

La production avec machines parallèles

Comme nous l'avons vu dans le chapitre précédent, l'un des décisions au niveau opérationnel qui a probablement des impacts sur les performances du laboratoire d'analyses physico-chimiques est la gestion opérationnelle des échantillons. Ces décisions deviennent plus complexes lorsque dans certaines étapes de l'analyse disposent d'un ensemble de machines identiques disposées parallèlement [24], [81], [67]. Cela implique dans un premier temps définir l'affectation des tâches aux machines, et ensuite, définir leur ordre de traitement. A cela, les conditions du cas étudié imposent des dates de disponibilité des tâches (r_i) différentes et une fonction objectif mesurant le retard causé ($\min \sum T_i$).

A notre connaissance, la seule publication qui résout le problème $P_m/r_i/\sum T_i$ est l'article de Yalaoui [140]. Dans cet article, plusieurs méthodes heuristiques ont été testées afin de résoudre le problème décrit. Une méthode appelée *ABHG* est signalée comme dominant dans la minimisation du retard, avec des temps de calcul acceptables. Également, dans cet article la génération d'instances est un sujet étudié. Les conclusions issues de cette étude sont reprises dans le chapitre 3.

Malgré le nombre réduit de publications concernant le problème $P_m/r_i/\sum T_i$, la suite de cette section présente des références traitant des problèmes proches, où des conclusions ou des méthodes intéressantes sont proposées.

Le problème où une seule machine est disponible est souvent étudié : Du et Leung [43] établissent que le problème est NP-difficile avec l'objectif de la minimisation du retard total. Une étude bibliographique à propos du problème avec une seule machine disponible a été présentée en 2009 par Koulamas [67]. Récemment, Gafarov *et al.* [49] ont développé un algorithme pseudo-polynomiale pour la minimisation du retard total, avec l'utilisation d'une seule machine. En 2013 Zhou et Liu [143] ont conclu que pour des problèmes structurés tels que l'ordonnancement de tâches avec la minimisation du retard total sur une seule machine, les méthodes à décomposition ont des performances comparables à celles des métaheuristiques souvent utilisées dans la résolution de ce type de problème combinatoire, en termes

de l'utilisation de mémoire et temps de calcul.

Le problème avec deux ou plusieurs machines en parallèle est traité dans certaines références comme un problème intéressant d'ordonnancement. Koulamas en 1994 [65] a réalisé une étude à propos du problème général de la minimisation de critères liés aux retards des tâches. Grâce aux conclusions avancées par l'auteur, on peut établir que le problème $P_m/r_i/\sum T_i$ est NP-difficile, car le problème sans les dates d'arrivée a été prouvé NP-difficile. Dans ce même article, l'auteur présente l'heuristique *KPM* pour le problème de machines parallèles mais avec des tâches disponibles dès l'instant 0 (les dates d'arrivée sont égales à 0). D'autres développements sont présentés par Jouglet et Savourey [61] pour la résolution du problème avec machines parallèles, et la minimisation du retard total pondéré.

Le problème de minimisation du makespan ($\sum C_i$) est souvent traité dans la littérature. Persma et Dijk [102] traitent le problème d'ordonnancement de l'atelier avec des machines indépendantes et minimisation du makespan. En 2011, Lin *et al.* [83] traitent un problème similaire, avec deux fonctions objectifs alternatives. En 2012, Dell'Amico *et al.* [6] comparent plusieurs méthodes de la littérature sur des instances connues du problème avec plusieurs machines en parallèle, et minimisation du makespan.

Parmi les problèmes d'ordonnancement à machines parallèles, certains ont été montrés comme ayant des méthodes de résolution exactes nécessitant de temps de calcul d'ordre polynomial par rapport à la taille des instances. Ainsi, certains de ces problèmes sont : le problème d'assignation ($Q/p_i = p; r_i/C_{max}$) [68], le problème à machines uniformes et minimisation de la somme des dates de fin ($Q_m/p_i = p; r_i/\sum C_i$) [39], le problème avec machines indépendantes et minimisation de la somme des dates de fin ($R/\sum C_i$) [28] [57], le problème avec fenêtres de temps et machines identiques ($P/p_i = p; r_i/\sum w_i C_i$), entre autres.

Certains cas où toutes les tâches ont des temps de traitement identiques ont été montrés comme ayant des temps de résolution polynomial par rapport à la taille de l'instance [50], [129]. Dans les cas traités dans ce mémoire, même si certaines tâches sont similaires, elles sont caractérisées par des temps de traitement différents, compris dans une plage de valeurs définie par le domaine d'activité. Pour les développements théoriques associés au chapitres 3, une plage de valeurs est pris dans la littérature (voir sections des résultats expérimentaux chapitres 3 et 4).

Par rapport à l'impact des fonctions objectif utilisées dans la difficulté dans la prise de décisions au niveau opérationnel, des travaux théoriques permettent d'extraire des conclusions pour des ateliers avec plusieurs machines en parallèle. En fait, tel qu'il est décrit par Lawler *et al.* [78] et Brucker [27] (voir figure 2.1) certaines fonctions objectif sont des généralisations d'autres fonctions objectif. Ainsi, dans le cas de la minimisation du retard total, cette

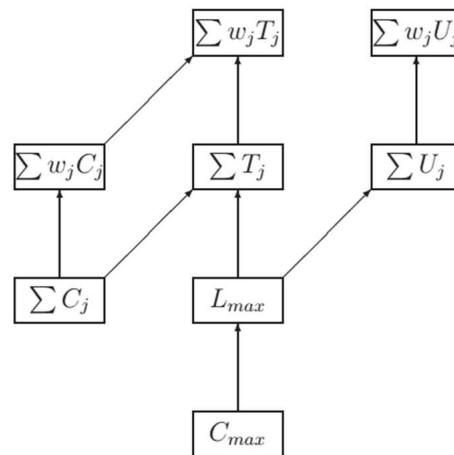


FIGURE 2.1 – Réduction des fonctions objectives [27] [78]

fonction objectif correspond à la généralisation des fonctions de minimisation du makespan et de la somme des dates de fin.

D'autres développements ont été réalisés pour des problèmes avec des fonctions objectif alternatives : Cao *et al.* [30] proposent pour un problème similaire, une méthode heuristique intéressante où des méthodes optimales sont associées à une recherche taboue avec l'objectif de minimiser une expression du retard accompagnée d'une notion économique due à l'installation des équipements. Le problème d'ordonnancement des machines parallèles avec des dates de fin souhaitée communes est traité dans [73] où des pénalités sont considérées pour le retard et l'avance, afin de proposer des systèmes de flux tiré ou «Just-in-time». Dans [69], les auteurs montrent comment l'automatisation des sites de production et l'implantation des systèmes d'aide à la décision, couplés aux systèmes informatiques au cœur de l'automatisation, atteignent le potentiel maximum des algorithmes de résolution de problèmes combinatoires.

D'autres interprétations du concept de retard total peuvent être trouvées dans certaines publications. Dans [117], le retard est considéré comme le temps passé après la date de fin souhaitée sans accumuler le temps d'attente avant le traitement d'une tâche déjà en retard. Cela peut être utilisé dans les cas de coût de stockage réduits. En 2012, Wang *et al.* [136] traitent un problème d'ordonnancement de machines indépendantes avec l'objectif de minimiser le nombre de tâches tardives et du makespan. Des méthodes de recherche locale ont été adaptées par les auteurs.

En 1999, Akker *et al.* [3] et Chen et Powel [33] ont considéré des méthodes de résolution

par génération de colonnes pour des problèmes d'ordonnancement avec la minimisation des fonctions objectifs additives (relatives aux dates de fin obtenues).

2.2 La production par lots dans les ateliers à cheminement unique

Dans le chapitre 1 nous avons vu le laboratoire d'analyses physico-chimiques comme étant une structure de recherche avec un fonctionnement similaire à un atelier de production composé de plusieurs opérations. Chacune des opérations est réalisée par un ou plusieurs équipements exclusivement dédiés à l'opération. La composition des lots de traitement est autorisée et souhaitable. Nous adressons ce problème comme $FHm, ((PM^{(k)})_{k=1}^m) / r_i, block / \sum T_i$, ou «Flow Shop Hybride».

Malgré le fait que pour ce problème, à notre connaissance, aucune référence ne propose des méthodes de résolution, cette section présente des références qui traitent des problèmes similaires au problème décrit.

2.2.1 Le Flow shop avec ressources uniques

Plusieurs études ont été réalisées pour le problème d'ordonnancement dans l'atelier à cheminement unique et une seule machine par opération. La principale fonction objectif considérée dans ces références est liée à la date de fin réelle des tâches sur la dernière opération de l'atelier. Ainsi, le makespan apparaît comme l'objectif le plus souvent étudié [96]. A propos de cette fonction objectif, l'article de Johnson [60] est considéré comme la première référence où le problème à cheminement unique («Flow Shop») et minimisation du makespan est traité. Pan et Ruiz en 2013 [96] ont effectué un résumé des études effectuées sur ce même problème. La minimisation des critères liés au retard des tâches ont apparu récemment pour problèmes avec des configurations d'atelier à cheminement unique. En 2008, Vallada *et al.* [132] présentent une étude bibliographique des références traitant le problème d'ordonnancement de l'atelier à cheminement unique et minimisation du retard total. Certaines des méthodes incluses par les auteurs ont été initialement créées pour la minimisation du makespan.

L'ordonnancement des ateliers à cheminement unique génère deux types de plannings : avec permutation et sans permutation. Une définition approfondie de cette différence est présentée dans le chapitre 4. Dans [96] les méthodes référencées proposent dans la plupart

des ordres de traitement avec permutation. Dans un article récent [89] des méthodes sans permutation sont présentées pour ce même problème. Si nous considérons N le nombre de tâches ou produits à ordonnancer, les ordonnancement avec permutation ont un indice de complexité de $N!$, tandis que pour ceux sans permutation est de $(N!)^S$, avec S le nombre d'opérations considérées.

2.2.2 Le Flow shop hybride

Lorsque l'atelier est composé d'opérations où deux ou plusieurs équipements sont disponibles, l'atelier est dit hybride ou flexible *HFSP* («Hybird Flow Shop Problem»). Ce problème a été prouvé fortement NP-difficile, à partir du cas avec deux opérations par Jaintinder et Gupta [55]. Plusieurs références ont effectué des études bibliographiques sur le problème *HFSP* : en 1999, Linn et Zhang [86] présentent un état de l'art où ils remarquent la différence en terme de la complexité entre les ateliers avec une seule ressource par opération, et les *HFSP*. En 2010 deux études bibliographiques sont apparues. Ribas *et al.* [106] classifient les références les plus pertinentes par les restrictions considérées pour les tâches et machines. Ruiz et Vasquez [109] présentent un ensemble de chiffres clés par rapport à la composition de la littérature à propos du problème *HFSP*. D'après les auteurs, seulement 6% des publications de l'ordonnancement des ateliers à cheminement unique traitent l'objectif de la minimisation du retard total.

D'autres problèmes d'ordonnancement d'ateliers à cheminement unique sont aussi traités avec des contraintes plus spécifiques : Tian et Cheng [128] minimisent le makespan dans le cas où l'information concernant les caractéristiques des tâches à traiter n'est disponible qu'au moment de son arrivée à l'atelier.

2.2.3 Le Flow shop avec production par lots

Le problème d'ordonnancement dans l'atelier du type *HFSP*, où le traitement par lots fait partie des décisions à considérer est, à notre connaissance, peu étudié. Tian et Cheng [128] proposent une série coefficients pour le calcul des bornes inférieures pour des algorithmes dédiés à la résolution de problème on-line (information indisponible avant l'arrivée de la tâche à l'atelier). Bellanger et Oulamara [13] incluent dans leur article une analyse des performances dans le pire cas d'une série d'heuristiques adaptées au problème *HFSP* avec deux opérations, dont une traitant les tâches par lots. Un problème similaire est traité par Xian et Tang [127] avec la minimisation du makespan. Sung et Min [120], et Chang et Young

[121] traitent des problèmes avec deux opérations, où au moins une traite les tâches par lots, et l'objectif est lié au retard total.

La suite de ce chapitre présente les méthodes le plus souvent utilisées dans la résolution des problèmes combinatoires similaires aux problèmes décrits précédemment. Les méthodes sont présentées dans deux sections séparées pour les méthodes exactes et les méthodes approchées.

2.3 Les méthodes de résolution exactes

Les méthodes exactes sont proposées pour résoudre de manière optimale les problèmes combinatoires. L'efficacité de ces méthodes dépend, entre autres facteurs, de la structure du problème. Nous incluons dans cette section la programmation linéaire et les méthodes par séparation et évaluation («Branch-and-Bound»).

2.3.1 Programmation linéaire

Un programme linéaire est un problème d'optimisation qui inclut les éléments suivants : les variables de décision, la fonction objectif et les contraintes. Les variables de décision sont typiquement représentées par un vecteur symbolisé par X . Une combinaison linéaire de ces variables associées à des coûts (ou profits) représente la fonction objectif. Cette fonction est accompagnée d'un sens d'optimisation (minimisation, maximisation). Les contraintes sont aussi des combinaisons linéaires des variables de décision, qui imposent des limites et représentent la structure du problème traité. Une solution réalisable est, par conséquent, toute solution, qui respecte les conditions exprimées par les contraintes et qui a un coût (ou profit) calculé avec la fonction objectif. Le type de variable de décision utilisé définit la programmation linéaire à employer. Ainsi, différents types de modèles de programmation linéaire peuvent surgir : des modèles de programmation linéaire binaire (variables binaires), de programmation linéaire en nombres entiers (variables entières) ou de programmation linéaire mixte (variables entières et réelles) ou *MIP* («Mixed Integer Problem»), entre autres.

La méthode de résolution la plus populaire pour résoudre programmes linéaires est l'algorithme simplex, lequel a été utilisé depuis sa création dans nombreux domaines [36]. Cette méthode et des améliorations sont utilisées dans des solveurs spécialisés tel que CPLEX, XPRESS, GAMS entre autres.

Le problème $P_m/r_i/\sum T_i$ est modélisé avec une modèle linéaire dans [140], où deux différents versions sont présentées. Unlu et Mason [130] comparent différents programmes

linéaires dans la résolution du problème avec machines parallèles. Des objectifs relatifs aux retards et aux tâches tardives sont étudiés dans [2], où la programmation linéaire et la génération de colonnes sont utilisées dans un problème avec machines en parallèle.

Un cas spécifique de minimisation du makespan et machines en parallèle, avec des temps de traitement identiques a été résolu dans un temps polynomial par Baptiste *et al.* [11] avec la programmation linéaire.

Dans Liao et Huang [84] 3 modèles de programmation linéaire sont proposées et analysées pour déterminer leur efficacité dans la résolution du problème d'ordonnancement dans l'atelier de type Flow shop sans permutation. Un problème avec deux opérations, où le traitement par lots est autorisé dans la deuxième opération, est modélisé et résolu avec un programme linéaire dans [137].

Tel qu'il est décrit par la plupart des auteurs, la méthode de programmation linéaire est limitée à résoudre des instances de petite taille, très souvent loin des instances réelles trouvées dans les milieux industriels.

2.3.2 Les méthodes de séparation et évaluation

L'énumération complète, qui correspond à tester toutes les solutions possibles pour un problème donné, est une méthode inexploitable en termes de l'utilisation de la mémoire et le temps d'exécution. Une exploration intelligente permet de réduire ces besoins, en évitant l'exploration des solutions inefficaces dès sa racine. C'est de cette manière que les méthodes de séparation et évaluation sont construites.

Les composants indispensables de ce type de méthodes sont une procédure pour la séparation, et une pour l'évaluation. La première procédure doit garantir l'exploration de toutes les solutions possibles, mais au même temps, elles doivent permettre une identification de solutions partielles, afin d'appliquer des procédures d'évaluation. Ces deuxièmes réduisent le nombre de solutions explorées en supprimant des solutions partielles qui conduisent à des solutions inefficaces. Les procédures d'évaluation doivent être mathématiquement justifiées. Ces méthodes sont aussi connues sous le nom de «Branch-and-Bound» ($B\&B$). Parmi les procédures d'évaluation les bornes inférieures sont souvent utilisées. Ces bornes calculent une estimation de la valeur de la fonction objectif qui dépasse normalement sa valeur optimale, mais qui n'est pas réalisable.

Des méthodes du type Branch-and-Bound ont été utilisées dans la résolution de problèmes proches des problèmes contenus dans ce mémoire.

La somme de retards total est minimisée pour le problème avec une seule machine dans [124] avec des méthodes de décomposition, inclus dans une procédure du type $B\mathcal{E}B$. Ces méthodes de décomposition utilisent les règles établies par Emmons [45]. En 2007 Szwarc [123] a proposé des nouvelles adaptations des règles de décomposition de Lawler [76] pour la résolution du problème avec une seule machine.

La contrainte des dates de disponibilité différentes a été étudié par Su et Chen en 2008 [119] afin de proposer une méthode de type $B\mathcal{E}B$. En 2013 Zhou et Liu [143] ont conclu que pour des problèmes structurés tel que la minimisation du retard avec une seule machine, les méthodes à décomposition hybridées à des $B\mathcal{E}B$, ont des performances comparables à celles des métaheuristiques souvent utilisées dans la résolution de ce type de problème combinatoire, en terme des temps de calcul.

Azizoglu et Kirka[9], Yalaoui et Chu [141], Shim et Kim [116] et Tanaka et Araki [126] proposent des méthodes exactes toutes à partir de la méthode «Branch and Bound». En 2009 Schaller [114] a publié des nouvelles bornes inférieures, améliorant ainsi celles décrites par Shim et Kim [116]. Chacune de ces méthodes est décrite comme plus efficace que les précédentes.

Santos *et al.* [111] et Hidri et Haouari [56] proposent une série de bornes inférieures pour différents versions du problème $HFSP$. Bellanger et Oulamara [13] proposent des bornes inférieures pour le $HFSP$, où une des opérations permet le traitement par lots.

2.4 Les méthodes approchées

Cette section présente quelques méthodes approchées adaptées à la résolution des problèmes combinatoires comme ceux traités dans ce mémoire. La caractéristique commune à ces méthodes, c'est qu'elles ne garantissent pas l'optimalité de la solution trouvée. Malgré cela, ces méthodes ont souvent des performances très bonnes pour la résolution de problèmes pour lesquels les méthodes exactes ne peuvent pas trouver des solutions optimales dans un intervalle de temps acceptable.

A la différence des méthodes exactes, les méthodes approchées ne sont pas limitées par la taille des instances à résoudre, facteur essentiel lors de leur implémentation dans des utilisations industrielles réelles.

La suite de cette section présente quelques méthodes approchées du type métaheuristique, pour la résolution des problèmes combinatoires proches de ceux étudiés dans cette thèse.

Les recherches locales et taboue

La recherche locale et la recherche tabou sont utilisées dans l'exploration des voisinages créés par des procédures qui garantissent des recherches intensifiées. Pour le cas de la recherche taboue, initialement présentée par Glover [53], des procédures de diversification sont aussi considérées.

L'intensification de la recherche implique la création d'un voisinage des solutions, afin de chercher autour d'une solution initiale, des possibles améliorations. La recherche locale explore ces voisinages. A ce type d'exploration, des méthodes de diversification sont ajoutées afin de sortir des optimaux locaux. Un optimum local est une solution avec des très bonnes performances, de tel sort qu'elle empêche la génération des nouveaux voisinages. Les méthodes de diversification peuvent être appliquées avec des listes de candidats (ou listes taboues) lesquelles sont construites de manière à éviter les optimaux locaux identifiés.

Certaines références présentent des travaux avec ces méthodes pour des problèmes similaires à ceux traités dans ce mémoire. Laguna *et al.* [71] proposent une série de recherches locales et méthodes taboue hybridées pour les problèmes avec fonctions objectifs proportionnelles aux retards. Dans la minimisation du makespan, Persma et Dijk [102] traitent un problème avec des machines indépendantes, avec un algorithme de recherche locale de voisinages efficaces. Tang et Luo en 2006 [127] ont adapté un algorithme de recherche locale avec plusieurs départs pour ce problème. Les auteurs proposent une analyse comparative des différents mouvements de diversification.

Armentano et Yamashita en 2000 [8] proposent une méthode du type recherche taboue avec des stratégies de diversifications innovantes pour le problème avec des dates de disponibilité identiques. Bilge *et al.* en 2004 [22] présentent dans leur article une méthode de recherche taboue appliquée au problème avec des machines parallèles et minimisation du retard total, où les dates d'arrivée des tâches sont différents de 0, mais en ajoutant la contrainte des temps de préparation non négligeables. Le problème avec des séquences entre les tâches est traité avec une recherche taboue dans [7].

Dans [26] une méthode de recherche taboue a été adaptée afin de résoudre le problème avec des machines indépendantes. Dans [84] des voisinages efficaces sont combinés avec des méthodes taboue afin de résoudre le problème d'ordonnancement de l'atelier du type flow shop sans permutation. Les performances obtenues sont confrontées avec les instances de Taillard [125].

Autres heuristiques

D'autres méthodes approchées ont été adaptées dans la littérature. En 2012, Yalaoui [140] a testé différents méthodes heuristiques pour le problème avec machines parallèles et minimisation du retard total. Des méthodes métaheuristiques comme l'utilisation des colonies de fourmis ou des méthodes adaptées créées pour la résolution de problèmes voisins ont été comparées à des méthodes classiques comme les listes de priorité (*SPT* «Shortest Processing Time», *EDD* «Early Due Date», etc).

Une variété de méthodes approchées ont été adaptées à la résolution des problèmes proches à ceux décrits précédemment. Des méthodes heuristiques ont été développées pour le problème avec machines parallèles, avec des dates d'arrivée des tâches identiques : Alidaee et Rosa en 1997 [4] proposent une heuristique basée sur le même principe que la méthode *MDD* («Modified Due Date») mais adaptée pour le problème à machines parallèles. Pfund *et al.* en 2008 [101] proposent une heuristique qu'ils ont appelé *ATCS* (Apparent Tardiness Cost with Setups). Le problème traité est très proche du problème $P_m/r_i/\sum T_i$, mais les retards sont pondérés, car les tâches n'ont pas la même importance. Biskup *et al.* en 2008 [23] ont proposé deux méthodes de résolution pour le problème sans les dates d'arrivée différentes. L'une de ces méthodes est la résolution d'une formulation mathématique du problème. La deuxième méthode est une heuristique qu'ils ont appelé *BHG*. Cette dernière a été reprise par Yalaoui [140] pour résoudre le problème $P_m/r_i/\sum T_i$, où les dates de disponibilité des tâches (r_i) différentes sont considérées.

Des métaheuristiques de différent nature ont été adaptées par Colorni *et al.* [35] pour résoudre des problèmes combinatoires difficiles comme l'ordonnancement dans l'atelier à cheminements indépendantes («Job-shop»). Les méthodes adaptées incluent des algorithmes génétiques, recherches taboue et des méthodes par colonie de fourmis. Pour un problème similaire, Dugardin *et al.* [44] ont testé différentes métaheuristiques entre autres un algorithme génétique et une colonie de fourmis. Blum *et al.* [25] ont réalisé une étude bibliographique des méthodes hybrides entre métaheuristiques et autres méthodes (métaheuristiques, programmation linéaire, programmation par contraintes, méthodes de séparation, ou programmation dynamique) afin de résoudre des problèmes combinatoires difficiles. Dorigo et Blum [41] proposent une étude bibliographique dans l'utilisation des algorithmes avec les colonies de fourmis, parmi lesquels, des problèmes d'ordonnancement ont été résolus. L'utilisation des algorithmes avec des colonies de fourmis sont aussi étudiés par Pedemonte *et al.* [100]. Les auteurs analysent les différentes caractéristiques de ces algorithmes avec les différentes interprétations données dans la résolution des problèmes de plusieurs domaines.

En 2011, Lin *et al.* [83] traitent un problème similaire au problème de machines parallèles,

mais avec deux fonctions objectifs alternatives. Des méthodes heuristiques sont proposées pour chacun des objectifs. Un algorithme génétique est proposé avec une paramétrisation effective pour la résolution des problèmes liés au retard.

Le problème d'ordonnancement de l'atelier à cheminement unique a été traité avec différentes méthodes adaptées à sa résolution. Par rapport au problème avec 2 opérations, il a été récemment traité dans [5] où un ensemble de méthodes métaheuristiques ont été utilisées à sa résolution. Ce même problème a été traité avec une méthode à deux phases par Su [118], afin de composer des lots de traitement, dans un cas où la deuxième opération de l'atelier peut traiter des lots. Des nouvelles métaheuristiques sont testées par Ching et Liao [34] basées sur des systèmes de immunoglobuline pour le problème *HFSP*. Les résultats sont comparés aux performances des métaheuristiques plus classiques comme l'algorithme génétique et la colonie de fourmis. Lee *et al.* [79] présentent des algorithmes qui ont été adaptés afin de prioriser les opérations les plus chargées dans les ateliers du type *HFSP*, en utilisant des estimations pour les dates d'arrivée et de fin souhaitée de l'opération priorisée.

L'atelier à cheminement unique où des plannings avec permutation sont acceptés a été largement étudié. En 2013, Pan et Ruiz [96] ont effectué des analyses statistiques sur 22 méthodes différentes afin d'identifier des méthodes dominantes, pour la minimisation de critères liées aux dates de fin réelles. Vallada et Ruiz [131] ont développé plusieurs versions d'algorithmes génétiques afin de résoudre le problème *HFSP*, avec des plannings à permutation, et minimisation du retard total.

2.5 Conclusion

Dans ce chapitre, nous avons présenté un état de l'art sur les principaux sujets traités dans ce mémoire. Nous avons inclut des références traitant les problèmes d'ordonnancement les plus proches des problèmes décrits dans le chapitre 1.

Certaines méthodes de résolution typiquement utilisées dans la résolution des problèmes combinatoires ont été également présentées, avec la classification souvent défini entre les méthodes garantissant des solutions optimales (exactes) et les autres (approchées).

Cette étude a montré que, à notre connaissance, peu de méthodes ont été adaptées à la résolution des problèmes d'ordonnancement, avec l'ensemble de contraintes définies dans le chapitre 1.

Pour le problème $P_m/r_i/\sum T_i$, certaines méthodes ont été appliquées à sa résolution, notamment dans [140]. Nous proposons dans le chapitre 3, un ensemble de méthodes pour