans la période Per, à cette même période dans l'emploi du temps fils;

Ajouter à la liste des examens non programmés de l'emploi du temps fils, Les examens programmés à cette période, dans l'un des deux emplois du temps pères, seulement;

Essayer de programmer dans l'emploi du temps fils, les examens non encore programmés, selon l'une des heuristiques LWD ou LCPP ;

Fait

{Pour traiter les cas ex aequo de LWD ou LCPP, utiliser la stratégie LD. S'il y a un cas d'ex aequo entre les périodes, alors programmer l'examen à la première période rencontrée. Si aucune période n'est valide, alors ajouter l'examen à la liste des examens non programmés pour être considéré à nouveau à la période suivante.};

Fait

4.3.6 Méthode de mise à jour de l'ensemble de référence

Cette méthode de mise à jour permet à l'ensemble de référence *R*, tant que de nouveaux éléments sont générés par la méthode de combinaison, de faire évoluer l'ensemble de ses éléments, afin d'explorer des nouvelles régions de l'espace de recherche. Si l'on veut éviter l'accélération de la convergence de cet ensemble, nous adoptons une méthode de mise à jour statique, qui n'est effectuée, que si toutes les combinaisons à traiter sont achevées. Deux variantes de mise à jour sont considérées :

Variante 1

Dans cette variante, la mise à jour de l'ensemble de référence *R* est réalisée, en considérant la seule mesure de qualité de la valeur de la fonction objectif. Une nouvelle solution améliorée *t*' remplace la plus mauvaise de *R1,* si son évaluation est meilleure. Nous utiliserons cette variante notée Q, pour analyser l'impact des bonnes solutions sur le processus de recherche.

Variante 2

Dans cette variante de mise à jour 2-Tiers qu'on notera QD, une autre mesure est introduite: la mesure de la diversité, symbolisée par D. Dans la variante QD, une nouvelle solution améliorée *t'*

remplace, également, la plus mauvaise dans *R1*, si son évaluation est meilleure. Si *t'* n'est pas retenue pour sa qualité, elle est considérée à nouveau pour sa diversité, afin de remplacer éventuellement, la plus mauvaise de *R2*.

4.3.7 Algorithme global de la Recherche Dispersée

Les différentes méthodes décrites précédemment, utilisées pour la mise au point de nouvelles approches de la Recherche Dispersée pour l'emploi du temps des examens, sont récapitulées dans l'Algorithme 4.11.

Algorithme 4. 11 Algorithme général de la Recherche Dispersée

---

{PSize : Taille de la population générée par la Méthode de Génération de Diversification.
MaxIter : Nombre maximal des itérations de la Recherche Dispersée.
R : Sous-ensemble de référence formé des sous-ensembles R1 et *R2.*
R1 : Sous-ensemble de R composé des b1 meilleures solutions de la population.
R2 : Sous-ensemble de R composé des b2 solutions les plus diversifiées de la population.
MaxCombin: Nombre maximal des itérations de la Méthode de combinaison.
Iter: Itération courante.
NbCombin : Compteur pour le nombre d'itération de la Méthode de combinaison.}

Créer une population de solutions différentes, de taille *PSize* avec *la Méthode de Génération de Diversification*;
Ordonner les solutions dans *P* en fonction de la valeur de leur fonction objectif ; {la meilleure en premier)}
Pour Iter=1 à MaxIter **faire**
NbCombin =0;
Construire le sous-ensemble de référence *R* avec *la Méthode de Construction de l'ensemble de référence;*
Tanque ((*R* peut être mise à jour) et (NbCombin ≤ MaxCombin)) **faire**
NbCombin = NbCombin +1;
Pour chaque paire de solution de *R* non combinée **faire**
Appliquer la *Méthode de combinaison* pour générer une nouvelle solution *t*;
Appliquer *la Méthode d'amélioration* pour obtenir *t**;
Si (*t** améliore la qualité de *R*) **alors**
Ajouter *t** à *R1* et retirer la plus mauvaise solution de *R1 ;*
Sinon si (t* améliore la diversité de *R*) **alors**
Ajouter *t** à *R2* et retirer la moins diversifiée de *R2* ;
Finsi
Finsi
Fait
Fin Tanque
Générer une nouvelle population, en conservant *les b1* meilleures solutions du *R* courant;
Finpour
Retourner la meilleure solution trouvée.

Chaque méthode est une technique à part, nécessitant d'être élaborée avec les plus grands soins, afin d'explorer efficacement des régions diversifiées et prometteuses de l'espace de recherche, pour atteindre des solutions élites.

4.4 Conclusion

Dans ce chapitre, nous avons décrit les procédures de base que nous avons proposées, afin de construire les premières approches évolutionnistes, basées sur la Recherche Dispersée, pour le problème d'emploi du temps des examens. Ces procédures sont: la génération de diversification, la construction de l'ensemble de référence, la construction des sous ensembles à combiner, la combinaison de ces sous-ensembles, l'amélioration des solutions, la mise à jour de l'ensemble de référence, et enfin la reconstitution de la population.

Dans le chapitre suivant, nous allons décrire les différentes procédures de base des premières approches de la Recherche Locales Guidée, pour le problème d'emploi du temps des examens.

Chapitre 5 : La Recherche Locale Guidée pour le problème d'emploi du temps des examens

5.1 Introduction

Dans ce chapitre, nous présentons également, les principales procédures de base, des premières approches, (à notre connaissance), de la Recherche Locale Guidée pour le problème d'emploi du temps des examens. Nous commençons par présenter quatre variantes de construction de la solution initiale, suivies de la description des différentes étapes de l'algorithme général pour quatre approches de la GLS. Ensuite, pour tenter d'améliorer les résultats finaux, nous introduisons, une stratégie d'aspiration, puis une stratégie de mouvements aléatoires et enfin la combinaison des deux, avant de conclure.

5.2 Procédures de construction de la solution initiale

Quatre variantes différentes de construction de solution sont décrites dans cette section.

5.2.1 Solution initiale aléatoire

Pour cette variante décrite dans l'Algorithme 5.1, nous gênerons la solution initiale d'une manière aléatoire. Chaque examen sélectionné aléatoirement est affecté à une période valide de pénalité minimale. Rappelons que, dans le cas où nous avons le choix entre plusieurs périodes, l'examen est programmé à la première période rencontrée. Si aucune période n'est valide, l'examen est affecté à la période associée à la liste des examens non programmés.

Algorithme 5. 1 Génération d'une solution aléatoire

{Pour générer un emploi du temps t.}
Répéter
    Sélectionner aléatoirement à partir de la liste d'examens un examen *e ;*
    Affecter l'examen *e* à la période valide causant la plus faible pénalité ;
Jusqu'à (l'épuisement des examens) ;
    {S'il y a un cas d'ex aequo entre les périodes, alors affecter l'examen à la première période rencontrée. Si aucune période n'est valide, alors ajouter l'examen à la liste des examens non programmés.}

5.2.2 Solution initiale basée sur Grasp utilisant LWD et SD

Pour la génération d'une solution initiale basée sur une Grasp utilisant LWD et SD, nous présentons trois variantes :

Première variante

La première variante consiste à générer la solution initiale en suivant la même démarche que celle décrite dans la variante G-LWD-SD, du chapitre 4, Algorithme 4.5.

Deuxième variante

La deuxième variante consiste à choisir la meilleure solution d'un ensemble de solutions, générées en suivant la même démarche, que celle décrite pour la variante G-LWD-SD. Voir l'Algorithme 4.5, chapitre 4.

Troisième variante

Cette variante consiste à choisir la meilleure solution, d'un ensemble de solutions, générées avec la méthode décrite dans l'Algorithme 5.2, désignée par G-LWD-SD-2, où les solutions sont générées soit avec SD soit avec LWD.

Pour maintenir la diversité de la population, nous affectons aux périodes qui n'ont pas été fréquemment utilisées, $c \times N$ (e.g. c=5% du nombre total des examens) examens du nombre total des examens. Les $N-(c \times N)$ examens restants sont ensuite, assignés aux périodes, en utilisant la RCL, comme décrit dans l'algorithme 5.2. Cette diversification de la population est activée, toutes les dix constructions de solutions réalisables.

Algorithme 5. 2 G-LWD-SD-2: une variante de génération utilisant une GRASP, LWD et SD

{*Pour générer un emploi du temps}*
Répéter
Affecter aléatoirement à la liste RCL un pourcentage du nombre total des examens *(e.g. 20%).*
Pour chaque examen *e* choisi dans l'ordre, à partir de la RCL ordonnée avec *LWD ou SD*
*(*e.g. 75% LWD suivis de 25% SD*). {Pour traiter les cas ex aequo nous employons la stratégie de LD.}*
Faire
Programmer l'examen *e* à une période valide causant la plus faible pénalité.
Mettre à jour l'ordre de la RCL ordonnée avec SD, pour préparer la prochaine itération ;
Fait
Jusqu'à (l'épuisement des examens).
{Pour traiter les cas ex aequo avec la SD, utiliser la stratégie LD. S'il y a un cas d'ex aequo entre les périodes, alors affecter l'examen à la première période rencontrée. Si aucune période n'est valide, alors ajouter l'examen à la liste des examens non programmés.}

5.3 Approches de la Recherche Locale Guidée pour le problème d'emploi du temps des examens

La résolution du problème d'emploi du temps des examens, doit produire un emploi du temps où, non seulement les examens sont tous programmés, mais aussi, la fonction objectif représentant la somme des pénalités[5] est optimisée. Dans cette section nous décrivons l'algorithme général, pour la construction de quatre approches de la Recherche Locale Guidée pour notre problème:

a) La première se base sur la génération aléatoire de la solution initiale.
b) la deuxième, une première hybridation de la GLS avec GRASP, notée GGLS-b, se base sur la G-LWD-SD.
c) la troisième, notée GGLS-c, repose sur le choix de la meilleure solution d'un ensemble de solutions initiales générées avec G-LWD-SD et non améliorées par une quelconque procédure de recherche locale.
d) la quatrième, une deuxième hybridation de la GLS avec GRASP, notée GGLS-d, choisit la meilleure solution à partir d'un ensemble de solutions initiales, générées avec G-LWD-SD-2, et améliorées par la procédure de recherche locale, décrite dans l'Algorithme 5.2.

La fonction *h,* employée par l'algorithme de recherche local dans ces approches, résulte de l'augmentation de la fonction objectif de départ *g,* par des pénalités. Elle est représentée par l'équation (5.1) suivante :

$$h(t) = g(t) + \lambda \cdot \sum_{i=1}^{K} p_i \cdot I_i(t) \qquad (5.1)$$

Où t représente une solution, λ un paramètre de l'algorithme de la GLS, à fixer de façon empirique, K le nombre de caractéristiques dites aussi attributs, p_i la pénalité pour la caractéristique i (initialisée à 0) et I_i pour indiquer si la solution t présente la caractéristique i ; $I_i(t) = 1$, si t présente la caractéristique i ,0 sinon. Les caractéristiques considérées sont celles non désirées dans une solution. Dans notre cas, la violation de chaque contrainte souhaitable peut être interprétée comme une caractéristique i avec une fonction indicatrice I_i . Le coût d'une caractéristique i, qu'on notera par co_i , est le même que celui présent dans la fonction objectif d'évaluation d'une solution (4.1), décrite dans le chapitre 4.

[5] Les pénalités correspondent aux violations des contraintes souhaitables.

Dans la GLS, lorsque la recherche locale est piégée dans un optimum local, les pénalités de certaines caractéristiques associées à cet optimum local sont augmentées. Ce changement de la fonction objectif conduira la recherche vers d'autres solutions candidates. Nous définissons l'utilité de pénalisation d'une caractéristique *i* présente dans un optimum local *t* *, comme suit:

$$util(t^*, f_i) = I_i(t^*) \cdot \frac{co_i}{1 + p_i} \tag{5.2}$$

Si une caractéristique n'est pas exhibée dans l'optimum local, alors l'utilité de la pénaliser est 0. Plus elle est pénalisée, plus grand devient le facteur $(1+ p_i)$, et donc, plus petite est l'utilité de la pénaliser encore. Dans un optimum local, les caractéristiques ayant une utilité maximale, verront leurs pénalités incrémentées de 1 :

$$p_i = p_i + 1 \tag{5.3}$$

Le paramètre régulateur λ, de la GLS, est employé pour ajuster le poids des valeurs des pénalités, dans la fonction objectif. Cependant, certains problèmes sont moins sensitifs à ce paramètre que d'autres.

Notons aussi, que lorsque l'on prend en considération le coût et la pénalité courante pour déterminer la caractéristique à pénaliser, nous distribuons l'effort de l'exploration dans l'espace de recherche. En effet, les solutions avec des caractéristiques de faibles coûts auront plus d'opportunités dans la recherche. Cependant, les pénalités peuvent conduire à explorer les solutions ayant des caractéristiques de coût plus élevé. Les étapes de notre approche de la GLS, ainsi que sa stratégie de recherche locale sont décrites dans l'Algorithme 5.4 et l'Algorithme 5.3.

Algorithme 5. 3 Une recherche locale de la GLS pour le problème d'emploi du temps des examens

{t : un emploi du temps.}
Répéter
Pour chaque période *p* d'emploi du temps *t* choisie aléatoirement **faire**
Pour chaque examen *e* de la période *p* **faire**
Calculer la pénalité d'affecter *e* dans chaque autre période valide ;
Si (la meilleure période a une pénalité inférieure à celle de la période *p*, en termes de *h)* **alors**
Affecter *e* à cette période,
Finsi
Fait
fait
Ordonner les examens non encore programmés avec la stratégie *SD* ;
Essayer d'affecter chaque examen à une période valide, pour obtenir un nouvel emploi du temps t' ;
$\Delta h = h(t') - h(t)$;
Si $\Delta h < 0$ **alors** t= t';
Finsi
Si ((g(t) < g(t*)) **alors** t*=t;
Finsi
Jusqu' à ($\Delta h \geq 0$);
Retourner t.
{Pour traiter les cas ex aequo avec la SD, utiliser la stratégie LD. S'il y a un cas d'ex aequo entre les périodes, alors affecter l'examen à la première période rencontrée. Si aucune période n'est valide, alors ajouter l'examen à la liste des examens non programmés.}

Algorithme 5. 4 Un algorithme général de la GLS pour le problème d'emploi du temps des examens

Démarrer avec une solution initiale t générée suivant l'une des quatre variantes de génération de solution : a), b), c) ou d) ;
t*=t ;{ t* : la meilleure solution de *g* trouvée jusque là.}
Initialiser toutes les valeurs de pénalité p_i à 0 ;
Répèter
Exécuter la stratégie de recherche locale décrite dans l'Algorithme 5.3 pour la fonction *h* définie dans l'expression (5.1) jusqu'à ce qu'un optimum local *t* soit atteint;
Pour chaque caractéristique *i* exhibée dans *t* **faire**
Calculer $util_i = co_i /(1+ p_i)$
Fait
Pour chaque caractéristique *i* ayant $util_i$ maximal **faire**
$p_i = p_i + 1;$
Fait
Jusqu'à (critère d'arrêt satisfait);
Retourner la meilleure solution trouvée jusqu'ici selon la fonction objectif *g :* t*.

La GLS est une méthode basée pénalité, où les attributs d'utilité maximale, trouvés dans un optimum local, sont pénalisés pour que la recherche puisse s'en échapper. Pour tenter d'améliorer ses performances, nous allons lui rajouter dans une première étape, la stratégie de l'aspiration, puis la stratégie des mouvements aléatoires et enfin, nous terminerons avec la combinaison des deux.

5.4 GLS et la Stratégie d'aspiration

Pour le cas de la Recherche Locale Guidée, le critère d'aspiration, basé sur le meilleur mouvement d'amélioration *(improved best aspiration move)*, est défini comme étant un mouvement qui a généré la meilleure solution obtenue jusque là, et qui n'aurait pas été choisi par la recherche locale utilisant la fonction objectif augmentée. Plus formellement, étant donnée une recherche locale et une fonction à minimiser, un critère d'aspiration pour la GLS est un mouvement d'une solution *s* à une solution *s'* acceptée par la recherche locale, lorsque $g(s') < g(s^*)$ (s* la meilleure solution trouvée jusqu'ici par rapport à g), alors que $h(s') > h(s)$, *h* étant la fonction augmentée. Autrement dit, un mouvement d'aspiration, dans la GLS, est un mouvement devant être normalement, rejeté par la recherche locale opérant sur la fonction augmentée, mais qui est accepté, lorsque le critère d'aspiration est appliqué. On se dit : comme le but c'est d'optimiser *g*, et il n'y a pas de raison d'ignorer une bonne solution, juste parce qu'on utilise une fonction augmentée pour aider à échapper à un optimum local et rendre plus efficace le processus de recherche. Le principe de la recherche locale de la GLS avec ce critère d'aspiration pour le problème d'emploi du temps des examens est décrit dans l'Algorithme 5.5 suivant :

Algorithme 5. 5 Une recherche locale de la GLS avec aspiration pour le problème d'emploi du temps des examens

---

{t : un emploi du temps.
Improv : booléen.}

improv= vrai ;
Répéter
  Pour chaque période *p* d'emploi du temps *t* choisie aléatoirement **faire**
    Pour chaque examen *e* de la période *p* **faire**
      Calculer la pénalité d'affecter *e* dans chaque autre période valide ;
      Si (la meilleure période à une pénalité inférieure à celle de *p*, en termes de h) **alors**
        Affecter *e* a cette période ;
      Finsi
    Fait
  Fait
Ordonner les examens non encore programmés avec la stratégie *SD***;**
Essayer d'affecter chaque examen à une période valide, pour obtenir un nouvel emploi du temps t' ;
Si (h(t') < h(t)) **alors** t= t' ;
  Sinon Si ((g(t') < g(t*)) **alors** t=t' ;
    Sinon improv= faux ;
  Finsi
Finsi
Si ((g(t) < g(t*)) **alors** t*=t;
Finsi
Jusqu'à (!improv);
Retourner t.
{Pour traiter les cas ex aequo avec la SD, utiliser la stratégie LD. S'il y a un cas d'ex aequo entre les périodes, alors affecter l'examen à la première période rencontrée. Si aucune période n'est valide, alors ajouter l'examen à la liste des examens non programmés.}

5.5 GLS et les mouvements aléatoires

La stratégie des mouvements aléatoires, parfois désignée par bruitage, est inspirée des travaux de Selman et al.,(1994). Elle joue un rôle important dans le Recuit Simulé (Amin 1999) et les méthodes de résolution du problème SAT: GSAT et Walksat (Selman et al.,1994]). L'objectif de cette stratégie des mouvements aléatoires permet à la GLS d'éviter d'être piégée dans une région de l'espace de recherche, notamment, pour les petites valeurs de λ. Pour cela, il suffit d'envisager des mouvements aléatoires durant le processus de recherche, en ajoutant à l'algorithme les instructions nécessaires.

Algorithme 5. 6 Une recherche locale de la GLS avec mouvements aléatoires pour le problème d'emploi du temps des examens (a)

{*t : un emploi du temps.*
Prandmove : une probabilité empirique entre 0 et 1.}

randmove = faux ;
improv= vrai ;
répéter
r= valeur aléatoire entre 0 et 1;
Si (r < *Prandmove*) **alors**
Pour chaque période *p* d'emploi du temps *t* choisie aléatoirement **faire**
Affecter chaque examen *e* de *p* à une autre période valide s'il en existe;
Fait
randmov*e* = vrai ;
Sinon
Pour chaque période *p* d'emploi du temps *t* choisie aléatoirement **faire**
Pour chaque examen *e* de la période *p* **faire**
Calculer la pénalité d'affecter *e* à chaque autre période valide;
Si (la meilleure période à une pénalité inférieure à celle de *p*, en termes de *h)* **alors**
Affecter *e* à cette période ;
Finsi
Fait
Fait
randmove=faux ;
Finsi
Ordonner les examens non encore programmés avec la stratégie *SD***;**
Essayer d'affecter chaque examen à une période valide, pour obtenir un nouvel emploi du temps t' ;
Δh= h(t') − h(t) ;
Si ((Δh >=0) **et (!**randmov*e*)) **alors**
Improv=faux ;
Sinon
t= t' ;
Finsi
Si (g(t) < g(t*)) **alors** t*=t ;
Finsi
Jusqu'à (! improv);
Retourner t.
{Pour traiter les cas ex aequo avec la SD, utiliser la stratégie LD. S'il y a un cas d'ex aequo entre les périodes, alors affecter l'examen à la première période rencontrée. Si aucune période n'est valide, alors ajouter l'examen à la liste des examens non programmés.}

Nous pouvons par exemple, choisir d'autoriser la sélection d'un mouvement aléatoire (avec une chance égale de choisir n'importe quel mouvement) du voisinage, avec une probabilité égale à *Prandmove* (fixée de manière empirique), et la sélection du meilleur mouvement du voisinage, avec une probabilité égale à *1-Prandmove*. Le principe de la GLS, avec ce critère de mouvements aléatoires pour le problème d'emploi du temps des examens, est décrit dans les deux variantes suivantes dont les étapes sont présentées dans l'Algorithme 5.6 et l'Algorithme 5.7.

Algorithme 5. 7 Une recherche locale de la GLS avec mouvements aléatoires pour le problème d'emploi du temps des examens (b)

{*t : un emploi du temps.*
m : une probabilité empirique

Répéter
r= une valeur aléatoire entre 0et 1;
Si (r <= *m*) **alors**
Pour chaque période *p* d'emploi du temps *t* choisie aléatoirement **faire**
Affecter chaque examen *e* de *p* à une autre période valide s'il en existe
Fait
Sinon
Pour chaque période *p* d'emploi du temps *t* choisie aléatoirement **faire**
Pour chaque examen *e* de la période *p* **faire**
Calculer la pénalité d'affecter *e* à chaque autre période valide;
Si (la meilleure période à une pénalité inférieure à celle de *p*, en termes de *h)* **alors**
Affecter *e* à cette période ;
Finsi
Fait
Fait
Finsi
Ordonner les examens non encore programmés avec la stratégie *SD***;**
Essayer d'affecter chaque examen à une période valide, pour obtenir un nouvel emploi du temps t' ;
Δh= h(t') − h(t) ;
Si Δh < 0 **alors** t= t' ;
Finsi
Si ((g(t) < g(t*)) **alors** t*=t ;
Finsi
Jusqu'à (Δh >= 0) ;
Retourner t.
{Pour traiter les cas ex aequo avec la SD, utiliser la stratégie LD. S'il y a un cas d'ex aequo entre les périodes, alors affecter l'examen à la première période rencontrée. Si aucune période n'est valide, alors ajouter l'examen à la liste des examens non programmés.}

5.6 GLS et la combinaison de l'aspiration et des mouvements aléatoires

Dans les deux sections précédentes, nous avons proposé deux algorithmes de recherche locale pour la GLS, utilisant respectivement, la stratégie d'aspiration et la stratégie des mouvements aléatoires, afin d'étudier leurs impact à améliorer les performances de l'approche. A présent, nous passons à leur combinaison, en présentant les algorithmes 5.8 et 5.9.

Algorithme 5. 8 Une recherche locale de la GLS avec aspiration et mouvements aléatoires pour le problème d'emploi du temps des examens (a)

{*t : un emploi du temps.*
Prandmove : une probabilité empirique entre 0 et 1.}

randmove = faux ;
improv= vrai ;
Répéter
r= valeur aléatoire entre 0 et 1;
Si (r < =*Prandmove*) **alors**
Pour chaque période *p* d'emploi du temps *t* choisie aléatoirement **faire**
Affecter chaque examen *e* de *p* à une autre période valide s'il en existe;
Fait
randmov*e* = vrai ;

Sinon
Pour chaque période *p* d'emploi du temps *t* choisie aléatoirement **faire**
Pour chaque examen *e* de la période *p* **faire**
Calculer la pénalité d'affecter *e* à chaque autre période valide;
Si (la meilleure période à une pénalité inférieure à celle de *p*, en termes de *h)* **alors**
Affecter *e* à cette période ;
Finsi
Fait
Fait
randmove=faux ;
Finsi
Ordonner les examens non encore programmés avec la stratégie *SD***;**
Essayer d'affecter chaque examen à une période valide, pour obtenir un nouvel emploi du temps t' ;
Δh= h(t') − h(t) ;
Si ((Δh >=0) **et (!**randmov*e*)) **alors**
Si (g(t') < g(t*)) **alors** t=t';
sinon improv= faux ;
Finsi
Sinon
t= t' ;
Finsi
Si (g(t) < g(t*)) **alors** t*=t ;
Finsi
Jusqu'à ((!improv) **ou** (Δh=0)) ;
Retourner t.

{Pour traiter les cas ex aequo avec la SD, utiliser la stratégie LD. S'il y a un cas d'ex aequo entre les périodes, alors affecter l'examen à la première période rencontrée. Si aucune période n'est valide, alors ajouter l'examen à la liste des examens non programmés.}

Ces derniers combinent l'aspiration avec respectivement l'Algorithme 5.6 et l'Algorithme 5.7.

Algorithme 5. 9 Une recherche locale de la GLS avec aspiration et mouvements aléatoires pour le problème d'emploi du temps des examens (b)

{t : un emploi du temps t.
m : une valeur empirique entre deux nombres, a et b.
r : une valeur générée aléatoirement entre deux nombres, a et b.}

improv= vrai ;
Répéter
r= valeur aléatoire entre a et b ;
Si (r < = m) **alors**
Pour chaque période *p* d'emploi du temps t choisie aléatoirement **faire**
Affecter chaque examen *e de p* à une autre période valide s'il en existe
Fait
Sinon
Pour chaque période *p* d'emploi du temps *t* choisie aléatoirement **faire**
Pour chaque examen *e* de la période *p* **Faire**
Calculer la pénalité d'affecter *e* à chaque autre période valide ;
Si (la meilleure période à une pénalité inférieure à celle de *p*, en termes de *h)* **alors**
Affecter e à cette période ;
Finsi
Fait
Fait
Finsi
Ordonner les examens non encore programmés avec la stratégie *SD*;
Essayer d'affecter chaque examen à une période valide, pour obtenir un nouvel emploi du temps t' ;

Δh= h(t') − h(t) ;
Si Δh<0 **alors** t= t' ;
Sinon Si ((g(t') < g(t*)) **alors** t=t';
Sinon improv= faux ;
Finsi
Finsi
Si (g(t) < g(t*)) **alors** t*=t ;
Finsi
Jusqu'à ((!improv) **ou** (Δh=0)) ;
Retourner t.

{Pour traiter les cas ex aequo avec la SD, utiliser la stratégie LD. S'il y a un cas d'ex aequo entre les périodes, alors affecter l'examen à la première période rencontrée. Si aucune période n'est valide, alors ajouter l'examen à la liste des examens non programmés.}

5.7 Conclusion

Dans ce chapitre, nous avons présenté les principales procédures de base, pour de nouvelles approches de la Recherche Locale Guidée (GLS) pour le problème d'emploi du temps des examens. Dans un premier temps, nous avons décrit quatre variantes de construction de la solution initiale. Puis, nous avons présenté les différentes étapes de l'algorithme général de quatre approches de la GLS pour ce problème. Par la suite, pour tenter d'améliorer les résultats finaux, nous avons ajouté à l'algorithme, la stratégie d'aspiration, puis la stratégie des mouvements aléatoires et enfin la combinaison des deux, avant de conclure.

Dans la suite, nous décrivons à travers une étude expérimentale illustrée, nos investigations portant sur la mise au point des premières approches, basées sur la Recherche Dispersée et la Recherche Locale Guidée, pour le problème d'emploi du temps des examens.

Chapitre 6: Résultats expérimentaux et analyse

6.1 Introduction

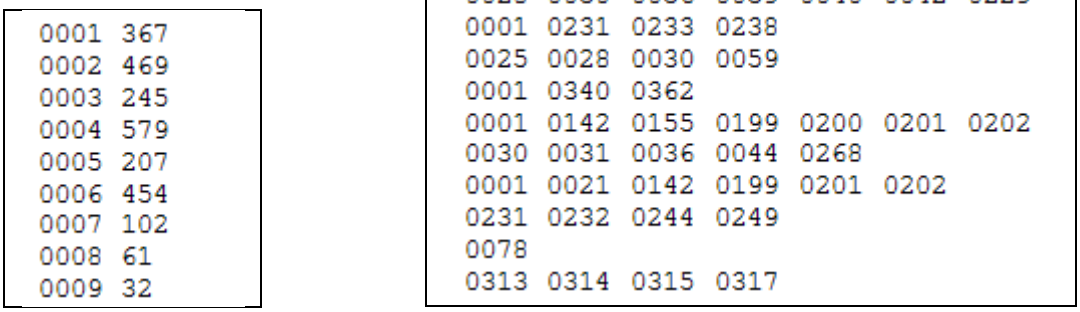
Alors que l'on commençait par s'interroger sur les caractéristiques du problème à résoudre avant d'élaborer une heuristique spécifique, l'avènement des métaheuristiques a en quelque sorte, inversé le processus. En effet, la mise au point de ces dernières, passe par la sélection d'un sous-ensemble de principes qui doivent être appropriés au problème à résoudre. La présentation d'une application des métaheuristiques en général, ne peut donc se faire, qu'en décrivant comment ces principes de base sont mis en œuvre, pour traiter le problème.

Ce chapitre est consacré à la description des différentes étapes suivies dans nos investigations sur la Recherche Dispersée et la Recherche Locale Guidée, pour le problème d'emploi du temps des examens. En premier lieu, nous présentons le cadre expérimental suivi de la description des benchmarks utilisés. Ensuite, nous détaillons les tests effectués et l'analyse des résultats obtenus par les différentes approches proposées.

6.2 Cadre expérimental

Toutes les variantes proposées sont codées en C++ dans l'environnement Windows et exécutés sur un PC avec un Core 2 duo 1.83 GHZ et 4 Go de Ram. Nos tests sont effectués sur les benchmarks de Carter et al.,(1996), décrits dans la Table 6.1, et disponibles via : ftp://ftp.mie.utoronto.ca/pub/carter/tesprob/. Ces benchmark sont utilisés par la quasi- majorité de la communauté du problème d'emploi du temps. Ils sont fournis, sous la forme de deux fichiers pour chaque problème: un fichier étudiants et un fichier examens. Dans le premier, on lit sur chaque ligne le numéro de l'examen suivi de son effectif, alors que dans le second, on lit sur chaque ligne le numéro de l'étudiant, suivi des numéros des examens auxquels il est inscrit. Voir Figure 6.1.

Dans la Table 6.1, le nombre de périodes correspond aux nombre maximal de créneaux horaires alloués à la session d'examens, tandis que la densité représente le ratio entre le nombre d'éléments de valeur égale à 1, sur le nombre total d'éléments dans la matrice des conflits.



Fichier examens

Fichier étudiants

Figure 6. 1 Fichiers des données

Les informations relatives à chaque problème d'examens sont résumées dans la Table 6.1.

Problèmes	Institution	Nombre de périodes	nombre des examens	Nombre des étudiants	Densité de la matrice des conflits
car92	Carleton University, Ottawa	32	543	18419	0.14
car91	Carleton University, Ottawa	35	682	16925	0.13
ear83	Earl Haig Collegiate Institute, Toronto	24	190	1125	0.27
hec92	Ecole des Hautes Etudes Commercials, Montreal	18	81	2823	0.42
kfu93	King Fahd University, Dharan	20	461	5349	0.06
lse91	London School of Economics	18	381	2726	0.06
sta83	St. Andrew's Junior High School, Toronto	13	139	611	0.14
tre92	Trent University, Peterborough, Ontario	23	261	4360	0.18
uta92	Faculty of Arts and Sciences, University of Toronto	35	622	21266	0.13
ute92	Faculty of Engineering, University of Toronto	10	184	2750	0.08
yor83	York Mills Collegiate Institute, Toronto	21	181	941	0.29

Table 6. 1 Benchmarks des problèmes d'examens

6.3 La Recherche Dispersée pour le problème d'emploi du temps des examens

6.3.1 Introduction

Dans cette section, nous présentons nos investigations sur la Recherche Dispersée (SS) pour le problème d'emploi du temps des examens considéré. Notre démarche est organisée en cinq étapes débutant par une étude de l'impact de la qualité des solutions initiales. Ensuite, quatre approches différentes de la SS sont analysées et commentées avant de conclure notre étude.

6.3.2 Impact de la qualité des solutions initiales sur les résultats

La Recherche Dispersée repose sur la phase de construction de la population de solutions initiales. Pour cela, nous avons étudié, avec une approche simple de la Recherche Dispersée pour le problème d'emploi du temps des examens, l'impact, sur les résultats finaux, des heuristiques de construction de solutions, Random, LWD, LD et SD. Voir Algorithme 4.1. Pour nos tests, nous avons utilisé la collection de problèmes de Carter et al. (1996) couvrant la plage des tailles et densités allant de la plus faible à la plus dense, formée par hec, sta, ear et ute.

Avant de décrire notre démarche d'investigation et d'ajustement de paramètres et de méthodes de base pour cette première approche, il est à noter, que dans tous nos tests, nous considérons pour chaque problème, la moyenne des résultats (pénalités, temps etc.) obtenus sur 10 exécutions différentes (différentes semences).

Dans Notre approche de la Recherche Dispersée, les principaux paramètres et méthodes de base à ajuster sont représentés dans la figure 6.2.

PopSize :	Taille de la population.
T :	Taille du tournoi.
R1 :	Taille du sous ensemble des solutions de meilleure qualité.
R2 :	Taille du sous ensemble des solutions les plus diversifiées.
Iter :	Nombre des itérations globales.
Q :	Mise à jour de l'ensemble des solutions de qualité de R.
QD :	Mise à jour de l'ensemble des solutions de qualité et les plus diversifiées de R.
2-SubSets :	Combinaison de sous-ensembles à 2 éléments.
3-SubSets :	Combinaison de sous-ensembles à 3 éléments.
Improv / NImprov	Solutions initiales améliorées/ Solutions initiales non améliorées

Figure 6. 2 Paramètres de la Recherche Dispersée basique

Une forme basique, non raffinée de l'approche de la Recherche Dispersée, génère en générale des solutions où ils restent des examens non programmés. Par conséquent, pour nos tests, nous avons

choisi le nombre d'examens restants (non programmés), comme critère de comparaison des quatre approches basiques utilisant chacune, une des heuristiques : Random, LWD, LD ou SD.

Dans les tests de cette étape, les paramètres et méthodes de base fixés pour l'approche basique de la Recherche Dispersée sont: PopSize =100, T=20, R1 =5, R2 =5, Iter =5, QD, 2-SubSets et NImprov.

Figure 6.3 et Figure 6.4 rapportent respectivement, les totaux des nombres moyens d'examens restants et des pénalités moyennes, obtenus avec chacune des heuristiques Random, LWD, LD et SD.

D'après ces résultats, et le test non paramétrique de Mann-Whitney comparant la distribution de l'approche basée sur la SD avec chacune des autres approches, nous observons que dans l'ensemble, l'heuristique SD est meilleure, comparée aux heuristiques Random, LWD, et LD en termes de pénalité moyenne et du nombre moyen d'examens restants. Cette Performance est certainement due à son mécanisme dynamique adaptatif qui la favorise pour un contexte évolutif comme celui de la Recherche Dispersée. Ainsi, nous pouvons affirmer que la génération d'une population de solutions initiales variées et de bonnes qualités s'impose de manière cruciale. Par conséquent, nous choisissons, dans une première approche de la Recherche Dispersée, la stratégie SD, pour construire les solutions initiales.

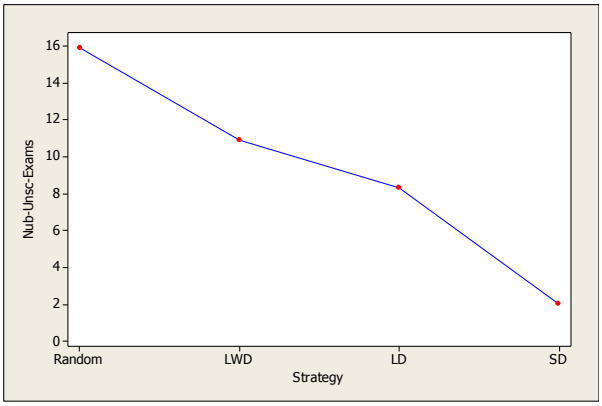


Figure 6. 3 Total des nombres moyens d'examens restants de hec, sta, ear et ute avec Random, LWD, LD et SD

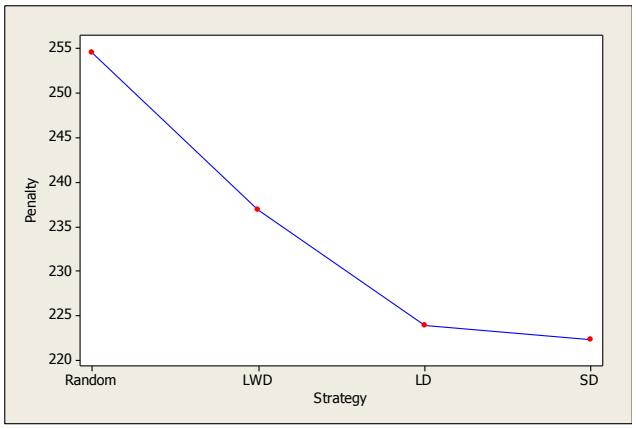


Figure 6. 4 Total des pénalités moyennes de hec, sta, ear et ute avec Random, LWD, LD et SD

6.3.3 SD-SS : Une SS basée sur la SD

Il est connu que pour valider une approche de résolution d'un problème difficile, basée sur une métaheuristique, il est nécessaire d'effectuer une étude empirique pour ajuster l'ensemble de ses paramètres et méthodes de base. Car, ces derniers jouent un rôle crucial et conditionnent son efficience et la qualité de ses résultats. Par conséquent, nous allons tout d'abord, dans le but d'améliorer les solutions précédentes, ajuster la taille de la population, la taille du tournoi, les tailles des sous ensembles de référence R1 et R2, ainsi que le nombre globale des itérations. Ensuite, pour les méthodes de base, nous déterminerons le type de combinaison des sous-ensembles ainsi que le type de mise à jour de l'ensemble de référence. Pour se faire, nous étudierons d'une part, l'impact de la combinaison des sous-ensembles de taille 2 et des sous-ensembles de taille 3, et d'autre part, l'impact de la mise à jour de l'ensemble de référence, de type qualité-diversité QD et de type qualité seule Q.

6.3.3.1 Ajustement des paramètres

Dans cette opération d'ajustement des paramètres de la SD-SS, nous allons procéder aux tests et à la sélection avec un ensemble de valeurs potentielles pour chaque paramètre. Une étude expérimentale intensive, faisant varier un seul paramètre à chaque étape, nous permettra de choisir parmi les valeurs obtenues, celles pour lesquelles nous avons un compromis, entre le temps d'exécution et la qualité des solutions.

Les paramètres considérés sont: la taille de la population, la taille du tournoi, les tailles des sous-ensembles de référence R1 et R2, et enfin le nombre des itérations globales.

a)Taille de la population

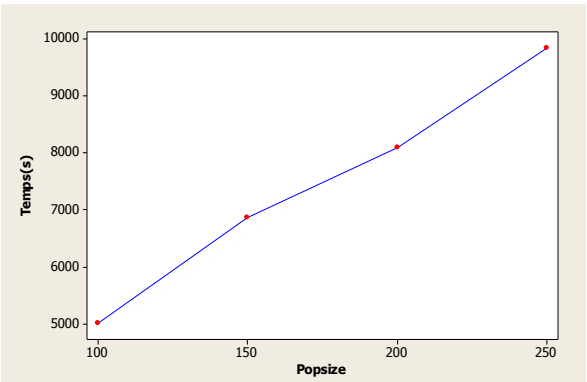
Intuitivement, dans une métaheuristique évolutive, l'opportunité d'explorer de nouvelles régions de l'espace des solutions augmente, lorsque la taille de la population croit. Malheureusement, le temps nécessaire à cette exploration croit aussi, et nous contraint à trouver un compromis entre la qualité des solutions et le temps de recherche.

Dans la section précédente, nous avons vu que l'approche basique de la Recherche Dispersée, basée sur chacune des heuristiques Random, LWD, LD et SD, produit souvent, des solutions non réalisables, où certains examens ne sont pas programmés. Par conséquent, dans une première étape, nous allons étudier, en fonction de la taille de la population initiale, les performances de la SD-SS.

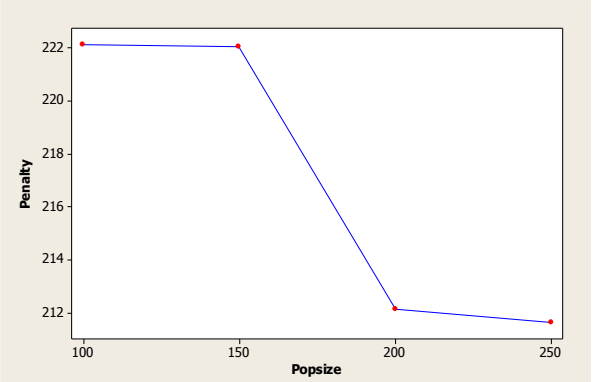
Initialement, les paramètres et méthodes de base fixés sont: T=20, R1=5, R2=5, Iter =5, QD, 2-SubSets et NImprov.

Les résultats obtenus avec les tailles, 100, 150, 200 et 250, sont représentés dans la Figure 6.5. Ils décrivent la variation sur dix exécutions différentes,des totaux des pénalités moyennes et des temps moyens pour les problèmes hec, sta, ear et ute. D'après ces résultats, le total des pénalités moyennes est plus élevé et à peu prés, du même ordre de grandeur, entre 100 et 150. Il décroit ensuite de manière importante, de 150 à 200, puis, plus lentement, de 200 à 250. Le total des temps moyens de traitement par contre, devient croissant.

Une manière intuitive, permettant d'avoir de bons résultats, serait d'effectuer des exécutions différentes pour ne prendre que les meilleures solutions où tous les examens sont affectés à des périodes valides. Néanmoins, notre objectif est de proposer une approche basée sur des stratégies élaborées pour produire des solutions, de sorte que le temps et la fréquence d'avoir des solutions non réalisables sur l'ensemble des exécutions, soient minimisés. Par conséquent, nous allons choisir la taille de 200, pour continuer à ajuster les autres paramètres, afin de satisfaire le compromis qualité-temps.



a) Total des temps moyens



b) Total des pénalités moyennes

Figure 6. 5 Impact de Popsize sur les performances de la SD-SS

b) Taille du tournoi

La taille du tournoi est un paramètre qui a une influence directe sur la diversification de la population initiale. Les paramètres et méthodes de base fixés dans les tests suivants pour son ajustement sont : PopSize =200, R1 =5, R2 =5, Iter =5, QD, 2-SubSets et NImprov. Dix exécutions différentes avec des tournois de taille 10, 20 et 30 sont effectuées pour chaque problème de test. Ensuite, le totale des pénalités, obtenues pour les 4 problèmes avec chaque taille, est reporté dans la Figure 6.6. D'après les résultats, nous observons que la taille 20 produit un total des pénalités pour les 4 problèmes, inférieur à celui des tailles 10 et 30. Par conséquent, nous allons l'adopter pour la suite de nos opérations d'ajustement.

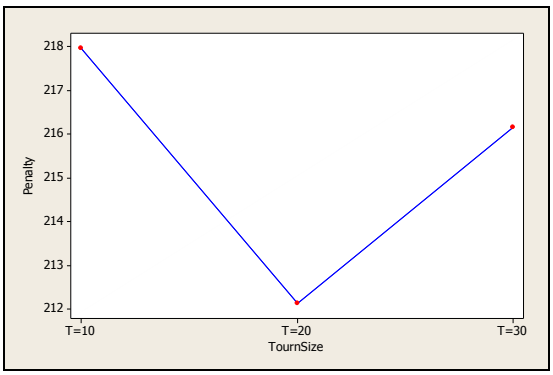


Figure 6. 6 Impact de la taille du tournoi sur le total des pénalités moyennes

c)Taille de R1 et R2

Dans cette phase, nous allons étudier la qualité des solutions finales pour des valeurs différentes de R1 et R2. Les autres paramètres et méthodes de base fixés sont : PopSize =200, T=20, Iter =5, QD, 2-SubSets et NImprov.

Les résultats resitués par les tests effectués, sont présentés dans la Figure 6.7, illustrant le total des moyennes des pénalités pour les 4 problèmes en fonction des tailles de R1 et R2. Notons, qu'avec la configuration (10,10), nous avons les meilleurs résultats, comparés aux autres. Nous pouvons expliquer ceci, par le fait, qu'hormis le facteur de la taille, c'est surtout, l'équilibre entre la qualité et la diversité qui a permis au processus de recherche, d'évoluer vers les régions intéressantes. Par conséquent, nous optons pour R1=10 et R2=10.

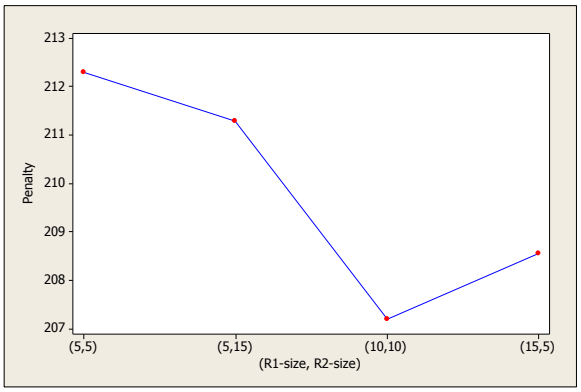


Figure 6. 7 Impact de la taille de R1 et R2 sur le total des pénalités moyennes

d) Nombre des itérations

Dans cette étude de l'impact du nombre des itérations globales nécessaires, sur la qualité des solutions, les paramètres et méthodes de base fixés sont: PopSize=200, R1 =10, R2 =10, QD, 2-SubSets et NImprov. La Figure 6.8 décrit l'influence de la variation de ce paramètre sur les totaux des pénalités et des temps d'exécution, pour les 4 problèmes. Nous constatons que le nombre d'itérations, supérieure à 2, améliore la qualité des solutions au prix d'un temps plus élevé. Par conséquent, pour un compromis qualité-temps, et un temps raisonnable, notamment, pour les problèmes de grande taille, nous nous limitons à deux itérations seulement, et nous abordons dans la suite, le choix des méthodes de base à adopter pour cette approche.

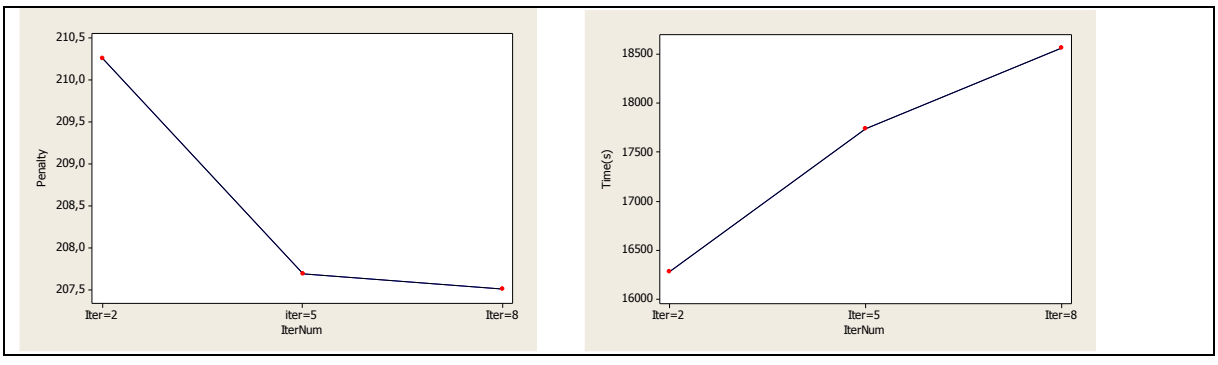


Figure 6. 8 Impact du nombre des itérations sur les performances de la SD-SS

6.3.3.2 Impact de la combinaison et la mise à jour sur la qualité des solutions

Après avoir ajusté les principaux paramètres de notre approche SD-SS, nous allons à présent, étudier l'impact de deux autres composantes de la Recherche Dispersée sur la qualité des résultats:

- *Le type de mise à jour de l'ensemble de référence:* Ici, nous considérons deux manières de mettre à jour l'ensemble de référence. Soit en améliorant seulement, la qualité notée Q, ou bien la qualité et la diversité notées QD (section 4.3.6).
- *Le type de combinaison des sous-ensembles:* Pour la diversification, la Recherche Dispersée permet de générer 4 types de sous-ensembles (section 4.3.5) à combiner, afin de produire de nouvelles solutions. Pour réduire le temps, nous nous limitons, dans notre étude sur l'impact de l'opération de combinaison, à considérer d'abord, la combinaison des éléments de 2-SubSets seulement, ensuite, nous leur rajoutons celle des éléments de 3-SubSets.

Les paramètres fixés pour nos tests sont : PopSize=200, T=20, R1=10, R2=10, Iter=2 et NImprov.

Les résultats de Figure 6.9 et Figure 6.10, représentent pour les quatre problèmes, l'impact du type de combinaison et du type de mise à jour de R, sur le total des pénalités moyennes et le total des temps d'exécution moyens.

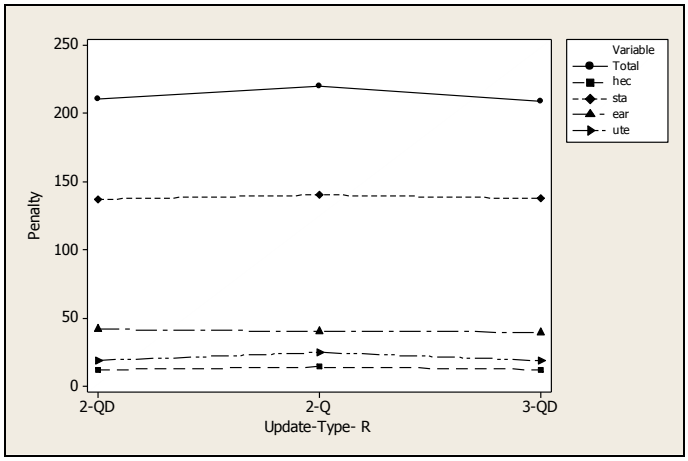
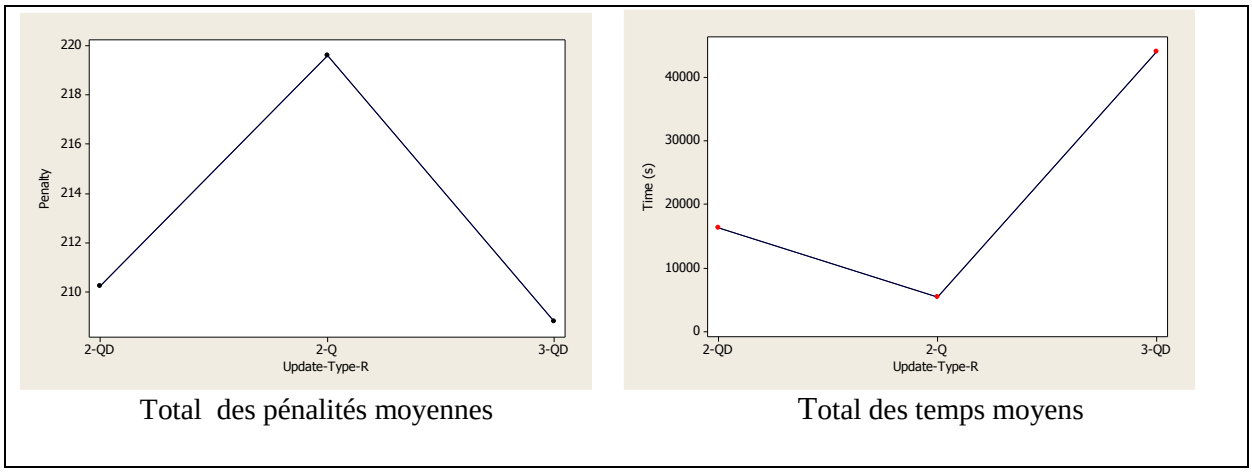


Figure 6. 9 Impact du type de combinaison et mise à jour de R sur les pénalités moyennes et leur total

En analysant ces résultats, nous constatons que les solutions sont meilleures, lorsque nous équilibrons, dans la mise à jour de R, entre la qualité et la diversité. Par ailleurs, bien que les meilleures solutions soient obtenues, lorsque nous combinons les 2-SubSets et les 3-SubSets en

même temps, nous avons constaté que la plus grande amélioration est principalement due, aux 2-SubSets. Le temps évidemment, augmente, lorsque nous faisons contribuer les 3-SubSets.



Total des pénalités moyennes

Total des temps moyens

Figure 6. 10 Impact du type de combinaison et mise à jour de R sur les performances de SD-SS

Dans la suite de notre travail, nous optons pour la combinaison QD et la mise à jour avec les 2-SubSets. Les paramètres et méthodes ajustés donc pour la SD-SS, seront : PopSize =200, T=20, R1 =10, R2 =10, Iter =2, QD, 2-SubSets et NImprov.

Dans un premier temps, nous avons effectué 10 exécutions différentes de la variante SD-SS 200/10/10 sur chacun des problèmes hec,sta,ear et ute. Dans la Table 6.2, nous avons reporté pour chacun des problèmes, la pénalité moyenne (Moy-pénalité), le temps moyen (Moy-temps (s)) exprimé en secondes, la meilleure solution (Meilleure), le nombre moyen d'examens restants (Moy-nb-exam-rest) et l'écart type moyen (Moy-écart-type). Notons que 200/10/10 désignent dans l'ordre, la taille de la population, la taille de R1 et la taille de R2.

Cependant, il est important de noter, que si nous changeons le moindre paramètre parmi ceux ajustés dans cette section, les résultats seraient plus ou moins bons, selon les nouvelles valeurs. Par ailleurs, l'utilisation de semences différentes de celles utilisées pour les tests de la Table 6.2, avec une première variante de la SD-SS, nous conduit également, à des résultats moyens légèrement différents. Pour le cas des meilleures solutions par contre, les différences sont relativement plus importantes, comme le montrent les résultats, d'une deuxième variante de la SD-SS, présentés dans Table 6.3.

Prob	Résultats	SD-SS, 200/10/10
Hec	Moy-pénalité Moy-temps (s) Meilleure Moy-nb-exam-rest. Moy-écart-type	12.19 321.9 11.29 0.2 0.4798
Sta	Moy-pénalité Moy-temps (s) Meilleure Moy-nb-exam-rest. Moy-écart-type	137.11 2657.7 136.09 0 0.8443
ear	Moy-pénalité Moy-temps (s) Meilleure Moy-nb-exam-rest. Moy-écart-type	41.86 6199.2 36.34 1 3.2789
ute	Moy-pénalité Moy-temps (s) Meilleure Moy-nb-exam-rest. Moy-écart-type	19.11 7104.7 18.13 0 0.4349
total	Moy-pénalité	210.27

Table 6. 2 Résultats d'une première variante de la SD-SS

Prob	Résultats de la SD-SS, 200/10/10	
Hec	Meilleure	10.63
Sta	Meilleure	138.12
ear	Meilleure	30.90
ute	Meilleure	22.24
uta	Meilleure	3.52

Table 6. 3 Les meilleures solutions avec une deuxième variante de la SD-SS

Notons aussi, que dans les résultats de la Table 6.2, pour les problèmes hec et ear, il reste respectivement 0.2 et 1 examen sans programmation, sur les 10 exécutions différentes. La question qu'on se pose alors est la suivante: comment peut-on minimiser, sur l'ensemble des 10 exécutions différentes, la fréquence d'avoir des solutions non réalisables?

Evidemment, augmenter la taille de la population initiale ou bien le nombre des itérations serait une réponse plausible à cette question. L'inconvénient, c'est que cette manière de faire nous conduira inévitablement à un temps d'exécution plus élevé.

Une autre solution à ce problème consisterait à améliorer la qualité des solutions initiales, que le processus global de la Recherche Dispersée raffinera. Pour cela, nous avons exploré deux autres variantes de la Recherche Dispersée, l'une basée sur l'hybridation de la stratégie LWD avec la stratégie SD, notée LWDSD-SD et l'autre basée sur l'hybridation de la méthode GRASP avec la Recherche Dispersée, notée G-SD-SS.

6.3.4 LWDSD-SS : une SS basée sur LWD et SD

Pour améliorer davantage nos résultats, nous proposons dans cette étape, une approche notée LWDSD-SS, basée sur l'Algorithme 4.2. Ce dernier hybride les stratégies LWD et SD en donnant la priorité à la première pour prendre en charge les examens ayant les plus grands degrés de conflits pondérés[6] (les plus difficiles à programmer), puis, termine la construction, en traitant le reste des examens avec la stratégie SD. Les paramètres et méthodes de base fixés pour l'étude de cette variante sont: Iter =2, QD, 2-SubSets et NImprov.

Les résultats par cette approche pour quelques tailles différentes de Popsize, R1 et R2 sont présentés dans la Table 6.4. En les analysant, nous remarquons que la LWDSD-SS avec 50/10/7 et 200/10/10, comparée à la SD-SS avec 200/10/10, améliore à priori, le total des pénalités des quatre problèmes hec, sta, ear et ute. Néanmoins, le nombre d'examens restants pour le problème hec est encore non nul.

Pour voir si cette amélioration est effective et non due à un pur hasard, nous avons utilisé le test de Mann-Whitney pour comparer la distribution de la configuration SD-SS 200/10/10 avec celle de LWDSD-SS 50/10/7. Le test de Mann-Whitney permet de résoudre le problème de comparaison de 2 moyennes d'échantillons indépendants, de distributions inconnues[7], afin de tester si les deux échantillons proviennent de la même population.

Pour notre cas, le test de Mann-Whitney révèle que la distribution des solutions de l'approche SD-SS n'est pas significativement différente de celle de l'approche LWDSD-SS, comparées avec un niveau de confiance de 99%, Figure 6.14. Nous pouvons, par conséquent conclure, que l'introduction de la stratégie LWD, dans les premières étapes de construction des solutions initiale, n'est pas suffisante pour l'amélioration effective de la qualité des résultats de la SD-SS.

[6] Le degré de conflits d'un examen est pondéré par le nombre d'étudiants impliqués dans les conflits qu'il a avec d'autres examens.

[7] Si la distribution des échantillons est normale, on utilise le test de Student pour deux d'échantillons indépendants.

Prob	Résultats	LWDSD-SS 50/5/5	LWDSD-SS 50/10/7	LWDSD-SS 200/10/10
hec	Moy-pénalité Moy-temps (s) Meilleure Moy-nb-exam-rest. Moy-écart-type	12.81 11.3 12.30 0.6 0.429	12.06 38.3 11.34 0.1 0.541	11.92 57.6 10.92 0.1 0.656
sta	Moy-pénalité Moy-temps (s) Meileure Moy-nb-exam-rest. Moy-écart-type	139.33 143 137.37 0 1.18	137.68 586.1 134.94 0 1.39	136.82 557.7 135.03 0 1.10
ear	Moy-pénalité Moy-temps (s) Meilleure Moy-nb-exam-rest. Moy-écart-type	41.16 338.5 36.62 1 2.582	39.85 1613.5 36.45 0 2.118	41.09 1366 34.42 0 3.20
ute	Moy-pénalité Moy-temps (s) Meilleure Moy-nb-exam-rest. Moy-écart-type	21.39 352.2 18.54 0.1 2.038	19.27 1669.9 18.14 0 0.871	18.75 1649.9 17.29 0 1.204
Totaux	Moy-pénalité	214.69	208.86	208.58

Table 6. 4 Résultats de l'approche hybride LWSSD-SS sur hec,sta,ear et ute

6.3.5 G-SD-SS : une SS hybridée avec une GRASP basée sur la SD

Dans cette troisième approche notée G-SD-SS, nous avons rajouté à la méthode de construction de l'approche précédente, SD-SS, un caractère adaptatif, comme décrit dans l'Algorithme 4.4. Les paramètres et méthodes de base fixés pour son étude sont : Iter =2, T=20, QD, 2-SubSets et NImprov. Les résultats obtenus pour des tailles différentes de Popsize, R1 et R2, sont reportés dans Table 6.5.

En comparant ces résultats avec ceux de la SD-SS, nous observons avec les configurations 200/101/10 et 50/10/7, une amélioration du total des pénalités moyennes, du temps moyen et du nombre moyen d'examens restants, pour les problèmes de test hec, sta, ear et ute. En outre, il est important de noter, que les pénalités moyennes de la configuration 50/10/7, restent comparables à celles de la première approche, malgré une taille inférieure de la population initiale. Le temps est par contre, sensiblement réduit.

Dans le cas de cette approche, le test de Mann-Whitney, révèle que la distribution des solutions de la configuration G-SD-SS 50/10/7 surpasse significativement celle des solutions de la configuration SD-SS 200/10/10, comparées avec un niveau de confiance de 95%, Figure 6.14. Nous en déduisons donc, que le caractère adaptatif, rajouté dans cette approche, joue un rôle principal dans l'amélioration de ses performances.

Dans la suite, nous proposons une combinaison des deux idées précédentes, dans une quatrième variante hybridée avec une Grasp, basée sur les stratégies LWD et SD, pour tenter d'améliorer davantage, les résultats.

Prob	Résultats	G-SD-SS 50/5/5	**G-SD-SS 50/10/7**	G-SD-SS 200/101/0
Hec	Moy-pénalité	13.56	**10.89**	10.88
	Moy-temps (s)	14.7	**81**	175.16
	Meilleure	12.21	**10.61**	10.20
	Moy-nb-exam-rest.	0	**0**	0
	Moy-écart-type	0.8	**0.147**	0.3743
Sta	Moy-pénalité	138.19	**138.75**	137.38
	Moy-temps (s)	180.2	**748.8**	1376.76
	Meilleure	135.58	**137.15**	134.30
	Moy-nb-exam-rest.	0	**0**	0
	Moy-écart-type	1.8219	**1.4959**	2.1899
Ear	Moy-pénalité	45.28	**34.21**	34.09
	Moy-temps (s)	604.6	**3648**	5006.33
	Meilleure	40.54	**32.34**	32.68
	Moy-nb-exam-rest.	1.4	**0**	0
	Moy-écart-type	3.6786	**1.8032**	1.0775
ute	Moy-pénalité	20.69	**18.23**	18.47
	Moy-temps (s)	466.9	**2393.1**	3106.73
	Meilleure	19.18	**17.57**	17.51
	Moy-nb-exam-rest.	0.2	**0**	0
	Moy-écart-type	1.7325	**0.5057**	0.7494
Totaux	Moy-pénalité	217.72	202.08	200.82

Table 6. 5 Résultats de l'approche adaptative G-SD-SS sur hec,sta,ear et ute

6.3.6 G-LWDSD-SS : une SS hybridée avec une GRASP basée sur LWD et SD

Dans cette quatrième variante notée G-LWDSD-SS et décrite par l'Algorithme 4.5, pour M comprise entre 25% et 30%, nous présentons une combinaison des deux idées proposées dans les deux approches précédentes. Cette combinaison peut être vue également, comme une variante de la méthode Grasp.

Les paramètres et méthodes de base fixés pour cette approche sont : Iter =2, QD, 2-SubSets et NImprov. Tout comme l'étape précédente, nous reportons dans la Table 6.6 les résultats obtenus avec cette approche, pour des tailles différentes de Popsize, R1 et R2.

Nous constatons que les résultats sont en deçà des objectifs escomptés, pour la configuration 50/5/5/. Par contre dans le cas 50/10/7, nous constatons que cette hybridation a une répercussion substantielle sur les performances de l'approche, bien que la taille de la population initiale soit ramenée à 50.

Prob	Résultats	G-LWDSD-SS 50/5/5	G-LWDSD-SS 50/10/7
hec	Moy-pénalité	12.81	10.86
	Moy-temps (s)	11.4s	87.3
	Meilleure	12.30	9.94
	Moy-nb-exam-rest.	0.6	0
	Moy-écart-type	0.429	0.625
sta	Moy-pénalité	139.33	138.7
	Moy-temps (s)	128.8	821.9
	Meilleure	137.37	136.76
	Moy-nb-exam-rest.	0	0
	Moy-écart-type	1.18	1.09
ear	Moy-pénalité	41.16	33.85
	Moy-temps (s)	346.2	3967.6
	Meilleure	36.62	31.72
	Moy-nb-exam-rest.	1	0
	Moy-écart-type	2.582	1.309
ute	Moy-pénalité	21.39	18.70
	Moy-temps (s)	408.1	2011.8
	Meilleure	18.54	18.04
	Moy-nb-exam-rest.	0.1	0
	Moy-écart-type	2.038	0.551
totaux	Moy-pénalité	214.69	202.11

Table 6. 6 Résultats de la G-LWDSD-SS sur hec, sta, ear et ute

Par ailleurs, le test de Mann-Whitney révèle que la distribution des solutions de l'approche G-LWDSD-SS avec la configuration 50/10/7 surpasse significativement celle de l'approche SD-SS, avec la configuration 200/10/10 comparées avec un niveau de confiance de 95%, Figure 6.14. Ceci nous conduit à dire, que cette combinaison, comparée à la SD-SS, améliore effectivement et substantielle les résultats en termes de qualité et de temps.

A présent, nous introduisons la procédure d'amélioration des solutions initiales dans cette variante, afin d'analyser son impact sur les performances. D'après les résultats présentés dans la Table 6.7, nous remarquons une faible amélioration des pénalités, pour une augmentation importante du temps. Par conséquent, pour établir un compromis entre la qualité des solutions et

le temps nécessaire pour les obtenir, nous optons pour la variante avec la configuration 50/10/7, sans amélioration des solutions initiales, dont les résultats sont représentés en caractères gras.

Prob	Résultats	G-LWDSD-SS 50/10/7/, Improv	**G-LWDSD-SS 50/10/7/, NImprov**
hec	Moy-pénalité Moy-temps (s) Meilleure Moy-nb-exam-rest. Moy-écart-type	10.74 157.5 9.89 0 0.467	**10.86** **87.3** **9.94** **0** **0.625**
Sta	Moy-pénalité Moy-temps (s) Meilleure Moy-nb-exam-rest. Moy-écart-type	138.16 1851.1 136.54 0 1.12	**138.7** **821.9** **136.76** **0** **1.09**
ear	Moy-pénalité Moy-temps (s) Meilleure Moy-nb-exam-rest. Moy-écart-type	33.83 8096.7 32.77 0 0.649	**33.85** **3967.6** **31.72** **0** **1.309**
ute	Moy-pénalité Moy-temps (s) Meilleure Moy-nb-exam-rest. Moy-écart-type	18.69 3543.4 18.26 0 2.254	**18.70** **2011.8** **18.04** **0** **0.551**
Totaux	Moy-pénalité	201.42	**202.11**

Table 6. 7 Impact de l'amélioration des solutions initiales dans G-LWDSD-SS

6.3.7 Comparaison des quatre approches

Dans cette étude comparative, la Table 6.8 présente les résultats complets des quatre approches, pour les quatre problèmes hec,sta,ear et ute. Figure 6.11 et Figure 6.12 présentent respectivement, les totaux des Pénalités moyennes, et des temps moyens de traitement des quatre problèmes, avec G-LWDSD-SS, G-SD-SS, LWDSD-SS et SD-SS.

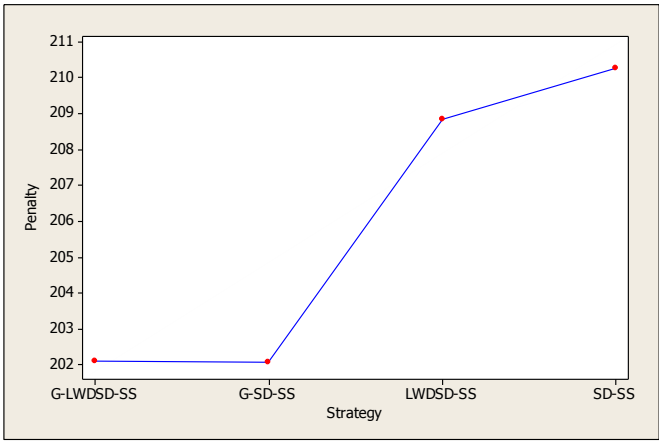


Figure 6. 11 Totaux des pénalités moyennes de hec, sta, ear et ute avec les quatre variantes

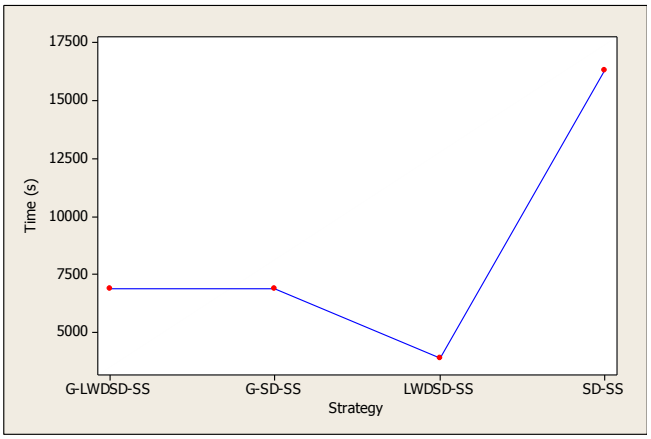


Figure 6. 12 Totaux des temps moyens de hec, sta, ear et ute avec G-LWDSD-SS, G-SD-SS, LWDSD-SS et SD-SS

Prob	Résultats	**G-LWDSD-SS 50/10/ 7**	**G-SD-SS 50/10/7**	**LWDSD-SS 50/10/7**	**SD-SS 200/10/10**
Hec	Moy-pénalité Moy-temps (s) Meilleure Moy-nb-exam-rest. Moy-écart-type	10.86 87.3 9.94 0 0.625	10.89 81 10.61 0 0.1470	12.06 38.3 11.34 0.1 0.541	12.19 321.9 11.29 0.2 0.4798
Sta	Moy-pénalité Moy-temps (s) Meilleure Moy-nb-exam-rest. Moy-écart-type	138.7 821.9 136.76 0 1.09	138.75 748.8 137.15 0 1.4959	137.68 586.1 134.94 0 1.39	137.11 2657.7 136.09 0 0.8443
ear	Moy-pénalité Moy-temps (s) Meilleure Moy-nb-exam-rest. Moy-écart-type	33.85 3967.6 31.72 0 1.309	34.21 3648 32.34 0 1.8032	39.85 1613.5 36.45 0 2.118	41.86 6199.2 36.34 1 3.2789
ute	Moy-pénalité Moy-temps (s) Meilleure Moy-nb-exam-rest. Moy-écart-type	18.70 2011.8 18.04 0 0.551	18.23 2393.1 17.57 0 0.5057	19.27 1669.9 18.14 0 0.871	19.11 7104.7 18.13 0 0.435
Totaux	Moy-pénalité	**202.11**	**202.08**	**208.86**	**210.27**

Table 6. 8 Comparaison des performances des quatre approches

A l'analyse de tous ces résultats, nous observons, en termes de pénalité, que sur l'ensemble des problèmes, les approches G-LWDSD-SS et G-SD-SS sont comparables entre elles et surpassent LWDSD-SS et SD-SS. En termes de temps, nous remarquons que la SD-SS est la plus lente des quatre alors que la LWDSD-SS est la plus rapide. Pour les approches G-LWDSD-SS et G-SD-SS,

les temps sont comparables aussi et se situent entres ceux de LWDSD-SS et SD-SS. Par ailleurs, les pénalités des quatre approches, pour chacun des problèmes, sont soit légèrement différentes, ou bien carrément différentes. Afin de pouvoir conclure que les idées proposées, apportent ou non une réelle amélioration, nous présentons dans la Figure 6.13, les Box plots des distributions des quatre approches sur les quatre problèmes.

A l'analyse de ces boites nous observons d'une part, la ressemblance des distributions des approches LWDSD-SS et SD-SS, et celle des approches G-LWDSD-SS et G-SD-SS et d'autre part, la différence des distributions des approches SD-SS et G-SD-SS et celles des approches SD-SS et G-LWDSD-SS.

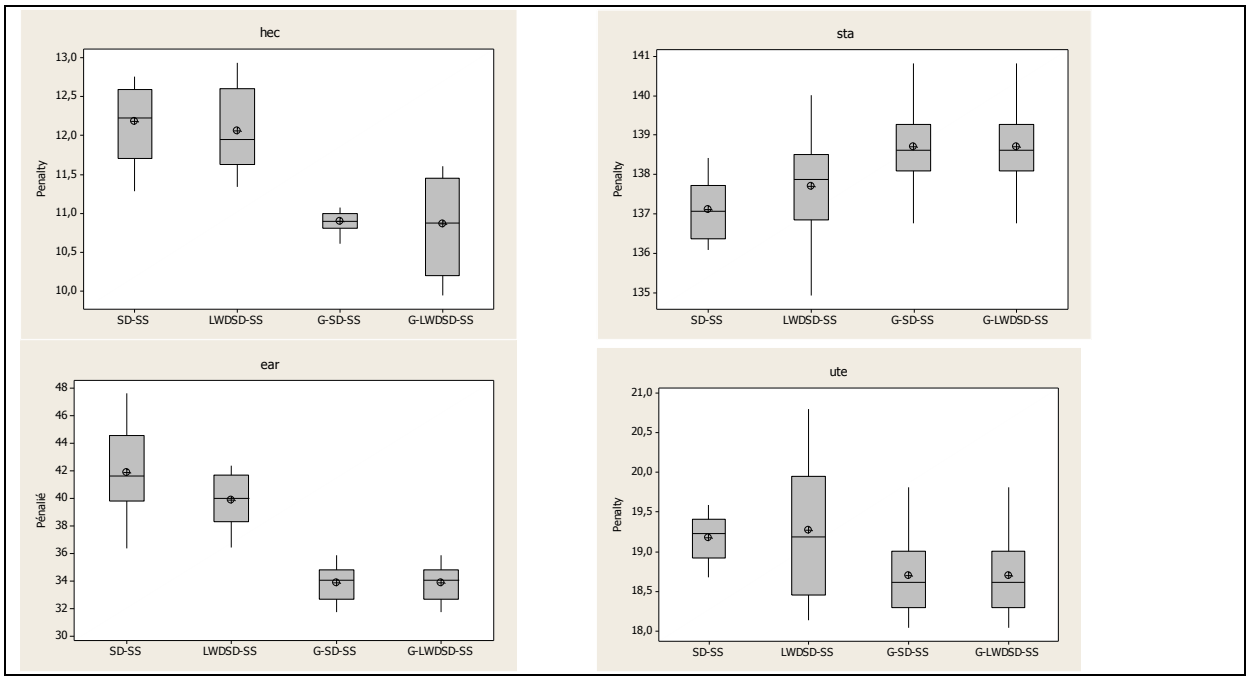


Figure 6. 13 Box plots des distributions des quatre approches pour hec, sta, ear et ute

Pour confirmer nos observations visuelles, nous utilisons le test de Mann-Whitney afin de comparer les distributions de nos approches, deux à deux. Le test de Mann-Whitney est un test non paramétrique, utilisé pour déterminer si deux échantillons d'observations indépendants proviennent d'une même distribution. L'hypothèse nulle H0 est qu'il n'y a pas de différence significative entre les deux échantillons, c'est-à-dire qu'ils proviennent de la même population et par conséquent, leurs distributions de probabilité sont égales.

D'après les résultats obtenus, présentés à la Figure 6.14, nous constatons que la distribution des solutions de l'approche SD-SS n'est pas significativement différente de celle de l'approche LWDSD-SS, comparées avec un niveau de confiance de 99%. Ceci prouve que l'hybridation des

heuristiques seule, n'apporte pas une amélioration significative. De même, la distribution de l'approche G-LWDSD-SS n'est pas significativement différente de celle de l'approche G-SD-SS comparées avec un niveau de confiance de 99%. Ceci est du sans doute à leur commun caractère adaptatif.

A contrario, on note pour les couples des approches (SD-SS, G-SD-SS) et (SD-SS, G-LWDSD-SS), que les solutions obtenues suivent deux distributions significativement différentes avec un niveau de confiance de 95%, en faveur des approches basées sur l'hybridation avec Grasp. Ceci peut être expliqué, par le fait, que le caractère adaptatif ne soit pas commun aux deux approches de chaque couple. Par conséquent, nous pouvons dire que les améliorations apportées par la G-SD-SS et la G-LWDSD-SS, par rapport à la SD-SS, marquant de façon claire la supériorité du facteur adaptatif de la Grasp sur la seule hybridation des heuristiques LWD et SD, sont effectives et non dues à un pur hasard.

En conclusion de cette étude, nous dirons, que toutes nos approches peuvent être utilisées pour traiter le problème d'emploi du temps des examens. Néanmoins, étant donnés les totaux des pénalités pour l'ensemble des problèmes de test, les approches G-LWDSD-SS et G-SD-SS sont bien les meilleures.

Prob	**p-value 99%** **LWDSD-SS / SD-SS** **50/10/7 200/10/10**
hec	0.6497
Sta	0.2121
ear	0.1617
ute	0.9698

Prob	**2*p-value 95%** **G-SD-SS / SD-SS** **50/10/7 200/10/10**
hec	0.0002
Sta	0.0036
ear	0.0002
ute	0.041

Prob	**2*p-value 95%** **G-LWDSD-SS / SD-SS** **50/10/7 200/10/10**
hec	0.0006
Sta	0.0036
ear	0.0002
ute	0.0413

Prob	**p-value 99%** **G-LWDSD-SS / G-SD-SS** **50/10/7 50/10/7**
hec	**0,9698**
Sta	1.000
ear	1.000
ute	1.000

Figure 6.14 Les P-valeurs des distributions des quatre approches comparées deux à deux

Dans la suite, nous présentons les résultats des approches G-LWDSD-SS et G-SD-SS, pour les benchmarks de Carter et al.,(1996).

6.3.8 Résultats de G-LWDSD-SS et G-SD-SS sur les benchmark de Carter et al.

Dans la Table 6.9 et la Table 6.10 nous présentons les résultats restitués par les approches : G-LWDSD-SS et G-SD-SS pour les benchmarks de Carter et al.,(1996).

Afin de réduire le temps de traitement, nous avons effectué nos tests en choisissant pour nos paramètres, Popsize/R1 /R2, les valeurs 50/10/7 dans le cas des problèmes de la Table 6.9, et les valeurs 50/5/5 dans le cas des problèmes de la Table 6.10.

Nous constatons à travers ces résultats, comme nous l'attendions, que les valeurs des pénalités moyennes, obtenues par la G-LWDSD-SS et la G-SD-SS, sont comparables pour tous les problèmes. Les temps moyens et les meilleures solutions par contre, sont pour certains problèmes, meilleurs avec la G-LWDSD-SS et pour d'autres, meilleurs avec la G-SD-SS.

Prob	**Résultats**	**G-LWDSD-SS 50 /10/ 7**	**G-SD-SS 50/10/7**
hec	Moy-pénalité Moy-temps (s) Meilleure Moy-nb-exam-rest. Moy-écart-type	10.86 87.3 9.94 0 0.625	10.89 81 10.61 0 0.1470
sta	Moy-pénalité Moy-temps (s) Meilleure Moy-nb-exam-rest. Moy-écart-type	138.7 821.9 136.76 0 1.09	138.75 748.8 137.15 0 1.4959
ear	Moy-pénalité Moy-temps (s) Meilleure Moy-nb-exam-rest. Moy-écart-type	33.85 3967.6 31.72 0 1.309	34.21 3648 32.34 0 1.8032
ute	Moy-pénalité Moy-temps (s) Meilleure Moy-nb-exam-rest. Moy-écart-type	18.70 2011.8 18.04 0 0.551	18.23 2393.1 17.57 0 0.5057

Table 6. 9 Résultats de G-LWDSD-SS et G-SD-SS sur des benchmarks de Carter et al.

Prob	Résultats	G-LWDSD-SS 50/5/5	G-SD-SS 50/5/5
yor	Moy-pénalité Moy-temps (s) Meilleure Moy-nb-exam-rest. Moy-écart-type	38.10 371.4 34.22 0 2.586	38.08 184.7 35.65 0 1.6915
tre	Moy-pénalité Moy-temps (s) Meilleure Moy-nb-exam-rest. Moy-écart-type	9.03 1035.4 8.86 0 0,1201	9.23 1052.7 8.58 0 0.3658
kfu	Moy-pénalité Moy-temps (s) Meilleure Moy-nb-exam-rest. Moy-écart-type	13.65 9078 13.10 0 0.3838	13.15 11170.2 12.06 0 0.8170
les	Moy-pénalité Moy-temps (s) Meilleure Moy-nb-exam-rest. Moy-écart-type	11.66 3660.6 11.02 0 0.3817	11.75 4150.4 11.01 0 0.6411
uta	Moy-pénalité Moy-temps (s) Meilleure Moy-nb-exam-rest. Moy-écart-type	3.70 9129.28 3.56 0 0.0902	3.93 16976.9 3.81 0 0.069
car91	Moy-pénalité Moy-temps (s) Meilleure Moy-nb-exam-rest. Moy-écart-type	5.34 18147 5.30511 0 0.0246	5.24 38306.7 5.06848 0 0.2341
car92	Moy-pénalité Moy-temps (s) Meilleure Moy-nb-exam-rest. Moy-écart-type	4.56 11706.3 4.50 0 0.04561	4.55 17624.3 4.48 0 0.0695

Table 6. 10 Résultats de G-LWDSD-SS et G-SD-SS sur des benchmarks de Carter et al.,(2)

6.3.9 Comparaisons des résultats de G-LWDSD-SS et G-SD-SS avec l'état de l'art.

Il existe dans la littérature plusieurs approches différentes proposées pour le problème d'emploi du temps des examens. Certaines sont basées sur une méthode de recherche locale, d'autres sont basées population. Ceci explique en partie, le fait que dans la plupart des travaux, le temps n'est pas rapporté. Par ailleurs, la majorité des papiers s'accordent à dire, que pour un problème

d'emploi du temps de grande taille, un temps de quelques heures, voir même quelques jours, est tout à fait acceptable, étant donné le caractère différé de l'opération de la production d'un emploi du temps, dans une institution.

Dans la Table 6.11, nous présentons les pénalités minimales de la G-LWDSD-SS et la G-SD-SS sur les benchmark de Carter et al., (1996), comparés à ceux de l'état de l'art. Rappelons aussi, qu'étant donnée la taille des problèmes, ces résultats sont obtenues avec 10 exécutions différentes pour les problèmes de test : hec, sta,ear ,ute,yor,tre, 7 pour le problème uta et 5 pour les problèmes car91 et car92.

A l'analyse de nos pénalités minimales comparées avec celles de l'état de l'art, où les meilleures de toutes sont représentées en caractères gras, nous pouvons dire, que celles, obtenues par la G-LWDSD-SS et la G-SD-SS sont comparables pour certains problèmes et compétitives, pour d'autres. En outre, nous remarquons par ailleurs aussi, qu'aucune des méthodes de l'état de l'art, n'affiche une suprématie pour tous les problèmes à la fois. Nous en concluons, que les premières approches de la Recherche Dispersée (à notre connaissance), que nous avons proposées pour le problème d'emploi du temps des examens, apportent une contribution effective pour le résoudre, et offrent au décideur la possibilité de trouver un compromis entre ses objectifs, pour un choix pertinent.

Datasets	**car91**	**car92**	**ear83**	**hec92**	**kfu93**	**lse91**	**sta83**	**tre92**	**uta92**	**ute92**	**yor83**
G-LWDSD-SS	5.30	4.50	31.72	9.94	13.10	11.02	136.76	8.86	3.56	**18.04**	**34.22**
G-SD-SS	5.07	4.48	32.34	10.61	12.06	11.01	137.15	8.58	3.81	**17.57**	35.65
(Abdullah et al., 2007a)	5.21	4.36	34.87	10.28	13.46	10.24	159.20	8.13	3.63	**24.21**	36.11
(Abdullah et Burke, 2006)	4.8	4.1	36.00	10.80	15.2	11.9	159.00	8.5	3.60	26.00	36.2
(Asmuni et al., 2005)	5.29	4.56	37.02	11.78	15.81	12.09	160.42	8.67	3.57	27.78	40.66
(Burke et al., 2004)	4.8	4.2	35.4	10.8	13.7	10.4	159.10	8.3	3.40	25.70	36.7
(Burke et al., 2005)	-	-	45.60	-	-	-	158.20	-	4.52	35.40	-
(Burke et al., 2007)	5.36	4.53	37.92	12.25	15.2	11.33	158.19	8.92	**2.88**	28.01	41.37
(Burke et Newell, 2003)	4.65	4.1	37.05	11.54	13.9	10.82	168.73	8.35	3.20	25.83	37.28
(Burke et 2006)	**4.42**	**3.75**	32.76	10.15	**12.96**	9.83	157.03	7.75	3.06	24.82	**34.84**
(Caramia et al., 2001)	6.6	6	**29.30**	**9.20**	13.8	**9.6**	158.20	9.4	3.50	24.40	36.2
(Carter et al., 1996)	7.1	6.2	36.40	10.80	14.0	10.5.	161.50	9.6	3.50	25.80	41.7
(Casey et Thompson, 2002)	5.4	4.4	34.80	10.80	14.1	14.7	**134.90**	8.7	-	25.40	37.5
(Côte et al., 2005)	5.4	4.2	34.20	10.40	14.3	11.3	157.00	8.6	3.50	25.30	36.4
(DiGaspero et Schaerf, 2001)	6.2	5.2	45.70	12.40	18	15.5	160.80	10.0	4.20	29.00	41.0
(Eley, 2006)	5.7	4.8	36.80	11.30	15.0	12.1	157.20	8.8	3.80	27.70	39.6
(Kendall et Mohd Hussin, 2005)	4.93	4.40	38.14	10.74	15.55	12.35	157.38	8.35	3.77	27.12	37.99
(Merlot et al., 2003)	5.1	4.3	35.10	10.60	13.5	10.5	157.30	8.4	3.50	25.10	37.4
Petrovic et al., (2003) [30]	4.50	3.93	33.75	10.83	13.82	10.63	165.27	**7.92**	3.14	25.33	36.53
(white et al., 2004)	5.73	4.63	45.80	12.90	17.1	14.7	158.00	8.94	4.44	29.00	42.3
(Yang et Petrovic, 2005)	4.5	3.93	33.70	10.83	13.82	10.35	151.15	**7.92**	3.14	25.39	36.35

Table 6. 11 Comparaison des pénalités minimales de G-LWDSD-SS et G-SD-SS avec l'état de l'art

6.4 La Recherche Locale Guidée pour le problème d'emploi du temps des examens

6.4.1 Introduction

Dans cette section consacrée à l'investigation de la Recherche Locale Guidée (GLS) pour le problème d'emploi du temps des exams, nous avons effectué en premier lieu, une étude de quatre approches de la GLS.

a) La première basée sur une génération aléatoire de la solution initiale.
b) La deuxième résultante d'une première hybridation avec GRASP, notée GGLS-b, et basée sur la G-LWD-SD, décrite par l'Algorithme 4.5.
c) la troisième, notée GGLS-c, reposant sur le choix de la meilleure solution à partir d'un ensemble de solutions initiales, générées avec la G-LWD-SD et non améliorées par une quelconque procédure de recherche locale.
d) la quatrième résultante d'une deuxième hybridation avec GRASP, notée GGLS-d, choisissant la meilleure solution à partir d'un ensemble de solutions initiales, générées avec la G-LWD-SD-2 (voir l'Algorithme 5.2) et améliorées par la procédure de recherche locale, décrite dans l'Algorithme 4.6.

A l'étape suivante, pour tenter d'améliorer davantage les performances de la meilleure des approches, nous avons étudié l'impact de l'introduction de la stratégie d'aspiration, puis celle des mouvements aléatoires, et enfin la combinaison des deux.

Enfin nos meilleures solutions, avec ces nouvelles approches, sont comparées avec l'état de l'art, avant de conclure.

6.4.2 Approches de la GLS pour le problème d'emploi du temps des examens

La première question qui nous est venue à l'esprit, en construisant une approche de la Recherche Locale Guidée pour le problème d'emploi du temps des examens; concerne la nature de la solution initiale. Doit-on la choisir aléatoire ou bien, faut-il la générer avec une heuristique appropriée ?

Pour répondre à cette question, nous avons effectué une série de tests que nous décrivons dans la suite.

6.4.2.1 Etude de la variante basée sur une solution initiale aléatoire

Dans cette première étape nous avons proposé une approche basique de la Recherche Locale Guidée, où le processus de recherche démarre avec une solution aléatoire générée à l'aide de l'Algorithme 4.1.

Expérimentation

Pour estimer les résultats en termes de temps et de qualité, afin d'affronter la résolution des problèmes de grande taille, nous nous somme expressément limité, dans un premier temps, aux problèmes hec et sta seulement. Nous avons dans cette première série de tests, effectué 10 exécutions différentes (différentes semences) pour chaque problème, avec un nombre d'itérations égal à 100, 1000 et 5000.

Pour étudier l'impact du paramètre λ, nous avons choisi comme dans (Mills 2003), d'effectuer nos investigations avec la plage des valeurs de l'ensemble suivant :

$\lambda \in \{0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1,2,3,4,5,6,7,8,9,10,20,30,40,50,60,70,80,90,100\}$.

Résultats et analyse

Dans la Figure 6.15, deux graphes rapportent en fonction de λ, les résultats obtenus pour chaque problème. Ils indiquent pour les trois nombres d'itérations 100,1000 et 5000 respectivement, le nombre moyen d'examens restants (non programmés) et le temps moyen d'exécution exprimé en secondes.

Les résultats obtenus montrent que pour chaque problème, le nombre d'examens, non programmés dans la solution finale, décroit lentement, avec l'augmentation du nombre d'itérations.

Comme nous l'avons remarqué avec les approches précédentes, une génération de solution initiale aléatoire produit dans ce cas également, généralement des solutions non réalisables. En plus, l'amélioration par rapport à la solution initiale est obtenue après un temps d'exécution important. Ceci nous conduit à déduire, que la Recherche Locale Guidée, démarrant avec une solution initiale aléatoire, nécessite un temps important pour atteindre des solutions de bonne qualité. Notons ici, que des travaux antérieurs, suggèrent pour le problème de coloration de graph, de démarrer une recherche locale avec une solution ad hoc plutôt qu'aléatoire (Hertz et al, 1987; Johnson et al. 1991). En effet, ils ont observé expérimentalement, qu'une solution initiale, de

bonne qualité, permet de consacrer le temps plus efficacement, à mieux explorer l'espace de recherche.

Par ailleurs, des travaux exploitant la Recherche Locale Guidée pour des problèmes d'affectation quadratique comparables au problème d'emploi du temps (paysage, rugosité), rapportent qu'elle nécessite, un temps variant de quelques heures à quelques jours (Mills 2003).

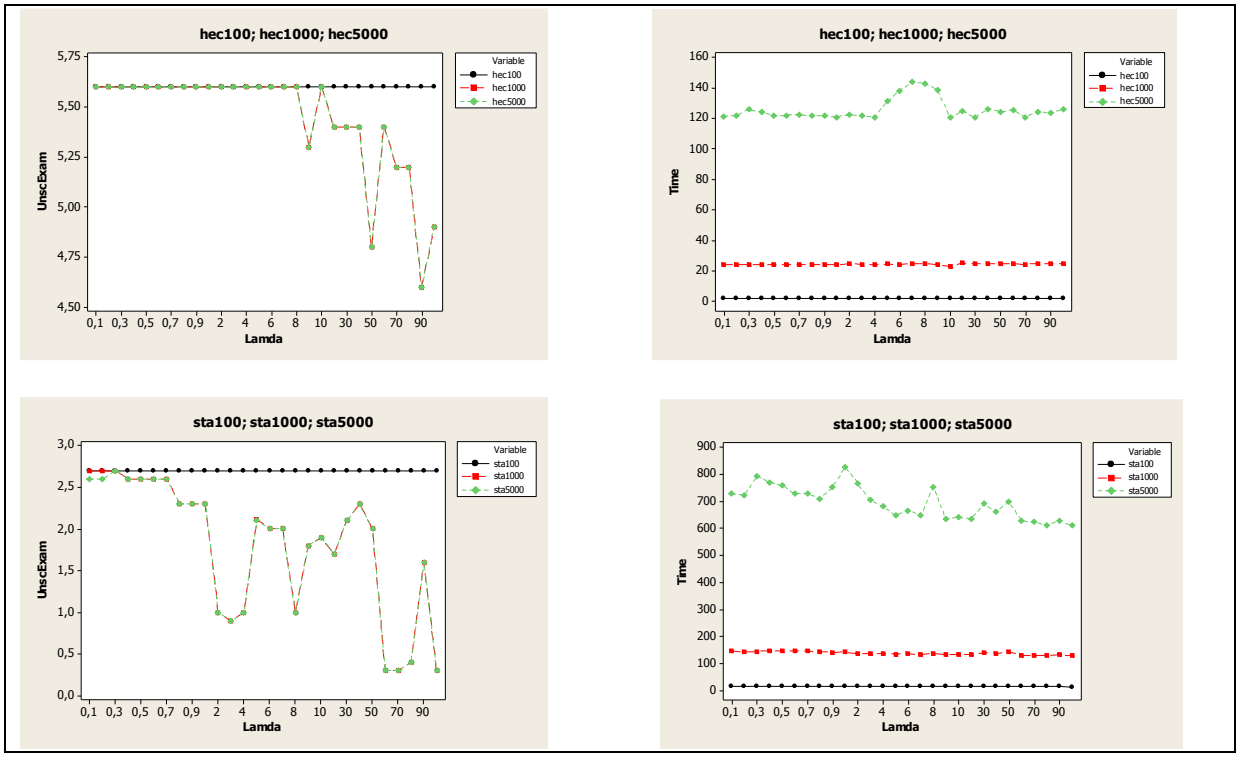


Figure 6. 15 Performances de la GLS pour hec et sta avec une solution initiale aléatoire

Ainsi, nous proposons dans la suite, de générer la solution initiale à l'aide d'une hybridation de la GLS avec une Grasp basée sur les heuristiques LWD et SD. Ceci accordera à la GLS, le plus d'opportunités de démarrer avec une solution ayant le moins d'examens non programmés, afin de consacrer le plus de temps, à l'opération de raffinage. Car avec une solution aléatoire, nous somme en face de deux problèmes : celui de trouver une période valide pour chaque examen, et celui de minimiser la valeur de la fonction objectif, représentant le coût (pénalité) associé à la violation des contraintes souhaitables. Trois variantes de cette hybridation sont donc, étudiées dans la suite.

6.4.2.2 Etude de l'approche GGLS -b

Pour cette approche notée GGLS-b, la solution initiale est générée selon la procédure G-LWD-SD décrite par l'Algorithme 4.5, avec M = 25%. Cette dernière exploite de manière adaptative une hybridation des heuristiques de coloration de graphe : LWD et SD.

Expérimentation

Nous avons effectué 10 exécutions différentes, sur quatre problèmes de Carter et al.,(1996), couvrant un ensemble de caractéristiques concernant la taille du problème et la densité de conflit: hec, sta, ear et ute. La taille du tournoi est fixée à 20. Le nombre des itérations est égal à 1000 pour hec et sta et 100 pour ear et ute. Enfin, le paramètre λ varie dans l'ensemble:
{0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1,2,3,4,5,6,7,8,9,10,20,30,40,50,60,70,80,90,100}.

Résultats et analyse

Dans la Figure 6.16, deux graphes rapportent en fonction de λ, les résultats de la GGLS-b pour chacun des problèmes de test considérés. Ils représentent respectivement, le nombre moyen d'examens restants (non programmés) et le temps moyen d'exécution exprimé en secondes.

Nous observons dans ce cas et après analyse, que les résultats obtenus pour les problèmes hec et sta sont meilleurs que ceux obtenus par la Recherche Locale Guidée, démarrant avec une solution initiale aléatoire. Néanmoins, le nombre des examens non programmés reste encore, non nul, même s'il est cette fois-ci, moindre.

Une solution à ce problème consisterait à augmenter le nombre des itérations. Cependant, comme le processus de la GGLS-b est très lent notamment, pour les problèmes de grande taille, dans la suite, nous allons choisir comme solution initiale, la meilleure d'un ensemble généré par l'Algorithme 4.5 avec M = 25%. Ainsi, la majeure partie du temps sera consacré à l'opération de raffinage.

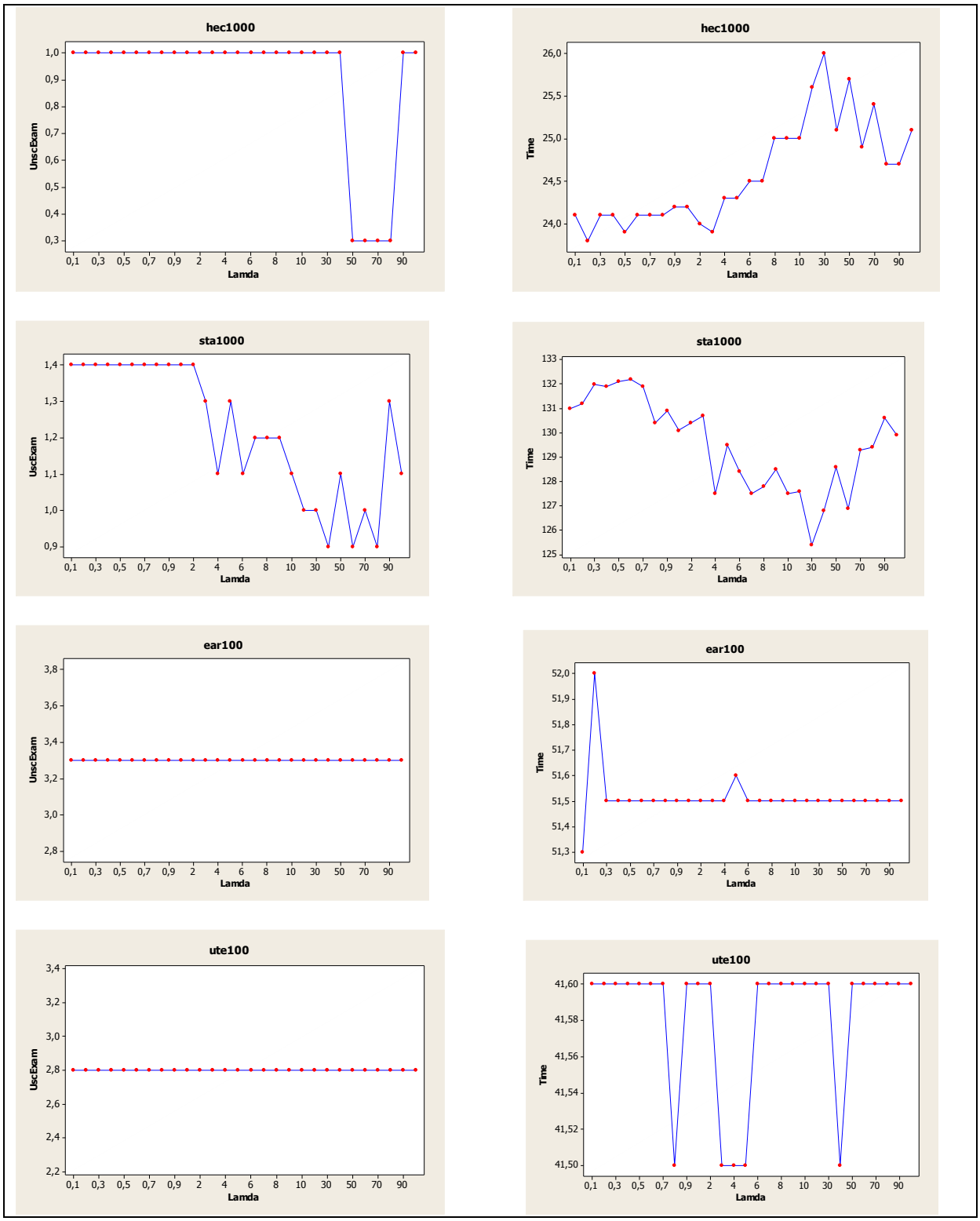


Figure 6. 16 Performances de la GGLS-b pour hec, sta, ear et ute

6.4.2.3 Etude de l'approche GGLS -c

Dans cette approche de la GLS notée GGLS-c, nous démarrons avec la meilleure des solutions d'un ensemble généré à l'aide de la procédure G-LWD-SD précédemment décrite. Avant de procéder aux tests permettant d'analyser le processus de recherche de la GGLS-c en fonction de λ, nous avons effectué des tests préalables pour fixer la taille de l'ensemble initial.

a- Tests préalables Pour fixer la taille de l'ensemble initial

Pour fixer la taille de l'ensemble de solutions initiales, nous avons effectué pour chaque problème de test 10 exécutions différentes, pour des valeurs de λ variant dans l'ensemble : { 0.1,1,0,100}, et des tailles de l'ensemble de solutions initiales égales à 100,200 et 220. Le nombre des itérations étant de 1000 pour hec et sta et 100 pour ear et ute.

Dans la Figure 6.17, pour chaque problème, un graphe rapporte en fonction de λ, la pénalité moyenne, pour les tailles 100, 200 et 220.

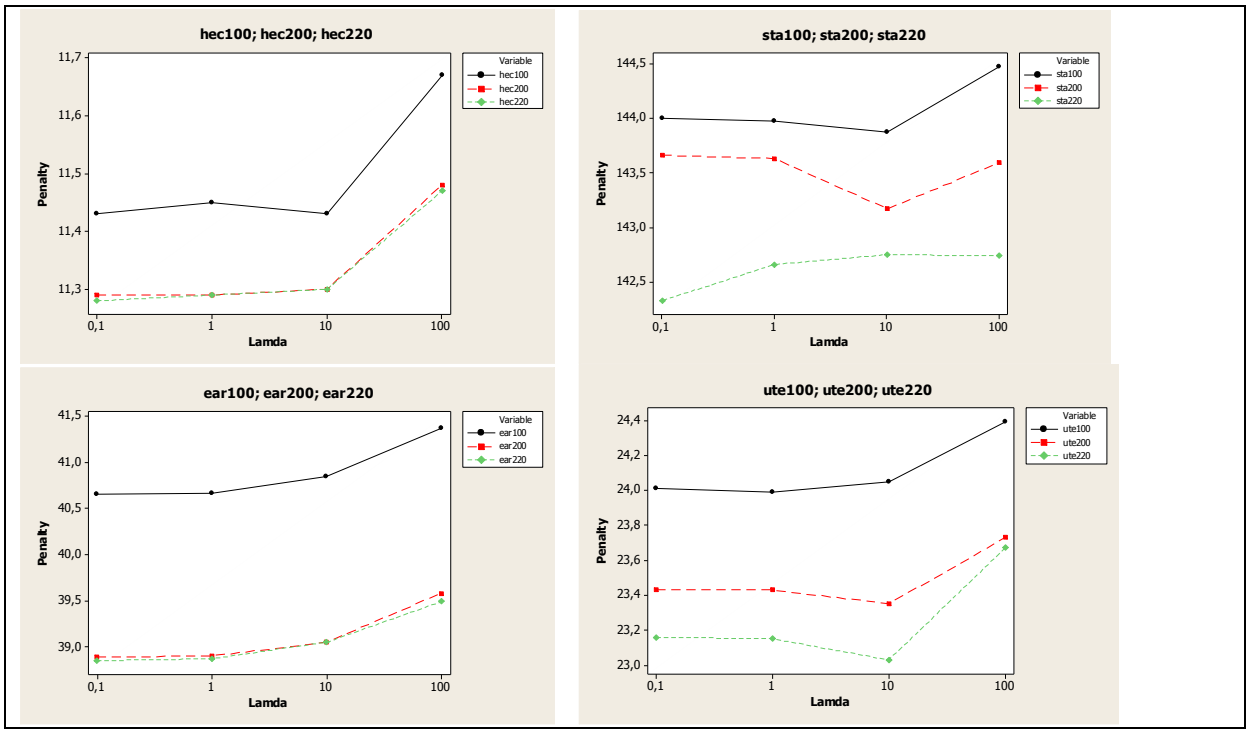


Figure 6. 17 Impact de la taille de la population

Nous remarquons qu'en générale, les résultats s'améliorent, avec l'augmentation de la taille de l'ensemble initial utilisé. Pour répondre au compromis qualité/temps, nous choisissons la taille 220 pour les tests suivants portant sur l'étude du comportement de la GGLS-c en fonction de λ.

b- Tests effectués sur l'approche GGLS –c

Expérimentation

Dans ces tests, nous avons fixé la taille de l'ensemble à 220. Le nombre d'itérations retenu est de 1000 pour hec et sta et de 100 pour ear et ute. Dix exécutions différentes sont effectuées pour chacun des problèmes hec, sta, ear et ute, afin de calculer pour chaque valeur de λ, la pénalité moyenne, la pénalité minimale, l'erreur standard moyenne, le temps moyen et le nombre moyen d'examens restants.

Résultats et analyse

Dans Figure 6.18, Figure 6.19, Figure 6.20, et Figure 6.21, cinq graphes rapportent en fonction de λ, les résultats de la GGLS-c obtenus respectivement pour les problèmes hec, sta, ear et ute. Les cinq graphes de chaque figure décrivent respectivement, de gauche à droite et de haut en bas, la pénalité moyenne, la pénalité minimale, l'erreur standard moyenne, le temps moyen et le nombre moyen d'examens restants.

Dans la Figure 6.18, on note pour le problème hec, les résultats suivants :

- Cette approche opère mieux, de $\lambda = 0.1$ à $\lambda = 9$, pour la pénalité moyenne et la pénalité minimale.
- La valeur minimale de la pénalité moyenne égale à 11.22, est obtenue pour $\lambda = 4$ et $\lambda = 6$, et la valeur minimale de la pénalité minimale égale à 10.65 est obtenue pour $\lambda = 5$, $\lambda = 6$ et $\lambda = 7$.
- Cette approche est peu sensitive à la variation de $\lambda = 0.1$ à $\lambda = 9$ pour la pénalité moyenne et la pénalité minimale.
- L'erreur standard est monotone de $\lambda = 0.1$ à $\lambda = 3$ et variable pour le reste.
- Le temps est monotone de $\lambda = 0.6$ à $\lambda = 40$, et variable pour le reste.
- Le nombre moyen des examens restants est nul.

Dans la Figure 6.19, nous avons pour le problème sta les résultats suivants :

- La meilleure pénalité moyenne égale à 143.27 est obtenue avec $\lambda = 8$ alors que la meilleure pénalité minimale égale à 140,18 est obtenue avec $\lambda = 20$.
- Cette approche est peu sensitive à la variation de $\lambda = 0.4$ à $\lambda = 1$ pour la pénalité moyenne et de $\lambda = 0.1$ à $\lambda = 3$ pour la pénalité minimale moyenne.
- L'erreur standard est monotone de $\lambda = 0.4$ à $\lambda = 4$ et variable pour le reste.
- Le temps a une allure variable.
- Le nombre moyen des examens restants est de 0.1. Autrement dit, seulement une solution sur dix exécutions a un examen restant.

Il est important de noter ici, que dans certains papiers, avec d'autres approches, les auteurs ne considèrent que les solutions sans examens restants. Dans notre cas, nous avons préféré donner une idée plus précise sur les tests effectués en indiquant que seulement, une solution sur les dix, a un examen non programmé.

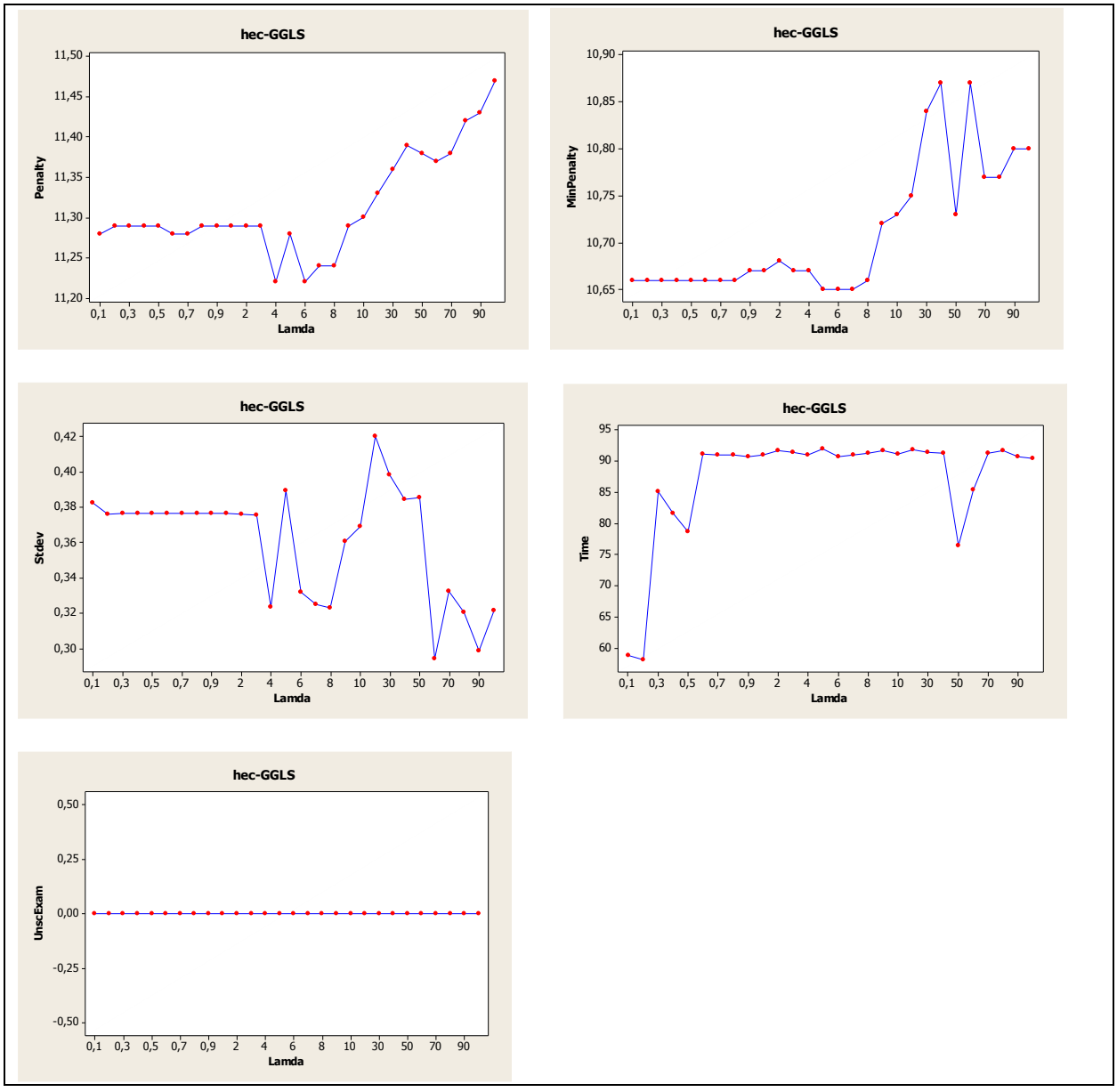


Figure 6. 18 Performances de la GGLS –c pour le problème hec

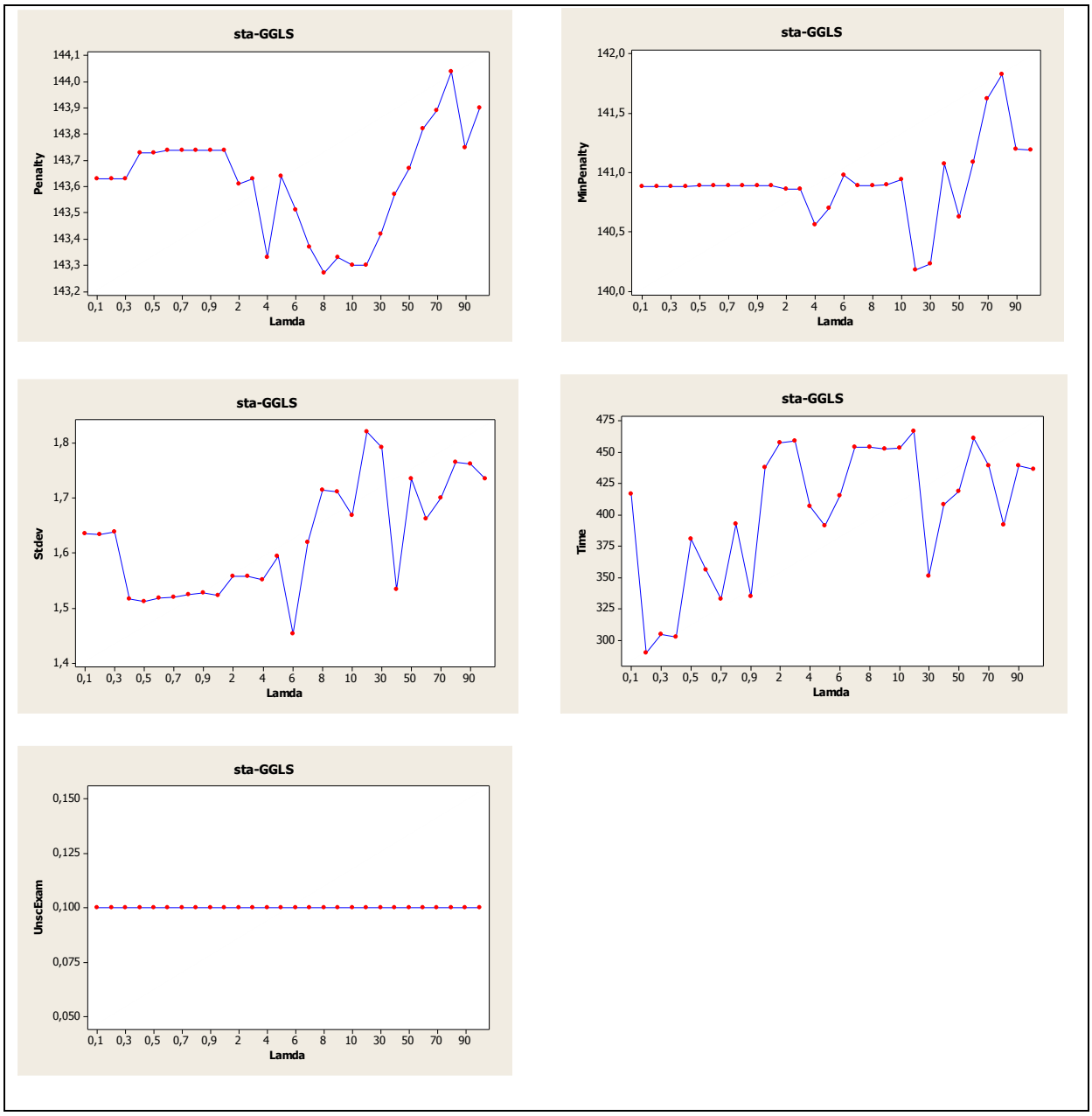


Figure 6. 19 Performances de la GGLS-c pour le problème sta

Dans la Figure 6.20, on note pour le problème ear les observations suivantes:

- la meilleure pénalité moyenne égale à 38.65 est obtenue avec λ =2 et la meilleure pénalité minimale égale à 35.82 est obtenue avec λ = 5.
- Cette approche est peu sensitive à la variation de λ = 0.1 à λ = 1 pour la pénalité moyenne et la pénalité minimale moyenne.
- L'erreur standard est monotone de λ = 0.2 à λ = 0.9, puis de λ = 30 à λ = 100 et variable pour le reste.
- Le temps est croissant de λ = 0.1 à λ = 60, puis décroissant de λ = 60 à λ = 70 et monotone pour le reste.
- le nombre moyen des examens restants est de 0.2, ce qui veut dire que pour ce problème, nous avons deux examens restants sur les dix solutions.

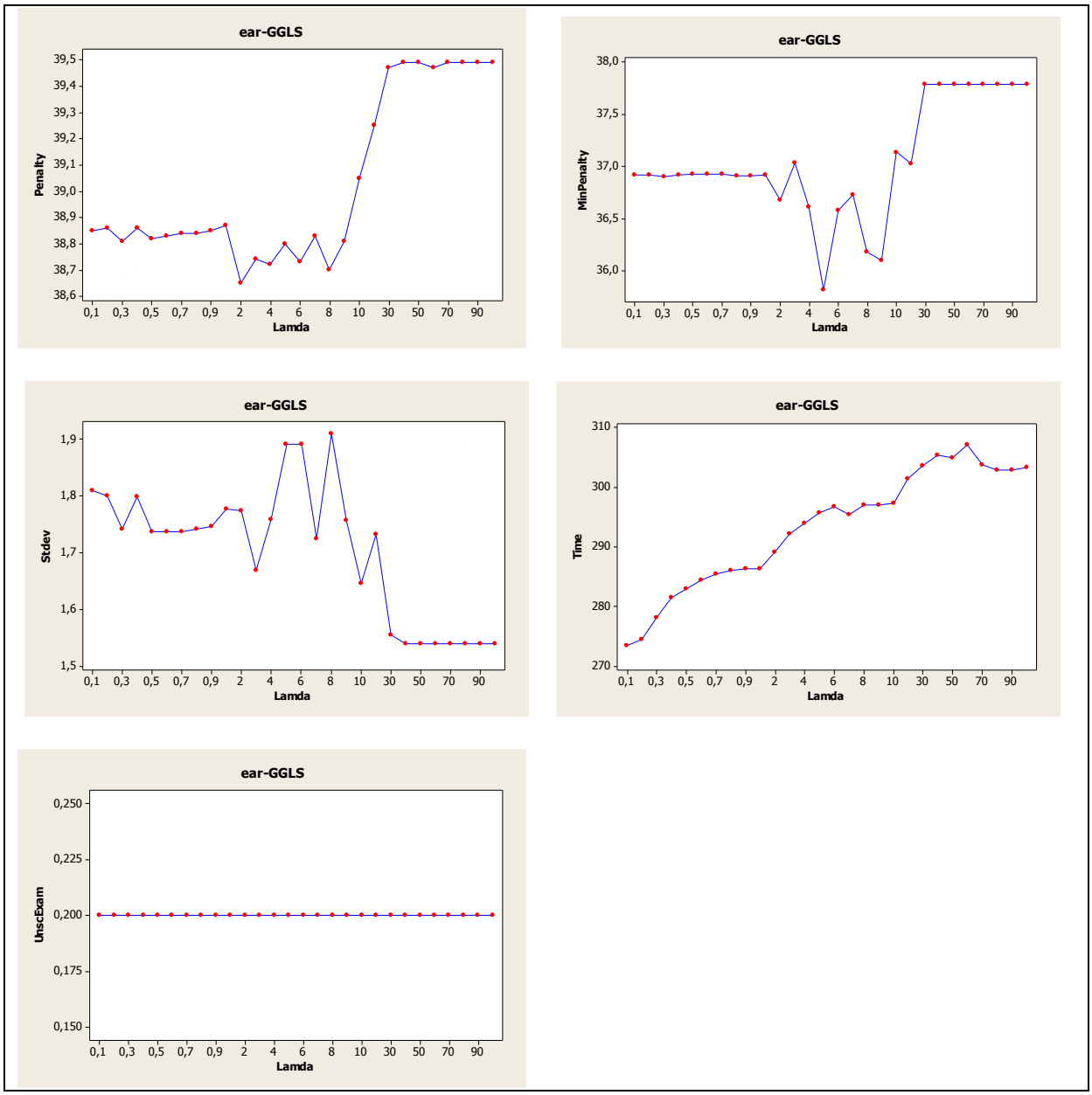


Figure 6. 20 Performances de la GGLS-c pour le problème ear

Dans la Figure 6.21, on note pour le problème ute que:

- Cette approche opère mieux de $\lambda = 0.1$ à $\lambda = 20$ pour la pénalité moyenne, la meilleure égale à 23.03 est obtenue avec $\lambda = 10$ alors que la meilleure pénalité minimale égale à 20.33 est obtenue avec $\lambda = 3$.
- Cette approche est peu sensitive à la variation de $\lambda = 0.1$ à $\lambda = 8$ pour la pénalité moyenne, et de $\lambda = 0.5$ à $\lambda = 2$ pour la pénalité minimale moyenne.
- L'erreur standard est monotone de $\lambda = 0.1$à $\lambda = 2$ et variable pour le reste.
- Le temps est croissant de $\lambda = 0.1$ à $\lambda = 3$ puis variable pour le reste.
- Le nombre moyen des examens restants est de 0.4, ce qui veut dire que pour ce problème, nous avons 4 examens restants sur les dix solutions.

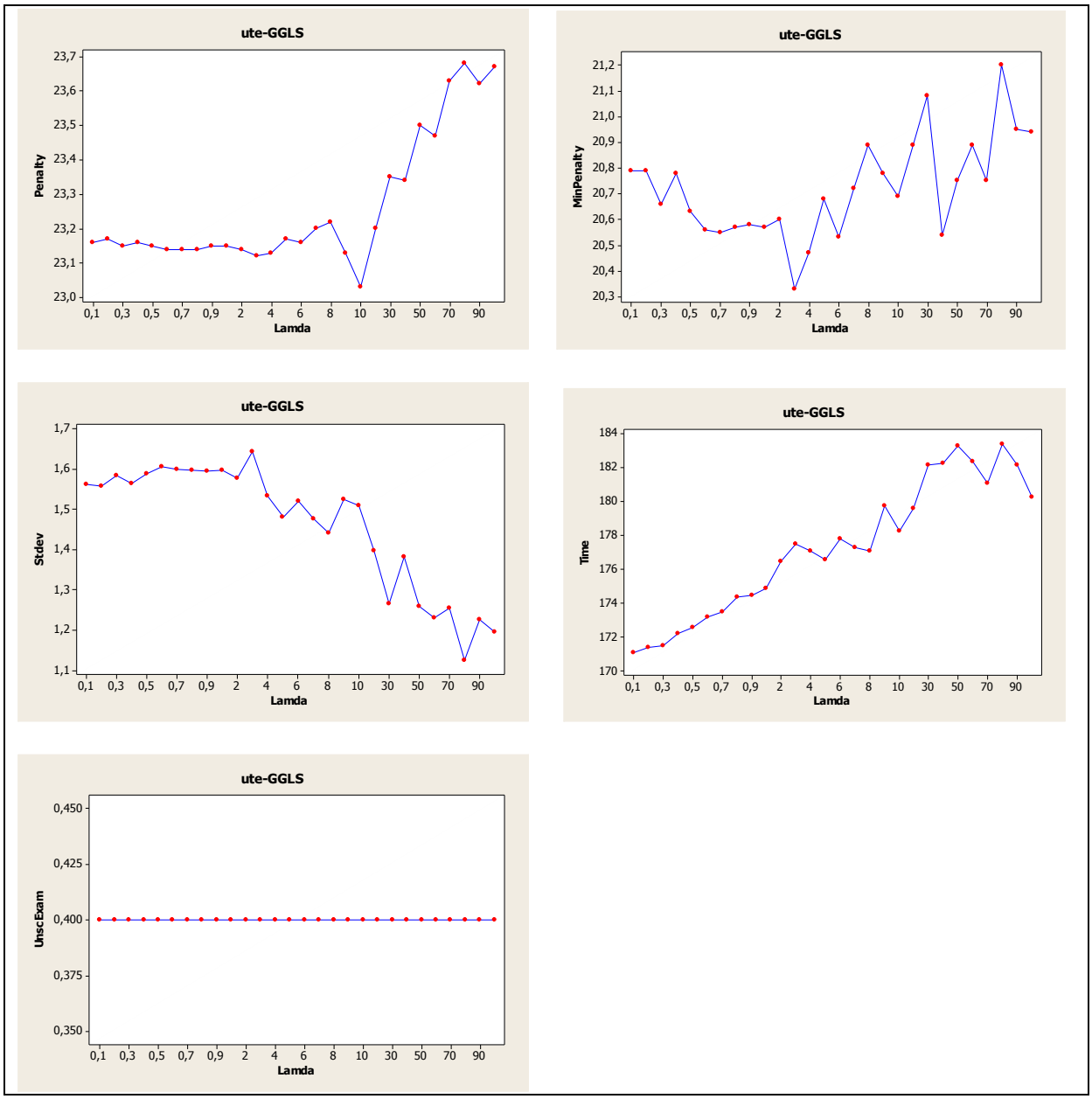


Figure 6. 21 Performances de la GGLS-c pour le problème ute

Nous observons à travers ces graphes, que les résultats obtenus dépendent généralement, des variations de λ, mais que certains problèmes sont moins sensitifs que d'autres. Ce phénomène apparaît généralement, dans le cas où la valeur de λ est grande, augmentant l'influence des pénalités dans la fonction objectif modifiée. Les petites valeurs de λ, au contraire, réduisent dans la plus part des cas, l'effet des pénalités et diminuent la probabilité d'avoir ces situations.

A présent nous nous posons la question suivante ?

Est-ce que ça vaut la peine d'augmenter le nombre des itérations pour améliorer ces résultats ?

Pour répondre à cette question, nous avons effectué pour le problème hec avec $\lambda = 90$, des tests avec les nombres d'itérations 100,1000,5000,50000,80000. Les résultats obtenus sont présentés dans la Table 6.12.

Pro Nbiter λ=90	**Hec 10**	**Hec 100**	**Hec 1000**	**Hec 5000**	**Hec 50000**	**Hec 80000**
Moy-Pénalité	11.44	11.44	11.44	11.44	11.44	11.33
Min-Pénalité	10.80	10.80	10.80	10.80	10.80	10.54
Moy-examens-restants	0	0	0	0	0	0
Erreur standard	0.2986	0.2986	0.2986	0.2986	0.2986	0.3985
Temps (s)	11.7	15.1	50	206.4	3350.6	4761.4

Table 6. 12 Impact du nombre des itérations sur la qualité des résultats pour hec

Nous observons d'après ces résultats, que l'amélioration n'apparait qu'à partir de 80000 itérations. Pour les problèmes de grande taille, ce nombre augmentera davantage. Ainsi, ceci justifie bien notre choix de générer une solution initiale de bonne qualité, dans un temps raisonnable, afin d'utiliser le reste du temps de l'approche pour le raffinage.

Dans la variante GGLS-d hybridant la Grasp avec la GLS, une autre forme d'exploitation des stratégies, LWD et SD, est utilisée pour générer l'ensemble de solutions à partir desquelles nous choisissons la meilleure, comme solution initiale. La méthode utilisée est décrite dans l'Algorithme 5.2. Les meilleures solutions, obtenues sur 10 essais différents, sont rapportées dans la Table 6.13.

Prob	**Meilleures solutions de GGLS-d**
Hec	11.34 λ=90
Sta	136.76 λ=37
ear	36.11 λ=7
ute	24.93 λ=5
uta	3.67 λ=0.7

Table 6. 13 Meilleures résultats de la variante GGLS-d

A l'analyse des résultats, nous avons constaté, que malgré l'amélioration des solutions initiales dans la GGLS-d, la variante GGLS-c opère nettement mieux. Ceci peut être expliqué par la

combinaison du rôle de la LWD dans les premiers stades de construction des solutions avec une mise à jour adaptative.

Dans la suite, nous allons approfondir nos investigations, sur la Recherche Locale Guidée, pour le problème d'emploi du temps des examens, en lui rajoutant une stratégie d'aspiration, puis une stratégie de mouvements aléatoires et enfin la combinaison des deux. A noter que l'approche GGLS-c sera désignée, tout cours, par GGLS.

6.4.3 Stratégie d'aspiration pour la Recherche Locale Guidée

Nous avons vu dans le chapitre précédent, que dans la Recherche Locale Guidée, certains attributs pénalisés agissent comme une barrière. Ils empêchent la GLS de visiter les solutions affichant les mêmes attributs, alors qu'elles peuvent être intéressantes. Ceci justifie donc, les besoins d'inclure, sans tenir compte de la valeur de λ, une stratégie d'aspiration permettant d'éviter cette situation.

Pour notre approche de la Recherche Locale Guidée, le critère d'aspiration choisi est basé sur le meilleur mouvement d'amélioration. Il est ici défini comme étant, l'acceptation d'un mouvement produisant la meilleure solution obtenue jusque là, et qui n'aurait pas été choisie, par la recherche locale utilisant la fonction objectif augmentée, sans stratégie d'aspiration.

Dans la suite, nous présentons les résultats de nos tests effectués sur la nouvelle approche de la GLS, notée Asp, résultante de la combinaison de la GGLS avec une stratégie d'aspiration.

Expérimentation

Pour étudier expérimentalement la nouvelle approche notée Asp, nous posons les mêmes hypothèses que celles des tests précédents.

Résultats et analyse

Les performances de l'approche Asp ainsi que celles de la GGLS sont reportées en même temps pour la comparaison, dans les graphes présentés par Figure 6.22, Figure 6.23, Figure 6.24 et Figure 6.25.

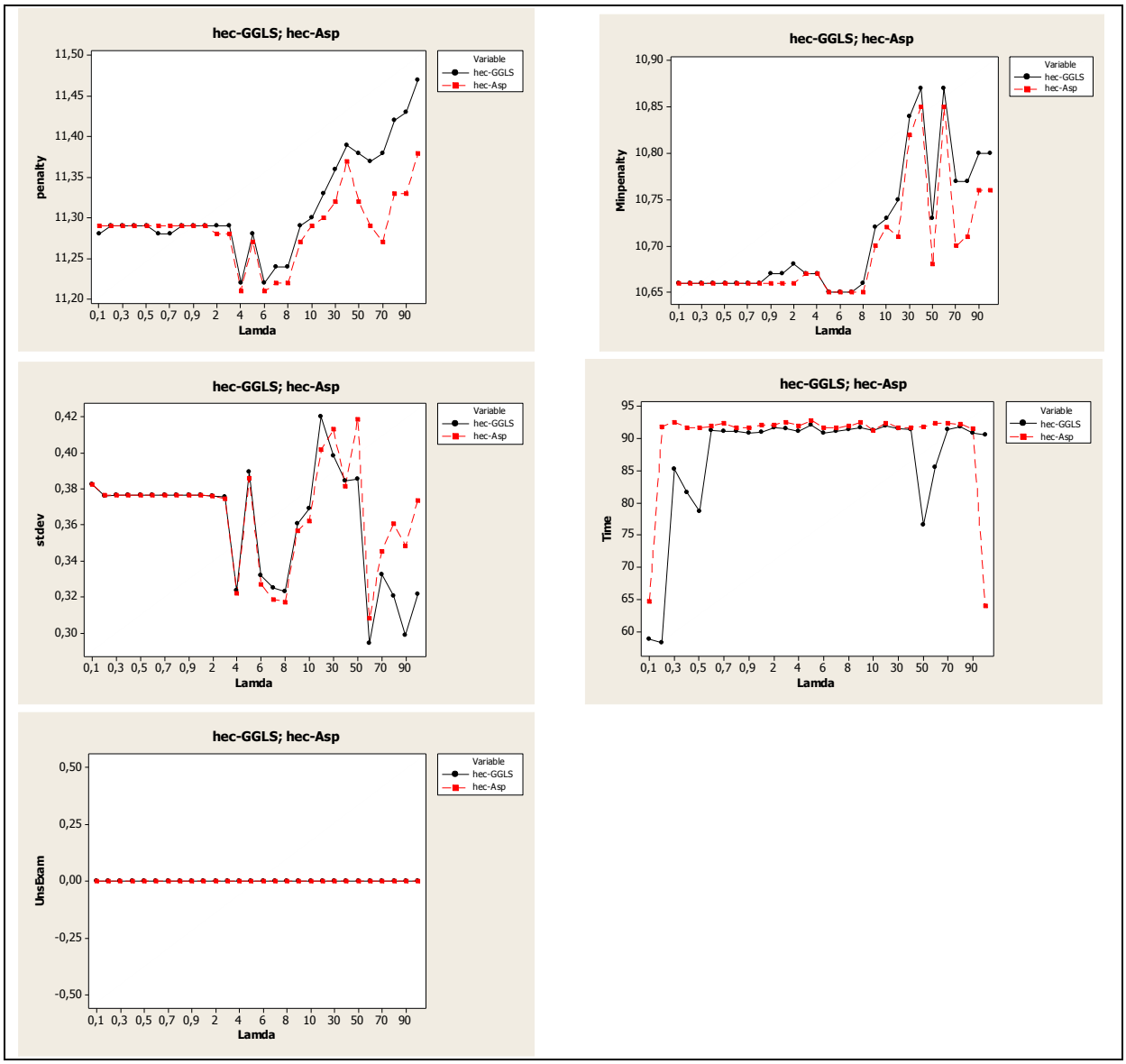


Figure 6. 22 Performances de la GGLS et Asp pour le problème hec

Dans la Figure 6.22 on note que pour le problème hec :

- L'Asp opère, de manière assez semblable à la GGLS, de λ = 0.1à λ = 6, pour la pénalité moyenne et de λ = 0.1à λ = 0.8, pour la pénalité minimale. Au delà l'Asp améliore les résultats pour les deux métriques, particulièrement avec les grandes valeurs de λ.
- L'erreur standard évolue d'une manière assez semblable de λ = 0.1à λ = 70 et reste dans le même ordre de grandeur pour le reste.
- Le temps est dans l'ensemble le même, avec une petite diminution de λ = 0.1 à λ = 90 en faveur de la GGLS, et une nette diminution pour λ = 100 en faveur de l'Asp.
- Le nombre des examens restants est le même que celui de la GGLS.

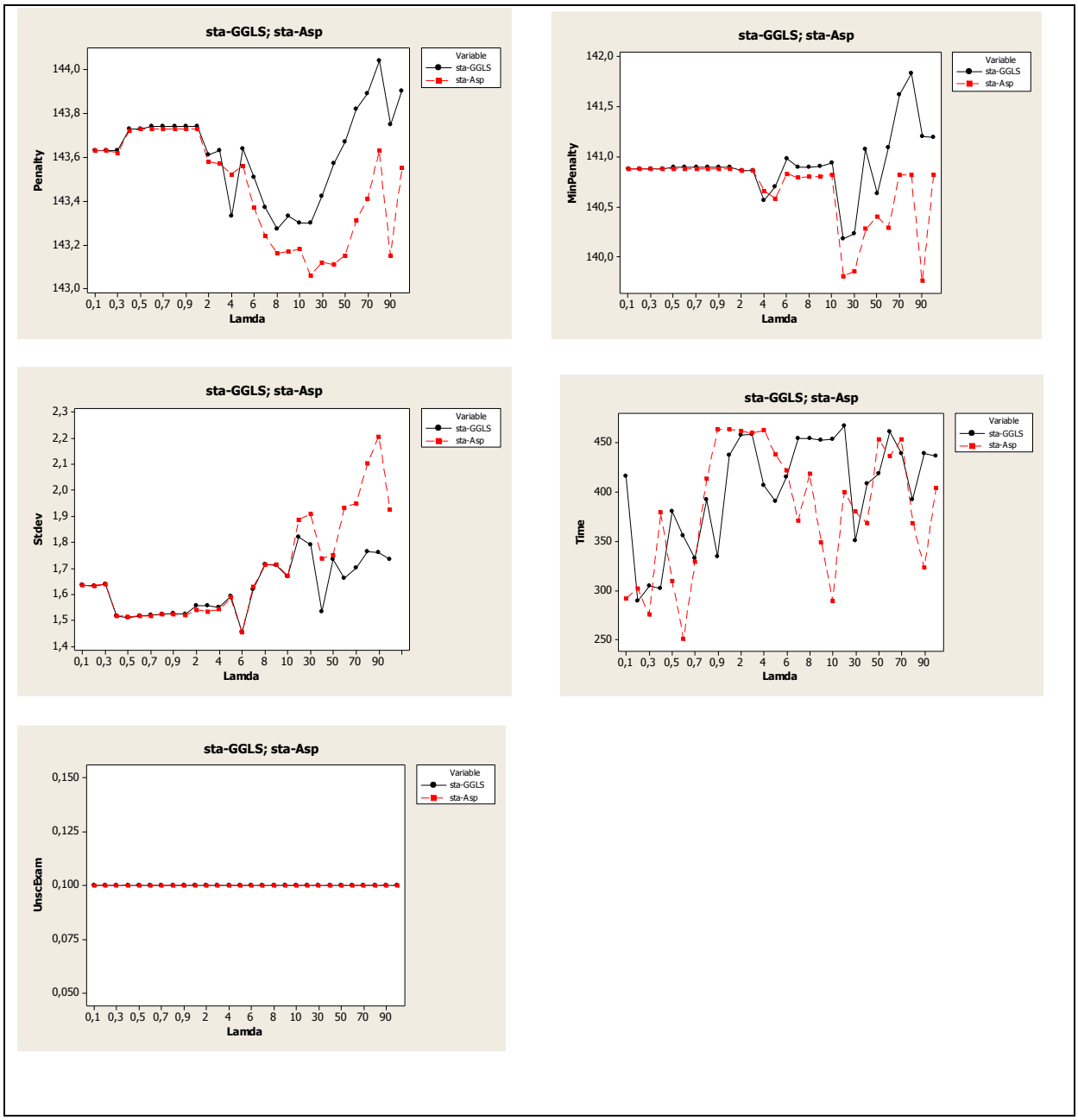


Figure 6. 23 Performances de la GGLS et Asp pour le problème sta

Dans la Figure 6.23 on note que pour le problème sta :

- L'Asp opère de manière identique à la GGLS de $\lambda = 0.1$ à $\lambda = 2$, pour la pénalité moyenne, et de $\lambda = 0.1$ à $\lambda = 3$, pour la pénalité minimale. Au-delà, l'Asp améliore légèrement les résultats pour les deux métriques, sauf pour $\lambda = 4$ dans le cas de la pénalité moyenne.
- L'erreur standard de l'Asp évolue de manière assez monotone de $\lambda = 0.1$ à $\lambda = 10$, puis augmente légèrement pour le reste.
- Le temps varie plus ou moins bien, sans trop dépasser le temps maximal de la GGLS.
- Le nombre des examens restants est le même que celui de la GGLS.

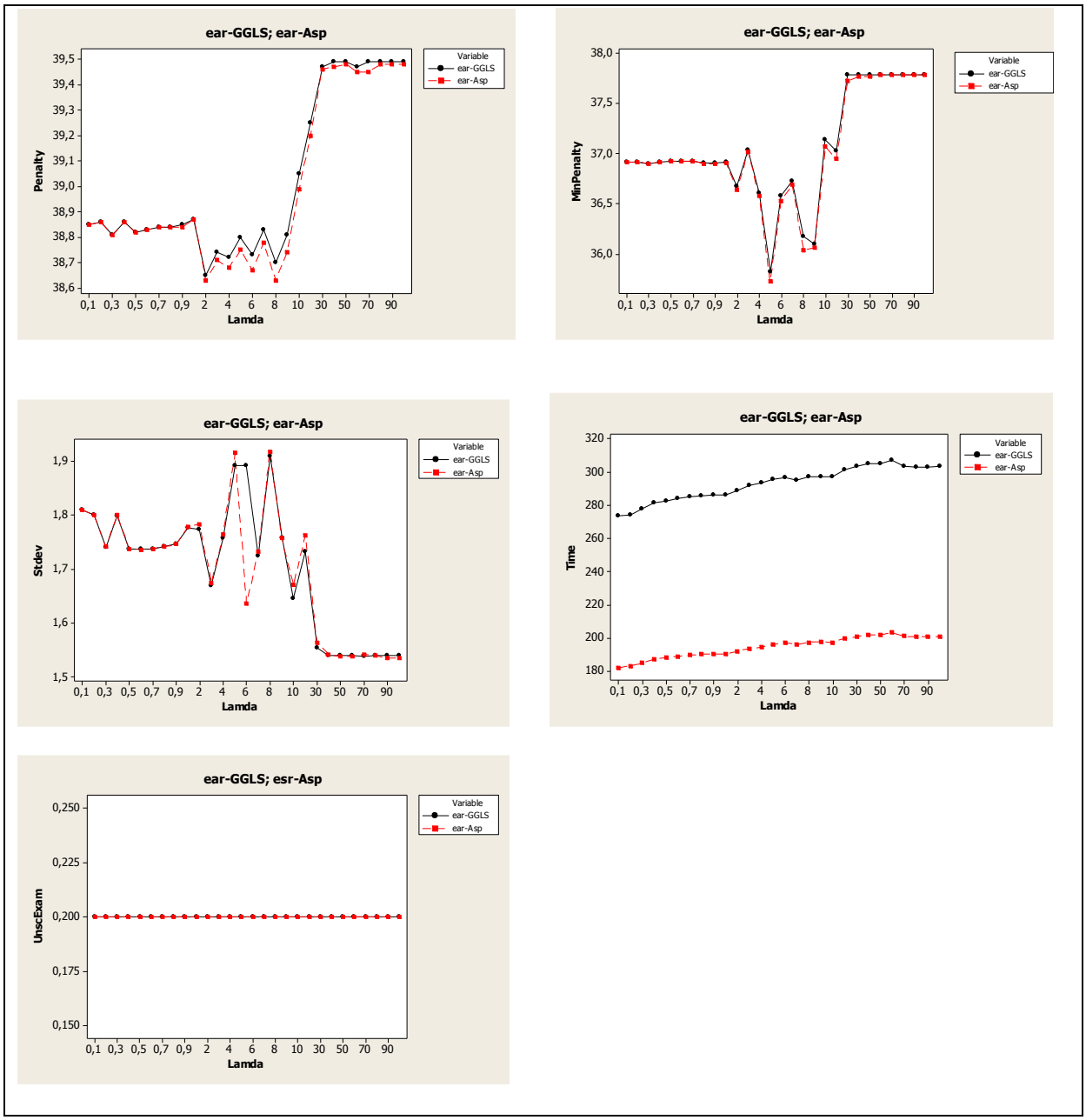


Figure 6. 24 Performances de la GGLS et Asp pour le problème ear

Dans la Figure 6.24 on note que pour le problème ear :

- Nous avons une faible amélioration de l'Asp par rapport à la GGLS à partir de $\lambda = 2$, pour la pénalité moyenne, et une variation assez semblable pour la pénalité minimale.
- L'erreur standard évolue de la même manière, sauf de $\lambda = 5$ à $\lambda = 7$ où elle est inférieure.
- Le temps est en nette diminution en faveur de l'Asp.
- Le nombre des examens restants est le même que celui de la GGLS.

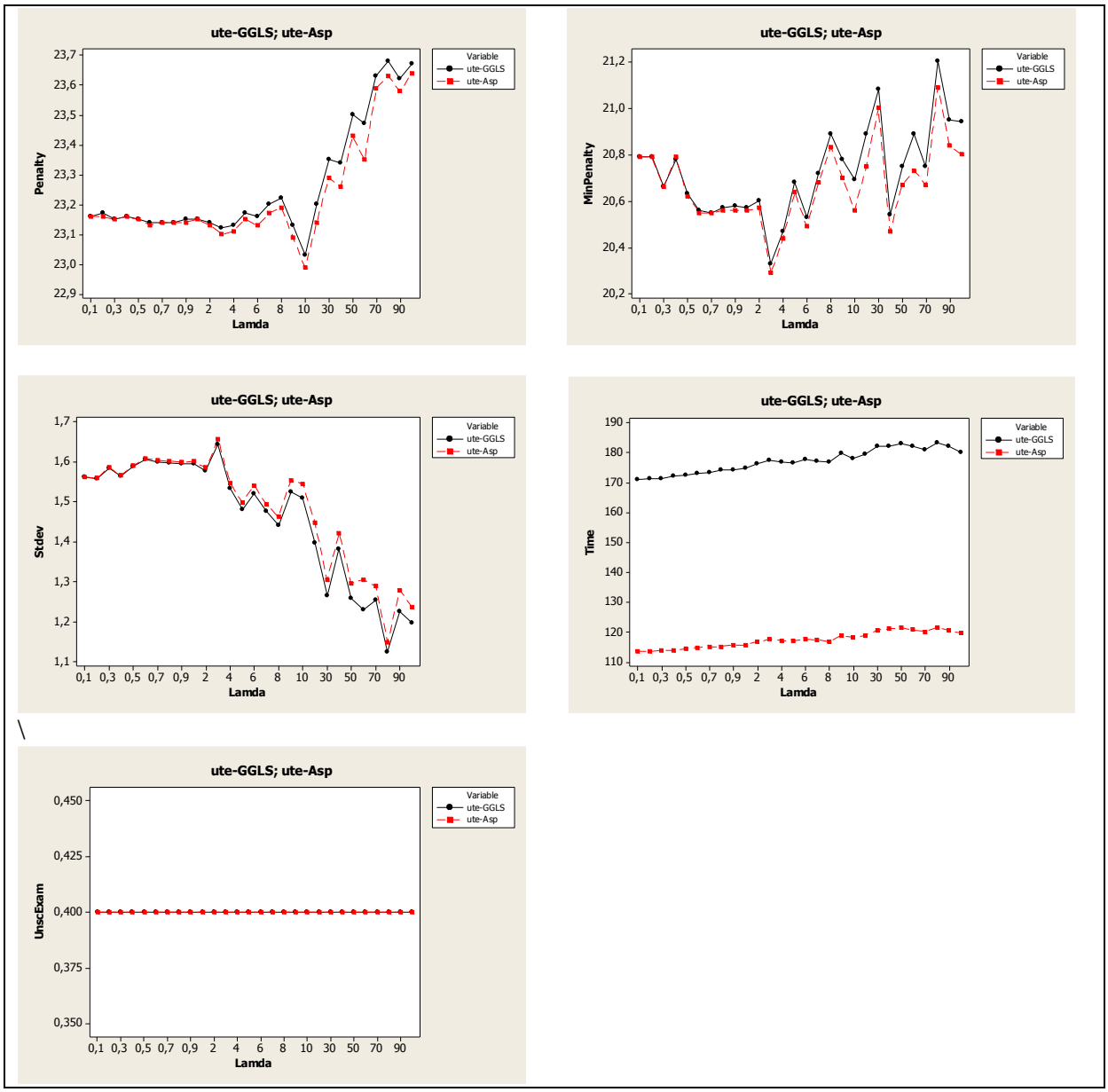


Figure 6. 25 Performances de la GGLS et Asp pour le problème ute

Dans la Figure 6.25 on note que pour le problème ute :

- L'Asp opère de manière identique à la GGLS, de $\lambda = 0.1$ à $\lambda = 3$ pour la pénalité moyenne, et de $\lambda = 0.1$ à $\lambda = 8$ pour la pénalité minimale. Pour le reste, elle marque une faible amélioration.
- L'erreur standard évolue d'une manière presque semblable.
- Le temps diminue nettement en faveur de l'Asp.
- Le nombre des examens restants est identique à celui de la GGLS.

D'après les résultats obtenus, illustrés par les graphes précédents, nous remarquons en comparant la GGLS à l'Asp, que généralement, le critère d'aspiration ne détériore pas les performances de la pénalité moyenne de la GGLS. Au contraire, il les améliore dans bien des cas. Cette amélioration

est due principalement, aux conditions d'utilisation de l'aspiration. Car celle-ci n'est appliquée que lorsqu'il existe un mouvement générant pour la fonction d'origine g, une solution de meilleure qualité que la meilleure trouvée jusque là, et qui n'aurait pas été prise si on tenait compte des pénalités.

La faible amélioration, notée pour certains problèmes, peut s'expliquer par la présence de plateaux où la GLS seule, opère déjà bien.

En conclusion, nous dirons que l'aspiration est une méthode d'amélioration qui évite d'ignorer des solutions intéressantes de la GGLS, omises à causes des pénalités auxquelles elles sont assujetties. Ceci apparait particulièrement utile, pour les problèmes hec et sta où l'aspiration a permis d'améliorer la GGLS, pour les grandes valeurs de λ ayant une plus grande influence sur les pénalités. Elle permet donc, d'augmenter l'ensemble des valeurs de λ produisant de bons résultats et réduit par voie de conséquence, la sensitivité à ce paramètre.

6.4.4 Stratégie des mouvements aléatoires pour la Recherche Locale Guidée

Pour permettre à notre approche basée sur la GGLS d'éviter d'être piégé dans une région de l'espace de recherche, principalement, lorsque la valeur de λ est petite, nous allons lui introduire une stratégie de mouvements aléatoires basée sur l'Algorithme 5.7 .

Expérimentation

Pour cette étude expérimentale, sur notre approche appelée Rand, un série de tests préalables, nous ont permis de constater que toutes les valeurs de *Prandmove* apportent des améliorations très légèrement différentes, les unes des autres. Cependant, le temps est croissant avec l'accroissement de ces valeurs. Pour un compromis qualité- temps, avec l'algorithme 5.7, nous avons fixé la valeur de m à 0.6. Pour l'étude de l'impact des mouvements aléatoires sur les performances de la GGLS, nous posons les mêmes hypothèses que celles des tests portant sur l'impact de l'aspiration.

Résultats et analyse

Les performances de la Rand ainsi que celles de la GGLS sont reportées en même temps, pour la comparaison, dans les graphes présentés par Figure 6.26, Figure 6.27, Figure 6.28 et Figure 6.29.

Dans la Figure 6.26 on note que pour le problème hec :

- La Rand, améliore les performances de $\lambda = 0.1$ à $\lambda = 30$, pour la pénalité moyenne, et de $\lambda = 0.1$à $\lambda = 40$, pour la pénalité minimale, excepté pour $\lambda = 0.4$ et $\lambda = 5$ où la solution est de qualité inférieure à celle de la GGLS. Au-delà, la GGLS seule est meilleure pour les deux mesures.
- L'erreur standard évolue de manière plus ou moins bonne, mais reste dans le même ordre de grandeur.
- Le temps évolue de manière plus ou moins bonne, de $\lambda = 0.1$ à $\lambda = 7$, mais augmente nettement pour le reste.
- Le nombre des examens restants est le même que celui de la GGLS.

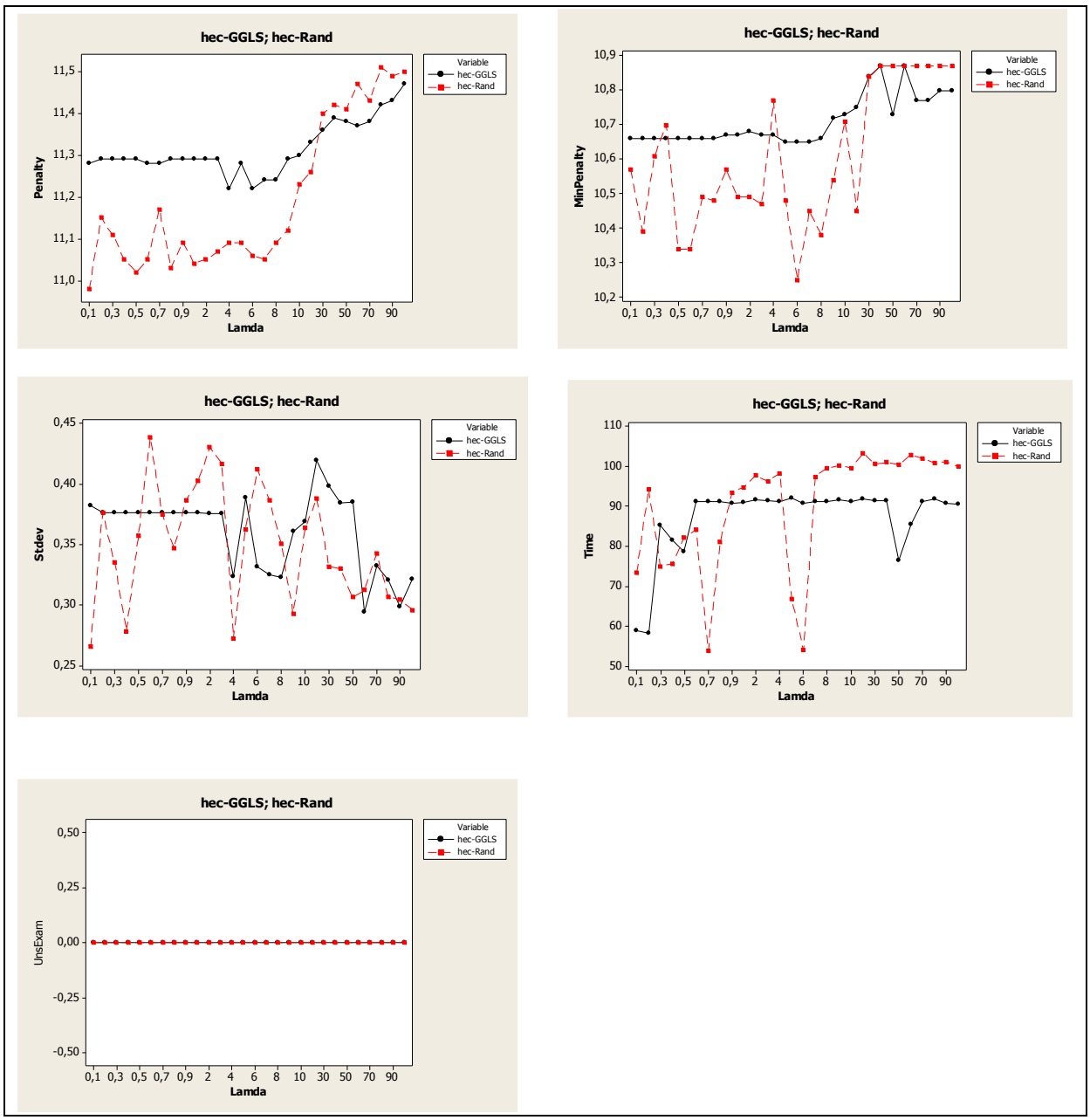


Figure 6. 26 Performances de la GGLS et Rand pour le problème hec

Dans la Figure 6.27 on note que pour le problème sta :

- La Rand améliore les performances de la pénalité moyenne et la pénalité minimale, pour toutes les valeurs de λ excepté λ = 30 dans le cas de la pénalité moyenne et λ = 20, dans le cas de la pénalité minimale.
- L'erreur standard affiche une variation plus ou moins grande par rapport à celle de la GGLS.
- le temps est généralement un peu plus élevé que dans le cas de la GGLS.
- Le nombre des examens restants est le même que celui de la GGLS.

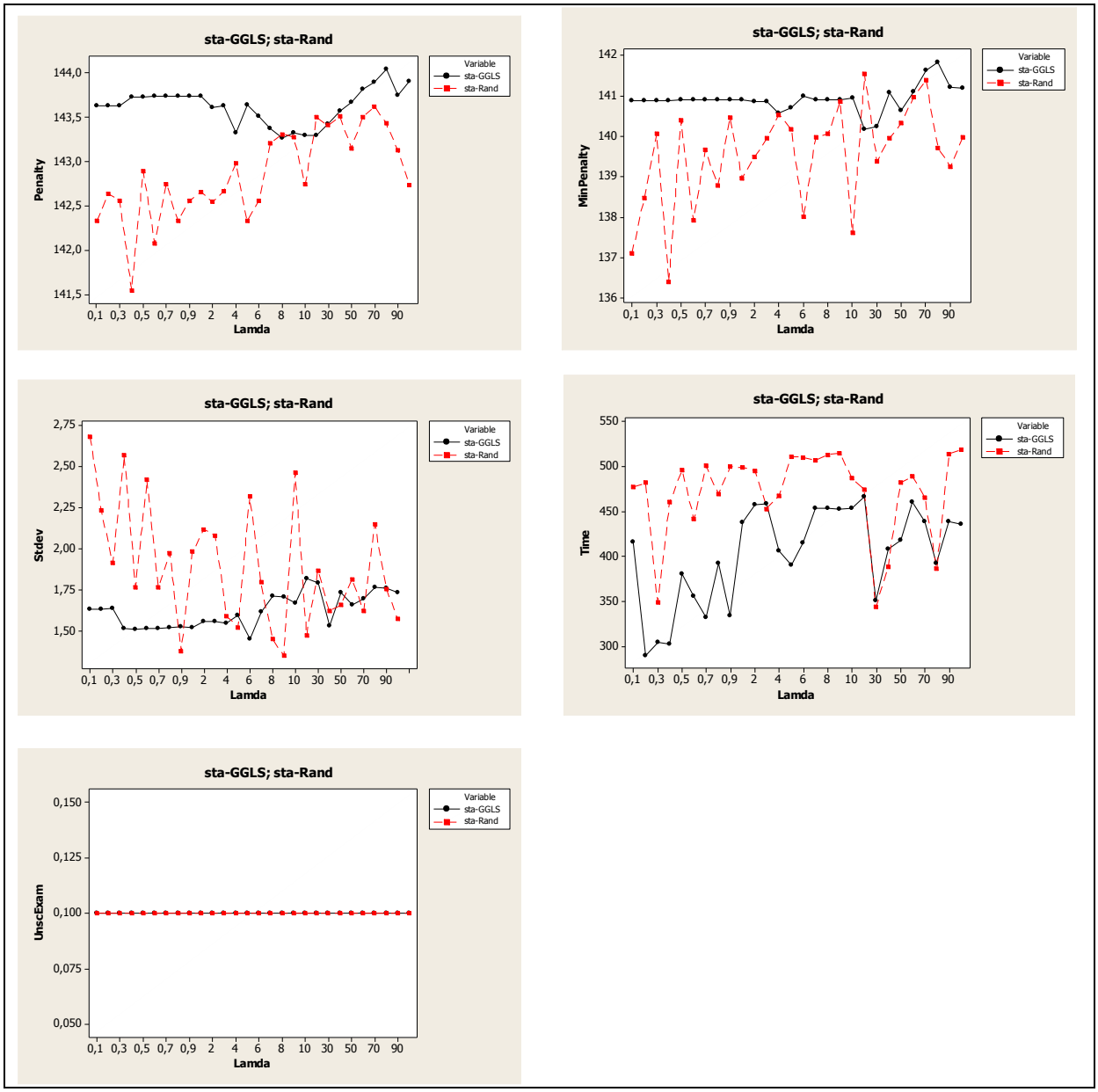


Figure 6. 27 Performances de la GGLS et Rand pour le problème sta

Dans la Figure 6.28 on note que pour le problème ear :

- La Rand améliore nettement les performances de la pénalité moyenne, pour toutes les valeurs de λ, et légèrement celles de la pénalité minimale, pour toutes les valeurs de λ, excepté, pour λ = 5.

- L’erreur standard affiche des variations de valeurs moindres.
- le temps est nettement inférieur à celui de la GGLS.
- Le nombre des examens restants est inférieur à celui de la GGLS.

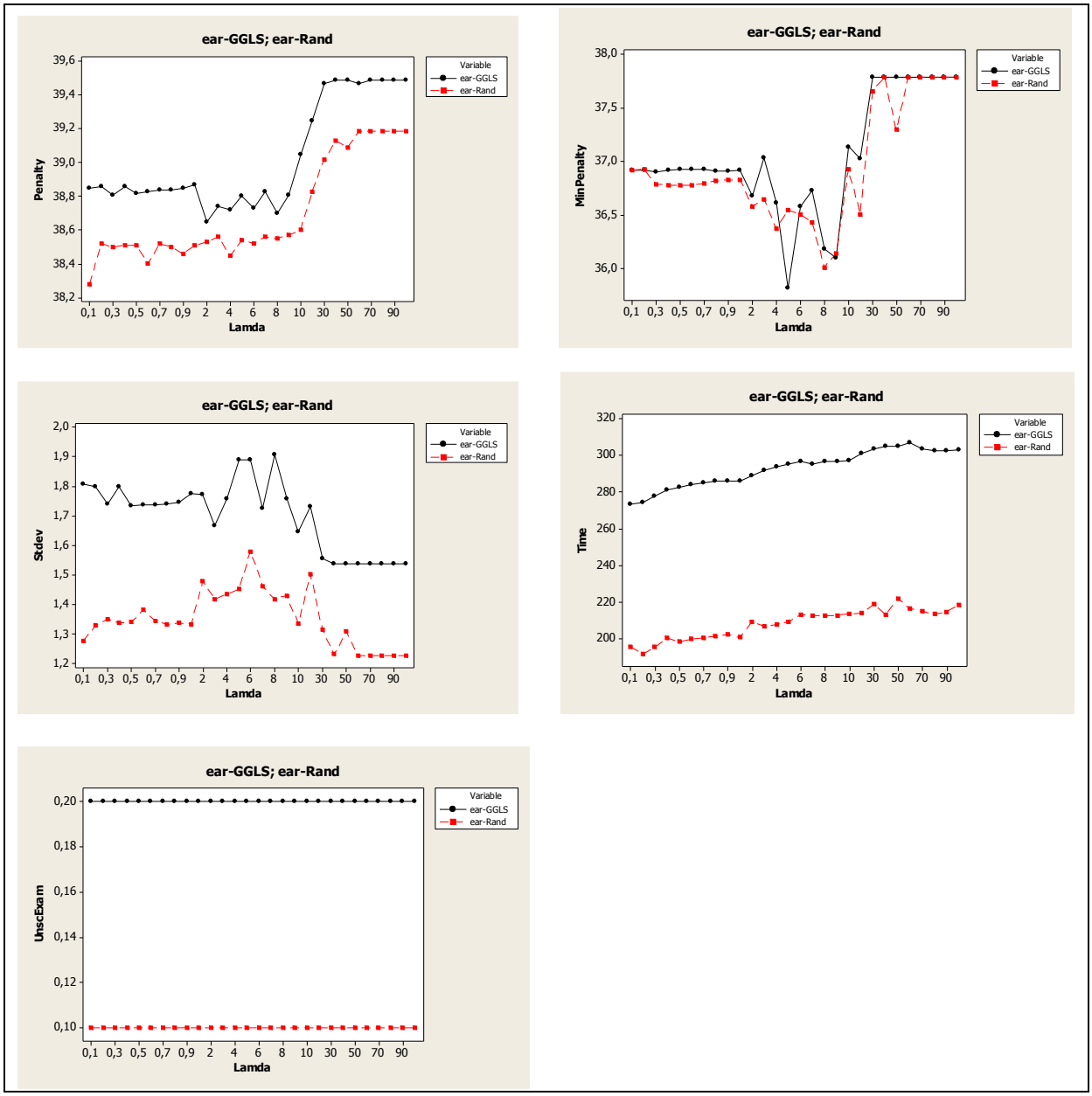


Figure 6. 28 Performances de la GGLS et Rand pour le problème ear

Dans la Figure 6.29 on note que pour le problème ute :

- La Rand améliore nettement, les performances de la pénalité moyenne, pour toutes les valeurs de λ, et légèrement, celles de la pénalité minimale, de λ = 0.1à λ = 10, sauf pour λ = 3,4 et 6.
- L’erreur standard est de valeur inférieur de λ = 0.1à λ = 4 et varie légèrement plus ou moins bien, pour les autres valeurs de λ.
- Le temps est nettement inférieur pour toutes les valeurs de λ.

- Le nombre des examens restants est inférieur à celui de la GGLS, et particulièrement nul de $\lambda = 0.1$ à $\lambda = 0.4$ et de $\lambda = 0.8$ à $\lambda = 3$.

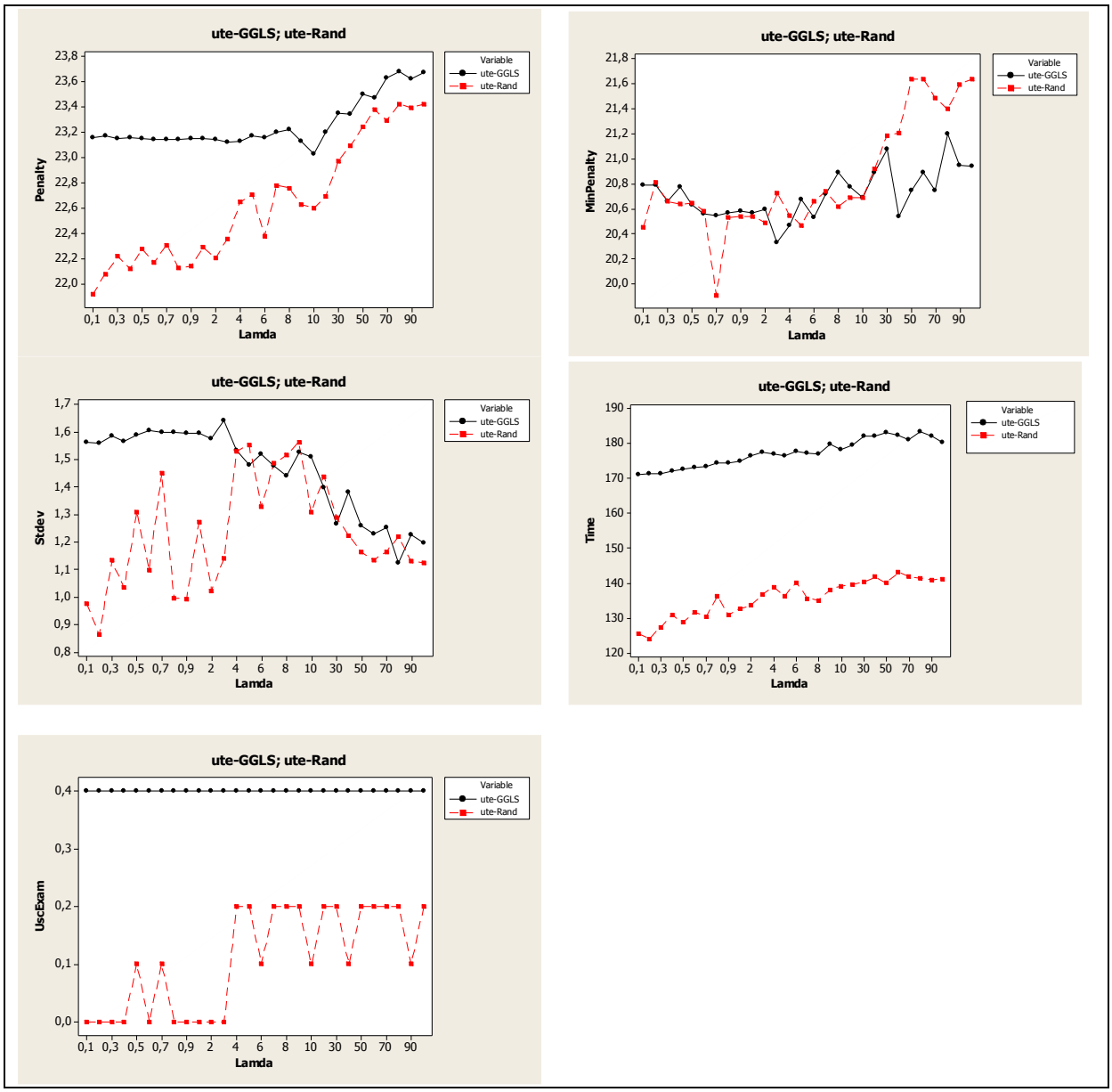


Figure 6. 29 Performances de la GGLS et Rand pour le problème ute

A la lumière des résultats obtenus, nous notons en général, que l'addition des mouvements aléatoires à la GGLS améliore la qualité des solutions, particulièrement, pour les petites valeurs de λ. Nous pensons ici, que l'espace de recherche de nos problèmes comprend beaucoup d'optima locaux et que ces mouvements aléatoires contribuent à la diversification de la recherche, en lui permettant de s'en échapper. Ceci est particulièrement, vérifié pour les petites valeurs de λ souvent, incapables de résoudre ce problème, car nécessitant, comme nous l'avons précédemment vu, un nombre d'itérations plus élevé, pour échapper à un optimum local. Lorsque λ prend des valeurs plus élevées, ces mouvements pourraient engendrer une légère détérioration, comme c'est

le cas avec hec et sta. Une explication à cela, pourrait être que les mouvements aléatoires ne soient pas, dans ces problèmes, aussi efficaces que les pénalités, gérées plus méthodiquement.

Par conséquent, on peut conclure, que la stratégie des mouvements aléatoire, peut contribuer de manière significative, à améliorer les performances de la GGLS pour des problèmes ayant beaucoup d'optima locaux, car, elle fournit un moyen non négligeable, permettant de s'en échapper. Particulièrement, lorsque λ prend de petites valeurs.

6.4.5 Aspiration et mouvements aléatoires pour la Recherche Locale Guidée

Dans les deux sections précédentes, nous avons étudié l'impact d'une stratégie d'aspiration puis, d'une stratégie de mouvements aléatoires, sur les performances de l'approche GGLS. D'après les résultats obtenus, nous avons constaté que, généralement, l'aspiration ne détériore pas les performances de la GGLS, mais qu'au contraire, elle les améliore dans bien des cas, particulièrement, pour les grandes valeurs de λ. Pour les mouvements aléatoires, nous avons remarqué une nette amélioration, pour les petites valeurs de λ. Dans la suite, nous présentons une étude expérimentale sur la combinaison de ces deux stratégies, afin d'analyser également, son impact sur les performances de la GGLS.

Expérimentation

Pour étudier l'impact de la combinaison des stratégies d'aspiration et des mouvements aléatoires sur les résultats finaux de la GGLS, nous avons effectué nos tests avec les mêmes conditions que celles adoptées dans les sections 6.4.3 et 6.4.4.

Résultats et analyse

Les performances de cette combinaison notée Rand-Asp, ainsi que celles de l'Asp, la Rand et la GGLS sont reportées, en même temps pour la comparaison, dans les graphes présentés par Figure 6.30, Figure 6.31 Figure 6.32 et Figure 6.33.

Dans la Figure 6.30 on note pour le problème hec, les observations suivantes:

- Pour la pénalité moyenne, la Rand-Asp est meilleure que la GGLS pour toutes les valeurs de λ, sauf pour $\lambda = 30$ où elle est comparable. Elle est meilleure que l'Asp de $\lambda = 0.1$ à $\lambda = 50$ excepté $\lambda = 30$ et de qualité inférieure de $\lambda = 50$ à $\lambda = 100$. Elle est également, meilleure que la Rand pour toutes les valeurs de λ.
- Pour la pénalité minimale, la Rand-Asp est meilleure que la GGLS et l'Asp de $\lambda = 0.1$ à $\lambda = 40$ et légèrement inférieure pour le reste. Elle est meilleure que la Rand pour toutes les valeurs de λ.

- L'erreur standard affiche une variation variable proche de celle de la Rand et plus ou moins grande que celles d'Asp et la GGLS.
- Le temps par contre, est légèrement supérieur ou égal a celui des autres de λ =0.2 à λ =40 et variable ailleurs.
- Le nombre des examens restants est nul et identique à celui des autres.

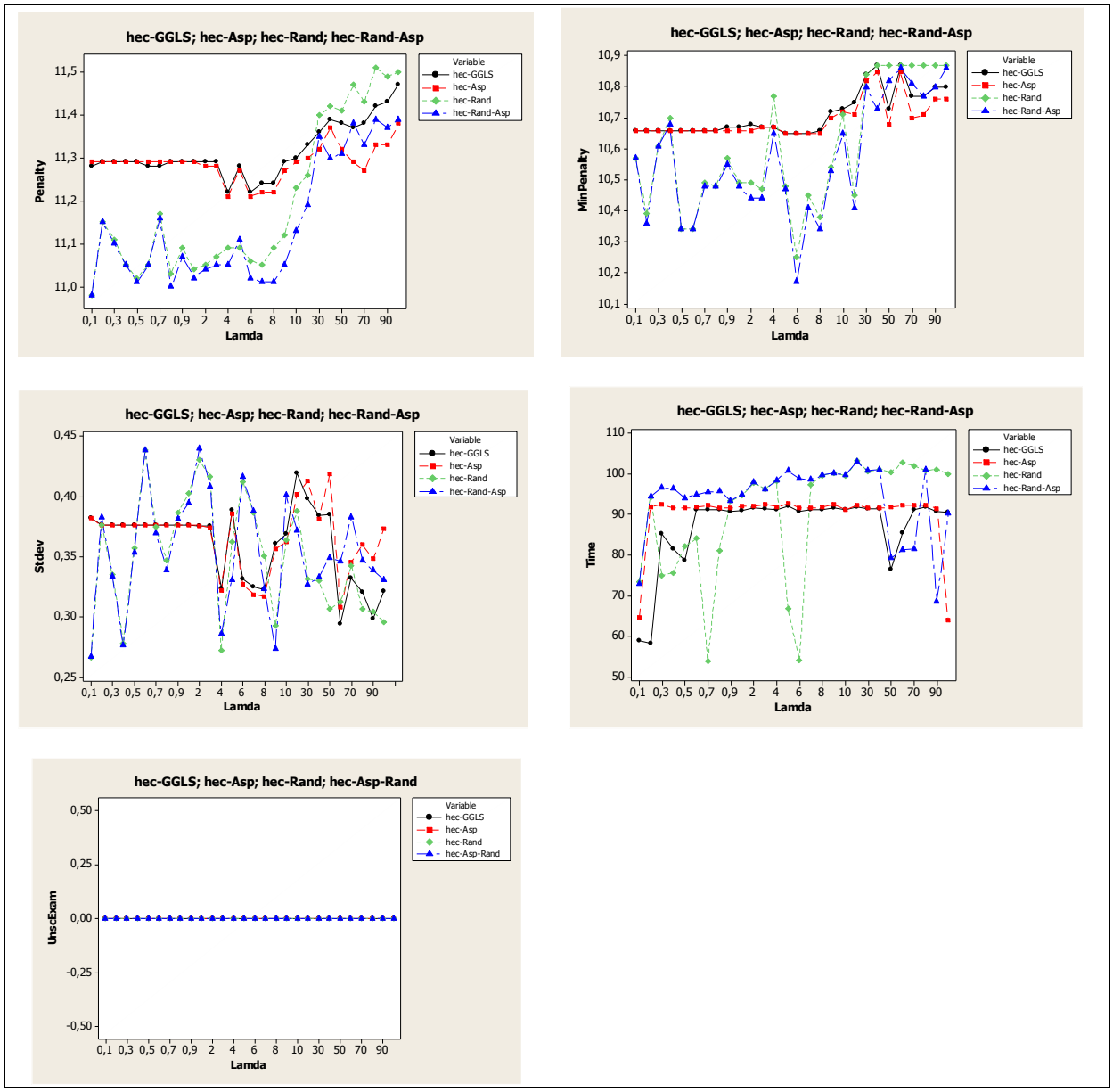


Figure 6. 30 Performances de la GGLS, Asp, Rand et Rand-Asp pour le problème hec

Dans la Figure 6.31, on note pour le problème sta, les observations suivantes :

- Pour la pénalité moyenne, la Rand-Asp est meilleure que la GGLS pour toutes les valeurs de λ. Elle est meilleure que l'Asp pour tous les λ, sauf pour λ=40 où elle est comparable. Elle est meilleurs que la Rand de λ =0.9 à λ=100 et comparable pour le reste.

- Pour la pénalité minimale, la Rand-Asp est meilleure que la GGLS pour toutes les valeurs de λ. Elle est meilleure que l'Asp, sauf de λ =60 à λ=70. Elle meilleure que la Rand pour toutes les valeurs de λ, sauf pour λ=8.
- L'erreur standard affiche une variation proche de celle de la Rand et plus grande que celle d'Asp et GGLS.
- Le temps par est variable dans le même ordre de grandeur.
- Le nombre des examens restants égal à 1, est identique à celui des autres.

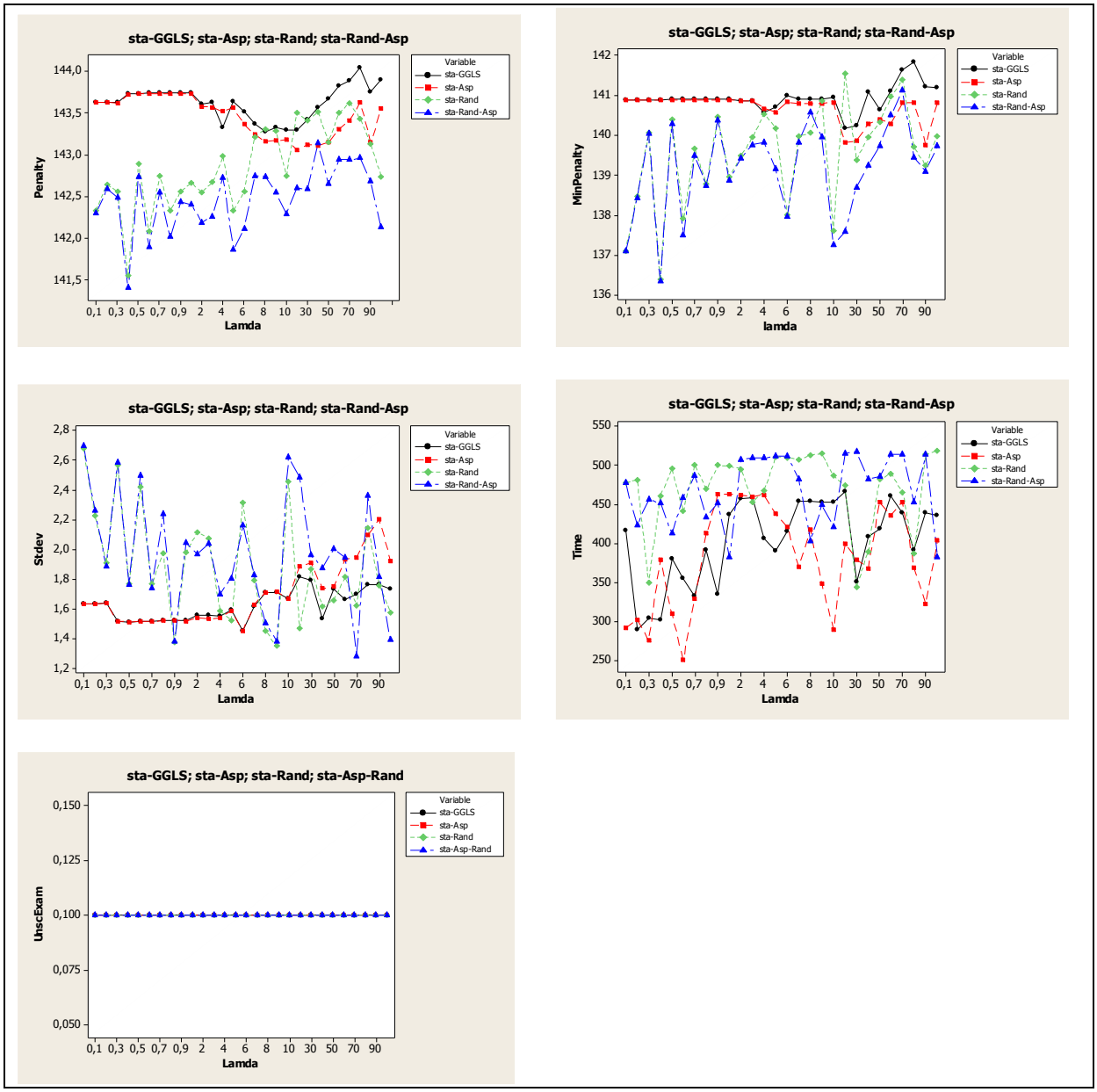


Figure 6. 31 Performances de la GGLS, Asp, Rand et Rand-Asp pour le problème sta

Dans la Figure 6.32 on note pour le problème ear les observations suivantes:

- Pour la pénalité moyenne, la Rand-Asp est meilleure que la GGLS et l'Asp. Elle est faiblement meilleure que la Rand de λ =5 à λ= 10 et comparable pour le reste.

- Pour la pénalité minimale, la Rand-Asp est comparable à la Rand pour toutes les valeurs de λ. Elle est meilleure que l'Asp et la GGLS, sauf pour λ =5 où ces dernières sont meilleures.
- L'erreur standard affiche une variation proche de celle de la Rand et des valeurs inférieures à celles d'Asp et la GGLS.
- Le temps est comparable à celui de la Rand, meilleur que celui de la GGLS et supérieur à celui de l'Asp.
- Le nombre des examens restants est égal à celui de la Rand et inférieur à celui d'Asp et la GGLS.

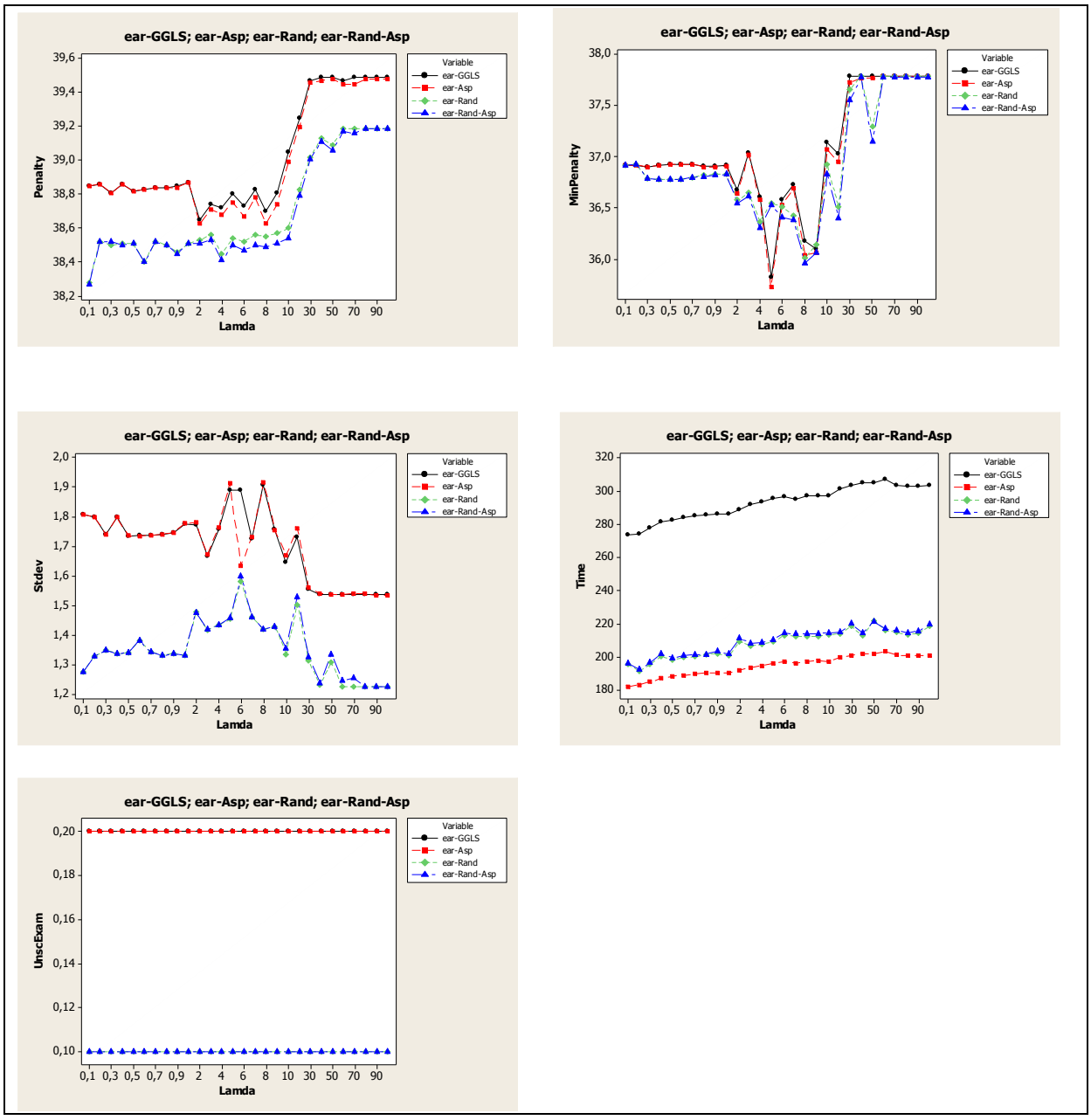


Figure 6. 32 Performances de la GGLS, Asp, Rand et Rand-Asp pour le problème ear

Dans la Figure 6.33 on note pour le problème ute les observations suivantes :

- La pénalité moyenne de la Rand-Asp est meilleure que celles de la GGLS et de l'Asp et comparable à celle de la Rand.
- Pour la pénalité minimale, la Rand-Asp est comparable à la Rand pour toutes les valeurs de λ. Elle est meilleure que l'Asp et la GGLS pour λ =0.1, λ =0.7 et mauvaise de λ =30 à λ= 100. Pour le reste, elle est légèrement plus ou moins meilleure.
- L'erreur standard affiche une variation comparable à celle de la Rand, assez proche de celles de la GGLS et l'Asp , sauf de λ=0.1 à λ=3 où elle affiche des valeurs inférieures.
- Le temps est comparable à celui de la Rand, meilleure que celui de la GGLS et mauvais, comparé à celui de l'Asp.
- Le nombre des examens restants est comparable à celui de la Rand, sauf pour de λ=20 à λ=40 et de λ=50 à λ=80, et inférieure à celui d'Asp et la GGLS.

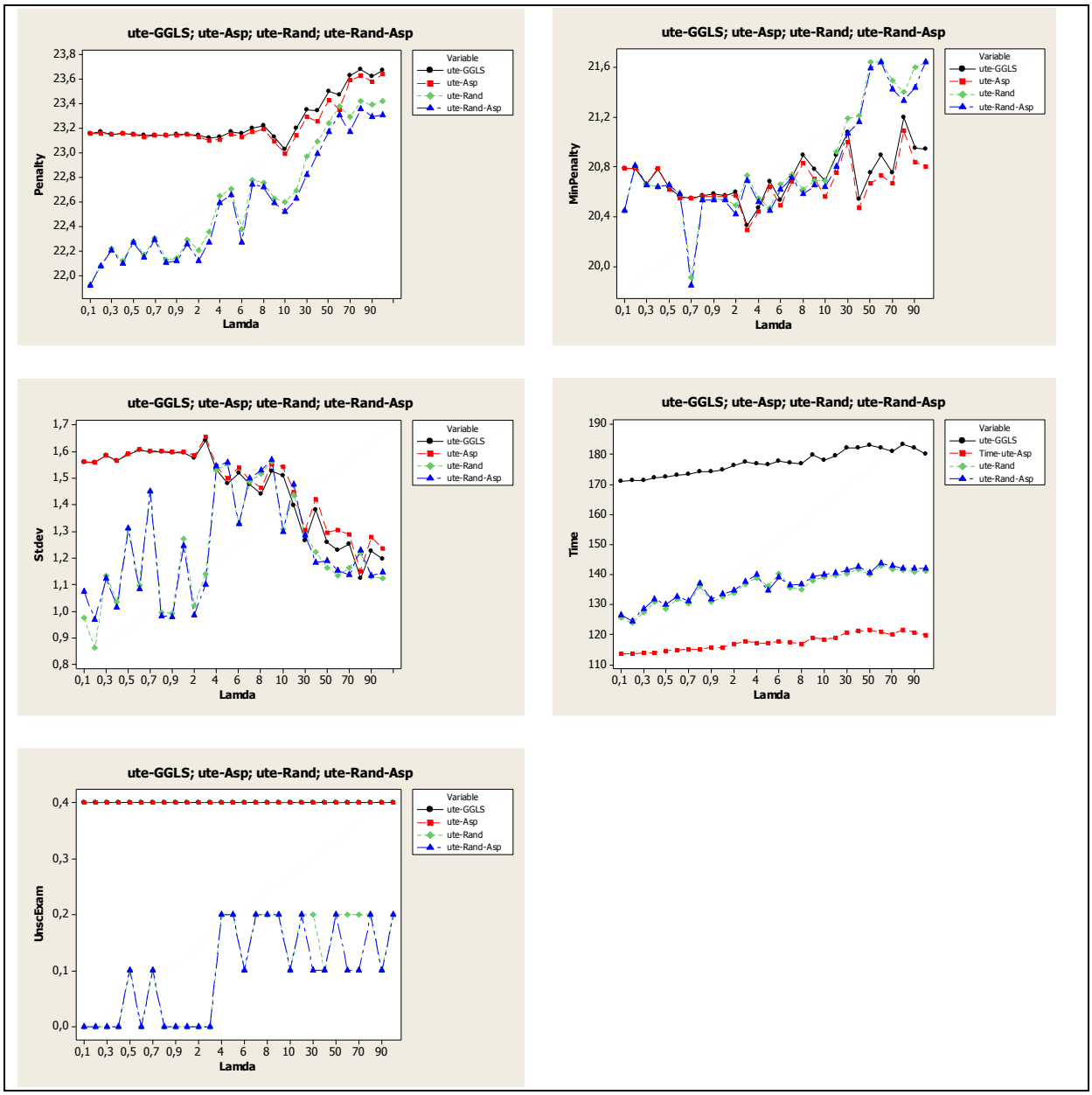


Figure 6. 33 Performances de la GGLS, Asp, Rand et Rand-Asp pour le problème ute

De manière générale, on note que les résultats des pénalités moyennes obtenues par la combinaison de la stratégie d'aspiration et la stratégie des mouvements aléatoires sur les problèmes de test hec sta ear et ute, sont généralement, comparable ou bien meilleurs que ceux obtenus par la GGLS, la Rand et d'Asp, excepté pour le problème hec où l'Asp est légèrement meilleure de $\lambda = 6$ à $\lambda = 100$. On remarque par ailleurs, que dans cette combinaison, l'aspiration réduit l'erreur relative à la stratégie des mouvements aléatoires. Car, son principe consiste à empêcher un mouvement si une meilleure solution est trouvée, afin d'effectuer à sa place, une aspiration.

Pour valider nos résultats, nous avons utilisé le test de Wilcoxon, pour comparer la distribution de la GGLS avec celle de chacune des stratégies Asp, Rand et Rand-Asp , pour chacun des problèmes hec, sta, ear et ute. Le test de Wilcoxon permet de comparer les moyennes de deux échantillons appariés, de distributions inconnues[8], afin de tester si elles proviennent de la même population.

Cette étude comparative montre, pour chacun des problèmes de test hec, sta, ear et ute, que la distribution de chacune des approches Asp, Rand et Rand-Asp, est significativement différente de la distribution de la GGLS et que l'approche Rand-Asp surpasse les autres avec un niveau de confiance de 99%. Par conséquent, les améliorations apportées par chacune d'elles sont bien effectives et non le fruit d'un simple hasard.

Nous pouvons donc conclure, que la combinaison de la stratégie d'aspiration avec celle des mouvements aléatoires, améliore substantiellement les performances de la GGLS et facilite l'opération d'ajustement du paramètre λ pour l'obtention de solutions de bonnes qualités. En effet, d'une part, parce qu'elle permet d'explorer certaines régions intéressantes de l'espace de recherche, ignorées à cause des pénalités, particulièrement, lorsque λ prend de grandes valeurs. Et d'autre part, parce qu'elle fournit un moyen non négligeable permettant d'échapper aux optima locaux, particulièrement, lorsque λ prend de petites valeurs (problèmes avec beaucoup d'optima locaux[9]).

[8] Si la distribution des échantillons est normale, on utilise le test de Student pour deux échantillons appariés.

[9] (Nécessitant un nombre d'itérations important, pour quitter un optimum local).

6.4.6 Résultats sur les benchmark de Carter et al.

Dans cette section, nous présentons les résultats des tests effectués par l'approche Rand-Asp sur les problèmes de Carter et al. (1996). Rappelons que pour les problèmes hec, sta, ear et ute, nous avons obtenus nos résultats avec 10 exécutions différentes et un ensemble initial de taille égale à 220. Pour les problèmes de grandes tailles restants, nous avons effectué 5 exécutions différentes et utilisé un ensemble de taille 50, voir Table 6.14 et Table 6.15.

6.4.7 Meilleures solutions des problèmes

Dans la Table 6.16 a) nous présentons également, pour les problèmes hec, sta ,ear et ute, les pénalités minimales, obtenues par l'approche Rand-Asp , sur 10 exécutions différentes, avec un ensemble initial de taille 220. Alors que dans la Table 6.16 b) nous reportons les pénalités minimales obtenues pour des problèmes restants, obtenues sur 5 exécutions différentes, avec un ensemble initial, de taille 50.

6.4.8 Comparaison des résultats obtenus avec l'état de l'art

Dans la Table 6.17, nous présentons les meilleures solutions obtenues avec la Rand-Asp sur les benchmark de Carter et al., (1996), comparés à ceux de l'état de l'art. Ces résultats sont obtenus avec 10 exécutions différentes pour les problèmes de test : hec, sta,ear ,ute, avec un ensemble initial de taille 220, et 5 exécutions différentes pour le reste des problème, avec un ensemble de taille 50. Les meilleures solutions des problèmes, sur l'ensemble de toutes les approches, sont représentées en caractères gras.

A l'analyse de ces résultats nous constatons que ceux obtenues par la Rand-Asp sont comparables avec plusieurs approches de l'état de l'art et compétitifs avec d'autres. Par conséquent, nous en déduisons que les premières approches de la Recherche Locale Guidée, (à notre connaissance), proposées pour le problème d'emploi du temps des examens, apportent une contribution concrète, permettant de le comprendre davantage et le résoudre.

Prob :yor						
Lamda	**Penaty**	**Min**	**Stdev**	**Time**	**Unsc**	**Min-improv**
0.1	37.02	33.95	2.2793	46.8	0	33.95
1	37.29	33.93	2.1971	47.4	0	33.91
10	38.05	35.10	2.2500	53.2	0	34.75
100	38.89	35.64	2.4487	56.6	0	34.61
Prob : tre						
Lamda	**Penaty**	**Min**	**Stdev**	**Time**	**Unsc**	**Min-improv**
0.1	8.98	8.49	0.3063	419.4	0	8.49
1	8.88	8.48	0.3274	445	0	8.44
10	9.06	8.68	0.2936	504.2	0	8.54
100	9.17	8.68	0.3197	497.6	0	8.543
Prob : kfu						
Lamda	**Penaty**	**Min**	**Stdev**	**Time**	**Unsc**	**Min-improv**
0.1	14.53	14.06	0.4534	3037.6	0	14.06
1	14.60	14.10	0.4729	2960.4	0	14.04
10	14.68	14.33	0.3617	3334.6	0	14.09
100	15.31	14.91	0.4114	3030.6	0.2	13.97
Prob : lse						
Lamda	**Penaty**	**Min**	**Stdev**	**Time**	**Unsc**	**Min-improv**
0.1	11.67	11.33	0.2681	2038.4	0	11.32
1	11.67	11.35	0.2764	2158.4	0	11.21
10	12.19	11.88	0.3134	2407	0	11.46
100	12.25	12.02	0.2677	2333.4	0	11.30

Table 6. 14 Résultats sur des benchmark de Carter et al.,

Prob : car91						
Lamda	**Penaty**	**Min**	**Stdev**	**Time**	**Unsc**	**Min-Improv**
0.1	5.35	5.08	0.2250	9046.4	0	5.07
1	5.35	5.10	0.2053	11208	0	5.05
10	5.51	5.27	0.1897	11915.5	0	5.08
100	5.57	5.31	0.2064	12374.6	0	5.03
Prob : car92						
Lamda	**Penaty**	**Min**	**Stdev**	**Time**	**Unsc**	**Min-mprov**
0.1	4.89	4.67	0.2360	5349	0	4.66
1	4.91	4.64	0.2518	5977.2	0	4.61
10	4.96	4.64	0.2340	6093.3	0	4.60
100	5.05	4.80	0.2337	6478.6	0	4.68
Prob : uta						
Lamda	**Penaty**	**Min**	**Stdev**	**Time**	**Unsc**	**Min-Improv**
0.1	3.60	3.52	0.0528	7774	0	3.52
1	3.61	3.54	0.0483	9358.8	0	3.50
10	3.75	3.68	0.0478	9754	0	3.50
100	3.75	3.68	0.0494	8797	0	3.53

Table 6. 15 Résultats sur des benchmark de Carter et al.,(suite)

6.5 Conclusion

Dans ce chapitre, nous avons exploré deux métaheuristiques pour le problème d'emploi du temps des examens : la première évolutionnaire, basée population, appelée Recherche Dispersée (Scatter Search, SS), tandis que la seconde est locale, dite Recherche Locale Guidée (Guided Local Search, GLS). Dans notre étude, nous avons proposé les premières approches (à notre connaissance) utilisant ces metaheuristiques. Les tests intensifs, effectués sur ces approches, montrent qu'elles sont compétitives et comparables aux approches les plus récentes de l'état de l'art.

Prob	Min-Penalty	λ	Time
Hec	10.17	6	98.8
Sta	136.35	0.4	452.0
ear	35.96[10]	8	214.0
ute	19.85	0.7	131.2

a)

Prob	Min-Penalty	λ	Time
Yor	33.93	1	47
Tre	8.48	1	445
Kfu	14.06	0.1	3038
Lse	11.33	0.1	2038
uta	3.521	0.1	7774
Car91	5.08	0.1	9046
Car92	4.64	1	5977

b)

Table 6. 16 Pénalités minimale sur des benchmark de Carter et al.

[10] L'approche Asp, donne : 35.73 avec lamda=5.

Datasets	**Car91**	**Car92**	**ear83**	**hec92**	**Kfu93**	**Lse91**	**Sta83**	**Tre92**	**Uta92**	**Ute92**	**Yor83**
Rand-Asp	5.08 λ =0.1	4.64 λ =1	35.59 λ =8	10.17 λ =6	14.06 λ =0.1	11.33 λ =0.1	136.35 λ =0.4	8.48 λ =1	3.52 λ =0.1	**19.85** λ =0.7	**33.93** λ =1
(Abdullah et al., 2007a)	5.21	4.36	34.87	10.28	13.46	10.24	159.20	8.13	3.63	**24.21**	36.11
(Abdullah et Burke, 2006)	4.8	4.1	36.00	10.80	15.2	11.9	159.00	8.5	3.60	26.00	36.2
(Asmuni et al., 2005)	5.29	4.56	37.02	11.78	15.81	12.09	160.42	8.67	3.57	27.78	40.66
(Burke et al., 2004b)	4.8	4.2	35.4	10.8	13.7	10.4	159.10	8.3	3.40	25.70	36.7
(Burke et al., 2005)	-	-	45.60	-	-	-	158.20	-	4.52	35.40	-
(Burke et al., 2007)	5.36	4.53	37.92	12.25	15.2	11.33	158.19	8.92	**2.88**	28.01	41.37
(Burke et Newell, 2003)	4.65	4.1	37.05	11.54	13.9	10.82	168.73	8.35	3.20	25.83	37.28
(Burke, et al., 2006)	**4.42**	**3.75**	32.76	10.15	**12.96**	9.83	157.03	7.75	3.06	24.82	**34.84**
(Caramia et al., 2001)	6.6	6	**29.30**	**9.20**	13.8	**9.6**	158.20	9.4	3.50	24.40	36.2
(Carter, et al., 1996)	7.1	6.2	36.40	10.80	14.0	10.5.	161.50	9.6	3.50	25.80	41.7
(Casey et Thompson, 2002)	5.4	4.4	34.80	10.80	14.1	14.7	**134.90**	8.7	-	25.40	37.5
(Côte et al.,2005)	5.4	4.2	34.20	10.40	14.3	11.3	157.00	8.6	3.50	25.30	36.4
(DiGaspero et Schaerf, 2001)	6.2	5.2	45.70	12.40	18	15.5	160.80	10.0	4.20	29.00	41.0
(Eley, 2006)	5.7	4.8	36.80	11.30	15.0	12.1	157.20	8.8	3.80	27.70	39.6
(Kendall et Mohd Hussin, 2005)	4.93	4.40	38.14	10.74	15.55	12.35	157.38	8.35	3.77	27.12	37.99
(Merlot et al., 2003)	5.1	4.3	35.10	10.60	13.5	10.5	157.30	8.4	3.50	25.10	37.4
Petrovic et al., (2003)	4.50	3.93	33.75	10.83	13.82	10.63	165.27	**7.92**	3.14	25.33	36.53
(white et al., 2004)	5.73	4.63	45.80	12.90	17.1	14.7	158.00	8.94	4.44	29.00	42.3
(Yang et Petrovic, 2005)	4.5	3.93	33.70	10.83	13.82	10.35	151.15	**7.92**	3.14	25.39	36.35

Table 6. 17 Comparaison des pénalités minimales de l'approche Rand-Asp avec l'état de l'art

Conclusion

Le problème d'emploi du temps consiste à affecter un ensemble d'événement à un nombre fini de périodes de temps tout en respectant un ensemble de contraintes. Il est parmi les problèmes les plus influents dans la gestion de certains événements, tels que les employés, les enseignements, les examens, les compétitions, les transports etc. La nature fortement combinatoire de ce problème l'inclut dans la classe des problèmes NP-difficiles.

Dans ce travail, nous avons décrit un modèle pour une catégorie de problèmes d'emploi du temps : le problème d'emploi du temps des examens. Nous avons détaillé ensuite notre démarche d'investigation pour l'élaboration, pour la première fois (à notre connaissance), de deux nouvelles catégories d'approches de l'intelligence artificielle pour le résoudre. La première évolutionnaire, basée sur la Recherche Dispersée (Scatter search, SS) (Glover, 1977), alors que la seconde est de type local, basée sur la Recherche Locale Guidée (Guided Local Search, GLS) (Voudouris, 1997).

La Recherche Dispersée (Scatter Search, SS)

Dans nos investigations sur la SS pour le problème d'emploi du temps des examens considéré, nous avons commencé par la comparaison de l'impact des heuristiques : Random, LWD, LD et SD, utilisées pour la construction des solutions initiales dans une approche basique de la Recherche Dispersée. Les résultats obtenus par les tests empiriques et la comparaison de la distribution de l'approche basée sur la SD avec chacune des autres approches en utilisant le test non paramétrique de Mann-Whitney nous révèlent que dans l'ensemble, l'heuristique SD est meilleure, comparée aux heuristiques Random, LWD, et LD en termes de pénalité moyenne et du nombre moyen d'examens restants. Ceci peut être expliqué principalement, par son caractère adaptatif. A l'étape suivante, nous avons menés sur l'approche utilisant la SD, notée SD-SS, une série de tests intensifs, pour ajuster ses paramètres et choisir une méthode de mise à jour de l'ensemble de référence et une méthode de combinaison permettant de faire évoluer le processus de recherche et d'établir un équilibre entre la diversification et l'intensification, afin de générer des solutions élites.

Au cours de nos expérimentations, nous avons remarqué que la qualité de la solution du problème peut être améliorée, soit par l'accroissement de la taille de la population, soit par l'augmentation du nombre d'itérations. Ceci se traduit évidemment, par un temps de traitement important, notamment pour les problèmes de grandes tailles. Comme notre objectif est de trouver des

emplois du temps où d'une part, tous les examens sont affectés à des périodes valides, et d'autre part la fonction objectif représentant la pénalité des contraintes souhaitables est minimisée, nous avons choisi de recourir à l'élaboration de stratégies ad hoc, pour l'amélioration des solutions initiales afin de consacrer le reste du temps à l'opération de raffinage. Trois approches différentes de la SS ont été proposées et testées pour atteindre cet objectif : L'approche LWDSD-SS basée sur une hybridation de la SS avec les heuristiques LWD et SD, L'approche G-SD-SS basée sur l'hybridation de la SS avec une Grasp, et enfin, l'approche G-LWDSD-SS combinant les deux idées précédentes

A l'issue de tests empiriques intensifs, nous avons pu conclure que dans la LWDSD-S, la prise en considération du degré de conflits pondéré maximal avec les autres examens, dans les premiers stades de construction des solutions initiale, n'améliore pas, réellement, la qualité des résultats finaux. Pour le cas de la G-SD-SS, nous avons par contre observé que le caractère, adaptatif rajouté, joue un rôle non négligeable dans l'amélioration des performances. Pour la dernière approche dite G-LWDSD-SS, nous avons trouvé que la combinaison des deux idées précédentes améliore substantiellement les résultats finaux, non seulement en termes de qualité, mais aussi en termes de temps.

Pour valider et confirmer ces améliorations, nous avons utilisé le test de Mann-Whitney pour comparer les distributions de nos approches deux à deux. Les résultats par ce test, révèle que la distribution des solutions de l'approche SD-SS n'est pas significativement différente de l'approche LWDSD-SS, comparées avec un niveau de confiance de 99%. Ceci prouve que l'hybridation des heuristiques seule, n'apporte pas une réelle amélioration significative. De même, l'approche G-LWDSD-SS n'est pas significativement différente de l'approche G-SD-SS. Ceci est dû sans doute à leur commun caractère adaptatif. A contrario, pour le couple : (SD-SS, G-SD-SS), nous avons confirmé que, la configuration G-SD-SS 50/10/7 surpasse significativement la configuration SD-SS 200/10/10, comparées avec un niveau de confiance de 95%. Il en est également de même, pour le couple: (SD-SS, G-LWDSD-SS). Ceci peut être expliqué, par le fait, que le caractère adaptatif ne soit pas commun, aux deux approches de chaque couple. Par conséquent, nous pouvons dire que les améliorations apportées par la G-SD-SS et la G-LWDSD-SS à l'approche initiale, SD-SS, sont bien concrètes et non dues à un pur hasard. En outre, toutes nos approches peuvent être utilisées pour traiter le problème d'emploi du temps des examens considéré. Néanmoins, la G-LWDSD-SS et la G-SD-SS sont les meilleures, et améliorent significativement, les résultats des performances de l'approche initiale SD-SS, aussi bien en termes de qualité que de temps.

Enfin, à la dernière étape, une comparaison avec l'état de l'art, des meilleurs résultats obtenus par nos approches (G-SD-SS et G-LWDSD-SS), montre qu'elles sont non seulement comparables, mais aussi compétitives dans bien des cas. D'ailleurs, il est important de souligner qu'aucune des approches de l'état de l'art ne surpasse les autres, pour tous les problèmes de test considérés.

La Recherche Locale Guidée (Guided Local Search, GLS).

Dans nos investigations sur la GLS pour le problème d'emploi des examens considéré, nous avons commencé par étudier une première variante basique de la GLS, démarrant avec une solution initiale aléatoire. Les tests, effectués sur un petit échantillon de problèmes de Carter et al.,(1996), restituent des solutions qui s'améliorent très lentement, au regard du temps imparti. En outre, le nombre des examens restants est généralement, non nul.

Pour remédier à cela et tenter d'atteindre des solutions élites, nous avons choisi d'améliorer la solution initiale par une méthode déterministe, afin de donner à la GLS, le plus de temps pour l'opération de raffinage. Car avec une solution aléatoire, nous somme en face de deux problèmes : celui de trouver une période valide pour chaque examen, et celui de raffiner la valeur de la fonction objectif représentant les violations des contraintes souhaitables. Trois variantes, hybridant une Grasp basée sur des heuristiques de coloration de graphes avec la Recherche Locale Guidée, sont donc, étudiées et comparées.

Dans la variante qui a produit les meilleurs résultats finaux par rapport aux autres, une solution initiale est construite de manière adaptative, avec une contribution de 25% de LWD et 75% de SD. En premier lieu, les 25% des examens ayant le plus grand nombre de conflits pondérés sont affectés chacun à une éventuelle période valide. Les examens restants sont ensuite traités par la SD adaptative, permettant de tenir compte des changements antérieurs de la solution en construction. Ainsi, plus de temps est consacré au raffinage méthodique de la solution initiale, afin de programmer tous les examens d'une part, et de réduire les pénalités au minimum possible, d'autre part. L'approche de la Recherche Locale Guidée basée sur cette variante est désignée par GGLS.

Les tests que nous avons effectués avec la GGLS montrent, que les résultats s'améliorent au fur et à mesure que le nombre d'itérations du processus de recherche augmente. Néanmoins, cette amélioration s'avère encore lente, et justifie bien notre choix précédent de générer une bonne solution initiale afin d'utiliser le reste du temps pour son raffinage. Nous remarquons par ailleurs

également, que les résultats restitués par la GGLS, sont suivant les problèmes, plus ou moins sensitifs aux variations du paramètre λ. Ce phénomène apparaît notamment, dans le cas où la valeur de λ est grande; augmentant l'influence des pénalités dans la fonction objectif modifiée. Les petites valeurs de λ en revanche, réduisent l'effet des pénalités et diminuent la probabilité d'avoir ces situations.

Par ailleurs, comme la GGLS est une approche basée sur une recherche locale, nous lui avons introduit deux stratégies qui se sont révélées utiles dans le cas la Recherche Tabou, et le recuit simulé etc.

En comparant les résultats de la GGLS avec ceux de l'Asp, une approche de la GGLS utilisant une stratégie d'aspiration, nous constatons que le critère d'aspiration ne détériore généralement pas les performances de la GGLS. Au contraire, il les améliore dans bien des cas. Cette amélioration est due principalement, au fait que l'aspiration n'est appliquée, que lorsqu'il existe un mouvement générant pour la fonction d'origine g, une solution de meilleure qualité que la meilleure trouvée jusque là, et qui n'aurait pas été prise, si on tenait compte des pénalités. L'aspiration permet donc à la GGLS, d'éviter d'ignorer des solutions intéressantes à causes des pénalités auxquelles elles sont assujetties. Ceci apparait particulièrement utile, pour les problèmes hec et sta, où elle a permis d'améliorer les performances de la GGLS pour les grandes valeurs de λ, ayant une plus grande influence sur les pénalités.

L'aspiration est en conclusion une stratégie utile permettant d'améliorer la GGLS, car elle augmente, l'ensemble des valeurs de λ produisant de bons résultats et réduit par voie de conséquence, la sensitivité de l'approche à ce paramètre.

Dans le cas de la stratégie de mouvements aléatoires permettant de s'affranchir de l'évaluation du voisinage, les résultats obtenus montrent aussi, que l'approche avec mouvements aléatoires, notée Rand, améliore les performances de la GGLS ; notamment, pour les petites valeurs de λ, où le processus de recherche nécessite un temps important pour quitter un optimum local. Nous pouvons déduire que l'espace de recherche de nos problèmes comprend beaucoup d'optima locaux et que les mouvements aléatoires contribuent à diversifier la recherche en l'aidant à s'en échapper. Lorsque λ prend des valeurs plus grandes, nous observons que ces mouvements pourraient engendrer une légère détérioration pour certains problèmes. Une explication à cela, est que les mouvements aléatoires ne sont pas dans ces problèmes, aussi efficaces que les pénalités gérées méthodiquement.

En conclusion, on peut dire que la stratégie des mouvements aléatoire peut contribuer à améliorer les performances de la GGLS, pour des problèmes ayant beaucoup d'optima locaux, car, elle fournit un moyen non négligeable de diversification de la recherche, particulièrement, lorsque λ prend de petites valeurs.

Enfin, à la dernière étape, nous avons étudié dans une approche de la GGLS notée Rand-Asp, la combinaison des deux stratégies précédentes. Les résultats obtenus sont alors, plus intéressants encore, dans la mesure où cette combinaison réduit l'erreur relative à la stratégie des mouvements aléatoires. En effet, elle empêche un mouvement, si une meilleure solution est trouvée, et effectue à la place une aspiration. Nous en déduisons par conséquent, que la combinaison de la stratégie d'aspiration avec celle des mouvements aléatoires améliore substantiellement, les performances de la GGLS et facilite l'opération d'ajustement du paramètre λ, en augmentant le nombre de valeurs permettant d'atteindre les meilleurs résultats. D'une part, parce qu'elle permet d'explorer certaines régions intéressantes de l'espace de recherche ignorées à cause des pénalités, particulièrement, avec les λ de grandes valeurs. Et d'autre part, parce qu'elle fournit un moyen non négligeable permettant d'échapper à des optima locaux ; particulièrement, lorsque λ prend de petites valeurs nécessitant un temps excessif.

Pour valider nos résultats, nous avons effectué une étude comparative des distributions de nos différentes approches de la Recherche Locale Guidée. Cette étude basée sur le test de Wilcoxon, montre pour chacun des problèmes de test, que la distribution de chacune des stratégies Asp, Rand et Rand-Asp, surpasse significativement celle de la GGLS avec un niveau de confiance de 95%. Ceci confirme donc, que les améliorations, apportées par chacune d'elles, sont bien effectives et non le résultat d'un pur hasard aussi.

Enfin, une étude comparative sur les benchmarks de Carter et al.,(1996) avec l'état de l'art montre, que nos meilleures solutions obtenues par l'approche combinant la stratégie d'aspiration et celle des mouvements aléatoires sont, comme dans le cas de la Recherche Dispersée, et tout comme la plupart des autres approches, comparables pour certains problèmes et compétitives pour d'autres.

Perspectives

En perspective, pour La Recherche Dispersée, nous envisageons d'étudier d'autres stratégies de combinaison et de recherche locale, et de paralléliser l'approche pour tenter de réduire le temps de

traitement. Dans le cas de La Recherche Locale Guidée, l'introduction d'autres stratégies d'aspiration pourrait conduire à une exploration plus profonde de l'espace de recherche, pour atteindre des solutions élites. De même, l'introduction d'autres stratégies de mouvements aléatoires pourrait s'avérer utile à la diversification de la méthode. Nous pensons par ailleurs également, que l'hybridation de la GLS avec une métaheuristique basée population, comme la Recherche Dispersée, pourrait aussi conduire à des résultats plus que prometteurs, vu leurs caractères complémentaires dans l'exploration de l'espace de recherche. Enfin, concernant le problème d'emploi du temps des examens, nous envisageons d'adopter nos approches, dans un contexte multiobjectif, pour des problèmes ayant des contraintes impératives et souhaitables différentes.

Bibliographie

(Aamodt et Plaza 1994) A. Aamodt, and E. Plaza, "Case-based reasoning: foundational issues, methodological variations and system approaches", AI Communications, Vol. 7, No. 1, pp.39-59, 1994.

(Ahmadi et al.,2003) S. Ahmadi, R. Barone, P. Cheng, P. Cowling, and B. McCollum, "Perturbation based variable neighbourhood search in heuristic space for examination timetabling problem", In Proceedings of multidisciplinary international scheduling: theory and applications (MISTA 2003), pp.155-171, Nottingham, 13-16, ISBN: 0-9545821-2-8, 2003, August, 2003.

(Ahuja et a.l 2001) R.K. Ahuja, J.B. Orlin, and D. Sharma, "Multi-exchange neighbourhood search algorithm for capacitated minimum spanning tree problem", Mathematical Programming, Vol. 91, pp. 71-97, 2000.

(Amin 1999) A. Amin. "Simulated Jumping", In Annals of Operations Research Vol. 86, pp. 23-38, 1999.

(Aarts et Lenstra 1997) E. Aarts and J. K. Lenstra "Local search in combinatorial optimisation", Chichester: John Wiley and Sons, 1997.

(Abdullah et Burke 2006) S. Abdullah, and E.K . Burke, "A multi-start very large neighborhood search approach with local search methods for examination timetabling", In: Long, D., Smith, S.F., Borrajo, D. and McCluskey, L. (Eds.): Proc. The International Conference on Automated Planning and Scheduling (ICAPS 2006), pp.334-337, Cumbria, UK , 2006.

(Abdullah et Burke 2007a) S. Abdullah, S. Ahmadi, E.K. Burke and M. Dror," Investigating Ahuja-Orlin's large neighbourhood search approach for examination timetabling", OR Spectrum, Vol. 29, No. 2, 351-372, 2007a.

(Abdullah et Ahmadi 2007b) S. Abdullah, S. Ahmadi, E.K. Burke and M. Dror and B. McCollum, "A tabu based large neighbourhood search methodology for the capacitated examination timetabling problem", Journal of Operational Research Society, Vol. 58, No.11 2, pp. 1494-1502, 2007b.

(Abramson 1991) D. Abramson, "Constructing school timetables using simulated annealing: sequential and parallel algorithms," Management Science, Vol. 37, No.1, pp. 98-113, 1991.

(Abramson et al.,1999) D. A. Abramson, H. Dang, and M. Krisnamoorthy, "Simulated annealing cooling schedules for the school timetabling problem," Asia-Pacific Journal of Operational Research, Vol. 16, pp. 1-22, 1999.

(Appleby et al.,1960) Blake J. S. Appleby, D. V. Blake, and E. A. Newman, "Techniques for producing school timetables on a computer and their application to other scheduling problems", The Computer Journal, Vol. 3, pp. 237-245, 1960.

(Arani et Lotfi 1989) T. Arani and V. Lotfi., "A three phased approach to final exam scheduling", IIE Trans., Vol. 21, No. 1, 86–96,1989.

(Asmuni et al.,2005) H. Asmuni, E.K. Burke, J.M. Garibaldi and B. McCollum, "Fuzzy multiple ordering criteria for examination timetabling", In Burke, E.K. and Trick, M. (Eds.): Proc.Practice and Theory of Automated Timetabling (PATAT V 2004), Lecture Notes in Computer Science, Springer-Verlag, Vol. 3616, pp.334–353, Pittsburgh, USA, 2005.

(Awad et Chinneck 2004) R. Awad, and J.Chinneck, "Proctor assignment at Carleton University", In: Interfaces Vol. 28, No. 2, pp. 58-71. 1998.

(Naji Azimi 2004) Z. Naji Azimi, "Comparison of metaheuristic algorithms for examination timetabling problem", Applied Mathematics and Computation, Vol. 16, No. 1-2, pp.337–354, 2004.

(Binato et Hery 2001) S. Binato, W.J. Hery, D. Loewenstern, and M.G.C Resende, "A greedy randomised adaptive search procedure for job shop scheduling," In Hansen, P. and Ribeiro, C.C. (Eds.): Essays and Surveys on Meta-Heuristics, pp.58–78, Kluwer Academic Publishers, Boston, 2001.

(Boizumault et al.,1996) P. Boizumault, Y. Delon and L. Peridy, "Constraint logic programming for examination imetabling", Journal of Logic Programming, Vol. 26, No. 2, pp. 217–233, 1996.

(Boughaci et al., 2008) D. Boughaci, B. Benhamou, H. Drias, "Scatter Search and Genetic Algorithms for MAX-SAT problems", Journal of Mathematical Modelling and Algorithms JMMA, Springer Netherlands, February, Vol. 7, pp. 101-124, 2008.

(Boughaci et al., 2007) D. Boughaci, B. Benhamou, H. Drias, "solving Max-SAT problems using a Scatter Search Metaheuristics", In Proceeddings of the COSI'07, pp.167-178,11-13 juin 2007.

(Boughaci et Drias 2005) D. Boughaci, and H. Drias, "Efficient and Experimental Meta-heuristics for MAX-SAT Problems", WEA 2005, LNCS Vol. 3503, 501-512, Springer-Verlag Berlin Heidelberg, 2005.

(Brelaz 1979) D. Brelaz, "New methods to colour the vertices of a graph," Communications of the Association for Computing Machinery, Vol. 22, No. 4, pp. 251–256, 1979.

(Brailsford et al.,1999) S. C. Brailsford, C. N. Potts and B. M. Smith, "Constraint satisfaction problems: Algorithms and applications", European Journal of Operational Research, Vol. 119, pp 557-581 1999.

(Burkeet al.,2004a) E.K. Burke, Y. Bykov, J.P. Newall and S Petrovic, " A time-predefined local search approach to exam timetabling problems", IIE Transactions on Operations Engineering, Vol. 36, No. 6, pp. 509–528 , 2004a.

(Burke et Bykov 2006a) E.K. Burke, Y. Bykov, " Solving exam timetabling problems with the flex-deluge algorithm", In: Proceedings of Practice and Theory of Automated Timetabling Conference (PATAT 2006), 30th August–1st September 2006, Brno, Czech , 2006a.

(Burke et al.,1994) E. K Burke, D. G. Elliman, et R. F. Weare, "A genetic algorithm based university timetabling system", In: 2nd East-West International Conference on Computer Technologies in Education, Vol. 1, pp. 35-40, 1994.

(Burke et al., 1995) E. K Burke, D. G. Elliman, and R. F. Weare, "A hybrid genetic algorithm for highly constrained timetabling problems", In: Proceedings of the 6th International Conference on Genetic Algorithms, pp. 605-610, 1995.

(Burke et Erben 2001) E. K., Burke, and W. Erben, (Eds.), Lecture notes in computer science: Vol. 2079. Practice and theory of automated timetabling III: selected papers from the 3rd international conference. Berlin: Springer. ISBN: 3-540-42421-0, 2001.

(Burke et Landa Silva 2004a) E. K. Burke, and J. D. Landa Silva, "The design of memetic algorithms for scheduling and timetabling problems", In W. E. Hart, N. Krasnogor, and J. E. Smith

(Eds.), Studies in fuzziness and soft computing: Vol. 166. Recent advances in memetic algorithms and related search technologies, pp. 289–312, Berlin: Springer, 2004a.

(Burke et Newall 2003a) E.K. Burke and J.P Newall, " Enhancing timetable solutions with local search methods", In: Burke, E.K. and Causmaecker, P.D. (Eds.): Proc. Practice and Theory of Automated Timetabling (PATAT 2002), Lecture Notes in Computer Science, Vol. 2740, pp.195–206, Springer-Verlag, Gent, Belgium , 2003a.

(Burke et Newall 2004b) E.K. Burke and J.P Newall, "Solving examination timetabling problems through adaption of heuristic ordering", Annals of Operations Research, Vol. 129, No. 2, pp. 107–,134, 2004b.

(Burke et Newall 1999) E. K., Burke, and J. P. Newall, "A multi-stage evolutionary algorithm for the timetable problem", IEEE Transactions on Evolutionary Computation, Vol. 3, No.1, pp. 63–74, 1999.

(Burke et al.,1996) E. K., Burke, J. P., Newall, et R. F. Weare, " A memetic algorithm for university exam timetabling", In E. K. Burke and P. Ross (Eds.), Lecture notes in computer science: Vol. 1153. Practice and theory of automated timetabling I: selected papers from the 1st international conference (pp. 241–250). Berlin: Springer.

(Burke et al., 1996) E. K. Burke, K. Jackson, J. H. Kingston, and R. Weare, "Automated university timetabling", the state of the art. The Computer Journal, Vol. 40, No. 9, pp. 565–571, 1996.

(Burke 2001) E.K. Burke, Y. Bykov and S. Petrovic, "A multicriteria approach to examination timetabling", In: Burke, E.K. and Erben, W. (Eds.): Proc. Practice and Theory of Automated Timetabling (PATAT 2000), Lecture Notes in Computer Science, Vol. 2079, pp.118–131, Springer-Verlag, Constance, Germany, 2001.

(Burke et al.,2005) E.K. Burke, M. Dror, B. McCollum, S. Petrovic, and R. Qu, "Hybrid graph heuristics within a hyper-heuristic approach to exam timetabling problems", In: Golden, B.L., Raghavan, S. and Wasil, E.A. (Eds.): Proc. The Next Wave in Computing, Optimization and Decision Technologie, Conference Vol. of the 9th Informs Computing Society Conference, Vol. 29, No. 1, pp.79–91, Springer, Maryland, 2005.

(Burke et a. 2003b) E.K. Burke, G. Kendall and E. Soubeiga, " A tabu search hyperheuristic for timetabling and rostering", Journal of Heuristics, Vol. 9, No. 6, 451–470, 2003b.

(Burke et al.,2007) E.K. Burke, B. McCollum, A. Meisels, S. Petrovic and R. Qu, " A graph-based hyper heuristic for educational timetabling problems", European Journal of Operational Research, Vol. 176, No.1, pp.177–192 , 2007.

(Burke et al.,1998) E.K., Burke, J. Newall and R.F Weare, " A simple heuristically guided search for the timetable problem", In: Alpaydin, E. and Fyte, C. (Eds.): The International ICSC Symposium on Engineering of Intelligent Systems (EIS'98), pp.574–579, University of La Laguna, ICSC Academic Press, Spain, 1998.

(Burke et al., 2002) E. K. Burke, and S. Petrovic, "Recent research directions in automated timetabling", European Journal of Operational Research Vol. 140, pp. 266–280, 2002.

(Burke et a. 2006b) E.K. Burke, S. Petrovic and R. Qu, "Case based heuristic selection for timetablingproblems", Journal of Scheduling, Vol. 9, No. 2, pp. 15–132, 2006b.

(Burke et al.1996) E. K.Burke, and P. Ross, "Practice and Theory of Automated Timetabling", lecture Notes in Computer Science, Vol. 1153, Springer-Verlag, 1996.

(Caramia et al.,2001) M. Caramia, P. DellOlmo, and G.F. Italiano, "New algorithms for examinations timetabling", In Naher, S. and Wagner, D. (Eds.): Proc. Algorithm Engineering 4th Int Workshop WAE 2000, Lecture Notes in Computer Science, Vol. 1982, pp.230–241, Springer-Verlag, Saarbrucken, Germany , 2001.

(Campos et al.,1999), V.G. Campos, M.Laguna et R. Marti , "Scatter search for the linear ordering problem, in New ideas in optimization", Eds. David Corne, Marco Dorigo and Fred Glover, McGraws-Hill, pp. 331-340. 1999.

(Campos et. al 2001) V.G. Campos, F. Glover, M.Laguna and R. Marti, "A experimentakl evaluation of scatter search for the linear ordering problrm", Journal of Global Optimization, Vol. 21, pp. 397-414, 2001.

(Carrasco et Pato 2001) M.P. Carrasco and M.V. Pato, "A multi-objective genetic algorithm for the class/teacher timetabling problem", The Practice and Theory of Automated Timetabling III: Selected Papers from 3rd International Conference on the Practice and Theory of Automated Timetabling (PATAT III), Konstanz, Germany, Lecture Notes in Computer Science 2079, Springer-Verlag. (Editors: E.K. Burke and W. Erben), pp 3-17, 2001.

(Carter et Johnson 2001) M. W. Carter et D. G. Johnson, "Extended clique initialisation in examination timetabling", Journal of the Operational Research Society (2001), Vol. 52, pp. 538-544, 2001.

(Carter et al.,1996) M.W. Carter, G. Laporte, and S.Y. Lee, "Examination timetabling: algorithmic strategies and applications", Journal of the Operational Research Society, Vol. 47, No. 3, pp. 373–383, 1996.

(Casey et Thompson 2002) S. Casey and J . Thompson, "GRASPing the examination scheduling problem", In: Burke, E.K. and Causmaecker, P.D. (Eds.): Proc. Practice and Theory of Automated Timetabling (PATAT 2002), Lecture Notes in Computer Science, Vol. 2740, pp.232–244, Springer-Verlag, Gent Belgium , 2002.

(Cavique et al.,2001) C. Cavique , C. Rego and I. Themido,"A scatter Search Algorithm for the Maximum Clique Problem", In Ribeiro, CC, Hansen, P. (Eds), Essays and Surveys in Metaheuristics , Kluwer- Academic Publishers, 2001

(Chen et Bushnell 1996) X. Chen and M. L. Bushnell, "Efficient branch and bound search with application to computer-aided design", Kluwer Academic Publishers, 1996.

(Cheng et al.,1996) C. Cheng, L. Kang, N. Leung, and G.M. White, "Investigations of a Constraint Logic Programming Approach to University Timetabling", In: [BR96], pp. 112-129, 1996.

(Christofides 1971) N. Christofides, "An Algorithm for the Chromatic Number of a Graph," The Computer Journal, Vol. 14, pp. 38-39, 1971.

(Cole 1964) A. J. Cole, "The preparation of examination timetables using a smallstore computer", Computer Journal, Vol.7, pp. 117-121, 1964.

(Colorni 1998) A Colorni, M. Dorigo, and V. Maniezzo," Metaheuristic for highschool timetabling", Computational Optimisation and Applications, 9, Kluwer Acad. Publ., Dodrecht, NL, pp. 275-298, 1998.

(Cook 1971) S.A. Cook, " The Complexity of Theorem Proving Procedures," In: Proceedings of the Third ACM Symposium on Theory of Computing, US, pages 151-158, 1971.

(Corne 1994) D. Corne, P. Ross, and H. Fang, "Evolutionary timetabling: Practice, prospects and work in progress", In P. Prosser (Ed.), Proceedings of UK planning and scheduling SIG workshop, 1994.

(Costa et Hertz 1997) D. Costa and A. Hertz, "Ant can colour graphs", Journal of the Operational Research Society, Vol. 48, pp 295-305, 1997.

(Côté et 2005) P. Côté, T. Wong, and R Sabourin, " Application of a hybrid multi-objective evolutionary algorithm to the uncapacitated exam proximity problem", In: Burke, E.K. and Trick, M. (Eds.): Proc. Practice and Theory of Automated Timetabling (PATAT 2004), Lecture Notes in Computer Science, Vol. 3616, pp.294–312, Springer-Verlag, Pittsburgh, USA, 2005.

(Cooper et Kingston 1996) T. B. Cooper, and J.H. Kingston, "The complexity of timetable construction problems", In E. K. Burke (Ed.), Lecture notes in computer science: Vol. 1153. Practice and theory of automated timetabling I: selected papers from the 1st international conference, Berlin: Springer, pp. 283–295, 1996.

(Csima et Gotlieb 1964) J. Csima, et C. C. Gotlieb, "Tests on a computer method for construction school timetables", Communication of the ACM, Vol. 7, No. 3, pp. 160-163, 1964.

(Davenport 1997) A. Davenport, "Extensions and Evaluation of GENET in Constraint Satisfaction", Ph.D. thesis, Department of Computer Science, University of Essex, 1997.

(Daniel Fealko 2006) R. Daniel Fealko, Doctoral Dissertation, Nova Southearstern University, 2006

(Desroches et al., 1978) S. Desroches, G. Laporte and J.M. Rousseau, HOREX , " A computer program for the construction of examination" , INFOR, Vol. 16, No. 3, pp 294-298, 1978.

(De Werra 1985) D. De Werra, " An introduction to timetabling problem", European Journal of Operational Research, Vol. 19, No.2 3, 151–162, 1985.

(De Werra 1997) D. De Werra, "The combinatorics of timetabling", European Journal of Operational Research Vol. 96, pp. 504-513, 1997.

(De Werra et al.,2002) D. De Werra, D., A. S.Asratian et S. Durand, "Complexity of some special types of timetabling problems", Journal of Scheduling, Vol.5, 171–183, 2002.

(Di Gaspero et Schaerf 2001) L. Di Gaspero and A. Schaerf, " Tabu search techniques for examination timetabling", In: Burke, E.K. and Erben, W. (Eds.): Proc. Practice and Theory of Automated Timetabling (PATAT 2000), Lecture Notes in Computer Science, Vol. 2079, pp.104–117, Springer-Verlag, Constance, Germany , 2001.

(Di Gaspero et Schaerf 2003) L. Di Gaspero and A. Schaerf, "Multi-neighbourhood local searchwith application to course timetabling", In: Practice and Theory of Automated Timetabling IV, E. K. Burke and P. De Causmaecker (Eds.), Lecture Notes in Computer Science, Vol. 2740, Springer-Verlag, pp. 262-275, 2003.

(Dorigo 1996) M. Dorigo, V. Maniezzo and A. Colorni, "The ant system: Optimisation by a colony of cooperating agents", IEEE Transactions on Systems, Man, and Cybernetics, Vol. 26, pp 29-41, 1996.

(Dowsland et Thompson 2005) K. A. Dowsland, and J. Thompson, "Ant colony optimisationfor the examination scheduling problem", Journal of Operational Research Society, Vol. 56, 426–438, 2005.

(Drias 2001) H. Drias, "Genetic Algorithm versus Scatter search and solving hard MAX-W-SAT problems", in proc of IWANN'2001, Lectures Notes in Computer Science, LNCS 2084, Springer Verlag, Granada, Spain, Vol. 1, pp. 586-593, 2001.

(Drias et Azi 2001) H. Drias and N. Azi, "Scatter search for SAT and MAX-SAT problems", in proc of the 33th IEEE SSST, Athens, OHIO, USA, (March 2001), pp. 105-109, 2001.

(Drias et Khabzaoui 2001) H. Drias and M. Khabzaoui, "Scatter search with random walk strategy for solving hard MAX-W-SAT problems", in proc of IEA-AIE'2001, Lectures Notes in Computer Science, LNAI 2070, Springer , Budapest Hungary, (June 2001), pp. 35-44, 2001.

(Drias et al.,2009) H. Drias, D. Daoudi, S. Kechid, "Scatter Search Approches For large Scale Information Retrieval", Nabic, 2009, World Congresson Nature and Biologically inspired Computing, Coimbatore India 9-11 December 2009

(Dueck 1993) G. Dueck, New optimisation heuristics, "The great deluge algorithm and the record-to-record travel", Journal of Computational Physics, Vol. 104, pp 86-92, 1993.

(Duong et Lam 2004) T. A. Duong, and K. H. Lam, "Combining constraint programming and simulated annealing on university exam timetabling", In Proceedings of the 2nd international conference in computer sciences, research, innovation and vision for the future (RIVF2004), pp. 205–210, Hanoi, Vietnam, 2–5 February, 2004.

(Easton et al.,2004) K. Easton, G. Nemhauser, and M. Trick, "Sports scheduling", InJ. Leung (Ed.), Handbook of scheduling: algorithms, models, and Performances analysis. Boca Raton: CRC Press, Chap. 52, 2004.

(Eley 2006) M. Eley, " Ant algorithms for the exam timetabling problem", In: Proceedings of Practice and Theory of Automated Timetabling Conference (PATAT 2006), 30th August–1st September, Brno, Czech , 2006.

(Erben 2001) W. Erben, "A grouping genetic algorithm for graph colouring and exam timetabling", In E. K. Burke and W. Erben (Eds.), Lecture notes in computer science: Vol. 2079. Practice and theory of automated timetabling III: selected papers from the 3rd international conference, pp. 132–156, Berlin: Springer, 2001.

(Even et al., 1976) S. Even, A. Itai, A. Shamir. "On the complexity of timetable and multicommodity flow problems". SIAM J Comput , Vol.5, no. 4, pp. 691-703, 1976.

(Faigle et Kern 1992) U. Faigle, W. Kern, "Some convergence results for probabilistic Tabu Search", ORSA Journal on Computing Vol. 4, pp. 32-37, 1992.

(Feo et Resende 1989) T.A. Feo, M.G.C. Resende, "A probabilistic heuristic for a computationally difficult set covering problem", Operations Research Letters, Vol. 8, pp. 67-71, 1989.

(Feo et Resende 1995) T.A. Feo, M.G.C. Resende, "Greedy randomized adaptive search procedures", Journal of Global Optimization Vol. 6, pp. 109-133, 1995.

(Festa et Resende 1995) P. Festa and M.G.C Resende, "GRASP, an annotated bibliography," In: Ribeiro, C.C. and Hanen, P. (Eds.): Proc. Essays and Surveys in Meta-heuristicspp, pp. 325–367, Kluwer Academic Publishers, Dorrecht, 2002.

(Fox 1993) B.L. Fox, "Integrating and accelerating tabu search, simulated annealing, and genetic algorithms", In Annals of Operations Research, Vol. 41, pp. 47-67, 1993.

(Foxley et Lockyer 1998) E. Foxley, and K. Lockyer, "The construction of examination timetables by computer", The Computer Journal, Vol. 11, No. 3, pp. 264-268, 1998.

(Galinier et Hao 1999) P. Galinier, J.K. Hao "Hybrid evolutionary algorithms for graph coloring", Journal of Combinatorial Optimization, Vol. 3, No. 4, pp.379-397,1999.

(Garey et Johnson, 1979) M.R. Garey and D.S. Johnson, Computers and contractibility a Guide to the Theory of NP-Completeness. W.H. Freeman, 1979.

(Garici et Drias 2005) M. Garici and H. Drias, "Cryotanalysis of Substitution Ciphers using Scatter Search", In: Proceedings of International Work Conference on the Interplay between Natural and Artificial Computation, IWINAC(2), Las Palmas, canary Islands Spain, pp. 31-40, 2005.

(Glover 1994) F. Glover, "Algorithms and Scatter Search: Unsuspected Potentials", Statistcs and Computing, Vol. 4, pp. 131-140, 1994.

(Glover 1977) F. Glover, "Heuristics for integer programming using Surrogate constraints", Decision Sciences, Vol. 1, pp. 156-166. 1977.

(Glover et Laguna 1997) F. Glover and M. Laguna, Tabu search. Kluwer Academic Publisher, ISBN 0-7923-8187-4, 1997.

(Glover 1998) F. Glover, "A Template for scatter Search and path relinking, in Artificial Evolution", Lecture Notes in Computer Sciences 1363, J.,K.,Hao, E. Lutton, E. Ronald, M.Shoenauer and D. Snyers (Eds), Springer, pp. 3-61, 1998.

(Glover et al.,2000) F. Glover, M. Laguna and R. Marti, " Fundamentals of Scatter Search and Path Relinking", Control and Cybernetics, Vol. 29, No. 3, pp. 653-684 , 2000.

(Glover et al.,2003)F. Glover, M. Laguna and R. Marti, "Scatter Search and Path Relinking: Advances and Applications", In: Glover F. and Kochenberger G. (Ed), Handbook of Meta-Heuristics, pp. 1-36, Kluwer, 2003.

(Glover 1989) F. Glover, "Tabu search Part I", Operations Research Society of America (ORSA), Journal on Computing, Vol. 1, pp. 109-206, 1989.

(Goldberg 1989) D.E. Goldberg, "Genetic algorithms in search, optimisation and machine learning", Addison Wesley, 1989.

(Gordeev 1989) E. Gordeev. "Decision Problems and their Solutions", Cybernetics - Unlimited Possibilities and Possible Limits, USSR Academy of Science, 1989, 5-48 (in Russian), 1989.

(Guyette et al.,1994) Guyette, Hamidian, and Tuazon, "A Rule-Based Expert System Approach to Class Scheduling", Computer Electrical Engineering, Vol. 20, No.2, pp. 151-162, 1994.

(Hadjidj et Drias 2010) D. Hadjidj and H. Drias, "A hybrid Grasp and Scatter Search for the Examination Timetabling Problem", In proc. of ICMWI'2010, International Conference on Machine and Web Intelligence, IEEEXplore, pp. 297-302, Algiers october 3-5 , 2010.

(Hadjidj et Drias 2010) D. Hadjidj and H. Drias, " Grasp and Guided Local Search for the Examination Timetabling Problem", Int.J. Artificial Intelligence and Soft Computing, Vol. 2, No. (1/2), pp.103-114, Inderscience Entreprises Ltd, 2010.

(Hadjidj et Drias 2008) D. Hadjidj and H. Drias, "Scatter Search and Graph Coloring Heuristics for the Examination Timetabling Problem", Int. Arab Journal of Information Technology, Vol. 5, No. 4, pp. 334-340, 2008.

(Hadjidj et Drias 2007) D. Hadjidj and H. Drias, "Investiguating a Scatter Search Approch for the Examination Timetabling Problem", In proc. of JNAM'2007, Journee Nationale sur les Applications des Mtaheuristiques, May 29,2007.

(Hadjidj et Drias 2007) D. Hadjidj and H. Drias, "Guided Local Search for the Examination Timetabling Problem", In proc. of COSI'2007 , Conference sur l'Optimisation et les Systemes d'Informations, pp. 589-599, Oran, June 21-23, 2007.

(Hadjidj et Drias 2005) D. Hadjidj and H. Drias, " Scatter Search for the Examination Timetabling Problem", In proc. of CIIA'05 , Conference Internationale en Informatique Appliquée, pp. 217-223, Bourdj bou Arreridj Algerie,19-21, 2005.

(Hamiez and Hao 2001) J.P.Hamiez,J.K. Hao, "Scatter search Graph Coloring" , in P. collet et al., editors, Vol. 2310 of Lncs, pp.168-179, Spring verlag, Germany, 2001

(Hao et Galinier 1999) J.K.Hao, P. Galinier, M. Habib, "Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes", Revue d'intelligence artificielle, 1999.

(Hertz et al, 1987) A. Hertz and D. de Werra. "Using tabu search techniques for graph coloring", Computing, Vol. 39, pp. 345–351, 1987.

(Holland 1975) J.H. Holland, "Adaptation in natural and artificial systems", Ann Arbor: University of Michigan Press, 1975.

(Johnson 1974) D.S. Johnson, "Fast Algorithms for Bin Packing", Journal of Computer and System Science, Vol. 8, No. 3, pp. 272-314, 1974

(Johnson et al. 1991) D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon, "Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning", Operations Research, Vol. 39, no. 3, pp.378–406, 1991.

(Kang et White 1992) L. Kang and G.M. White, "A logic approach to a resolution of constraints in timetabling", European Journal of Operational Research, Vol.61, pp 306-317, 1992.

(Karp 1972) R.M. Karp, "Reducibility among combinatorial problems", In: Miller, R.E. and Thatcher, W. Eds. Complexity of Computer Computations, pp.85–103, Plenum Press, New York , 1972.

(Kendall et Mohd Hussin 2005) G. Kendall, and N. Mohd Hussin, "An investigation of a tabu search based hyperheuristic for examination timetabling", In: Kendall, G., Burke, E. and Petrovic, S. (Eds.): Multidisciplinary Scheduling; Theory and Applications, 1st International Conference MISTA'03, pp.309–328, University Press, Nottingham, UK , 2005.

(Kennedy et Eberhart 1995) J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization", Proceedings of the 1995 IEEE International Conference on Neural Networks 1942-1948. Piscataway, NJ: IEEE Service Center, 1995.

(Kennedy et Eberhart 2001) J. Kennedy and R. C. Eberhart, Swarm Intelligence, (Denise E.M. Penrose), San Francisco: Morgan Kaufmann Publishers, 2001.

(Kilby et al.,1999) P. Kilby, P. Prosser, et P. Shaw, "Guided Local Search for the Vehicle Routing Problem with time windows", In Meta-heuristics: Advances and trends in local search paradigms for optimization, S. Voß, S. Martello, I. Osman, and C. Roucairol, Eds. Kluwer Academic, pp. 473–486, 1999.

(Kingston 2004) J. H. Kingston, "A tiling algorithm for high school timetabling", In: Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling, E. K. Burke and M. Trick (Eds.) pp 233-250, 2004.

(Kirkpatrick et al.,1983) S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimisation by simulated annealing", Science, Vol. 220, pp. 671-380, 1983.

(Kolodner et Leake 1996) J.L. Kolodner, and D. D. Leake, "A tutorial introduction to case-based reasoning", In: LEAKE, D. (ed.) Case-Based Reasoning: Experiences, Lessons, and Future Directions, pp. 31 – 65. AAAI Press/The MIT Press, 1996.

(Kong et Kwok, 1999) S.C. Kong and L.F. Kwok, "A Conceptual Model of Knowledgebased Timetabling System", Knowledge-Based Systems, Vol. 12, pp. 81-93, 1999.

(Kwan 2004) R. Kwan, "Bus and train driver scheduling", In J. Leung (Ed.), Handbook of scheduling: algorithms, models, and performance, Boca Ratom: CRC Press, Chap. 51, 2004.

(Lajos 1996) G. Lajos, Complete university modular timetabling using constraint logic programming, 1996.

(Lawler et Wood 1966) E. L. Lawler and D. E. Wood, "Branch-and-bound methods: A survey", Operations Research, Vol.14, pp 699-719, 1966.

(Leong et Yeong 1987) T.Y. Leong and W.Y. Yeon, " Examination Scheduling a Quadratic Assignment Perspective", In: roceeding of the International Conference on Optimization Techniques and Applications, Ballarat, Australia, pp. 550-558, 1987.

(Lin et Kernighan 1973) S. Lin, B.W. Kernighan, "An efficient heuristic for the traveling-salesman problem", Operations Research Vol. 21, pp. 498-516, 1973.

(Lin 2002) S.L.M. Lin, "A broker algorithm for timetabling problem," In: E.K. Burke and P. De ausmaecker (eds), (2002), Proceedings of the 4th International Conference on Practice and Theory of Automated Timetabling, 21st-23rd August 2002, KaHo St.-Lieven, Gent, Belgium, pp. 372-386, 2002.

(Marti et al.,2006)R. Marti, M. Laguna and F. Glover, " Principales of scatter search", European Journal of operational research, Vol. 169, No.2, pp. 359-372, 2006.

(Luna et al.,2006) F. Luna, A.J. Nebro, B. Dorronsoro, E.Abla, P. Bouvry, and L. Hgie, "Optimal Broadcasting in Metropolitan MANET's using Multiobjective Scatter Search", in EvoCOMNET 3rd European Workshop on Evolutionary Computation in Communication Network and Connect Systems, Budapest, Hungray, April, 2006

(Mavridou et al.,1998) T. Mavridou, P.M. Pardalos, L.S. Pitsoulis, and M.G.C. Resende, " A GRASP for the biquadratic assignment problem", European J. Operational Research, Vol.105, No.3, pp. 613–621, 1998.

(Merlot et al.,2003) L.T.G. Merlot, N. Boland, B.D. Hughes, and P.J. Stuckey, "A hybrid algorithm for examination timetabling problem", In: Burke, E.K. and Causmaecker, P.D. (Eds.): Proc. Practice and Theory of Automated Timetabling (PATAT 2002), Lecture Notes in Computer Science, Vol. 2740, pp.207–231, Springer-Verlag, Gent, Belgium, 2003.

(Metropolis et al.,1953) N. Metropolis, A. W Rosenbluth., M. N.Rosenbluth, , A. H. Teller and E. Teller, " Equation of state calculations by fast computing machines", Journal of Chemical Physics, Vol. 21, No. 6, pp.1087-1092, 1953.

(Miles 1975) R. Miles, "Computer timetabling: a bibliography", British Journal of Educational Technology, Vol. 6, No. 3, pp. 16–20, 1975.

(Mladenovi et Hansen 1997) N. Mladenovi´c, and P. Hansen, "Variable neighbourhood search", Computers and Operations Research, Vol. 24, No. 11, pp. 1097–1100, 1997.

(Mills et Tsang 2000) P. Mills, and E.Tsang, "Guided local search for solving SAT and weighted MAX-SAT Problems", In SAT2000, I. Gent, H. van Maaren, and T. Walsh, Eds. IOS Press, pp. 89–106, 2000.

(Mills et Tsang 2003) P. Mills, and E.Tsang, "Applying an Extended Guided local search to the quadratic assignment problem", Annals of operations research, Vol. 118, No. 1-4, pp.121-135, 2003.

(Moscato 1999) P. Moscato, "Memetic algorithms: A short introduction. In: New Ideas in Optimisation", D. Corne, M. Dorigo and F. Glover (Eds.), McGraw Hill, pp. 219-234, 1999.

(Pacquete et Fonseca 2001) L. Pacquete and C.M. Fonseca, "A study of examination timetabling with multi-objective evolutionary algorithm", The Proceedings of the 4th Meta-heuristics International Conference (MIC 2001), pp 149-154, 2001.

(Papadimitriou et Steiglitz 1982) C.H. Papadimitriou, K. Steiglitz, "Combinatorial optimization – algorithms and complexity", Prentice Hall, 1982.

(Petrovic et Burke 2004) S. Petrovic, and E. K. Burke, "University timetabling", In J. Leung (Ed.), Handbook of scheduling: algorithms, models, and performance analysis. Boca Raton: CRC Press. Chap. 45, 2004.

(Petrovic et al.,2003) S. Petrovic, Y. Yang, and M. Dror, " Case-based initialisation for examination timetabling", In Proceedings of the 1st Multidisciplinary Intl. Conf. on Scheduling: Theory and Applications (MISTA 2003), 13–16 August, Nottingham, UK , 2003.

(Puchinger et Raidl 2005) J. Puchinger and G. R. Raidl, "Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification", In: Proceedings of the First International Work-Conference on the Interplay Between Natural and Artificial Computation. Vol. 3562 of Lecture notes in computer science, Springer-Verlag, Berlin, Heidelberg, pp 41-53, 2005.

(Resende et Pitsoulis 2000) M.G.C. Resende, and L.S. Pitsoulis, "Greedy randomized adaptive search procedures", In: Pardalos, P.M. and Resende, M. (Eds.): Handbook of Applied Optimization, pp.168–183, University Press, Oxford , 2000.

(Rhydian 2008) L. Rhydian, "A survey of meta-heuristic-based techniques for University Timetabling problems", OR Spectrum, Vol. 30, No.1, pp.167–190 , 2008.

(Rong et Burke 2005) Q. Rong and E.K. Burke, "Hybrid variable neighbourhood hyper-heuristics for exam timetabling problems", Electronic Proceedings of the 6th Meta-heuristics International Conference (MIC 05), Vienna, Austria, 2005.

(Rong 2009) Q. Rong, E.K. Burke, B. McCollum, L.T. Merlot, and S.Y Lee, "A survey of search methodologies and automated system development for examination timetabling", Journal of Scheduling, Vol. 12, No.1, pp. 55–89, 2009.

(Ross 1998) P. Ross, E. Hart, and D. Corne, "Some observations about GA based exam timetabling", In E. K. Burke and M. W. Carter (Eds.), Lecture notes in computer science: Vol. 1408. Practice and theoryof automated timetabling II: selected papers from the 2nd international conference, Berlin: Springer, pp. 115–129, 1998.

(Ross 2003) P. Ross, E. Hart, and D. Corne, "Genetic algorithms and timetabling", In A. Ghosh and S. Tsutsui (Eds.), Advances in evolutionary computing: theory and applications, NewYork: Springer, pp. 755–771, 2003.

(Ross 2004) P. Ross, J. G. Marin-Blazquez, and E. Hart, "Hyper-heuristics applied to class and exam timetabling problems", In Proceedings of the 2004 congress on evolutionary computation (CEC2004), Washington: IEEE Press, pp. 1691–1698, 2004.

(Schaerf 1999) A. Schaerf, "A survey of automated timetabling", Artificial Intelligent Review, Vol.13, No.2, pp. 87–127, 1999.

(Schmidt et Strohlein 1979) E. A. Schmidt, and T. Strohlein, "Timetable construction—an annotated bibliography", The Computer Journal, Vol.23, pp.307–316,1979.

(Selman et al.,1994) A. Selman, H. Kautz, and B. Cohen, "Noise Strategies for Improving Local Search", In Proceedings AAAI-94, pp. 337-343, 1994.

(Sierksma et al.,1996) G. Sierksma, G. A. Tijssen and P. Van Dam, "Linear and Integer Programming", Theory and Practice. Marcel Dekker Publishers, Inc. New York, 1996.

(Terashima-Marin et al.,1999) H. Terashima-Marin, P. Ross, and M. Valenzuela-Rendon, "Evolution of constraint satisfaction strategies in examination timetabling", In Proceedings of the genetic and evolutionary conference, Orlando, FL, pp. 635–642, 1999.

(Thompson et Dowsland 1998) J. Thompson, and K. Dowsland, "A robust simulated annealing based examination timetabling system", Computers and Operations Research, Vol. 25, No.7, 637–648, 1998.

(Tripathy 1984) A. Tripathy, "School Timetabling - A case in large binary integer linear programming", Management Science, Vol. 30, pp 1473-1489, 1984.

(Ulder et al.,1991) N.L.J. Ulder, E.H.L. Aarts, H.J. Bandelt, P.J.M. van Laarhoven, E. Pesch, "Genetic local search algorithms for the traveling salesman problem", Lecture Notes in Computer Science 496, 1991.

(Voudouris 1997) C. Voudouris, "Guided Local Search for Combinatorial Optimisation Problems", Ph.D. thesis, Department of Computer Science, University of Essex, 1997.

(Voudouris 1999) C. Voudouris, and E. Tsang, "Guided local search", Europ. J. Oper. Res. Vol. 113, No.2, 469-499, 1999.

(Welsh et Powell 1967) D.J.A. Welsh, and M.B. Powell, "The upper bound for the chromatic number of a graph and its application to timetabling problems", The Computer Journal, Vol.11, pp. 41-47, 1967.

(White et Chan 1979) G.M. White and P.W. Chan, "Towards the construction of optimal examination schedules", INFOR, Vol.17, No.3, pp 219-229, 1979.

(White et Haddad 1983) G.M. White and M. Haddad, "A heuristic method for optimising examination timetabling schedules which have day and night course", Computers and Education, Vol.7, pp 235-238, 1983.

(White et al.,2004) G.M. White, B.S. Xie, and S. Zonjic, " Using tabu search with longer-term memory and relaxation to create examination timetables", European Journal of Operational Research, Vol.153, No.16, pp. 80–91, 2004.

(White et Zhang 1998) G.M. White and J. Zhang, "Generating complete university timetables by combining tabu search with constraint logic", The Practice and Theory of Automated Timetabling II: Selected Papers from 2nd International Conference on the Practice and Theory of Automated Timetabling (PATAT II), Toronto, Canada, Lecture Notes in Computer Science 1408, Springer-Verlag. (Editors: E.K. Burke and M. Carter), pp. 187-198, 1998.

(Wren 1996) A.Wren, Scheduling, "timetabling and rostering a special relationship?", In E. K. Burke and P. Ross (Eds.), Lecture notes in computer science: Vol. 1153. Practice and theory of automated timetabling I: selected papers from the 1st international conference, Berlin: Springer, pp. 46–75, 1996.

(Yang et Petrovic 2005) Y. Yang, and S. Petrovic, "A novel similarity measure for heuristic selection in examination timetabling", In: Burke, E.K. and Trick, M. (Eds.): Proc. Practice and Theory of Automated Timetabling (PATAT 2004), Lecture Notes in Computer Science, Vol. 3616, pp.377–396, Springer-Verlag, Pittsburgh, USA, 2005.