Projet de fin d'étude

Introduction de modalités adaptatives dans le logiciel de planification évolutionnaire Divide-and-Evolve

Auteur :
Alexandre Quemy

Encadrants :
Marc Schoenauer
Christian Goût

Inria - Equipe TAO
INSA Rouen - Département Génie Mathématique

24 juin 2014

Remerciements

Je tiens tout d'abord à remercier Marc Schoenauer qui a accepté de me confier ce projet, malgré la distance, et dont les conseils et l'aide ont été précieux pour le concrétiser. Mes remerciements également à Christian Goût qui a accepté de suivre le projet tout au long de l'année, encore une fois malgré la distance.

Je voudrais tout particulièrement remercier le Jazz Rock Cafe à Cracovie, sans lequel ce projet et rapport aurait été terminé bien plus tôt.

Table des matières

Abbréviations

AEs	**A**lgorithmes **E**volutionnaires
DAE	**D**ivide **a**nd **E**volve
EMOAs	**A**gorithmes **é**volutionnaires **M**ulti-Objectif
MOPs	**M**ulti**o**bjectives **P**roblems
PPTs	**P**roblemes de **P**lanification **T**emporelle
YAHSP	**Y**et **A**nother **H**euristic **S**earch **P**lanner

Chapitre 1

Contexte & état de l'art

1.1 Introduction

L'optimisation est un domaine en perpétuelle évolution, avec une valorisation par les entreprises de plus en plus importante des outils et méthodes de résolution. Notamment puisqu'un nombre important de problèmes posés en optimisation a des applications directes ou est directement élaboré pour répondre à un besoin concret de ces agents économiques.

Le développement de l'informatique et de la puissance de calcul a permis de résoudre des problèmes plus complexes et des instances de ces problèmes de taille toujours plus importante. Cependant, l'explosion combinatoire et d'autres phénomènes similaires (citons simplement l'irrégularité de l'espace de recherche) sont un soucis quand à la mise en pratique de certaines méthodes pour des problèmes de grande complexité ou des instances de taille trop importante. Cela incite évidemment la recherche à élaborer de nouvelles méthodes afin de palier à ces soucis mais également de proposer un panel de méthodes dont les objectifs ne sont pas forcément les même, quand bien même elles résoudraient un même problème : quelques fois un problème requiert une résolution exacte où le temps de calcul importe peu (toute proportion gardée) alors que d'autres fois, l'obtention d'une solution rapidemment est primordiale, même si celle-ci n'est pas la meilleure.

Notons également que la majorité des problèmes qui trouvent une application dans la vie réelle sont en réalité multi-objectifs. C'est à dire qu'il ne s'agit pas simplement de trouver les meilleures paramètres pour optimiser une certaine fonction de coût mais d'optimiser

simultanément plusieurs fonctions intrinsèquement liées et évidemment antinomiques. Toute entreprise cherche à optimiser la qualité de ses produits en réduisant ses coûts de production. Il s'agit bien là d'objectifs contradictoires, d'autant qu'il n'existe pas une unique solution comme c'est généralement le cas en cas d'objectif unique, mais d'un ensemble de solutions optimales (en un sens à définir) qui réalisent des compromis.

Toute la difficulté de l'optimisation multi-objectifs est de trouver ces solutions de compromis pour les proposer à l'utilisateur et le guider dans ses choix.

La nécessité d'obtenir des solutions de grande qualité (sur des problèmes multi mais également mono-objectif) en un temps raisonnable voire très court à conduit à imaginer des algorithmes hybrides, tirant parti des méthodes directes de type descente de gradient et des algorithmes donnant une solution approchée comme les métaheuristiques par exemple. La synergie de ces deux approches a été validée expérimentalement par un certain nombre de solveurs, dont Divide and Evolve et constitue à ce jour un domaine de recherche très actif, faisant naître de nouvelles problématiques.

Divide and Evolve (DaE) est un solveur pour le problème de planification temporelle en intelligence artificielle dont la particularité est de chercher, à l'aide d'un algorithme évolutionnaire, des sous-problèmes supposés faciles à résoudre, afin de les donner à résoudre à un solveur classique embarqué. L'approche a été validée expérimentalement et a gagné le problème *deterministic temporal satisficing* durant la compétition ICP 2011. La caractéristique remarquable de DaE est, comme tout algorithme évolutionnaire, qu'il peut être « facilement » modifié de sorte à résoudre des problèmes multi-objectifs. C'est alors la première approche Pareto à ce jour pour la résolution d'un problème de planification temporelle multi-objectif.

Si un certain nombre d'expérimentations montrent le succès de MO-DaE, l'absence d'instance multi-objectif pour le problème de planification temporelle est un soucis pour tester de manière exhaustive les capacités du solveur.

De même, un certain nombre de paramètres et de stratégies internes restent à établir afin de tirer le meilleur du solveur.

Les objectifs de ce projet de fin d'étude sont donc les suivants :

— Mise au point de problèmes *benchmarks* de référence pour lesquels la forme du front de Pareto est paramètrable, ainsi que la difficulté du problème, dans le but d'observer les réponses de DAE dans différentes situations.
— Conception, programmation et validation expérimentale de l'adaptation *online* de paramètres.
— Réglages *offline* ou *online* des principaux paramètres de DAE.

Dans ce chapitre nous proposons de développer le contexte gravitant autour du projet : d'une part la présentation du problème de planification temporelle et sa représentation, et d'autre part les travaux effectués jusqu'alors sur DAE, de la preuve de concept jusqu'aux tests de performances de MO-DAE.
Un second chapitre est dédié à l'état de l'art de l'optimisation multi-objectif, des méthodes classiques aux algorithmes évolutionnaires à estimations de distribution.

Les chapitres suivants présentent le travail effectué. Dans le chapitre 3, l'extension au cas multi-objectif d'un problème bien connu de planification temporelle et une méthode de résolution sont présentée. Les performances de la méthode, au travers d'un solveur C++, sont démontrées empiriquement. Le chapitre 4 se concentre sur l'optimisation de paramètres et particulièrement l'optimisation *online*, c'est à dire au cours de la recherche, de l'objectif à faire optimiser par le solveur embarqué par DAE. Ce chapitre sera également le théatre de certaines réflexions sur DAE menant à de nouvelles perspectives.

1.2 Planification temporelle en Intelligence Artificielle

Le problème de planification peut s'exprimer par un modèle d'état où S est l'ensemble des états possibles. $s_0 \in S$ est l'état initial du problème et $S_G \subseteq S$ est un ensemble de buts à atteindre. $A(s)$ est l'ensemble des actions applicables à l'état s. On dispose d'une fonction dite de transition, définie par $f : A \times S \to S' \subseteq S$ telle que $f(a, s) = s'$ où S' est l'ensemble des états pouvant être atteints depuis l'état s par l'application d'une action $a \in A(s)$.

Une solution du problème est donnée par une suite d'actions transformant s_0 en $s_G \in S_G$. Une telle solution est appelée un **plan**.

Le problème de planification temporelle étend le problème de planification par la possibilité d'avoir des actions concurrentes.

1.2.1 Langage PDDL

PPDL pour *Planning Domain Definition Language* est une standardisation pour la description du *domaine* lié au problème de planification en intelligence artificielle. Inspiré notamment de STRIPS [1], c'est le langage utilisé pour l'*International Planning Competition* (IPC). Il s'agit du langage utilisé par DAE.

Le domaine définit les actions ainsi que les atomes composant l'environnement. La section *requirements* permet de donner quelques caractéristiques au problème comme l'utilisation de typage pour les prédicats, l'utilisation de durées de coûts pour les actions, etc.
La section *predicates* déclare l'ensemble des atomes permettant de définir l'état de l'environnement. Enfin, la section *action* permet de définir les actions qui vont agir sur les prédicats, ainsi que d'en définir des pré-conditions ou diverses caractéristiques. Le langage ne se limite pas à ces quelques sections mais à elles seules elle permettent de définir des problèmes simples.

```
(define (domain DOMAIN_NAME)
  (:requirements [:strips] [:equality] [:typing] [:adl])
  (:predicates (PREDICATE_1_NAME ?A1 ?A2 ... ?AN)
               (PREDICATE_2_NAME ?A1 ?A2 ... ?AN)
       ...)

  (:action ACTION_1_NAME
    [:parameters (?P1 ?P2 ... ?PN)]
    [:precondition PRECOND_FORMULA]
    [:effect EFFECT_FORMULA]
   )

  (:action ACTION_2_NAME
```

1. STRIPS pour *STanford Research Institute Problem Solver* désigne à la fois un algorithme de planification et le langage formel utilisé par celui-ci.

```
    ...)

  ...)
```

Une fois le domaine définis, il reste à créer le problème. Pour cela il faut déclarer les instances des objets définis dans le problème (principalement leur nombre), ainsi qu'initialiser l'ensemble des prédicats qui permettent de caractériser leur état.
Enfin, une description des buts à atteindre et éventuellement des métriques à utiliser par le solveur complète l'instance.

```
(define (problem PROBLEM_NAME)
  (:domain DOMAIN_NAME)
  (:objects OBJ1 OBJ2 ... OBJ_N)
  (:init ATOM1 ATOM2 ... ATOM_N)
  (:goal CONDITION_FORMULA)
  )
```

1.3 Divide And Evolve

L'ensemble de cette section est largement inspiré des diverses publications de l'équipe TAO sur le sujet et fait office de synthèse pour le lecteur découvrant ces travaux.

Il existe très peu de tentatives de résolution des PPTs par des algorithmes évolutionnaires et probablement aucune dans la version multi-objectifs du problème.

Divide and Evolve se présente comme une approche mémétique de la résolution du PPT, mais contrairement aux approches mémétiques classiques qui vont utiliser la méthode locale pour améliorer les solutions proposées par l'AE, Divide And Evolve peut être vu comme une analogie au paradigme *Divide and Conquer* dans lequel on essaye de diviser un problème complexe en sous-problèmes plus simples à résoudre. L'AE ne traite pas le problème en tant que tel mais va chercher des sous-problèmes plus simples à résoudre pour une méthode locale, en espérant que la résolution des sous-problèmes soit plus rapide que la résolution du problème dans sa globalité.

Considérons $P =< A, O, I, G >$ un PPT définis par le domaine constitué des atomes A et des objets O, l'état initial I et l'état à atteindre G. Pour résoudre P, l'idée principale de DAE_X est de trouver une séquence d'états $(S_i)_{1 \leq i \leq n}$ à l'aide d'un algorithme évolutionnaire et de donner les sous-problèmes $P_i =< A, O, S_i, S_{i+1} > \ \forall i \in \{0 \dots n\}$ (avec $I = S_0$ et $S_n = G$) à résoudre à un solveur embarqué X. La résolution de chaque sous-problème P_i produit un plan σ_i dont la concaténation (et éventuellement une compression pour tenir compte du caractère concurrent de certaines actions en planification temporelle) permet d'obtenir un plan σ, solution au problème P. Si un sous-problème n'est pas résolu par le solveur embarqué, l'individu est fortement pénalisé (qualifié de *unfeasible*) ce qui entrainera sa suppression à terme par l'algorithme évolutionnaire.

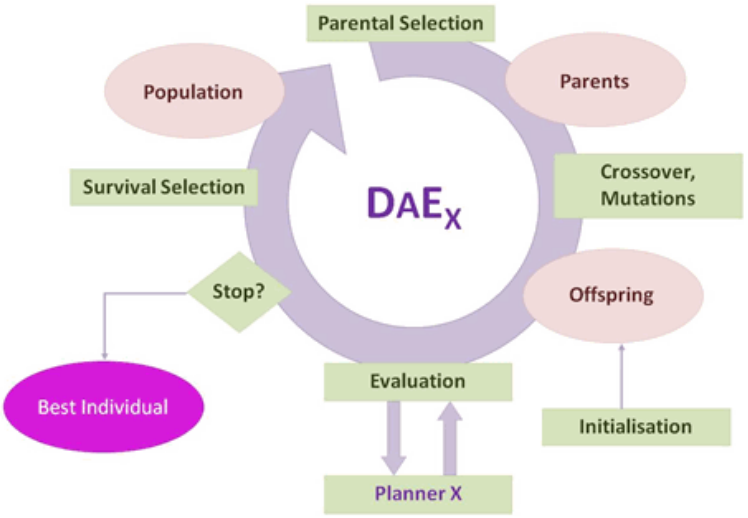


FIGURE 1.1: Illustration du fonctionnement de DAE.

1.3.1 Représentation de l'AE

1.3.1.1 Individus & Initialisation

Chaque individu est codé par une liste de longueur variable d'états de S et est donc défini par $(s_i)_{i \in \{1 \dots n\}}$ où n, longueur de l'individu, est inconnue et variable au cours de la résolution. Le but initial s_0 et le $s_G \equiv s_{n+1}$ ne seront pas encodés directement dans le génotype (il faudra cependant s'assurer que s_n correspond à un état depuis lequel on peut atteindre s_{n+1}.

Ces états représentent les sous-buts par lesquels on doit impérativement passer et donc indique les sous-problèmes à résoudre par le solveur local.

Notons que la taille de l'espace des listes d'états explose très rapidemment avec l'augmentation du nombre d'objets. Cependant, chaque état peut être décrit partiellement, en ne tenant compte que des objets instanciés et c'est pourquoi un individu sera codé par une liste d'états partiels, c'est à dire une liste de listes d'atomes qui doivent être vrais.

La méthode utilisée pour l'initialisation des individus est une heuristique développée par Haslum et Geffner [1], h^1, qui estime pour chaque atome le moment au plus tôt pour lequel il peut devenir vrai. En utilisant l'ordre induit par le temps au plus tôt on peut restreindre l'ensemble des atomes candidats pour chaque état partiel et en vérifiant que l'on n'ajoute que des atomes qui ne sont pas mutuellement exclusifs on peut construire un individu faisable.

1.3.1.2 Opérateur de croisement

De part la nature séquentielle des individus (chaque état se suit), les opérateurs de croisement choisis sont les opérateurs de croisement à 1-point et 2-points afin de préserver au maximum la chronologie.

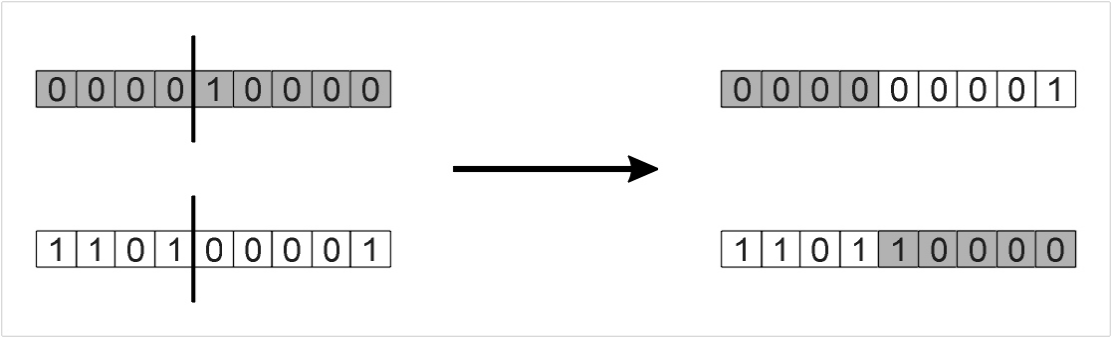


FIGURE 1.2: Illustration du croisement à un point sur une chaine de bits.

On peut voir que la longueur des enfants ne sera potentiellement pas la même que celle des parents. Par ailleurs, un seul enfant sera gardé : celui qui respecte au mieux l'ordre chronologique et les relations d'exclusion entre les atomes.

Le choix des états à choisir pour le croisement peut être uniforme ou basé sur une métrique si celle-ci peut être définie pour le problème considéré.

1.3.1.3 Opérateur de mutation

De part la représentation de l'individu, deux types de mutation sont utilisés, pour un total de 4 opérateurs :

— Au niveau de l'individu

— Au niveau du gêne

Les opérateurs de mutation font en sorte de garder une chronologie approximative entre les différents états S_i qui caractérisent un individu et la consistance de ces états en interdisant des atomes qui seraient mutuellement exclusifs.

Au niveau de l'individu Deux opérateurs sont utilisés : `AddState` et `DelState` qui vont respectivement ajouter un état à la liste, résultant d'un individu de taille $n+1$, et retirer un état résultat d'un invidiu de taille $n-1$.

Le choix de l'état à choisir peut être fait selon plusieurs méthodes :

— Choix uniforme.

— Dernier état atteint en cas d'impossibilité de résolution des sous-problèmes par le solveur local.

— Sous-problème le plus compliqué (métrique à définir, par exemple le nombre de *backtracks*) résolus par le solveur local.

Au niveau du gène On peut évidemment directement modifier un gène c'est à dire, un état S_i en ajoutant ou supprimant un atome(respectivement `AddAtom` et `DelAtom`). Le choix est uniforme parmi les atomes.

1.3.2 Le choix du solveur embarqué

En théorie, n'importe quel solveur de planification peut être utilisé avec DAE. Cependant, les résultats obtenus avec un solveur optimal, CTP, ont été décevants, en partie probablement par le temps CPU nécessaire à la résolution exacte des sous-problèmes (les appels au solveur local étant très nombreux : pour chaque génération, il faut évaluer chaque individu, c'est à dire résoudre n_i sous problèmes où n_i est la longueur de l'individu i. Avec 10 générations, 30 individus et une longueur moyenne de 5 sous-problèmes, nous obtenons 1500 appels au solveur local). A contrario, l'expérience à montré qu'il

n'était pas nécessaire d'obtenir une solution optimale aux sous-problèmes pour obtenir une solution de bonne qualité sur le problème global. Ainsi, l'utilisation d'un solveur sous-optimal donnera de meilleurs résultats.

Pour les différentes expérimentations, le solveur utilisé est YAHSP (*Yet Another Heuristic Search Planner*). Pour plus de détails sur ce solveur, on renvoie à l'article [2].

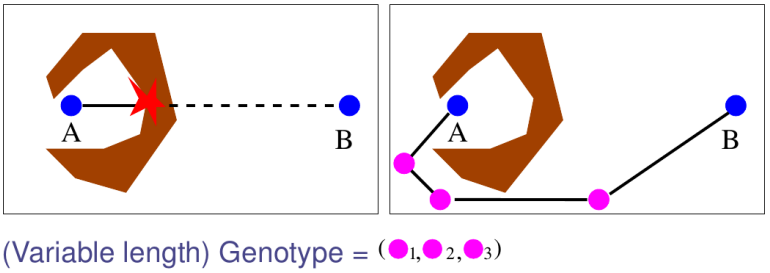


FIGURE 1.3: Paradigme DAE : le rôle d'oracle de l'algorithme génétique.

1.3.3 MO-DAE

Les algorithmes évolutionnaires peuvent facilement être modifiés pour tenir compte de plusieurs objectifs. Ainsi, il s'agira principalement de changer le processus de sélection (c'est à dire comment sélectionner les individus pour l'étape de *crossover* et comment intégrer les nouveaux individus dans la population).
MO-DAE est la version multi-objectifs de DAE. La difficulté majeure rencontrée est qu'à priori, il n'existe pas de solveur de planification multi-objectifs alors que le processus de sélection requiert une évaluation de l'ensemble des objectifs pour chaque individu.

Cependant, comme cela a été introduit dans la section sur PDDL, PDDL 3.0 permet d'ajouter des quantitées appelées des *Soft Constraints* qui n'intérfèrent pas dans le processus de recherche mais sont simplement calculées à la fin de celui-ci. Il est ainsi possible d'appeler YAHSP pour qu'il optimiser un problème par rapport à un objectif f_1 et qu'il calcul la valeur de plan σ ainsi obtenue par rapport à un objectif f_2. L'individu est donc pleinement évalué et il est possible d'appliquer des procédures de *ranking* classiques des algorithmes évolutionnaires multi-objectifs. Pour plus de détails sur les procédures de *ranking*, voir paragraphe 2.4.1.

L'idée est donc de donner un sous-problème à résoudre au solveur embarqué en lui disant quel objectif optimiser. Le choix de l'objectif à optimiser fait partie des paramètres à

étudier. Actuellement, la stratégie utilisée pour les tests de performances se base sur une sélection stochastique où la probabilité de chaque objectif est caractérisée par des poids définis par l'utilisateur.

D'autres stratégies restent à établir, notamment une stratégie auto-adaptative où le choix des stratégies serait porté par chaque invividu ou chaque gène de l'individu et évoluerait au cours de l'algorithme avec les individus.

1.4 ParadisEO

ParadisEO est un framework open-source développé en C++ et dédié au développement de méthodes d'optimisation approchées classiques, multi-objectifs, parallèles et hybrides. L'implémentation de DAE est basée sur ParadisEO, notamment sur le module MOEO pour l'optimisation multiobjectifs.

ParadisEO est composé de plusieurs modules reposant tous sur une bibliothèque avec laquelle il a fusionné fin 2012.

1.4.1 Evolving Objects

Evolving Objects était un *framework* indépendant de ParadisEO, destiné à l'élaboration de métaheuristiques à base de population comme les algorithmes génétiques.

Développé initialement par l'équipe TAO de Inria Saclay, menée par Marc Schoenauer, le framework fut par la suite maintenu par Thales avant de fusionner avec ParadisEO, dont le maintien est effectué à la fois par Thales et l'équipe DOLPHIN à Inria Lille.

La totalité des autres modules de ParadisEO reposent sur Evolving Objects.

Chapitre 2

Optimisation multiobjectif

2.1 Problème d'optimisation multiobjectif

2.1.1 Définition et notations

Un problème d'optimisation multiobjectif (MOP) se formule de la manière suivante :

$$\begin{cases} opt & f(x) \\ x \in \Omega \\ g(x) \leq 0 \end{cases}$$

Avec $f(x) = (f_i)_{1 \leq i \leq n}$, vecteur objectif, $g(x) = (g_i)_{1 \leq i \leq n}$ vecteur des contraintes et Ω l'espace des variables de décision (par exemple $\mathbb{R}^m$).

L'opérateur général *opt* indique juste un but d'optimisation. Si en optimisation mono-objectif cet opérateur correspond soit à une minimisation soit à une maximisation, en optimisation multiobjectif il peut s'agir d'une minimisation de certains objectifs et une maximisation d'autres. Dans la suite et pour faciliter les notations on considérera un problème de minimisation sur l'ensemble des objectifs.

On distingue l'espace des variables de décision, que nous noterons E, de la projection de cet espace par f, appelé espace des objectifs et noté F. Dans la littérature des métaheuristiques, on parle également de *landscape*.

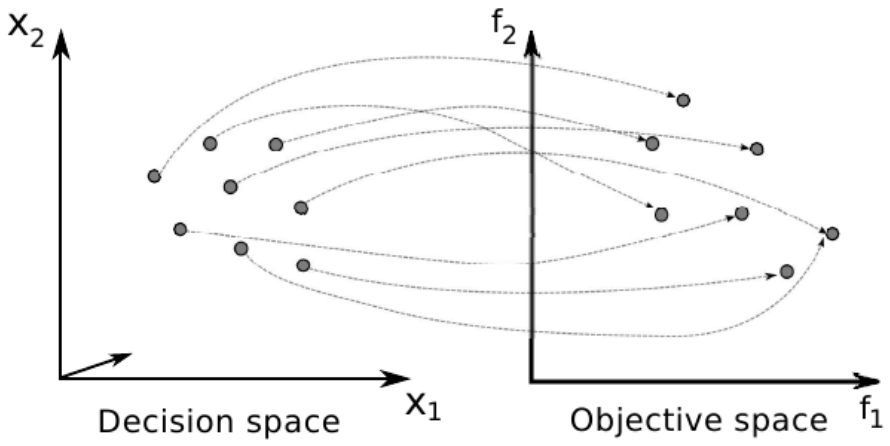


FIGURE 2.1: Projection de l'espace des variables de décision dans l'espace des objectifs.

2.2 Front de Pareto

Contrairement aux problèmes mono-objectifs, les MOPs ne présentent pas une unique solution [1] mais un ensemble de solutions réalisant un équilibre dit de Pareto. Simplement, il s'agit des solutions pour lesquelles il n'est pas possible d'améliorer un objectif sans en détériorer au moins un autre.

Ainsi, l'objectif des algorithmes d'optimisation multi-objectifs est de trouver cet ensemble de solutions optimales au sens de Pareto ou une approximation de celui-ci. Pour cela, en plus de la propriété de convergence, il nous faut une propriété d'uniformité pour s'assurer qu'au delà d'obtenir des solutions proches du front de Pareto, nous obtenons des solutions réparties sur ou au voisinage de l'ensemble du front de Pareto, minimisant ainsi la perte d'information potentiellement utile au décideur.

La difficulté de résolution des MOPS réside dans le fait qu'il n'existe pas une unique définition d'une solution optimale, puisque la relation de dominance que nous établirons dans le section suivante n'implique qu'un ordre partiel sur les solutions, le choix final revenant au décideur. De plus, la plupart des problèmes multi-objectifs sont dits *intraitables* [2] dans le sens où il existe des instances telles que la taille de l'ensemble de Pareto est exponentielle en la taille du problème, faisant de la majorité des problèmes multi-objectifs des problèmes de la classe de complexité $NP-$SPACE [3]. Enfin, l'impact de la forme et structure du front de Pareto sur l'efficacité des méthodes est également à

1. Solutions dans l'espace des objectifs et non dans l'espace des variables de décision.
2. *untractable* en anglais.

prendre en compte comme nous le verrons dans la section dédiée aux méthodes classiques.

Définition 2.1. Soit $x, y \in \Omega$, x domine y si $\begin{cases} \forall i \in \{1, \dots, n\},\ f_i(x) \succeq f_i(y) \\ \exists j \in \{1, \dots, n\},\ f_j(x) \succ f_j(y) \end{cases}$

On rappelle que la notation $f(a) \succ f(b)$ signifie que $f(a)$ est meilleur que $f(b)$ au sens du problème, c'est à dire $f(a) > f(b)$ pour un problème de maximisation et $f(a) < f(b)$ pour un problème de minimisation. Par extension, on utilisera la même notation directement entre les individus : $a \succ b$ signifie que a domine b.

Cette relation définit un ordre partiel sur les solutions car certaines solutions ne sont pas comparables entre elles.

Définition 2.2. Soit $x, y \in \Omega$, x domine strictement y, noté $x \succ\succ y$, si $\forall i \in \{1, \dots, n\}, f_i(x) \succ f_i(y)$

Définition 2.3. Soit $x, y \in \Omega$, x domine faiblement y, noté $x \succeq y$, si $\forall i \in \{1, \dots, n\},\ f_i(x) \succeq f_i(y)$

Définition 2.4. Soit $x, y \in \Omega$, x est non-comparable à y, noté $x \,||\, y$, si $\neg(x \succeq y) \wedge \neg(y \succeq x)$

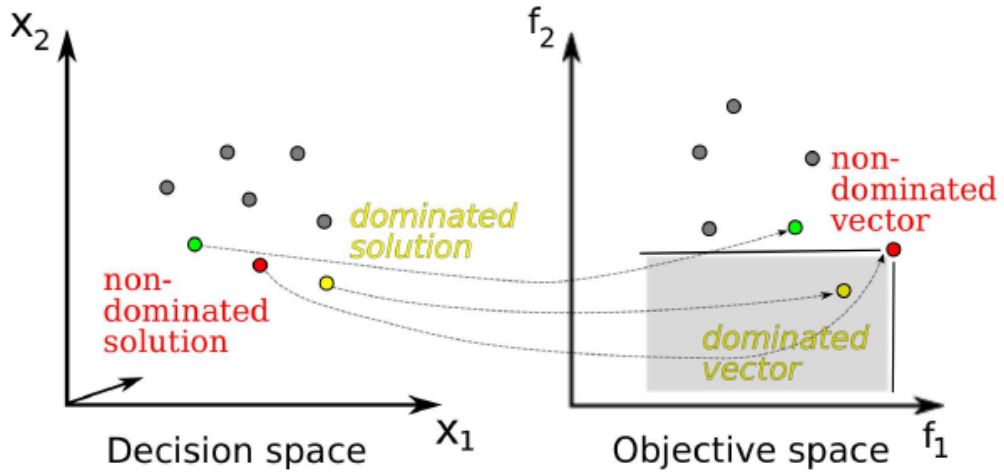


FIGURE 2.2: Problème de maximisation. Le point jaune est dominé par le point rouge qui est meilleur sur au moins un objectif (ici les deux). Le point rouge n'est dominé par aucun autre point. Il n'est pas possible de comparer le point vert et rouge puisque le vert est meilleur sur l'objectif 2 mais moins bon sur l'objectif 1 : la relation de dominance induit un ordre partiel.

On peut étendre ces relations aux ensembles de solutions de la manière suivante :

Définition 2.5. Soit $A, B \subset \Omega$, $A \succ B$, respectivement $A \succ\succ B$, respectivement $A \succeq B$, si $\forall b \in B \exists a \in A,\ a \succ b$, respectivement $a \succ\succ b$, respectivement $a \succeq b$.

C'est à dire que chaque élément de B doit être dominé par au moins un élément de A. On peut préciser alors ce qu'est un ensemble meilleur qu'un autre de la manière suivante :

Définition 2.6. Soit $A, B \subset \Omega$, A est meilleur que B, noté $A \triangleright B$, si $(A \succ B) \wedge (A \neq B)$.

Le Front de Pareto représente l'ensemble des solutions qui ne sont pas dominées.

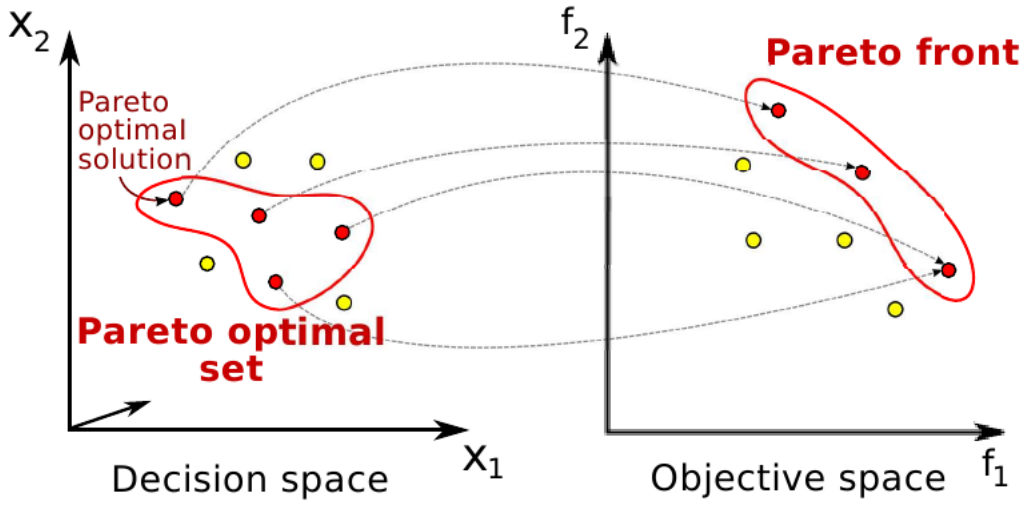


FIGURE 2.3: Illustration de l'ensemble des solutions Pareto-optimales et leur projection dans l'espace des objectifs : le Front de Pareto.

Définition 2.7. Le point idéal I est défini comme le point dont les coordonnées sont les meilleurs valeurs de chaque objectif des points du front de Pareto : $I_i = min(f_i(x) \mid x \text{ non-dominé})$, dans le cas d'un problème de minimisation.

Définition 2.8. Le point Nadir N est défini comme le point dont les coordonnées sont les pires valeurs de chaque objectif des points du front de Pareto : $N_i = max(f_i(x) \mid x \text{ non-dominé})$, dans le cas d'un problème de minimisation..

2.3 Méthodes classiques

On dénote un grand nombre d'approches pour la résolution de MOPs. On peut cependant scinder ces approches en deux grandes catégories. Une première catégorie, dite « non Pareto » ou « classique » consiste à transformer un problème multi-objectif en un ou

plusieurs problèmes mono-objectifs, soit en fixant des préférences sur les critères, en aggrégeant les critères ou en modifiant les contraintes du problème initial. La seconde catégorie, dite « Pareto » s'oppose à la première dans le sens où elle ne transforme pas le problème initial : les objectifs sont traités sans aucune distinction.

Dans cette section nous présentons brièvement les différentes méthodes non Pareto pour la résolution de MOPs. Nous pouvons classer ces méthodes en deux catégories : les méthodes sans contrainte et les méthodes avec contraintes.

Nous présentons également quelques outils mathématiques et approches récentes pour résoudre certains problèmes liés à ces méthodes.

2.3.1 Généralités sur les méthodes d'aggrégation

Les méthodes d'aggrégation reformule le problème de la manière suivante :

$$\begin{cases} min \\ x \in \Omega \\ g(x) \leq 0 \end{cases} (U(f(x))$$

Avec U une fonction d'aggrégation en ce sens qu'elle a valeur dans un espace de dimension 1. Il s'agit donc d'exprimer numériquement la préférence, c'est à dire que $\forall x, y,\ x \succ y \Leftrightarrow U(x) > U(y)$.

Parmi les fonctions d'aggrégation les plus utilisées nous retrouvons une fonction additive de la forme suivante :

$$U = \sum_{i=1}^{n} \omega_i f_i$$

A laquelle nous ajoutons généralement une contrainte sans perte de généralité : $\sum_{i=0}^{n} |\omega_i| = 1$.

Dans ce cas nous pouvons appliquer des méthodes classiques d'optimisation mono-objectif, en faisant varier le vecteur ω.

Toutefois, cette méthode présente plusieurs inconvénients :

1. Le calcul *à priori* des poids n'est pas aisé.
2. Traiter les intéractions entre les différents objectifs pose problème.
3. Le modèle est sensible au changement d'échelle.

4. Certains objectifs peuvent se compenser, entrainant une perte d'information.
5. Il est impossible d'atteindre les zones concaves de l'espace des objectifs réalisables.

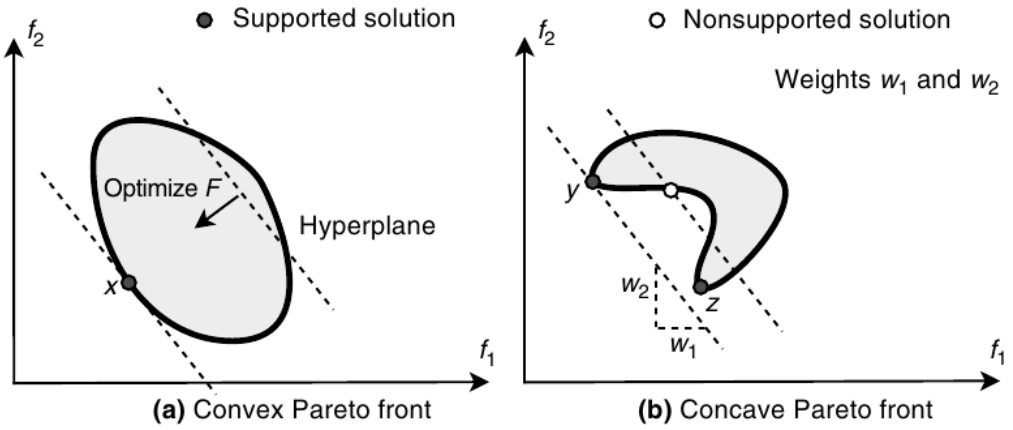


FIGURE 2.4: Illustration des difficultés des méthodes d'aggrégation à atteindre les parties concaves du Front de Pareto.

Les intéractions entre les objectifs peuvent être de différentes natures. Par exemple, il est possible de vouloir que l'influence d'un objectif soit plus grande lorsqu'un second est réduit ou n'est pas là car il existe une forte corrélation entre ces objectifs.
On peut souhaiter que la satisfaction d'un objectif soit autant valorisée que la satisfaction de plusieurs objectifs (interchangeabilité) ou, au contraire, que la satisfaction d'un objectif soit très peu valorisée par rapport à celle de plusieurs objectifs (complémentarité).

Exemple 2.1. *Grabisch [4]*

Imaginons un problème à deux objectifs f_1 et f_2, tout deux à valeur dans $[0,1]$ et 3 individus a,b et c avec les évaluations suivantes :

$$f_1(a) = 0.4 \quad f_1(b) = 0 \quad f_1(c) = 1$$
$$f_2(a) = 0.4 \quad f_2(b) = 1 \quad f_2(c) = 0$$

On peut légitimement penser qu'il vaut mieux obtenir une évaluation moyenne sur l'ensemble des critères plutôt qu'une faiblesse apparente dans au moins des critères. Cela amène à une préférence du type : $a \succ b \sim c$. Nous cherchons une pondération (ω_1, ω_2) des deux objectifs.

Des relations de préférences nous tirons :

$$\begin{cases} b \sim c \Leftrightarrow \omega_1 = \omega_2 \\ a \succ b \Leftrightarrow 0.4(\omega_1 + \omega_2) > \omega_2 \end{cases}$$

Ce qui conduit à $0.8\omega_2 > \omega_2$ *qui est bien évidemment impossible.*

Pour palier ce problème, plusieurs approches ont été développées, notamment l'utilisation d'intégrales de Lebesgue généralisées qui étendent l'intégration par rapport à une mesure aux mesures non-additives. Ces intégrales permettent d'exprimer l'intéraction entre plusieurs facteurs. On parle plutôt d'opérateur d'aggrégation dans un contexte discret.

Les deux principales intégrales non-additives utilisées sont l'intégrale de Choquet et l'intégrale de Sugeno. Le principal désavantage de ces méthodes est qu'elles requièrent une connaissance à priori du système de préférences sur les solutions contrairement à l'approche Pareto qui ne requiert aucune connaissance à priori. Cette connaissance à priori peut être difficile à déterminer ou changer selon le problème traité ou même l'instance considérée.

Pour de plus amples détails, on renvoie à l'article [5] pour une axiomatisation de l'intégrale de Choquet comme opérateur d'aggrégation, et de manière plus générale aux articles [6][7][8].

2.3.2 Méthode ϵ-contrainte

L'approche consiste à choisir un objectif et transformer l'ensemble des $m-1$ autres objectifs en contraintes de la manière suivante (en considérant un problème de minimisation) :

$$\begin{cases} \underset{x \in \Omega}{min} & f(x) \\ g(x) \leq 0 \\ f_j(x) < \epsilon_j, \quad \forall j \neq i \end{cases}$$

Comme avec la méthode d'aggrégation, il faut appliquer plusieur fois la résolution du problème, en faisant varier le vecteur ϵ. Cette méthode permet de trouver les parties concaves du front mais nécessite un grand nombre de résolutions ainsi qu'un choix judicieux du vecteur ϵ impliquant une connaissance à priori du problème traité.

2.3.3 Méthode de Tchebychev ou min/max

L'approche consiste à choisir un point de référence (ou *but*), et choisir comme fonction objectif la norme entre ce point de référence et un point de l'espace des objectifs. Si l'on peut utiliser n'importe quelle norme, généralement c'est la norme 1 et la norme ∞ (encore appelée norme de Tchebychev).

Avec cette dernière norme, le problème se réécrit ainsi :

$$\begin{cases} \underset{x \in \Omega}{min} \ \underset{1 \leq i \leq m}{max} (B_i - f_i(x)) \\ g(x) \leq 0 \end{cases}$$

La qualité des solutions obtenues avec cette méthode dépend fortement du point de référence choisi. Habituellement, le point choisi est le point idéal. Comme le Front de Pareto exact n'est pas connu à l'avance, le point idéal évolue donc au fur et à mesure de la recherche puisqu'il dépend des points non-dominés courants.

2.3.4 Méthode du but à atteindre

La méthode du but à atteindre mélange la méthode de Tchebychev dans le sens où elle utilise un point de référence B et la méthode ϵ-contrainte car elle transforme le problème en lui ajoutant des contraintes. Ces contraintes modélise la distance par rapport au but à atteindre (contrairement à Tchebychev où la distance est donné par une norme). Le problème se reformule donc de la manière suivante :

$$\begin{cases} \underset{x \in \Omega}{min} \quad \lambda \\ g(x) \leq 0 \\ f_j(x) - \lambda \omega_i \leq B_j, \quad \forall j \neq i \end{cases}$$

Le paramètre λ contrôle la direction de la recherche. Là encore, la méthode permet d'accéder aux points dans les zones concaves du front mais doit être itérée plusieurs fois et le choix des paramètres est crucial pour obtenir des résultats dans un temps raisonnable.

2.3.5 Discussion

Nous avons présenté ici quelques unes des méthodes classiques de résolution de MOPs. Ces méthodes présentent plusieurs désavantages. Certaines sont incapables de traiter des problèmes convexes, et la majorité est très sensible à la forme du Front de Pareto. Un autre désavantage est de devoir relancer plusieurs fois l'algorithme pour obtenir des solutions différentes appartenant au Front. Ceci est dû à la nature de ces méthodes qui consistent à transformer un problème multi-objectif en problème mono-objectif qui ne renverra qu'une seule solution.

Nous présentons toutefois quelques outils pour permettre de résoudre ces problèmes, notamment en établissant un modèle de préférences sur les critères permettant une résolution efficace et de pallier partiellement au problème de sensibilité. Cependant, cela implique une connaissance *à priori* et par la suite nous considèrerons qu'il n'existe pas de modèle de préférence (i.e. tous les critères ont la même importante).

2.4 Métaheuristiques pour l'optimisation multiobjectif

De part leur nature difficile, il existe très peu de méthodes exactes pour la résolution de MOPs, c'est pourquoi nous présentons ici des méthodes stochastiques approchées qui ont été appliquées avec succès à une large variété de problèmes académiques comme des problèmes réels.

Les EAMOs sont essentiellement des algorithmes évolutionnaires pour lesquels ont a modifié le processus de sélection afin de tenir compte de l'aspect multi-objectif du problème. Ainsi, en théorie, n'importe quel algorithme évolutionnaire mono-objectif peut être transformé en algorithme multi-objectif dont le but est d'approcher le Front de Pareto.

La sélection est basée sur un scalaire, la *fitness* et l'une des principales problématique est donc d'élaborer des méthodes d'évaluation des individus qui permettent de tenir compte de l'ensemble des objectifs. La seconde problématique est d'éviter la dérive génétique qui consisterait à n'approximer qu'une petite partie du Front de Pareto, voire un unique point, comme c'est le cas par exemple avec les méthodes classiques utilisant

l'aggrégation. Diverses techniques ont donc été développées dans le but de permettre une diversité génétique importante.

Nous présentons dans un premier temps les concepts communs à ces méthodes, puis quelques unes des méthodes les plus connues et efficaces.

2.4.1 Procédure de ranking

On appelle procédure de *ranking* les différentes manières d'évaluer les individus au sein d'un EAMOs. Toutes ces procédures se basent sur la notion de dominance au sens de Pareto.

— **Dominance count** : nombre d'individus qui dominent l'individu considéré.

$$C(x) = card\{y; y \succ x\}$$

— **Dominance strength** : nombre d'individus que domine l'individu considéré.

$$S(x) = card\{y; x \succ y\}$$

— **Pareto rank** : numéro du front de Pareto auquel appartient l'individu considéré.

Toutes ces procédures de ranking n'introduisent que des ordres partiels entre les individus et il est donc important d'avoir une seconde mesure, introduisant un ordre total, entre les individus pour distinguer deux individus ayant le même rang. Le critère alors utilisé se base sur la densité des solutions.

2.4.2 Densité de solutions - Mesure de diversité

Les mesures de densité permettent de favoriser la diversité génétique et donc d'améliorer l'approximation du Front de Pareto en offrant une meilleure couverture de celui-ci. Elles permettent aussi de différencier deux individus dont le rang serait identique comme cela peut être le cas entre un nombre important d'individus (par exemple dans le cas de l'utilisation du Pareto Rank, tous les individus sur un même front auront le même rang).

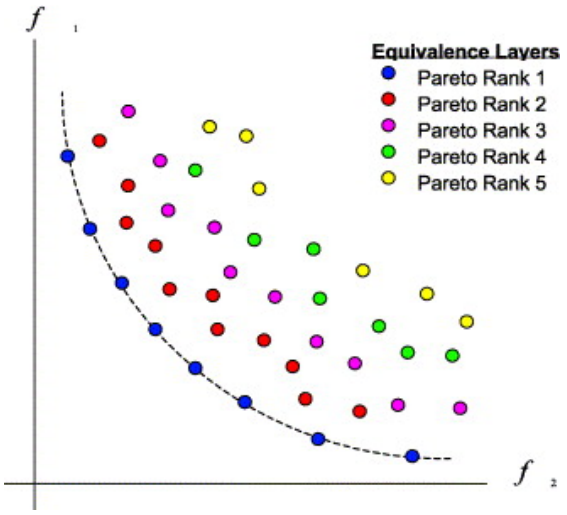


FIGURE 2.5: Illustration du rang de Pareto (Pareto rank). En bleu le Front de Pareto (rang 1), en rouge le Front de Pareto obtenu en retirant les éléments bleus (rang 2), et ainsi de suite.

Ces mesures représentent des distances entre les individus avec l'objectif de favoriser les individus isolés. Notons qu'elles peuvent être basées sur l'espace des objectifs ou l'espace des variables de décision.

2.4.2.1 Heuristique de partage

Dans le but de maintenir une diversité au sein de la population et empêcher une convergence vers un unique optimum au sens de Pareto, Goldberg et Richardson proposent en 1987 [9] une heuristique de partage, appelée également technique de *sharing*.
L'objectif est de favoriser, à qualité égale, un individu isolé par rapport à un individu issu d'une zone peuplée. Pour cela, on applique un coefficient dit de *sharing* au moment de l'évaluation de l'individu en fonction d'un seuil de voisinage σ_{share}.

On calcule la fitness de l'individu de la sorte :

$$f_s(x_i) = \frac{f(x_i)}{\gamma_i}$$

Avec :

$$\gamma_i = \sum_{j=1}^{N} S(d(x_i, x_j))$$

Et :

$$S(x) = \begin{cases} 1 - (\frac{x}{\sigma_{share}})^{\alpha} & x > \sigma_{share} \\ 0 & sinon \end{cases}$$

Si cette technique est efficace dans bon nombre de cas, la principale difficulté est l'élaboration de la distance d entre les individus (cependant, elle peut être indifféremment phénotypique ou génotypique). Elle introduit également deux paramètres, σ_{share} et α (facteur d'influence), qu'il est difficile de régler autrement qu'empiriquement puisqu'ils dépendent d'une part du problème étudié et d'autre part de la distance établie.

Enfin, le coût en calculs n'est pas négligeable puisqu'à chaque évaluation de la population de taille N, on introduit un ajout de l'ordre de $O(N^2)$. Diverses méthodes au coût moindre ont été développées. Citons le *clustering* (voir Figure 2.6) ou la niche dynamique.

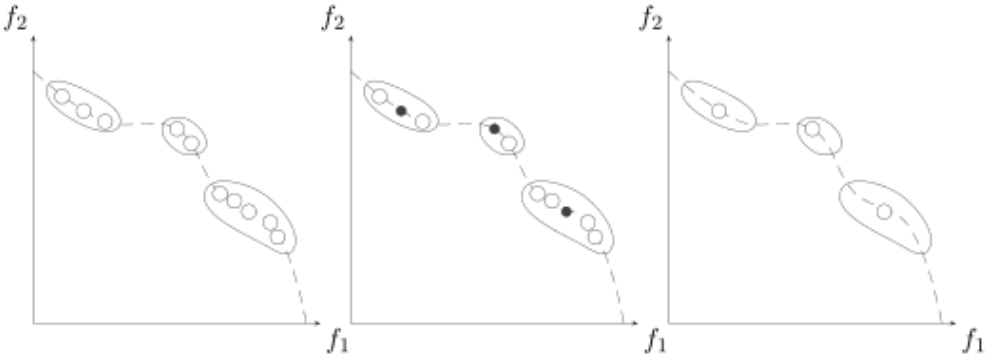


FIGURE 2.6: Illustration du clustering en 3 étapes (de gauche à droite) : identification des groupes, identification de l'individu le plus proche du barycentre, réduction du groupe à cet individu.

Notons également que le succès de cette technique dépend de la taille de la population au regard du nombre d'optima locaux. Si le ratio de la taille de la population sur le nombre d'optima locaux est inférieur à 1 il est clairement impossible d'atteindre l'ensemble de ces optima. Plus ce ratio tend vers 1 par valeur supérieure et plus l'influence du *sharing* est faible puisque les zones peuplées deviendront plus rares au fur et à mesure de la dispersion des individus près des optima (et c'est le rôle du paramètre α que de corriger cette influence).

2.4.2.2 Mesure de surpeuplement : *crowding*

La technique a été introduite par De Jong dès 1975 [10] et a subi de nombreuses améliorations par la suite, notamment par Blickle [11]. On présente ici la mesure de surpeuplement décrite et utilisée par Deb pour l'algorithme NSGA-II dans [12]. L'objectif est de maximiser la somme des distances d'un individu à ces voisins immédiatemment

meilleurs et moins bons pour chacun des critères. Cela implique donc un tri de la population pour chacun des critères.

Pour chaque objectif c, tri de la population.

$$d_c(x_i) = d(x_i, x_{i-1}) + d(x_i, x_{i+1})$$

$$d_{peuplement} = \sum_c d_c(x)$$

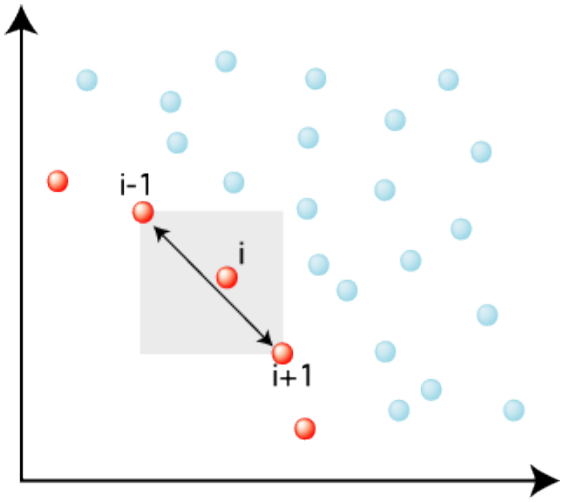


FIGURE 2.7: Illustration de la distance de surpeuplement *crowding*), notamment utilisée par NSGA-II. Il s'agit du périmètre de l'hypervolume (ici de dimension 2) décrit par les individus $i-1$ et $i+1$.

La distance utilisée est très souvent la valeur obtenue pour l'objectif c après normalisation. Auquel cas, la distance de surpeuplement est le périmètre de l'hypercube décrit par les voisins de l'individu concerné.

L'inconvénient de cette mesure est qu'elle nécessite K tris en $Nln(N)$ où K est le nombre d'objectifs et N la taille de la population.

2.4.2.3 K-nearest neighbors

La recherche du k-ième plus proche voisin est un problème courant en algorithmique qui trouve des applications dans des domaines variés comme la reconnaissance de formes, l'approximation de fonctions ou la compression de données.

L'idée est de favoriser les individus dont le k-ième plus proche voisin est éloigné afin d'obtenir une meilleure dispersion des individus. L'implémentation peut amener un surcoût

de calculs important, notamment pour des populations de taille importante. En effet, la recherche naïve, dite « recherche linéaire » est en $O(N)$ pour un unique individu, où N est la taille de la population et donc de l'ordre de $O(N^2)$ pour l'ensemble de la population.
Des techniques plus intelligentes se basent sur le partitionnement de l'espace de recherche, en utilisant des structures de données adaptées (kd-tree, Hilbert-tree, ...) mais ces méthodes ne passent pas à l'échelle à cause de la « malédiction de la dimension » impliquant des performances inférieures à la recherche linéaire dans la cadre d'un nombre trop important de dimensions.

D'autres techniques peuvent être mises en place comme une recherche approximative, en utilisant un algorithme *ϵ-approximate nearest neighbor search* [13].

2.4.2.4 Hypercube

Une autre mesure de diversité consiste à simplement discrétiser, à priori, l'espace de recherche en hypercube et de favoriser les individus dans les hypercubes les moins peuplés.

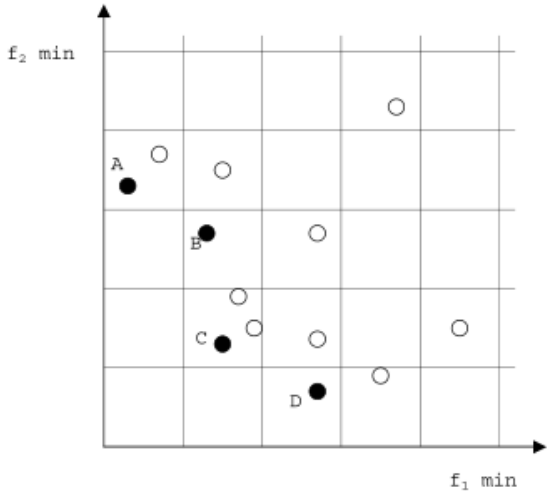


FIGURE 2.8: Illustration de la discrétisation : les points B et D ont une mesure d'encombrement de 0 contre 1 pour A et 2 pour C. En favorisant les individus avec une faible mesure d'encombrement on s'assure d'orienter la recherche vers les zones du Front de Pareto les moins représentées.

La difficulté provient du choix de la discrétisation : comment connaître à priori la taille des hypercubes qui permettent d'éviter la formation d'agglomérats d'individus. De même, la problème de la malédiction de la dimension subsiste.

Des idées d'amélioration pourraient être de rediscrétiser l'espace au cours de l'algorithme afin de l'adapter aux informations obtenues aux générations précédentes (par exemple le nombre moyen d'individus à l'intérieur d'un hypercube, ou la distance moyenne du k-ième voisin), voire d'affiner la discrétisation uniquement aux endroits nécessaires (un petit peu comme un maillage à pas variable dans la méthode des éléments finis), par exemple à l'aide d'un kd-tree. Mais quid du surcoût de calculs par rapport au gain de diversité obtenu ? À ma connaissance, ces propositions n'ont jamais été testées ou étudiées.

2.4.2.5 Contribution à l'hypervolume

Considérons un point r de référence tel que $\forall x \in \Omega, x \succ r$. On définit l'hyper-rectangle $H(x, r)$ comme l'hyper-rectangle de diagonale (x, r). L'hypervolume d'un ensemble $X \in \Omega$ par rapport à un point de référence r, est alors $H_r(X) = |\bigcup_{x \in X} H(x, r)|$. La contribution d'un individu $y \in X$ à l'hypervolume se définit par $CH_r(y) = H_r(X) - H_r(X \setminus y)$.

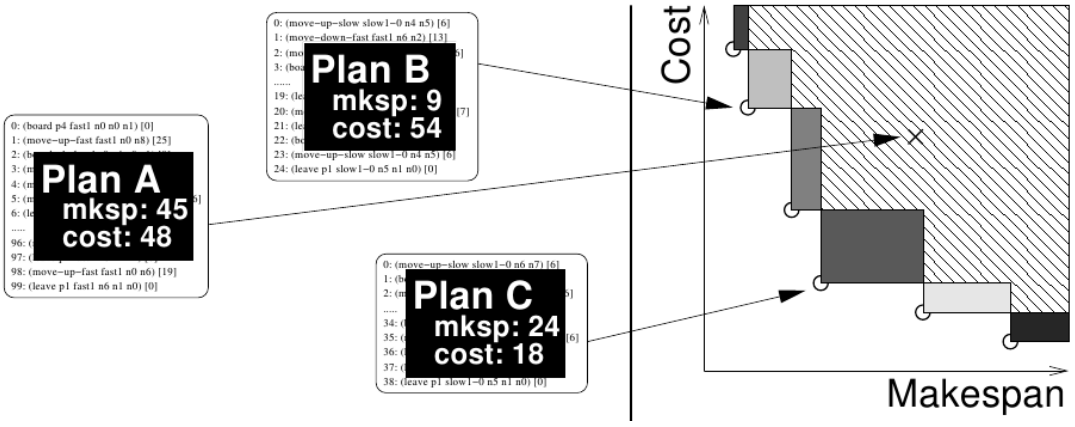


FIGURE 2.9: La surface hachurée représente l'hypervolume de l'ensemble des solutions non dominées. Les surfaces grisées représente la contribution individuelle des solutions à cet hypervolume.

Cette méthode présente un calcul possible en $O(nlogn)$ où n est le cardinal de X pour 2 objectifs. Le passage à l'échelle n'est pas assuré à cause de la « malédiction de la dimension ». De plus, il s'agit d'une mesure phénotypique uniquement.

2.4.2.6 ϵ-dominance

Un individu x ϵ-domine un individu y, noté $x \underset{\epsilon}{\succ} y$ si $\forall i (1+\epsilon) F_i(x) \geq F_i(y)$ dans le cas d'une maximisation ($(1+\epsilon) F_i(x) \leq F_i(y)$ pour une minimisation).

2.5 Les Algorithmes Évolutionnaires Pareto

Différents algorithmes évolutionnaires basés sur l'approximation du front de Pareto on été développés. On peut classer ces algorithmes en plusieurs catégories. Chaque EAMO est caractérisé principalement par sa méthode de sélection et l'utilisation d'une politique élitiste ou non. Parmi les approches Pareto, on peut distinguer deux grandes familles historiques : les algorithmes non-élitistes, qui sont apparus en premier lieu, puis les algorithmes élitistes. Les méthodes non-élitistes ne présentent pas de mécanisme permettant de sauvegarder les individus Pareto-optimaux trouvés au cours de la recherche ou prendre en compte la répartition des individus sur l'ensemble du Front de Pareto.

Ainsi, les algorithmes élitistes se caractérisent par l'utilisation d'une archive ou une population externe pour stocker les individus non-dominés au cours de la recherche, l'utilisation d'une métrique de mesure de la répartition des individus dans l'espace phénotypique et par la préférence pour les solutions non-dominées.

La section suivante propose un aperçu des algorithmes historiques et de leurs caractéristiques. Il ne s'agit évidemment pas d'une liste exhaustive, mais plutôt de présenter les premiers algorithmes introduisant les différents concepts abordés à la section précédente.

2.5.1 MOGA

Proposé par Fonseca et Fleming en 1993 [14], MOGA est l'un des premiers algorithmes évolutionnaires multi-objectifs ayant pour but d'approximer le Front de Pareto.

L'algorithme MOGA pour *Multiple Objective Genetic Algorithm* se base sur un algorithme génétique classique à la différence qu'il va attribuer la *fitness* de chaque individu en fonction du nombre d'individus qui le domine (*dominance count*). L'avantage de cette méthode est la facilite d'implémentation. La contrepartie est la faible répartition

des solutions sur le front de Pareto entrainant jusqu'à une convergence vers une unique solution.

Les performances de l'algorithme peuvent toutefois être améliorées par l'utilisation d'une heuristique de partage.

2.5.2 NGSA

Proposé par Srivinas et Deb en 1993 [15], *Non Dominated Sorting Algorithm* est basé sur un calcul de *fitness* séparant les individus en groupes en fonction du degré de domination de chaque individu (*pareto rank*).

L'heuristique d'évaluation est incrémentale et se déroule comme suit :

- Recherche des individus non dominés dans la population : ils forment la première frontière de Pareto
- Attribution d'une *fitness* à chaque individu, basée sur la frontière, mais avec une correction de type partage pour obtenir une diversité suffisante.
- Suppression de ce groupe de la population.

L'algorithme est répété jusqu'à ce que la population soit vide, en veillant à ce que les individus de la frontière $n + 1$ aient une fitness inférieure à tous les individus de la frontière n.

Malgré une évaluation plus coûteuse par rapport à l'approche MOGA, NGSA permet une meilleure diversité des solutions sur la frontière de Pareto. Notons qu'une caractéristique de NGSA est de ne pas proposer d'archive pour le maintien de la diversité.

2.5.3 NGSA-II

Cette seconde version de l'algorithme NGSA proposé par Deb en 2000 [16] permet de résoudre les problèmes de la première version, à savoir un coût de calcul important, aucun mécanisme d'élitisme et utilisation d'une heuristique de partage. Une amélioration de l'algorithme d'identification des fronts permet d'obtenir une complexité en $O(kN^2)$ avec k le nombre d'objectifs et N la taille de la population. À la place de l'heuristique de *sharing*, Deb utilise une mesure de surpeuplement.

Algorithm 1 Algorithme NSGA-II

Générer une population P_0 de N individus aléatoirement.
while Aucun critère d'arrêt n'est satisfait **do**
  Sélectionner par tournoi des individus de P_i.
  Appliquer les opérateurs de variation pour obtenir de nouveaux individus R_i.
  Calculer toutes les Fronts de $P_i \cup R_i$.
  P_{i+1} est $P_i \cup R_i$ réduit aux N meilleurs individus selon la relation de préférence.
end while

Ainsi, chaque individu reçoit deux indicateurs : son rang r et sa distance de surpeuplement d, qui correspondent respectivement à la qualité et la diversité. Pour définir un ordre total sur les individus, une relation de préférence est défini : $x \succ y \Leftrightarrow (r_x < r_y) \cup [(r_x = r_y) \wedge (d_x > d_y)]$.

2.5.4 NPGA

NPGA pour *Niched Pareto Genetic Algorithm* utilise une sélection sous forme de tournoi un peu particulier. Deux individus sont sélectionnés de manière uniforme dans la population initiale et sont comparés à une sous-population également sélectionnée au hasard, dont la taille t est un paramètre de l'algorithme.

Si l'un des individus domine l'ensemble de la sous-population rentenue, alors il est placé dans la population de la génération suivante. Dans tous les autres cas, une heuristique de partage est appliquée pour déterminer lequel des deux individus va survivre.

Le paramètre t de la taille de la sous-population permet d'affecter la convergence de l'algorithme. Les expérimentations de Horn et Napfliotis, qui ont proposé cette méthode en 1993 [17], montre que si $t \approx 1\%$ de la taille N de la population totale, alors il n'y a aucun élistime car trop de solutions dominées et la convergence sera lente. Au contraire, si $t > 20\%$ alors la pression de la sélection est trop importante ce qui implique une convergence prématurée de l'algorithme qui ne produira pas une bonne approximation du Front de Pareto.

La valeur retenue comme étant susceptible d'allier représentativité de la population et élitisme est d'environ 10% de la taille totale de la population. Èvidemment, cette valeur peut varier au cours de l'algorithme afin de déclencher une convergence de l'algorithme à un moment précis. L'enjeu est donc de trouver une stratégie d'évolution permettant d'ajuster cette valeur au cours de l'algorithme en fonction de l'instance.

2.5.5 SPEA & SPEA-II

Proposé par Zitzler et Thiele en 1998 [18], SPEA pour *Strength Pareto Evolutionary Algorithm* utilise une sélection basée sur le critère de *Dominance Strength* et *Dominance Count*. Cette sélection n'apporte pas de mécanisme pour la diversité mais une technique de *clustering* sur la population pour limiter la taille de l'archive. La sélection s'effectue en tenant compte de l'ensemble population et archive.

Dans SPEA-II, c'est une technique du k-plus-proche-voisin qui est utilisée pour améliorer les performances de l'algorithme originel [19].

2.5.6 PAES & PESA

PAES pour *Pareto Archived Evolution Strategy* a été proposé par Knowles et Corne en 1999 [20]. La particularité de cette méthode est qu'elle n'utilise pas de population mais une recherche locale, basée sur une stratégie d'évoluation parmi les plus simples (1+1) : elle n'utilise qu'un individu à la fois pour la recherche de solution. Elle utilise donc une archive pour stocker les individus Pareto-optimaux courants. Enfin, elle utilise une technique de discrétisation en hypercube de l'espace objectifs.

La discrétisation est adaptative et dépend des bornes phénotypiques de l'archive courante, ce qui ne nécessite pas le réglage à priori du pas de discrétisation. Les auteurs utilisent une structure de données QuadTree ou Octree pour un espace de dimension 2 ou 3.

Les auteurs ont montré qu'une généralisation de (1+1) à un algorithme $(\lambda + \mu)$ n'apportait pas un gain significatif.

Proposée deux ans plus tard par les même auteurs [21], PESA (*Pareto Envelope-based Selection Algorithm*) est une adaptation de PAES aux algorithmes à base de populations. La particularité de l'algorithme est qu'il utilise une archive dont il se sert pour générer une nouvelle population à chaque génération (plutôt que de simplement faire évoluer la population et mettre à jour l'archive au fur et à mesure). Comme PAES, une discrétisation est utilisée et la mesure de diversité est le nombre d'individus présents dans le même hypercube. Par contre, PAES n'utilisait le facteur d'encombrement que

Algorithm 2 PAES avec une archive A

Générer d'une solution candidate c et ajout de c à A.
while Aucun critère d'arrêt n'est satisfait **do**
  Sélectionner d'une solution c' dans le voisinage de s_i.
  if $c' \succ c$ **then**
    Supprimer c de A et ajouter c'.
    $c \leftarrow c'$
  else if $\nexists a \in A, a \succ c'$ **then**
    if A est pleine. **then**
      Ajouter c' à A.
    else if $\exists a \in A$ telle que c' est dans une région moins encombrée que a. **then**
      Ajout de c' à A et suppression de l'élement dans la zone la plus encombrée.
    else if c' est dans une région moins encombrée que c **then**
      $c \leftarrow c'$
    end if
  end if
end while

pour la mise à jour de l'archive alors que PESA l'utilise également pour la sélection des individus.

Notons que les auteurs ont comparés PESA, PAES et SPEA et ont conclu à des résultats statistiquement meilleurs pour PESA sur un nombre de générations donné, tant en terme de diversité qu'en terme de pourcentage de solutions Pareto optimales trouvées..

2.5.7 IBEA

IBEA pour *Indicator Based Evolutionary Algorithm* est, comme son nom le suggère, un algorithme évolutionnaire à base de population. Il a été introduit par Zitzler et Künzli en 2004 [22]. Sa particularité est d'utiliser un indicateur binaire pour la sélection (c'est à dire une fonction $I : \Omega \times \Omega \rightarrow \mathbb{R}$), ce qui lui permet de se passer de technique de maintient de la diversité puisque cet indicateur est censé capturer à la fois la qualité d'un individu en terme de *fitness* mais également en terme de diversité. L'algorithme se déroule ensuite comme un algorithme évolutionnaire très classique.

Les deux variantes les plus utilisées de l'algorithmes sont IBEA_{H^-} et IBEA_{ϵ} qui utilisent respectivement l'indicateur d'hypervolume I_{H^-} et l'indicateur I_{ϵ^-} pour assigner la qualité des individus.

Typiquement, la qualité d'un individu est obtenue en sommant la valeur de l'indicateur par rapport à l'ensemble des autres individus de la population (après normalisation des

vecteurs objectifs) :

$$f(x) = \sum_{y \in P \setminus \{x\}} I(x, y)$$

L'indicateur doit donc mesurer la « perte de qualité » de l'individu x s'il était retiré de la population.

Notons que IBEA utilise une affectation un peu différente : $f(x) = -\sum_{y \in P \setminus \{x\}} e^{\frac{I(x,y)}{cm_x}}$, avec k un facteur d'échelle dépendant de l'indicateur I et $m_x = \max_{y \in P \setminus \{x\}} I(x, y)$. Dans ce sens, on cherche donc à minimiser f.

2.5.8 ϵ-MOEA

ϵ-MOEA introduit en 2002 par Laumans [23] se base sur un critère d'ϵ-dominance. Il s'agit d'un algorithme génétique stationnaire, c'est à dire que les individus obtenus par croisement sont réinsérés dans la population au détriment d'anciens individus, moins bons, qui seront supprimés.

Deb propose une version discrétisée additive en 2003 [24], illustrée par la figure 2.10.

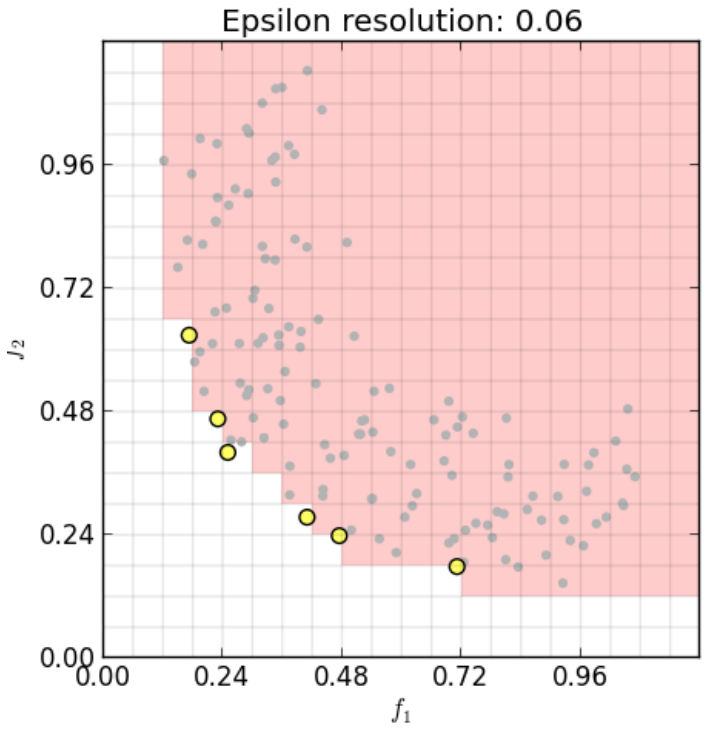


FIGURE 2.10: Illustration de la version discrétisée de ϵ-MOEA.

L'avantage de ϵ-MOEA est son temps de calcul très réduit. L'expérience montre un temps de calcul inférieur d'un facteur 10 comparé à NSGA-II, 20 par rapport à PESA et jusqu'à 100 avec SPEA. Cependant, l'approximation du Front de Pareto est généralement un petit peu moins bonne qu'avec ces algorithmes. De plus, il est nécessaire de fixer à priori les ϵ pour chaque objectif. Au final, ϵ-MOEA offre une alternative de qualité aux algorithmes précités.

2.6 Comparaison d'algorithmes multiobjectifs

2.6.1 Difficultés et critères

La totalité des algorithmes et méthodes proposés sont stochastiques et proposent des mécanismes parfois radicalement différents pour aborder la résolution de problèmes d'optimisation multi-objectifs. Il est donc impossible de définir des critères uniques et une méthodologie certaine pour la comparaison de ces algorithmes, rendant parfois le paysage de l'optimisation stochastique un peu difficile à appréhender en terme d'efficacité. Pire, certains indicateurs peuvent donner des résultats contradictoires rendant l'analyse un peu plus périlleuse.

Une règle d'or se dégage toutefois, à savoir ne jamais tirer aucune conclusion d'une unique exécution (*run*) puisque l'observation d'une trajectoire ne peut donner un résultat significatif.

Il s'agit ensuite de répondre principalement aux questions suivantes, de manière non exhaustive : quelle est la fréquence des découvertes intéressantes ? pour deux ensembles d'approximation donnés, lequel est meilleur et comment quantifier cette différence ? quelle méthode de validation statistiques de ces résultats utiliser ? quels problèmes sont pertinents pour servir de références ? ...

2.6.2 Fonction et surface empirique d'atteinte

Il s'agit d'un méthode présentée par Fonseca et al en 2001 [25] et améliorée en 2005 [26].

Un *run* r atteint $u \in \mathbb{R}^m$ au temps t s'il existe un individu v d'une population au temps $t' < t$ tel quel $v \succeq u$. On note $r(t) \trianglerighteq u$.

On obtient ainsi un estimateur de la probabilité d'atteinte d'un point de l'espace objectif :

$$a_t(u) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{(r_i(t) \trianglerighteq u)}$$

Lorsque le front exact de Pareto est connu, il est possible de tracer le ratio d'atteinte au courants du temps de chacun des points du front et d'identifier les points ou régions plus difficilement accessibles. Il est cependant à noter que cet indicateur est très pénalisant, dans le sens où il faut atteindre **exactement** le vecteur objectif. Il est en effet possible d'être très proche d'un point appartenant au Front de Pareto avec une grande probabilité pour un algorithme A et très loin avec un algorithme B. La fonction empirique d'atteinte ne permet pas de rendre compte de ces différences.

Pour remédier à ce problème, Knowles propose en 2005 [27] une approche plus visuelle, nommée les surfaces d'atteinte. Il s'agit de calculer les enveloppes maximales et minimales de tous les *runs* et de discrétiser l'espace entre les deux enveloppes (par exemple certains quantiles). On estime ainsi la probabilité d'atteinte d'une surface (la surface entre deux quantiles), ce qui permet une meilleure visualisation des performances d'un algorithme et également de rendre compte de la distance moyenne des points du front exact.

Pour comparer deux algorithmes, A et B, à l'aide des surfaces d'atteinte, la procédure est similaire. Il s'agit de calculer les enveloppes miniales et maximales de tous les *runs* (A et B confondus). Après discrétisation, on calcule les fonctions empiriques d'atteinte de A et B et l'on estime la validité des résultats à l'aide d'un test de Kolmogorov. Cette méthode permet de visualiser les différences significatives de performances entre A et B.

2.6.3 Indicateurs d'hypervolume

On rappelle que l'hypervolume d'un ensemble $X \in \Omega$ par rapport à un point de référence r, est alors $H_r(X) = |\bigcup_{x \in X} H(x, r)|$.

À partir de cette quantité, on peut comparer deux algorithmes entre eux ou éventuellement observer les performances d'un algorithme par rapport à un ensemble de référence comme le front de Pareto exact s'il est connu.

On construit donc un indicateur binaire à partir la définition de l'hypervolume :

$$H_r^-(X,Y) = H_r(X) - H_r(Y)$$

Si X est le Front de Pareto exact, plus la valeur obtenue est faible et plus l'on est proche de X, une valeur de 0 signifiant que l'on a atteint X. Suivre l'évolution de l'hypervolume au court du temps permet de tirer des conclusions intéressantes sur la dynamique et les performances d'un algorithme donné.

Dans le cas où l'on ne dispose pas du front exact, et considérant un problème de minimisation, la valeur de l'hypervolume d'un ensemble X supérieure à celle d'un ensemble Y signifie à priori que X est meilleur que Y.

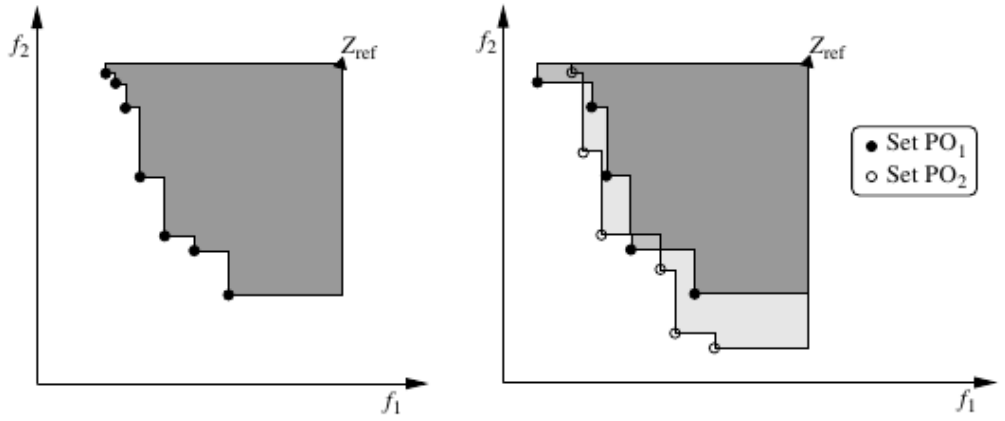


FIGURE 2.11: Illustration de l'hypervolume entre deux ensembles de points.

Chapitre 3

Construction de problèmes multi-objectifs pour les PPTs

3.1 Introduction

Contrairement au problème de planification mono-objectif, la planification multi-objectifs souffre d'un manque de problèmes de référence et d'instances dont le Front de Pareto exact serait connu. Ainsi, tester et comparer les algorithmes de résolution de tels problèmes est presque impossible. Ce chapitre propose une méthode pour générer diverses instances de tailles variables pour le problème de planification temporelle multi-objectifs, en fournissant le Front de Pareto exact et un contrôle sur la forme du front. Un logiciel, le ZenoSolver a été développé et ses capacités et performances sont démontrées à la section 3.3. Enfin, quelques instances singulières de très grandes tailles sont proposées ainsi qu'une tentative de résolution par DaE$_{\text{YAHSP}}$.

3.1.1 L'existant

De nombreux problèmes *benchmarks* paramétrables pour les problèmes d'optimisation multi-objectifs continus existent, notamment le fameux ZDT ou IHR proposés il y a 20 ans), pour lesquels le Front de Pareto exact peut être calculé analytiquement, et les difficultés connues et paramétrées (par exemple la dimension, la forme du front, existence d'optima de Pareto locaux, ...). Pour l'optimisation combinatoire, la situation

est moins favorable et, s'il existe des problèmes de toute taille, leurs Fronts de Pareto ne sont généralement pas connus de manière précise, sauf pour les instances les plus simples (voir par exemple MOCOLIB [1], qui offre de nombreuses instances pour des problèmes de combinatoire bien connus).

Le contexte de la suite de *benchmarks* proposés est le problème de la planification en intelligence artificielle : un domaine de planification est défini par un ensemble de prédicats qui définissent l'état du système, un ensemble d'actions possibles qui peuvent être exécutées lorsque leurs préconditions sont satisfaites, ce qui à pour conséquence d'amener le système dans un nouvel état. Une instance du problème de planification est défini sur un domaine donné par une liste d'objets, utilisés pour instancier les prédicats dans le but de définir un état initial et un état à atteindre. Le but est d'arriver à fournir un *plan faisable*, c'est à dire, un ensemble d'actions qui, une fois appliqué, transforme l'état initial du système en l'état à atteindre.

Un problème de planification simple dans le domaine de la logistique, inspiré par le fameux problème ZenoTravel de la compétition IPC (voir 3.1) implique des villes, des passagers et des avions. Un avion peut voler d'une ville à une autre lorsqu'un lien existe, ce qui implique une durée indiquée par le lien. Un avion peut voler à vide ou prendre un passager à son bord. Une instance du problème est définie par le nombre de villes centrales, les liens entres elles, le nombre de passagers et le nombre d'avions. Le but, en version mono-objectif, est de transporter tous les passagers de la ville I à la ville G avec le *makespan* minimal (c'est à dire la durée total pour tous les avions). Comme les avions peuvent voler de manière concurrente, ce problème *benchmark* appartient à la planification *temporelle*. Un précédent travail [28] a proposé une version multi-objectif de ce *benchmark*, appelé MultiZenoTravel, en ajoutant un *coût* à payer à l'atterrissage : le second objectif est de minimiser le *coût total* du plan. Ce travail avait montré le potentiel de ce problème à fournir des fronts divers et des difficultés variés, mais en ne fournissant le Front de Pareto exact uniquement sur de petites instances.

Au delà de proposer une méthode générique pour calculer le Front de Pareto pour des instances de diverses complexités, MultiZenoTravel *benchmark* permettra de tester différentes méthodes génériques de décomposition (aggrégation par somme pondérée, décomposition de Tchebycheff, approche *Boundary Intersection*) sur des instances larges,

1. http://www.mcdmsociety.org/MCDMlib.html

concaves, non-uniformes, ... d'un problème d'optimisation combinatoire pour lesquelles le front exact est connu.

En premier lieu, nous présentons dans la Section 3.2 formellement le problème MULTI-ZENOTRAVEL, en fournissant quelques propriétés sur les plan optimaux. Basée sur ces propriétés, la Section 3.3 propose l'algorithme ZENOSOLVER pour calculer de manière exhaustive le Front de Pareto de ces instances. Un échantillon de résultats expérimentaux montre la diversité des Fronts de Pareto qu'il est possible d'obtenir et donne quelques mesures de performances sur ces instances de grande taille. Enfin, nous présentons dans la Section 3.4 quelques résultats sur les instances larges proposées précédemment, obtenus par Divide-and-Evolve, le seul solveur de planification évolutionnaire basé sur une approche Pareto[29], et quelques résultats en utilisant une approche par aggrégation via des sommes pondérées sur des instances au front non-convexe.

3.2 MULTIZENOTRAVEL problem

3.2.1 Instances

On introduit ici quelques notations relatives au problème introduit dans la Section **??**. Une instance MULTIZENOTRAVEL (Figure 3.1) est définie par :

- n villes centrales formant une clique non-orientée.
- $c \in \mathbb{R}^n$, où c_i est le coût de passage par la ville c_i [2].
- $D \in \mathbb{R}^n \times \mathbb{R}^n$, où D_{ij} est le temps de vol de la ville c_i à c_j.
- $d^I \in \mathbb{R}^n$, où d^I_i est le temps de vol de la ville c_I à c_i.
- $d^G \in \mathbb{R}^n$, où d^G_i est le temps de vol de la ville et c_i à c_G.
- p avions, initialement en c_I ;
- t personnes, initialement en c_I, devant rejoindre c_G.

Le but est de transporter les t personnes de c_I en c_G, en minimisant la durée totale et le coût du plan.

Afin de rendre l'identification du front exact, nous avons imposé la contrainte suivante :

- Symétrie : $\forall i \in [1, n], d^I_i = d^G_i$. On parlera donc de l'unique vecteur d.

2. Par abus de notation, c_i désigne également la i-ième ville centrale. On dénotera par c_I et c_G respectivement la ville initiale et de destination qui sont chacune reliée à toutes les autres villes.

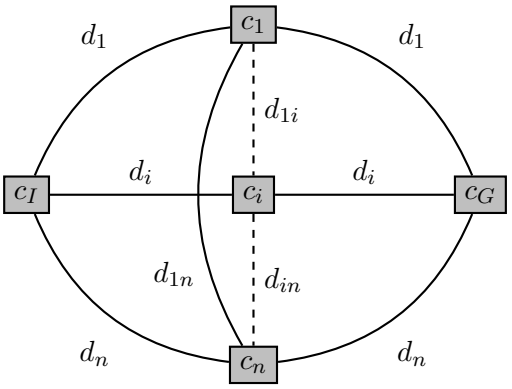


FIGURE 3.1: Une vue schématique d'une instance MULTIZENOTRAVEL.

Sans perte de généralité, on supposera que les paires (d_i, c_i) sont mutuellement exclusives (dans le cas contraire, les deux villes peuvent être « fusionnées » ce qui résulterait en un problème à $n-1$ villes équivalent au problème original).

Nous analysons maintenant quelques propriétés des fronts de Pareto de tels problèmes, ce qui nous permettra de définir un algorithme efficace de résolution en un temps raisonnable.

3.2.2 Front de Pareto

Hypothèse forte : Nous ferons l'unique hypothèse que pour toute paire de villes (i, j), $d_i + d_j < d_{ij}$ [3].

Proposition : Les plans Pareto-optimaux sont les plans pour lesquels exactement $2t - p$ villes sont utilisées.

Preuve : Considérons un plan dans lequel un passager vole de c_i à c_j. En utilisant le même avion, la même personne aurait pû voler de c_i à c_G directement, et l'avion aurait pû revenir vide en c_j. Le plan continue de manière inchangée par la suite, ne créant aucun temps d'attente grâce à l'hypothèse forte sur les durées. Le coût total du plan est inchangé mais la durée totale du plan est inférieure ou égale : le nouveau plan domine donc l'orignal au sens de Pareto.

En itérant le même raisonnement pour chaque personne, et pour chaque avion vide, nous concluons qu'il n'existe pas de vol entre les villes centrales dans des plans optimaux au

3. Cela peut sembler peu réaliste pour un problème réel de logistique. Cependant, nous faisons la conjecture que ce résultat est toujours valable pour l'hypothèse faible, telle que pour chaque triplet c_i, c_j, c_k ($c_0 = I$ and $c_{n+1} = G$), $d_{ik} \leq d_{ij} + d_{jk}$ (inégalité triangulaire).

sens de Pareto. Ainsi, pour amener les t personnes de la ville initiale à la ville finale, chaque personne doit passer par une seule ville : t vols sont nécessaires depuis la ville c_I vers c_i et t vols de c_i vers c_G. Bien sur, $t - p$ vols à vide sont nécessaires, ce qui conduit au résultat. □

3.2.3 Plans Pareto-optimaux

Un Plan Possiblement Pareto-optimal (PPP) est donc défini par :

- $e \in [1, n]^t$, où e_i représente la ville par laquelle va passer la personne i.
- $w \in [1, n]^{t-p}$, où les $(w_i)_i$ représente les villes pour les vols de retour vers c_I.
- Un ordonnancement pour les $4t - 2p$ arcs correspondants. On notera qu'il existe plusieurs ordonnancements faisables, c'est à dire résultant en un plan valide pour p avions.

On considère implicitement que tous les avions vont terminer en C_G car un retour à vide à C_I serait inutile.

Coût : Pour un PPP $C = (e, w)$ donné, le coût de **tout** plan utilisant (seulement) ces villes est déterminé de manière unique.

$$\text{Cost}(C) = \sum_{e_i \in e} c_{e_i} + \sum_{w_i \in w} c_{w_i}$$

Durée d'un PPP : La durée d'un PPP est telle qu'elle est la durée la plus courte d'un plan faisable qui résulte de $4t - 2p$ arcs.

Bornes sur la durée d'un PPP : Une borne maximale triviale pour la durée d'un PPP C, notée $M_S(C)$, est la durée du plan séquentiel (c'est à dire le plan pour lequel un seul avion effectuerait le transport de tous les passagers, et borne minimale triviale, $M_L(C)$, est donnée par le plan « parfait » au sens où aucun avion ne resterait jamais inoccupé.

$$M_S(C) = 2\sum_{i \in E} d_i + \sum_{i \in W} d_i$$
$$M_L(C) = \frac{M_S(C)}{p}$$

3.2.3.1 Borne gloutonne et domination gloutonne

Une meilleure borne supérieure peut rapidemment être obtenue en parallélisant de manière gloutonne les différents vols. Voir plus loin.

Un PPP peut être pénalisé si sa borne minimale est pire que la borne gloutonne d'un autre PPP dont le coût / risque est meilleur.

Domination gloutonne : Pour deux PPP C et C' donnés, C *domine de manière gloutonne* C' si $M_G(C) \leq M_L(C')$ et $Cost(C) \leq Cost(C')$

De nombreux PPP peuvent ainsi être pénalisés de cette manière.

3.2.3.2 Symétrie, admissibilité et cardinalité

Les $2t-p$ villes visitées prennent valeurs dans un ensemble de cardinal n. Il existe donc au plus $n^{(2t-p)}$ PPP possibles. Cependant, un tel ensemble contient un nombre important de doublons qui peuvent être facilement supprimés en ordonnant les indices : **Définition** : Un PPP P est *admissible* si $P \in E \times W$, avec $E = \{e \in [1,n]^t; \forall i \in [1, t-1], d_{e_i} \geq d_{e_{i+1}}\}$ et $W = \{w \in [1,n]^{t-p}; \forall i \in [1, t-p-1], d_{w_i} \geq d_{w_{i+1}}\}$.

Nombre de PPP admissibles : Considérons l'espace Γ_k^m des m-tuples tel que $\forall u \in \Gamma_k^m \quad ||u||_1 = k$. Le cardinal de Γ_k^m est le nombre de combinaisons avec répétitions de k élements dans un ensemble de m élements, c'est à dire $\binom{m+k-1}{k}$.

Il existe une bijection entre l'espace E et Γ_t^n ainsi qu'entre l'espace W et Γ_{t-p}^n. Il existe donc une bijection entre l'ensemble des PPP admissibles et le produit cartésien de Γ_t^n et Γ_{t-p}^n. Ainsi, on peut conclure que le nombre de PPP admissibles est le cardinal de

$\Gamma_t^n \times \Gamma_{t-p}^n$, c'est à dire $\binom{n+t-1}{t}\binom{n+(t-p)-1}{t-p}$.

Exemple : $t = 3$,$n = 3$,$p = 2$. Le nombre de PPP admissibles est 30.

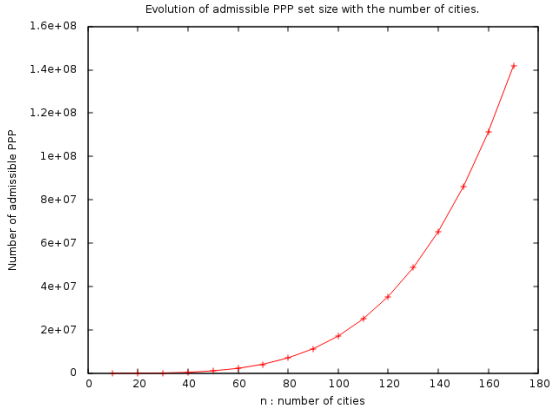


FIGURE 3.2: Évolution de la taille de l'ensemble des PPP admissibles en fonction de n ($t = 3$, $p = 2$).

3.2.4 Calculée la durée optimale

3.2.4.1 Motifs de vol

Il existe trois différents motifs qui peuvent être effectués par les avions, représentés par les figures 3.3, 3.4, 3.5. Nous pouvons noter que le dernier motif nécessite deux avions et est non-préemptif (c'est à dire que lorsque le premier avion dépose une personne dans la ville centrale, l'autre avion peut le récupérer plus tard). Ces motifs proviennent du fait que si un avion est vide, il devrait seulement voler vers l'ouest et vers l'est autrement. Dans le cas contraire, cela voudrait dire que le plan résultant comprendrait plus de villes que nécessaire.

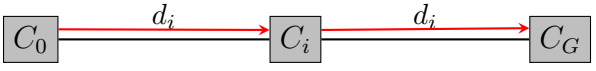


FIGURE 3.3: Motif 1

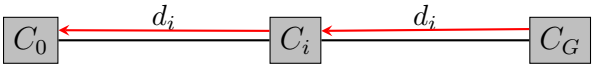


FIGURE 3.4: Motif 2

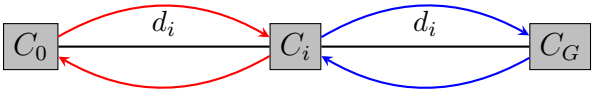


FIGURE 3.5: Motif 3

Notons également que le dernier motif est le seul qui implique une ≪ intéraction ≫ entre les avions. Enfin, pour une ville i donnée, le temps par avion nécessaire pour compléter le motif est le même pour chacun des motifs.

On notera le cardinal du premier motif et du second motif d'un PPP α_E et α_W respectivement et pour le dernier motif β. On réfèrera à la partie ouest et est des motifs 3 par β_E et β_W respectivement.

3.2.4.2 Répartition des motifs pour un PPP admissible

Pour un PPP donné, β peut être encadré comme suit : $0 \leq \beta \leq t - p$. Comme un motif 3 requiert un déplacement depuis c_I vers l'est et un autre depuis c_G vers l'ouest, un encadrement plus fin peut être obtenu par la nombre de composantes de w qui peuvent être associées à un élément de e.

Exemple :

$$C = (3, 1, 1)(2, 1)$$

La première borne supérieure est égale à deux mais en réalité, il est impossible de réaliser un motif 3 avec la ville 2 puisqu'elle n'est pas présente dans le tuple e.

Pour un β donné, le cardinal de chacun de motifs est complètement identifié :

$$\begin{cases} \beta \\ \alpha_W = t - p - \beta \\ \alpha_E = t - \beta \end{cases}$$

Notons que pour un β donné, il existe de multiples choix pour les villes impliquées dans les motifs 3. Chaque choix (liste de villes) sera appelé β_{set}, et l'ensemble de tous les β_{set} possible est appelé le β-PowerSet.

3.2.4.3 La méthode

La durée optimale pour un PPP admissible C et pour un β donné est la durée la plus faible parmi tous les $\beta_{set} \in \beta$-PowerSet. Pour chaque β_{set}, l'idée est de résoudre dans un premier temps un sous-problème obtenu en supprimant tous les Motifs 3 et d'en tenir compte uniquement dans une seconde étape. Les deux étapes peuvent être effectuées par des algorithmes gloutons à faible coût.

Première étape : L'algorithme glouton 3 retourne une durée optimale pour un PPP sans aucun Motif 3 (c'est à dire, avec $\beta = 0$), en allouant les vols les plus longs en premier lieu. Considérons un PPP $C = (e, w)$ et β_{set}. Soient $e' = e \backslash \beta_{set}$ et $w' = w \backslash \beta_{set}$ (obtenus en supprimant les éléments de β_{set} de e et w). Le nouveau PPP $C' = (e', w')$ est par définition sans Motif 3.

Nous avons un nouveau PPP C', avec la répartition suivante des motifs :

$$\begin{cases} \beta = 0 \\ \alpha_W = t - p \\ \alpha_E = t \end{cases}$$

Ainsi, on peut lui appliquer l'Algorithme 3. Comme nous en avons besoin pour la seconde étape, l'algorithme retourne les durées $(D_i)_i$ pour tous les avions. Bien évidemment, la durée optimale de C' est $\max_i(D_i)$.

Exemple :

$t = 5$, $p = 3$, $d = (2, 4, 6)$
$C = (3, 2, 2, 1, 1)(2, 1) \implies 0 \leq \beta \leq 2.$

Le β-PowerSet est $\{\{2, 1\}, \{2\}, \{1\}\}$. Si l'on choisit $\beta_{set} = \{2\}$, alors, $C' = (3, 2, 1, 1)(1)$, avec seulement $p = 4$.

L'Algorithme 3 sur C' retourne les durées suivantes :

Algorithm 3 Calcul optimal de la durée pour $\beta = 0$

$maxE \leftarrow 1\,;\, maxW \leftarrow t+1$ {Start with longest durations both sides}
$D_i \leftarrow 0\ \forall i = 1, \ldots, p$ {'Private' makespan for plane i}
$S_i \leftarrow EAST\ \forall i = 1, \ldots, p$ {Direction for next flight of plane i}
while $maxW \leq 2t - p$ **do**
  $k \leftarrow \mathrm{ArgMin}_i(D_i)$ {Schedule next in line (in c_I or c_G)}
  if $S_k = EAST$ **then**
    $D_k \leftarrow D_k + 2d_{e_{maxE}}$
    $S_k \leftarrow WEST\,;\, maxE \leftarrow maxE + 1\,;$
  else
    $D_k \leftarrow D_k + 2d_{w_{maxW}}$
    $S_k \leftarrow EAST\,;\, maxW \leftarrow maxW + 1\,;$
  end if
end while
while $maxE \leq t$ **do**
  $k \leftarrow \underset{i, S_i = EAST}{\mathrm{ArgMin}}\ (D_i)$
  $D_k \leftarrow D_k + 2d_{c_{maxE}}$
  $maxE \leftarrow maxE + 1\,;$
end while
return $(D_i)_i$ {All private makespans are needed for the second step}

p_1 :	$D_1 = 12$
p_2 :	$D_2 = 8$
p_3 :	$D_3 = 12$

Remarque : L'algorithme écrit part du principe que les d_i sont décroissants mais il est simple de l'écrire de sorte à ne plus avoir cette contrainte. La manière efficace de faire si l'on traite l'ensemble des PPP admissibles est de trier d et de déplacer également en conséquence les c_i.

Second étape : Cette étape consiste à dispatcher les différents Motifs 3 entre les avions, en tenant compte de leur durée propre $(D_i)_i$ retournée par la première étape. Cette distribution se fait de manière séquentielle et gloutonne, en assignant le Motif 3 de plus longue durée en premier, aux deux avions avec la durée courante la plus faible.

Cependant, au sein du Motif 3, si le second avion atterrit dans la ville centrale avant que la personne n'ait été amenée là par le premier avion, l'avion doit attendre et ainsi, il est possible que la durée du plan ne soit pas une simple distribution gloutonne de toutes les durées.

Donc, la durée optimale pour des plans réalisables avec ce PPP et ce β_{set} est 20.

Proposition : On peut toujours construire un plan valide avec la durée retournée par l'algorithme.

Preuve par construction :

Chaque motif 3 a un temps pré-requis pour ne pas avoir de temps d'attente qui est $T_i = d_i$ où c_i est la ville impliquée dans le motif 3. Considérons un Motif 3 effectué par les avions p_1 et p_2, au travers de la ville c_i. Leur planning ressemblerait à :

$$p_1 : \quad c_I \to \ldots \to c_I \to \overset{\Diamond}{c_i} \to c_I \to \ldots \to c_G$$
$$p_2 : \quad c_I \to \ldots \to c_G \to \overset{\blacklozenge}{c_i} \to c_G \to \ldots \to c_G$$

où $\Diamond$ marque le moment où p_1 dépose la personne dans la ville c_i et $\blacklozenge$ marque un possible point d'attente pour l'avion p_2. L'idée est de placer tous les $\Diamond$ le plus tôt possible dans le plan, en planifiant toutes les parties ouest des Motifs 3 dès que possible et de placer les $\blacklozenge$ le plus tard possible, en planifiant la partie est le plus tard possible.

Considérons uniquement des avions tels qu'ils vont effectuer au moins un motif 3 (autrement, la construction est triviale).

1. On sélectionne l'avion avec le nombre maximum de Motifs 3 à effectuer. En cas d'égalité, on sélectionne l'avion avec le temps de pré-requis le plus haut. En cas d'égalité, on sélectionne l'avion avec la durée totale la plus longue.
2. On commence la construction par les motifs 1 et 2 uniquement.
3. On regarde l'ensemble des motifs 3 qui ne sont pas encore démarrés et pour chaque élément de cet ensemble, on ajoute la partie est à la fin du plan de l'avion par ordre décroissant de temps pré-requis.
4. On regarde l'ensemble des motifs 3 qui sont déjà démarrés et pour chaque élément de cet ensemble, on ajoute la partie ouest au début du plan de l'avion considéré, par ordre croissant du temps réel pré-requis. Le temps réel de pré-requis pour un motif 3 donné est le temps maximum tel que il n'y a pas de temps d'attente.

Cette valeur peut être calculée grâce au plan de l'autre avion impliqué dans le motif 3.

Proposition : La durée retournée par l'algorithme est optimal pour un PPP C et un β_{set} donné.

Preuve :

— Le temps incompressible pour transporter l'ensemble des passagers, en accord avec un β_{set} donné, est $T = 4 \sum_{i \in \beta_{set}} d_i + 2 \sum_{i \in \{E', W'\}} d_i$. Un plan théoriquement optimal est donc un plan pour lequel la répartition de ce temps est optimale entre chaque avion (c'est à dire tel que aucun temps d'attente n'apparait, pour aucun avion).

— L'algorithme ci-dessus donne un répartition optimale des temps entre les p avions. Ainsi, si un plan peut être construit+ à partir des durées fournies, alors il est optimal pour un PPP C et un β_{set} donné.

— Comme il existe une méthode de construction d'un telle plan, on peut conclure que l'algorithme retourne une durée optimale. On peut aussi conclure que dans ce cas, il est toujours possible de construire un plan sans aucun temps d'attente.

□

Exemple : $t = 7$, $p = 3$, $d = (2, 4, 6), C = (3, 3, 2, 2, 2, 1, 1)(3, 3, 2, 1), 0 \leq \beta \leq 3$.
En choissant $\beta_{set} = \{3, 2, 1\}$ on obtient le sous-problème $C' = (3, 2, 2, 1)(3)$ avec $t' = 4$. En résolvant C' (Étape 1) retourne les durées D_i^1 dans le tableau ci-dessous. En ajoutant les Motifs 3, cela donne un plan complet, avec les durées D_i^2.

Étape 1 sans Motif 3

p_1	$D_1 = 12$
p_2	$D_2 = 24$
p_3	$D_3 = 8$

Étape 2 incluant les Motifs 3

p_1	$D_1 = 32$
p_2	$D_2 = 28$
p_3	$D_3 = 32$

p_i	D_i^1	D_i^2	Motifs 3
p_1	12	32	2
p_2	24	28	1
p_3	8	32	3

$$p_3 : \quad c_I \to c_2 \to c_G \to \overset{\blacklozenge_3}{c_3} \to c_G \to \overset{\blacklozenge_2}{c_2} \to c_G \to \overset{\blacklozenge_1}{c_1} \to c_G$$
$$p_2 : \quad c_I \to \overset{\lozenge_1}{c_1} \to c_I \to c_2 \to c_G \to c_3 \to c_I \to c_2 \to c_G$$
$$p_3 : \quad c_I \to \overset{\lozenge_3}{c_3} \to c_I \to \overset{\lozenge_2}{c_2} \to c_I \to c_3 \to c_G$$

Donc, il n'y a pas de temps d'attente entre les Motifs 3 et la durée optimale d'un plan est 32 pour ce PPP.

Complexité : Dans le pire des cas, pour un PPP donné, $w \subset e$ et $w_i \neq w_j$ si $i \neq j$. Donc pour chaque valeur de β il y a $\binom{t-p}{\beta}$ choix possibles de β_{set}. Comme $0 \leq \beta \leq t-p$, il faudra effectuer 2^{t-p} *itérations* (c'est à dire déterminer la durée optimale pour un PPP et un β_{set} donnés). La complexité dans le pire des cas est donc $2^{t-p}\binom{n+t-1}{t}\binom{n+(t-p)-1}{t-p}$.

Remarque : En pratique on atteint le pire des cas en nombre d'itérations pour très peu de PPP admissibles. Le nombre de PPP admissibles tels que toutes les valeurs de w soient différentes et présentes dans e (la pire configuration) augmente cependant avec n et p et diminue avec t.

3.3 ZenoSolver

ZenoSolver est un logiciel développé en `C++11` dédié à la génération et résolution exacte des instances du problème MultiZenoTravel. Dans un premier temps, il permet de régler l'ensemble des paramètres de l'instance afin d'ajuster la difficulté ou d'obtenir des Front de Pareto différents. En particulier, les vecteurs c et d sont générés en utilisant deux fonctions définies par l'utilisateur, f et g, telles que $c_i = x_c f(i) + y_c$ et $d_i = x_d g(n-i) + y_d$, s'assurant alors que les deux objectifs sont antagonistes. Les facteurs d'échelle (a_c, a_d) et de translation (b_c, b_d) donne un contrôle plus fin de la forme du Front et sont définis par défaut, respectivement, à 1 et 0. Secondemment, ZenoSolver permet de générer le fichier `PDDL` correspondant à l'instance, ce qui permet d'être directement utilisé par tous les logiciels standards de planification. Enfin, ZenoSolver calcule le Front de Pareto exact, en utilisant l'algorithme décrit plus haut, en itérant sur l'ensemble $E \times W$, en stockant pour chaque valeur du coût total, le PPP avec la meilleure durée optimale connue à ce jour, ce qui permet d'éviter une construction explicite de

l'ensemble des PPP admissibles. En utilisant la propriété de domination gloutonne, ZenoSolver implémente une méthode d'élagage qui vérifie si le PPP courant est dominé par un autre PPP déjà stocké. Comme la durée optimale est inférieure ou égale à la durée de la borne supérieure M_S, cela conduit à un élagage plus efficace. En effet, comme les PPP sont générés dans un ordre de coût approximativement croissant, cela évite d'itérer sur l'ensemble des PPP pour vérifier la relation de dominance.

Algorithm 4 ZenoSolver : main algorithm.

```
Initialise S a map indexed by costs.
for all e ∈ E do
  for all w ∈ W do
    if PPP made from e and p is not dominated regarding S then
      Calculate optimal makespan for this PPP.
      Insert the makespan if it is better than the previous for the same cost.
    end if
  end for
end for
Extract the exact Pareto front from S.
```

Déterminer si le PPP courant est dominé peut être effectué en $O(h)$ où h est le nombre de différents coûts existant. L'insertion est en $O(log\ h)$ mais peut être réduite à $O(1)$ en utilisant un *indice* sur l'insertion (notamment car l'on connait l'ordre par coût approximativement croissant de génération des PPP). Une borne supérieure évidente sur h est donnée par $(2t-p)(\max_i(c_i) - \min_i(c_i))$. La mémoire requise est également en $O(h)$, ce qui est approximativement la taille du Front de Pareto (voir Table 3.6).

3.3.1 Les algorithmes

3.3.1.1 Génération des PPP admissibles

Les PPP admissibles sont le résultat du produit cartésien entre l'ensemble des combinaisons avec répétitions de t objets parmi $n-t+1$ et l'ensemble des combinaisons avec répétitions de $t-p$ objets parmi $n-t+p+1$. Autrement dit, l'enjeu est de générer de manière efficace l'ensemble des combinaisons avec répétitions.

Pour cela, l'algorithme implémenté est celui décrit par Donald Knuth dans [30]. En plus de donner d'excellentes performances, il permet la génération séquentielle des combinaisons par ordre lexicographique par exemple (et dans notre cas par ordre de coûts

de PPP presque croissant). Cette génération séquentielle est très utile pour des optimisations mémoires, permettant par exemple d'éviter de stocker l'ensemble des PPP disponibles (en créant un itérateur « produit cartésien » sur les deux ensembles.)

3.3.1.2 Génération de l'ensemble des parties d'un ensemble

Pour un PPP donné, il faut pouvoir générer le β-PowerSet. Le cardinal de cet ensemble, dans le pire des cas, est 2^{t-p} ce qui croît de manière exponentielle.Étant donné le nombre important de fois où un tel ensemble va être généré, il est important d'avoir un algorithme efficace de génération de l'ensemble des parties d'un ensemble.

L'algorithme implémenté se base sur une représentation binaire d'un ensemble, avec comme contrainte que deux ensembles de même cardinalité ont le même nombre de bits à 1 (Snoob, pour Same Number of One Bit en anglais). Il est ainsi facile de générer au moins un sous-ensemble E_i d'un ensemble E donné pour chaque cardinalité possible (c'est à dire $\forall i \in [0, card(E)]$) d'une partie de cet ensemble E. Pour un E_i donné, des opérations bits à bits permettent facilement d'itérer sur l'ensemble des parties de l'ensemble E de cardinalité i.

L'astuce est décrite pour la première fois par Henry S. Warren, Jr. et très utilisé notamment dans les algorithmes d'intelligence artificielle pour les échecs.

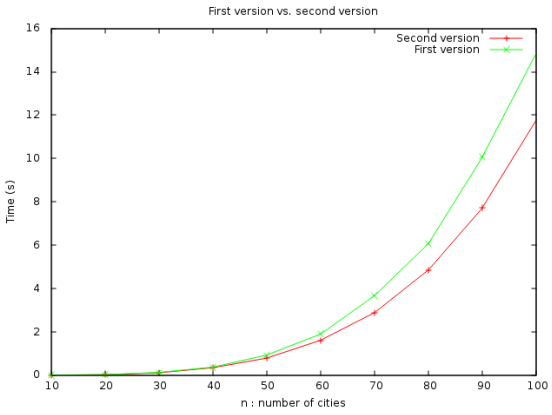
3.3.2 Performances empiriques

Complexité empirique[4] : Le nombre d'itérations (Section 3.2.4) dans le pire des cas est $2^{t-p}\binom{n+t-1}{t}\binom{n+(t-p)-1}{t-p}$. En pratique cependant, ce nombre est non seulement influencé par le nombre de PPP mais aussi et surtout par leurs structures : augmenter n ne change pas en général de manière significative le nombre d'itérations (la borne supérieure est 2^{t-p}, indépendante de n). Cependant, augmenter t augmente cette borne supérieure et augmente très fortement le nombre moyen d'itérations par PPP.

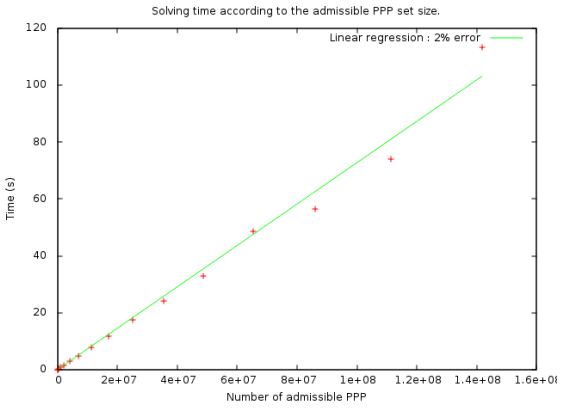
Des expériences préliminaires utilisaient une version de l'algorithme avec une instanciation explicite de l'ensemble $E \times W$, résultant en un manque de mémoire très rapide, même pour les instances de taille moyenne.

4. Résultats détaillés : https://descarwin.lri.fr/tiki-index.php?page=Multi-Zeno.

Le graphique ?? montre l'évolution du temps de résolution de ces instances. On compare ici la version originale (construisant explicitement l'ensemble des PPP) de l'algorithme avec une version optimisée en mémoire.



Il s'agit d'une moyenne de 30 résolutions pour chaque taille de problème et par algorithme. En plus d'être très peu gourmande en mémoire, la seconde version semble également plus rapide. Évidemment, on observe également que le temps de résolution croit de manière exponentielle.



Le graphique ?? montre le rapport entre le temps de résolution et la taille de l'ensemble des PPP admissibles. Une régression linéaire au sens des moindres carrés indique une erreur de 2%. L'évolution de la résolution est donc purement linéaire par rapport à la taille de l'ensemble des PPP admissibles. Ce n'est pas le cas pour la version non-optimisée de l'algorithme à cause du phénomène de *swap* à partir d'un certain nombre de villes.

Élagage ou non ? : L'efficacité de la méthode d'élagage semble être dépendante de l'instance. En fixant n, t et p, différents fonctions génératrices résultent en différents

nombres d'itérations et temps CPU. Les bénéfices de la méthode d'élagage repose très fortement sur le nombre moyen d'itérations par PPP et donc, est bien plus efficace lorsque l'on augmente t. De plus, augmenter n avec une valeur de t petite, tout en utilisant l'élagage peut dégrader les performances en terme de temps CPU, quand bien même le nombre d'itérations diminue (et peut devenir inférieur au nombre de PPP).

Il y a cependant un certain nombre de cas où l'efficacité de l'élagage est claire. Par exemple, avec $n = t = 9$: ZenoSolver implique 1.26 10^9 itérations et 2222 secondes sans élagage. En ajoutant l'élagage, avec $f(i) = \sqrt{i}$ et $g(i) = i$ il n'effectue plus que 119000 itérations en 26 secondes, ou 36000 itérations et 53 secondes pour $f(i) = log(i)$ et $g(i) = \sqrt{i}$. De plus amples résultats sont disponibles dans les Tables 3.1, 3.2, 3.3, 3.4, 3.5.

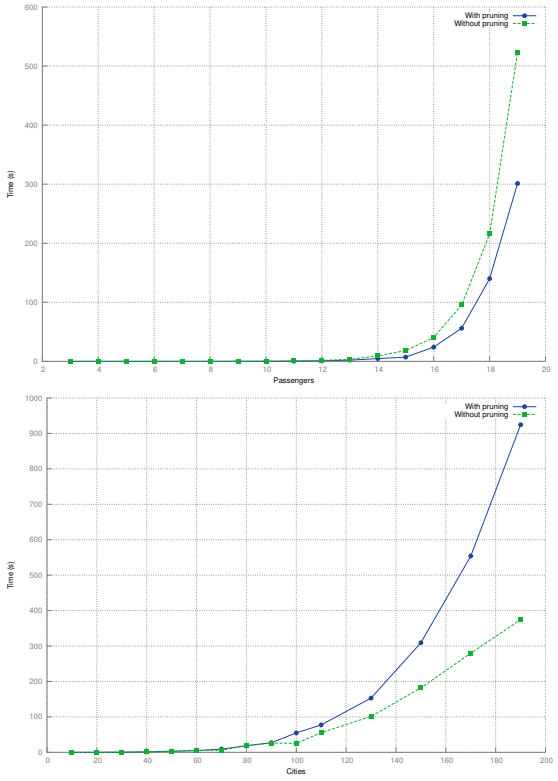


FIGURE 3.6: Temps en fonction du nombre de passagers t ou de villes n.

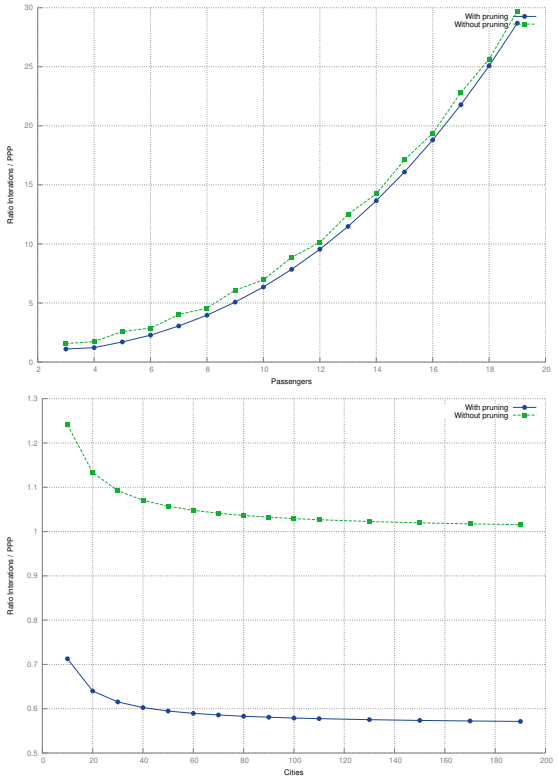


FIGURE 3.7: Ratio du nombre d'itérations sur le nombre de PPP, en fonction du nombre de passagers t ou de villes n.

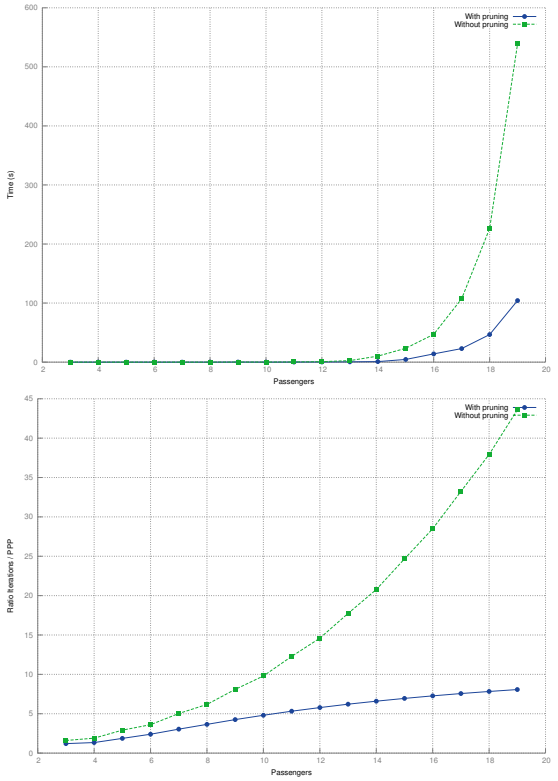


FIGURE 3.8: Temps et ratio pour une instance concave.

TABLE 3.1: Augmentation simultanée de n et t avec $f(i) = g(i) = i$.

n	t	p	PPP Size	Iterations	S Size	Front Size	Time (ms)
3	3	2	30	33	9	5	0
4	4	2	350	408	19	10	1
5	5	2	4410	6387	33	17	6
6	6	2	58212	109831	51	26	117
7	7	2	792792	1930385	73	37	2278
8	8	2	11042460	34648348	99	50	43572
9	9	2	156434850	630225670	129	65	1036772
10	10	2	2245709180	11600589455	163	82	20785211

TABLE 3.2: Augmentation simultanée de n et t avec $f(i) = g(i) = log(i)$.

n	t	p	PPP Size	Iterations		S Size	Front Size	Time (ms)	
3	3	2	30	48	36	15	5	0	0
4	4	2	350	770	314	74	10	0	0
5	5	2	4410	12810	1323	389	19	12	3
6	6	2	58212	219912	3050	890	29	207	25
7	7	2	792792	3868032	6376	1322	36	6613	276
8	8	2	11042460	69320988	12226	1785	54	127332	3918
9	9	2	156434850	1260758070	26128	2276	77	2283654	58104
10	10	2	2245709180	23202174710	50428	2781	111	46723743	955131

TABLE 3.3: Augmentation simultanée de n et t avec $f(i) = log(i)$ and $g(i) = \sqrt{i}$.

n	t	p	PPP Size	Iterations		S Size	Front Size	Time (ms)	
3	3	2	30	48	36	15	5	0	0
4	4	2	350	770	368	83	10	0	0
5	5	2	4410	12810	2297	409	19	11	9
6	6	2	58212	219912	4881	857	36	207	24
7	7	2	792792	3868032	8956	1349	37	6237	266
8	8	2	11042460	69320988	19888	1856	65	120383	3526
9	9	2	156434850	1260758070	36756	2482	83	2240856	52954
10	10	2	2245709180	23202174710	65359	3144	104	46216951	1161831

TABLE 3.6: Échantillon d'instances larges : paramètres & statistiques de génération.

Inst.	n	t	p	Generating functions		Pareto#	h	PPP(k)	Iter.(k)	Time
1	20	6	2	$\frac{5}{2}i+\frac{(i \bmod 2)}{10}$	$\frac{5}{2}i+\frac{(i \bmod 2)}{10}$	409	4015	1568220	3317140	16h46
2	3	21	2			61	861	53	233	2006s
3	10	10	3			383				
4	200	3	2	$\sqrt{i}$	$\sqrt{i}$	538	4963	270680	3906	1845s
5	200	3	2	$\sqrt{i}$	i	399	797	270680	32066	1666s
6	8	26	25	$\sqrt{i}$	i	15	190	34176	60457	4240s

TABLE 3.4: Augmentation simultanée de n et t avec $f(i) = \sqrt{i}$ and $g(i) = i$.

n	t	p	PPP Size	Iterations		S Size	Front Size	Time (ms)	
3	3	2	30	48	36	9	5	0	0
4	4	2	350	770	419	19	10	0	0
5	5	2	4410	12810	3014	33	17	10	3
6	6	2	58212	219912	14739	51	26	223	25
7	7	2	792792	3868032	48512	73	37	6510	163
8	8	2	11042460	69320988	47838	99	65	115977	1709
9	9	2	156434850	1260758070	119686	129	65	2222907	26661
10	10	2	2245709180	-	195214	163	110	-	702926

TABLE 3.5: Augmentation simultanée de n et t avec $f(i) = g(i) = \frac{5}{2}i + \frac{(i \bmod 2)}{10}$.

n	t	p	PPP Size	Iterations		S Size	Front Size	Time (ms)	
3	3	2	30	48	30	15	6	0	0
4	4	2	350	770	622	84	20	0	0
5	5	2	4410	12810	2245	369	31	12	3
6	6	2	58212	219912	172759	890	115	222	349
7	7	2	792792	3868032	166183	1875	121	6639	785
8	8	2	11042460	69320988	51806458	3335	366	114788	235221
9	9	2	156434850	1260758070	10530973	5435	233	2154571	176702

3.3.3 Échantillon d'instances larges

Les Figures 3.9,3.10,3.11,3.12,3.13,3.14 montrent le Front de Pareto exact de six instances larges avec différentes complexités et différentes formes, sélectionnées pour servir de référence. La Table 3.6 donnent les paramètres utilisés par ZENOSOLVER pour les constuire, ainsi que certaines statistiques sur la génération. Aucune instance n'est linéaire (même si la tendance globale semble linéaire). Notons également la distribution non-uniforme des points de l'instance 6 et le peu de points de l'instance 6 malgré la complexité de l'instance (26 personnes), à cause du ratio faible $\frac{p}{t}$.

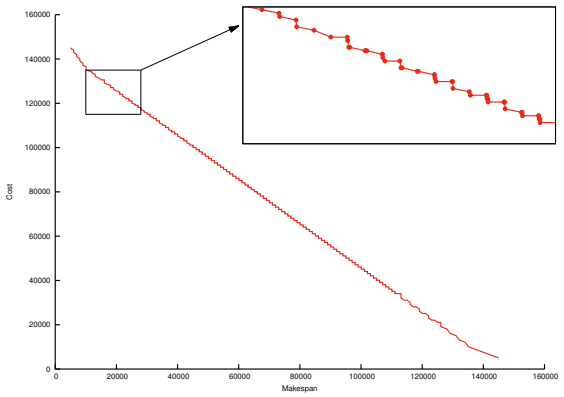


FIGURE 3.9: Instance 1 : la tendance générale apparaît linéaire, avec certaines parties convexe sur les extrémintés. Cependant, un zoom montre un front totalement destructuré.

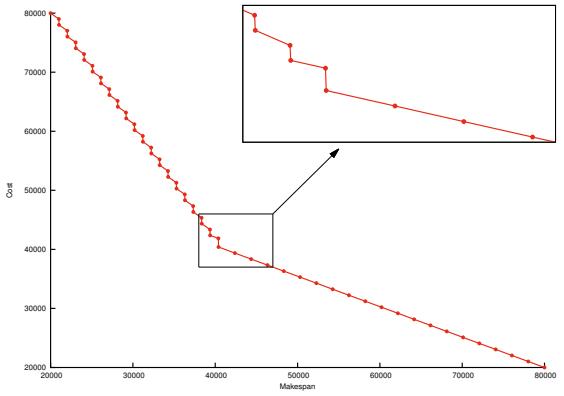


FIGURE 3.10: Instance 2 : une tendance convexe. Le partie inférieure est totalement linéaire alors que la partie supérieure est faite de motifs concaves réguliers.

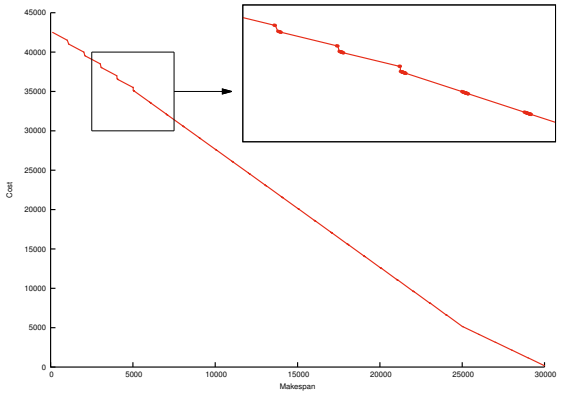


FIGURE 3.11: Instance 3 : à peu près linéaire, un zoom montre une répartition non uniforme des points. Les amas de points sont cependant régulier sur le front global.

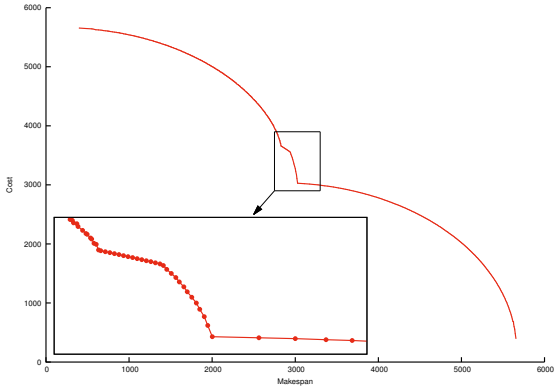


FIGURE 3.12: Instance 4 : une tendance concave. La partie inférieure est à peu près régulière contrairement à la partie supérieure qui montre une répartition non uniforme et non régulière des points. On note également une petite partie concave au milieu qui casse la symétrie de la tendance générale.

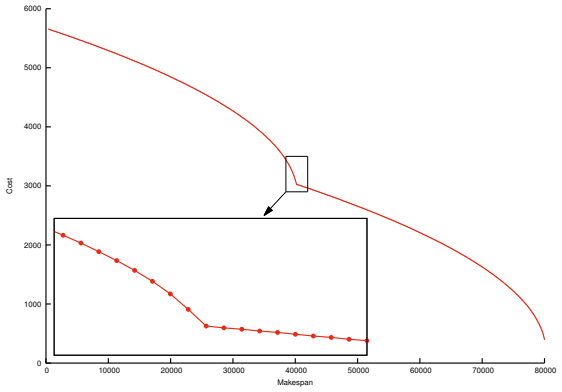


FIGURE 3.13: Instance 5 : ressemble à l'instance 3. Cependant, la distribution des points est régulière et uniforme.

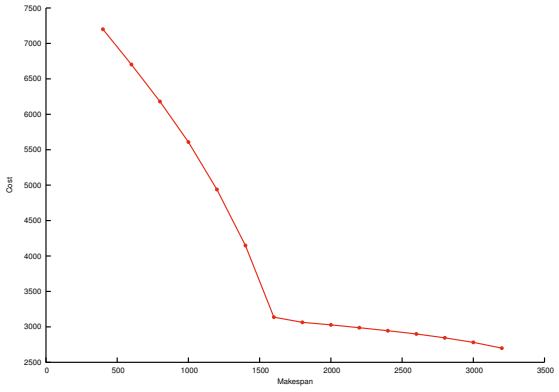


FIGURE 3.14: Instance 6 : seulement 15 points en dépit de la complexité de l'instance.

3.4 Expériences Multi-objectifs

3.4.1 Conditions expérimentales

L'article précédent [28] a montré que IBEA$_{H-}$ obtient statistiquement les meilleurs résultats parmi divers algorithmes évolutionnaires pour DAE$_{\text{YAHSP}}$. Ainsi, toutes les

expériences ont été conduites avec IBEA$_{H^-}$. Les paramètres internes de DAE$_{\text{YAHSP}}$ont été optimisés grâce à la plateforme de méta-optimisation PARAMILS en utilisant la métrique de différence d'hypervolume (*Hypervolume Difference Indicator*) H^- comme détaillé dans [31]. Pour chaque instance, 20 expériences indépendantes (*run*) on été effectuées avec un temps de 5400 seconds par run (1800 pour l'instance 6). Toutes les calculs de performance, comparaisons et d'indicateurs ont été effectués grâce à PISA [5].

3.4.2 DAE$_{\text{YAHSP}}$ sur les instances larges

La surface d'atteinte sur l'instance 1 (Figure 3.15-gauche) montre une bonne atteinte du Front de Pareto exact, même si seuls quelques points sont atteints (plus la zone est foncée plus la probabilité d'atteindre cette région est grande - une zone blanche indique une partie jamais atteinte par aucun des 20 runs). Cependant, la surface d'atteinte est uniforme sur l'ensemble du front. Les surfaces sur les instances 2 et 3 sont loins du front exact et seuls 2 points sont trouvés pour l'instance 3 sur les 383 composant le front. Au contraire, même avec un budget temporel restreint, la plupart des points du front de l'instance 6 sont trouvés, à l'exception de certains dans la zone la plus concave.

On peut remarquer que, même si n est plus grand sur l'instance 6 que sur l'instance 2, l'ajout d'avions résulte en un front plus facile à atteindre. Ce résultat est plutôt surprenant puisque l'espace de recherche de DAE$_{\text{YAHSP}}$ augmente avec la valeur de p. Une raison à cela pourrait être le nombre maximum d'états dans une décomposition, qui est fixée à 20 et pourrait être trop faible pour obtenir des sous-problèmes facile à résoudre, résultant ainsi en des résultats décevants sur les instances 2 et 3.

3.4.3 Pareto vs Aggrégation par somme pondérée

Enfin, nous proposons un bref aperçu de résultats comparant la version multi-objectif de DAE$_{\text{YAHSP}}$ et sa version mono-objectif utilisant une aggrégation par somme pondérée des objectifs. L'instance choisie est concave, similaire à l'instance 4 ci-dessus, mais avec uniquement 30 villes, résultant en un Front de Pareto de 66 points). Les conditions expérimentales sont les même que dans l'article [31]. Un run pour l'aggrégation consiste en 11 runs indépendants, avec un paramètre de point α prenant valeur de 0 à 1 par pas

5. http ://www.tik.ee.ethz.ch/pisa/

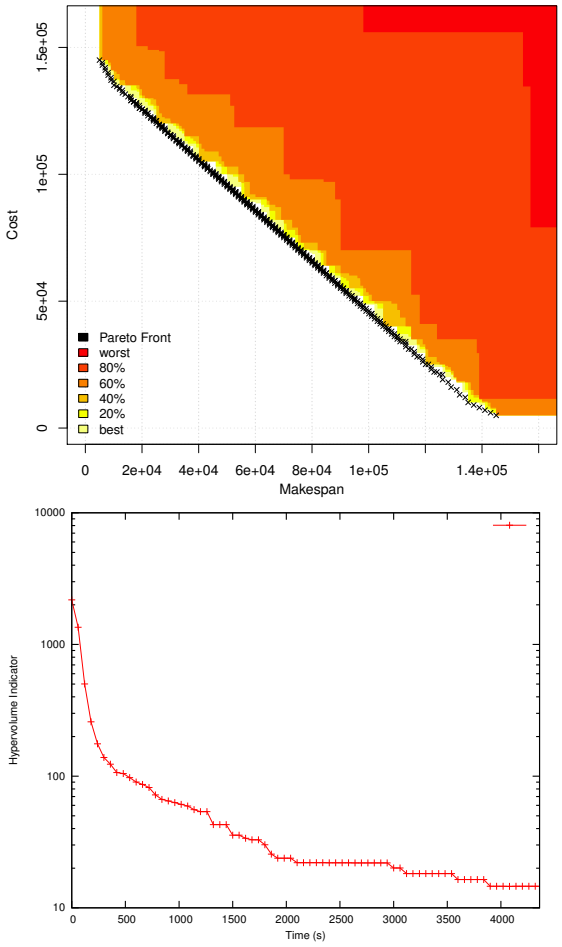


FIGURE 3.15: Surface d'atteinte et indicateur d'Hypervolume pour l'instance 1.

de 0.1. Cependant, lors que chaque run pour l'approche Pareto bénéficie d'un budget temporel de 5400s, le même temps est alloué pour chaque valeur de *alpha*. Ainsi, le budget total pour la version utilisant l'aggrégation est 11 fois supérieur à celui de l'approche Pareto.

Malgré cet énorme avantage, la surface d'atteinte (Figure **??**) montre que dans le cas de l'approche Pareto, le font exact est déjà déterminé et visible après 900s, même en ne considérant que le pire des 20 runs. Au contraire, même le meilleur des 20 runs est encore très loin du front exact, sauf pour quelques points qui n'appartiennent pas aux parties concaves, comme confirmé par les fronts cumulés (fusion des fronts obtenus par les 20 runs, sur la Figure 3.22).

Ces tendances, bien que préliminaires, confirment cependant le fait bien connu selon lequel une aggrégation par somme pondérée a des difficultés à atteindre les parties concaves des Front de Pareto.

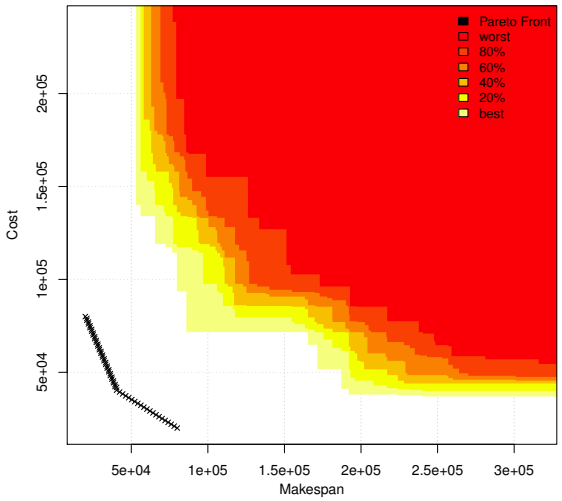


FIGURE 3.16: Surface d'atteinte pour l'instance 2.

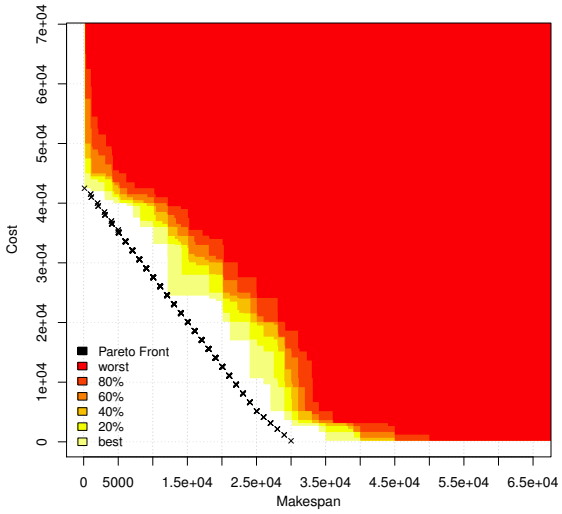


FIGURE 3.17: Surface d'atteinte pour l'instance 3.

La différence de surface d'atteinte indique les surfaces les plus probables d'être atteinte par un algorithme comparé à un autre. Sur cette instance concave, les parties en faveur de l'approche Pareto sont toujours plus proche du front exact que celles en faveur de l'aggrégation. Ainsi, il est clair que l'approche Pareto est bien plus performante que l'aggrégation et ceci montre la difficulté de l'aggrégation à atteindre les parties concaves du Front de Pareto.

3.4.4 Étude de concavité

Le but de cette étude est de caractériser la difficulté d'une instance à être résolue en considérant la concavité du Front de Pareto. Plus précisemment, nous comparons des

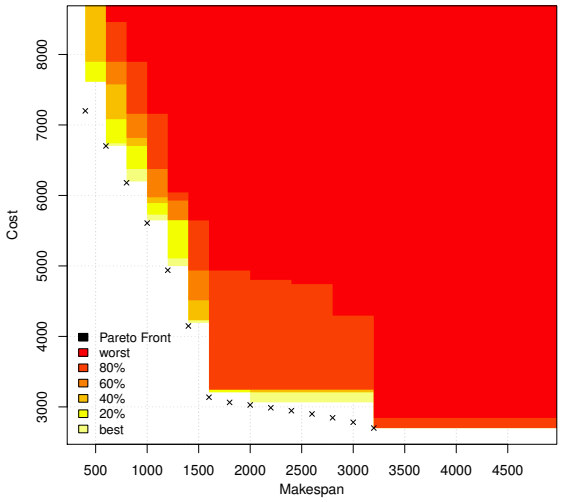


FIGURE 3.18: Surface d'atteinte pour l'instance 6.

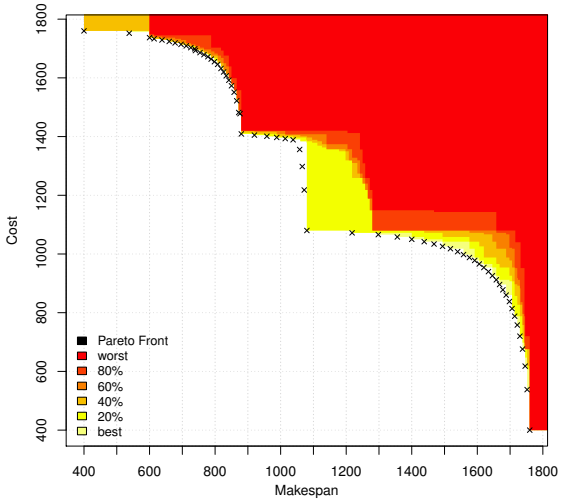


FIGURE 3.19: Surface d'atteinte après 900s sur une version réduite de l'instance 4, approche Pareto.

instances avec des fronts dont la « profondeur » des parties concaves varient.

3.4.5 Instances

Nous avons sélectionnée deux instances MULTIZENOTRAVEL avec la même taille de l'espace de recherche, en utilisant 6 personnes, 3 villes centrales et deux avions.
Les fonctions génératrices du vecteur de coûts et de durées sont les même pour les deux instances. La seule différence provient des coefficients utilisés pour générer ces vecteurs.

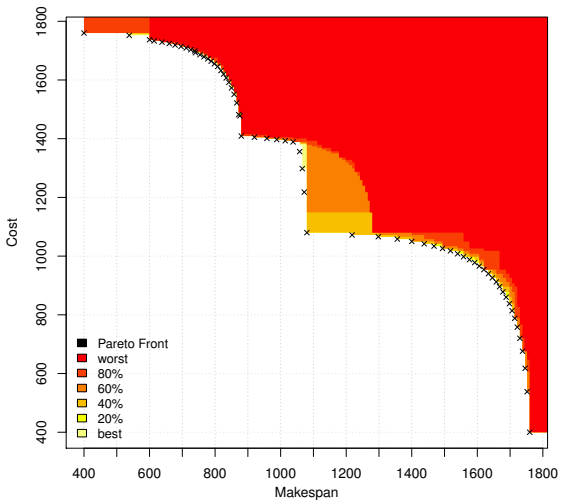


FIGURE 3.20: Surface d'atteinte après 5400s sur une version réduite de l'instance 4, approche Pareto.

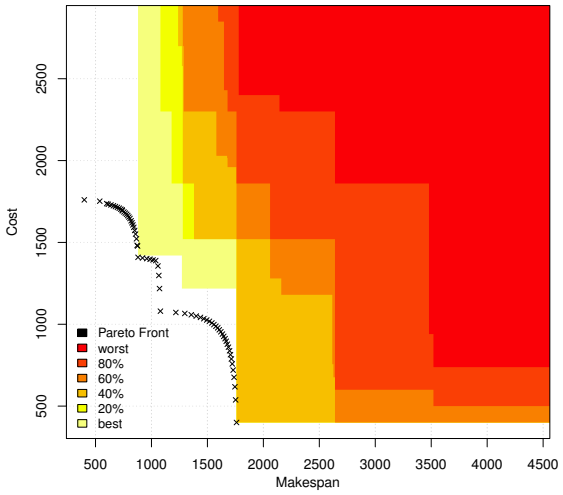


FIGURE 3.21: Surface d'atteinte après 5400s sur une version réduite de l'instance 4, approche Aggrégation

3.4.6 Tests

Pour chaque instance, 30 tests ont été effectués, avec 300 secondes pour chacun. Pour comparer la difficulté à atteindre les parties concaves du front, nous avons divisé l'ensemble des points du front en deux catégories. La première décomposition est montrée par la Figure 3.24. Un premier groupe est uniquement composé de points au premier plan (*forefront*), en dehors des parties concaves, alors que le second est fait des points les plus extérieurs du front, que nous appellerons *backfront*.

Une seconde décomposition à été faite, en considérant cette fois un second groupe composé de tous les points n'appartenant pas au *forefront*.

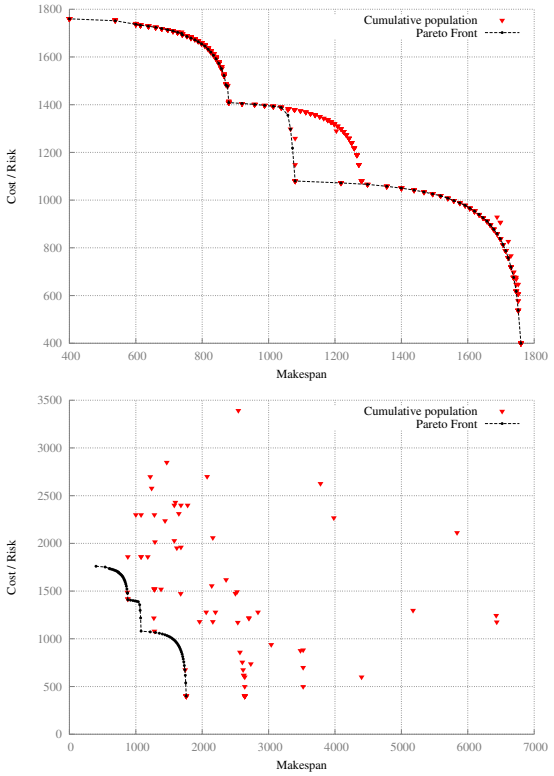


FIGURE 3.22: Fronts cumulés pour l'approche Pareto puis l'approche Aggrégation.

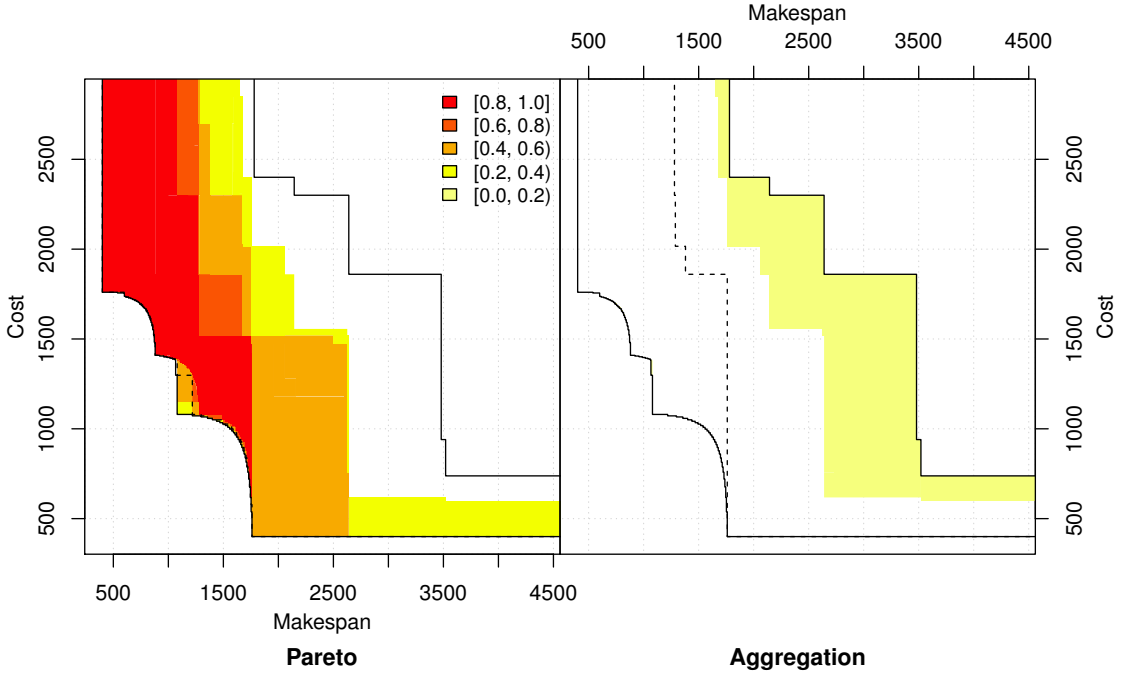


FIGURE 3.23: Différence de surface d'atteinte entre Pareto et l'Aggrégation.

La Figure 3.26 montre le ratio moyen d'atteinte des points du premier groupe, du second groupe et la moyenne. À gauche il s'agit de la première décomposition et à gauche de la seconde.

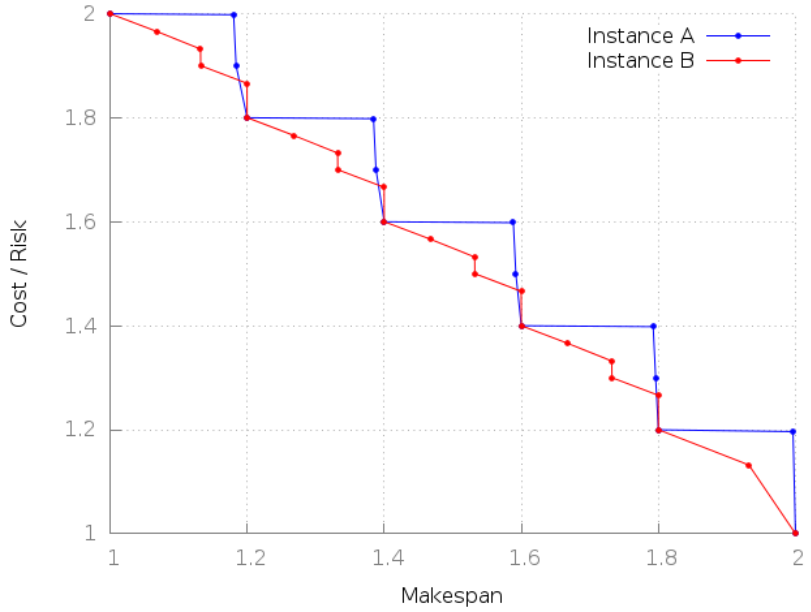


FIGURE 3.24: Front de Pareto normalisés pour les instances A et B.

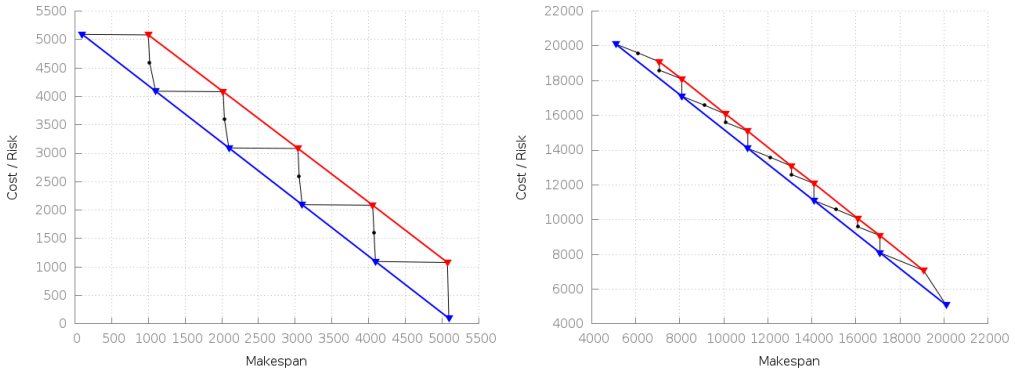


FIGURE 3.25: "Forefront" et "backfront" pour les instances A et B.

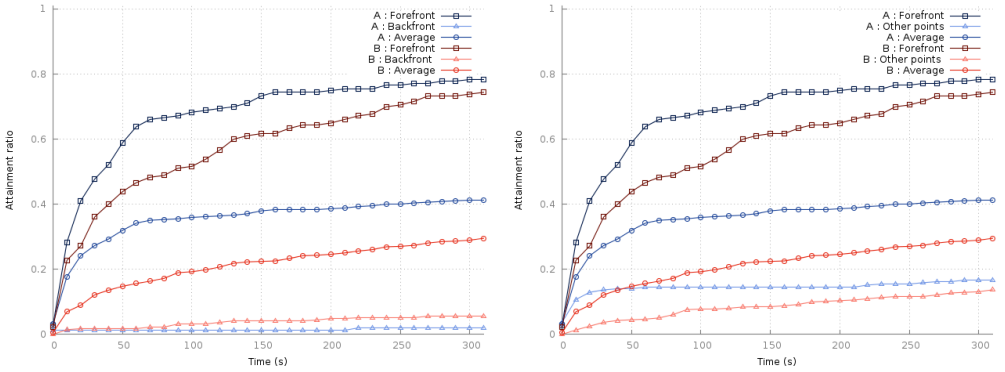


FIGURE 3.26: Ratio d'atteinte groupé.

Il est intéressant de noter qu'une large part du ratio moyen d'atteinte provient du premier groupe, dans les deux décompositions. De plus, ce ratio dans le premier groupe est toujours supérieur au ratio du second groupe, même en considérant que dans la seconde

décomposition, le cardinal du second groupe est bien plus grand que celui du premier groupe. En effet, il y a 6 points dans le forefront pour la première et second instance, contre 9 points pour le second groupe de la seconde décomposition pour la première instance, et 17 pour la seconde instance.

Un test des rangs-signés de Wilcoxon à un niveau de confiance de 95% sur les ratios d'atteinte indique que les points du premier groupe sont bien plus faciles à atteindre que ceux du second groupe, pour les deux décompositions et les deux instances. Pour la seconde décomposition, et une hypothèse nulle que le premier groupe est aussi facile à atteindre que le second (ie. que les ratios d'atteinte sont les même), le rejet de l'hypothèse nulle est très fort, avec, respectivement pour l'Instance A et B une p-value de 1.214e-10 et 1.017e-10. Si l'on compare les groupes de la seconde décomposition, entre les instances cette fois, les premiers groupes ne sont pas équivalents entre l'instance A et B (p-value de 0.0002534). Cela provient certainement de la sensibilité des instances, dont la complexité semble varier très fortement même en ne modifiant qu'un paramètre auxilière (facteur d'échelle par exemple). Il semblerait donc que les instances ne sont pas réellement comparables. Cependant, en comparant le second groupe, la p-value est bien plus basse que pour le premier groupe, indiquant un rejet bien plus fort de l'hypothèse nulle selon laquelle les groupes sont égaux (p = 2.912e-10).

Il ressort deux choses de cette étude. D'une part, malgré la facilité apparante comparée à une approche aggrégation, l'approche Pareto semble tout de même sensible devant la forme du front et atteindre plus difficilement les parties concaves. La seconde chose est qu'il semblerait que la distance entre les points hors des parties concaves et les points des parties concaves influence la facilité de résolution. Ce résultat préliminaire est à relativiser puisque le test statistique a également rejeté l'hypothèse d'égalité que nous aurions aimé avoir entre le premier groupe sur les deux instances.

3.5 Discussion et perspectives

La suite de benchmarks proposée ouvre la porte à des comparaisons plus en profondeur dans le domaine de l'optimisation combinatoire où il n'existait pas d'instance large à notre connaissance. En particulier, derrière la méthode basique d'aggrégation par somme pondérée servant ici de résultat préliminaire, de plus amples expériences doivent être

menées avec l'état de l'art des méthodes de décompositions pour lesquels les composantes coopèrent (par exemple, de la famille MOEA/D [32]). Grâce au benchmark MultiZeno-Travel et à l'algorithme ZenoSolver (disponible via https://descarwin.lri.fr), de telles expériences sont désormais possibles avec la connaissance de la solution exacte au problème. Dans le domaine de la planification cela va conduire à des comparaisons plus justes entre approche Pareto et non-Parreto (voir par exemple, [33]).

Chapitre 4

Optimisation de paramètres

4.1 Introduction

4.1.1 Généralité sur l'optimisation de paramètres

Il est bien connu que l'un des désavantages des métaheuristiques provient du nombre conséquent de paramètres à régler, souvent empiriquement, pour obtenir des résultats satisfaisants. La difficulté provient du fait qu'à priori les paramètres optimaux dépendent du problème et même de l'instance. Il est donc d'usage d'appliquer une campagne de tests empiriques afin de décider des paramètres optimaux à utiliser lors d'une phase d'exploitation.

L'optimisation de paramètres consiste donc à concevoir des méthodes permettant d'optimiser les paramètres d'un algorithme de manière automatique. On distingue deux grandes catégories parmi ces méthodes, à savoir l'optimisation *off-line* et *online*. La première catégorie consiste à relancer de multiples fois l'algorithme, avec divers paramètres, diverses instances ou problèmes et prendre une décision quand aux meilleurs paramètres. La seconde catégorie optimise les paramètres au cours de l'algorithme principal, ne nécessitant ainsi pas de relancer l'algorithme.

À priori, la première catégorie ne permet donc d'obtenir que des paramètres statiques. Une seule valeur, considérée comme optimale, est retournée à la fin de l'optimisation d'un paramètre. Cependant, plusieurs arguments peuvent être avancés pour favoriser

des paramètres non-statiques. Les algorithmes évolutionnaires sont des procesus dynamiques. L'utilisation de paramètres statiques durant toute la recherche va à l'encontre de cette dynamique. Il n'y a aucune raison pour qu'un paramètre soit optimal durant toute la durée de la recherche. Au contraire, la valeur optimale de certains paramètres dépend du temps et l'optimisation *online* consiste donc à trouver et traquer ces valeurs optimales des paramètres qui influencent la qualité des solutions obtenues.

Au sein des stratégies d'optimisation de paramètres *online*, il existe trois grandes catégories : les stratégies statiques, les stratégies adaptatives et les stratégies auto-adaptatives. Les stratégies statiques consistent à modifier périodiquement un paramètre de manière aveugle, sans tenir compte de l'environnement et du processus de recherche. Citons par exemple le paramètre de *température* dans un algorithme du recuit simulé, qui dans la version originale, est modifié sans aucun apport d'information extérieure (par exemple $T_{n+1} = 0.9T_n$). A contrario, les stratégies adaptatives vont se baser sur une connaissance acquise durant le processus de recherche, appelée *feedback*, pour décider la manière dont il faut modifier les paramètres. Un exemple classique et basique serait la « règle du $\frac{1}{5}$ » proposée par Rechenberg qui consiste à augmenter la fréquence des mutations lorsque plus d'un cinquième des mutations produisent des individus de meilleur qualité et de la diminuer dans le cas contraire. Cette stratégie peut cependant être mise à défaut assez facilement. Enfin, les stratégies auto-adaptatives consistent à intégrer le paramètre à optimiser directement dans le génotype des individus. S'il s'agit de la fréquence de mutation par exemple, chaque individu portera sa propre fréquence de mutation, qui sera elle même sujette à mutation. Ces stratégies sont dites *for free* car le paramètre est optimisé grâce au processus d'optimisation principal et n'induisent aucun coût de calcul supplémentaire. L'idée derrière ces stratégies réside dans le fait que si la valeur d'un paramètre est mauvais, comme ce paramètre à une influence sur la qualité d'un individu, cet individu sera naturellement éliminé par les processus de sélection et de remplacement et il ne subistera ainsi que les individus ayant des paramètres optimaux.

En pratique, les stratégies auto-adaptatives sont en général plus performantes. Cependant, des stratégies adaptatives bien conçues et utilisant à bon escient l'information du contexte arrivent à proposer de meilleurs résultats. Citons par exemple la métaheuristique CMA-ES, proposée par Hansen en 1996 [34], qui consiste à faire évoluer la matrice de variance-covariance des évaluations des individus de la population et de se servir des trajectoires des individus dans l'espace objectif pour projecter les directions

optimales. CMA-ES peut être vu, globalement, comme un gradient stochastique, d'un point de vue probabilité comme une maximisation du maximum de vraisemblance, ou d'un point de vue analyse de données comme une Analyse en Composante Principale (ACP) sur laquelle on va garder l'ensemble des axes d'information (l'ACP consistant à résumer les données dans un espace de dimension n en les exprimant dans la base de dimension $m < n$ qui résume le mieux les données).

Une autre notion est également importante lorsque l'on cherche à optimiser un paramètre : à quel niveau de granularité doit s'appliquer ce paramètre ? Les principaux niveaux de granularité sont : au niveau de la population, au niveau de l'individu et au niveau du gène. Si l'on reprend l'exemple du taux de mutation, la question est de savoir si le paramètre est commun à l'ensemble de la population, unique pour chaque individu ou appliqué à chaque gène. Empiriquement, les meilleurs résultats ont été obtenus au niveau de granularité le plus fin, c'est à dire au niveau du gène.

Évidemment, ces classifications historiques tendent à devenir plus floues puisqu'on peut trouver des algorithmes mélangeant des *feedbacks* basés sur de l'information disponible au niveau de la population et de l'individu, certaines méthodes combinent une stratégie auto-adaptative avec une correction basée sur un *feedback*, etc.

4.1.2 L'existant au sein de DaE

L'ensemble des paramètres de DaE étaient statiques. Parmi ces paramètres, le choix de l'objectif à optimiser lors d'un appel à YAHSP présente un intérêt tout particulier. La publication [28] a montré expérimentalement que le choix de cet objectif est particulièrement important sur la qualité des solutions, ainsi que sur la répartition uniforme sur le front. De plus, l'optimisation *offline* des poids grâce à l'outil ParamILS retourne des poids de 0.5 pour chacun des deux objectifs (durée et coût), c'est à dire un choix purement uniforme. À priori, les arguments avancés dans l'introduction sont toujours valables et il est donc intéressant d'essayer de trouver une stratégie d'optimisation *online* pour espérer augmenter significativement la qualité des résultats produits par DaE.

Le solveur YAHSP, comme tout solveur de planification mono-objectif n'est capable que de prendre en compte un unique objectif. $\text{DaE}_{\text{YAHSP}}$ se charge donc de donner à YAHSP la stratégie à utiliser au moment de la résolution d'un sous-problème.

Le critère à optimiser est originellement porté par la population : l'objectif à utiliser lors d'un appel à YAHSP est sélectionné selon une distribution de probabilité définit par l'utilisateur et immuable au cours du temps d'exécution.

Dans ce chapitre nous présentons l'élaboration, la conception et la validation de stratégie d'optimisation pour l'objectif à optimiser par YAHSP. Il sera découpé en plusieurs parties. Une première partie essayera de modéliser le problème mathématiquement, en définissant le problème d'optimisation *online* d'un point de vue contrôle de processus stochastiques ainsi que la notion de stratégie.

Dans une section 4.2 nous présentons les choix de modélisation et les différents éléments communs aux stratégies. Nous présenterons ensuite dans la partie 4.3 le protocole général pour les différentes expériences qui ont été faite, ainsi que les outils et la méthodologie pour l'analyse des résultats

Les parties suivantes, 4.4, 4.5, 4.6 et 4.7, présenterons tour à tour les différentes approches développées, respectivement les stratégies statique, adaptative, gloutonne et auto-adaptative, ainsi que les résultats empiriques obtenus pour chacune. Chacune de ses parties essayera de tirer des conclusions des observations ou des hypothèses sur les raisons des résultats obtenus.

Fort des quelques idées qui se sont dégagées de ces premières parties expérimentales, nous proposerons une version différente du processus classique d'évaluation, qui cherche à mieux qualifier la facilité « structurelle » d'un individu de l'algorithme génétique à être optimisé là où la version classique utilise la qualité du plan trouvé. Pour cela nous utiliserons l'échantillonnage de la zone de l'espace objectif accessible par un individu. Nous verrons de quelle manière la stochasticité et sous-optimalité de YAHSP conduit à biaiser l'algorithme génétique dans le cas d'une évaluation classique. Nous introduirons pour cela une méthode pour comparer l'effet respectif de YAHSP et de l'algorithme génétique sur le résultat final.

Enfin, dans une dernière partie, nous conclurons en résumant l'ensemble des travaux et grandes idées qui se dégagent de ce chapitre, tout en proposant des axes d'améliorations.

4.2 Modélisation

4.2.1 Problème d'optimisation de paramètres en ligne

On peut voir un algorithme génétique à base de population utilisant une archive comme suit :

$$\begin{aligned} X : (\Omega, F, \mathbb{P}) \times [0,T] \times \mathcal{P} &\to E \times A \\ (\omega, t, p) &\mapsto X_{U(\omega)}(p, \omega) \end{aligned}$$

X est un processus stochastique, c'est à dire une suite de variables aléatoires indexées dans le temps (ici par la génération), définis sur un espace probabilisé, à valeur dans un espace d'états (par exemple $(\{0,1\}^m)^k$ dans le cas d'une population de k individus codés comme une chaine de bits de longueur m) et l'espace des archives (un élément appartient à cet espace s'il est constitué d'éléments non-dominés pour la relation de dominance, on a donc $A \subset E$), contrôlé par un ensemble de paramètres et conditionné par un temps d'arrêt U. On filtre l'espace probabilisé par la filtration canonique du processus X : $F_n^X = \sigma\{X_0, X_1, \ldots, X_n\}$, c'est à dire la plus petite tribu rendant mesurables les variables aléatoire $X_0, X_1, \ldots, X_n$.

Le temps d'arrêt U est donc un temps d'arrêt *pour* la filtration F_n^X. La génération à laquelle le processus va s'arrêter n'est en général pas déterministe puisqu'il peut s'agir d'un temps CPU, d'un nombre de générations avec une population stationnaire, ou de l'attente d'une certaine qualité par exemple.

Un élément $p \in \mathcal{P}$ est appelé une configuration. L'espace des configurations P est le produit cartésien d'un certain nombre d'ensembles, par exemple $P = \{0,1\}^{k_1} \times \mathbb{R}^{k_2} \times \mathbb{N}^{k_1}$. Une composante de p est appelé paramètre. Par abus de langage on parle souvent de p comme d'un vecteur de paramètres.

On définit également l'espace S des stratégies, c'est à dire l'espace des fonctions $s : F_n^X \to \mathcal{P}$. Ce sont les fonctions qui, connaissant une certaine quantité d'information à un instant t, vont retourner une configuration.

Le problème (Π_o) d'optimisation de paramètres en ligne peut se formuler de la sorte : « Trouver $s^* \in S$, tel que $\forall s \in S, X_U(s^*, \omega) \succeq X_U(s, \omega)$ ».

La définition de la relation $\succeq$ dépend du problème étudié et à priori de l'utilisateur. Dans notre cas, c'est un ensemble de métriques qui conditionnent le succès d'une stratégie, notamment la métrique d'hypervolume H^- sur les archives retournée par X_t muni de la stratégie s^* et d'une autre stratégie s, mais aussi les différences significatives de surfaces d'atteinte et un test de Kolmogorov. Nous détaillerons le protocole expérimental et les métriques utilisés dans la partie 4.3.

Dans notre cas, il n'existe pas à priori de modèle analytique décrivant le processus X (même si des travaux fournissent un cadre analytique solide pour des algorithmes à base de populations simples, comme [35]) et il semble donc compliqué de travailler analytiquement sur le problème (Π_o).

Notons également que la définition proposée pour (Π_o) ne tient pas compte du niveau de granularité du paramètre. Certains paramètres, comme la taille de la population, ne sont typiquement applicables qu'à l'ensemble de la population, alors que des paramètres comme le taux de mutation peuvent être appliqué au niveau de l'individu ou directement du gène.

Cela nous amène à la réflexion que l'on peut voir le problème d'optimisation de paramètres comme un problème de contrôle stochastique. Cela justifie notamment une approche par *feedback* où l'on cherche à déterminer la commande optimale à appliquer au système pour minimiser un certain coût, comme c'est le cas par exemple pour le problème « linéaire quadratique » proposé par Kalman en 1960. La principale différence dans notre cas avec ce dernier exemple est que l'absence de modèle analytique empêche l'étude théorique *à priori* du problème et nous condamne à une étude plus empirique. On ne peut pas obtenir des conditions d'existence d'une commande optimale [1] si l'on ne possède pas de modèle analytique.

Les difficultés de modélisation sont nombreuses. Tout d'abord la multitude des méthodes et des approches tend à rendre difficile un modèle unique. La complexité des processus sous-jacents rend l'étude globale également très difficile, encore plus dans le cas du multi-objectif. Ainsi, pour la thèse de Cerf [35], un critère de convergence par rapport à la taille de la population est donné pour un algorithme génétique à base de population le plus simple qui soit, sans croisement par exemple. Si les techniques employées sont applicables à des processus plus compliqués ou pour des opérateurs différents, l'effort

1. Conditions de Jacobi dans le cas de la commande linéaire quadratique.

est considérable et est plus lent que le rythme de publication de nouvelles méthodes d'optimisation.

Au vu de ce constat, l'approche abordée par la suite est plutôt classique : conception d'une méthode heuristique, campagne de tests, validation statistique et comparaison avec d'autres méthodes.

4.2.2 Modélisation d'une stratégie

Au vu de ces éléments, il apparait que la modélisation la plus naturelle pour déterminer l'objectif à utiliser est une distribution de probabilité et non simplement un objectif porté par l'individu ou le gène.

On place à partir de maintenant dans le cadre de l'optimisation de l'objectif à passer à YAHSP.

Soit x un individu de Ω où Ω change selon le niveau de granularité (soit la population, l'ensemble des individus ou l'ensemble des sous-problèmes composant les individus). Notons $P^j(x)$ la loi de probabilité associée au choix de l'objectif à optimiser à la génération i pour l'individu x que l'on peut expliciter comme suit :

$$P^j(x) = \begin{pmatrix} p^j(o_1|x) \\ \vdots \\ p^j(o_i|x) \\ \vdots \\ p^j(o_l|x) \end{pmatrix}$$

Avec $O = (o_i)_{1 \leq i \leq l}$ l'ensemble des évènements « choisir l'objectif i ».

Une stratégie est donc une suite itérative modifiant la distribution de probabilité P. L'objectif est de trouver la distribution de probabilité optimale au sens où à chaque génération, le choix de l'objectif à optimiser est le *meilleur* en un sens à définir. On ne s'occupera par la suite que de P.

À priori, à l'instant initial, la connaissance de l'objectif *optimal* à utiliser n'est pas connu. Un tirage aléatoire et uniforme de l'objectif semble donc une première solution, sans connaissance à priori du problème. Quand bien même l'objectif sélectionné apporte

une amélioration à l'individu, même s'il semble légitime de vouloir continuer à l'utiliser, un autre objectif peut éventuellement apporter un meilleur résultat. Ainsi, cette idée justifie le fait d'utiliser une distribution de probabilité sur l'ensemble des objectifs, et effectuer un tirage selon cette distribution chaque fois que nécessaire, plutôt que porter simplement la valeur de l'objectif courant. De manière générale, il est possible de simuler le fait de ne porter qu'un seul objectif en mettant la probabilité de celui-ci à 1.

4.2.2.1 Indicateurs

Les indicateurs permettent d'établir une relation d'ordre entre deux vecteurs objectifs. Ils seront utiles à la fois pour établir le *feedback* des stratégies adaptatives mais également lorsque l'on parlera d'échantillonnage à la section 4.8.2. C'est pourquoi on les présente dès à présent. Notons que selon l'indicateur, un vecteur objectif sera meilleur s'il est affecté d'une valeur faible tandis que pour certain c'est le contraire. On utilisera le signe $\underset{\lambda}{\succ}$ pour parler du meilleur au sens de l'indicateur λ.

On peut classer les indicateurs en deux catégories. Une première catégorie contient des indicateurs ternaires utilisant comme troisième opérande le vecteur objectif de la génération précédente. Les indicateurs de la seconde catégorie peuvent être qualifiés d'absolus dans le sens où ils ne dépendent pas de la génération précendente. Cependant, cette classification n'est pas fixée puisqu'on peut imaginer des indicateurs mixtes ou des indicateurs totalement différents, utilisant l'archive courante et l'archive locale (archive de l'échantillon).

Amélioration relative des individus entre la génération $j-1$ et j. On définit pour cela un vecteur $\Delta f^j(x) = \begin{pmatrix} \Delta f_1^j(x) \\ \vdots \\ \Delta f_m^j(x) \end{pmatrix}$ où l'on calcule $\Delta f_i^j(x)$, l'amélioration relative de x par rapport à l'objectif i entre la génération $j-1$ et j comme suit :

$$\Delta f_i^j(x) = \frac{f_i^{j-1}(x) - f_i^j(x)}{f_i^{j-1}(x)}$$

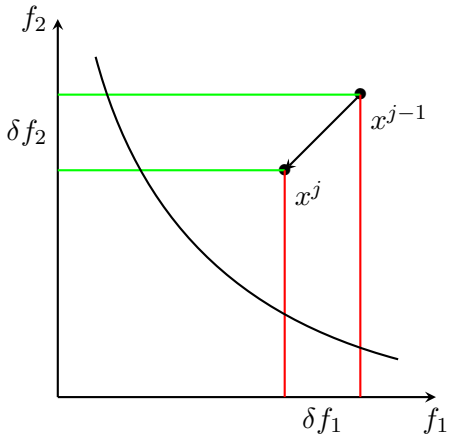


FIGURE 4.1: Évolution d'un individu dans l'espace des objectifs entre la génération $j-1$ et la génération j. Les quantités δf_1 et δf_2 sont les améliorations absolues de l'individu apportées par les opérateurs de variation.

Ainsi, on définit l'indicateur $\lambda^j_{\Delta^+}(x) = \sum_i \Delta f_i^j(x)$ qui valorise l'amélioration peu importe l'axe : une amélioration sur un axe peut être suffisamment importante pour compenser une perte sur un autre axe.

On définit également l'indicateur $\lambda^j_{\Delta^\times}(x) = \prod_i (\Delta f_i^j(x) 1_{f_i^j(x)>0} + 1_{f_i^j(x)\leq 0})$ qui mesure le volume du parallélotope rectangle [2] dont la dimension est égale au nombre d'axes améliorés (et suivant ces axes évidemment). On favorise ainsi largement les améliorations sur le plus d'axes possibles.

L'avantage de ces indicateurs est qu'ils ne sont pas, à priori, sensibles à la différence d'échelle entre les objectifs, mais plutôt à la facilité ou difficulté d'optimisation selon un axe. Cependant, lorsqu'un individu est évalué cela signifie qu'il a été modifié auparavant, et donc a subit une mutation ou un croisement. Ce changement de structure peut impliquer un large biais sur le point de référence utilisé. En effet, si la moitié des gènes ont changés, est-ce qu'il est encore raisonnable d'utiliser le vecteur objectif de l'évaluation précédente comme point de référenc pour mesure l'amélioration ?

Norme On peut définir un indicateur absolu $\lambda_{||.||_p}(x) = ||f(x)||_p$. Comme généralement, la norme 1,2 et ∞ semblent les plus naturelles. On notera cependant que ces indicateurs sont fortement biaisés lorsque les objectifs ne sont pas normalisés, la normalisation impliquant une connaissance à priori sur les bornes des objectifs.

2. c'est à dire un parallélotope à hyperfaces contiguës orthogonales.

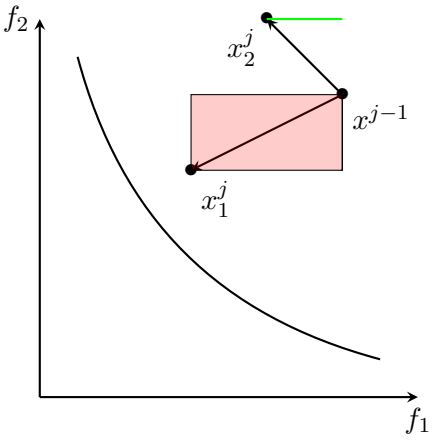


FIGURE 4.2: Illustration de l'indicateur $\lambda^j_{\Delta^\times}$. Si l'individu x à la génération $j-1$ mène à l'individu x_1 à la génération suivante, alors l'indicateur retournera l'aire rouge. S'il mène à x_2, seul l'axe d'amélioration sera prit en compte et l'indicateur retournera la longueur du segment vert.

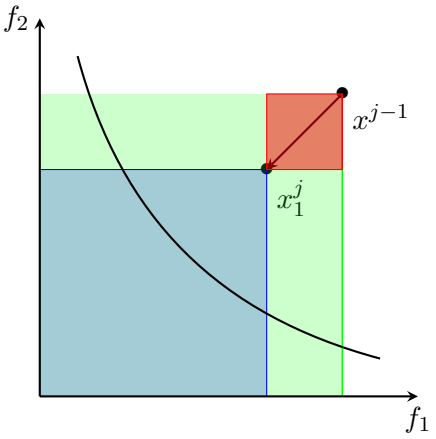


FIGURE 4.3: Illustration de l'indicateur λ_M. La valeur de l'indicateur pour l'individu x à la génération j sera la surface délimitée par les axes du repère et les axes vert, moins la surface délimitée par les axes du repères et les axes bleus, à laquelle on ajoute la surface du carré rouge.

De manière analogue à précédemment, on peut utiliser l'hypersurface, mais cette fois formée par le vecteur considéré et l'origine : $\lambda_{H_O}(x) = \prod_i f_i(x)$.

Un compromis On se propose de définir un nouvel indicateur qui mixe les indicateurs précédants :

$$\lambda_M(x) = \lambda_{H_O}(r) - \lambda_{H_O}(x) + \lambda^j_{\Delta^\times}(x)$$

Algorithm 5 Séquence d'évaluation initiale d'un individu x.

```
YAHSP_Initialization(rand())          {Initialize YAHSP with a random seed. }
o_i ← rand(p)        {Select an objective according to the probability distribution p.}
YAHSP_Optimize(o_i)
x.state ← feasible
for all x_i in x do
  x.subplan[i] ← YAHSP_Solve(x_i)
  if x_i not solved in b_max then
    x.state ← unfeasible
    break
  end if
end for
if x.state = feasible then
  x.plan ← YAHSP_Compress(x.subplan)
  x.objVector ← setObjectiveVector(x)
end if
```

4.2.3 Stratégies : concepts généraux

Une stratégie est caractérisée par 4 éléments :

- C'est un foncteur qui renvoie un objectif (au sens de YAHSP).
- Elle possède un niveau de granularité parmi : population, individu et gène.
- Elle possède une fonction de mise à jour prenant en paramètre un indicateur de qualité relatif au dernier objectif retourné.
- Elle possède une fonction de mutation.

Evidemment, une stratégie statique qui consiste à avoir une distribution de probabilité qui n'évolue pas en fonction du temps aura une fonction de mutation qui n'aura aucun effet, tout comme sa fonction de mise à jour. Cela permet cependant d'écrire une version d'évaluation tenant compte de la stratégie, quelque soit cette stratégie.

L'algorithme 5 indique la séquence d'origine pour l'évaluation d'un individu. Notons que la fonction `SetObjectiveVector` cache diverses étapes. En effet, si YAHSP n'a pas réussit à trouver un plan valide, il peut renvoyer divers codes d'erreur qui servent à établir un vecteur objectif fortemment pénalisé pour s'assurer que l'individu disparaisse rapidemment par le processus de sélection et remplacement.

L'appel à `updateStrategy` et `mutateStrategy` cache évidemment un branchement pour déterminer à quel niveau doit s'appliquer la mise à jour et la mutation éventuelle de la stratégie. λ représente un des indicateurs présenté plus haut.

Algorithm 6 Séquence d'évaluation d'un individu x avec strategie.

```
x.objVector ← unfeasible
x.objVector ← setObjectiveVector(x)
if StratLevel = Population then
  o_i ← strategy()
else if StratLevel = Individual then
  o_i ← x.strategy()
end if
YAHSP_Initialization(rand())          {Initialize YAHSP with a random seed. }
YAHSP_Optimize(o_i)
s ← feasible
for all x_i in x do
  if StratLevel = Gene then
    o_i ← x_i.strategy()
    YAHSP_Initialization(rand())      {Initialize YAHSP with a random seed. }
    YAHSP_Optimize(o_i)
  end if
  x.subplan[i] ← YAHSP_Solve(x_i)
  if x_i not solved in b_max then
    s ← unfeasible
    break
  end if
end for
u ← setObjectiveVector(x)
updateStrategy(λ(u))
mutateStrategy()
if x.state = feasible then
  x.plan ← YAHSP_Compress(x.subplan)
  x.objVector ← setObjectiveVector(x)
end if
```

4.3 Protocole expérimental et comparaison

On présente ici brièvement les outils externes utilisés lors des expériences puis l'ensemble du protocole expérimental.

4.3.1 Outils

4.3.1.1 ParamILS

ParamILS est un *framework* dédié aux réglages de paramètres d'algorithme et à leur configuration. Il est possible de l'utiliser sur n'importe quel algorithme dont les paramètres à optimiser peuvent être discrétisées. Il s'agit d'un meta-optimiseur qui va

Algorithm 7 Première étape de l'algorithme ParamILS.

$p \leftarrow$ configuration par défaut.
repeat
  $p' \leftarrow p$
  for all $p'' \in \mathcal{V}(p')$ dans un ordre aléatoire **do**
    if $p'' \succ p$ **then**
      $p \leftarrow p''$
      break
    end if
  end for
until $p' = p$
return p

chercher à optimiser la valeur de paramètres par rapport à un objectif fixé par l'utilisateur (médian de la qualité des solutions, temps moyen d'exécution,...) au regard d'un ensemble d'instances d'un problème donné.

ParamILS a été utilisé pour bon nombre de solveurs, tant académique qu'industriel afin d'augmenter substentiellement la qualité des résultats de ces solveurs, notamment pour des problèmes NP-difficiles tels que le problème de statisfaction de contrainte (SAT), programmation linéaire en nombre entier (PLNE) ou mixte (MIP), la planification ou en général tout problème de décision.

Le principe de ParamILS repose sur un algorithme de recherche locale. L'utilisateur spécifie l'ensemble $\mathcal{P}_d$ des configurations discrétisées, à partir de $\mathcal{P}$ l'ensemble des configurations de l'algorithme A à optimiser. L'utilisateur fourni également une relation de comparaison entre deux éléments de $\mathcal{P}_d$ et éventuellement une relation de voisinage $\mathcal{V}$ entre les éléments de $\mathcal{P}_d$. Une configuration initiale p_0 est donnée. Une première étape, nommée *IterativeFirstImprovement* d'après les auteurs, est décrite par l'algorithme 7.

La deuxième phase consiste à effectuer les étapes suivantes tant qu'aucun critère d'arrêt n'est atteint (typiquement un nombre de *runs*, un temps CPU ou une qualité) :

1. Perturbation de la configuration courante.
2. Appel à *IterativeFirstImprovement* avec la configuration courante.
3. Si la configuration perturbée est meilleure elle devient la configuration courante.
4. Avec une faible probabilité[3], la configuration courante est remplacée par une configuration aléatoire.

3. Les auteurs conseillent une probabilité de 0.01. Voir [36] pour plus de détails.

Il existe plusieurs variantes dans l'algorithme (BasicILS, FocusILS,...) mais l'idée reste sensiblement la même. Pour davantage de détails sur ces algorithmes nous renvoyons à [36] et [37]. Les auteurs fournissent également des preuves de convergences de ParamILS vers la configuration optimale et des intervalles de confiance sur les résultats obtenus. Enfin, l'utilisation de ParamILS avec DAE$_{\text{X}}$ a montré son efficacité notamment au travers de l'article [38].

ParamILS fourni également des variantes pour l'optimisation d'algorithme parallèle, et de nombreux mécanismes utiles pour la reproductibilité des expériences, notamment un système permettant de fournir les graines utilisées pour chacune des itérations et des appels à l'algorithme à optimiser.

4.3.1.2 PISA

PISA Performance Assessment Suit est un ensemble d'outils statistiques pour l'étude des performances d'algorithmes d'optimisation approchée, et plus particulièrement dédié à l'optimisation multi-objectif. L'objectif de PISA est de fournir un cadre de travail pour la comparaison d'algorithmes multi-objectifs sur différents problèmes.

Le vocabulaire de PISA parle de *Selector* pour les algorithmes et de *Variator* pour les problèmes ou instances de problèmes. Les deux sont indépendants et reliés par un *Monitor* qui s'occupe d'agréger et de transmettre les résultats des *runs* aux différents outils statistiques. Un *Monitor* décrit donc la procédure de comparaison, et permet également d'intégrer des outils de visualisation et de contrôle des résultats.

Parmi les différents outils qui peuvent composer un *Monitor*, on retrouve des outils de normalisation et de filtre (extraction du Front de Pareto d'un ensemble par exemple), des outils dédiés au calcul d'indicateurs (on retrouve l'ϵ-indicateur, l'hypervolume H^+ ou H^- notamment), des outils statistiques (test de Wilcoxon, Student, etc.) et des outils plus spécialisés ou dédiés à la visualisation (surface d'atteinte, fonction d'atteinte,...).

L'intérêt de PISA est de fournir des outils *clefs en main*, faciles d'utilisation, éprouvés et unifiés. De plus amples détails peuvent être trouvé dans le rapport technique [39] ou directement sur le site internet de PISA [4].

4. http ://www.tik.ee.ethz.ch/pisa/

4.3.2 Protocole et méthodologie

Le protocole initial consiste en une première phase d'optimisation utilisant ParamILS couplé à PISA. En effet, ParamILS a besoin d'une mesure de performance pour déterminer une relation d'ordre entre les configurations qu'il explore. Pour cela un script `bash` est utilisé pour faire la correspondance entre le formalisme de ParamILS et le formalisme de DAE au niveau des paramètres [5], pour appeler DAE, appeler PISA pour calculer l'hypervolume du résultat retourné par DAE et enfin, fournir le résultat à ParamILS.

Pour les instances larges avec la stratégie statique, la longueur des *runs* est de 5400 secondes, ce qui correspond à une heure et demie. Le choix est relativement arbitraire et se base sur les précédentes études [28][38][31] qui utilisait des *runs* de 900 secondes pour l'instance MULTIZENOTRAVEL9, qui correspond aux paramètres $n = 3$, $t = 9$, $p = 2$. Après analyse des résultats, nous verrons que l'évolution de l'hypervolume devient relativement stationnaire bien avant cette limite de 5400 secondes ce qui indiquerait une convergence de l'algorithme. Ce n'est donc, à priori, pas tellement le résultat à la fin des 5400 secondes qui importe, mais plutôt l'ensemble de la dynamique obtenu sur l'ensemble des *runs* et à priori, un temps supérieur à la convergence de l'algorithme semble une meilleure chose pour observer cette dynamique.

De même, nous avons étendu le temps total d'utilisation de ParamILS à quatre jours contre deux dans les études précédentes. Cela ne permet certainement pas une exploration exhaustive de l'espace des configurations mais nous observons tout de même une réduction de la variance entre le meilleur et le moins bon des runs entre le passage de ParamILS et les *runs* utilisant la meilleure configuration retournée par ParamILS.

Une fois le processus d'optimisation de ParamILS, 20 *runs* de 5400 secondes sont effectués avec la meilleure configuration. C'est sur cette série que l'ensemble des résultats sera basé. Toutes les métriques seront acquises grâce à PISA, ou éventuellement, pour celles ne requérant aucun calcul, par de simples scripts `bash`.

On calcule les métriques suivantes :

5. On court-cirtuite les critères d'arrêt par *run* de ParamILS (qui servent à pénaliser les *runs* ne retournant pas de résultat ou de résultat dans un temps donné, en utilisant les critères d'arrêt de DAE. Il y a plusieurs raisons à cela. Premièrement DAE revoie toujours un résultat, et secondement, ParamILS terminerait brusquement DAE s'il ne s'arrêtait pas exactement après le temps demandé. Or, selon les paramètres de DAE (notamment b_{max} et la taille de population), le temps entre deux générations et donc l'écriture sur disque des résultats courant peut largement dépasser des centaines de secondes et amener un bon *run* a être pénalisé.

— L'évolution de la différence d'hypervolume entre le front courant et le front exact au cours du temps. Cette métrique permet de se rendre compte à quel point on est proche du front et également, dans une certaine mesure, si l'on est uniformément dispersé sur le front. On ne donne pas de courbe moyenne car selon les *runs*, qui ne varient que par la graine du générateur aléatoire, la durée d'une génération n'est pas la même et la sauvegarde du front courant peut ne pas avoir lieu au même moment. Il est donc difficile d'avoir une courbe moyenne pertinente. De plus, la courbe moyenne peut, comme nous le verrons dans l'analyse des résultats, cacher des disparités au sein des runs ou certains comportements.

— Les vecteurs objectifs cumulés, à tous les runs et à tout temps. La visualisation de tous les vecteurs objectifs obtenus permet de visualisation, pour une instance la structure de l'espace objectif, si celui-ci fait apparaître des motifs particuliers, ainsi que la manière dont l'algorithme va échantillonner cet espace : différente densité selon la région, disparité selon les *runs* (on dispose d'une version colorée où chaque *run* possède une couleur différente), mise en exerbe de certains zones difficiles d'atteinte.

— Les fronts cumulés de Pareto. Il s'agit de visualiser l'ensemble des vecteurs objectifs constituant les fronts de Pareto obtenu pour chacun des *runs*. Cela permet d'avoir une intuition sur la qualité des fronts mais cache la dynamique du processus : il est possible d'un algorithme converge bien avant les 5400 secondes et qu'au final, une large part du temps ne servent pas beaucoup dans l'évolution du front.

— Les surfaces d'atteinte. Les surfaces d'atteinte permettent de quantifier la probabilité qu'un algorithme atteigne une certaine région de l'espace objectif. Nous choissons de tracer les quantiles suivants : 0.20, 0.40, 0.60 et 0.80, en plus des deux *runs* extrêmes. La encore, on masque la dynamique du procédé mais l'on a accès un moyen très visuel d'observer la qualité d'un algorithme.

— La diversité des vecteurs objectifs. Ce n'est pas, à ma connaissance, une métrique qui a déjà été utilisé pour estimer la qualité d'un algorithme multi-objectif à base de population, mais elle permet cependant d'observer certains aspects de la dynamique de l'algorithme. Il s'agit du ratio entre les vecteurs objectifs différents et le nombre d'individus dans la population. Cela cache évidemment la surjectivité de la fonction d'évaluation f qui peut, pour différent individus de génotype, associer un même vecteur objectif (en parculier, dans le problème MultiZenoTravel,

il existe des plans totalement différents qui mènent au même vecteur objectif). Cependant, cela permet, en collaboration avec les autres métriques, de mettre en évidence une convergence de l'algorithme ou tout du moins de qualifier la diversité de sa population. Nous verrons que les résultats dépendent non seulement de l'instance, comme c'est le cas pour l'hypervolume, mais également de la stratégie d'optimisation des paramètres que l'on met en place, et qu'une diversité faible rapidemment n'implique pas de mauvais résultats.

On n'utilisera pas le ratio d'atteinte d'un point du front ou de tous les points car d'une part ils sont très nombreux ce qui ne permettrait pas d'avoir des graphiques lisibles, mais également car ils sont beaucoup trop restrictifs : on peut être très proche d'un point du front sans l'atteindre, mais cette proximité n'est pas capturée par cet indicateur, contrairement aux surfaces d'atteinte.

Notons que nous n'avons pas relancé ParamILS pour la stratégie auto-adaptative, principalement par des raisons de temps. Comme l'ensemble des paramètres « travaillent » ensemble lors d'un *run*, il est possible que pour obtenir le meilleur des autres stratégies testées, il aurait fallu repasser par cette étape de meta-optimisation. Cependant, après coup, il s'avère que les résultats sont moins bons que la stratégie statique et que ces mauvais résultats proviennent certainement de la manière dont on évalue les individus (voir la section 4.8) et impactent certainement les résultats même sur la stratégie statique.

Certaines instances ou séries ont bénéficié d'un protocole différent, auquel cas ces différences sont expliquées avant l'analyse des résultats.

4.4 Stratégie statique

4.4.1 Présentation

Une stratégie statique consiste à simplement tirer au hasard un objectif selon une distribution de probabilité définit par l'utilisateur. Une telle stratégie part du principe qu'à tout moment et pour tout individu des poids optimaux peuvent être trouvés.

Comme il s'agit de la stratégie la plus simple et déjà implémentée par YAHSP, nous l'utiliserons comme point de référence pour comparer avec d'autres stratégies. Pour donner

un maximum de chances à ces stratégies, nous optimiserons l'ensemble des paramètres (incluant la distribution de probabilité) en utilisant ParamILS.

Un des désavantages de cette stratégie est de demander explicitement à l'utilisateur la distribution de probabilité. Comme il a été montré que la stratégie a un impact important sur les résultats obtenus, le choix est d'autant plus compliqué pour l'utilisateur. En passant par processus d'optimisation *offline* comme ParamILS, le temps total requit pour obtenir les résultats s'accroit considérablement. Même avec une stratégie supprimant l'intervention de l'utilisateur il reste d'autres paramètres statiques. Cependant, cela permet de faire diminuer considérablement le cardinal de l'espace des configurations, et à temps égal, on peut espérer que ParamILS trouve de meilleurs paramètres statiques menant à des résultats finaux meilleurs.

4.4.2 Les paramètres

TABLE 4.1: Paramètres trouvés par ParamILS pour la stratégie statique.

Paramètres	Instance 1	Instance 4	Instance 7	Instance 9
lenght_weigth	1	2	4	5
cost_weigth	1	1	1	4
makespan_max_weigth	1	1	5	1
makespan_add_weigth	1	1	1	0
bmax-fixed	1000000	100000	1000000	10000
popSize	100	200	200	300
proba-change	0.8	0.8	0.0	0.0
proba-cross	0.2	0.1	0.8	0.5
proba-del-atom	0.5	0.2	0.1	0.8
proba-mut	0.8	0.8	1	1
radius	3	1	3	5
w-addatom	1	7	10	3
w-addgoal	1	7	1	5
w-delatom	3	3	7	5
w-delgoal	1	3	1	0

Il est intéressant de noter que pour les instances les plus compliquées en théorie (7 et 9), le poids de la longueur du plan est important voire le plus important comme pour l'instance 9. La probabilité de modifier un atome lors d'une mutation est également nulle alors que la probabilité de croisement est relativement haute. En comparaison, pour les instances 1 et 4, la probabilité de changer un atome est haute par rapport à la probabilité de croisement et l'influence de la longueur du plan semble moins importante.

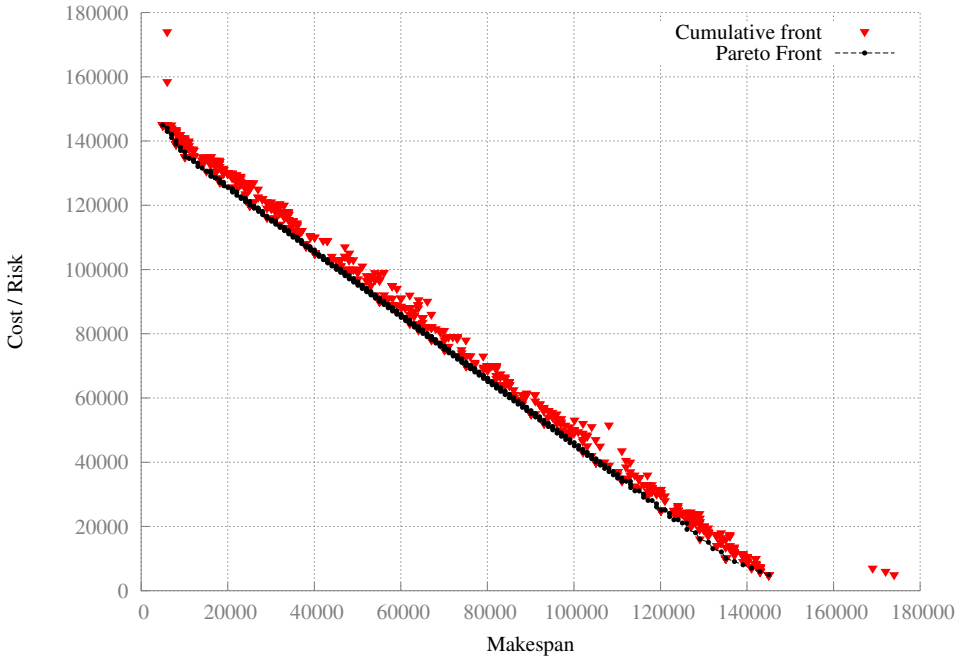
On note également une distribution parfaitement uniforme pour le choix de l'objectif avec l'instance 1 et relativement uniforme pour l'instance 4.

De même, la probabilité de mutation est très élevée pour l'ensemble des instances.

4.4.3 Résultats empiriques

4.4.3.1 Instance 1

FIGURE 4.4: Instance 1 : Fronts de Pareto cumulés pour tous les runs, à la fin de chaque run.



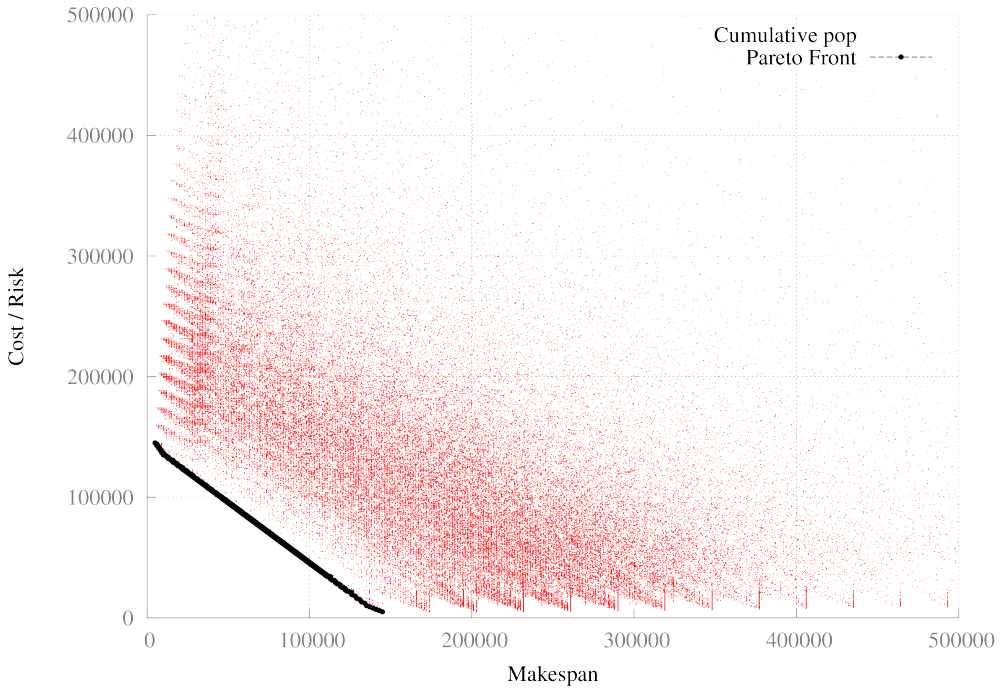
La figure 4.4 montre l'ensemble des Front de Pareto trouvé pour chaque run. Si l'ensemble des points ne sont pas trouvés à chaque instant, les fronts sont tout de même relativement proches du front exact par rapport à l'ensemble des vecteurs objectifs atteignables.

Il est cependant intéressant de noter malgré la qualité des fronts trouvés, l'échantillonnage de l'espace objectif possède clairement une zone moins dense proche du front comme en atteste la figure 4.5. On peut également noter l'apparition de certains motifs sur les extrémités. À ce stade, il pourrait s'agir soit de la manière dont l'algorithme visite l'espace objectif ou de la structure de l'espace objectif induite par l'instance. En effet, il n'y a peut être aucun plan qui permet d'atteindre un point dans l'espace « vide » entre deux

motifs des extrémités. Globalement, il semblerait que l'espace objectif soit échantilloné sans structure apparente.

La version colorée (figure 4.6) permet de visualiser si chaque *run* échantillonne de la même manière l'espace objectif ou si certains *runs* se concentrent dans certaines zones. Ce n'est à priori pas le cas et ne le sera pour aucune stratégie ou aucune instance.

FIGURE 4.5: Instance 1 : Population cumulées pour tous les runs et à tout temps.

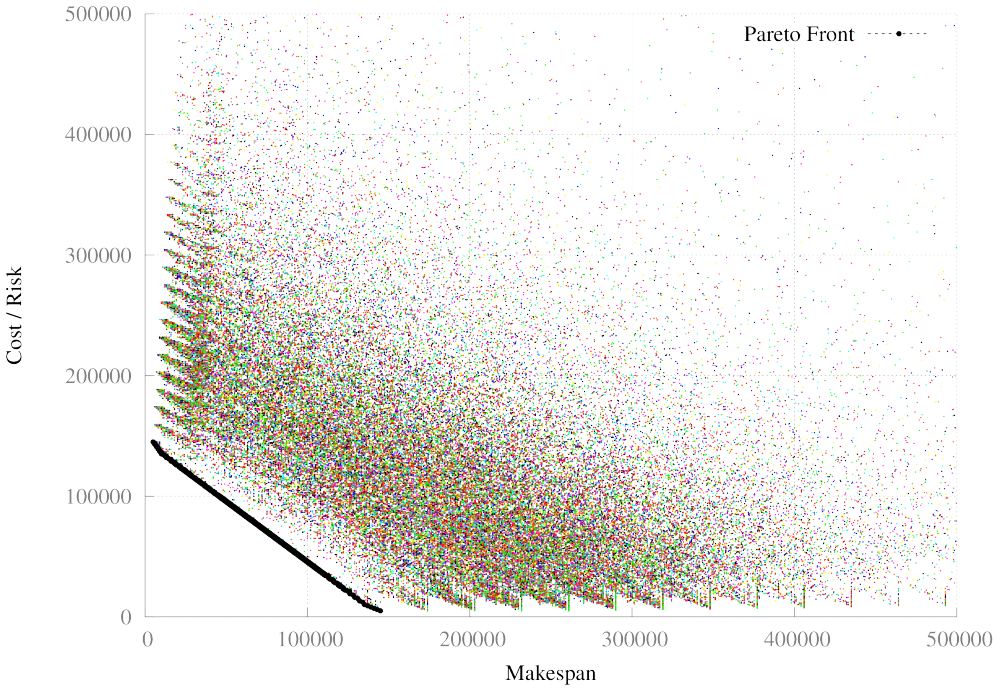


Les surfaces d'atteintes de la figure 4.7 cachent cette disparité près du front et indique une faible variance : il est assez difficile de distinguer le gradient de couleur, le moins bon des *runs* étant tout de même assez proche du front exact.

Un phénomène intéressant au niveau des trajectoires de l'hypervolume de la figure 4.8 et qui est caché si l'on trace uniquement l'hypervolume moyen est le « saut » qui semble s'opérer à des moments variés selon les *runs*. Il semblerait exister 2 phases : la première phase consiste en une diminution très rapide de l'hypervolume, qui, comme nous le verrons plus tard, s'explique probablement par les capacités ou plutôt limites du solveur local. Cette diminution conduit à un état plus ou moins stationnaire. Enfin, un saut se manifeste, vers un autre état lui aussi globalement stationnaire.

La diversité des vecteurs objectifs ne fait pas apparaître de structure particulière, si ce

FIGURE 4.6: Instance 1 : Population cumulées pour tous les runs et à tout temps (version colorée).



n'est une répartition globale relativement uniforme, avec cependant une petite concentration entre 0.5 et 0.8. La version colorée ne permet pas d'identifier de structure au niveau des *runs*.

FIGURE 4.7: Instance 1 : Surfaces d'atteinte pour tous les runs.

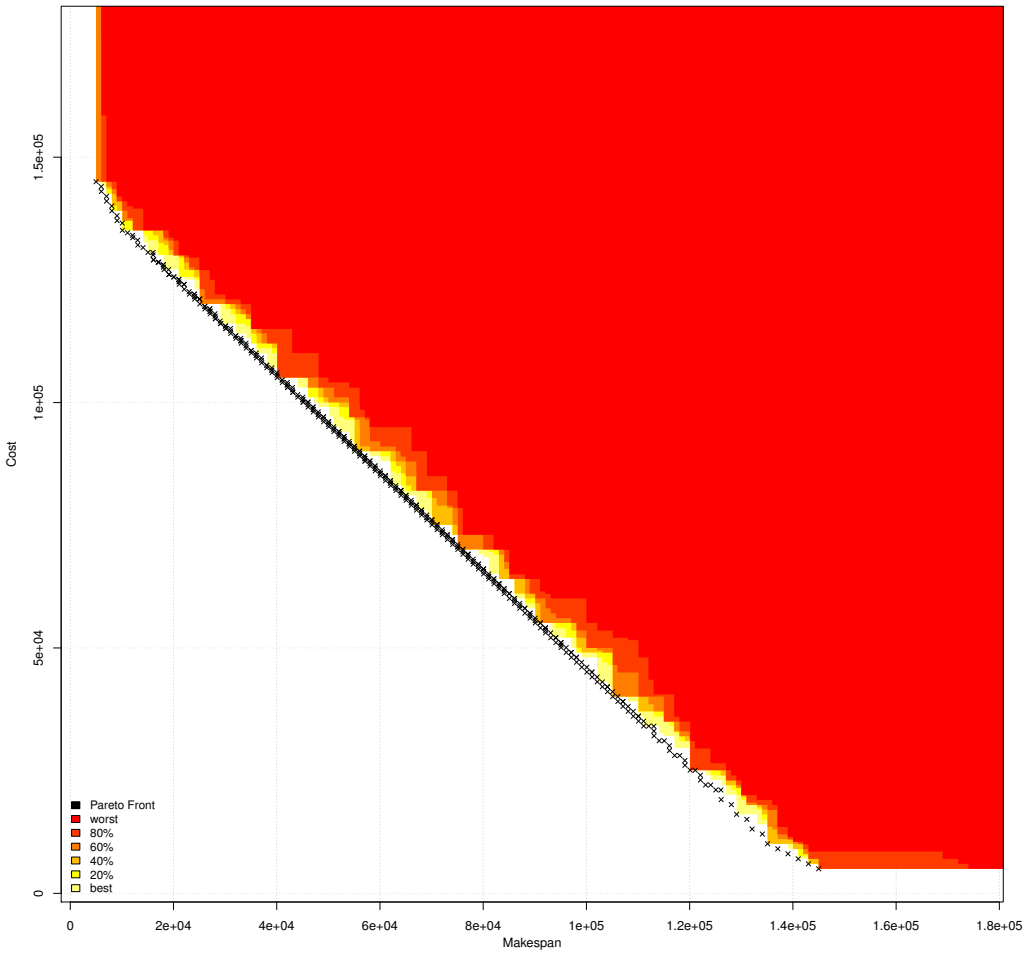


FIGURE 4.8: Instance 1 : Trajectoires de l'hypervolume pour tous les runs.

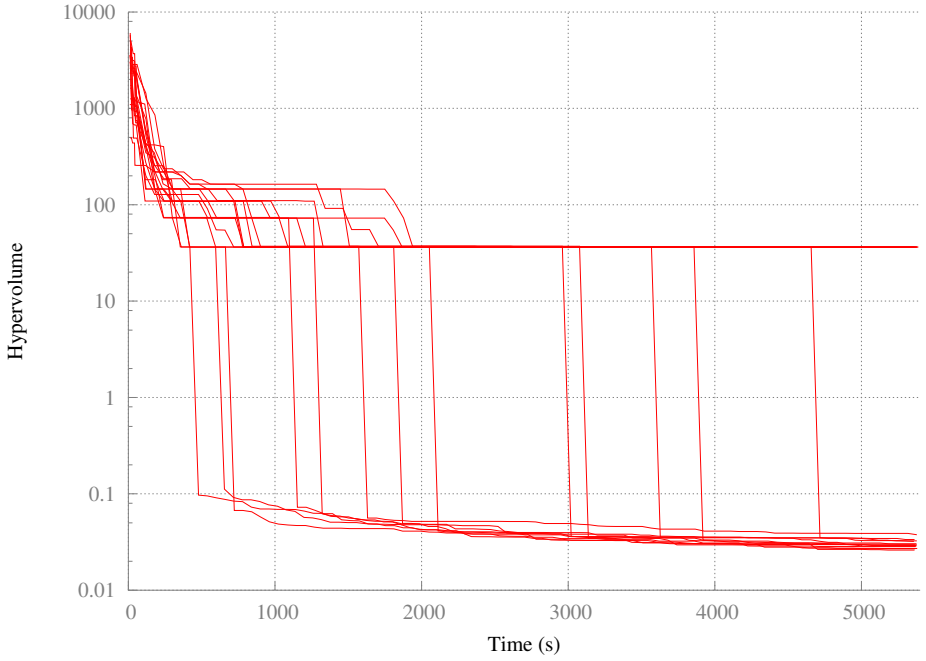


FIGURE 4.9: Instance 1 : Diversité des vecteurs objectifs pour tous les runs.

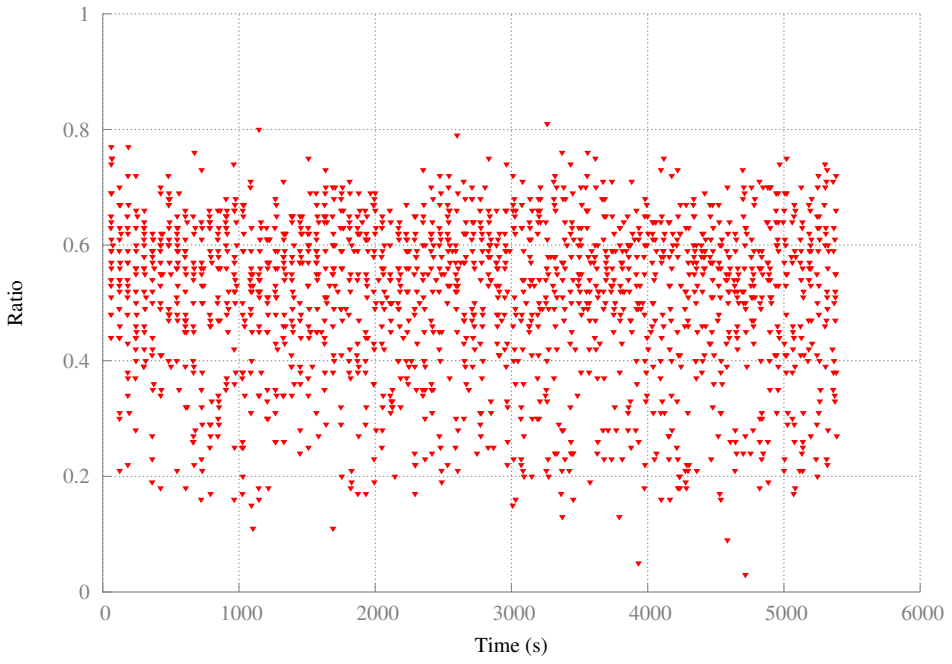
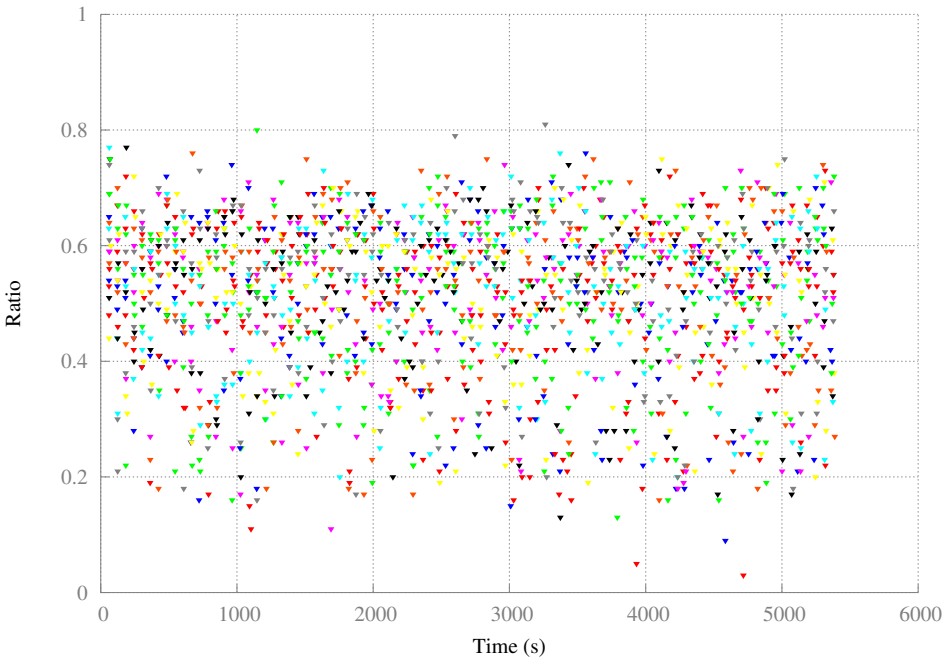
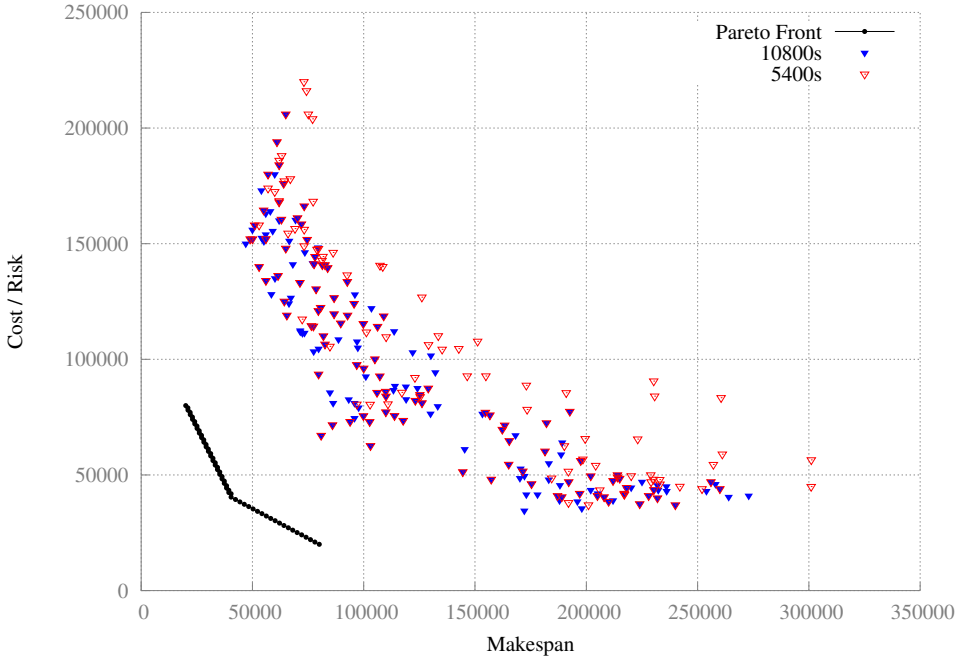


FIGURE 4.10: Instance 1 : Diversité des vecteurs objectifs pour tous les runs (version colorée).



4.4.3.2 Instance 4

FIGURE 4.11: Instance 4 : Fronts de Pareto cumulés pour tous les runs, à la fin de chaque run (5400 puis 10800s).



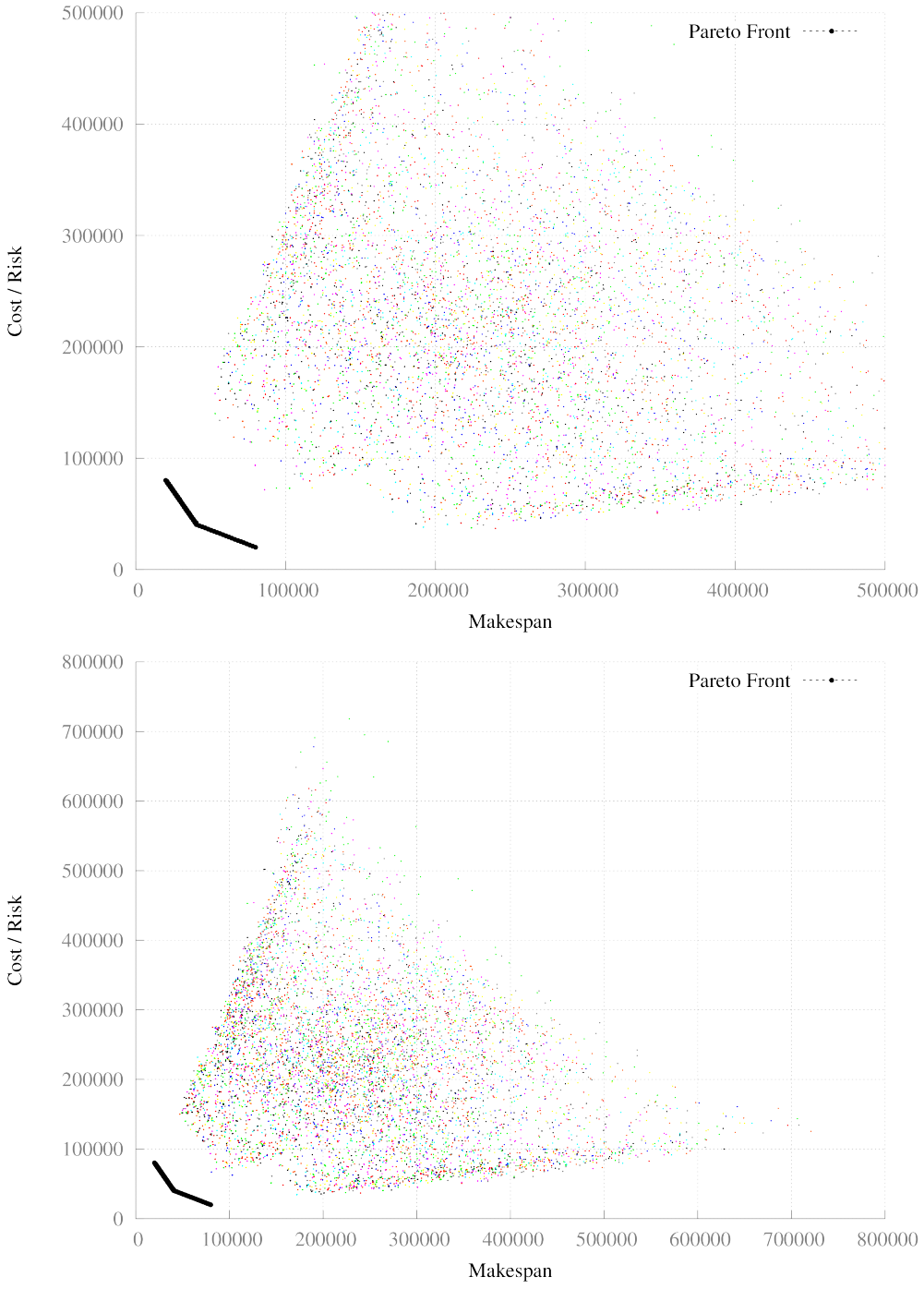
La figure 4.11 montre l'évolution des fronts cumulés pour tous les *runs* entre 5400 et 10800 secondes. Il est très clair que la progression a été quasi-inexistante par rapport aux premiers instants si l'on se réfère à la figure 4.15 de l'hypervolume. Il est raisonnable de penser que cette progression entre 5400s et 10800s est en réalité le produit de mutations très chanceuses et non pas d'un phénomène d'optimisation des structures des individus qui tendrait à rendre les sous-problèmes qu'ils représentent plus facile à résoudre et conduirait à de meilleurs plans.

À l'instar de l'instance 1, l'échantillonnage de l'espace objectif ne présente pas de structure particulière. Nous n'observons plus les motifs qui apparaissaient sur les extrémités. Le passage de 5400s à 10800s ne fait qu'échantillonner avec un peu plus d'insistance des zones déjà explorées. On observe une zone peu ou pas échantillonnée proche du front, ce qui est cette fois tout à fait logique à la vue des fronts cumulés. Cette zone « à vide » est également confirmée par les surfaces d'atteinte ou l'hypervolume qui ne présente pas de « sauts ».

En première hypothèse il semblerait qu'il existe une zone de l'espace objectif, proche du front, qui soit plus ou moins dur à atteindre. Il existe deux potentielles raisons : il

FIGURE 4.12: Instance 4 : Population cumulées pour tous les runs et à tout temps (5400 puis 10800s).

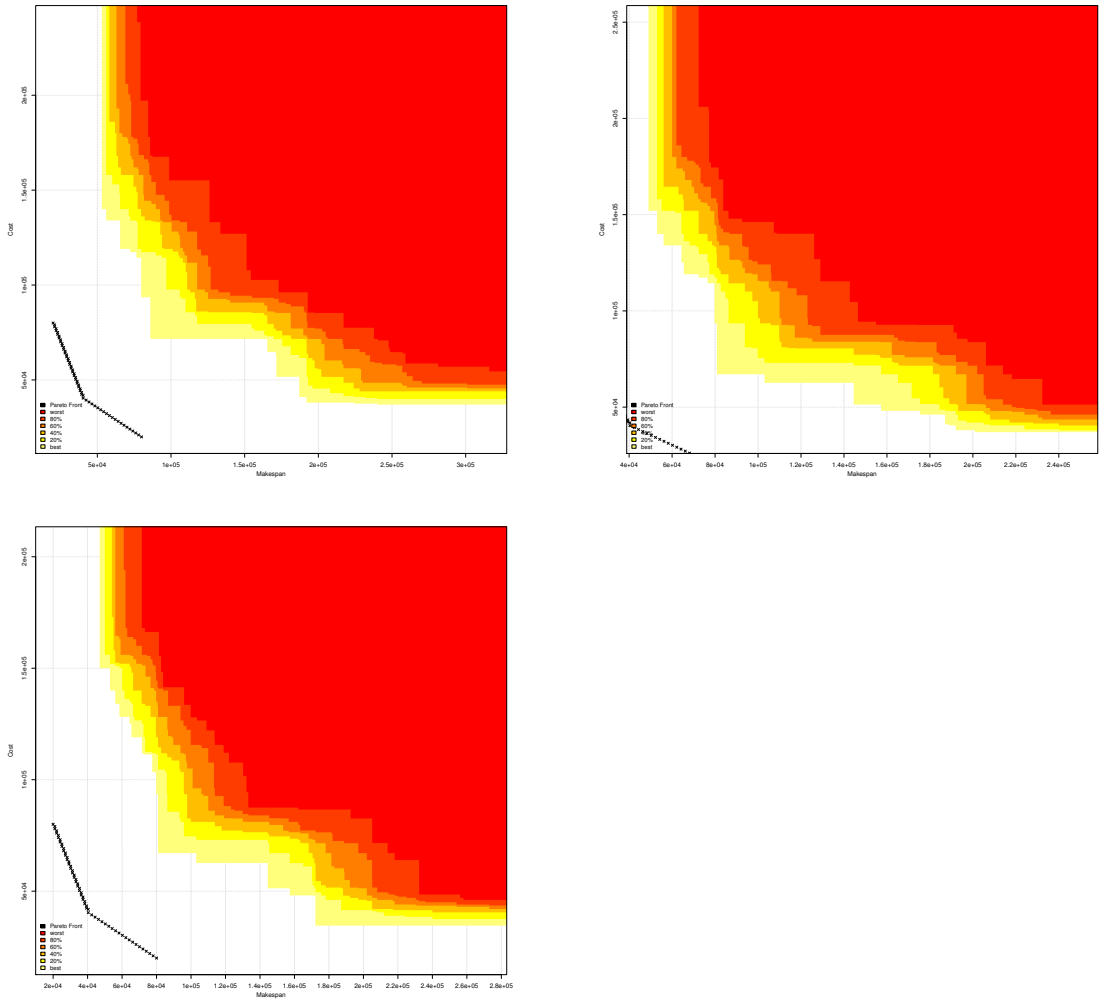
FIGURE 4.13: Instance 4 : Population cumulées pour tous les runs et à tout temps (version colorée) (5400 puis 10800s).



y a peu de plans qui résultent en des vecteurs de cette zone ou la structure entre les individus de cette zone et celle échantillonnée n'est pas la même et il est donc difficile d'atteindre cette zone par des cross-over ou des mutations ponctuelles.

La seconde hypothèse semble la plus vraisemblable et la question qui se pose alors est : est-il possible d'accéder au Front de Pareto sans passer par cette zone de l'espace objectif ? La réponse semble dépendre de l'instance. Dans le cas de l'instance 1 il semble clair qu'il est possible d'atteindre le front sans échantillonner cette zone, alors que dans le cas de l'instance 4 il semblerait que cela ne soit pas possible.

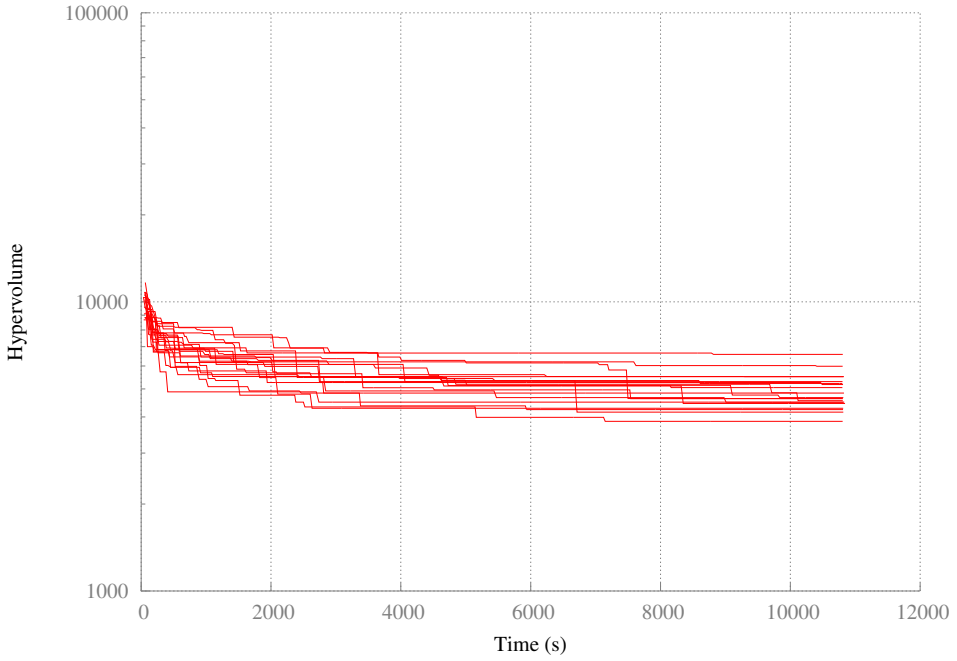
FIGURE 4.14: Instance 4 : Surfaces d'atteinte pour tous les runs (2500, 5400 puis 10800s).



Les surfaces d'atteinte de la figure 4.14 présentent également une variance assez faible, sur l'ensemble du processus. On distingue cependant bien mieux la zone d'atteinte du meilleur runs. En règle générale, plus la variance est grande au sein des surfaces d'atteinte

et plus on peut penser que l'influence du hasard est importante dans les résultats obtenus par l'algorithme. Au sein de DAE, le hasard intervient à deux niveaux : les opérateurs de diversités et également l'évaluation via YAHSP.

FIGURE 4.15: Instance 4 : Trajectoires de l'hypervolume pour tous les runs.



Comme on pouvait s'y attendre, l'hypervolume ne présente pas de sauts et devient pratiquement stationnaire après quelques minutes, aucune trajectoire ne se distinguant particulièrement. On remarque ici aussi deux phases : l'une de décroissance très rapide de l'hypervolume, puis une stabilisation.

La diversité des vecteurs objectifs est très surprenante puisqu'elle décroit de manière exponentielle comme en atteste la figure 4.16. On voit pourtant que l'on part d'une diversité assez importante, de l'ordre de 85%, ce qui d'autant plus significatif que le nombre d'individus est de 200. Autrement dit, sur 200 individus, nous obtenons 170 différents points dans l'espace objectif. Au contraire, après 300 à 400 secondes, la diversité tombe sous les 10% et la version colorée de la figure 4.17 permet de suivre clairement la trajectoire du *run* représenté en vert. Notons que l'atteinte de la stationnarité intervient au même instant que la stationnarité de l'hypervolume.

Cependant, l'instance 1 présente aussi une certaine stationnarité et la diversité reste relativement haute et surtout sans structure durant l'ensemble du processus.

FIGURE 4.16: Instance 4 : Diversité des vecteurs objectifs pour tous les runs.

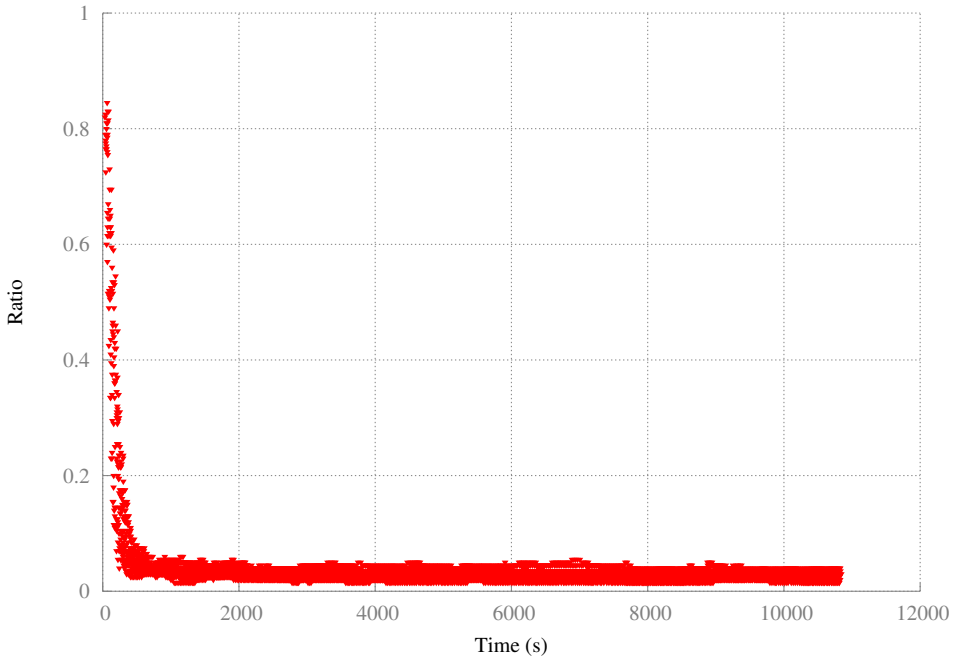
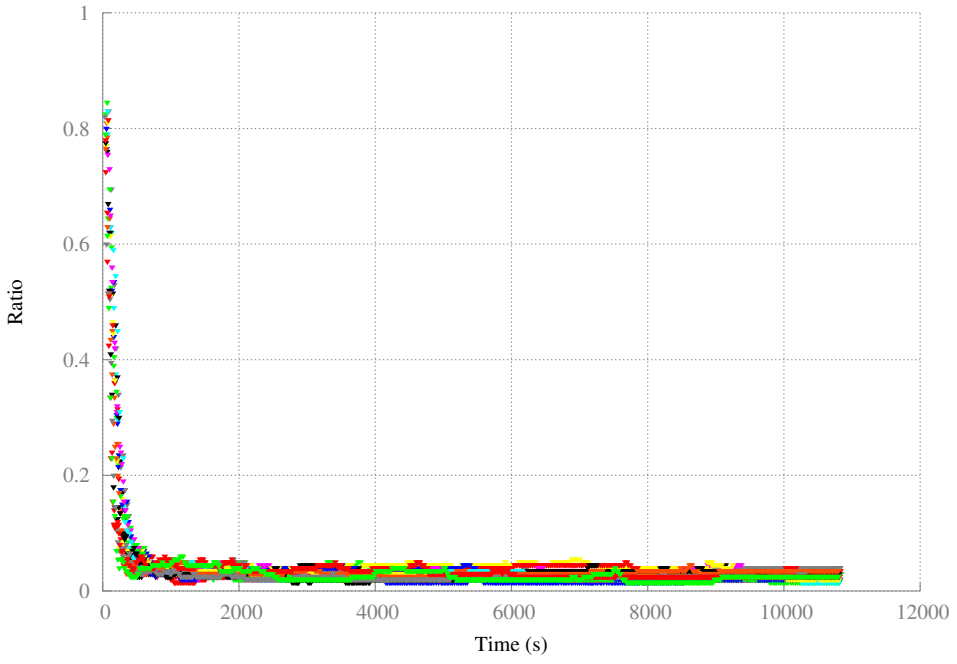


FIGURE 4.17: Instance 4 : Diversité des vecteurs objectifs pour tous les runs (version colorée).



Evolution au cours du temps Il s'agit ici de comparer valider statistiquement l'analyse qui a été faite sur les fronts cumulés de l'image 4.11. La figure 4.18 présente côte à côte les surfaces d'atteinte respectivement après 10800 et 5400 secondes, ainsi que les différences de probabilité d'atteinte. La figure 4.19 présente la même chose mais pour 10800 et 2500 secondes.

Ces deux ensembles d'images fournissent un résultat clair : les *runs* sont statistiquement meilleurs au fur et à mesure que le temps avance. Les différences entre 2500s et 10800s puis 5400s et 10800s sont particulièrement clair puisqu'il n'existe aucune zone où 2500s ou 5400s ne soient significativement meilleurs que 10800s. Un test sur les fonctions d'atteinte de premier et second ordre désignent toujours 10800s comme statistiquement la meilleure version.

L'algorithme est capable de continuer à progresser malgré une diversité extrêmement faible. Cela ne répond cependant pas à la question de savoir si ces progressions sont le fruit du hasard.

La baisse subite et brutale de la diversité des vecteurs objectifs dans la population étant intriguante, nous avons essayé de mettre en perspective l'évolution de la population au cours du temps dans l'espace objectif. Ainsi, la figure 4.20 montre l'évolution des vecteurs objectifs aux temps 100, 200, 500 et 2000 secondes, soit après environ 1%, 1.8%, 4.6% et 18% du temps total de 10800 secondes dont ont bénéficiés les *runs* les plus longs.

Il est assez impressionnant de voir comment l'ensemble de la population va converger vers certains individus et ce de manière très rapide. Après 4.6% du temps observé, des 200 vecteurs objectifs potentiellement différents que l'on pourrait obtenir à l'aide de la population, il n'en reste que 8 vecteurs différents alors qu'ils étaient très nombreux après environ 1% du temps observé. Il est évident que cette situation ne favorise pas la découverte de nouveaux points qui tendent à se faire uniquement à l'aide du hasard. En effet, en considérant qu'une large part de la population résulte en le même vecteur objectif il est raisonnable de penser qu'ils possèdent le même génotype (ce qui serait à confirmer avec l'étude du nombre de plans permettant de donner le même vecteur objectif, étude difficile à réaliser du fait de la combinatoire très élevée). S'il n'y a pas de mutation, seul un croisement à un point résultat d'un seul individu est conservé : celui qui conserve le mieux l'ordre chronologique approximatif des différents états qui le compose. Si l'on sélectionne deux individus identiques x et x' et qu'on les croise, on

FIGURE 4.18

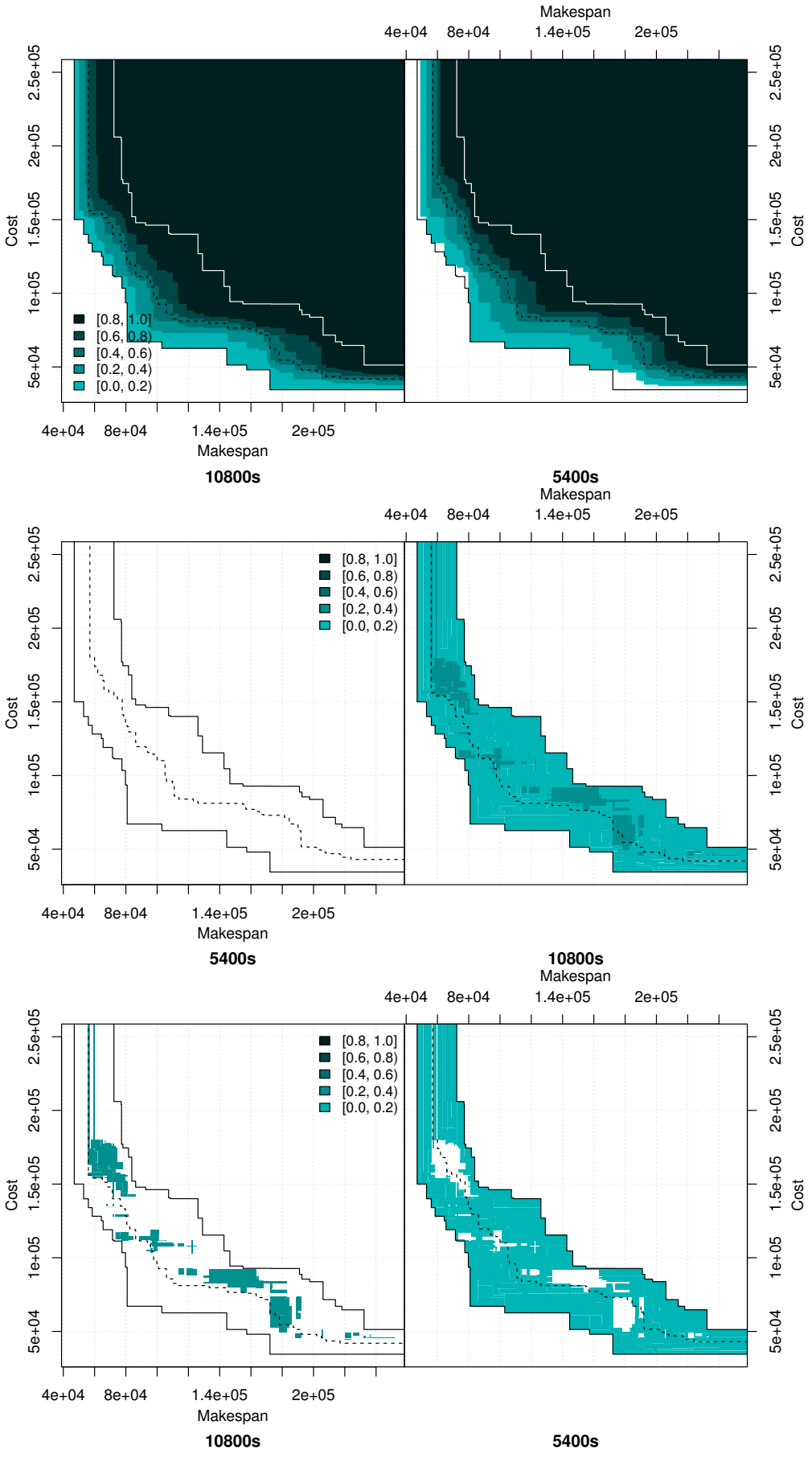
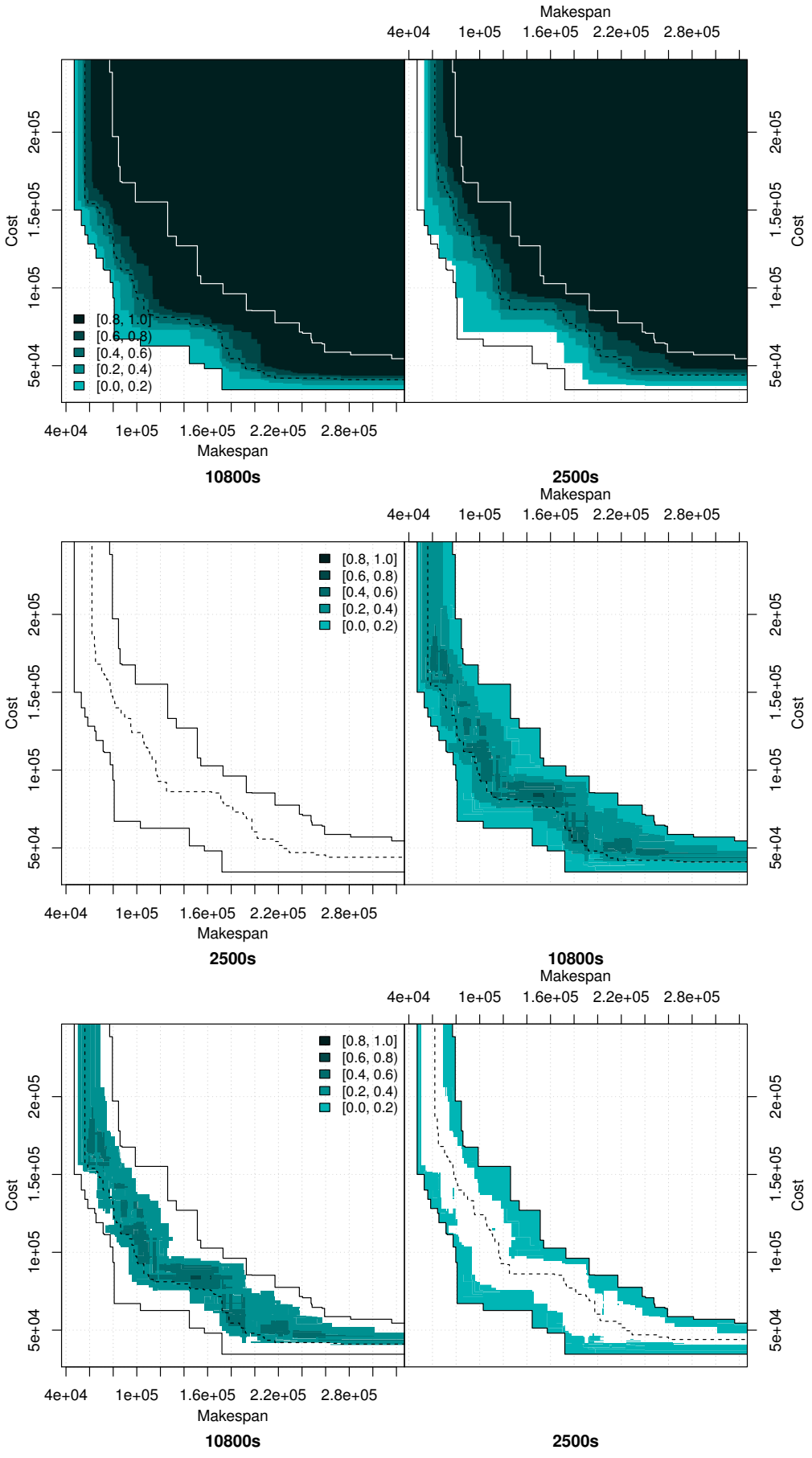

Makespan
4e+04
8e+04
1.4e+05
2e+05
Cost
5e+04
1e+05
1.5e+05
2e+05
2.5e+05
[0.8, 1.0]
[0.6, 0.8)
[0.4, 0.6)
[0.2, 0.4)
[0.0, 0.2)
10800s
5400s
5400s
10800s
10800s
5400s

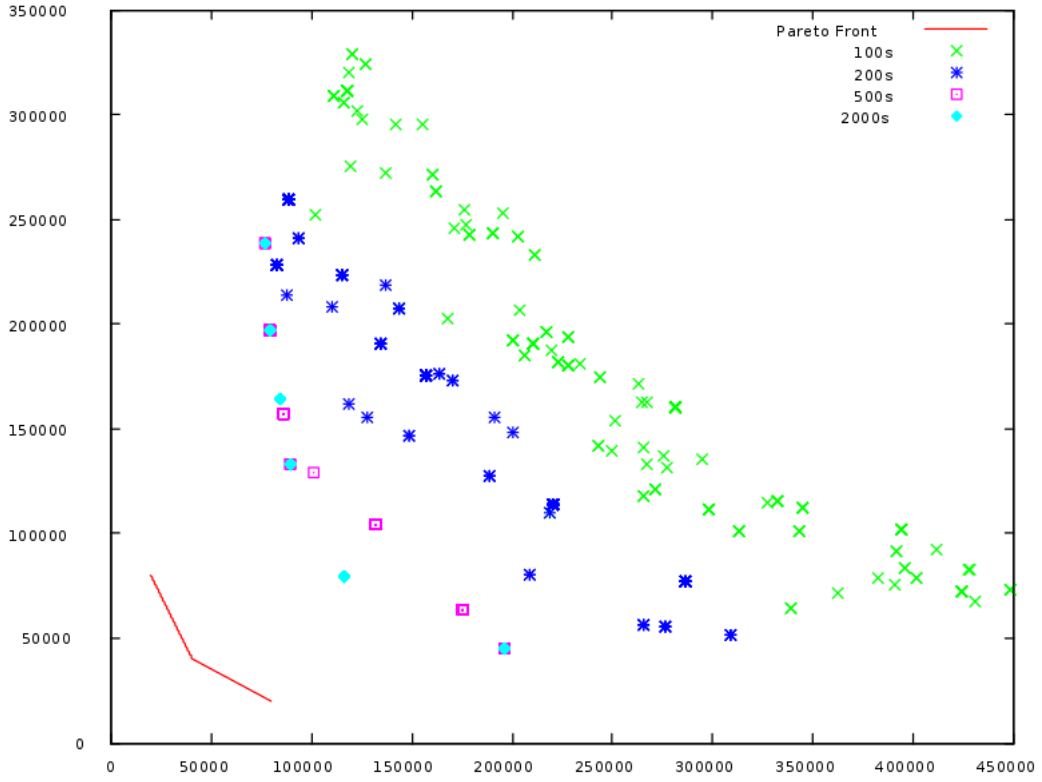
FIGURE 4.19


Makespan
4e+04
1e+05
1.6e+05
2.2e+05
2.8e+05
Cost
5e+04
1e+05
1.5e+05
2e+05
[0.8, 1.0]
[0.6, 0.8)
[0.4, 0.6)
[0.2, 0.4)
[0.0, 0.2)
10800s
2500s
2500s
10800s
10800s
2500s

obtiendra un individu y_i au sein de la famille $(y_i)_i$ des individus obtenus pour chaque position possible du croisement. Si quelque soit y_i le vecteur objectif de y_i est dominé par celui de x, alors y_i sera progressivement éliminé de la population voire non intégré. En règle générale, si l'individu généré est moins bon par rapport à l'indicateur utilisé par IBEA, alors il n'a que très peu de chance d'être intégré ou de subsister. Il résulte donc d'une convergence prématurée de l'algorithme qui n'arrive pas à rétablir suffisamment de diversité pour relancer le processus. Croiser des individus ne sert plus à rien et seules les mutations permettent d'apporter de la diversité.

Lorsqu'une mutation est chanceuse et produit un individu dont le vecteur objectif n'est pas dominé (ou dont la fitness qui dépend de l'indicateur utilisé est suffisamment bonne), il faut alors qu'elle soit encore plus chanceuse pour que le potentiel croisement avec les peu d'individus différents puisse mener à nouveau à un individu qui soit meilleur, ce qui dépend alors également du point de croisement choisi, etc.

FIGURE 4.20: Instance 4 : Evolution des populations (static).



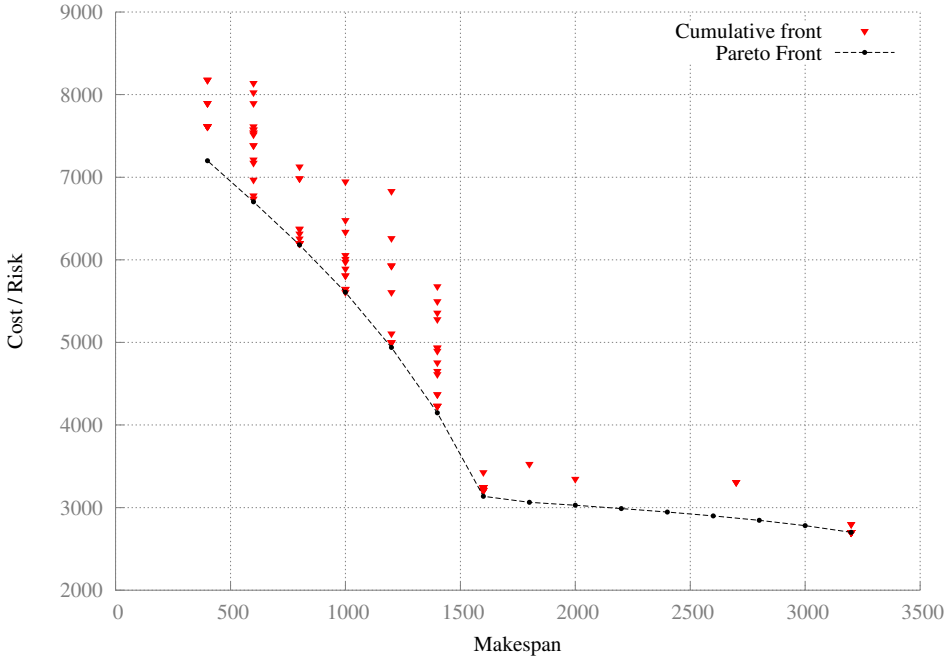
4.4.3.3 Instance 7

L'instance 7 est probablement la plus compliquée en terme de combinatoire d'après les paramètres qui ont servis à sa génération via le ZenoSolver. Les résultats sont cependant clairement meilleurs ou tout du moins différents de ceux de l'instance 4.

Les fronts cumulés de la figure 4.21 indique que la zone du front la plus compliquée à atteindre est la zone faiblement convexe dont les plans possèdent un faible coût. Il est intéressant de noter également qu'il semblerait que les points des front soient alignés pour un *makespan* donné, ce qui serait révélateur de la structure de l'espace objectif ou à défaut, de la manière dont DAE le parcours.

La figure 4.22 montrant la population cumulée confirme cette observation. Des motifs apparaissent très clairement. L'espace objectif est strié verticalement, comme si pour une valeur donnée du *makespan* il existait assez de plans aux coûts différents pour faire apparaître des lignes qui semblent continues. On remarque aussi que certaines stries semble plus « visitées » que d'autres, et que ces lignes favorables sont équidistantes.

Figure 4.21: Instance 7 : Fronts de Pareto cumulés pour tous les runs, à la fin de chaque run.



Contrairement à ce que laisserait penser la figure 4.21 des fronts cumulés, une majorité de points du front ne sont pas atteint, même sur la partie gauche, même si l'on en est

FIGURE 4.22: Instance 7 : Population cumulées pour tous les runs et à tout temps.

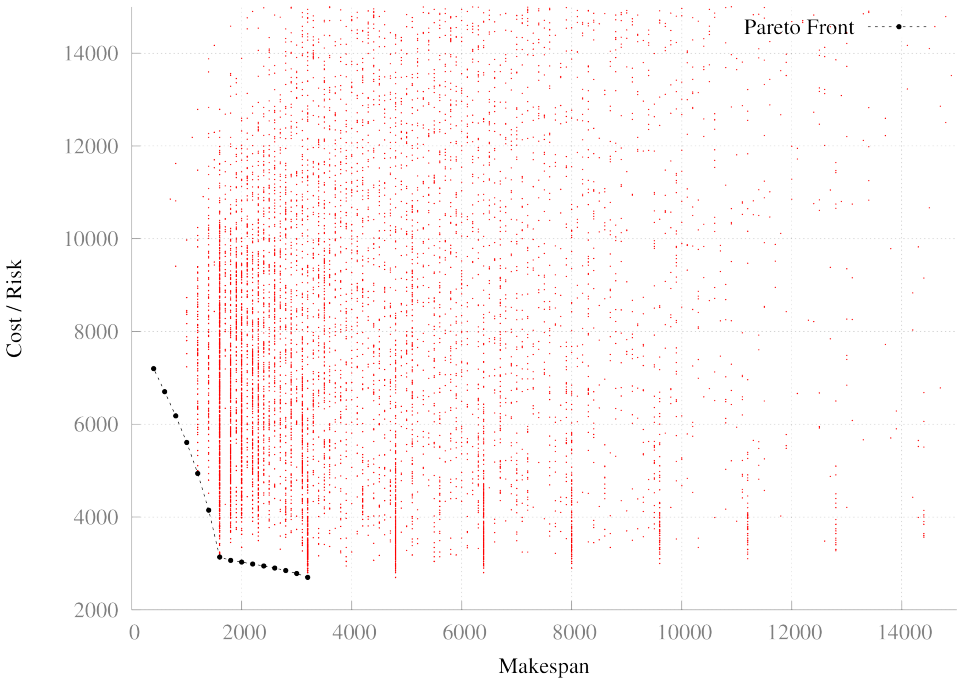
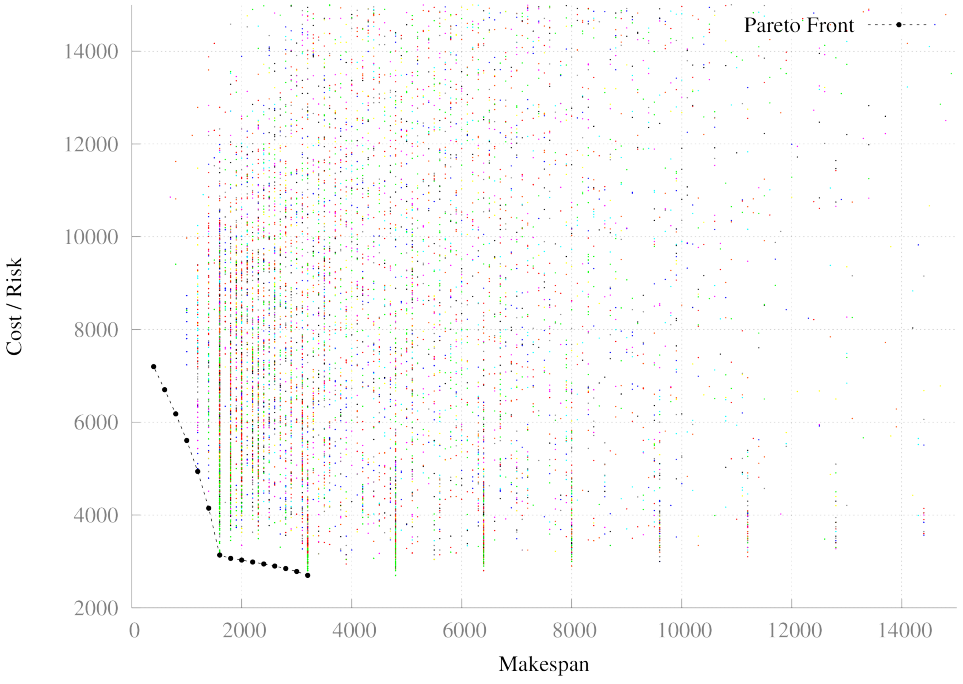
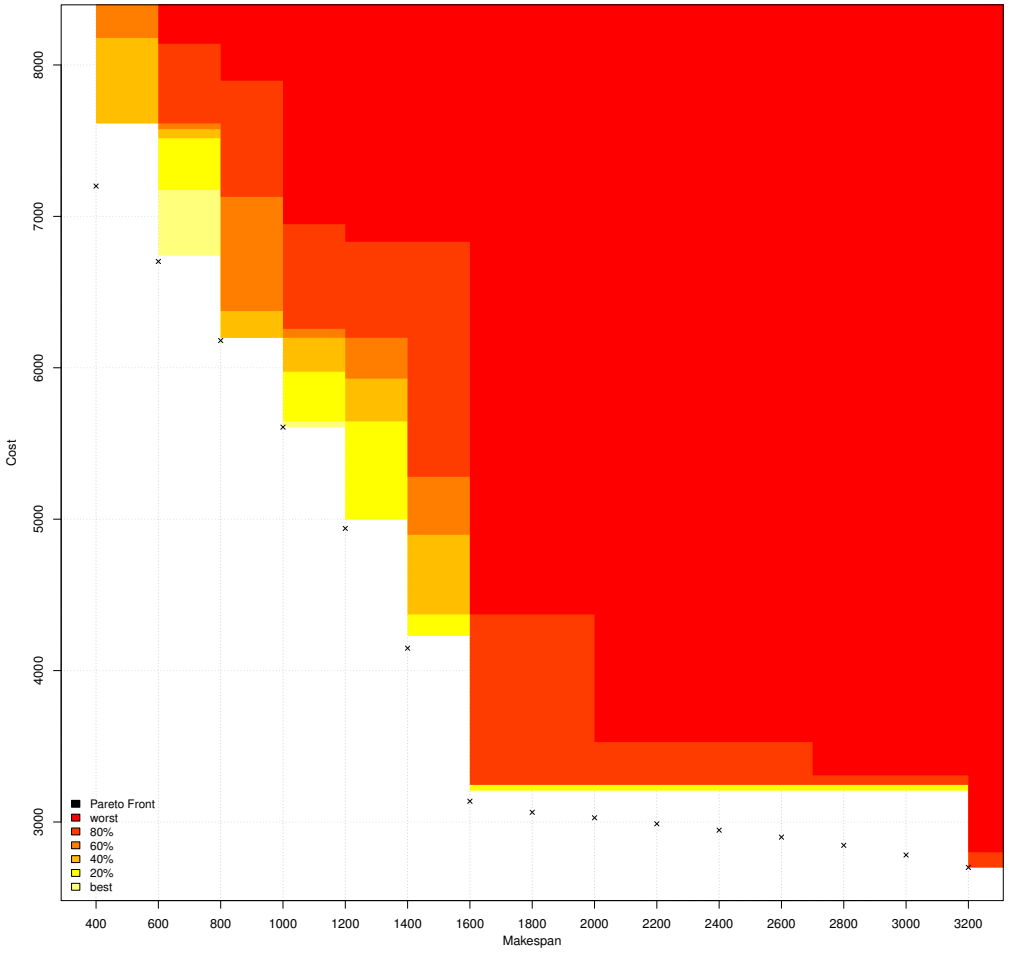


FIGURE 4.23: Instance 7 : Population cumulées pour tous les runs et à tout temps (version colorée).



très proche, comme en témoigne les surfaces d'atteinte de la figure 4.24. Cette dernière permet à nouveau d'observer cette structure en stries.

FIGURE 4.24: Instance 7 : Surfaces d'atteinte pour tous les runs.



Les trajectoires de l'hypervolume sont cette fois très différentes de ce que l'on a pu observer jusqu'à présent. Tout d'abord, de part la difficulté de résolution du problème de planification et par la taille de la population, le temps entre deux générations est très important, ce qui explique que les trajectoires ne commencent pas à 0 ou aux alentours mais parfois après plus de 1500 voire 2000 secondes. On remarque par contre des sauts extrêmement importants pour la majorité des runs. L'importance des sauts cachent peut-être la première phase de descente extrêmement rapide de l'hypervolume puis la stagnation.

L'évolution de la diversité des vecteurs objectifs, montrée par les figures 4.26 et 4.27 indique également une structure différente. La diversité est tout d'abord très élevée, à

100% pour certains *runs*, et va à partir d'un moment, diminuer drastiquement. La version colorée permet réellement d'observer ce phénomène, par exemple avec la trajectoire « cyan », « noire » ou « verte » qui se distingue particulièrement bien. Notons que cette baisse intervient quelques générations après le saut dans l'hypervolume du *run* correspondant.

On retrouve donc le comportement de l'instance 4, à savoir une baisse subite de la diversité, mais cette fois elle n'intervient pas tout de suite dans le processus. On retrouve également le comportement de l'instance 1 vis à vis de l'hypervolume, à savoir un saut à un moment donné, variable selon les *runs*, mais cette fois sans le maintient de la diversité.

FIGURE 4.25: Instance 7 : Trajectoires de l'hypervolume pour tous les runs.

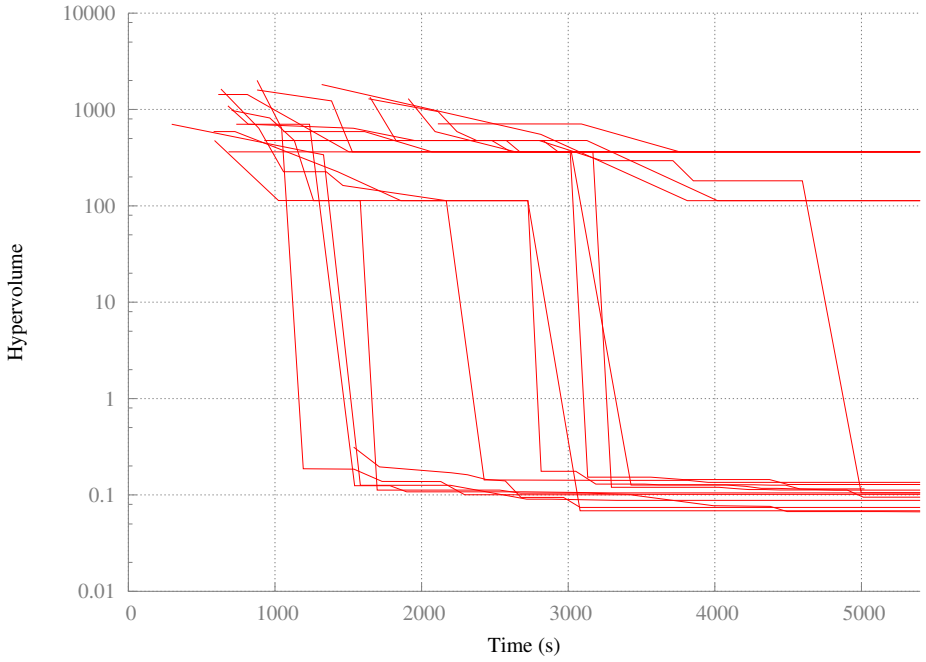


FIGURE 4.26: Instance 7 : Diversité des vecteurs objectifs pour tous les runs.

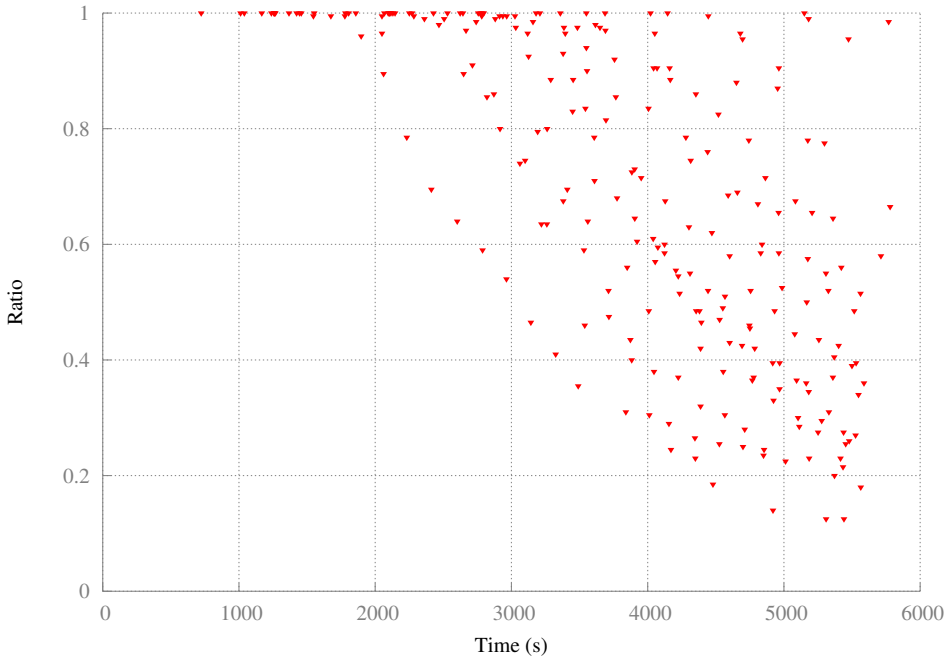
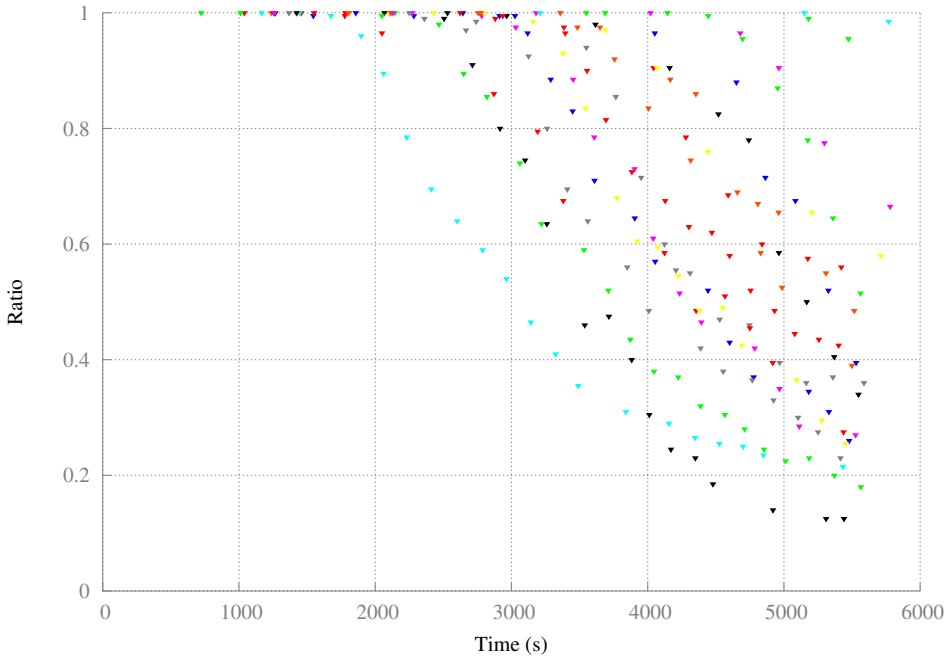


FIGURE 4.27: Instance 7 : Diversité des vecteurs objectifs pour tous les runs (version colorée).



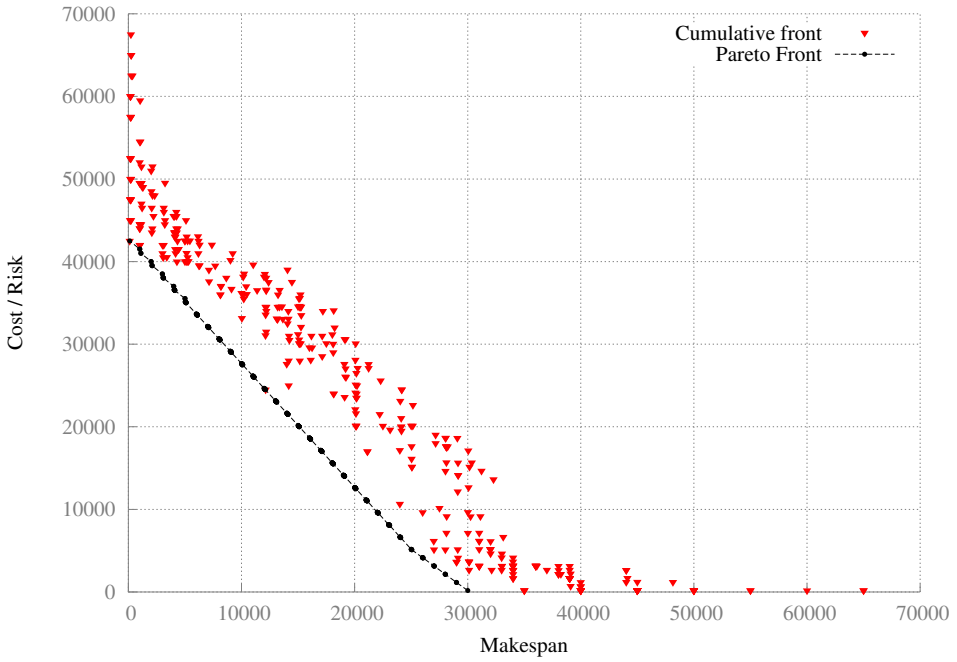
4.4.3.4 Instance 9

Comme nous avons pu le faire remarquer dans le chapitre précedant, l'instance 9 semble linéaire et proche de l'instance 1 mais en réalité elle propose des regroupements de points là où les points sont répartis uniformément sur l'instance 1 et un zoom montre également une structure peu régulière, à l'instar de l'instance 1.

Les fronts cumulés semblent indiquer à la fois qu'il est plus facile d'atteindre les extrémités du front, puisque ce sont aux extrémités que les seuls points du front exact sont trouvés, mais à la fois qu'il est difficile d'y accéder comme en atteste les points les plus loin du front exact (avec un coût minimum, mais 65000 en *makespan*, ou inversement, un *makespan* minimal et un coût de plus de 65000).

Il est intéressant de noter que l'espace objectif visité (figure 4.29) rassemble beaucoup de caractéristiques observés jusqu'à présent. Ainsi, il apparait des motifs sur les flans, une zone très densément échantillonnée, et des stries.

FIGURE 4.28: Instance 9 : Fronts de Pareto cumulés pour tous les runs, à la fin de chaque run.



Les surfaces d'atteinte ne font que traduire les fronts cumulés. Cependant, les trajectoires de l'hypervolume semblent un peu moins stationnaires que précédemment, avec aucun saut brutal. On peut tout de même, si l'on exclut les deux meilleurs runs, observer une convergence uniforme. On observe par contre, et de façon relativement marquée la

FIGURE 4.29: Instance 9 : Population cumulées pour tous les runs et à tout temps.

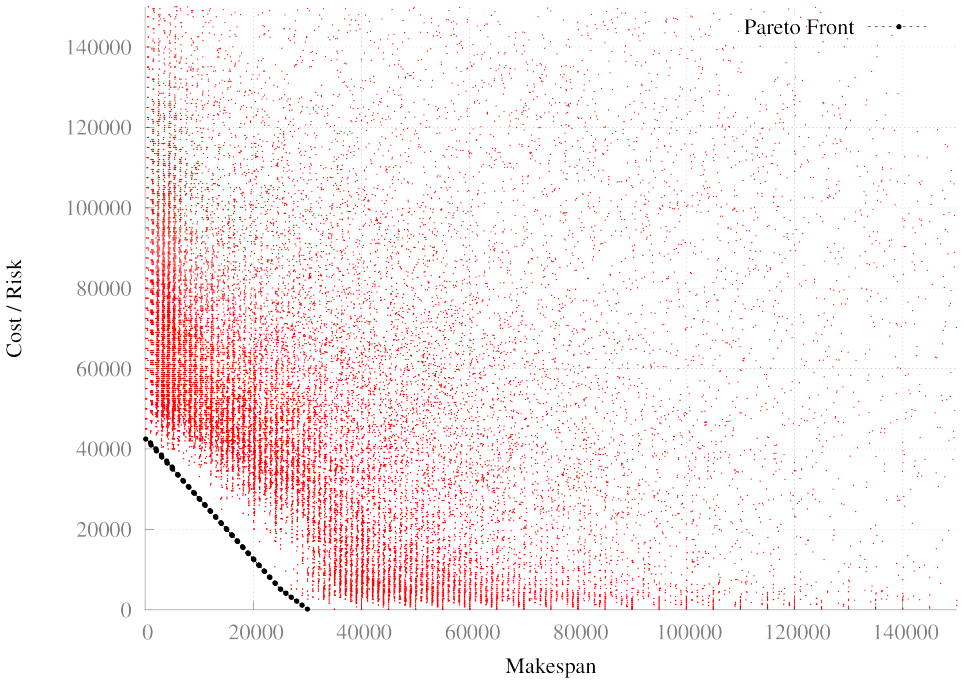
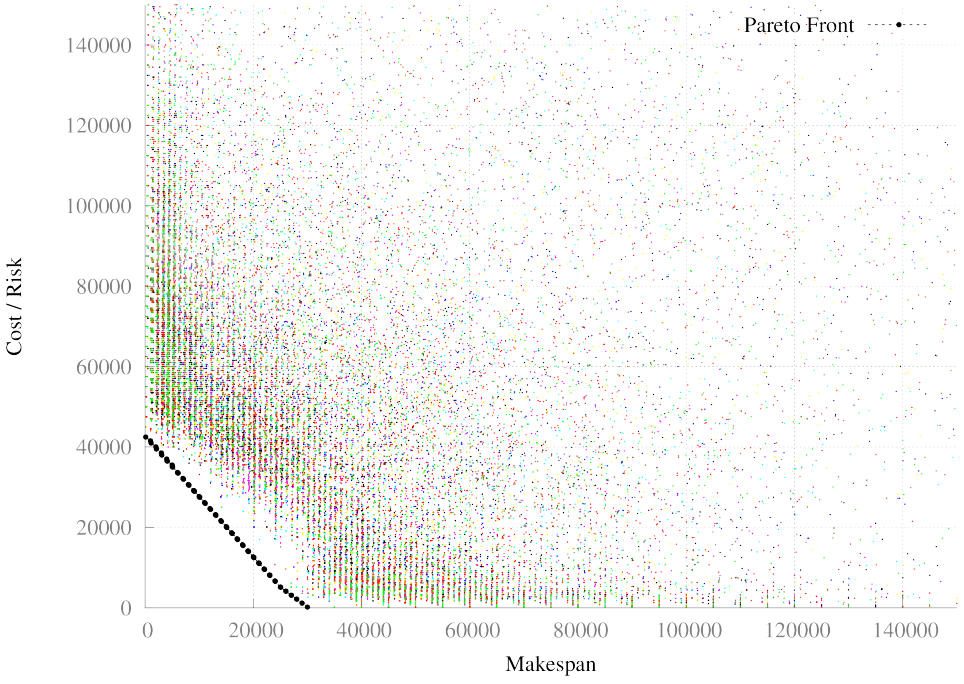
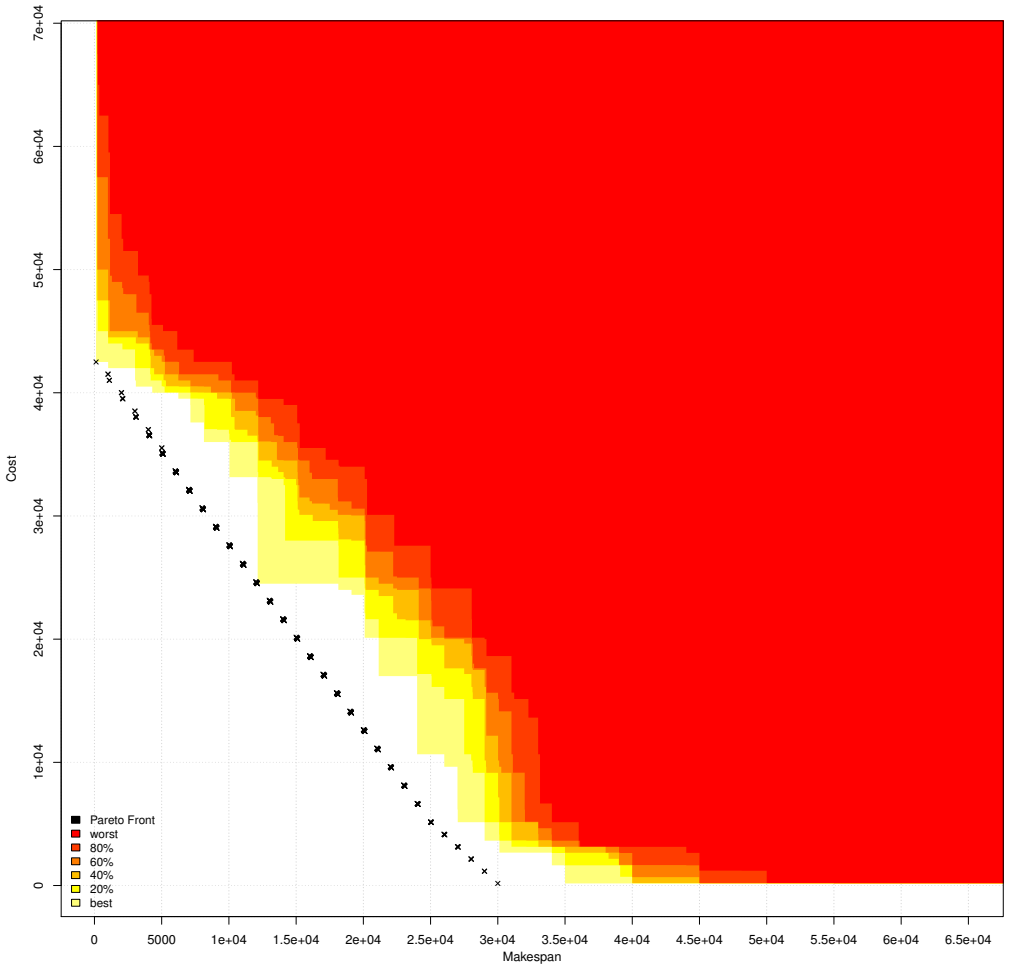


FIGURE 4.30: Instance 9 : Population cumulées pour tous les runs et à tout temps (version colorée).



diminution brutale et très rapide de l'hypervolume d'un facteur supérieur à 10 dans les premières minutes.

FIGURE 4.31: Instance 9 : Surfaces d'atteinte pour tous les runs.



La diversité des vecteurs objectifs reste très haute mais semble avoir tendance à diminuer pour tous les runs.

FIGURE 4.32: Instance 9 : Trajectoires de l'hypervolume pour tous les runs.

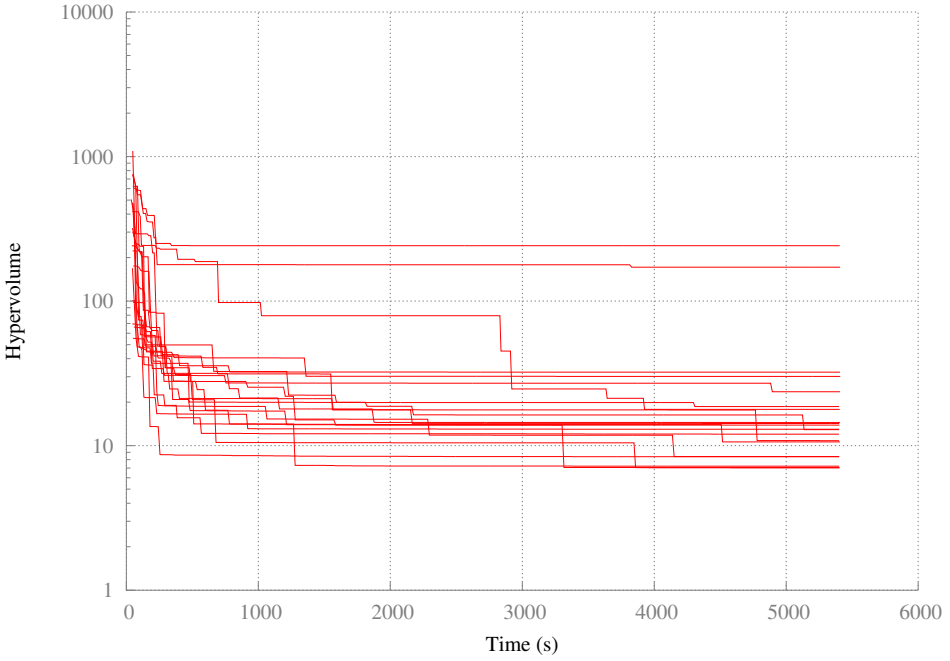


FIGURE 4.33: Instance 9 : Diversité des vecteurs objectifs pour tous les runs.

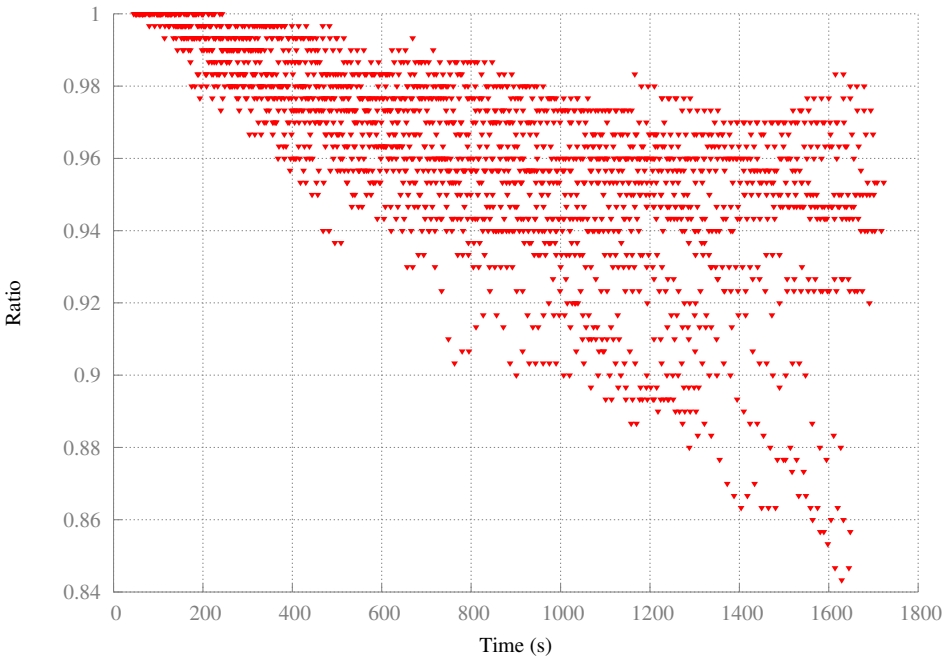
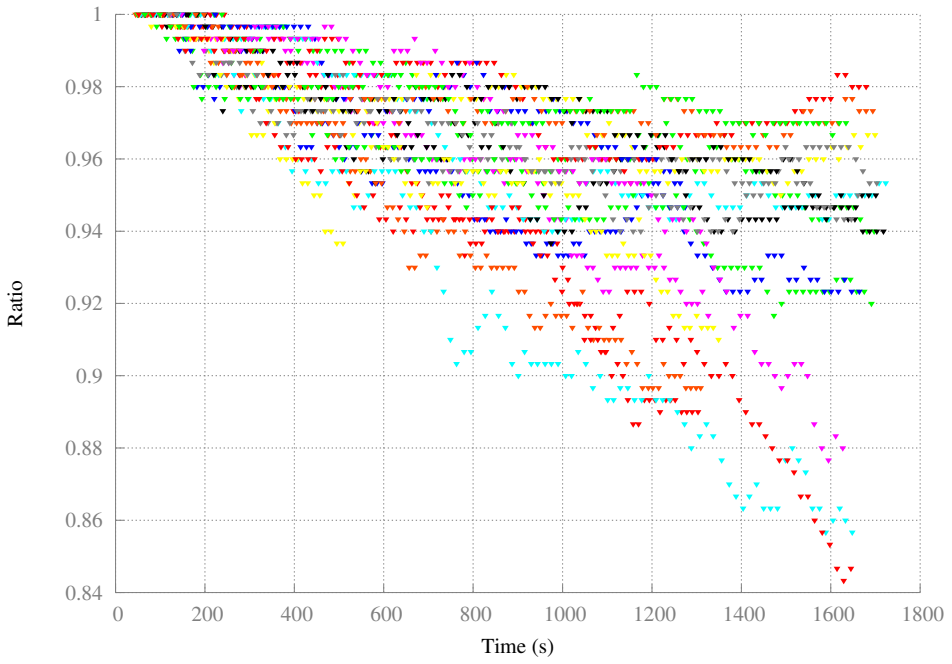


FIGURE 4.34: Instance 9 : Diversité des vecteurs objectifs pour tous les runs (version colorée).



4.4.3.5 Zeno9

Bien que l'instance soit beaucoup plus petite en terme de combinatoire, on retrouve de fortes similitudes avec l'instance 1 et 7, notamment une zone qui semble difficile à atteindre proche du front et des sauts caractéristiques dans l'hypervolume.

FIGURE 4.35: Instance Zeno9 : Fronts de Pareto cumulés pour tous les runs, à la fin de chaque run.

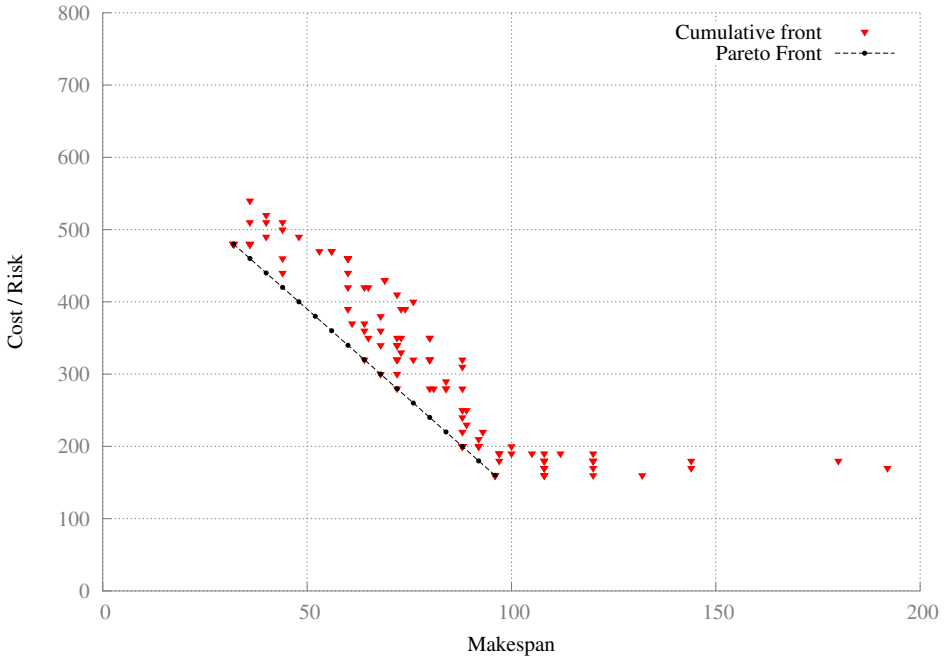


FIGURE 4.36: Instance Zeno9 : Population cumulées pour tous les runs et à tout temps.

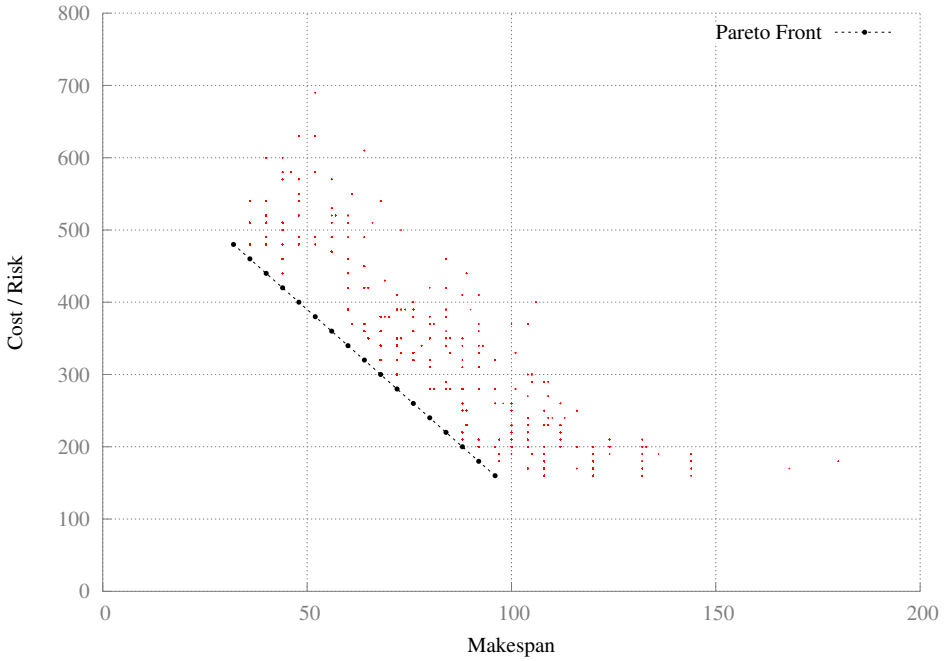


FIGURE 4.37: Instance Zeno9 : Population cumulées pour tous les runs et à tout temps (version colorée).

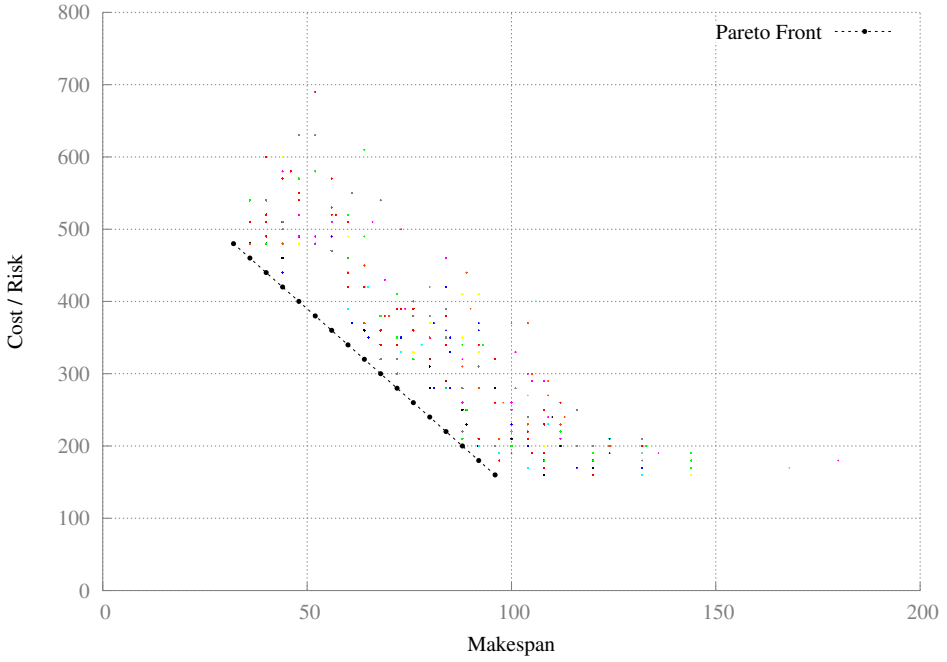


FIGURE 4.38: Instance Zeno9 : Surfaces d'atteinte pour tous les runs.

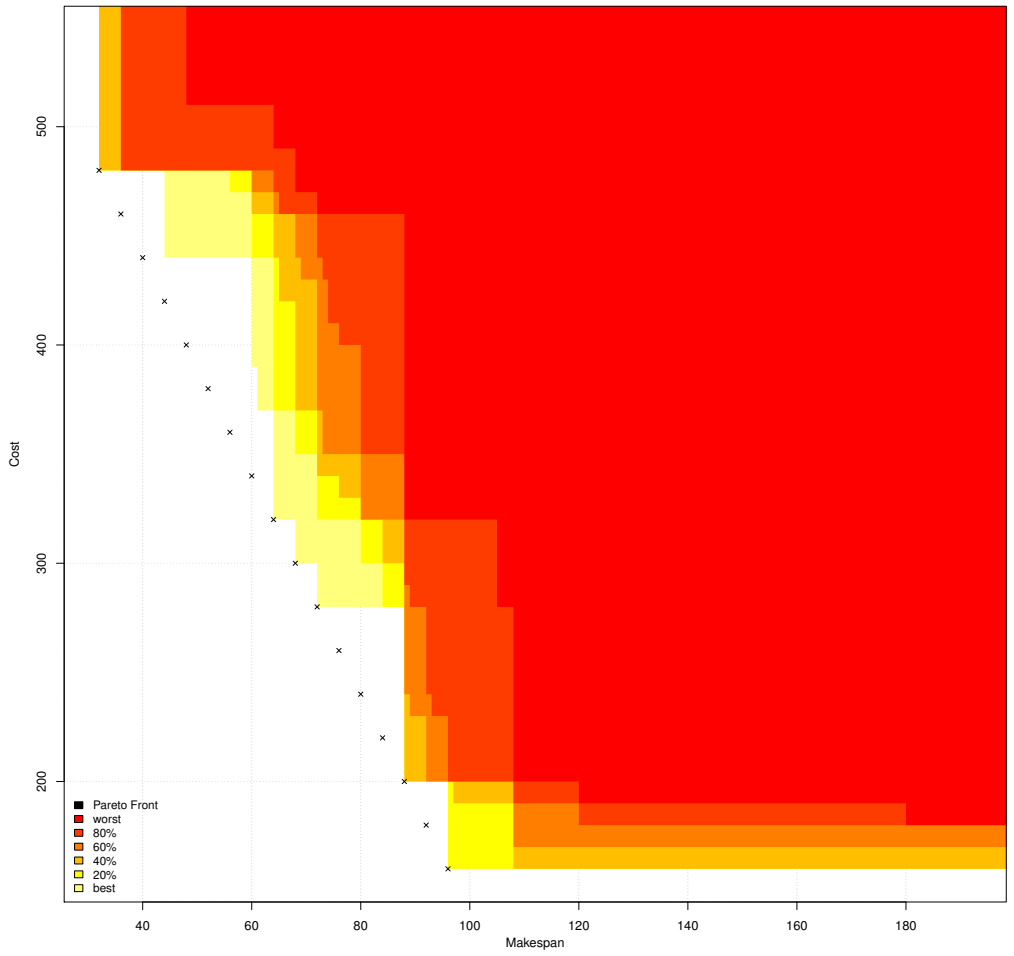


FIGURE 4.39: Instance Zeno9 : Trajectoires de l'hypervolume pour tous les runs.

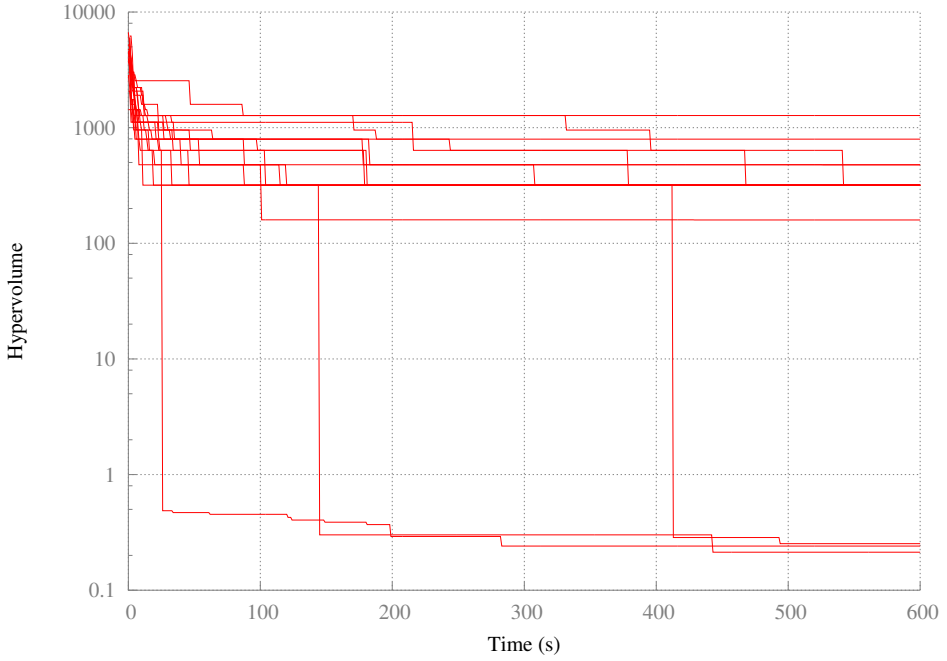


FIGURE 4.40: Instance Zeno9 : Diversité des vecteurs objectifs pour tous les runs.

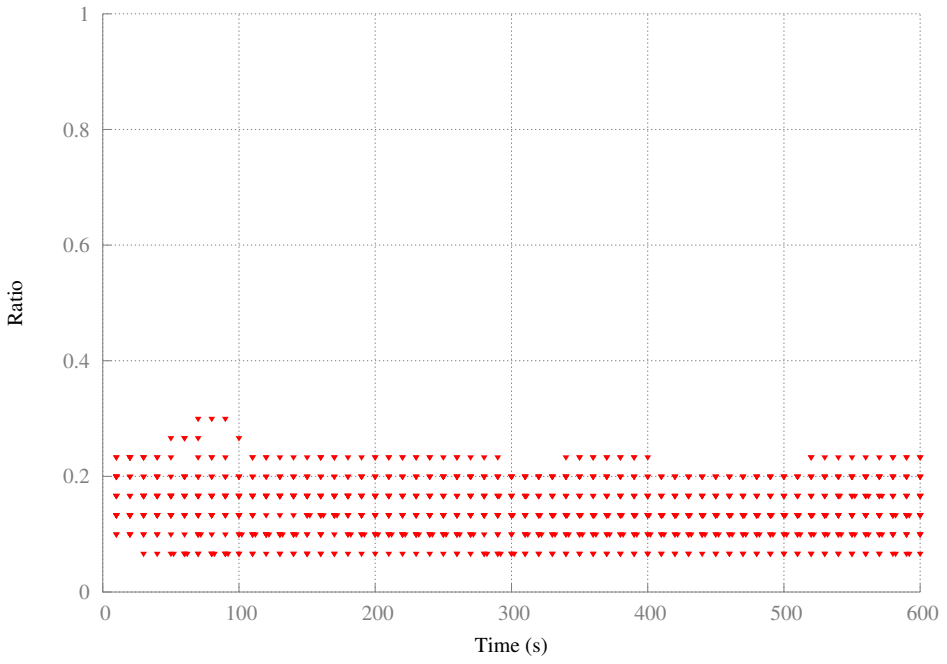
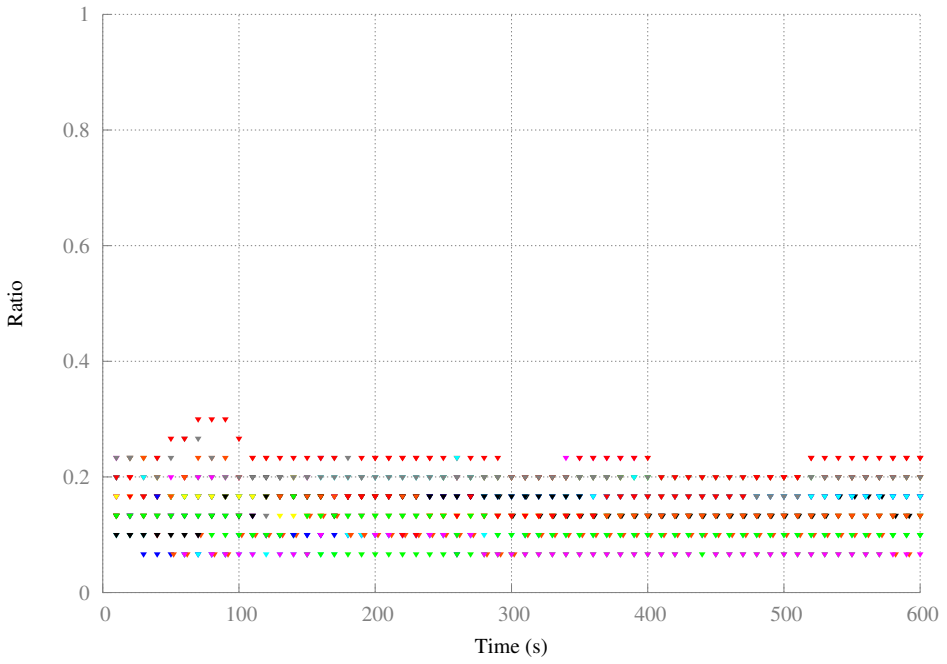


FIGURE 4.41: Instance Zeno9 : Diversité des vecteurs objectifs pour tous les runs (version colorée).



4.4.4 Conclusion

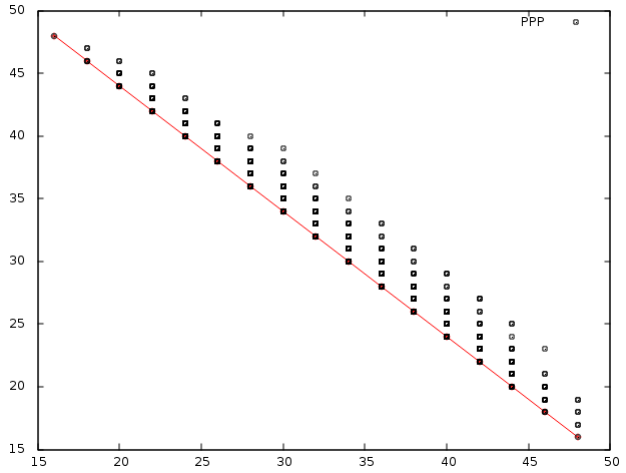
De la stratégie classique on peut tirer quelques conclusions et hypothèses qu'il faudra valider avec d'autres stratégies. L'instance 1 fait apparaître un échantillonnage difficile de la zone proche du front, malgré de très bons fronts obtenus pour chacun des runs. Cette zone est donc masquée par les surfaces d'atteintes mais peut être confirmée par les trajectoires de l'hypervolume qui montrent un saut brutal de l'hypervolume.

Cette zone difficile d'accès se retrouve également de façon très marquée avec l'instance 4 et l'instance 9. Dans une moindre mesure, l'hypervolume de l'instance 7 révèle également des sauts et la partie inférieure du front exacte semble difficile à atteindre.

Lorsque le front semble trop difficile à atteindre, on observe une baisse brutale de la diversité. Pour l'instance 4, cela se produit rapidemment compte tenu du fait que la difficulté à progresser intervient très tôt alors que pour l'instance 9 cela intervient plus tard.

Le cas de Zeno9 est intéressant puisque pour cette instance on dispose des points résultants de PPP et le nombre d'occurrence par points. Comme on peut l'observer sur la figure 4.42, l'ensemble de la zone qui semble difficile à atteindre regroupe le plus de PPP. À priori il y a donc bien des plans qui permettent d'accéder à cette zone. On peut par ailleurs noter qu'il y a plus de plans au milieu du centre que sur les extrémités qui sont pourtant les zones que DAE atteint le plus facilement.

FIGURE 4.42: Vecteurs objectifs des Plan Potentiellement Pareto-optimaux de l'instance Zeno9.



4.5 Stratégie adaptative

4.5.1 Ajout d'information

Le principe des stratégies adaptatives est d'utiliser un *feedback*, de l'information disponible au cours de la recherche afin d'orienter le choix des nouveaux paramètres. Nous pouvons réécrire notre distribution de probabilité :

$$P^j(x) = \begin{pmatrix} p^j(o_1|I^{j-1}(x)) \\ \vdots \\ p^j(o_i|I^{j-1}(x)) \\ \vdots \\ p^j(o_m|I^{j-1}(x)) \end{pmatrix}$$

Avec $I^{j-1}(x)$ une information qualitative sur x disponible à la génération $j-1$.

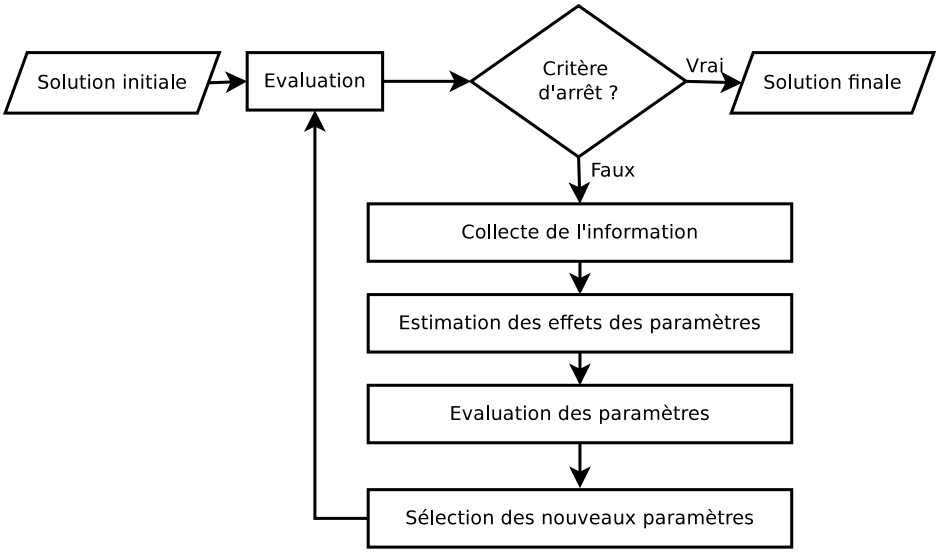
L'enjeu principal est de trouver une définition de I qui permette d'aboutir à des estimations de poids optimaux à la génération j.

4.5.2 Processus d'adaptation

Une stratégie d'adaptation consiste en les étapes suivantes :

— Collecte de l'information disponible.
— Estimation des effets des valeurs courantes des paramètres.
— Attribution de la qualité des valeurs courantes des paramètres.
— Sélection de nouveaux paramètres.

Le processus de recherche commence avec une population initiale, souvent générée aléatoirement ou avec l'aide d'information *à priori* le problème. Cette population évolue de manière itérative jusqu'à la rencontre d'un critère d'arrêt. À chaque génération, les individus de la population sont évalués. Dans le cadre de l'optimisation multi-objectifs, les critères sont multiples : ce sont des critères de qualité et de diversité qui sont utilisés, en plus du vecteur des objectifs. Dans le cadre plus particulier de DAE, la faisabilité d'un individu apporte égale une information (ou plus particulièrement sa non-faisabilité, en raison d'un sous-problème trop difficile à résoudre, d'une décomposition trop longue,...).



La collecte d'information consiste donc à trouver quelles sont les informations pertinentes afin de pouvoir attribuer une certaine qualité aux paramètres courants et d'inférer sur les nouveaux paramètres. L'estimation des effets est une étape délicate puisque les algorithmes étant stochastiques, pour un même effet supposé, des résultats différents peuvent être obtenus. De plus, c'est l'effet conjoint de l'ensemble des paramètres de l'algorithme qui influe sur la qualité générale des solutions et quantifier l'effet d'un paramètre seul semble d'autant plus délicat.

Enfin, la sélection de nouveaux paramètres se résume à un choix de compromis entre garder les meilleurs paramètres connus jusqu'à présent et explorer l'espace des configurations.

4.5.3 Stratégie de collecte d'information

Comme énoncé précédemment, il y a 2 principaux critères de la qualité d'un ensemble de solutions retournées par un algorithme multi-objectifs avec une approche Pareto : la qualité des solutions et la diversité. La qualité est définie par un indicateur utilisant notamment les valeurs du vecteur objectifs : indicateur d'hypervolume, indicateur ϵ,... La diversité est également importante pour obtenir une meilleure couverture du front.

À priori l'information n'a pas la même pertinence au fonction du niveau de granularité. Par exemple, si la distribution de probabilité est portée au niveau de la population, une diminution de l'hypervolume de la population entre deux générations semble pertinents. Cependant, au niveau de l'individu, si celui-ci se trouve dominé à la génération $j-1$ et

l'est toujours à la génération j, l'hypervolume est susceptible de ne pas avoir changer. Il est pourtant possible que l'individu est fortement été amélioré.

4.5.3.1 Niveau de la population

Au niveau de la population, nous disposons des indicateurs classiques d'hypervolume ou ϵ-indicateur. Par exemple, dans le cadre de l'hypervolume, l'information disponible sur la population x à l'instant j est défini comme l'information aux temps précédents et la différence entre l'hypervolume de x à l'instant j et $j-1$ d'après le point de référence r.

$$I(x)^j = I(x)^{j-1} \cap H_r^-(x^j, x^{j-1})$$

4.5.3.2 Niveau de l'individu

Au niveau de l'individu, l'information pouvant être utilisée est le vecteur objectif à l'itération j : $f^j(x)$

L'information disponible est donc la série temporelle des vecteurs objectifs :

$$I(x)^j = I(x)^{j-1} \cap f^j(x)$$

Nous disposons également de l'information sur le statut de la résolution de l'individu. Notons S^j le statut après résolution de l'individu. Cela peut servir à fortement pénaliser le choix de paramètres conduisant à un individu non-faisable.

L'information totale disponible est donc :

$$I(x)^j = I(x)^{j-1} \cap [f^j(x) \cap S^j(x)]$$

4.5.3.3 Niveau des gènes

Au niveau des gènes, l'information est plus restreinte. Il n'est pas nécessairement pertinent d'accéder au vecteur objectif de chaque sous-plan puisque le plan final résultera de la compression de ceux-ci. On pourra cependant utiliser la même information que pour l'individu, avec cependant pour difficulté d'estimer l'influence du gène au niveau

de l'individu.

Une information supplémentaire dont on dispose est le premier sous-problème qui n'a pas été résolu, ou la cause de non-résolution (notamment si la décomposition était trop longue ou le dernière état (immuable puisque le but du problème) n'a pas pu être atteint). Cette information est par ailleurs déjà traitée au niveau global lorsque l'on assigne le vecteur objectif. En effet, si aucun plan n'a été trouvé, le vecteur objectif associé à l'individu sera fortement pénalisé, mais cette pénalité dépend de la raison pour laquelle aucun plan n'a été trouvé.

Ainsi, si un sous problème intermédiaire n'a pas été résolu, le vecteur objectif sera fonction du nombre de sous-but résolus ainsi que de la distance de Hamming du but qui n'a pas été atteint.

4.5.4 Stratégie d'estimation des effets des paramètres

La stratégie d'estimation des effets des paramètres permet de faire le lien entre l'espace des solutions et l'espace des paramètres. Les stratégies les plus courantes consistent simplement à utiliser un indicateur comme l'amélioration des individus par rapport à une solution de référence (solution précédente, meilleure solution connue à ce jour ou un quantile spécifique). La thèse [40] propose une stratégie d'estimation des effets plus évoluée, basée sur l'inférence bayesienne (Bayesian Effect Assessment), dont elle valide statistiquement les performances. C'est cette stratégie que nous retenons car elle n'implique pas nécessairement un surcoût de calculs.

Notons λ un indicateur dépendant de l'information I disponible. On définit la fonction :

$$e^j(x, o_i) = \begin{cases} e^+ & \text{si } \lambda^j(x) \succ \lambda^{j-1}(x) \\ e^- & sinon \end{cases}$$

En d'autres termes, si l'utilisation de l'objectifs o_i à l'itération $j-1$ conduit à une amélioration à l'itération j, d'après l'indicateur λ on marque positivement ce choix. La méthode s'adapte à n'importe quel indicateur d'amélioration et vient donc s'incrire *au dessus* des stratégies plus simples.

Cependant, les variations de la qualité d'un individu ne proviennent d'un ensemble de paramètres et il convient d'essayer d'estimer la relation de cause à effet entre un paramètre et l'évolution d'une solution.

Pour cela nous utilisons la probabilité d'amélioration de l'individu à l'instant j sachant l'objectif à utiliser. Notons n_i le nombre de fois où l'objectif o_i a été utilisé et n_i^+ le nombre de fois où l'on a marqué positivement l'objectif o_i.

$$p^j(e^+|o_i) = \frac{n_i^+}{n_i}$$

On conclut à l'impact d'un objectif particulier en fixant un seuil $\alpha \in [0, 1]$. Si $p^j(e^+|o_i) > \alpha$ alors l'objectif o_i à un impact positif.

Notons que l'on initialise avec $p^j(e^+|n_i^j = 0) = 1$.

4.5.5 Stratégie d'évaluation de la qualité des paramètres

Basée sur l'information disponible et l'estimation de l'effet de la valeur du paramètre courant, la stratégie d'évaluation vise à faire le pont entre la configuration courante des paramètres et le choix de la valeur des paramètres suivants. L'état de l'art utilise une fenêtre temporelle de h itérations pour estimer la qualité des paramètres, cependant, l'expérience montre qu'au niveau des individus ou des gènes dans DAE, le nombre moyen d'itération (c'est à dire d'évaluation) durant la durée de vie de l'individu ou du gène est relativement faible et donc on n'utilisera pas de fenêtre temporelle (ou plutôt une fenêtre de taille 1).

Plusieurs stratégies peuvent être envisagées, notamment basée sur l'effet moyen (Average Quality Attribution) ou la valeur extrême (Extreme Quality Attribution) au cours des h dernières itérations. D'autres méthodes plus évoluées avec prédiction (Predictive Quality Attribution) utilisant une projection temporelle via une régression linéaire ou des processus ARMA plus ou moins complexes (ARIMA par exemple).

Average Quality Attribution Considérant un indicateur λ, avec cette stratégie, la qualité q de l'objectif o_i à l'itération j est défini par la valeur de moyenne de l'indicateur par rapport au nombre de fois où l'objectif a été utilisé durant les h précédentes

générations :

$$q^j(o_i) = \frac{1}{n_i} \sum_{k=j-1-h}^{j-1} \lambda^k(x)$$

Extreme Quality Attribution Considérant un indicateur λ, avec cette stratégie, la qualité q de l'objectif o_i à l'itération j est défini par la meilleure valeur de l'indicateur obtenue durant les h précédents générations :

$$q^j(o_i) = \max_{j-1-h \leq k \leq j-1} \left(\frac{\lambda^k(x)}{\lambda(x^*)}\right)$$

Avec $\lambda(x^*)$ la meilleure valeur rencontrée jusque là.

4.5.6 Stratégie de sélection des paramètres

La dernière étape d'un stratégie adaptative vise à déterminer la nouvelle valeur des paramètres à utiliser lors de la prochaine itération.

4.5.6.1 Probability Matching

Il s'agit d'une méthode sous-optimale d'apprentissage par renforcement pour projeter la performance d'une valeur du paramètre en fonction de ses performances passées. La valeur projetée est basée sur les fréquences d'amélioration selon la valeur du paramètre. Elle consiste à modifier la distribution de probabilité P telle que la probabilité d'obtenir l'objectif à optimiser est proportionnelle à la qualité estimée de cet objectif.

$$p^j(o_i) = \frac{q^j(o_i)p^j(e^+|o_i)}{\sum_k q^j(o_k)p^j(e^+|o_k)}$$

La probabilité de tirer l'objectif o_i à la génération i est proportionnelle à sa qualité pondérée par l'estimation de son effet. Notons que la mise à jour n'est faite que toutes les h générations pour permettre d'estimer la qualité convenablement. Cependant, avec cette mise à jour, il est possible que la probabilité de choisir un objectif devienne 0, hors le processus de recherche n'étant pas stationnaire, pour assurer que cet objectif puisse rester disponible, Thierens propose un probabilité plancher $p_{\min}$. Le calcul deviens donc :

$$p^j(o_i) = p_{\min} + (1 - l * p_{\min}) \frac{q^j(o_i)p^j(e^+|o_i)}{\sum_k q^j(o_k)p^j(e^+|o_k)}$$

4.5.6.2 Adaptive Pursuit

La poursuite adaptative a été conçu dans le but d'améliorer les performences de la méthode Probability Matching. L'idée est de récompenser la valeur qui a donné le meilleur résultat après h itérations en augmentant sa probabilité d'être choisi dans le futur et en diminuant d'autant les autres valeurs.

On pose $o^* = \underset{1 \leq k \leq l}{argmax}(q^j(o_k)p^j(e^+|o_k))$.

$$p^j(o_i) = \begin{cases} (1-\beta)p^{j-1}(o_i) + \beta \text{ si } o_i = o^* \\ (1-\beta)p^{j-1}(o_i) \ sinon \end{cases}$$

Avec β une constante correspondant à la proportion qu'à la meilleure valeur à l'emporter sur les autres. Encore une fois, pour éviter une probabilité 0 de certains objectifs, on fixe une probabilité minimale $p_{\min}$.

$$p^j(o_i) = \begin{cases} p^{j-1}(o_i) + \beta(p_{\max} - p^{j-1}(o_i) \text{ si } o_i = o^* \\ p^{j-1}(o_i) + \beta(p_{\min} - p^{j-1}(o_i) \ sinon \end{cases}$$

En considérant la $p_{\max} = 1-(l-1)p_{\min}$. Pour s'assurer que la somme des probabilités est égale à 1, la valeur de $p_{\min}$ doit être inférieur à $\frac{1}{l}$. D'après [41] une valeur recommandée est $\frac{1}{2l}$ ce qui permet de s'assurer que le meilleur objectif est tiré au moins une fois sur deux.

4.5.7 Détecter un changement : test de Page-Hinkley

Les opérateurs de diversité peuvent apporter de nouveaux éléments au sein des individus qui vont *complètement* modifier la valeur optimale à utiliser pour le paramètre considéré. L'enjeu est de détecter un tel changement et de réinitialiser le processus d'optimisation.

Pour cela, on peut utiliser le test cumulatif de Page-Hinkley qui vise à détecter les sauts de moyenne. Ce test est connu pour sa robustesse et son faible coût en temps de calcul.

$$M^0 = 0, \quad M^j = \sum_{i=0}^{j} (q^i(P) - \mu + \frac{\delta}{2})$$

$$m^j = \underset{0 \leq i \leq j}{max} M^i$$

Avec δ un facteur de tolérance ou d'écart à priori et $q^i(P)$ la qualité de la distribution de probabilité P à l'itération i. D'après [42], la valeur de δ devrait être fixée à 0.15. On maintient à jour une estimation μ de la moyenne des améliorations dû à la stratégie courante (c'est à dire à la distribution de probabilité courante).

On détecte une rupture dans la qualité de la distribution courante lorsque $m^j - M^j > \alpha$ avec α un seuil fixé à l'avance par l'utilisateur. Au final, on réinitialise la distribution de probabilité P, ainsi que l'estimateur μ, M et l'efficacité de chaque objectif.

On propose de mesurer la qualité de la distribution comme la somme du produit efficacité / qualité de chaque objectif :

$$q^j(P) = \sum_{k}^{j} q^j(o_k) p^j(e^+|o_k)$$

4.5.8 En résumé et facteurs

On initialise la distribution de probabilité avec $\frac{1}{l}$ ou éventuellement de manière aléatoire ou avec des poids fournis par l'utilisateur si l'on dispose d'une connaissance à priori.

À chaque itération, la mise à jour s'effectue de la manière suivante :

1. Mise à jour de l'efficacité $e(o_i)$.
2. Calculer $q(o_i)$ et mettre alors à jour P.
3. Mettre à jour μ, M et m.
4. Si un saut est détecté, réinitialiser P, μ, M, m.

4.5.9 Les paramètres

TABLE 4.2: Paramètres trouvés par ParamILS pour la stratégie adaptative.

Paramètres	Instance 1	Instance 4	Instance 7	Instance 9
bmax-fixed	100000	100000	100000	100000
popSize	30	20	200	30
proba-change	0.8	0.2	0.2	0.8
proba-cross	0.2	0.1	1.0	0.2
proba-del-atom	0.5	0.1	1.0	0.5
proba-mut	0.8	1	0.8	0.8
radius	3	10	7	3
strat-level	Goal	Indi	Indi	Goal
w-addatom	1	3	7	1
w-addgoal	1	3	3	1
w-delatom	3	0	7	3
w-delgoal	3	10	7	1

Il est intéressant de noter que pour l'instance 1, pratiquement aucun changemnet au niveau des paramètres n'intervient entre la stratégie statique et la stratégie adaptative, si ce n'est la valeur de b_{max} divisée par 10, la taille de la population également divisée par 10 et un poids au niveau des mutations.

A contrario, on observe des changements importants pour les autres instances. Dans un premier temps, la taille de la population est systématiquement divisée par 10, sauf pour l'instance 7. C'est pourtant l'instance dont les paramètres changent le plus d'une stratégie à l'autre puisque si l'on avait une probabilité `proba-change` de 0.0 pour la stratégie statique, elle est de 1 pour la stratégie adaptative.

Globalement, on peut dire que les paramètres de diversité sont plus élevés.

4.5.10 Résultats empiriques

4.5.10.1 Spécificités du protocole

Au vu du nombre de facteurs à tester, il était assez peu réaliste de réaliser une optimisation de paramètre via ParamILS pour combinaison de facteurs, tester l'influence, changement de combinaison, relancer ParamILS, et ainsi de suite. De fait, une seule configuration a été testé, en utilisant ParamILS. Les résultats n'étant pas à la hauteur de l'espérance, nous avons décidé de tester d'autres stratégies permettant d'obtenir de meilleurs résultats.

La configuration testée est utilise l'indicateur λ_{Δ^+}, sans estimation, Average Quality Attribution comme stratégie d'évaluation de la qualité des paramètres, Adaptive Pursuit comme stratégie de sélection des nouveaux paramètres et une détection des sauts avec Page-Hinkley.

4.5.10.2 Instance 1

Les fronts cumulés de la figure 4.43 sont plus éloignés que ceux de la stratégie statique. On a cependant une forte présence de points proches du front exact sur les extrémités et assez peu au centre. Ce constat est d'autant plus flagrant sur la figure 4.44 décrivant la population cumulée. Très clairement il apparait un large biais en faveur des valeurs faibles du coût. On retrouve cependant les même motifs sur le flan, indiquant probablement qu'il s'agit plus d'une structure liée à l'instance qu'à l'effet de la stratégie choisie.

Les surfaces d'atteinte sont surprenantes vis à vis de ces premiers constats. La figure 4.46 indique une très grande probabilité d'atteindre l'extrémité formée des points de faible *makespan* alors qu'il s'agit de la zone la moins échantillonnée de l'espace objectif. La différence entre le meilleur *run* et le reste des *runs* semble s'accroître plus l'on considère les points du fronts possédant un *makespan* de plus en plus élevé.

Les trajectoires de l'hypervolume semble également posséder la même structure qu'avec la stratégie statique, c'est à dire une diminution très rapide puis un saut. La diversité des vecteurs objectifs décrite par les figures 4.48 et 4.49 est par contre totalement différente de la stratégie statique. En effet, si l'on était relativement uniforme autour de 0.6, on est

FIGURE 4.43: Instance 1 : Fronts de Pareto cumulés pour tous les runs, à la fin de chaque run.

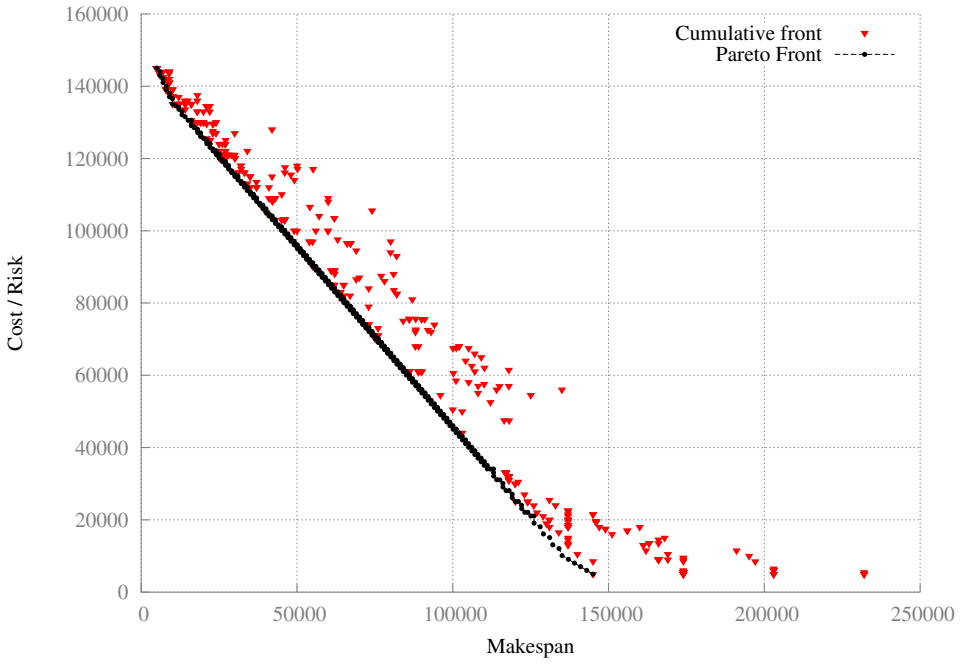


FIGURE 4.44: Instance 1 : Population cumulées pour tous les runs et à tout temps.

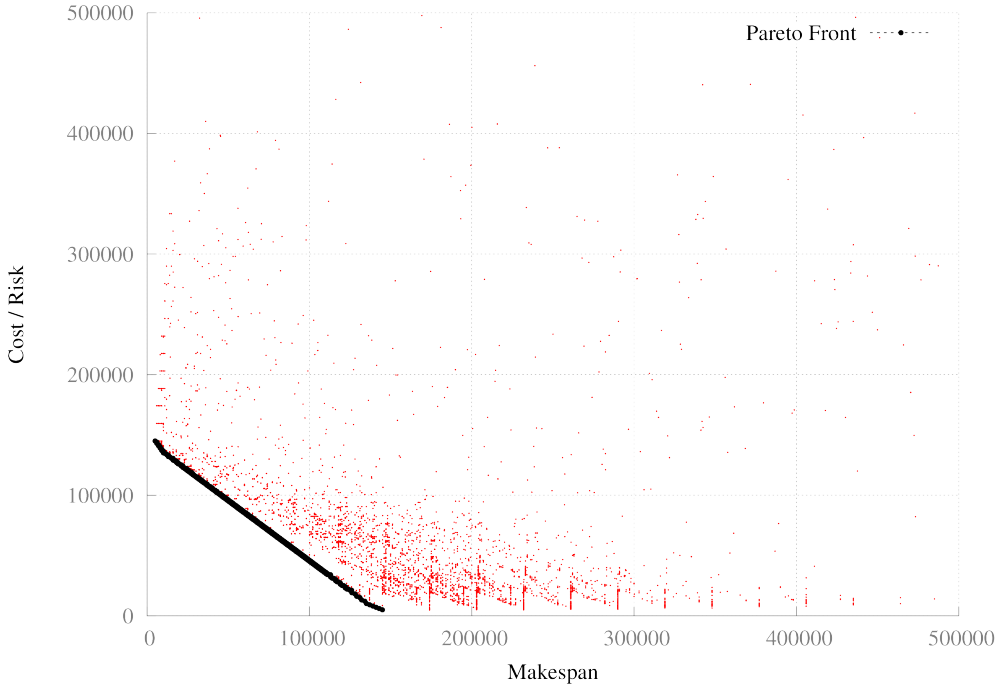
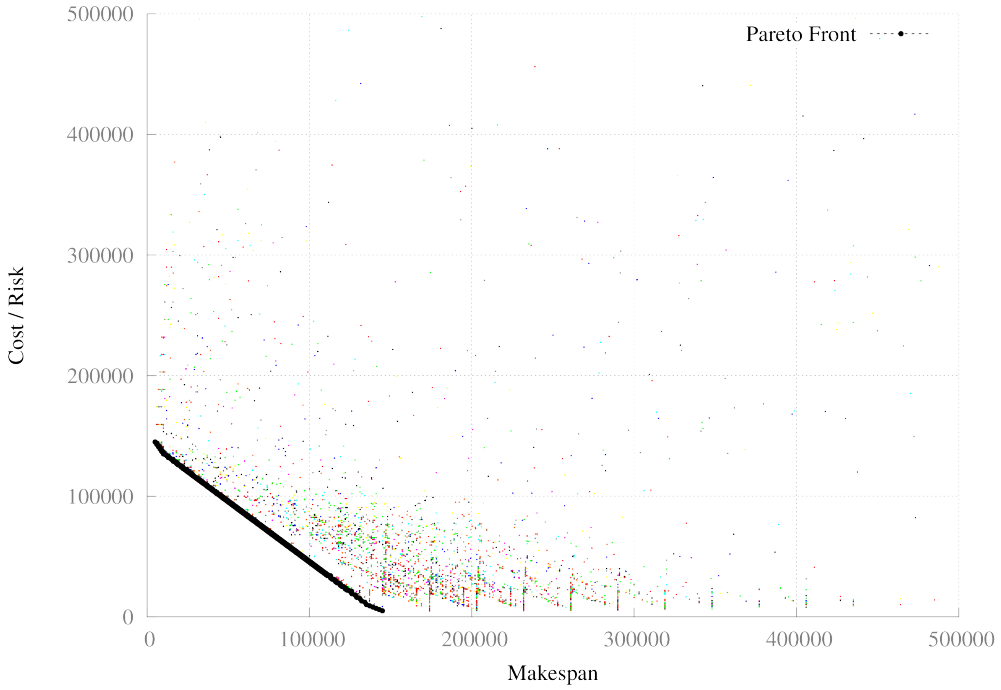


FIGURE 4.45: Instance 1 : Population cumulées pour tous les runs et à tout temps (version colorée).

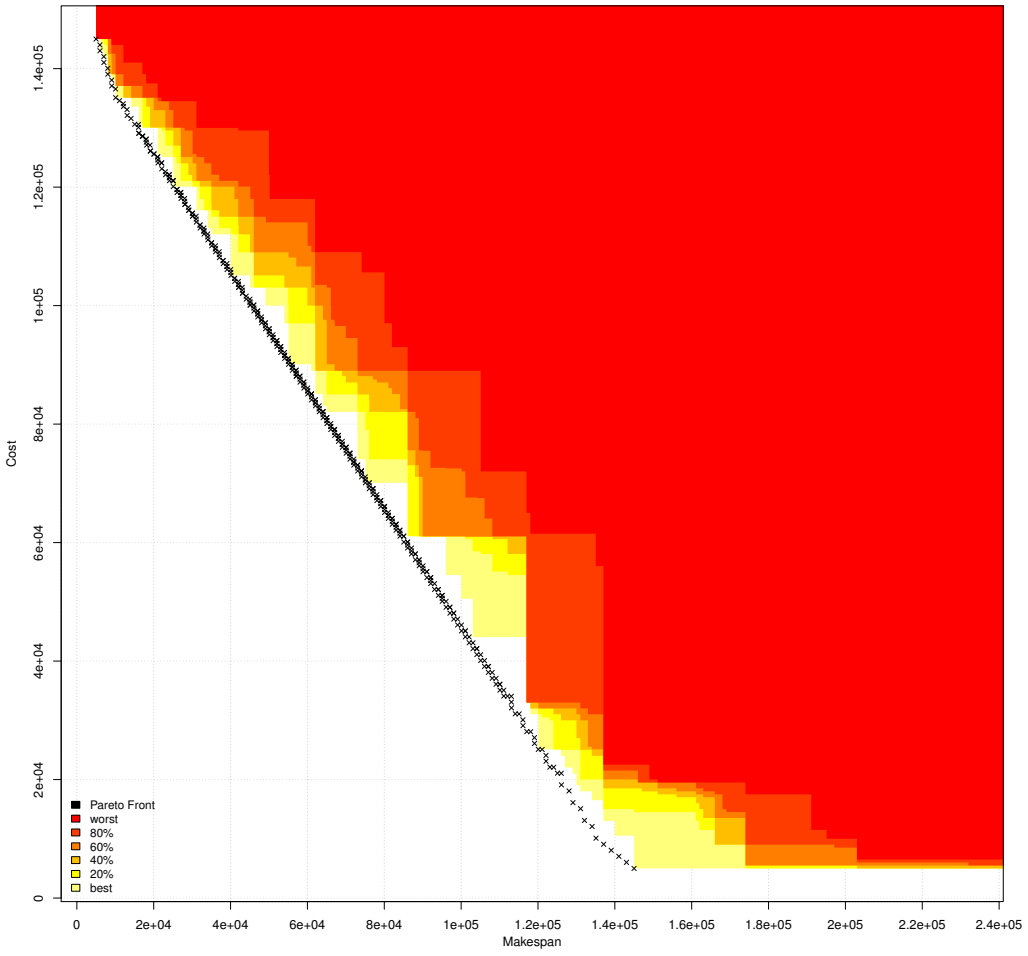


ici beaucoup plus bas, sans grande fluctuation au sein d'une trajectoire. On remarque par contre une très forte diversité à l'instant initial.

Comparaison avec la stratégie statique Il n'est pas très difficile de voir à l'aide de la figure 4.50 qui montre les surfaces d'atteinte de la stratégie adaptative aux côtés de la stratégie statique, que la stratégie statique domine. Ce verdict est confirmé à la fois par les figures 4.51 et 4.52 mais également par les tests statistiques sur les fonctions d'atteinte du premier et second ordre.

Le figure 4.54 superpose les trajectoires de l'hypervolume pour les deux stratégies, tandis que la figure 4.54 superpose l'évolution de la diversité des vecteurs objectifs. En bleu apparaît la stratégie statique et en rouge la stratégie adaptative. Il semblerait que l'hypervolume descende un peu plus rapidemment pour la stratégie adaptative tandis que la stratégie classique propose des sauts d'amplitude plus importante. Ce qu'il est intéressant de noter c'est qu'une fois le saut effectué, l'évolution de l'hypervolume est quasi-stationnaire, peu importe la stratégie. Le saut apparaît également d'une génération à l'autre. Si l'on étudie le front courant entre la génération précédant le saut et la

FIGURE 4.46: Instance 1 : Surfaces d'atteinte pour tous les runs.



génération juste après le saut, on obtient en réalité qu'il est provoqué par plusieurs individus, à la fois l'amélioration de certains individus et à la fois par l'ajout de nouveaux individus.

Par exemple, pour le premier run avec la stratégie classique, nous avons 45 points appartenant au front à la génération i et 47 à la génération $i+1$. Un point a été amélioré ($(6000, 174000) \rightarrow (6000, 145000)$), un point retiré ($7000, 145000$, dominé par l'amélioration précédemment citée), et 3 nouveaux points ($(143020, 7520)$, $(139000, 11510)$ et $(21020, 128510)$). Ce schéma se reproduit quelque soit le *run*.

FIGURE 4.47: Instance 1 : Trajectoires de l'hypervolume pour tous les runs.

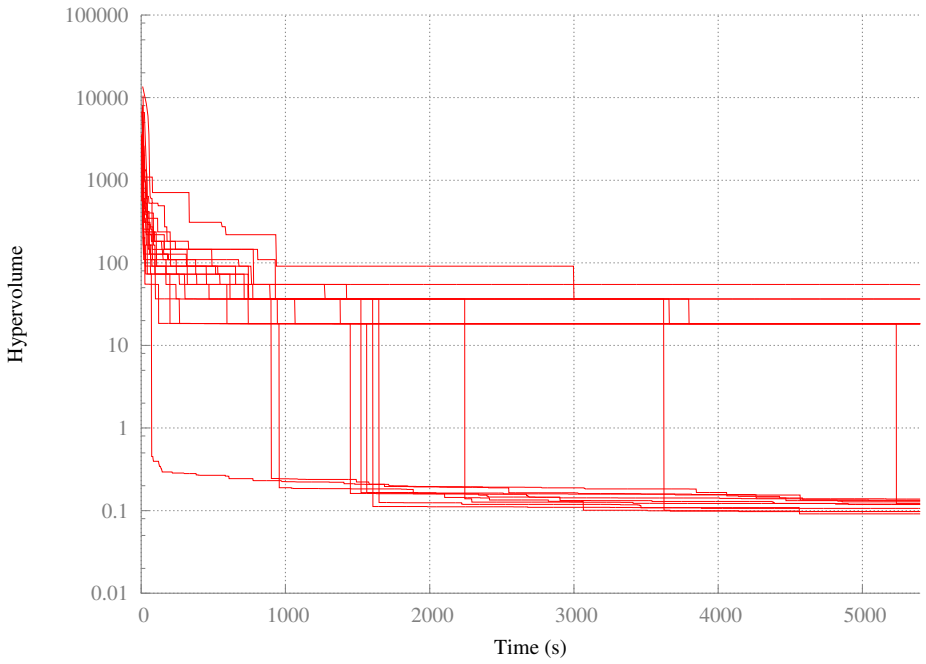


FIGURE 4.48: Instance 1 : Diversité des vecteurs objectifs pour tous les runs.

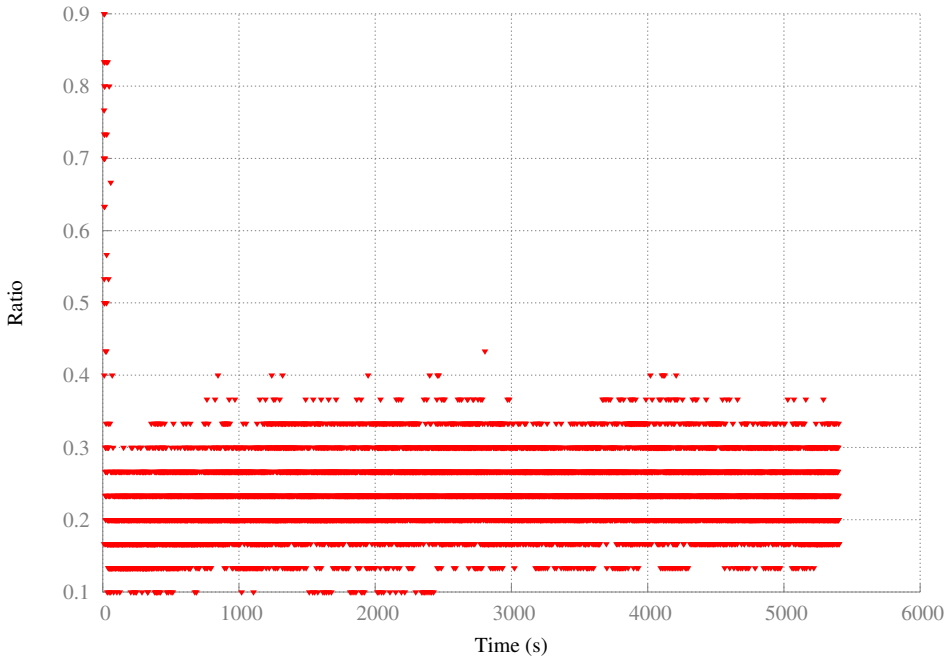


FIGURE 4.49: Instance 1 : Diversité des vecteurs objectifs pour tous les runs (version colorée).

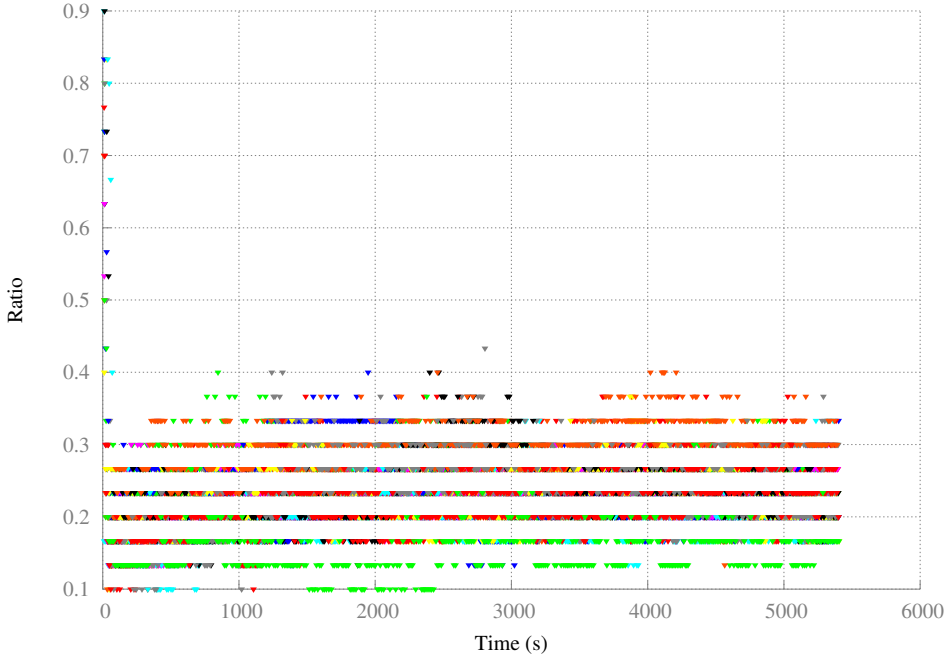


FIGURE 4.50: Instance 1 : Comparatif des surfaces d'atteinte.

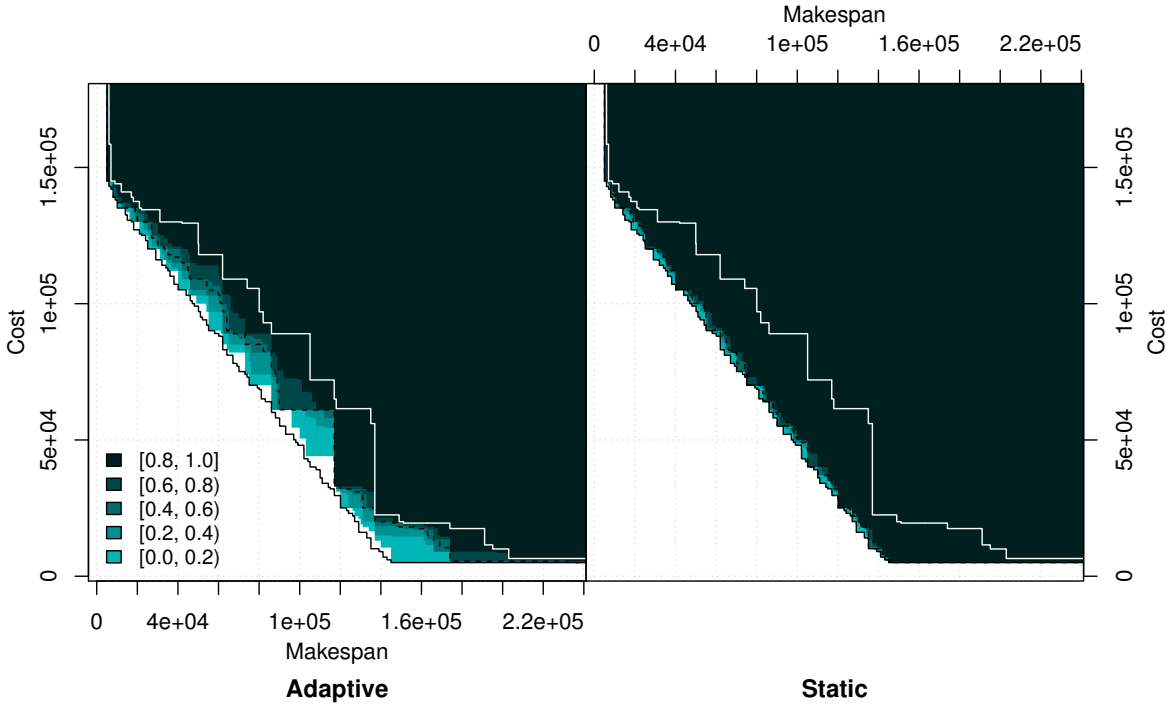


FIGURE 4.51: Instance 1 : Adaptative comparée à Statique.

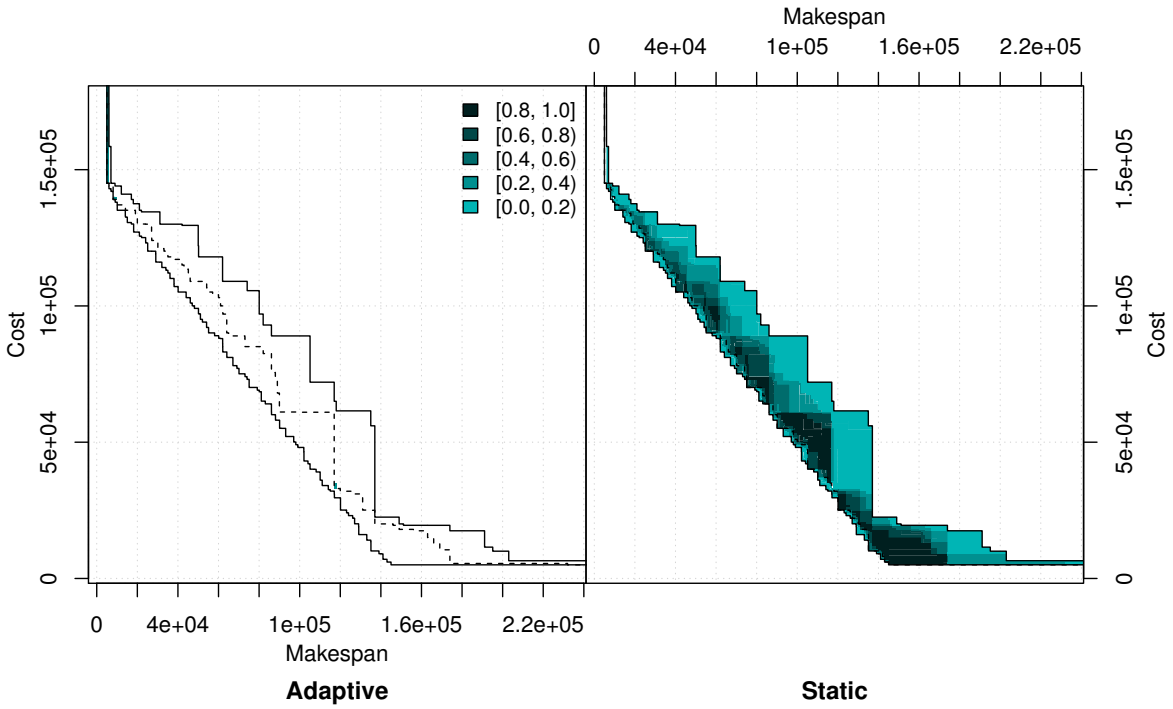


FIGURE 4.52: Instance 1 : Statique comparée à Adaptative.

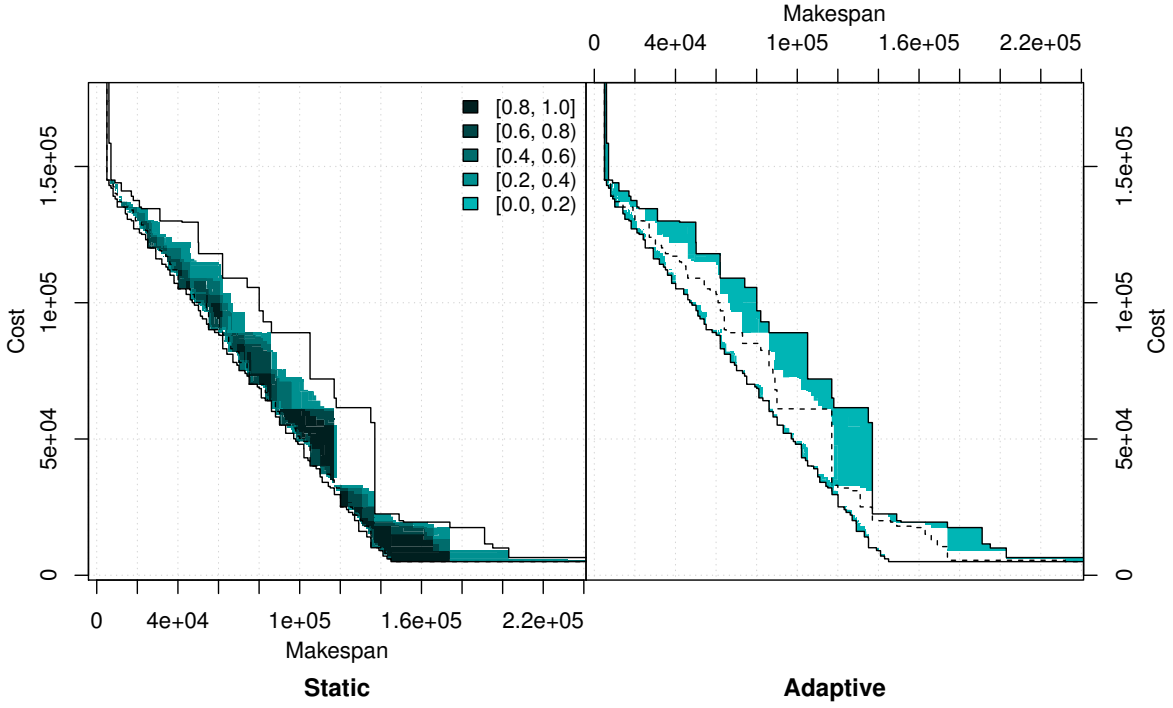


FIGURE 4.53: Instance 1 : Comparaison de l'hypervolume.

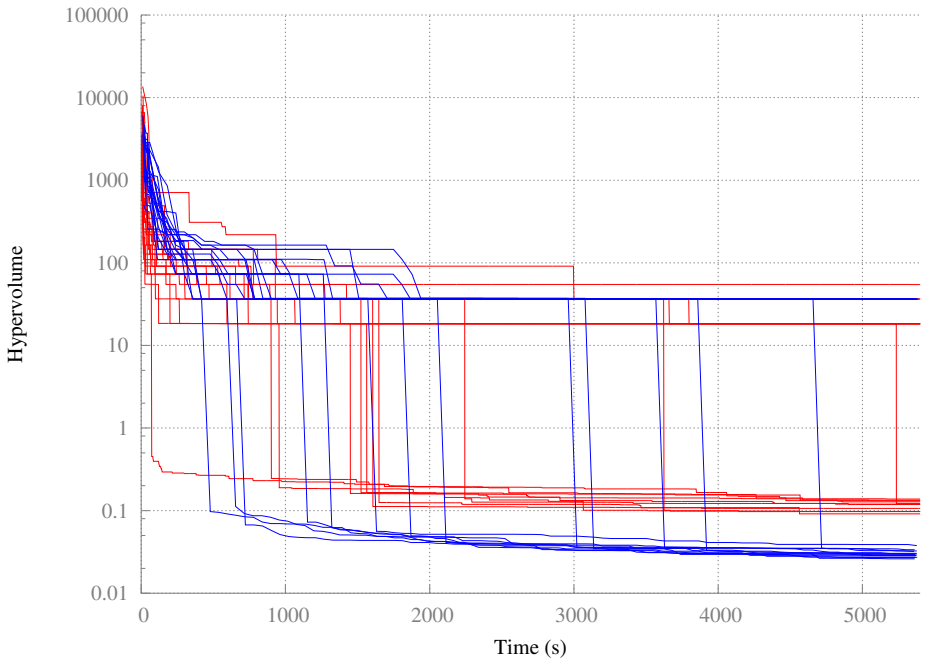
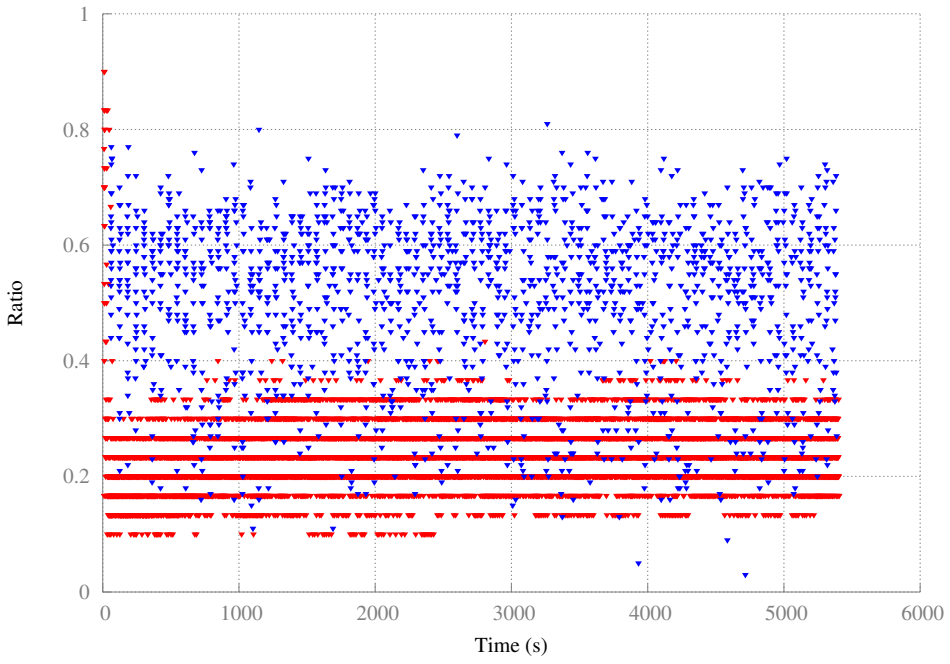


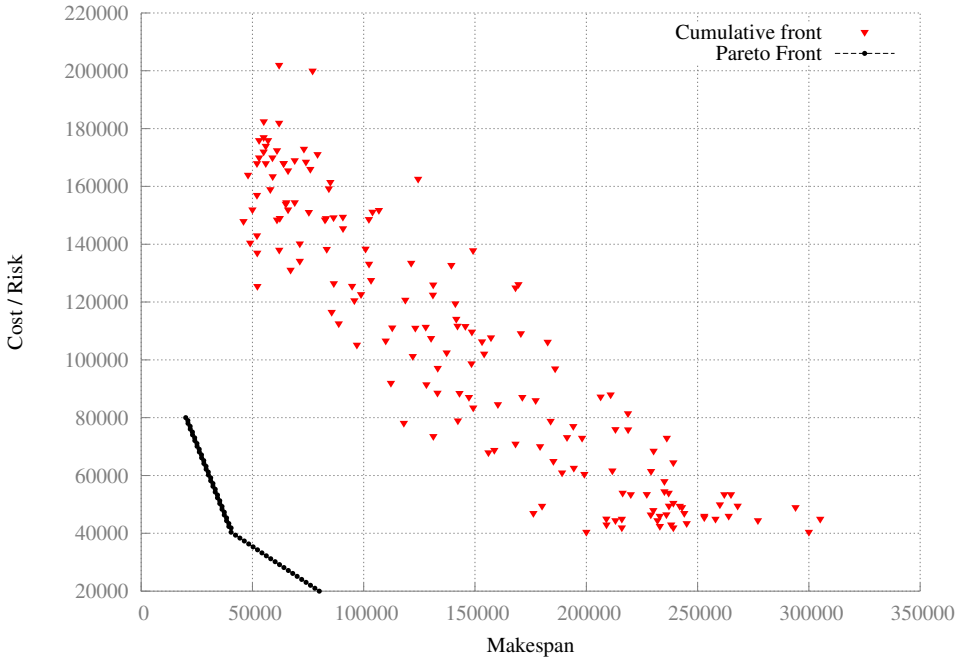
FIGURE 4.54: Instance 1 : Comparaison de la diversité des vecteurs objectifs.



4.5.10.3 Instance 4

Si les fronts cumulés n'indiquent rien de notable par rapport à ceux de la stratégie statique, on remarque cependant sur la figure 4.56 de la population cumulée, la densité des vecteurs objectifs visités est plus faible qu'avec la stratégie classique, comme on a pu l'observer avec l'instance 1.

FIGURE 4.55: Instance 4 : Fronts de Pareto cumulés pour tous les runs, à la fin de chaque run.



Les résultats des surfaces d'atteintes sont clairement décevants. Il apparait une très grande disparité entre le meilleur et le moins bon des *runs*, d'autant plus par rapport aux résultats obtenus pour la stratégie classique. L'hypervolume quant à elle, semble présenter une variance très réduite entre les différentes trajectoires, pour n'importe quel moment. Il semblerait que ce soit un effet d'échelle au vu des surfaces d'atteinte.

Enfin, la diversité est extrêmement basse et ne présente pas cette structure décroissante où l'on pouvait distinguer les trajectoires, comme avec la stratégie statique.

Comparaison avec la stratégie statique Là encore, les résultats sont nettement en deçà de la stratégie classique, malgré une meilleure probabilité d'atteindre les zones de faibles *makespan* pour la stratégie adaptative comme en atteste la figure 4.63. À contrario, les trajectoires de l'hypervolume semble être similaires. Enfin, la structure de

FIGURE 4.56: Instance 4 : Population cumulées pour tous les runs et à tout temps.

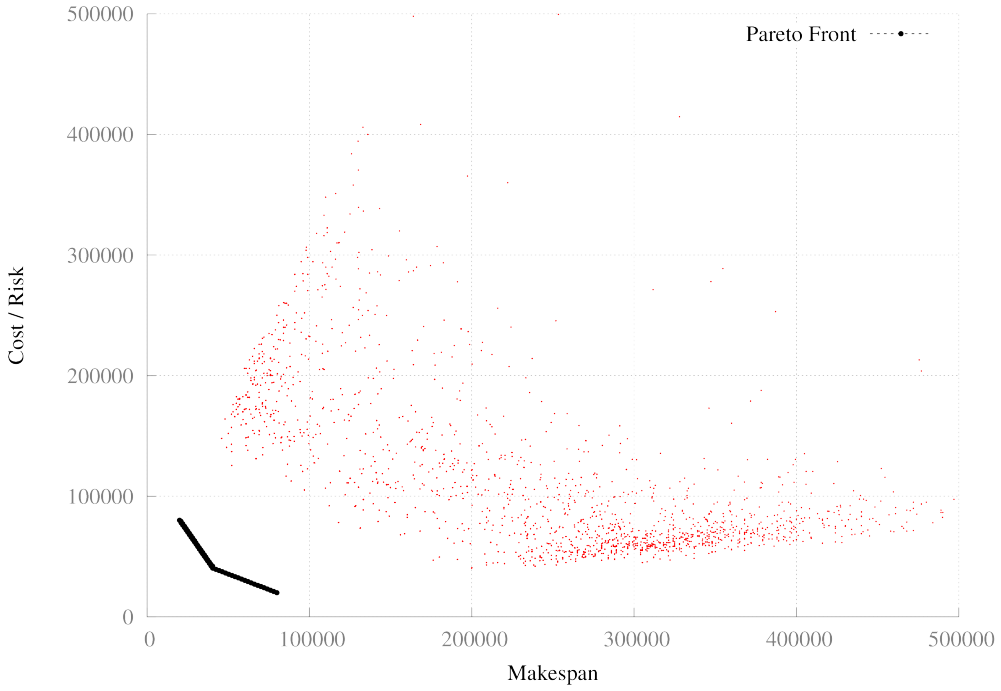


FIGURE 4.57: Instance 4 : Population cumulées pour tous les runs et à tout temps (version colorée).

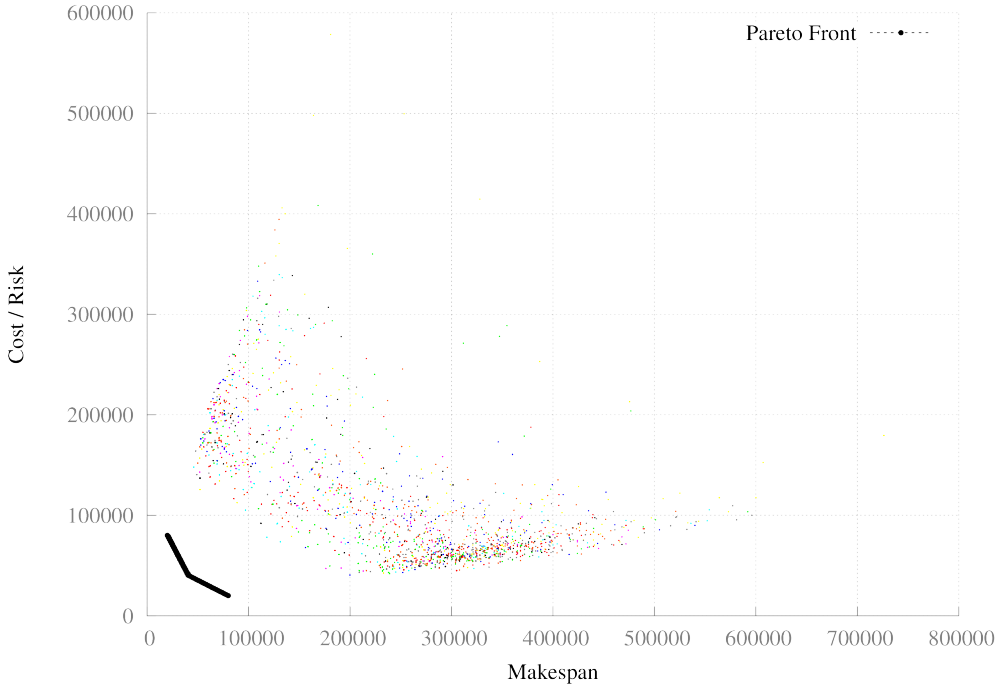
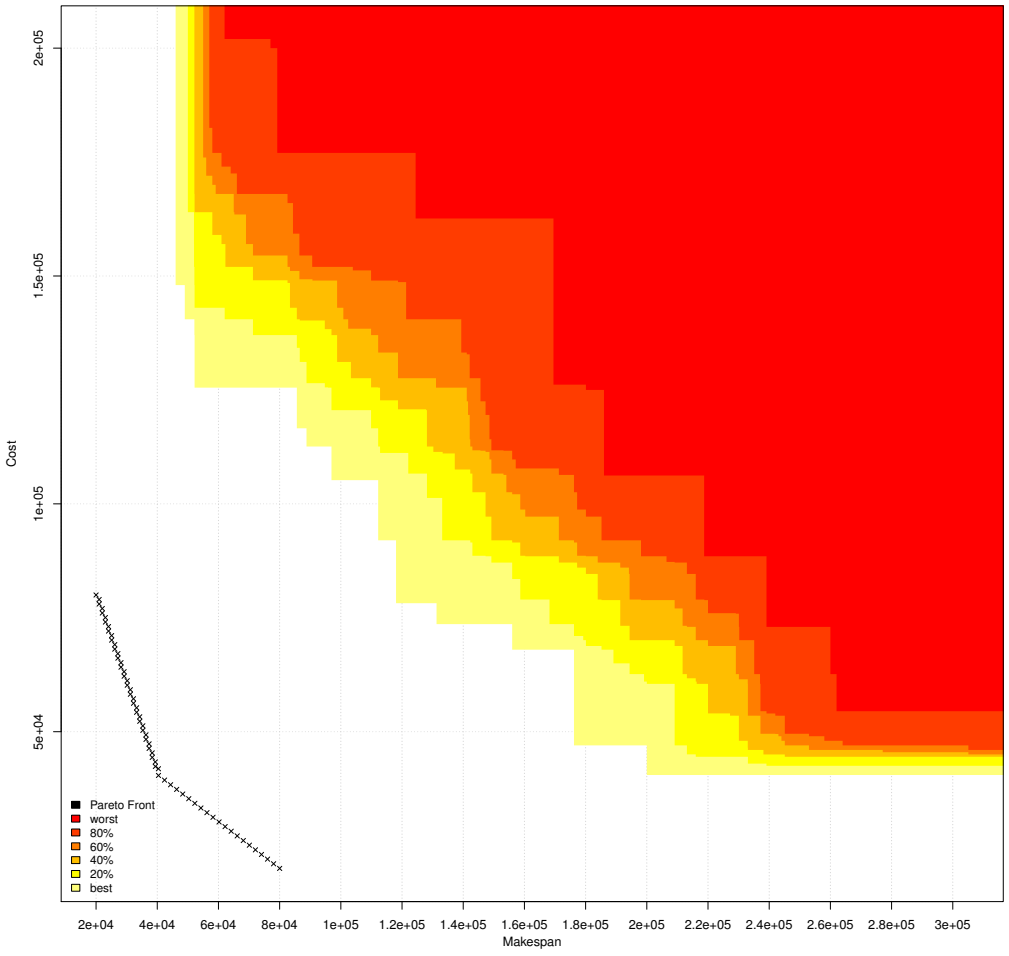


FIGURE 4.58: Instance 4 : Surfaces d'atteinte pour tous les runs.



la diversité des vecteurs objectifs est très différentes puisqu'avec la stratégie adaptative, la diversité est bloquée entre 0.1 et 0.3 globalement, alors que pour la stratégie statique, on commence très haut avant de très rapidemment se retrouver sous la diversité plancher de la stratégie adaptative.

Évidemment, les tests statistiques sur les fonctions d'atteinte rejette l'hypothèse nulle qui consiste à dire que la distribution des surfaces atteintes est similaire d'une stratégie à l'autre, résultat qui était visible sur la quasi-totalité des graphiques.

FIGURE 4.59: Instance 4 : Trajectoires de l'hypervolume pour tous les runs.

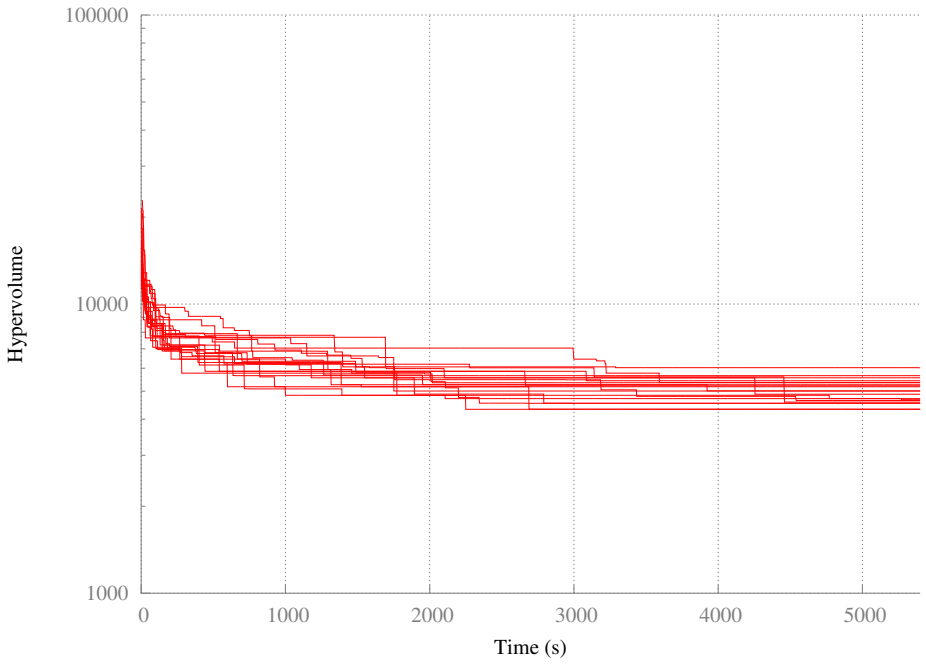


FIGURE 4.60: Instance 4 : Diversité des vecteurs objectifs pour tous les runs.

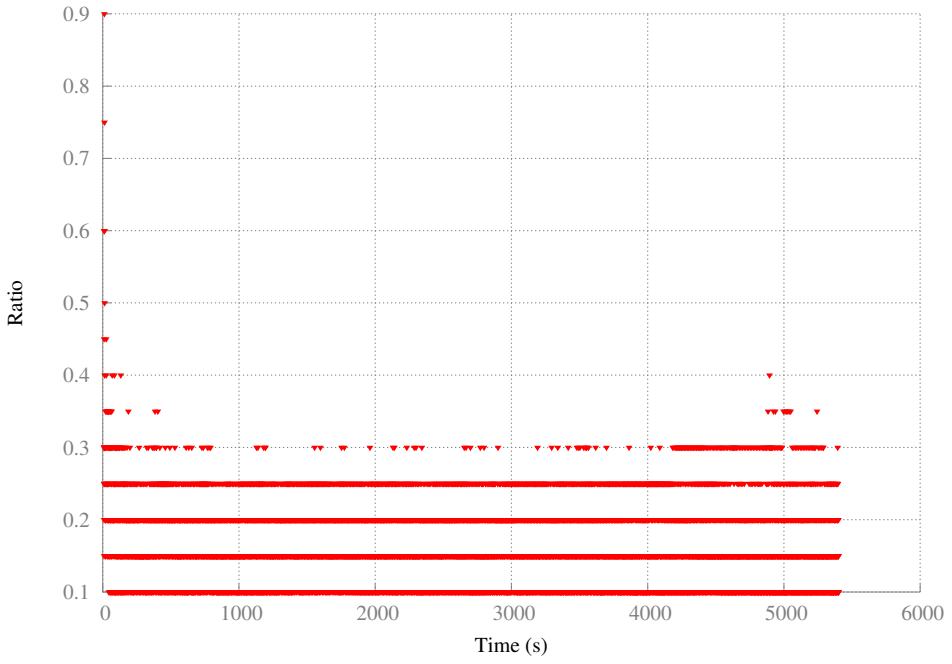


FIGURE 4.61: Instance 4 : Diversité des vecteurs objectifs pour tous les runs (version colorée).

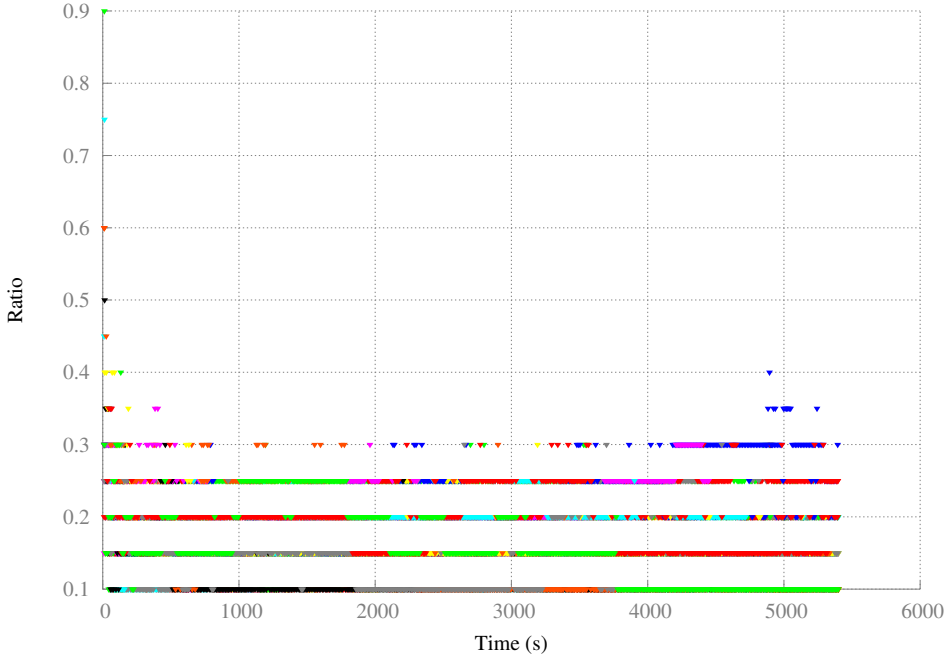


FIGURE 4.62: Instance 4 : Comparatif des surfaces d'atteinte.

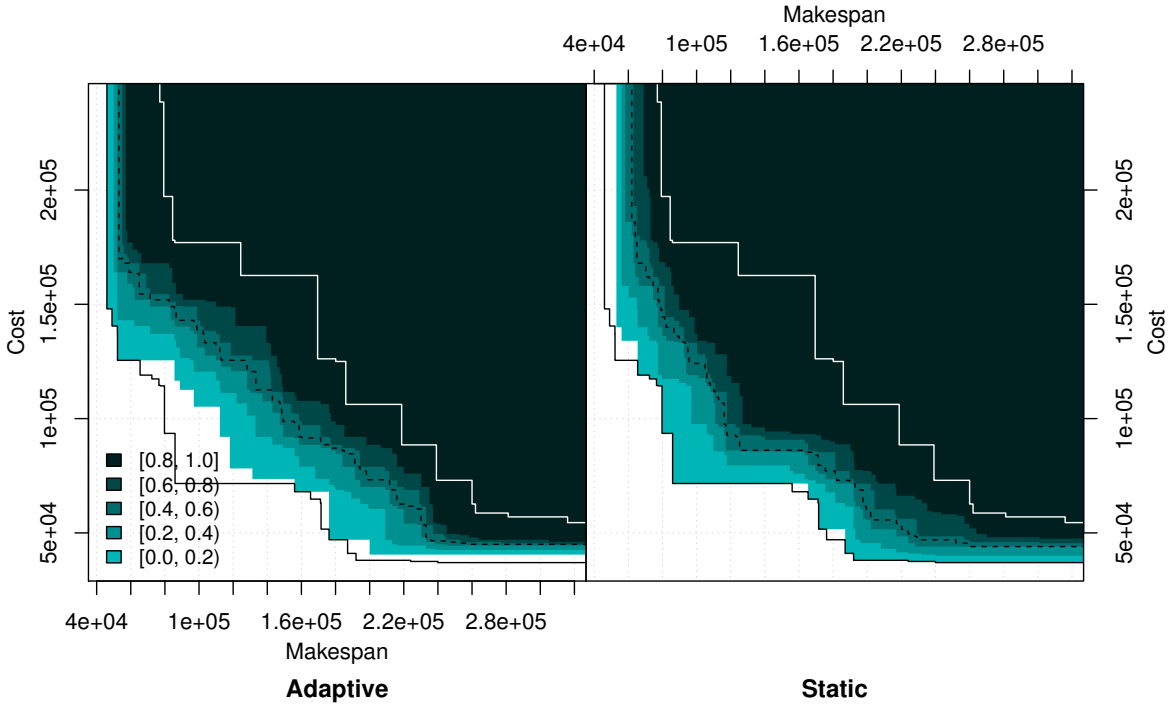


FIGURE 4.63: Instance 4 : Adaptative comparée à Statique.

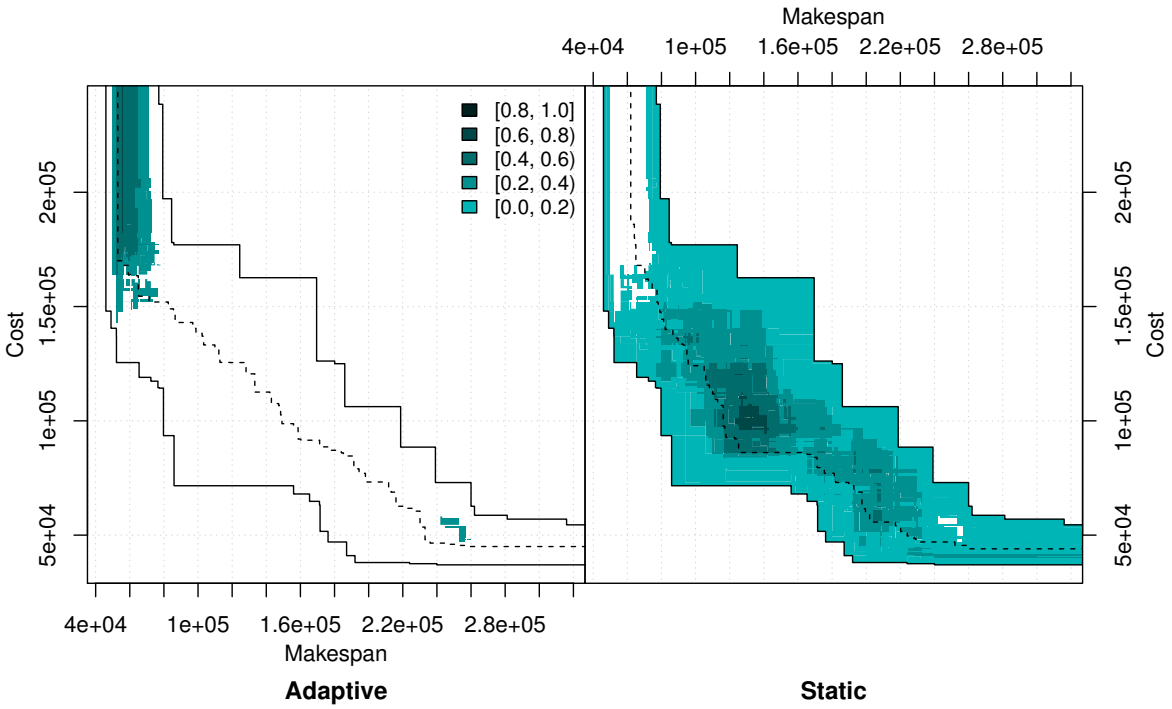


FIGURE 4.64: Instance 4 : Statique comparée à Adaptative.

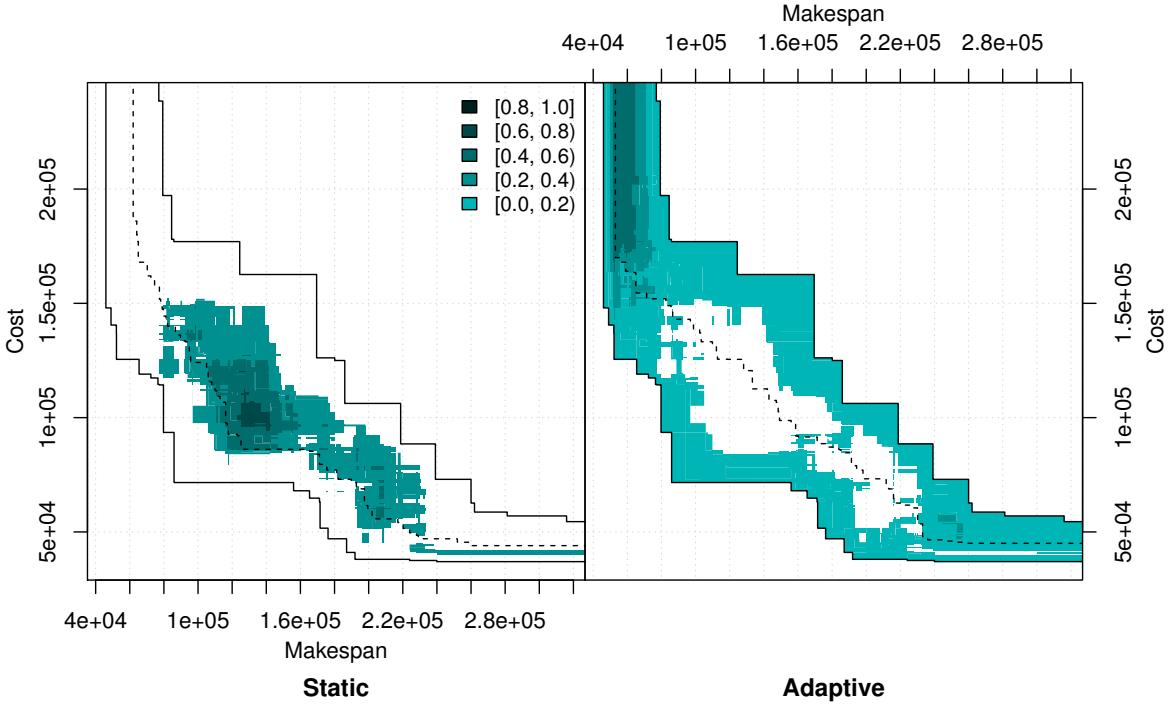


FIGURE 4.65: Instance 4 : Comparaison de l'hypervolume.

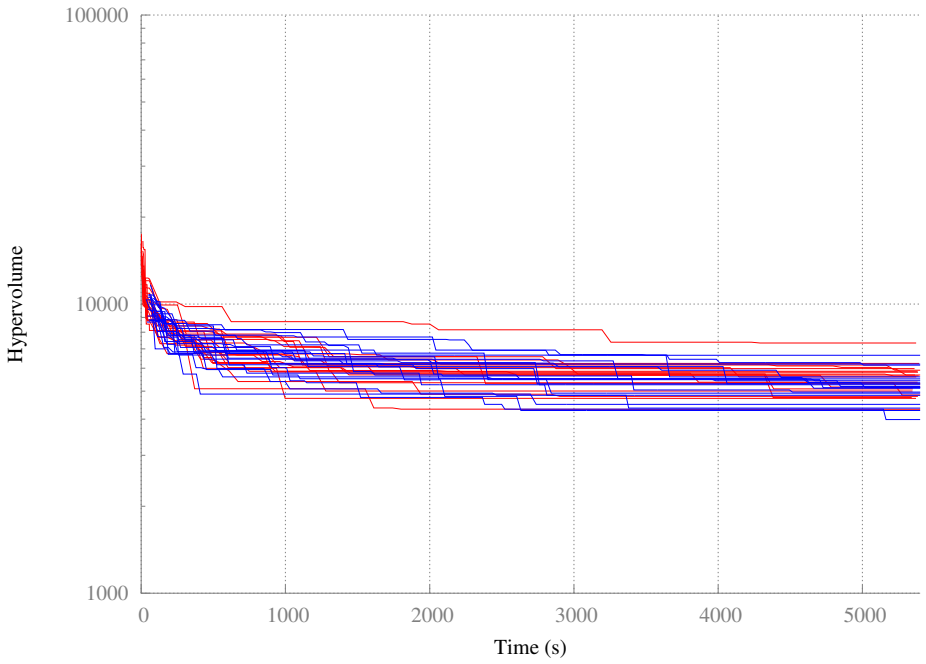
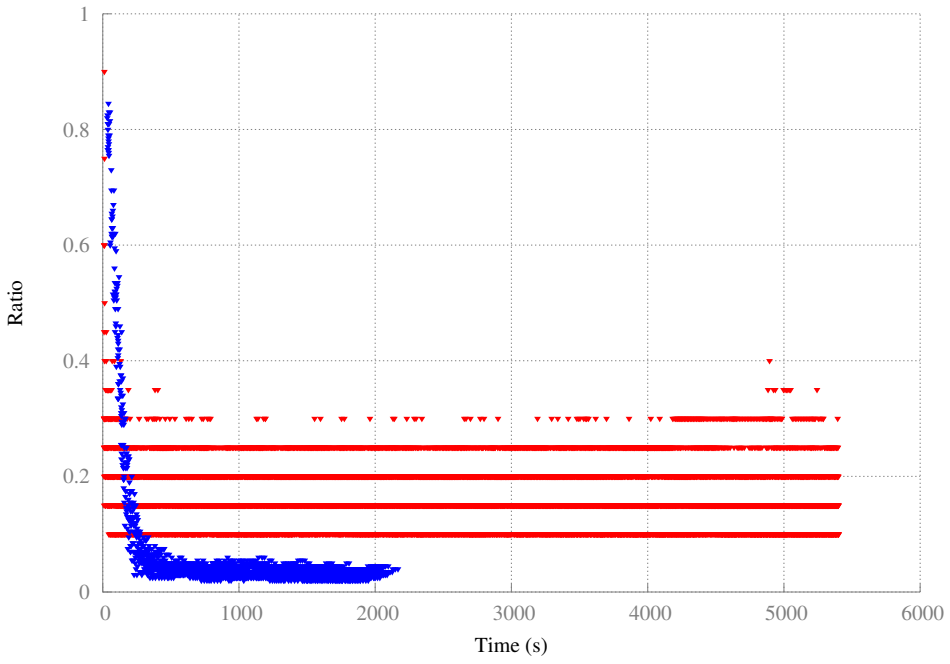


FIGURE 4.66: Instance 4 : Comparaison de la diversité des vecteurs objectifs.



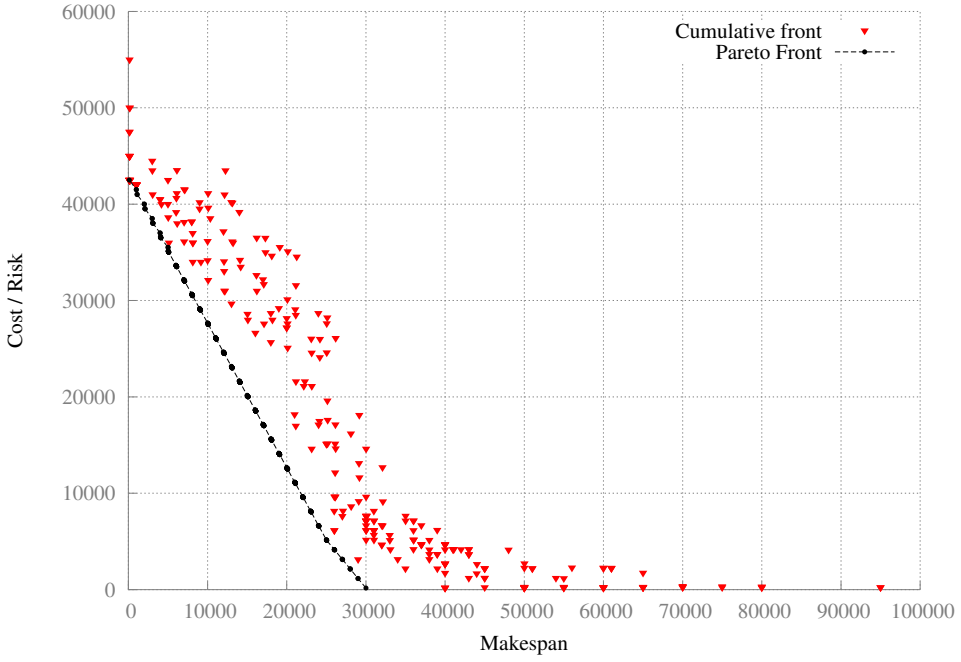
4.5.10.4 Instance 7

Malheureusement, à cause de problèmes techniques et contraintes de temps, il n'y a pas de résultat de la stratégie adaptative sur l'instance 7.

4.5.10.5 Instance 9

Une nouvelle fois, l'espace objectif est beaucoup moins visité que pour la stratégie statique, comme en témoigne la figure 4.68. La même structure de l'espace objectif apparaît à nouveau. Il est intéressant également de noter que la structure de la diversité des vecteurs objectifs est semblable à celle obtenue pour l'instance 1 et 4, ce qui laisserait à penser que la diversité des vecteurs objectifs est influencé par la stratégie utilisée.

FIGURE 4.67: Instance 9 : Fronts de Pareto cumulés pour tous les runs, à la fin de chaque run.



Comparaison avec la stratégie statique Si l'on observe la figure 4.74, la différence n'est pas très flagrante, surtout en considérant le *run* médian (en pointillés noirs). Les différences dans les surfaces d'atteinte, sur les figures 4.75 et 4.76 sont plus modérées que pour les instances précédentes. Cependant, le test statistique sur les fonctions d'atteinte indique, pour le premier comme pour le second ordre, un rejet de l'hypothèse nulle. Il semblerait donc que la stratégie statique soit meilleure, même si la différence n'est

FIGURE 4.68: Instance 9 : Population cumulées pour tous les runs et à tout temps.

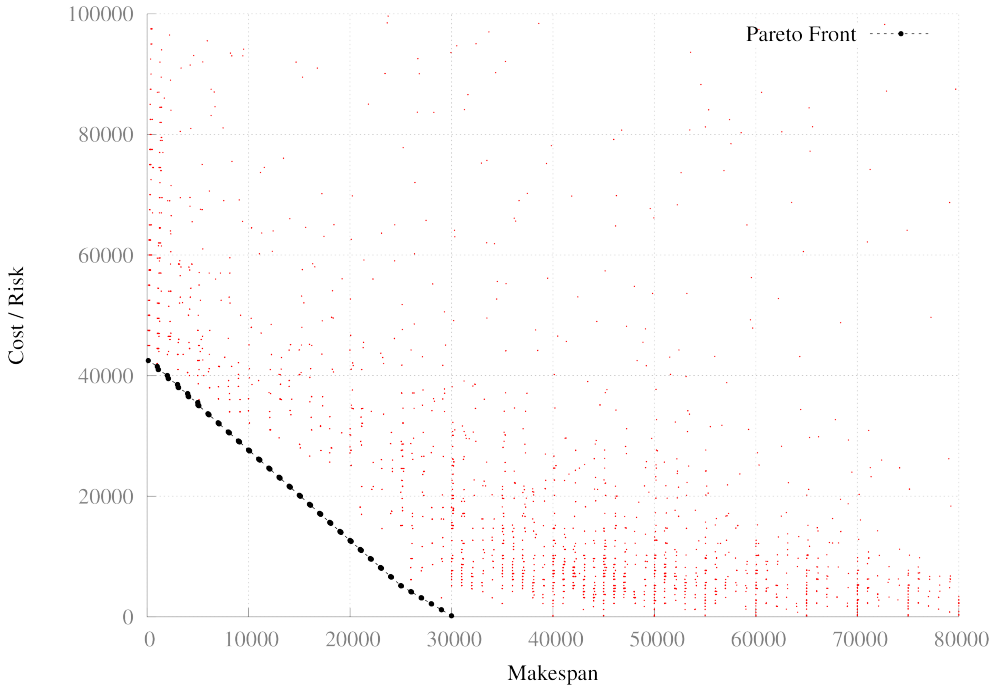


FIGURE 4.69: Instance 9 : Population cumulées pour tous les runs et à tout temps (version colorée).

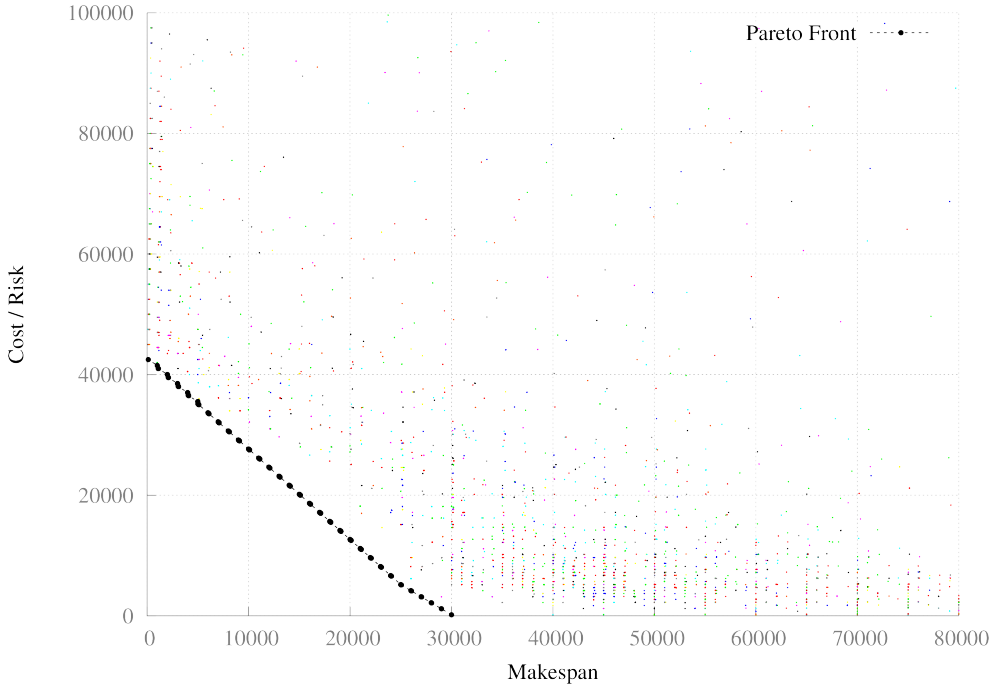
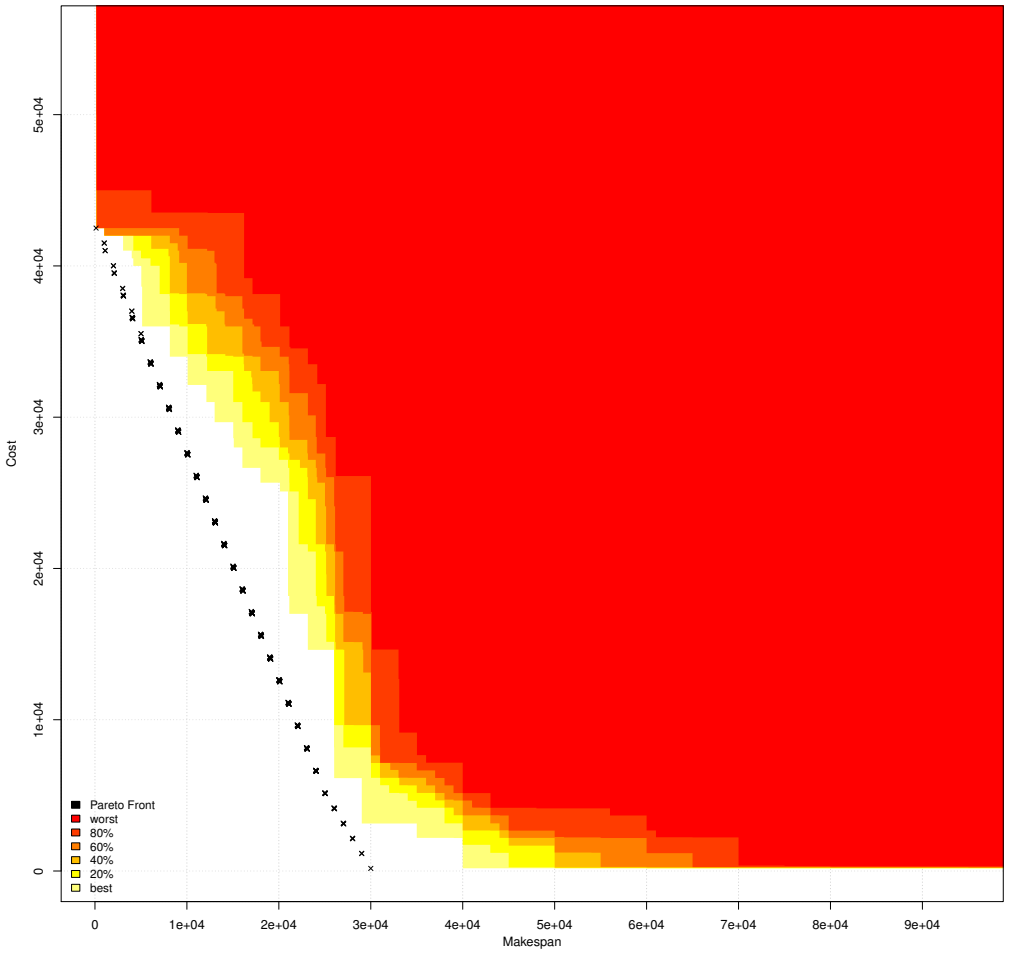


FIGURE 4.70: Instance 9 : Surfaces d'atteinte pour tous les runs.



pas si flagrante. L'hypervolume, figure 4.78, par exemple est une nouvelle fois largement identique dans la tendance, et la stratégie statique bénéficie de 2 *runs* mieux lotis, et que l'on peut par ailleurs appercevoir sur les surfaces d'atteinte, 4.31. Il est clair cependant que la diversité est bien plus grande pour la stratégie classique que pour la stratégie adaptative.

FIGURE 4.71: Instance 9 : Trajectoires de l'hypervolume pour tous les runs.

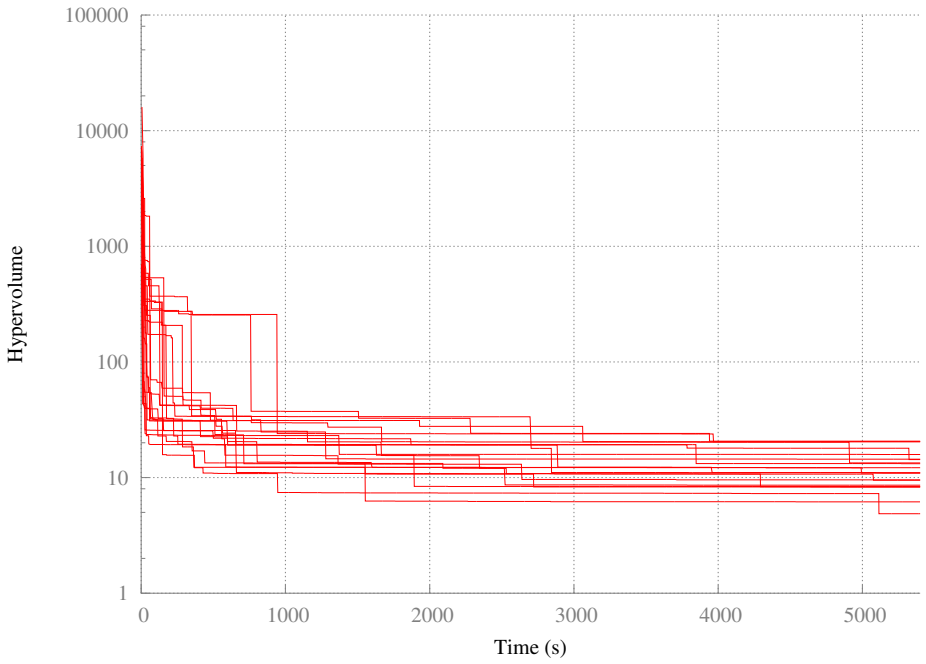


FIGURE 4.72: Instance 9 : Diversité des vecteurs objectifs pour tous les runs.

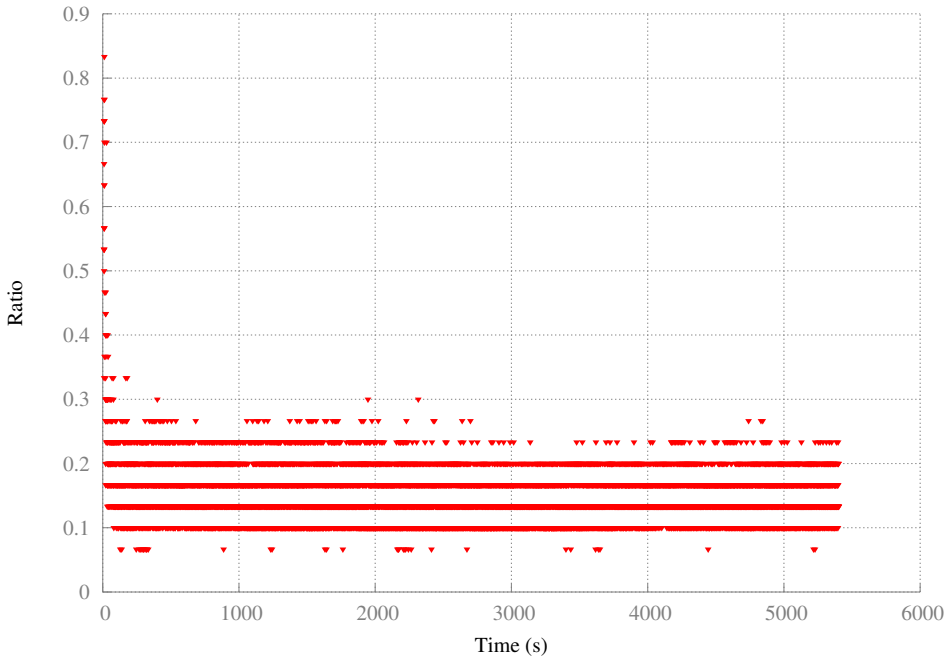


FIGURE 4.73: Instance 9 : Diversité des vecteurs objectifs pour tous les runs (version colorée).

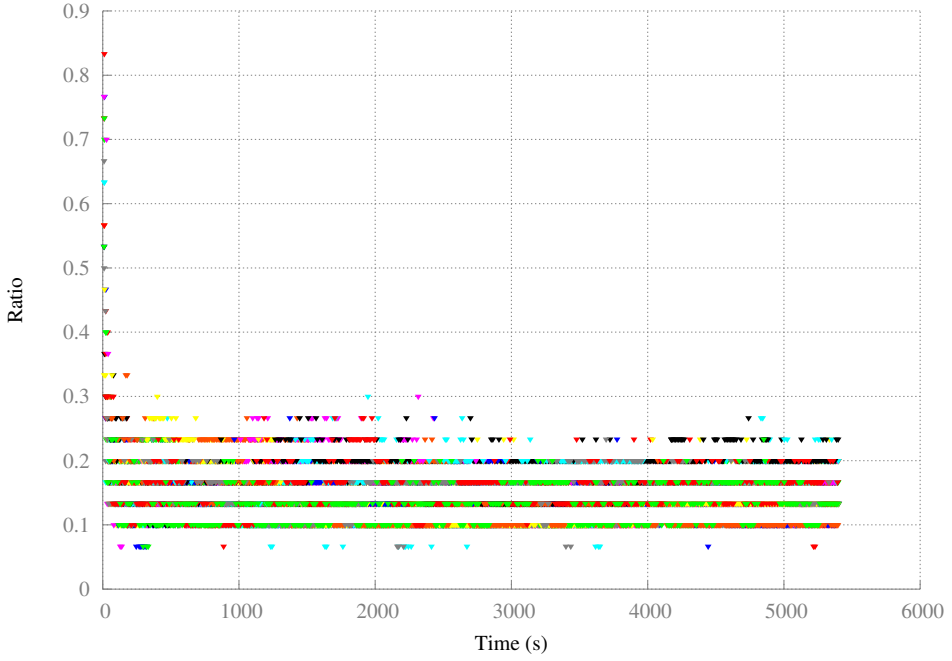


FIGURE 4.74: Instance 9 : Comparatif des surfaces d'atteinte.

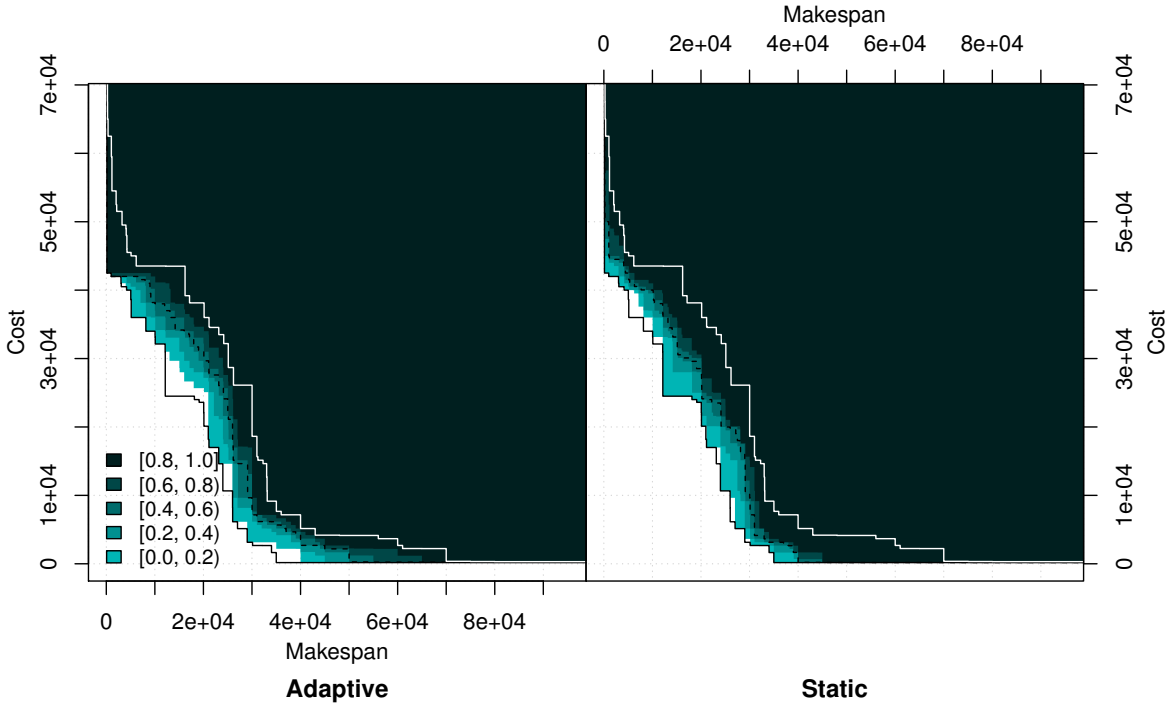


FIGURE 4.75: Instance 9 : Adaptative comparée à Statique.

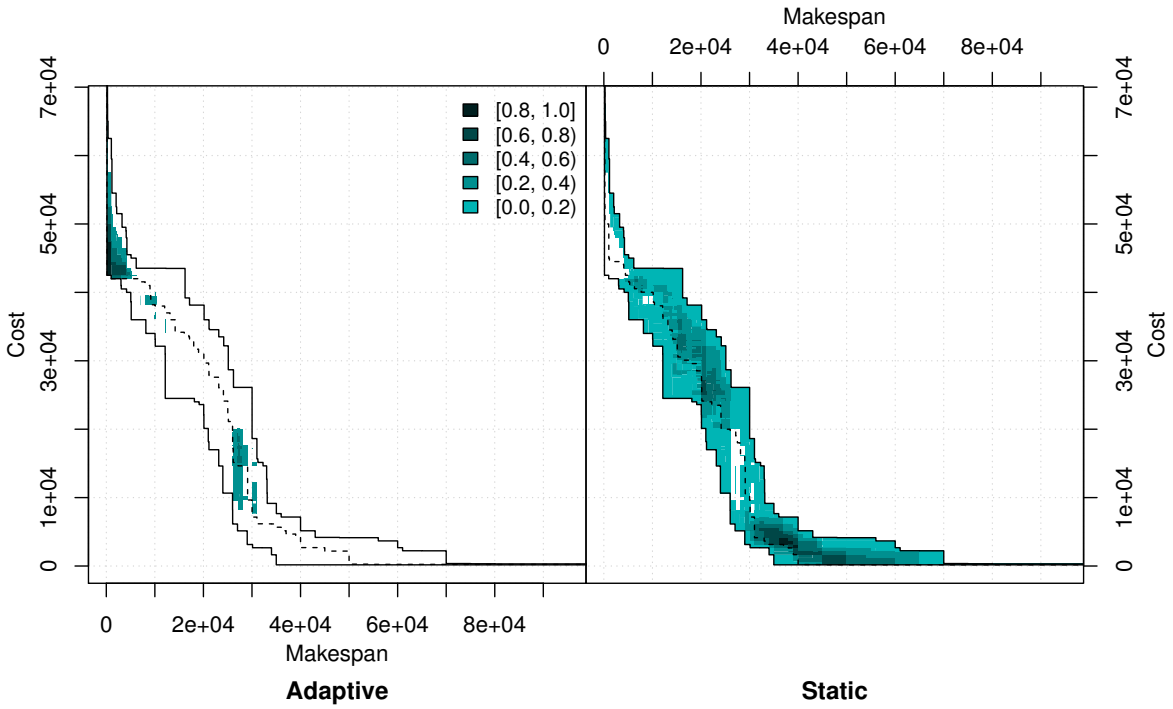


FIGURE 4.76: Instance 9 : Statique comparée à Adaptative.

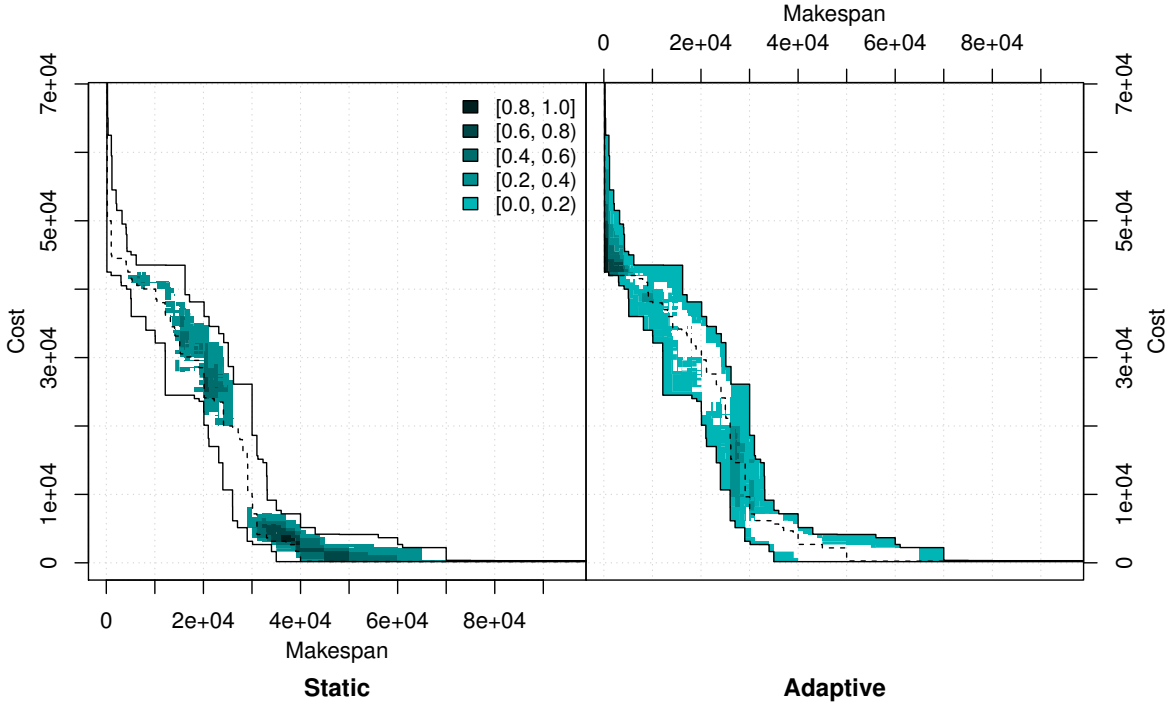


FIGURE 4.77: Instance 9 : Comparaison de l'hypervolume.

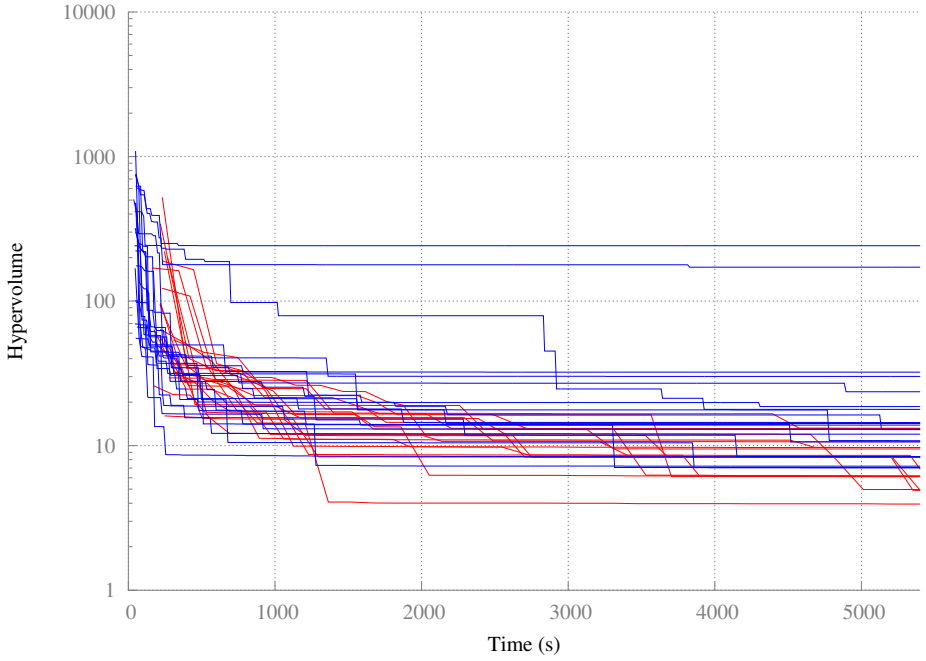
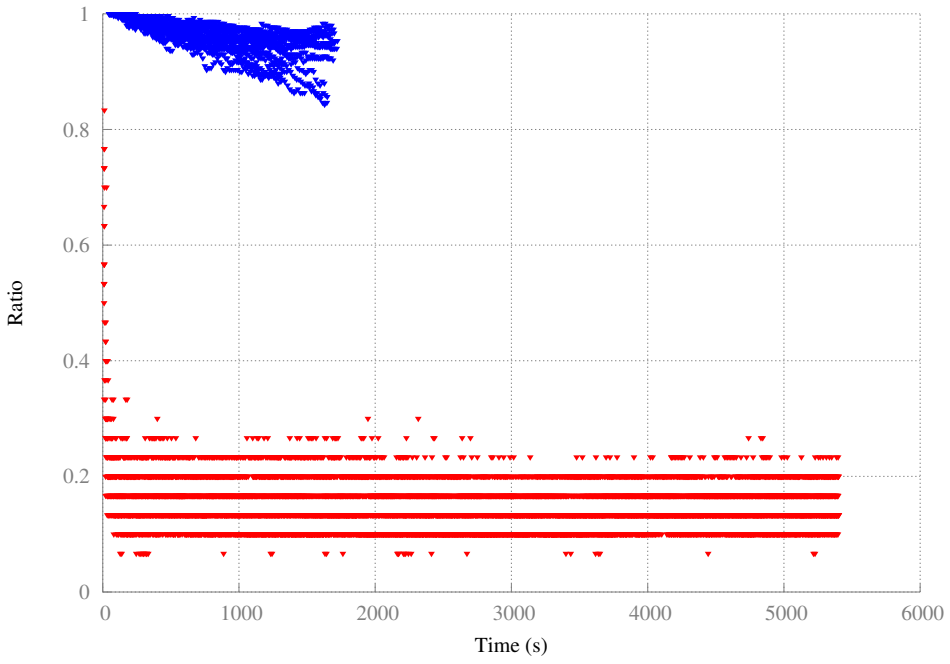


FIGURE 4.78: Instance 9 : Comparaison de la diversité des vecteurs objectifs.



4.5.11 Conclusion

La première conclusion qui apparaît est que la diversité semble dépendre de la stratégie utilisée. À contrario, si l'espace objectif visité dépend de la stratégie, on retrouve la même structure de l'espace objectif et le même comportement de l'hypervolume.

Malgré tout, les résultats sont décevants quelque soit l'instance. On pourrait éventuellement régler via ParamILS les hyper-paramètres comme le seuil et la tolérance du test de Page-Hinkley. Cependant, à la vue de ces résultats, nous avons décidé d'essayer d'autres stratégies plutôt que de prendre le temps de tester et paramétrer l'ensemble des facteurs de la stratégie adaptative, notamment une stratégie dite « gloutonne » et une stratégie auto-adaptative.

4.6 Stratégie gloutonne

La stratégie gloutonne consiste à évaluer un individu pour chacun des objectifs et de choisir le meilleur des vecteurs objectifs ainsi atteignable. Ainsi, dans le cas présent, nous avons 4 objectifs possibles à passer à YAHSP, ce qui correspondra donc à 4 appels au solveur et donc, vraissemblablement [6], à une moyenne de 4 fois plus de temps CPU par évaluation.

L'idée derrière est de se dire qu'entre chaque évaluation d'un individu, l'individu a été modifié et qu'il n'est pas nécessaire que l'objectif qui était le meilleur à l'instant t soit le même à l'intant $t+1$. Du coup, si l'on considère que l'objectif à utiliser via YAHSP a une grande importance sur les plans trouvés, faire le meilleur choix localement parmi tous les objectifs peut mener à se rapprocher du front plus vite, malgré le surcoût induit par les différents appels à YAHSP.

Définition 4.1. Soit u et v deux vecteurs objectifs atteignables à partir du vecteur objectif w.

$$u \succ v \Leftrightarrow \sum_{i=1}^{m} w_i - u_i > \sum_{i=1}^{m} w_i - v_i$$

6. Vraissemblablement seulement puisque selon l'objectif le temps de résolution peut fortement varier.

Cela correspond à choisir l'indicateur d'amélioration absolue par rapport au point de référence w définit plus haut.

Il est possible que les vecteurs u et v soient dominés par w, mais il faut nécessairement en choisir un des deux.

Notons qu'avec cette seule relation, il est possible, par exemple, de choisir u qui est dominé par w au détriment de v qui ne l'est pas. Une version plus élitiste consisterait à sélectionner en priorité le vecteur qui domine le second, et dans un second temps, en cas d'impossibilité à les comparer, de se référer à la relation précédente.

On propose également quelques variantes :

— **Sélection** *stricte* : Soit u et v deux vecteurs atteignables, l'idée consiste à sélectionner en priorité le vecteur qui domine le second, et dans un second temps, en cas d'impossibilité à les comparer, de se référer à la relation $\succ$ définit plus haut.
— **Amélioration relative** : Dans la relation $\succ$ ci-dessus, c'est l'amélioration absolue par rapport au point de référence w qui est utilisée. L'utilisation de l'amélioration relative pourrait éviter d'éventuels problèmes avec des instances dont les deux objectifs n'auraient pas le même ordre de grandeur par exemple. [7]
— **Objectifs restreints** : Pour limiter le surcoût induit par les 3 appels supplémentaires à YAHSP il pourrait être intéressant de ne tester qu'avec les deux objectifs du problème (makespan_max, cost) plutôt qu'avec les quatres stratégies de YAHSP (les deux précédentes auxquelles s'ajoutent length et makespan_add).
— **Archive** : On teste pour chaque vecteur objectif atteignable s'il est sur le front connu. Si c'est le cas, on le choisit.

4.6.1 Résultats empiriques sur les instances larges

4.6.1.1 Particularités du protocole

Dans un premier temps, une série d'expériences a été réalisée sur les instances larges avec les même paramètres que ceux de la stratégie statique pour l'instance correspondante. Les résultats sont statistiquement inférieurs à la stratégie statique, mais des tests ont conduits à s'interroger sur les paramètres optimaux pour la stratégie gloutonne. Une

7. On pourrait en réalité tester avec n'importe quel indicateur λ définit plus tôt.

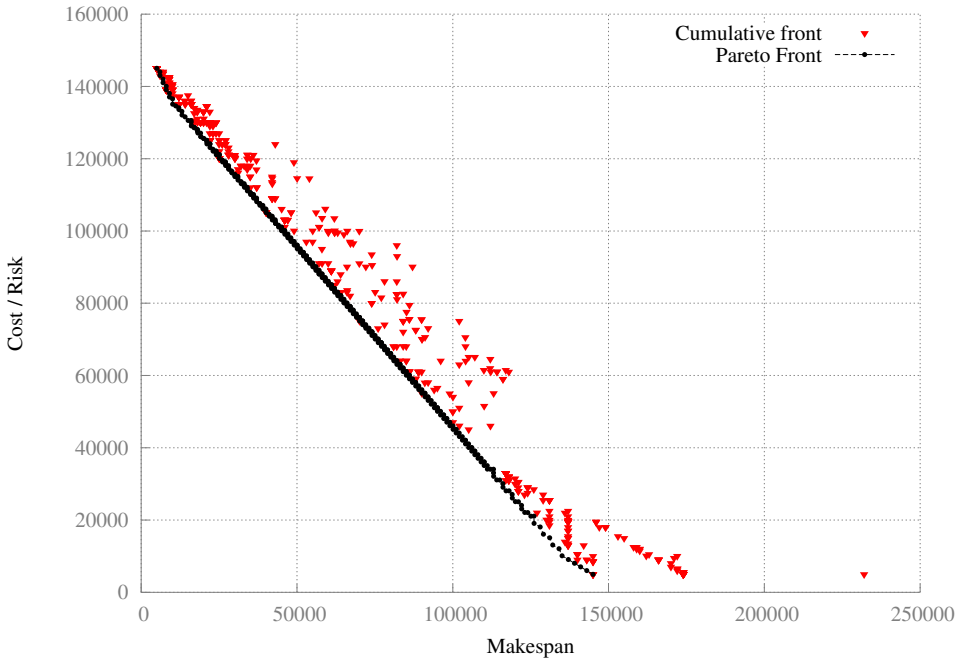
seconde série a été réalisé en augmentant fortement les paramètres de la diversité et en divisant par un facteur entre 100 et 10000 la valeur de b_{max}.

4.6.1.2 Série 1

Les paramètres Les paramètres de cette série sont ceux de la table 4.3 auquel on retire évidemment les poids sur les différents objectifs.

Instance 1 Les fronts cumulés sont plutôt semblables à ceux de la stratégie adaptative et apparaissent clairement moins bons que ceux de la stratégie statique. On retrouve à nouveau une zone moins explorée au niveau du front exact, comme pour la stratégie statique. La zone échantillonnée semble tout de même plus dense et moins disparate que pour la stratégie statique. Il semblerait que la recherche se concentre donc plus proche du front.

FIGURE 4.79: Instance 1 : Fronts de Pareto cumulés pour tous les runs, à la fin de chaque run.



La figure 4.82, représentant les surfaces d'atteinte, présente les même similarités avec la stratégie adaptative, à avoir une meilleure atteinte des zones du front avec un faible *makespan*, malgré un échantillonnage parfaitement différent de l'espace objectif.

FIGURE 4.80: Instance 1 : Population cumulées pour tous les runs et à tout temps.

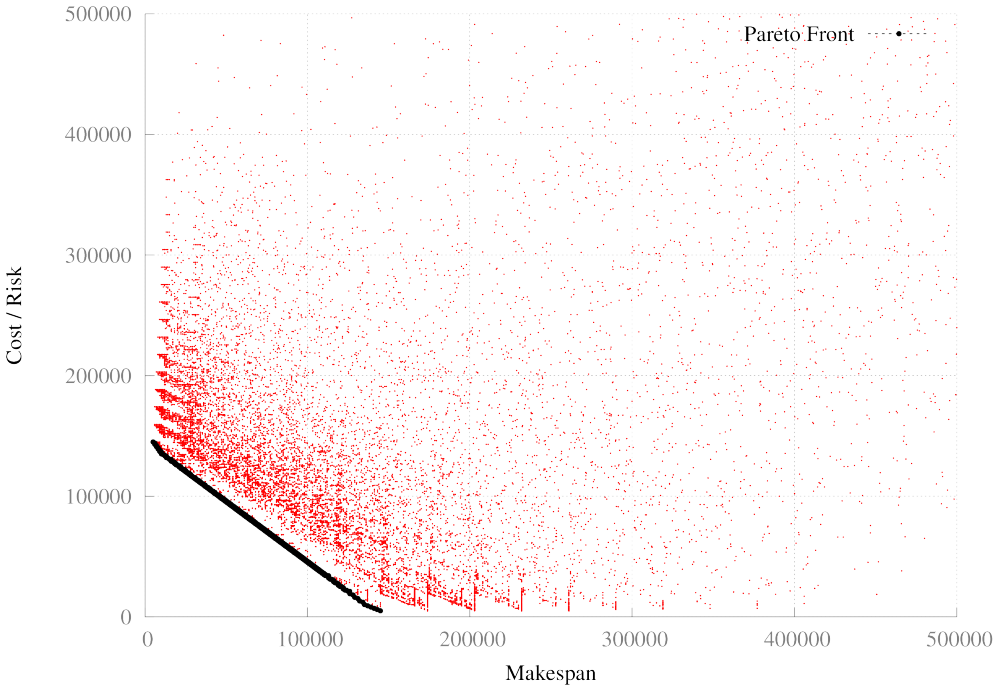


FIGURE 4.81: Instance 1 : Population cumulées pour tous les runs et à tout temps (version colorée).

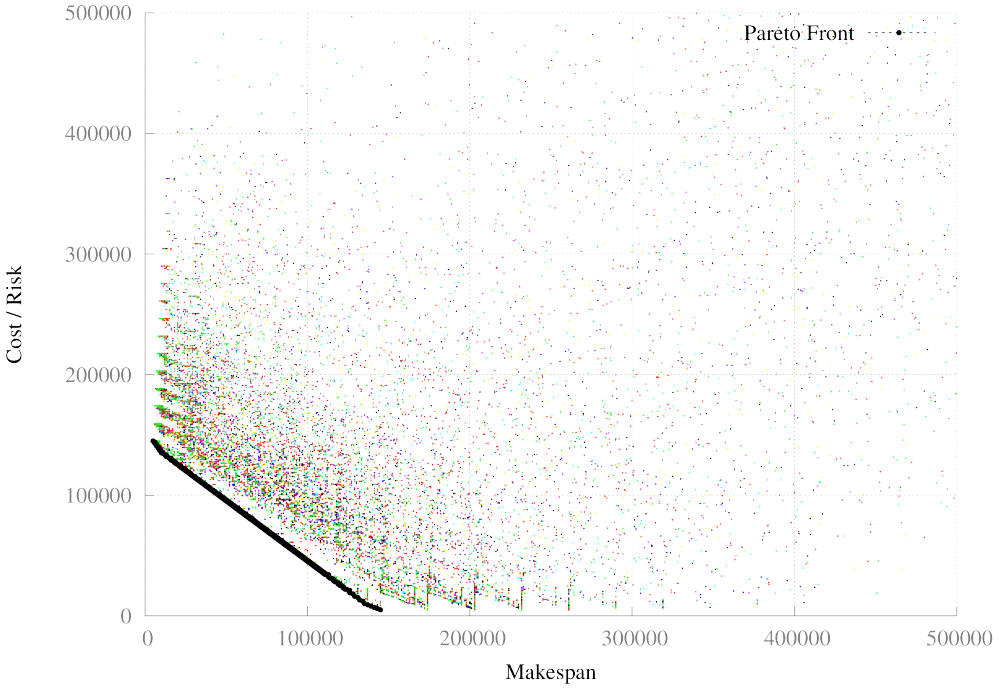
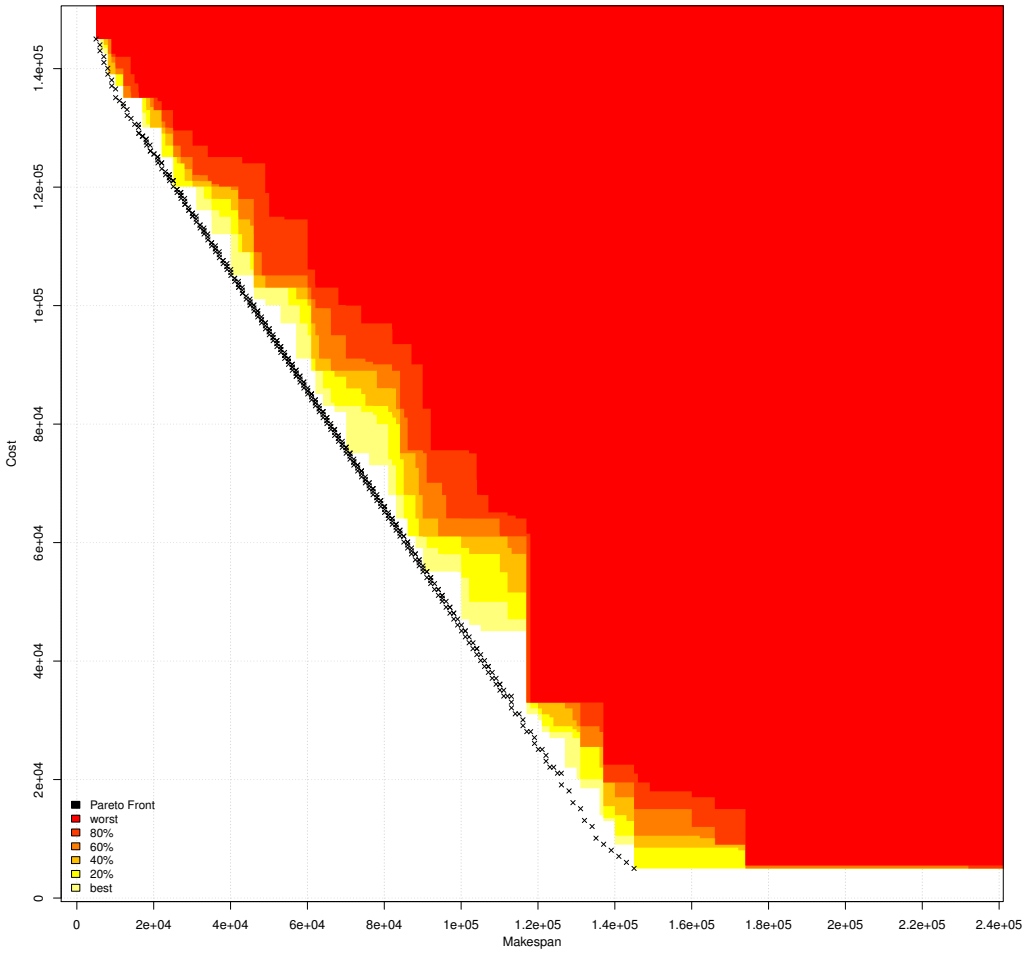


FIGURE 4.82: Instance 1 : Surfaces d'atteinte pour tous les runs.



En fois encore, les trajectoires de l'hypervolume possède ce « saut » caractéristique de l'instance 1 et la diversité ressemble.

On observe clairement sur la figure 4.84 et 4.85 une structure de l'évolution de la diversité bien différente de la stratégie adaptative ou de la stratégie statique : partant d'une diversité proche de 100%, les trajectoires sont distinguables et vont décroitres de plus en plus lentement jusqu'à se stabiliser vers une tendance entre 35 et 50%, avec une amplitude entre les *runs* légèrement croissante au cours du temps.

Comparaison avec la stratégie statique Sans grande surprise, les résultats de la comparaison entre la stratégie statique et gloutonne mènent aux même commentaires et conclusions qu'avec la stratégie adaptative : la stratégie statique est statistiquement meilleure que la stratégie gloutonne.

FIGURE 4.83: Instance 1 : Trajectoires de l'hypervolume pour tous les runs.

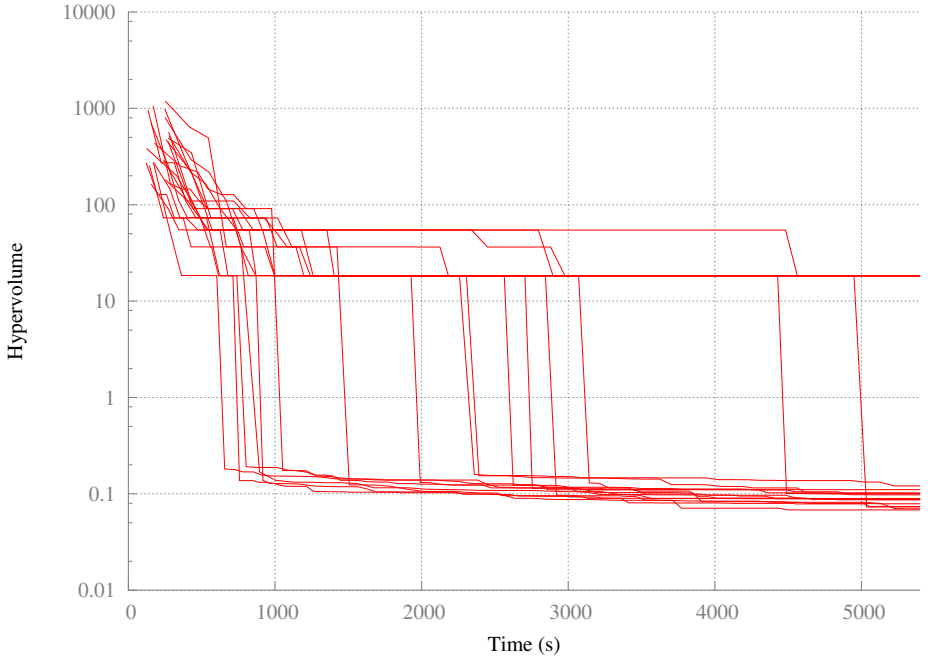


FIGURE 4.84: Instance 1 : Diversité des vecteurs objectifs pour tous les runs.

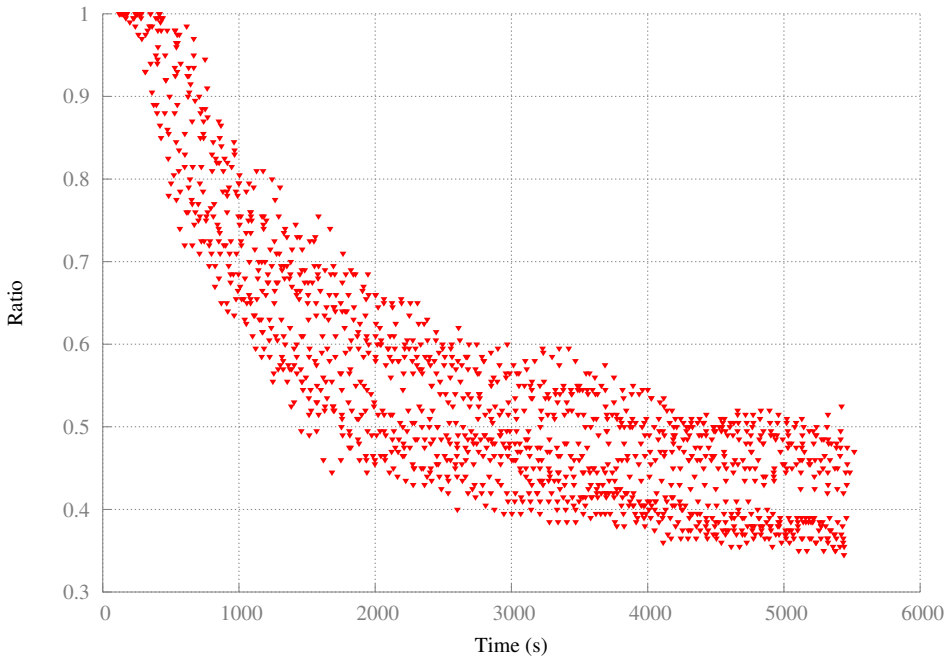


FIGURE 4.85: Instance 1 : Diversité des vecteurs objectifs pour tous les runs (version colorée).

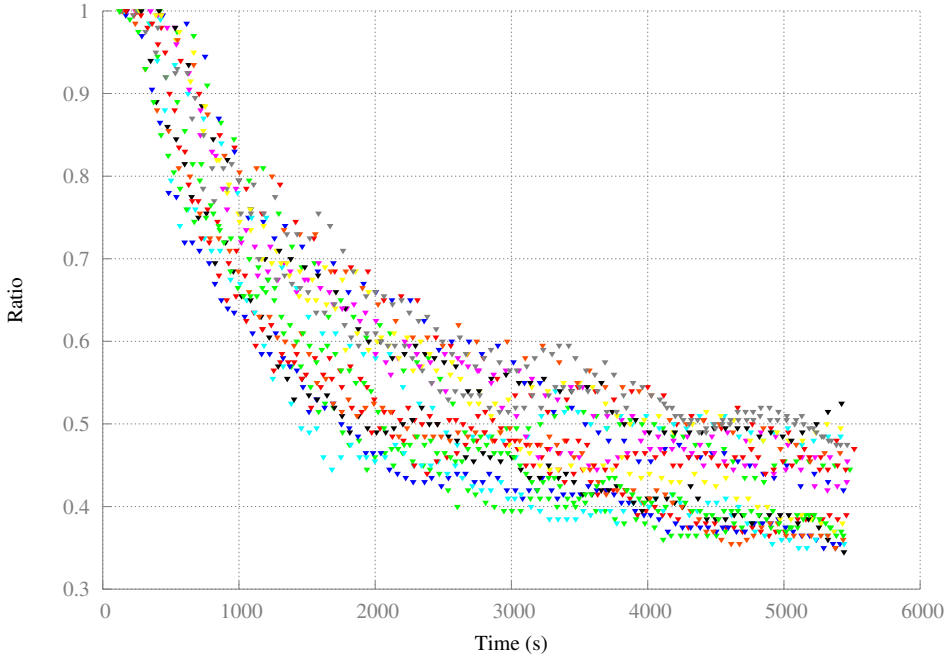


FIGURE 4.86: Instance 1 : Comparatif des surfaces d'atteinte.

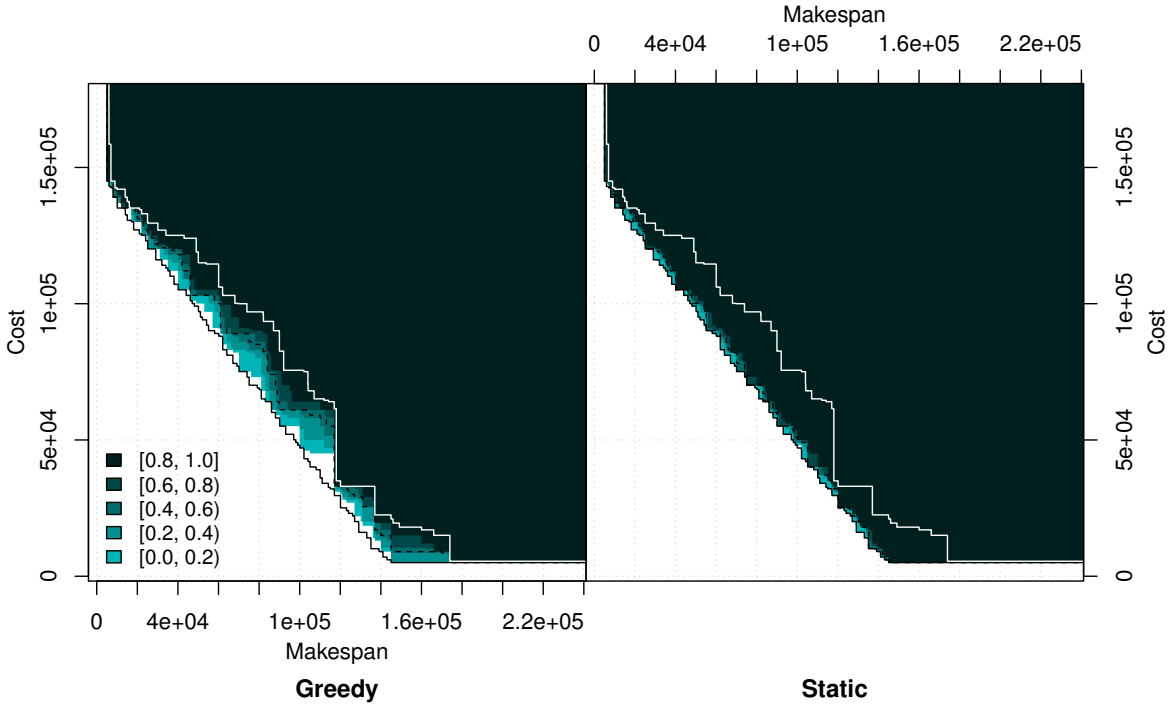


FIGURE 4.87: Instance 1 : Gloutonne comparée à Statique.

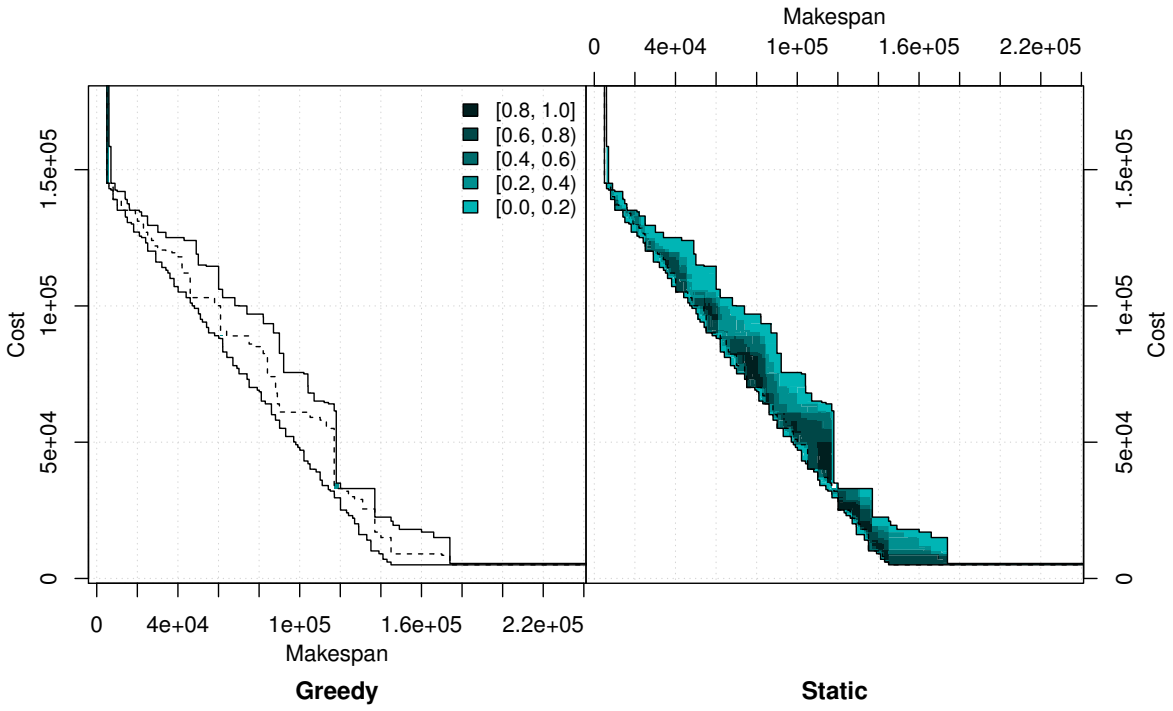


FIGURE 4.88: Instance 1 : Statique comparée à Gloutonne.

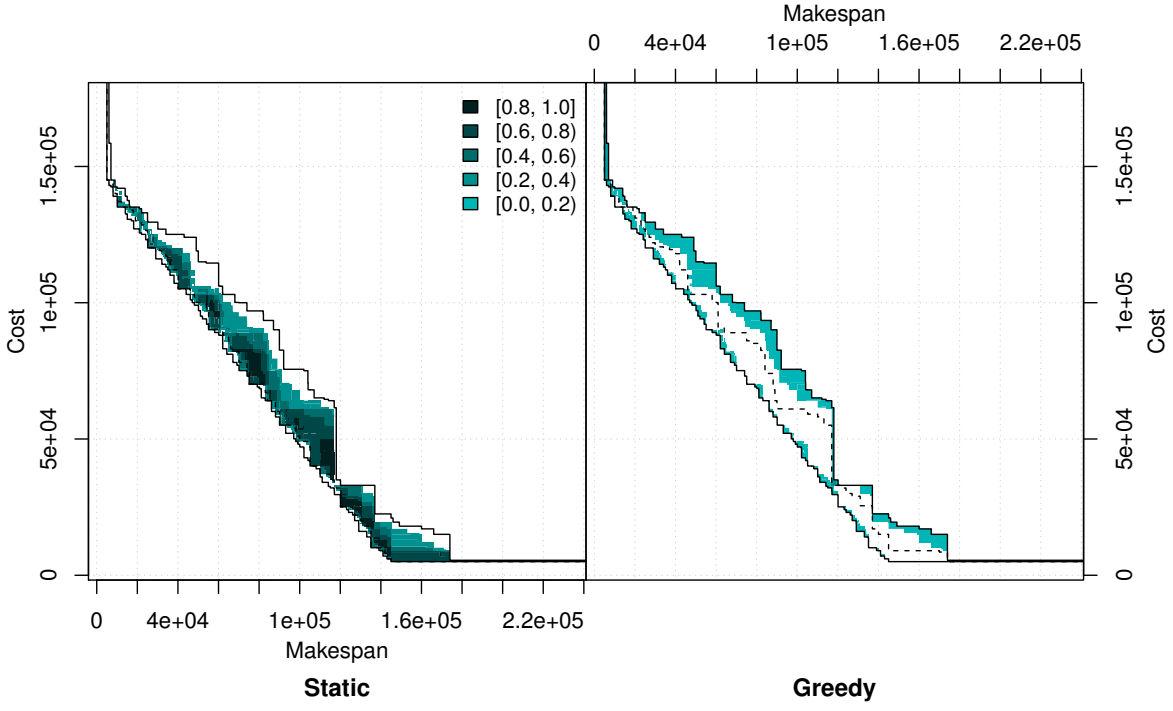


FIGURE 4.89: Instance 1 : Comparaison de l'hypervolume.

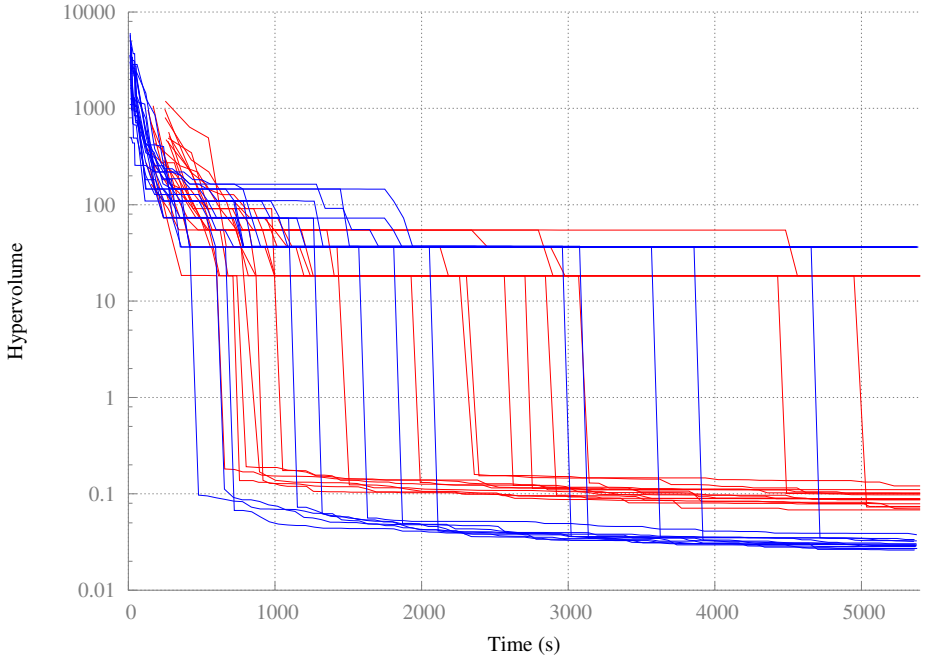
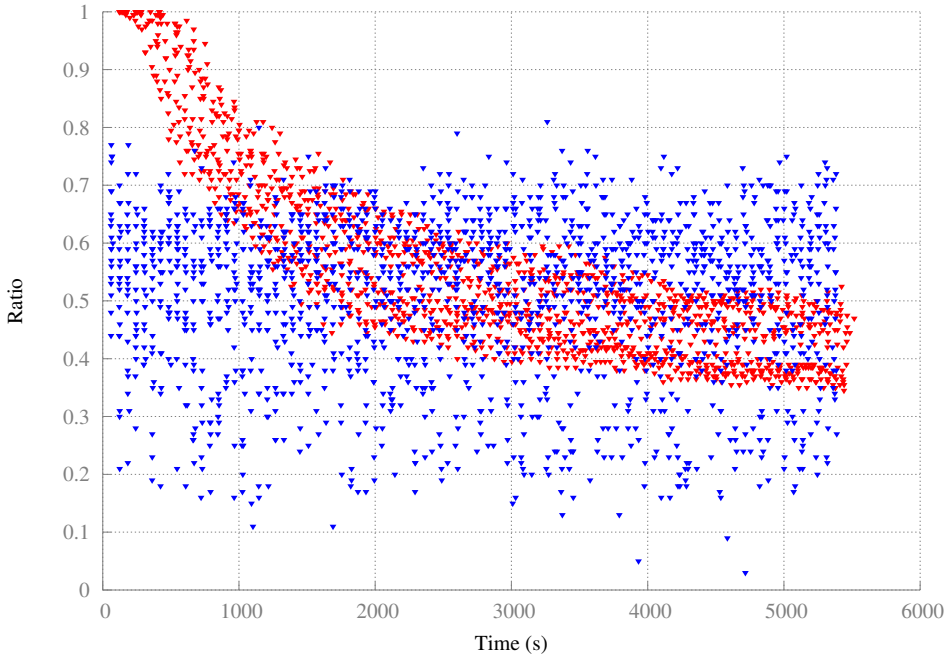


FIGURE 4.90: Instance 1 : Comparaison de la diversité des vecteurs objectifs.



Instance 4 Si les front cumulés n'amènent pas de remarque particulière, on note cependant que la population cumulée présente une densité un peu plus importante sur les extrémités de la zone échantillonnée par rapport à la stratégie classique qui était plus uniforme.

FIGURE 4.91: Instance 4 : Fronts de Pareto cumulés pour tous les runs, à la fin de chaque run.

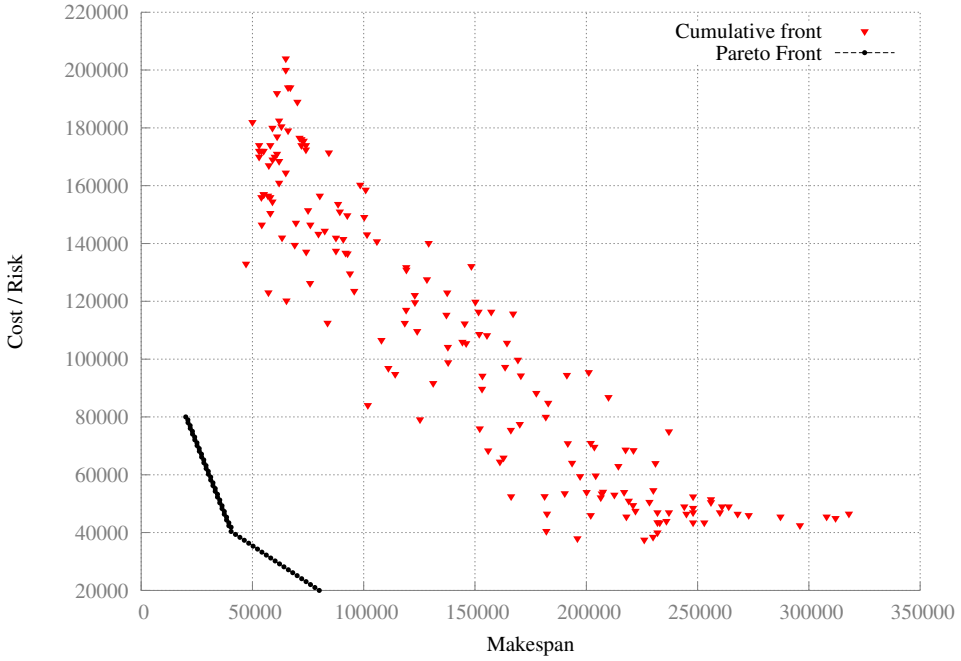


FIGURE 4.92: Instance 4 : Population cumulées pour tous les runs et à tout temps.

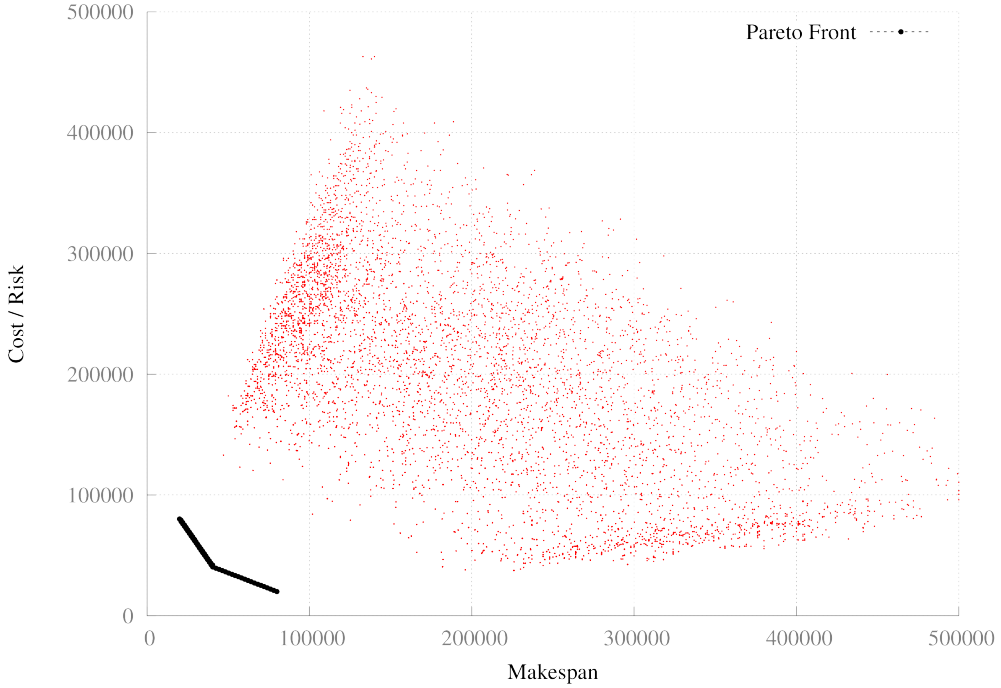
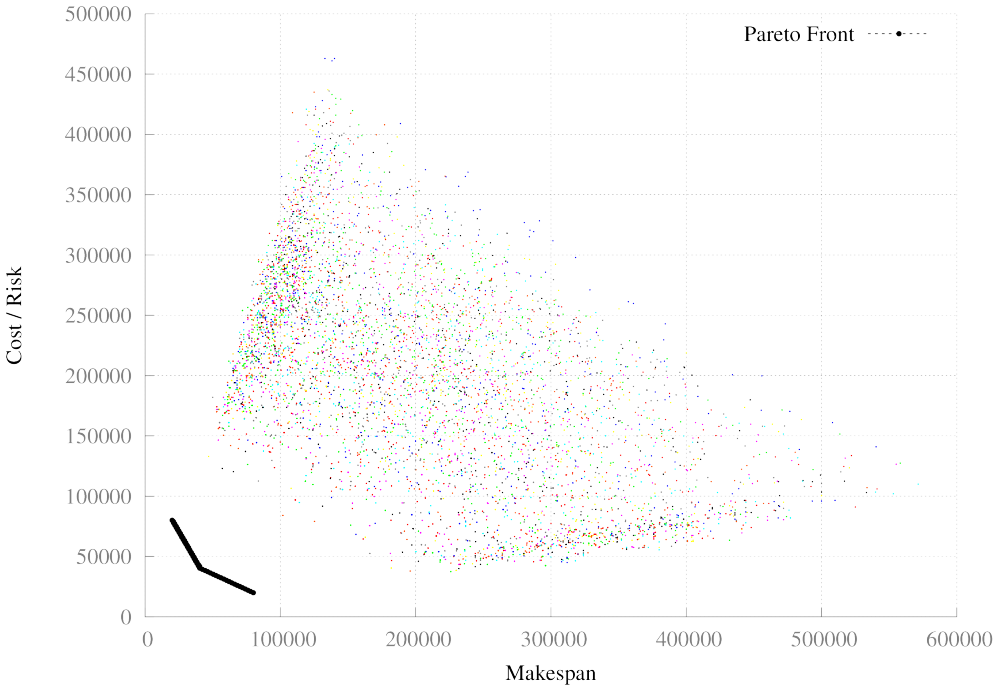


FIGURE 4.93: Instance 4 : Population cumulées pour tous les runs et à tout temps (version colorée).



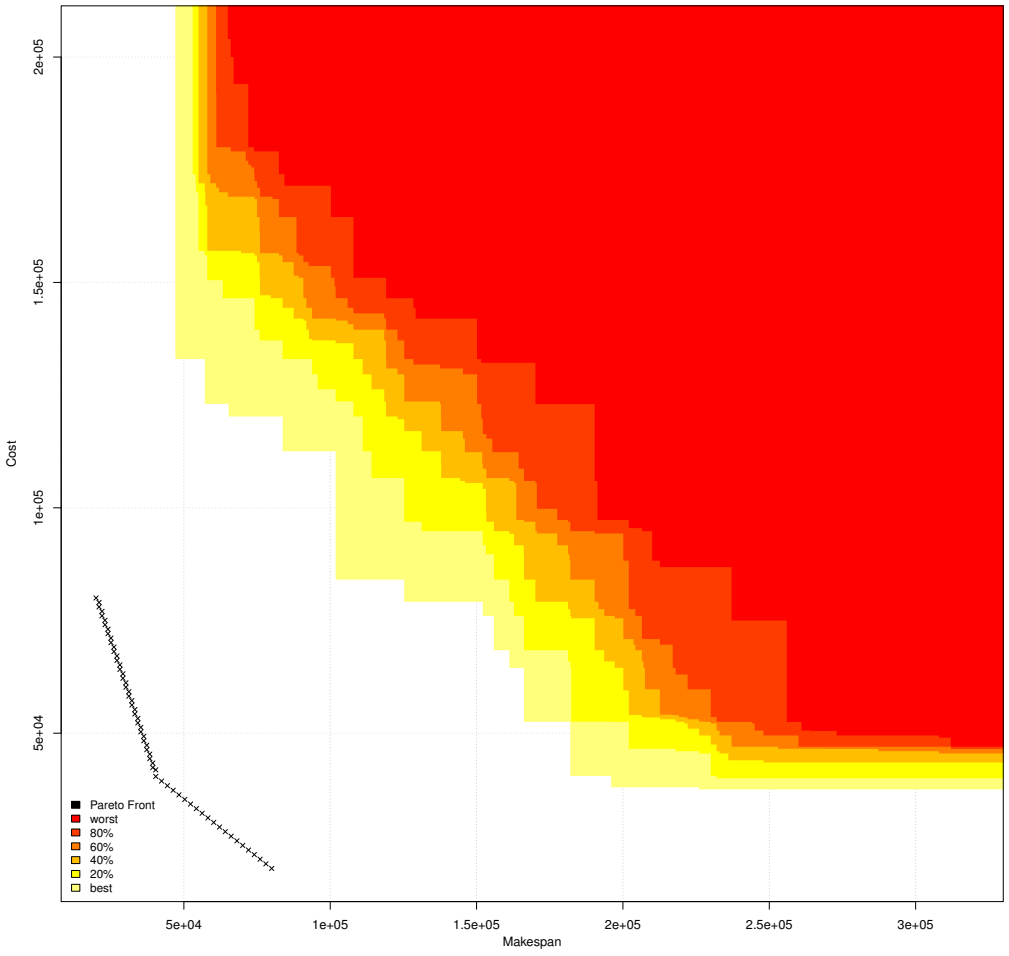
De même, les surfaces d'atteinte de la figure 4.94 présentent la même structure que pour la stratégie classique, c'est à dire une variance entre les *runs* assez importante.

Notons pour l'hypervolume une distinction avec tous les autres tracés : il ne semble pas y avoir de point de départ commun à tous les *runs*. On obtient une quasi-stationnarité de l'hypervolume immédiatement. En réalité, si l'on observe le début des trajectoires, elles apparaîssent plus tard que sur les autres graphiques comme pour la stratégie classique sur l'instance 7. Comme la trajectoire apparaît après la seconde évaluation de la population pour des raisons techniques [8] il est fort probable que cette seconde évaluation réussissent à faire fortement baisser l'hypervolume. Cette hypothèse semble confirmée par la superposition de l'hypervolume de la figure 4.102. Si l'on peut voir la stratégie gloutonne comme 4 évaluations en une, finalement on voit que la capacité de l'hypervolume à baisser semble moins dépendre des mécanismes de l'algorithme génétique que de la capacité du solveur local.

On observe par exemple sur la figure 4.96 une trajectoire similaire à celle observée pour la stratégie classique. À première vue, la stratégie gloutonne met plus de temps à voir

8. Une première évaluation est effectuée pour initialiser le foncteur d'évaluation, notamment avec une méthode d'estimation du paramètre b_{max} à utiliser, qui correspond au nombre de noeuds que peut parcourir YAHSP durant un appel.

FIGURE 4.94: Instance 4 : Surfaces d'atteinte pour tous les runs.



la diversité s'effondrer mais cela vient uniquement du fait que comme il faut 4 appels à YAHSP par évaluation et que le b_{max} est le même que pour la stratégie statique, la durée entre génération est plus longue. Il faut probablement environ le même nombre de générations avant d'atteindre le niveau le plus bas.

Comparaison avec la stratégie statique Les différences d'atteinte des figures 4.99 et 4.100 confirment ce que l'on pouvait observer avec les fronts cumulés, à savoir que les probabilités d'atteindre un point sont plus grandes sur les valeurs extrêmes du coût ou du *makespan*.

FIGURE 4.95: Instance 4 : Trajectoires de l'hypervolume pour tous les runs.

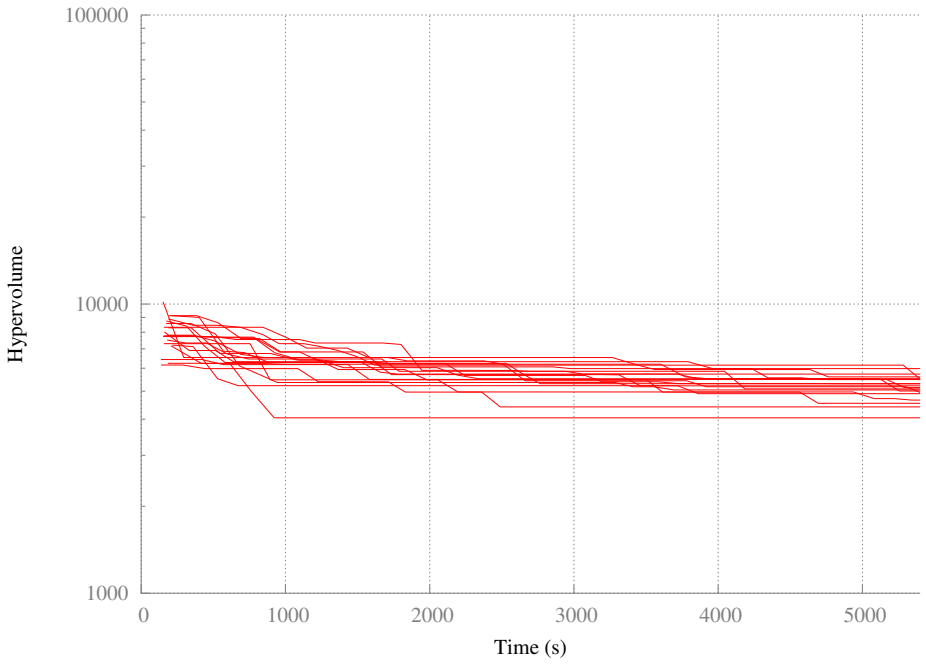


FIGURE 4.96: Instance 4 : Diversité des vecteurs objectifs pour tous les runs.

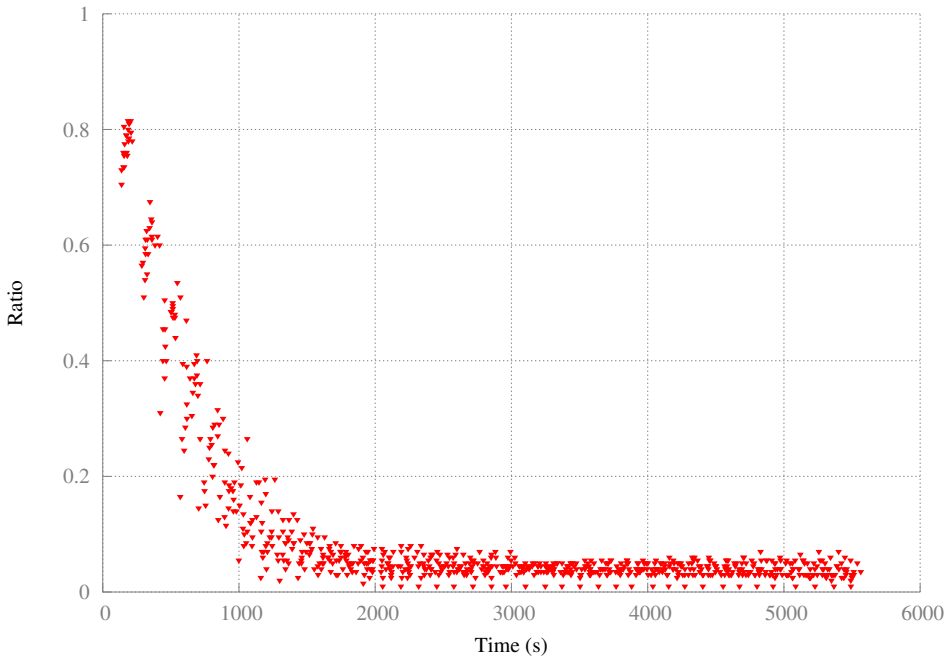


FIGURE 4.97: Instance 4 : Diversité des vecteurs objectifs pour tous les runs (version colorée).

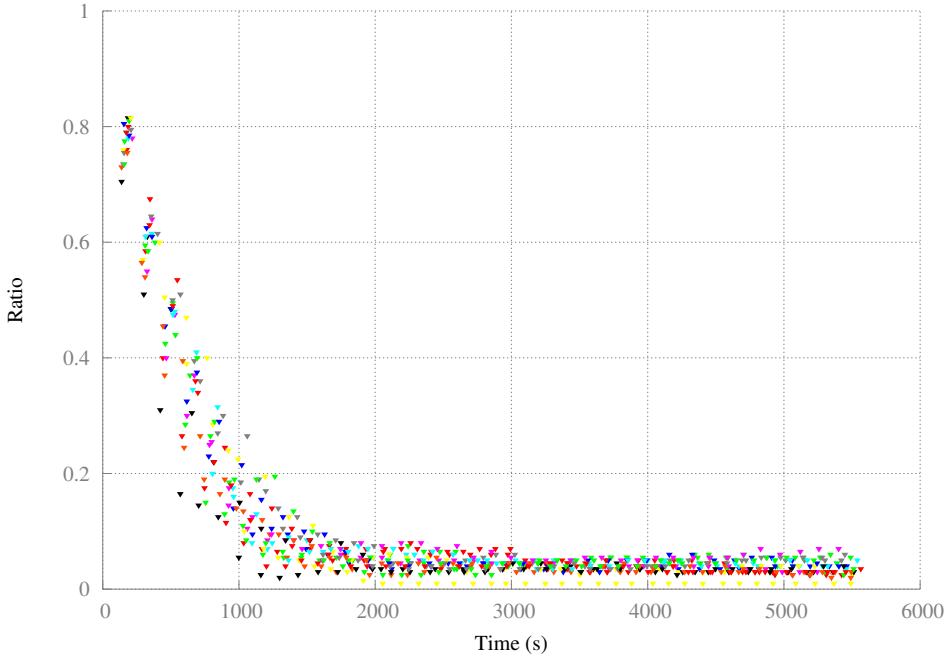


FIGURE 4.98: Instance 4 : Comparatif des surfaces d'atteinte.

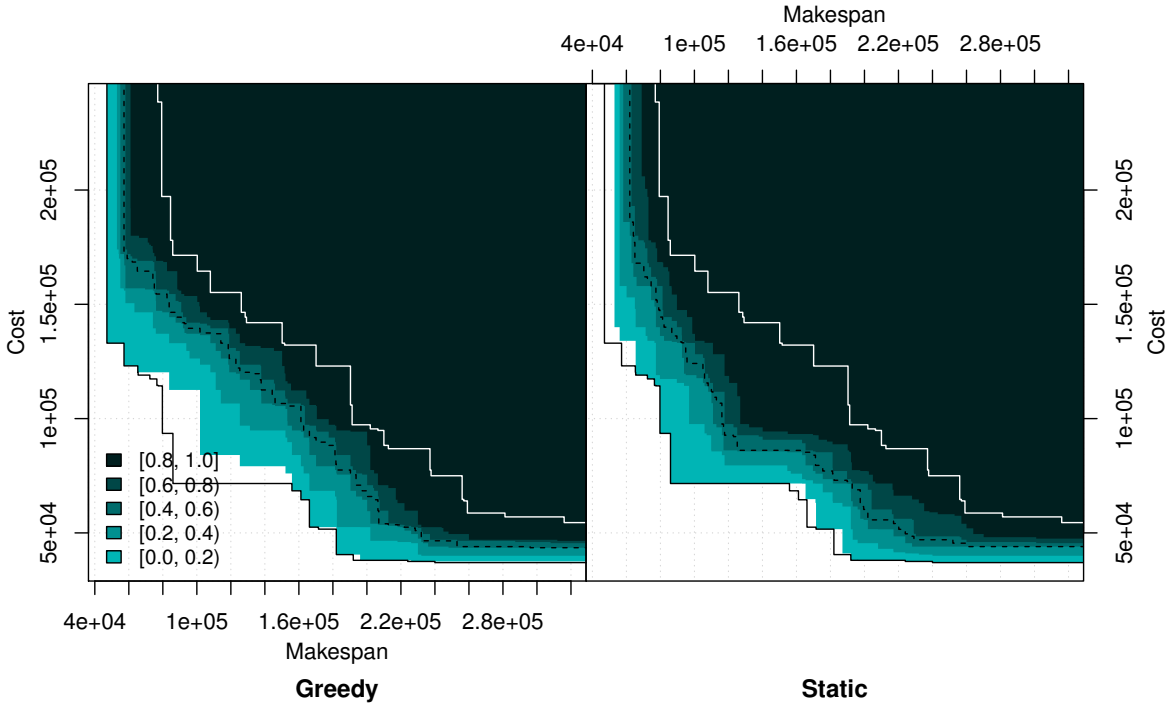


FIGURE 4.99: Instance 4 : Gloutonne comparée à Statique.

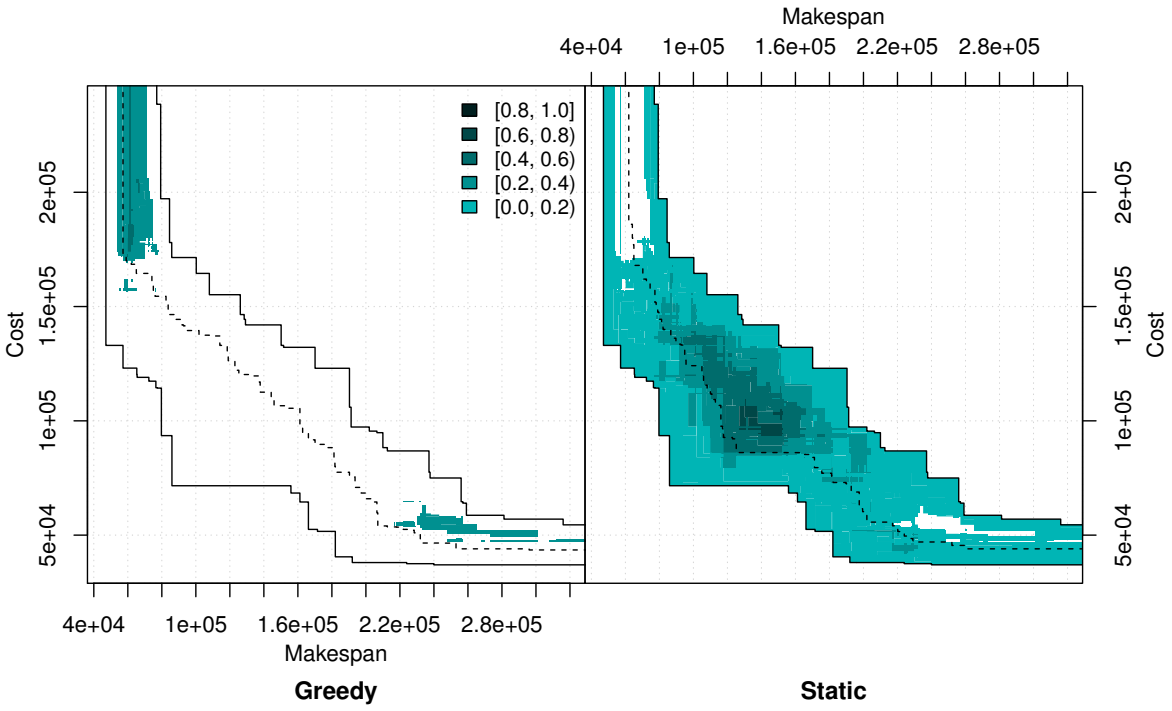


FIGURE 4.100: Instance 4 : Statique comparée à Gloutonne.

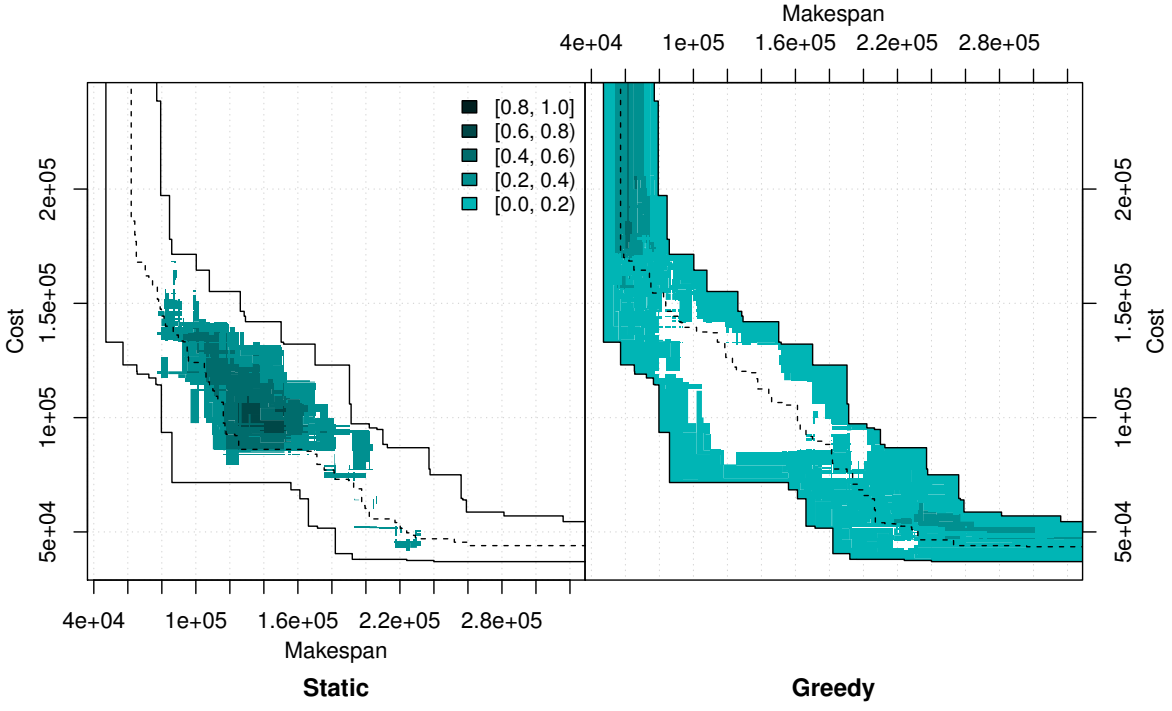


FIGURE 4.101: Instance 4 : Comparaison de l'hypervolume.

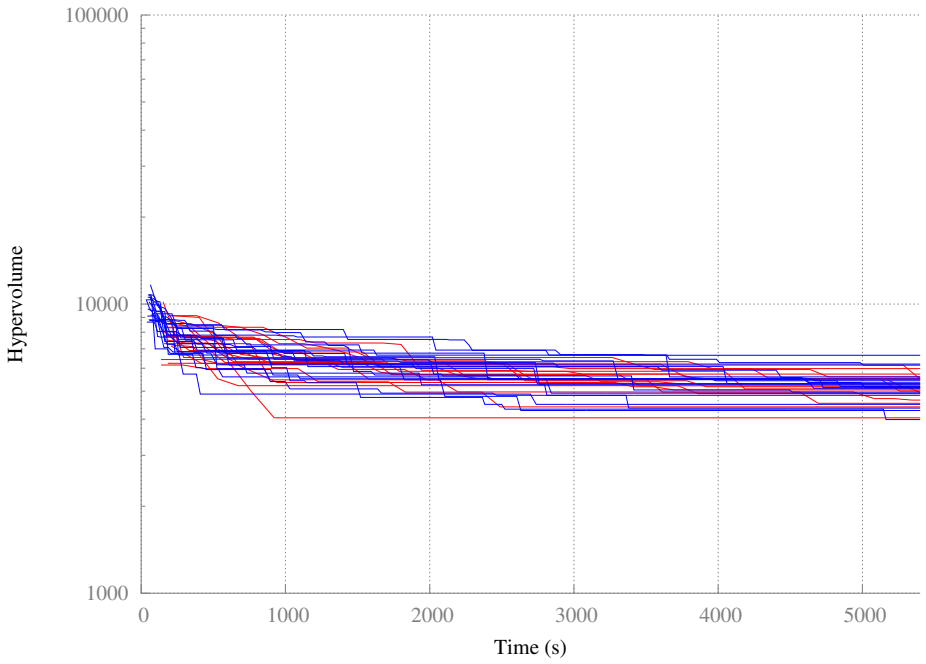
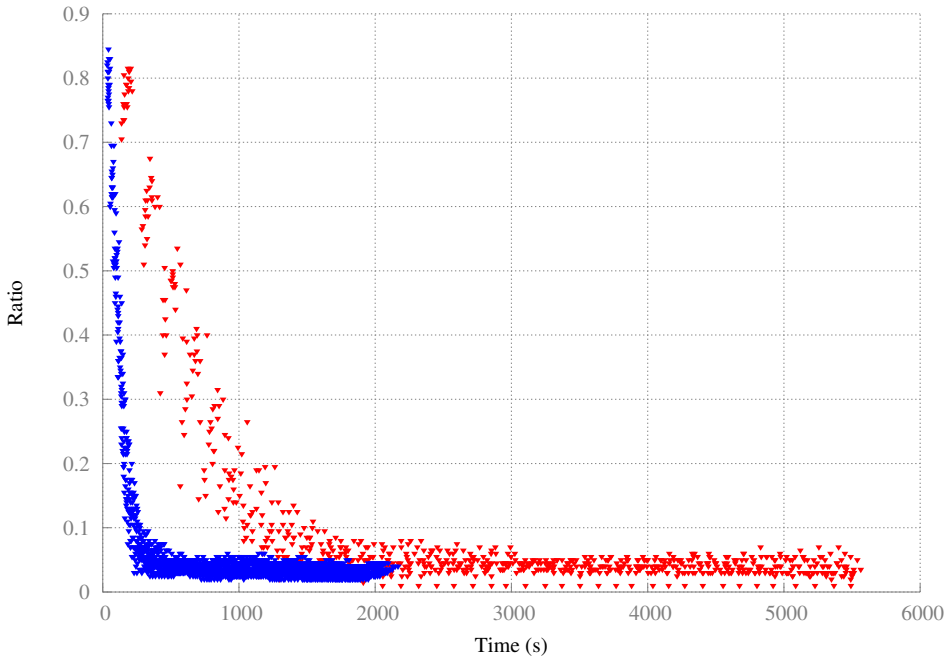
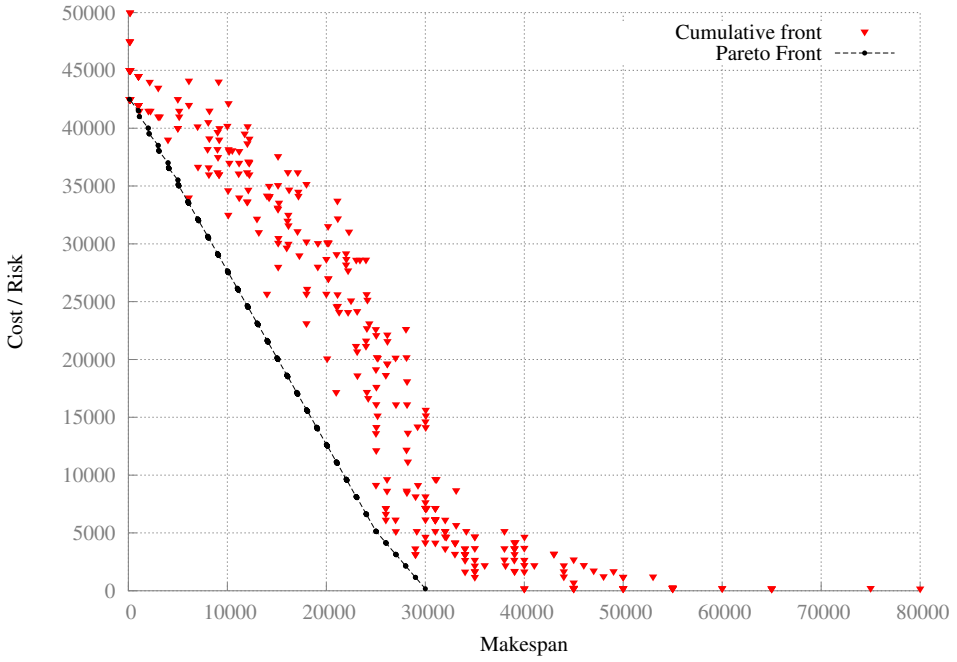


FIGURE 4.102: Instance 4 : Comparaison de la diversité des vecteurs objectifs.



Instance 9 Tout comme pour l'instance 1 et 4, la stratégie gloutonne semble présenter de grandes similarités avec la stratégie statique. Les fronts cumulés se ressemblent fortement et présentent une zone difficile d'atteinte proche du front, qui s'estompe sur les extrémités du front exact. La zone échantillonnée de l'espace objectif est également similaire et en accord avec les fronts cumulés. On retrouve également le même phénomène qu'avec l'instance 4 sur les trajectoires de l'hypervolume, à savoir qu'elles commencent avec un décalage temporel dû à la durée d'une évaluation, plus longue puisque composée de 4 appels à YAHSP mais qu'elles restent similaires à ce qu'on peut observer avec la stratégie statique (voire figure 4.114).

FIGURE 4.103: Instance 9 : Fronts de Pareto cumulés pour tous les runs, à la fin de chaque run.



Comparaison avec la stratégie statique Si les tests de statistique sur les fonctions d'atteinte donne la stratégie classique meilleure que la stratégie gloutonne, les résultats ne semblent pas visuellement différents si l'on observer les surfaces d'atteinte. Il est difficile de dire distinguer la meilleure des stratégies à partir de la figure 4.110 et surtout à partir des différence d'atteinte montrées sur les figures 4.111 et 4.112, notamment parce que les zones meilleures pour la stratégie classique se situent vers le centre et sur les valeurs du bas coûts, tandis que les zones meilleurs pour la stratégie gloutonne se situent également vers le centre et les valeurs de faible *makespan*.

FIGURE 4.104: Instance 9 : Population cumulées pour tous les runs et à tout temps.

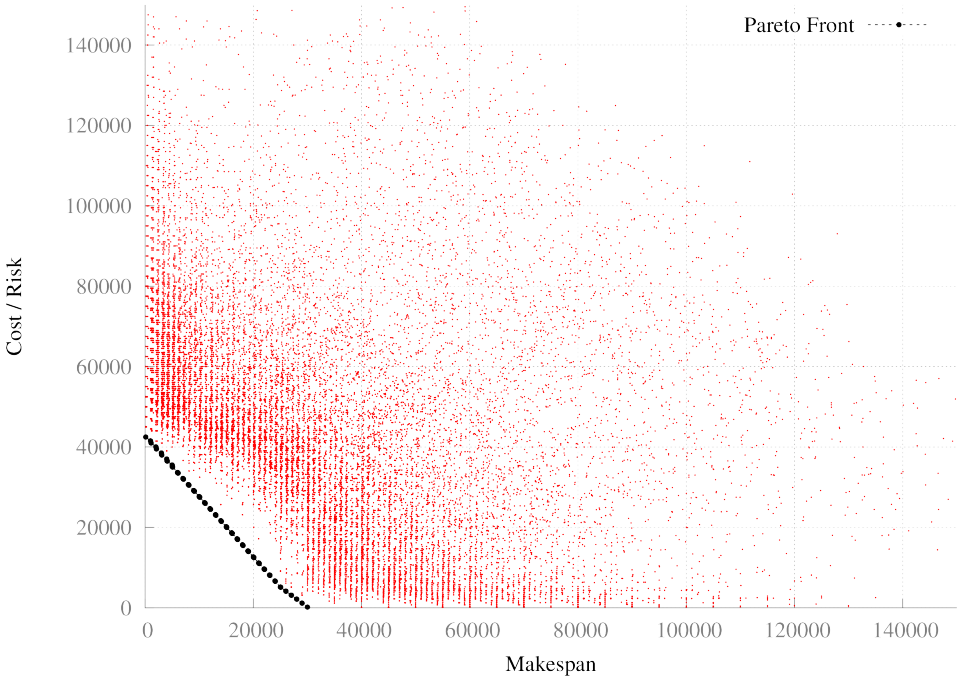


FIGURE 4.105: Instance 9 : Population cumulées pour tous les runs et à tout temps (version colorée).

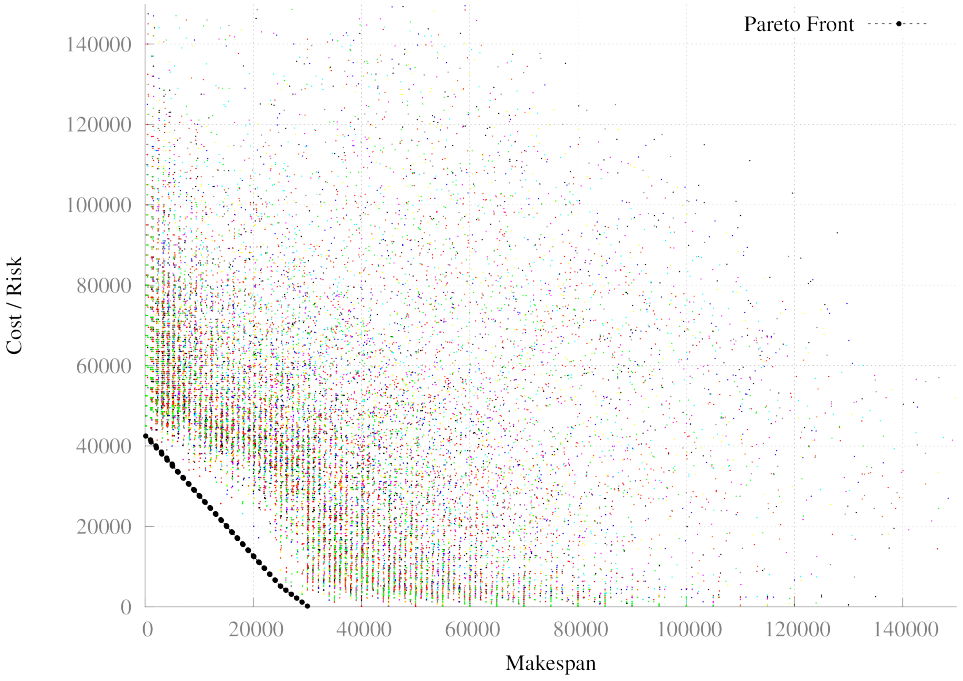
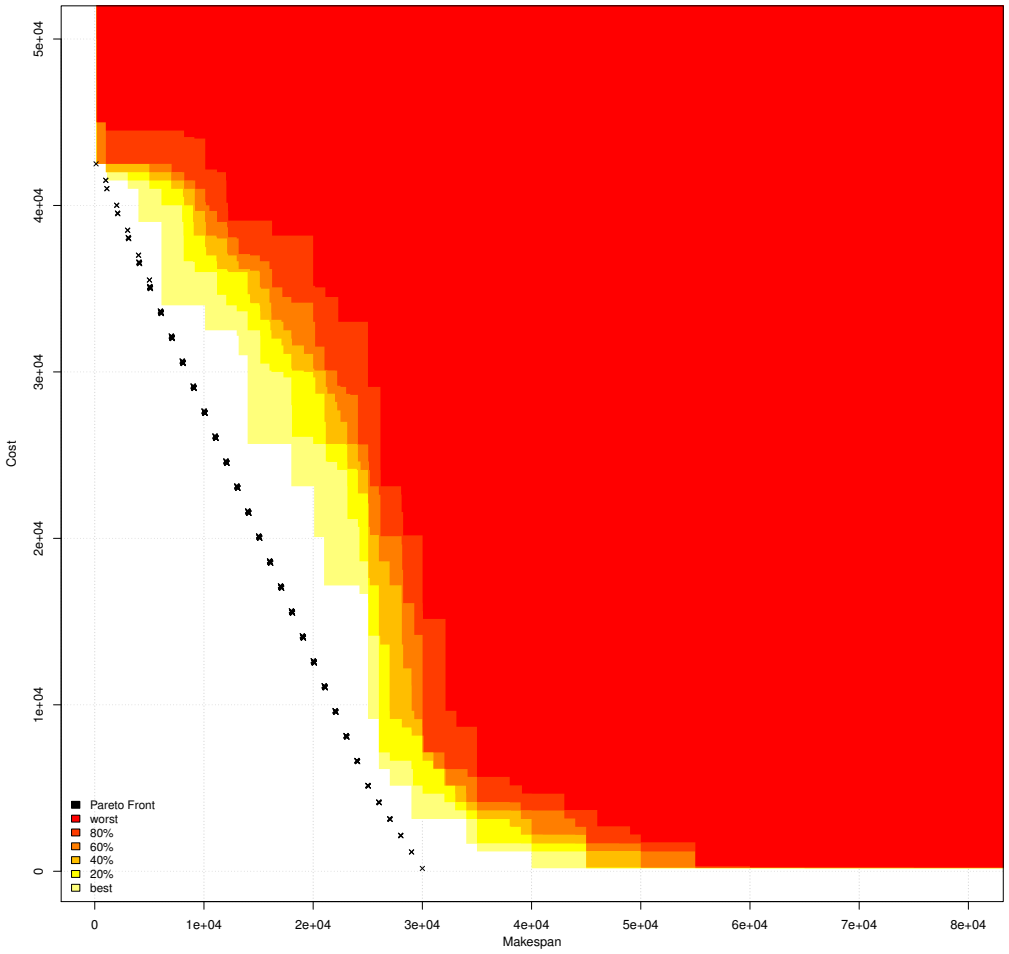


FIGURE 4.106: Instance 9 : Surfaces d'atteinte pour tous les runs.



La diversité des vecteurs objectifs semble, à l'instar de l'instance 4, suivre la même structure de manière retardée dans le temps.

FIGURE 4.107: Instance 9 : Trajectoires de l'hypervolume pour tous les runs.

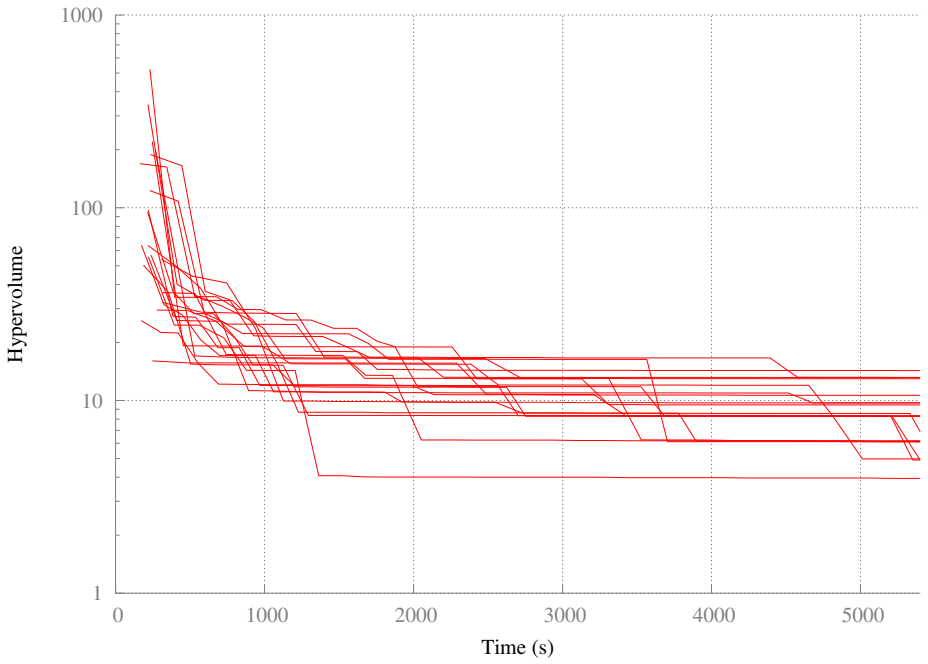


FIGURE 4.108: Instance 9 : Diversité des vecteurs objectifs pour tous les runs.

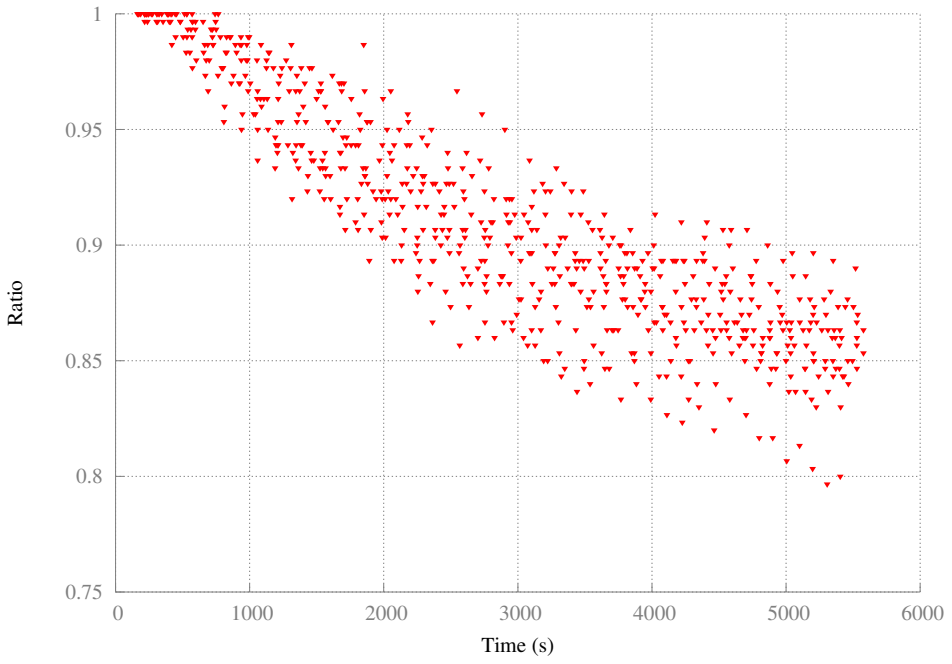


FIGURE 4.109: Instance 9 : Diversité des vecteurs objectifs pour tous les runs (version colorée).

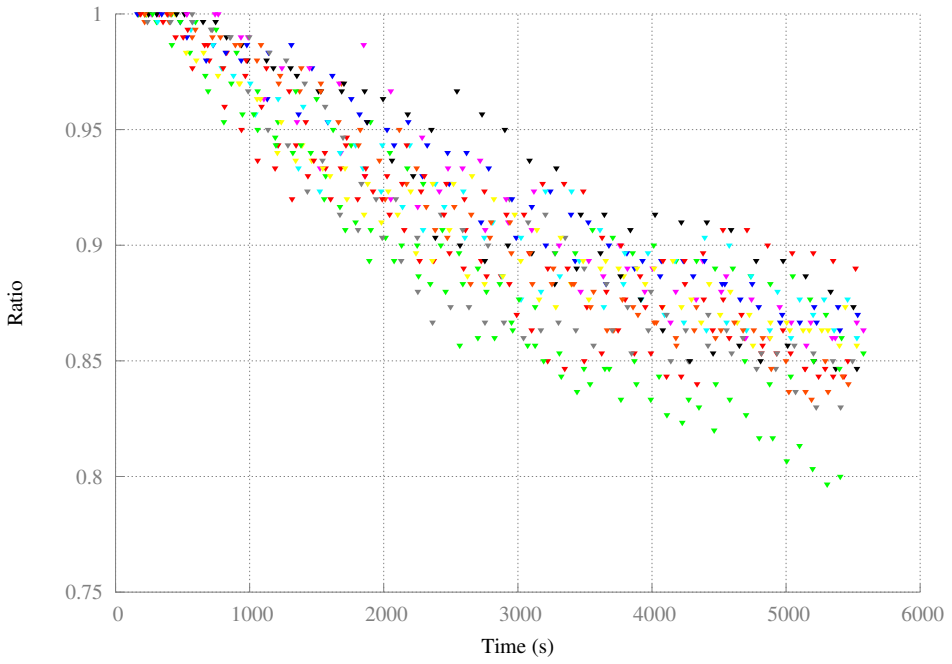


FIGURE 4.110: Instance 9 : Comparatif des surfaces d'atteinte.

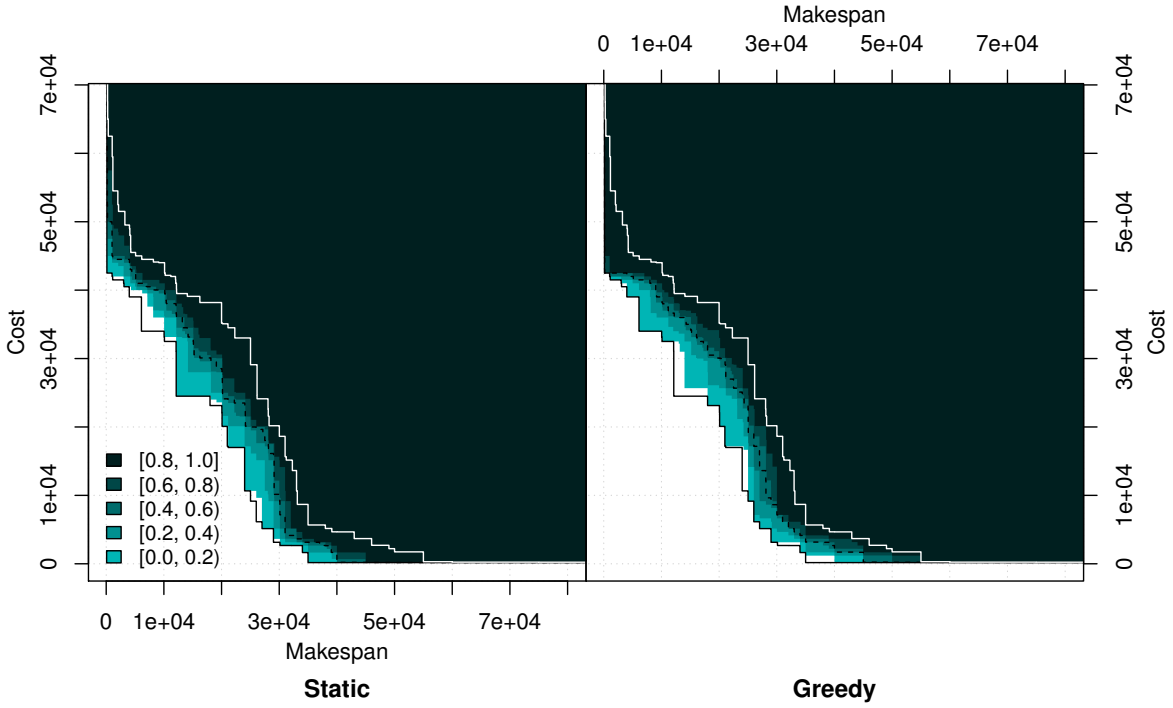


FIGURE 4.111: Instance 9 : Gloutonne comparée à Statique.

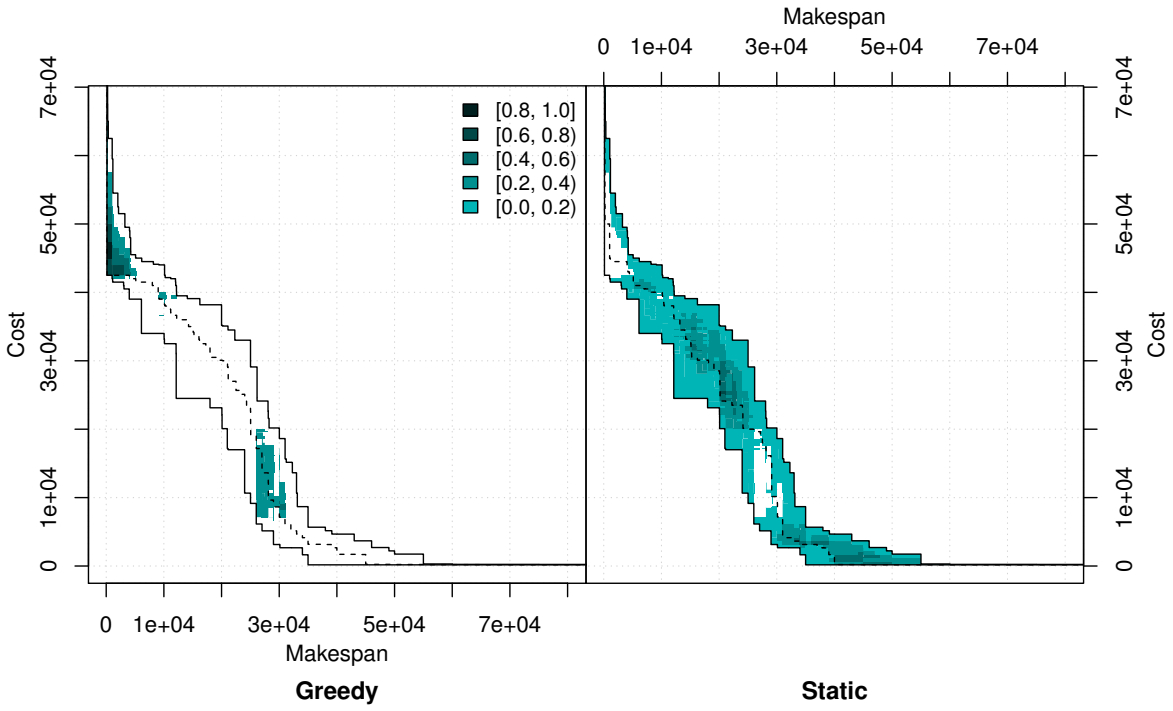


FIGURE 4.112: Instance 9 : Statique comparée à Gloutonne.

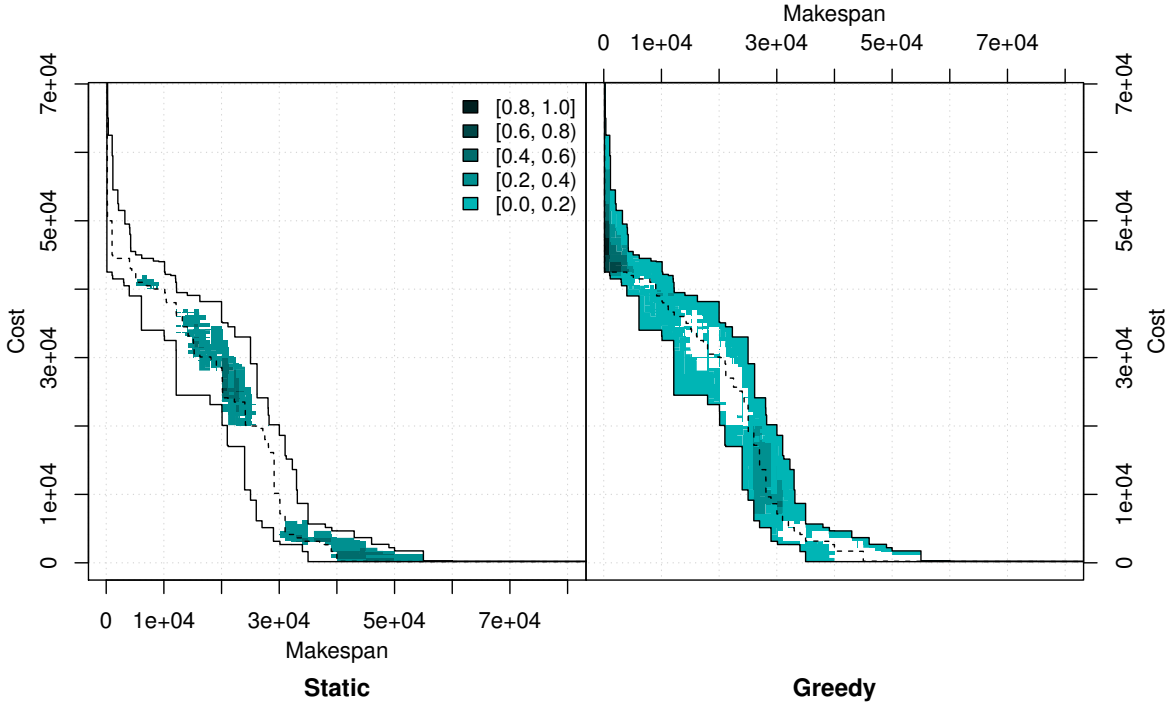


FIGURE 4.113: Instance 9 : Comparaison de l'hypervolume.

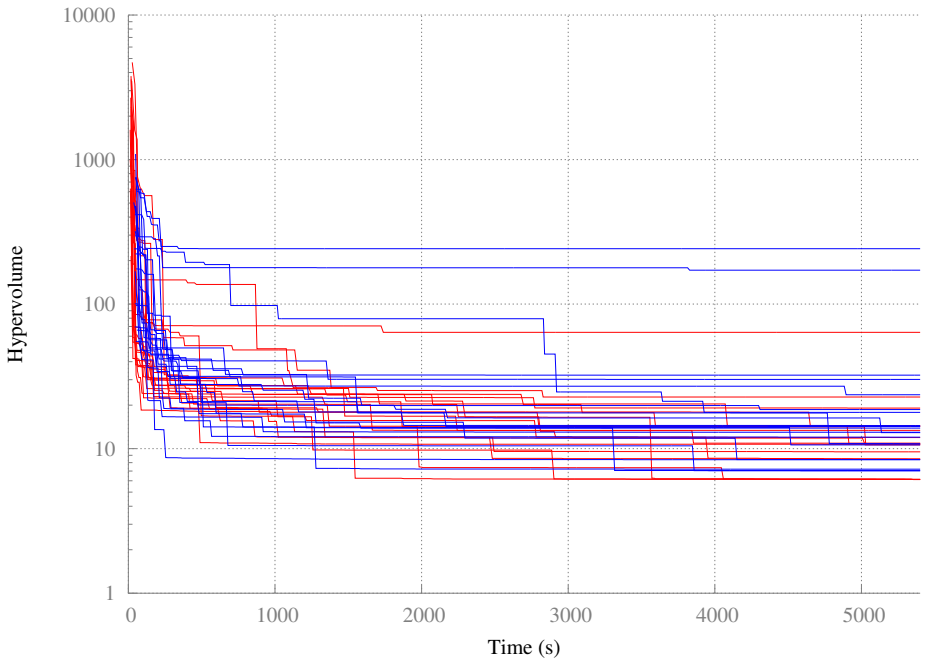
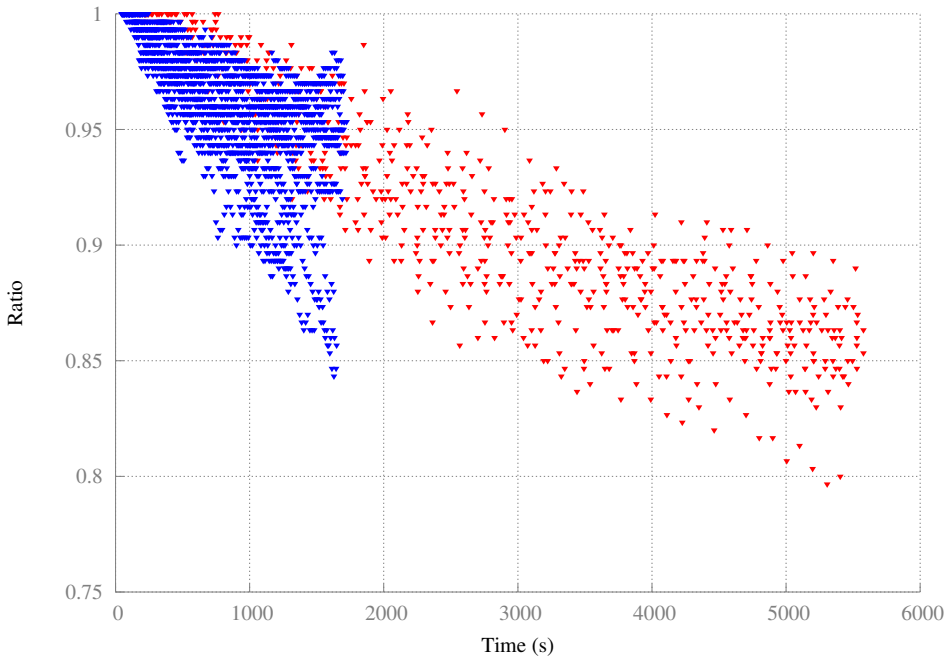


FIGURE 4.114: Instance 9 : Comparaison de la diversité des vecteurs objectifs.



4.6.1.3 Série 2

Paramètres Les valeurs de b_{max} ont été choisis comme la première puissance de 10 telle que l'archive finale n'était pas uniquement composée de plans infaisables.

TABLE 4.3: Paramètres trouvés par ParamILS pour la stratégie statique.

Paramètres	Instance 1	Instance 4	Instance 7	Instance 9
bmax-fixed	10	1000	100	100
popSize	30	30	30	30
proba-change	1	1	1	1
proba-cross	1	1	1	1
proba-del-atom	1	1	1	1
proba-mut	1	1	1	1
radius	10	10	10	10
w-addatom	7	7	7	10
w-addgoal	7	7	5	7
w-delatom	7	3	7	3
w-delgoal	0	3	0	3

Instance 1 Les fronts cumulés (figure 4.115) sont comparables à ceux de la stratégie gloutonne avec les paramètres originaux. On note cependant que les points extrêmes sont un peu plus éloignés. On retrouve cette constatation sur les surfaces d'atteinte de la figure 4.118 puisque le moins bon des *runs* se retrouve largement en retrait par rapport à la version avec paramètres originaux.

Ce qui est plus remarquable est que malgré des surfaces d'atteinte extrêmement similaires, la zone échantillonnée de l'espace objectif est considérablement réduite.

Les trajectoires de l'hypervolume (figure 4.119) sont également très intéressantes puisqu'elles présentent une première phase de diminution de l'hypervolume encore plus marquée que pour toutes les autres stratégies. Une diminution des ressources accordées à YAHSP lors d'un appel conduit à abandonner plus vite si un plan ne peut être trouvé. Réessayer plusieurs fois une résolution sur un même individu conduit à augmenter, jusqu'à une certaine limite, ses chances de trouver un plan. Il semblerait qu'il soit possible même avec des valeurs extrêmement faibles de b_{max} de trouver des plans faisables. Le compromis se situe dans le fait que plus les sous-problèmes sont difficiles à résoudre, plus il faudra un b_{max} important pour voir YAHSP les résoudre en un essai. En réduisant drastiquement b_{max}, on réduit également le temps par essai de YAHSPṠi l'on considère que DAE doit travailler à fournir des individus « faciles » au solveur local, b_{max} pourrait

FIGURE 4.115: Instance 1 : Fronts de Pareto cumulés pour tous les runs, à la fin de chaque run.

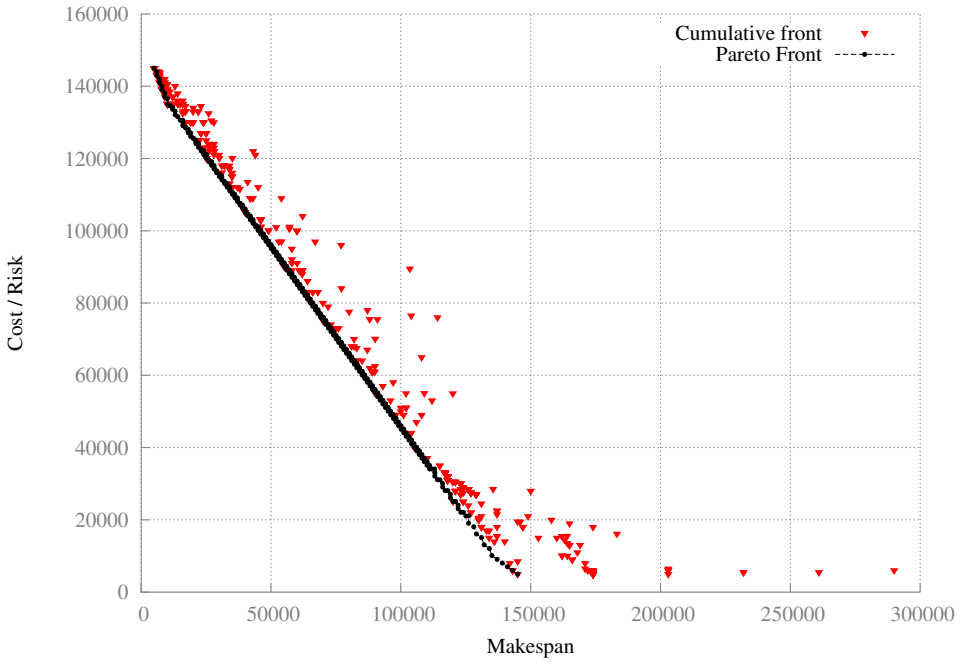


FIGURE 4.116: Instance 1 : Population cumulées pour tous les runs et à tout temps.

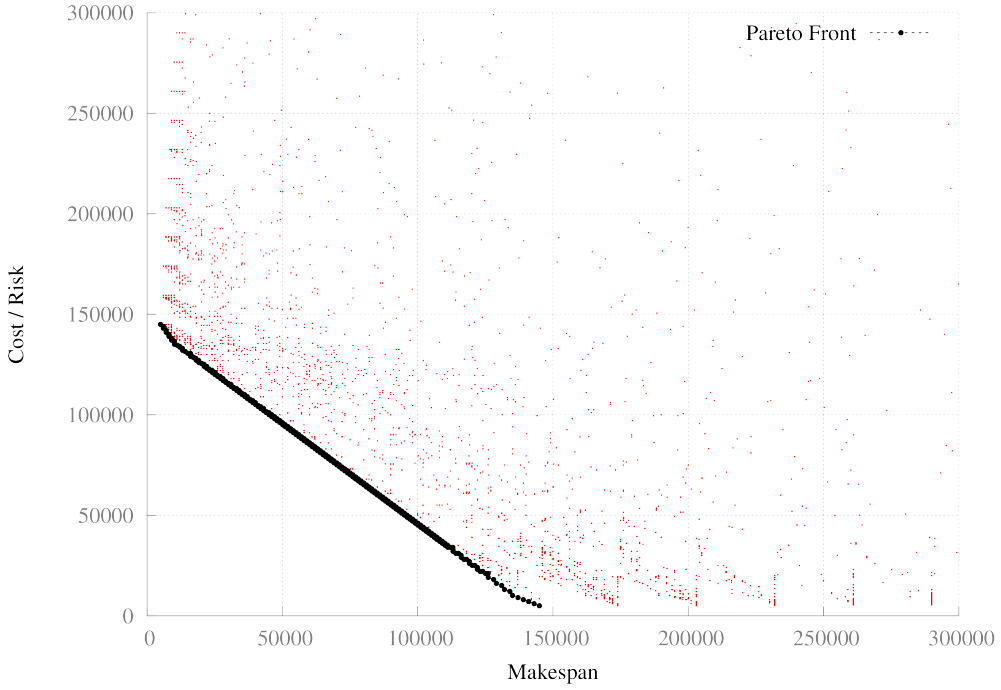
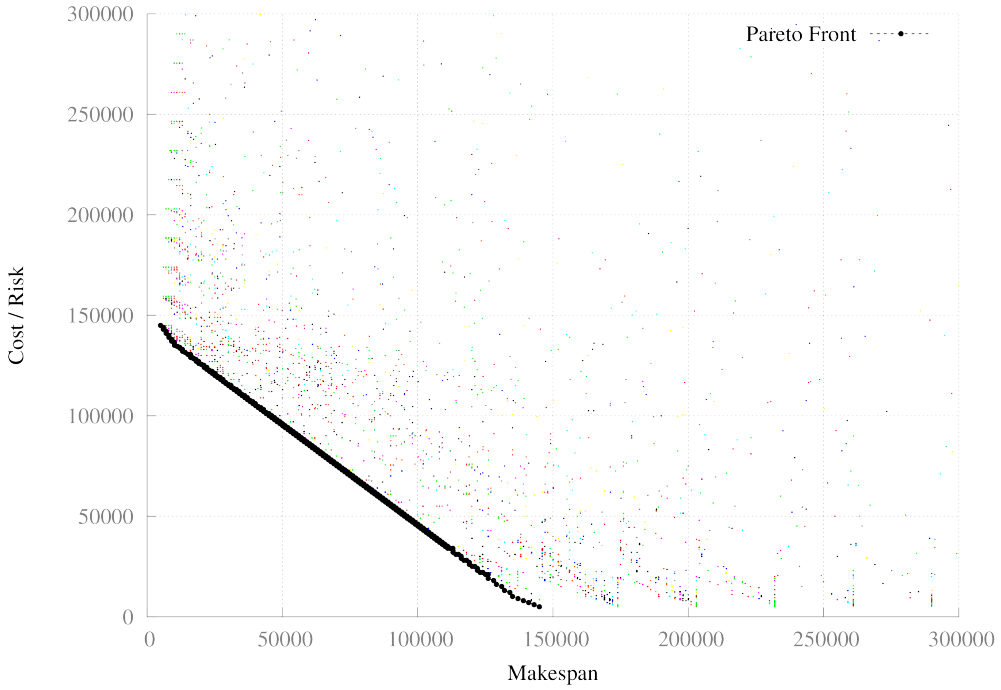


FIGURE 4.117: Instance 1 : Population cumulées pour tous les runs et à tout temps (version colorée).

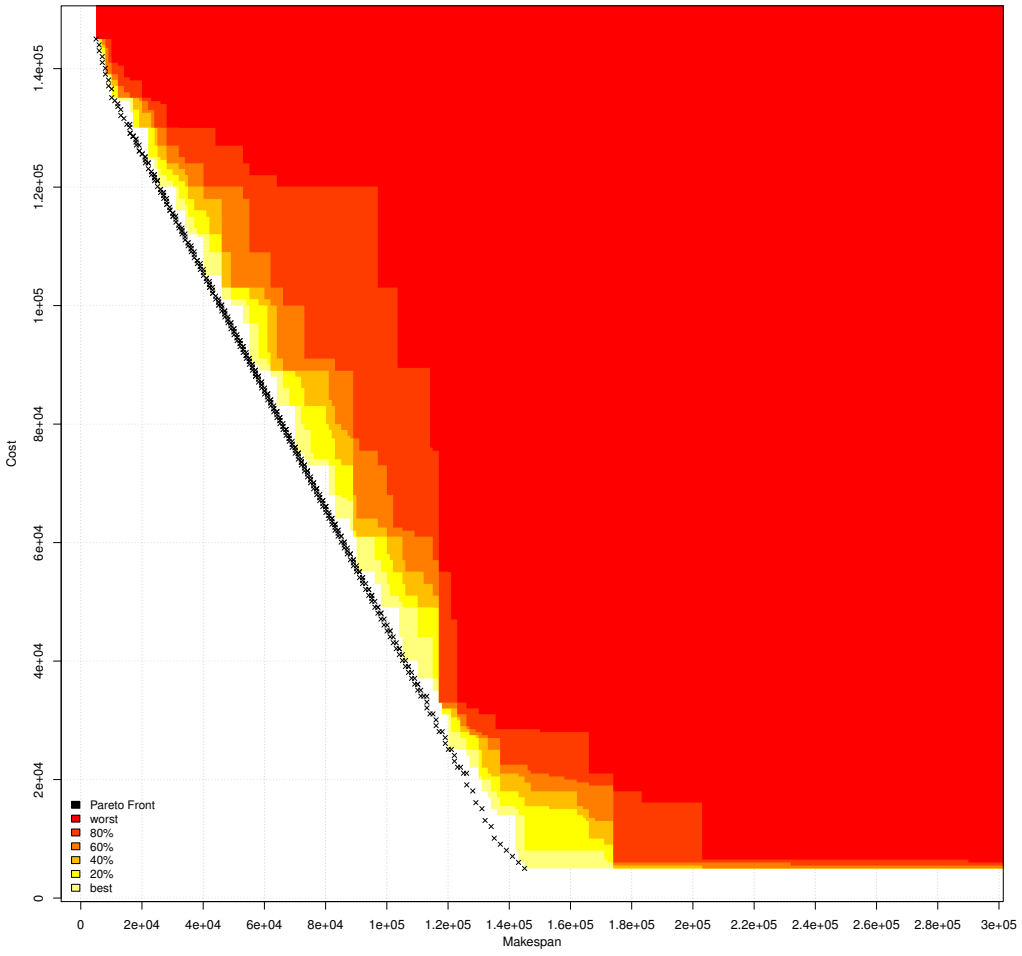


être considéré comme un bon indicateur de la difficulté que représente à individu comme nous en discuterons dans la section 4.8 qui traite de l'évaluation.

La diversité des vectors objectifs semble suivre la même structure que pour les paramètres originaux. On constate simplement une amplitude plus grande entre les différents *runs*.

Comparaison avec la stratégie statique Si l'on est encore une fois moins bon que la stratégie statique, il est a noté une légère amélioration par rapport à la version avec les paramètre originaux, que l'on peut expliquer par une plus grand diversité très certainement.

FIGURE 4.118: Instance 1 : Surfaces d'atteinte pour tous les runs.



Instance 4 De même que pour l'instance 1, l'espace échantillonné est moins dense et concentré sur les extrémités. Les trajectoires de l'hypervolume retrouvent la première phase décroissante et la diversité reste tout le long du processus proche de 1. Ce dernier résultat était prévisible par les paramètres de diversité qui ont été largement augmenté.

Comparaison avec la stratégie statique On observe cependant des résultats moins bons qu'avec les paramètres originaux et donc, qu'avec la stratégie statique, probablement signe que la valeur de b_{max} n'est pas assez importante pour permettre de trouver des plans faisables de meilleurs qualités.

FIGURE 4.119: Instance 1 : Trajectoires de l'hypervolume pour tous les runs.

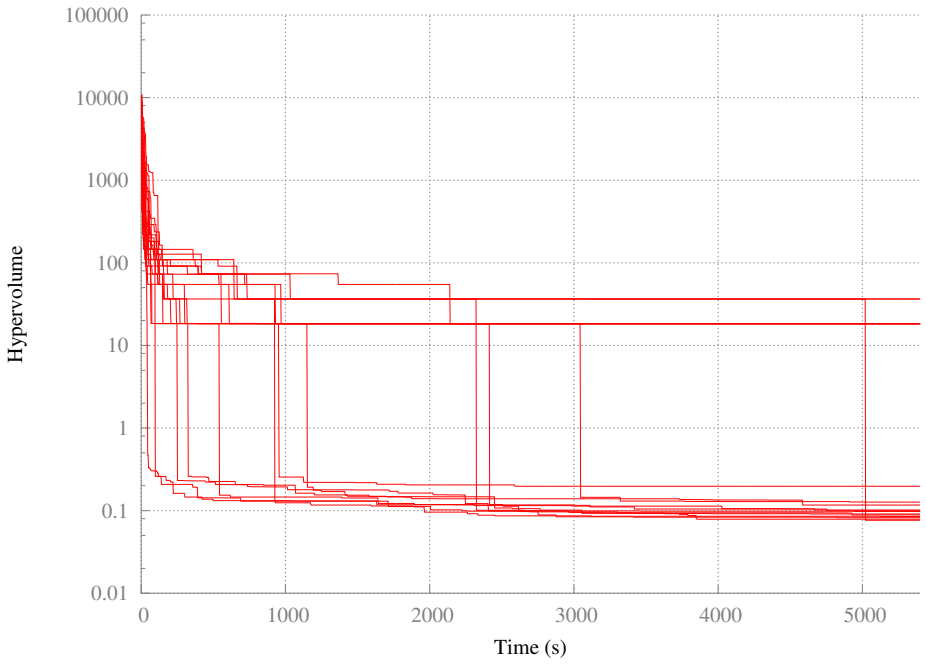


FIGURE 4.120: Instance 1 : Diversité des vecteurs objectifs pour tous les runs.

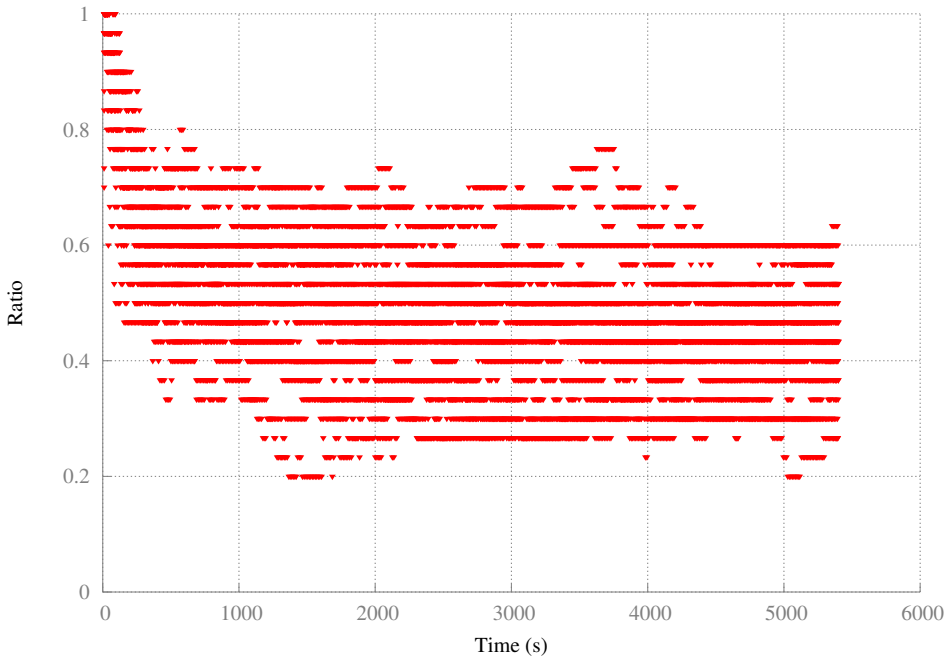


FIGURE 4.121: Instance 1 : Diversité des vecteurs objectifs pour tous les runs (version colorée).

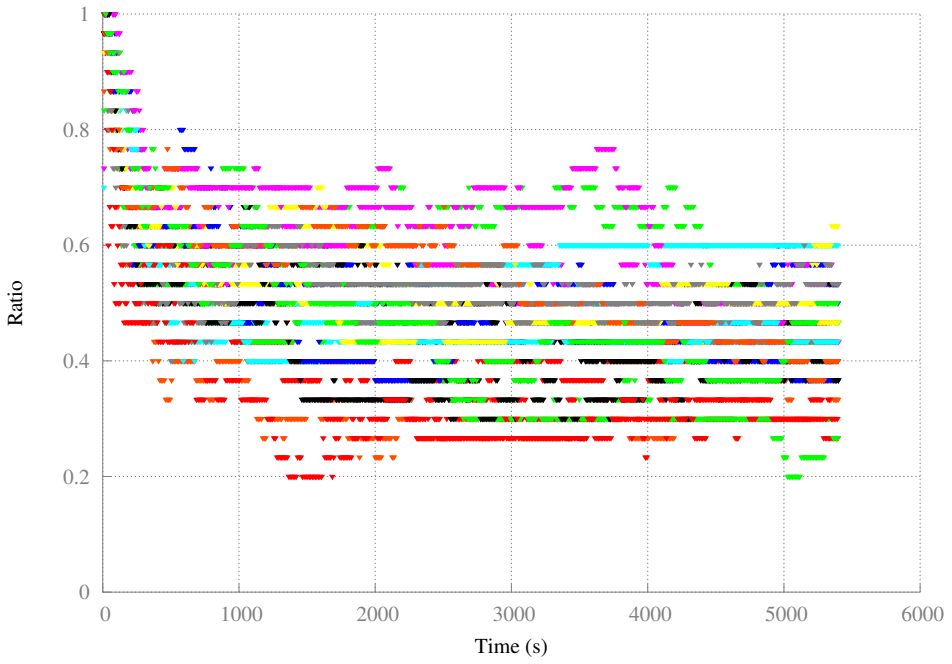


FIGURE 4.122: Instance 1 : Comparatif des surfaces d'atteinte.

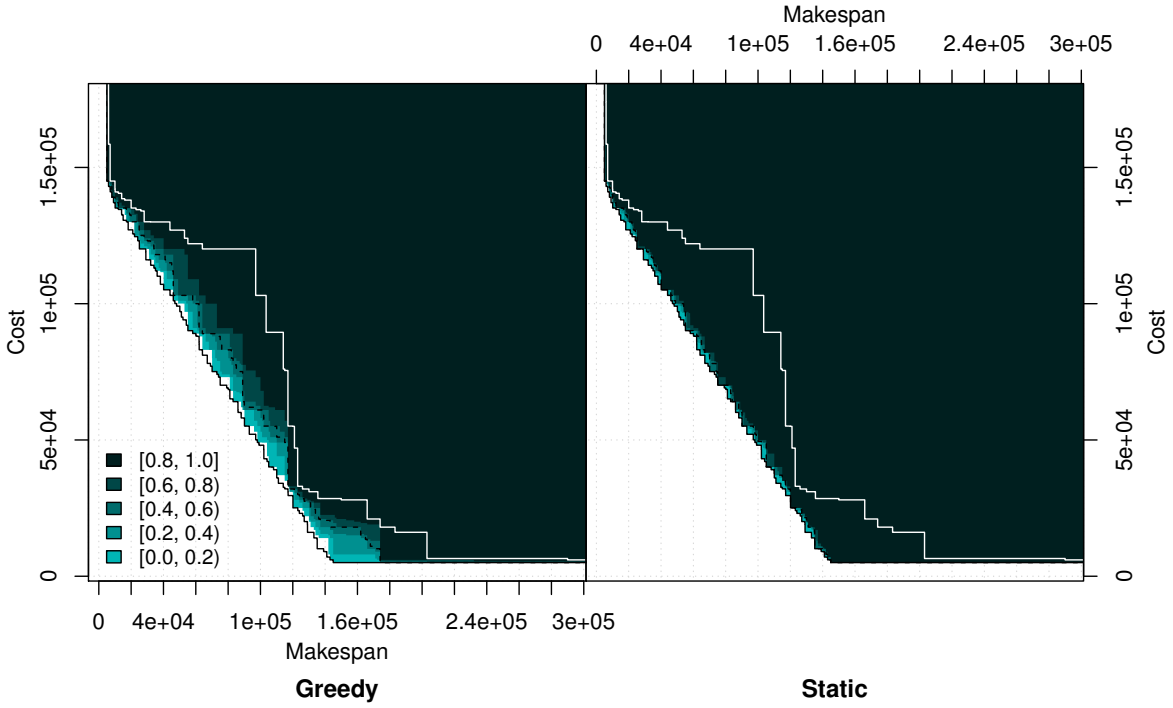


FIGURE 4.123: Instance 1 : Gloutonne comparée à Statique.

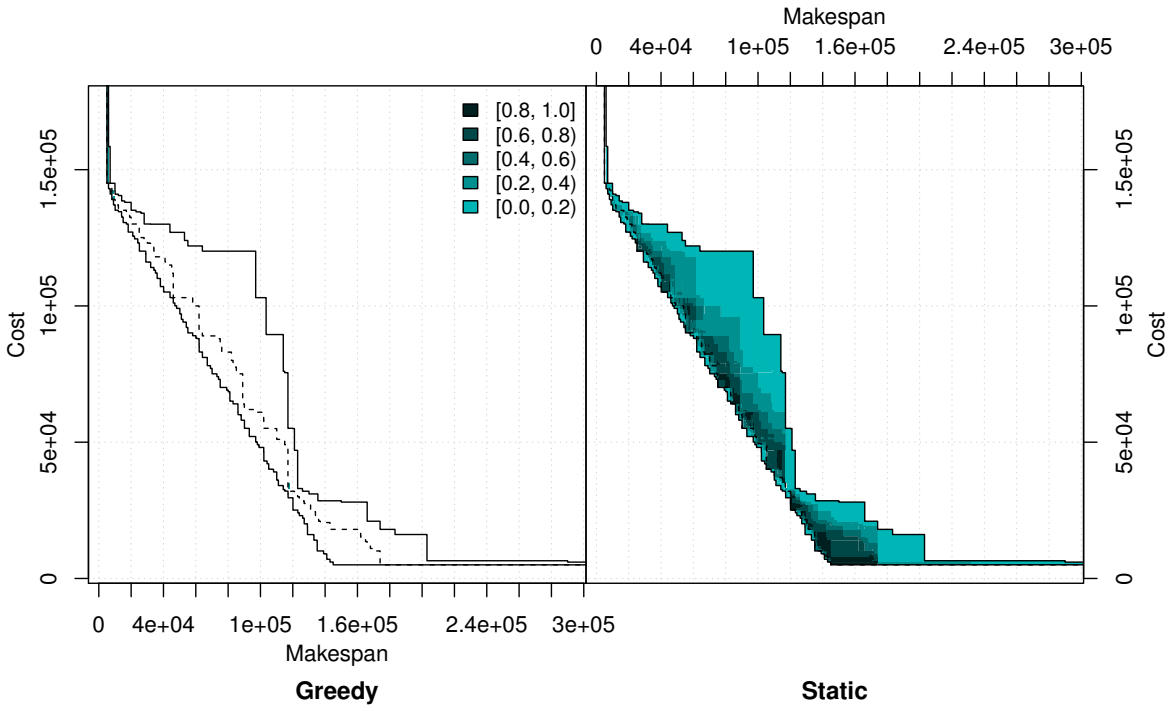


FIGURE 4.124: Instance 1 : Statique comparée à Gloutonne.

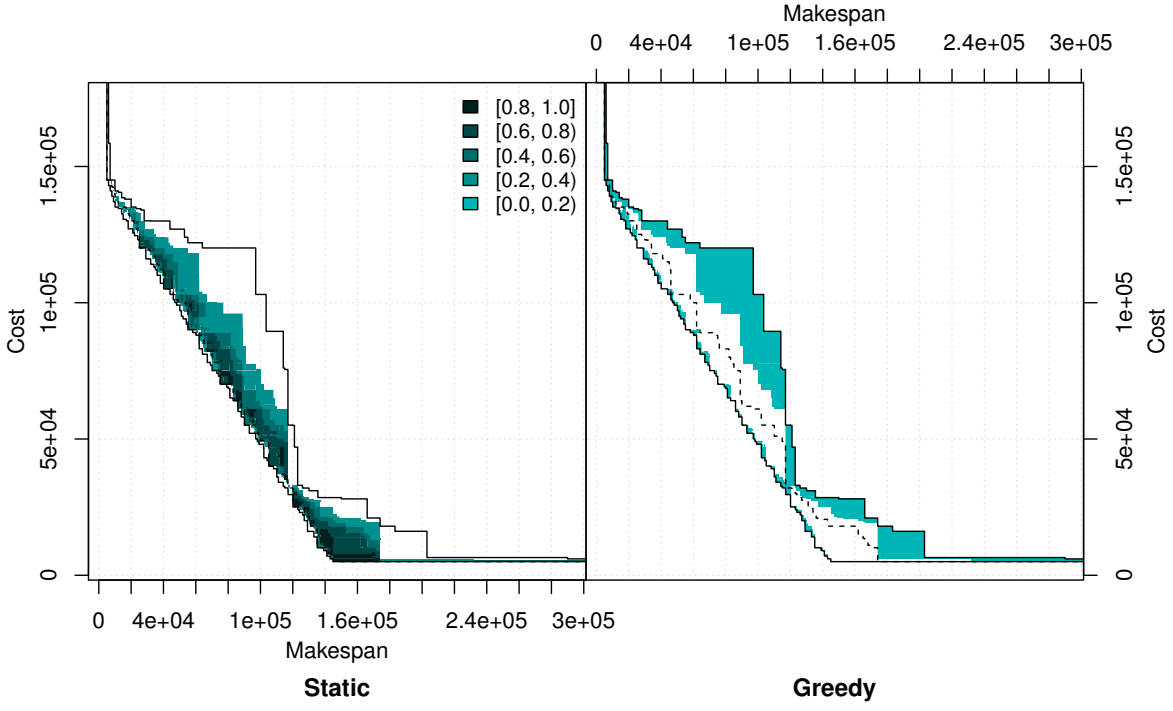


FIGURE 4.125: Instance 4 : Fronts de Pareto cumulés pour tous les runs, à la fin de chaque run.

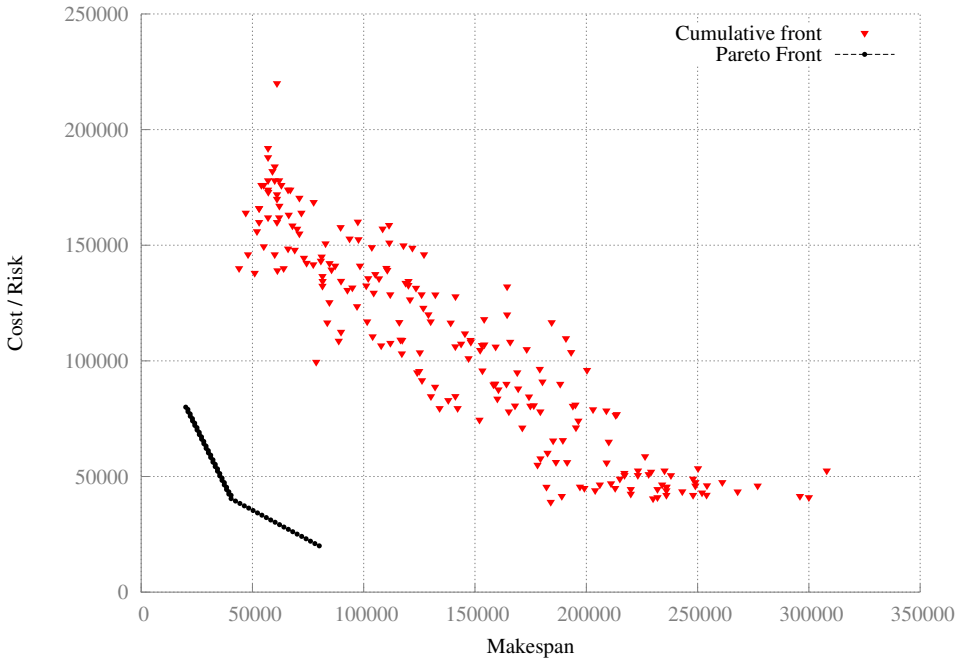


FIGURE 4.126: Instance 4 : Population cumulées pour tous les runs et à tout temps.

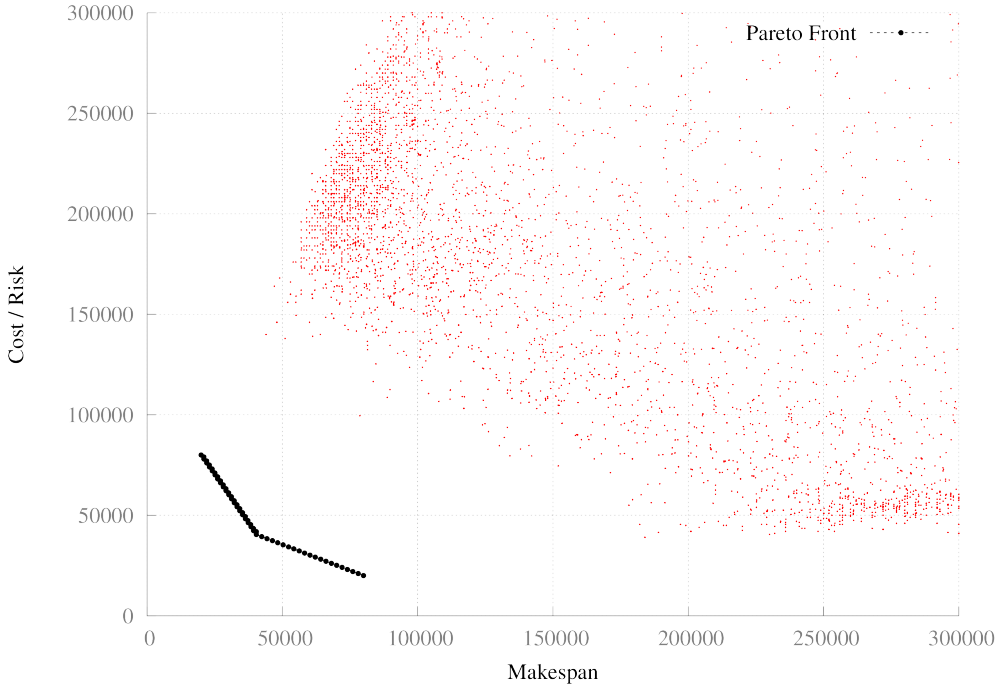
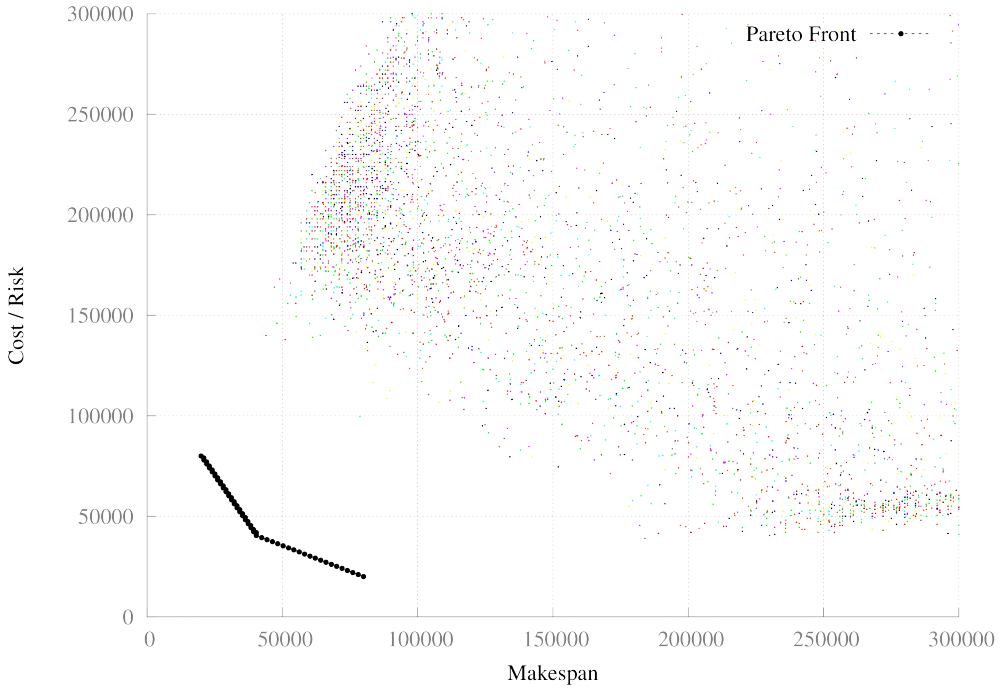
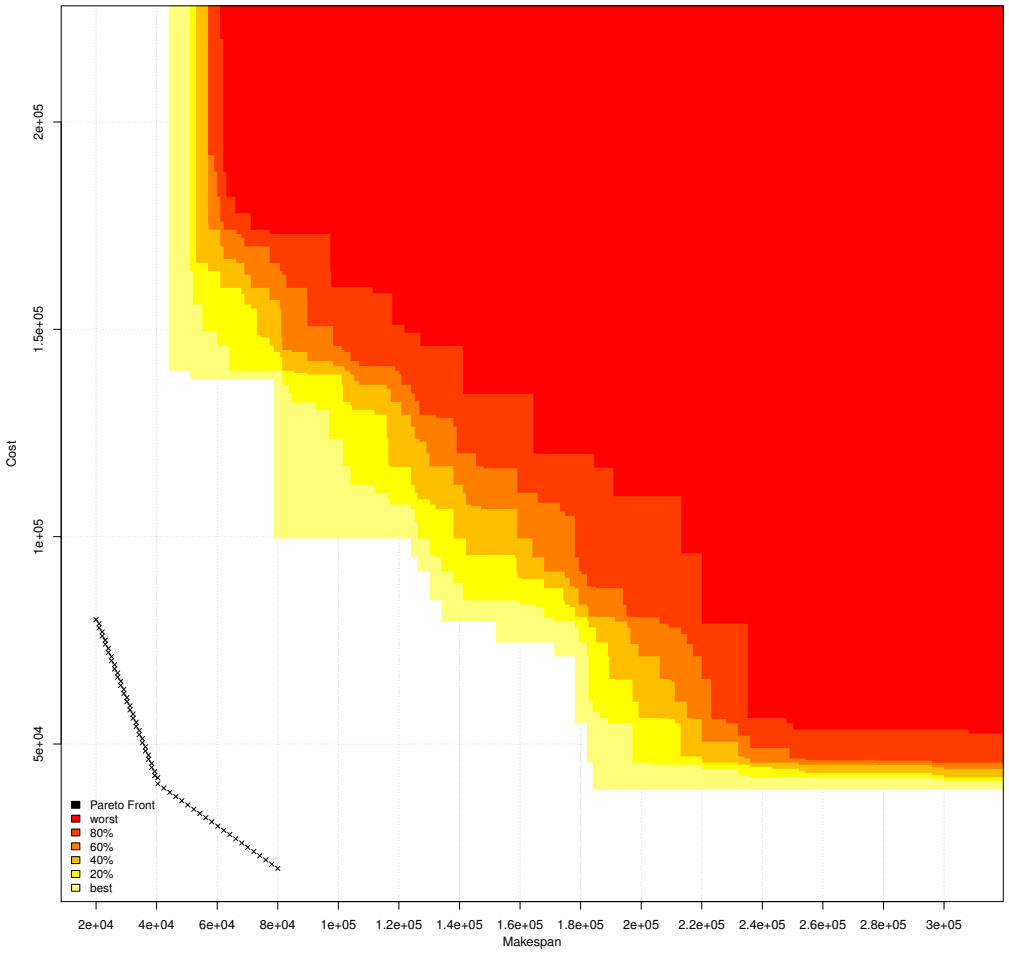


FIGURE 4.127: Instance 4 : Population cumulées pour tous les runs et à tout temps (version colorée).



Instance 7 L'instance 7 révèle un comportement intéressant que l'on peut certainement imputer au paramètre b_{max}. Certaines zones de l'espace objectif restent inaccessibles et ce de manière très significative comme en atteste la figure 4.135 ou 4.138.

FIGURE 4.128: Instance 4 : Surfaces d'atteinte pour tous les runs.



Instance 9 Une nouvelle fois, la zone échantillonnée est beaucoup moins dense qu'avec les paramètres originaux et les surfaces d'atteinte un peu moins bonnes également. Cependant, les courbes d'hypervolume présentent l'intéressante particularité d'avoir une première phase de diminution plus brutale avec un b_{max} faible qu'avec un b_{max} important. Là encore on retrouve une très grande diversité qui ne semble pas diminuer, ou très légèrement.

Comparaison avec la stratégie statique

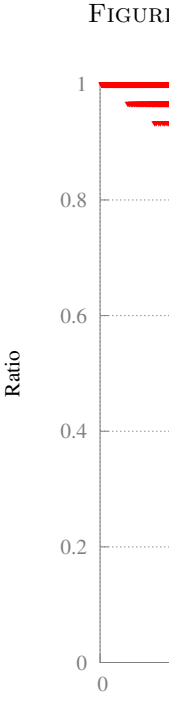


FIGURE 4.129: Instance 4 : Trajectoires de l'hypervolume pour tous les runs.

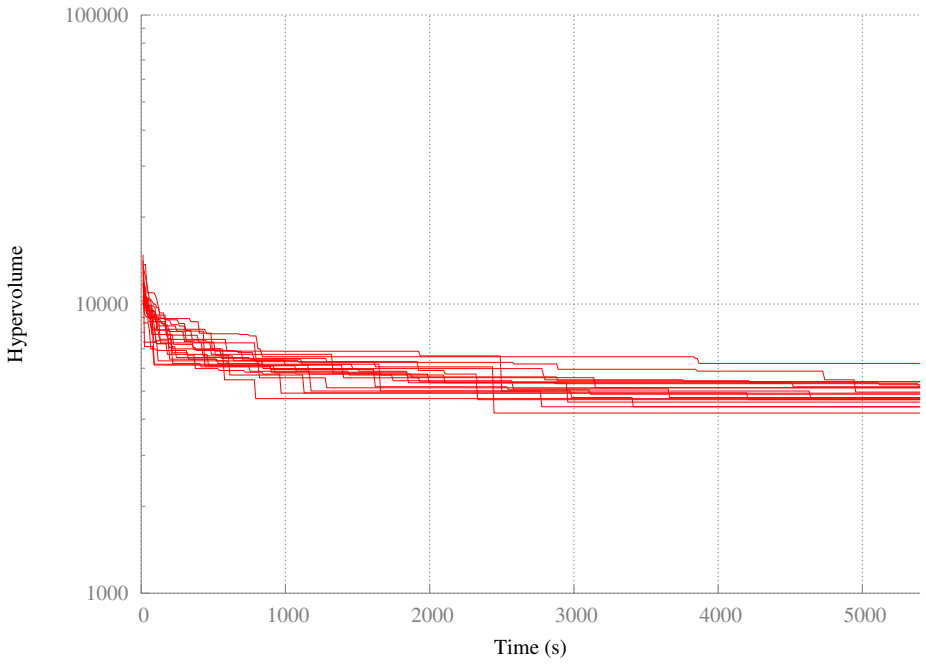


FIGURE 4.130: Instance 4 : Diversité des vecteurs objectifs pour tous les runs.

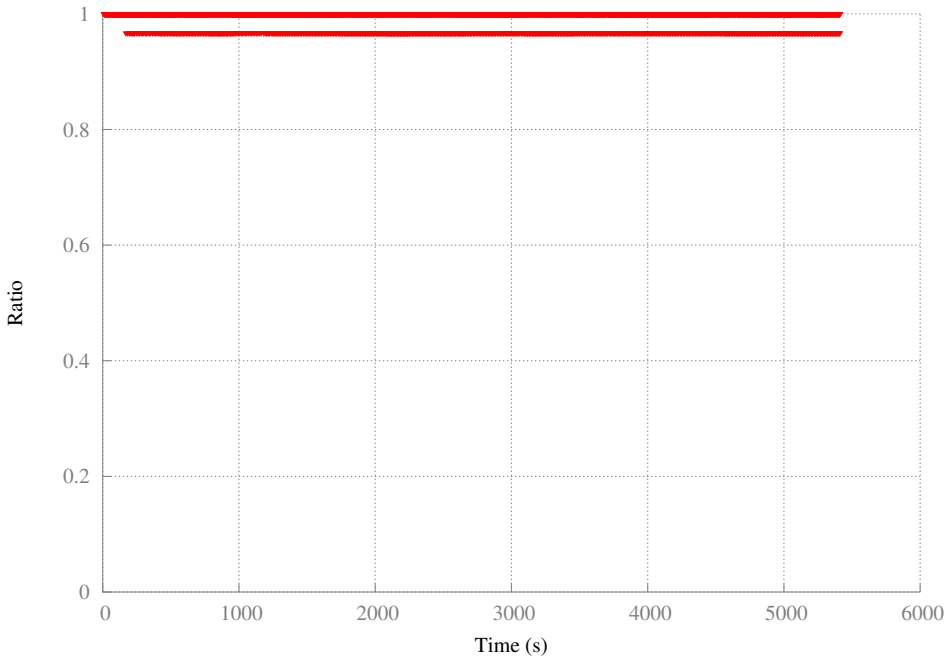


FIGURE 4.131: Instance 4 : Diversité des vecteurs objectifs pour tous les runs (version colorée).

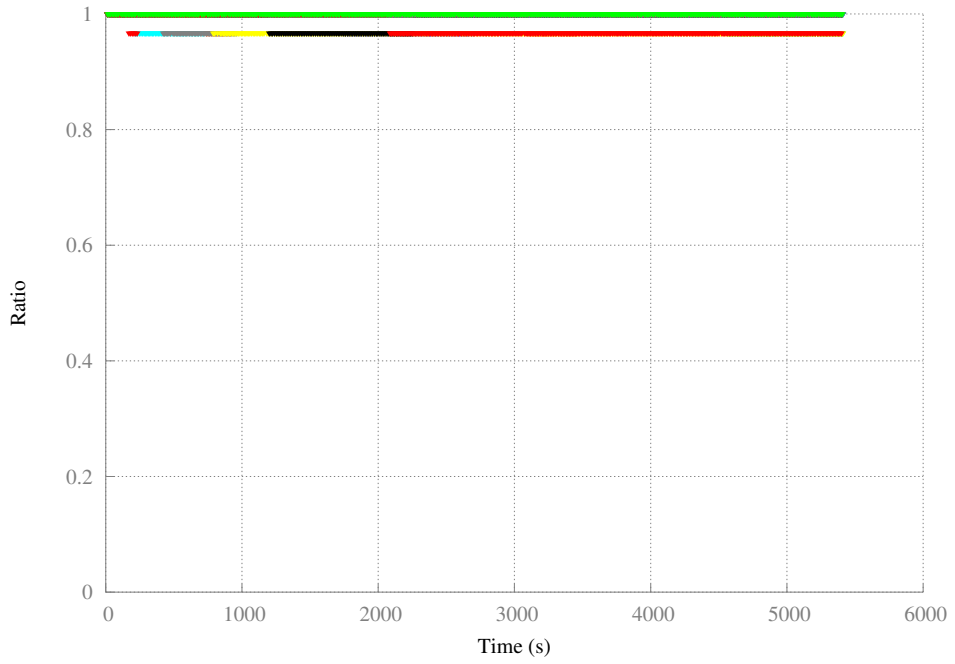


FIGURE 4.132: Instance 4 : Comparatif des surfaces d'atteinte.

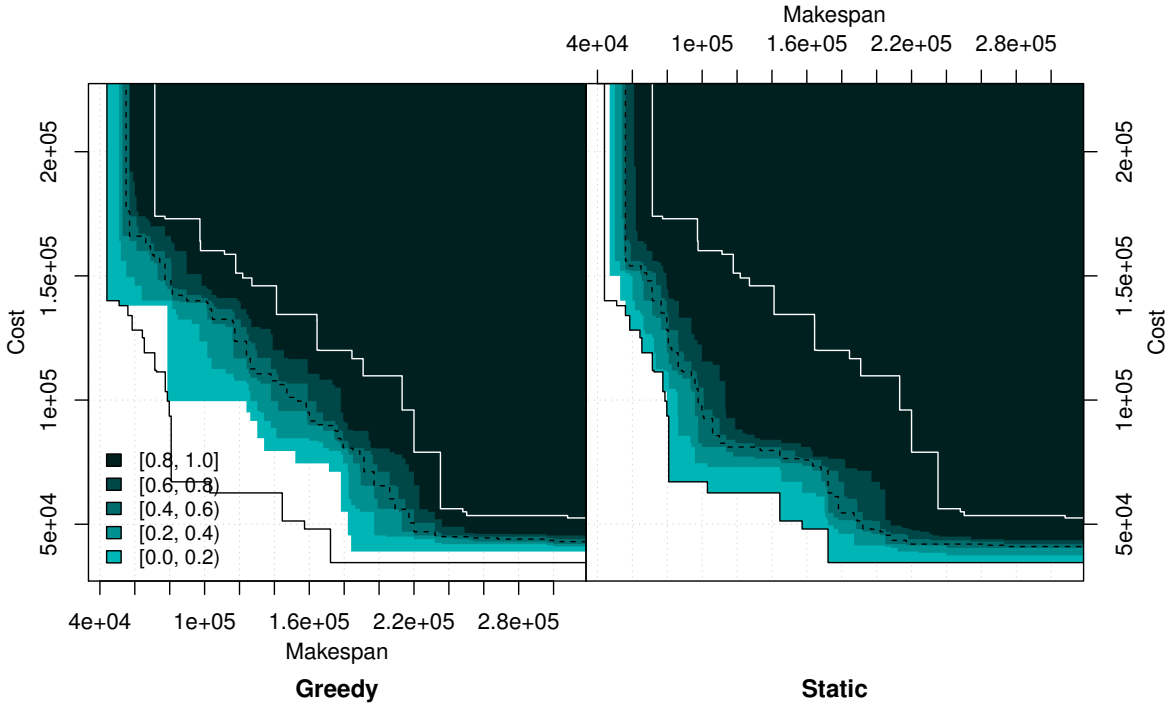


FIGURE 4.133: Instance 4 : Gloutonne comparée à Statique.

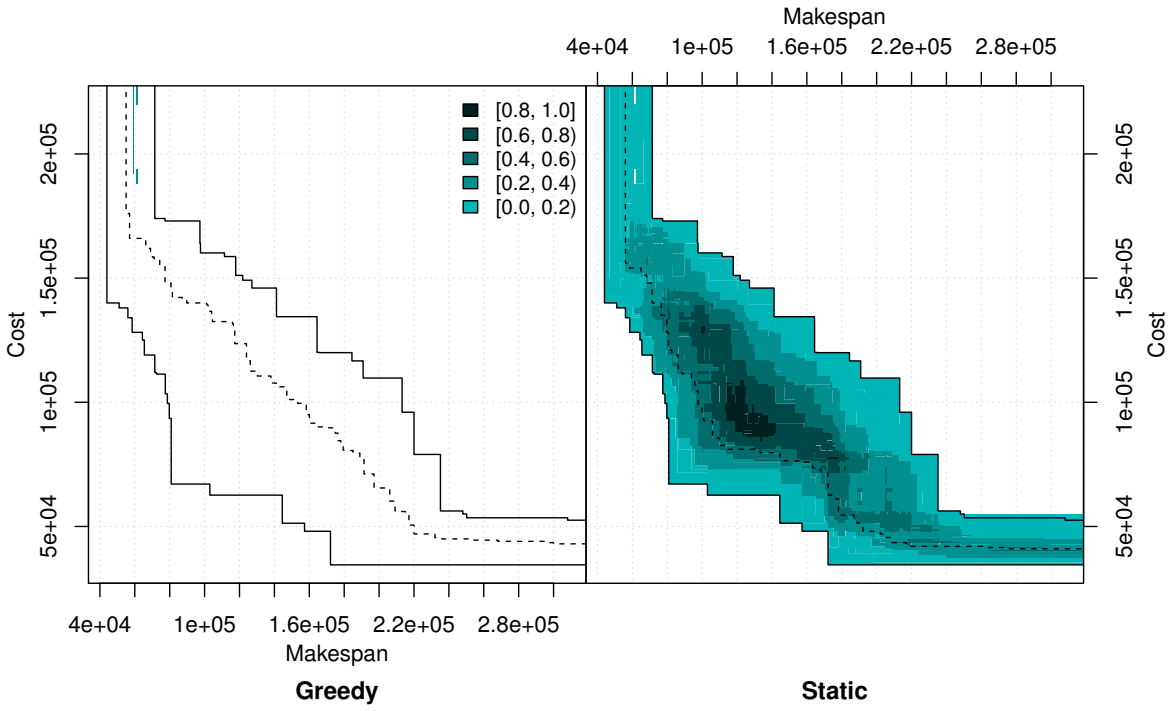


FIGURE 4.134: Instance 4 : Statique comparée à Gloutonne.

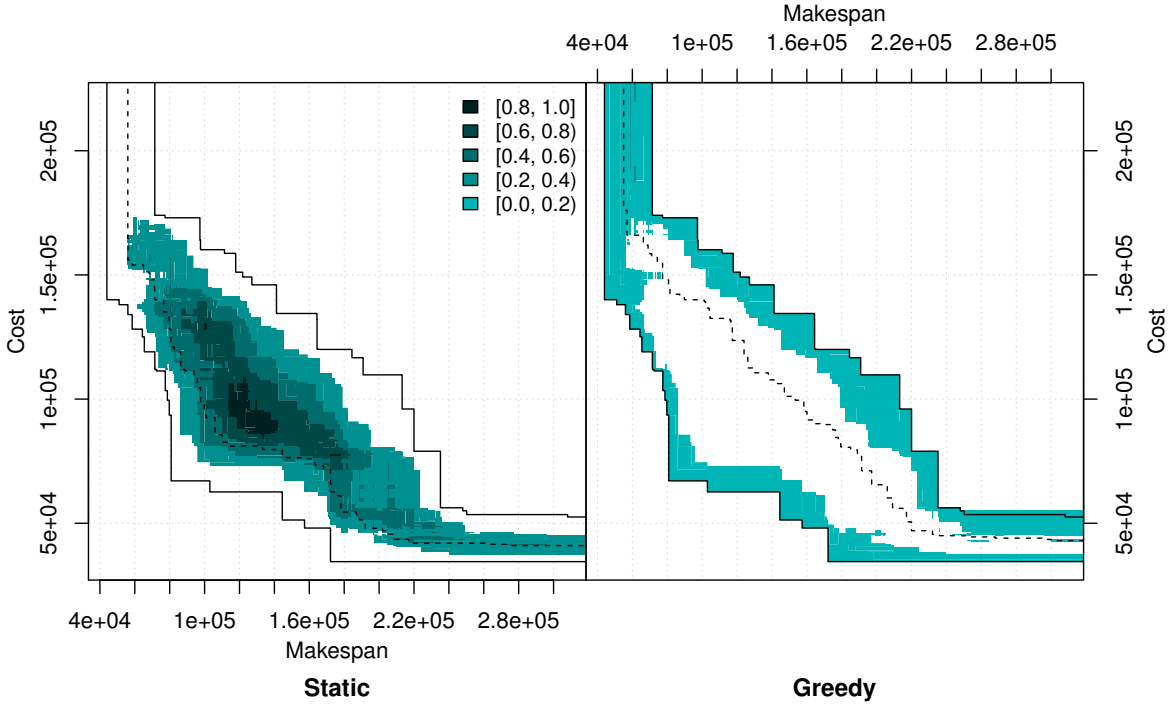


FIGURE 4.135: Instance 7 : Fronts de Pareto cumulés pour tous les runs, à la fin de chaque run.

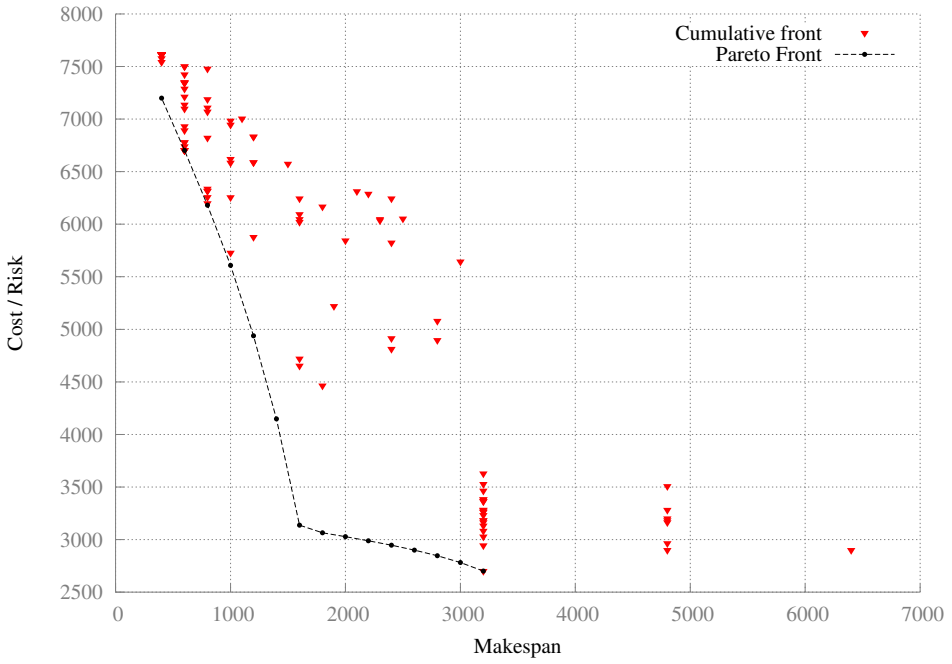


FIGURE 4.136: Instance 7 : Population cumulées pour tous les runs et à tout temps.

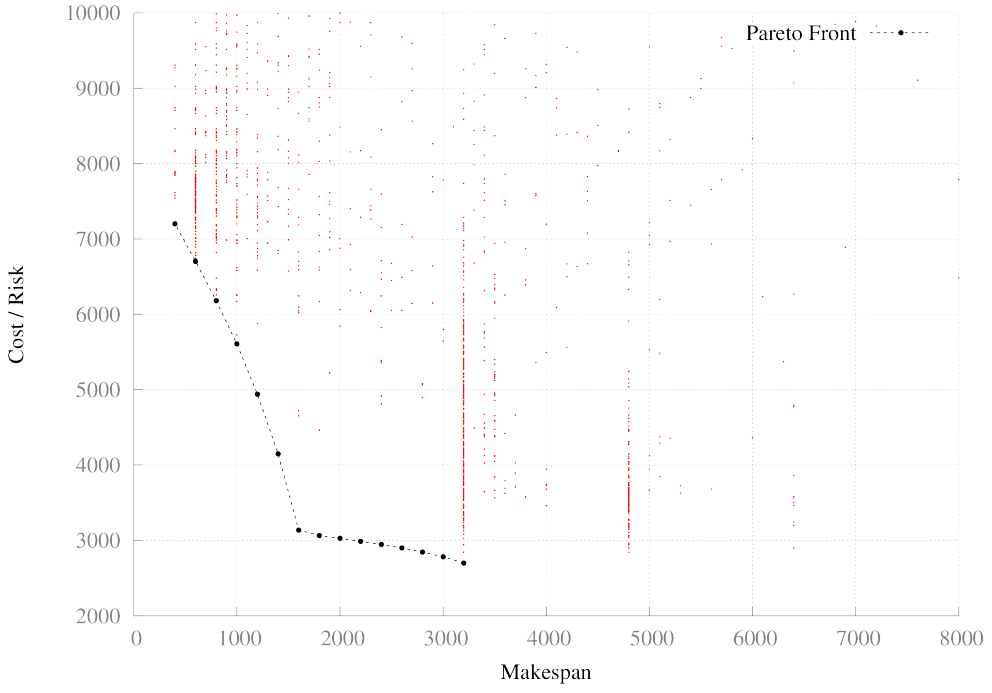
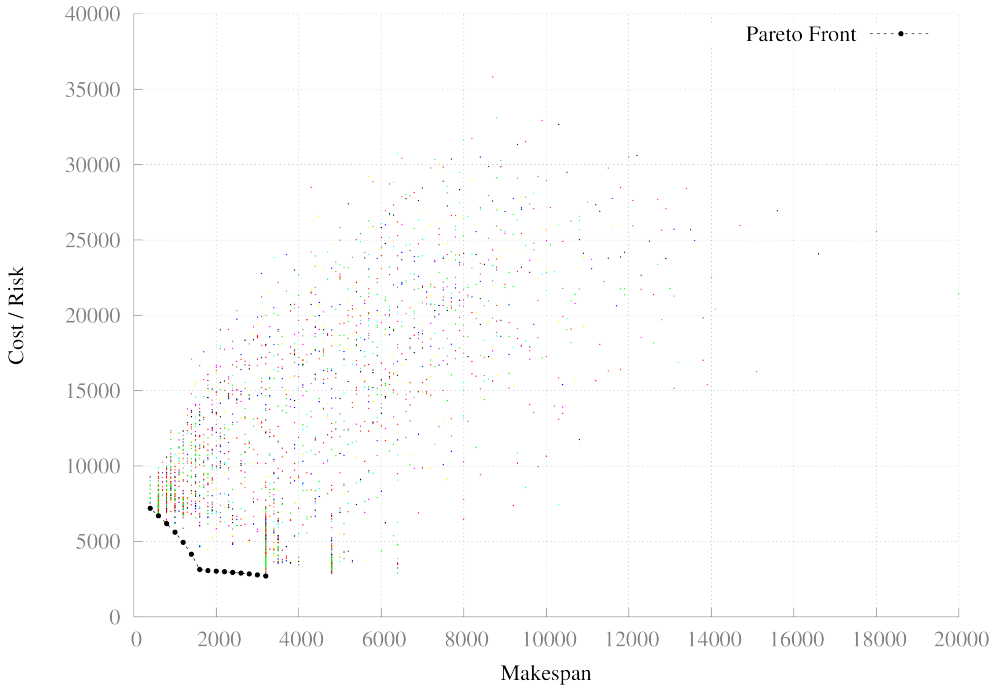


FIGURE 4.137: Instance 7 : Population cumulées pour tous les runs et à tout temps (version colorée).



4.6.2 Le cas de Zeno9

4.6.2.1 Protocole

Trois variantes de la stratégie gloutonne ont été testé : la version basique décrite en introduction, la version avec une sélection stricte qui consiste à sélection un plan s'il appartient à l'archive courante sans rappeler YAHSP sur les autres objectifs, et l'utilisation de l'amélioration relative plutôt que l'amélioration absolue.

Comme précédemment, pour chaque version 20 runs ont été réalisé. Chaque run, d'une durée de 600 secondes, a été lancé sur une machine équipée d'un `Intel(R) Core(TM) i3 CPU M 380 @ 1`

Les paramètres sont les suivants :

La stratégie statique utilise les paramètres trouvés précédemment sur cette instance sur une machine similaire. La stratégie gloutonne bénéficie d'un b_{max} très réduit et de paramètres favorisant la diversité à l'extrême et au contraire, une population largement augmentée. Il s'agit d'un réglage empirique motivé par les résultats précédents. Il est

FIGURE 4.138: Instance 7 : Surfaces d'atteinte pour tous les runs.

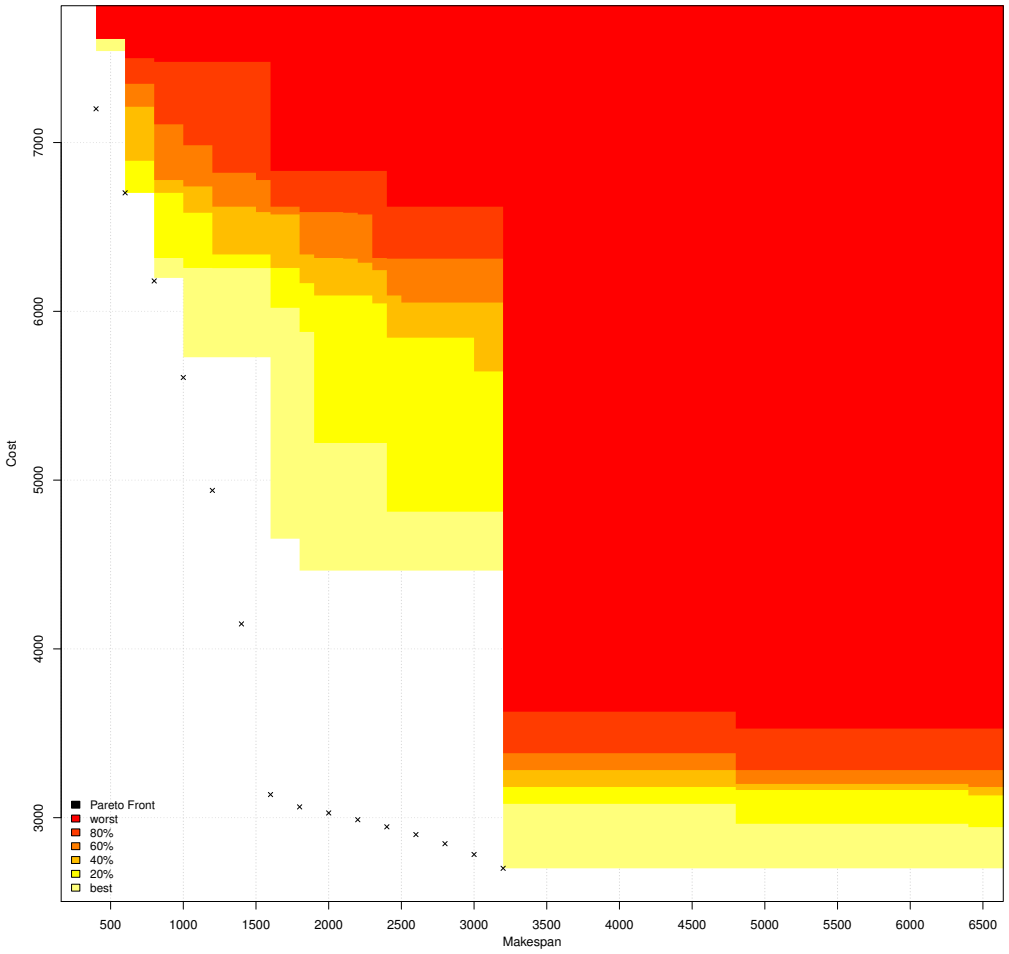


TABLE 4.4: Paramètres utilisés pour l'instance Zeno9

Paramètres	Stratégie statique	Stratégie gloutonne
lenght_weigth	0	-
cost_weigth	1	-
makespan_max_weigth	0	-
makespan_add_weigth	1	-
bmax-fixed	1000	100
popSize	30	300
proba-change	0.8	1
proba-cross	0.2	1
proba-del-atom	0.5	1
proba-mut	0.8	1
radius	3	10
w-addatom	1	5
w-addgoal	1	5
w-delatom	3	5
w-delgoal	1	5

FIGURE 4.139: Instance 7 : Trajectoires de l'hypervolume pour tous les runs.

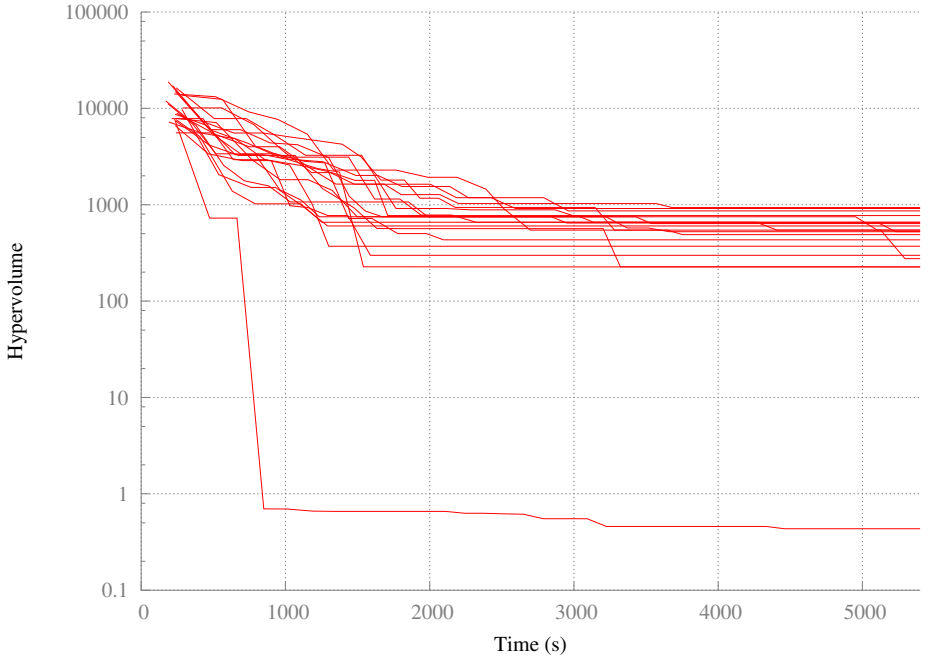


FIGURE 4.140: Instance 7 : Diversité des vecteurs objectifs pour tous les runs.

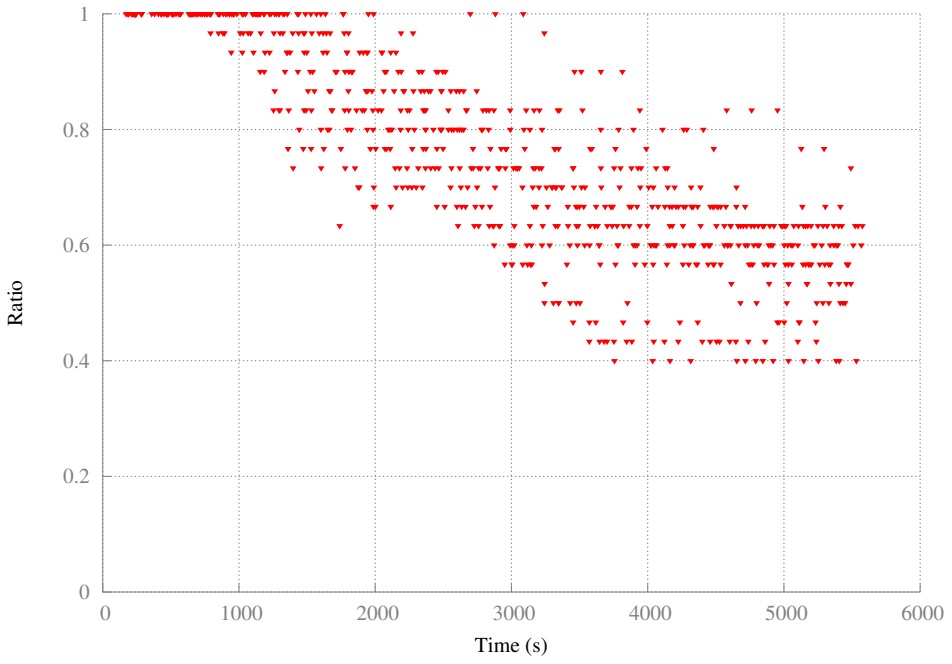


FIGURE 4.141: Instance 9 : Diversité des vecteurs objectifs pour tous les runs (version colorée).

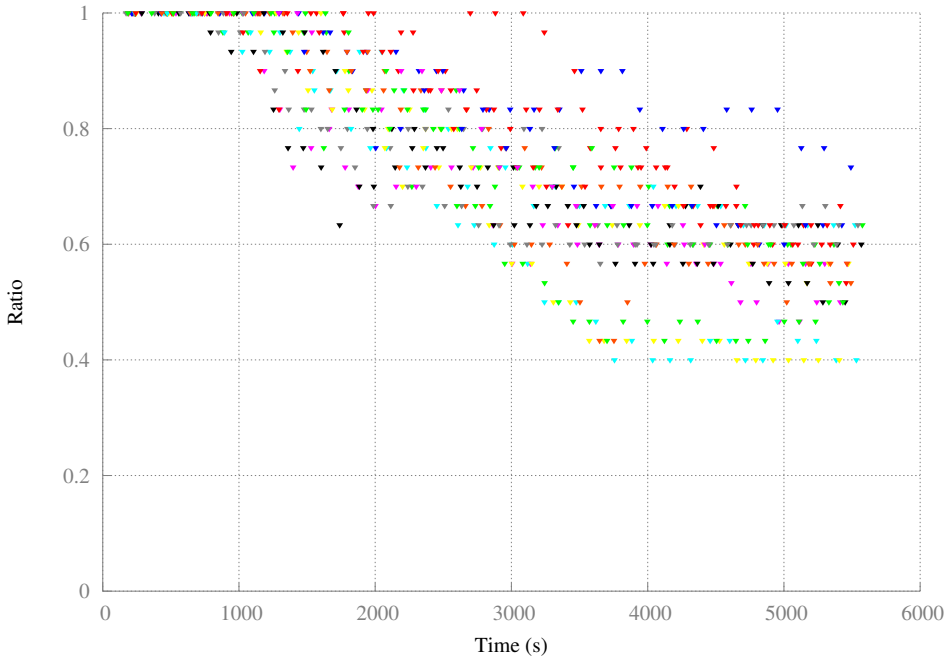


FIGURE 4.142: Instance 9 : Fronts de Pareto cumulés pour tous les runs, à la fin de chaque run.

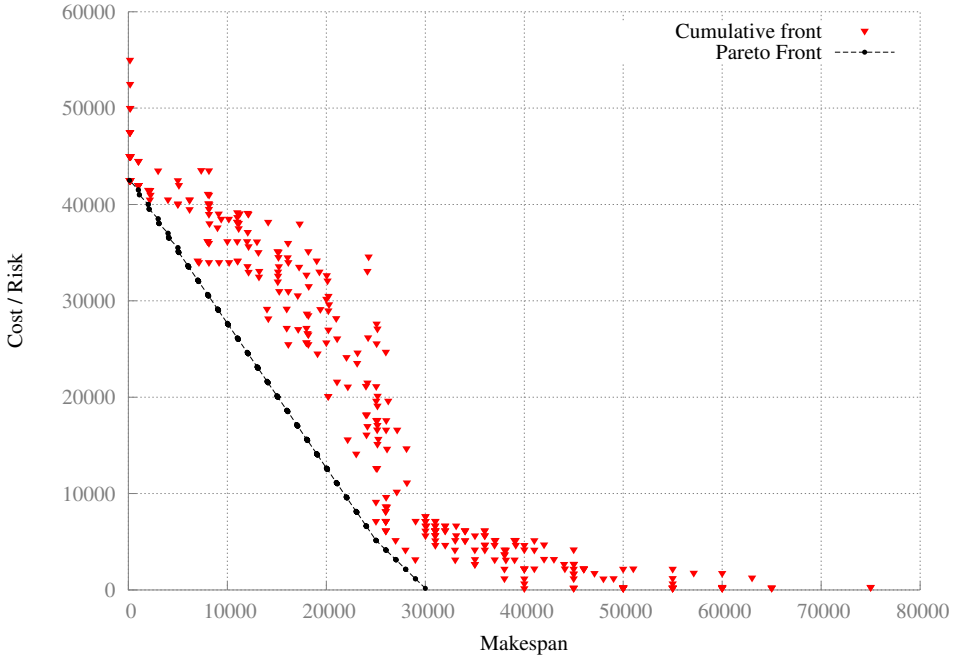


FIGURE 4.143: Instance 9 : Population cumulées pour tous les runs et à tout temps.

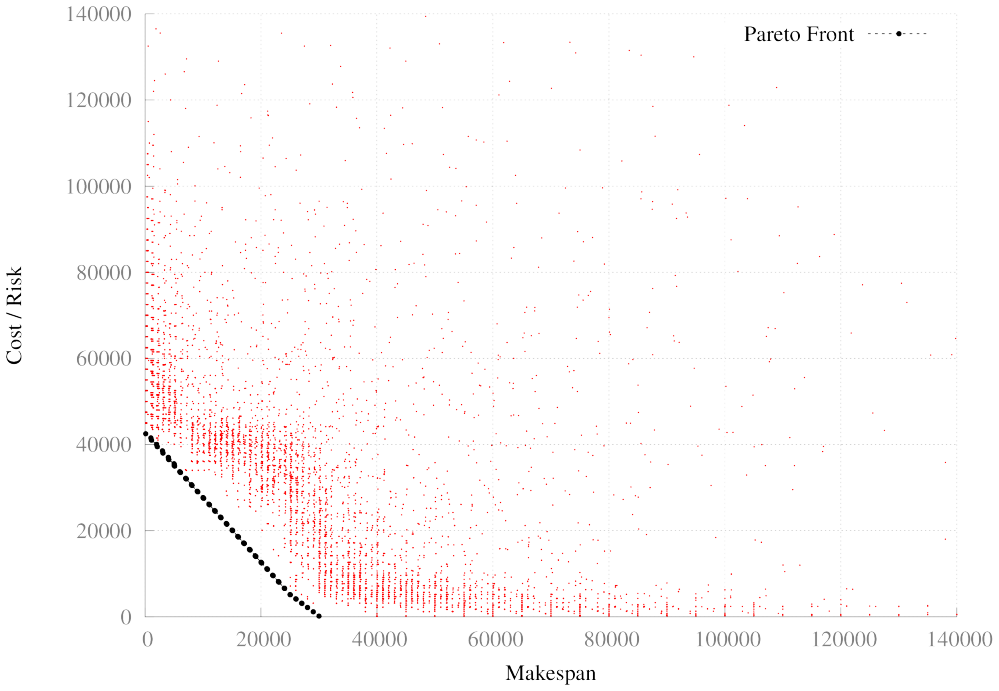


FIGURE 4.144: Instance 9 : Population cumulées pour tous les runs et à tout temps (version colorée).

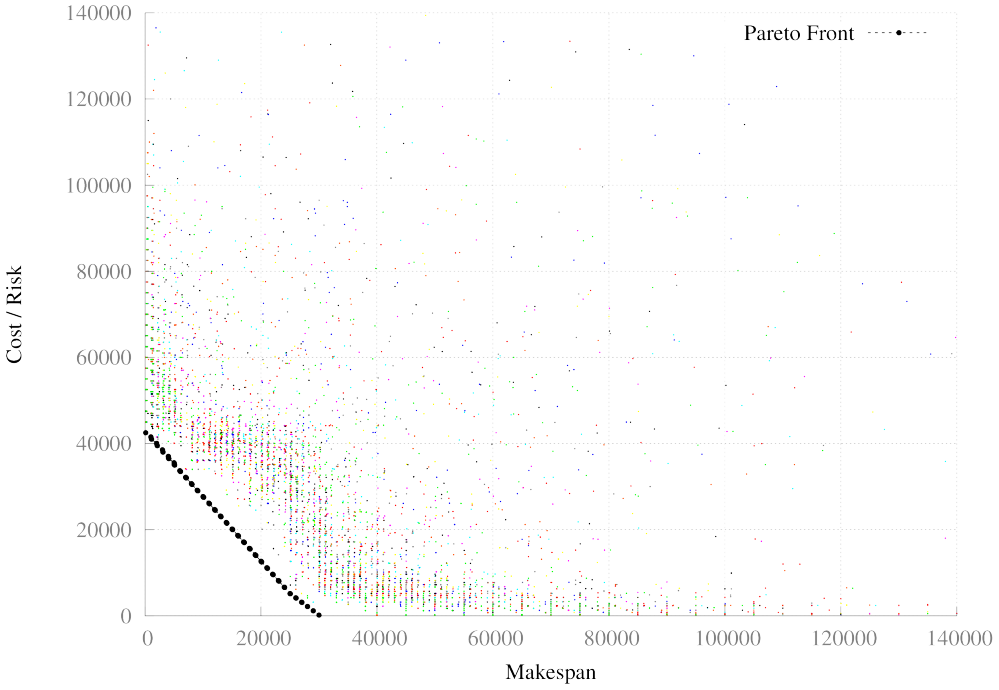
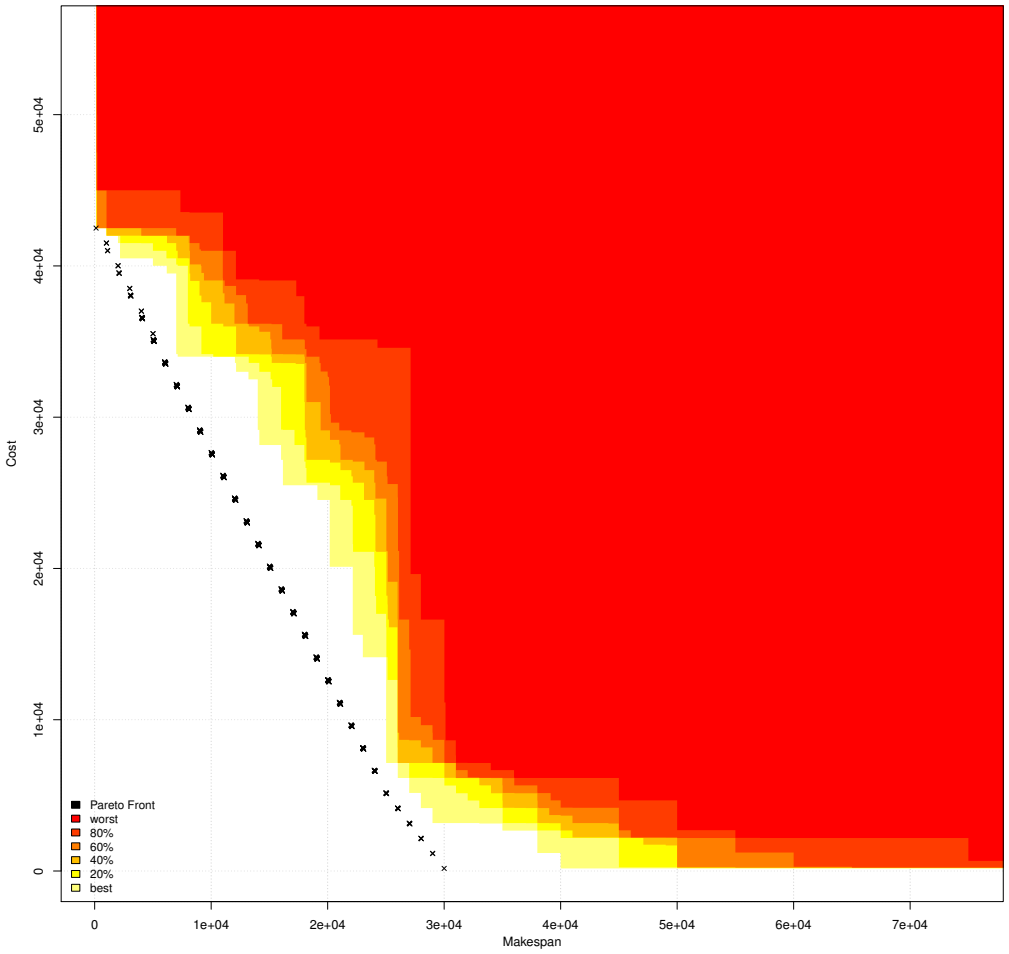


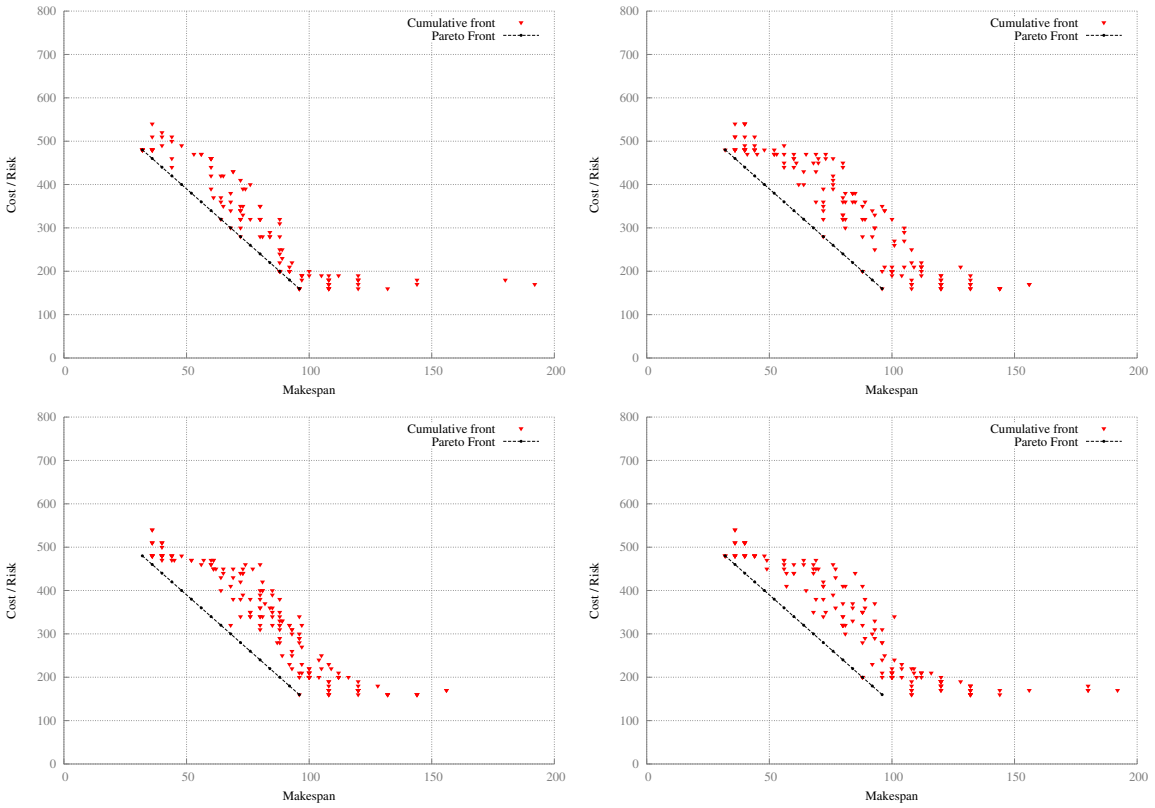
FIGURE 4.145: Instance 9 : Surfaces d'atteinte pour tous les runs.



possible d'utilise un b_{max} de l'ordre de seulement 10 mais les résultats ne semblent pas nécessairement meilleurs.

Le choix de ne pas utiliser les même paramètres est motivé par les premières expériences sur les instances larges. Lorsque DAE avec une stratégie gloutonne utilise les paramètres optimaux trouvés via ParamILS pour une stratégie statique, alors les résultats sont statistiquement moins bons. Or, il apparaît que pour obtenir le meilleur de la stratégie gloutonne, il faut réduire b_{max} et à priori augmenter la taille de population et les facteurs de diversité.

FIGURE 4.152: Fronts de Pareto cumulés. Version statique (haut, gauche), gloutonne basique (haut, droite), gloutonne sélection stricte (bas, gauche), gloutonne amélioration relative (bas, droite).



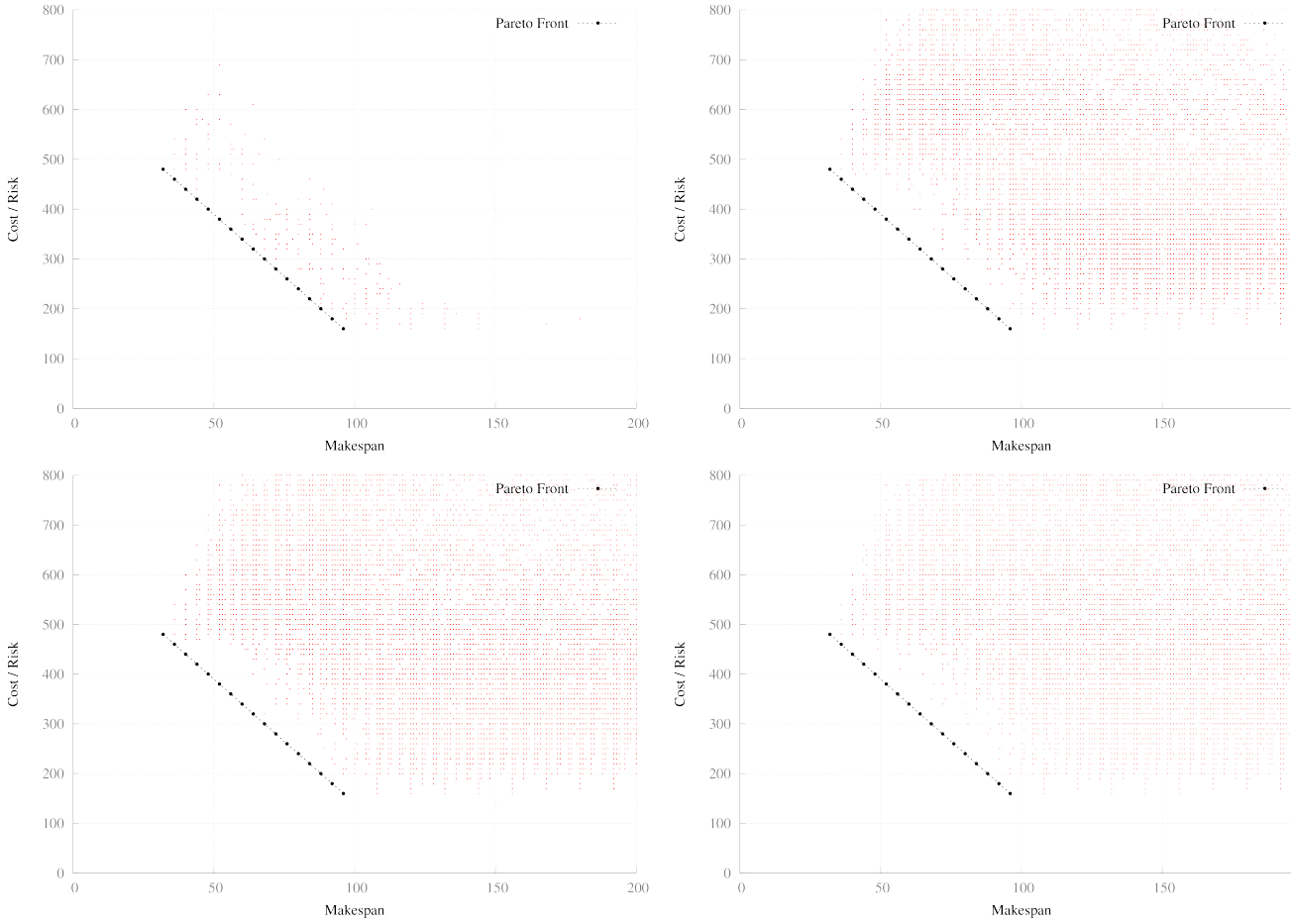
4.6.2.2 Résultats

La figure 4.152 montrant les fronts cumulés donne très clairement gagnante la stratégie statique. On distingue un schéma commun, et que l'on retrouve également pour les autres instances globalement linéaires (instance 1 et 9) : les surfaces d'atteinte semble posséder un point d'inflexion au centre de leur frontière. Les faibles valeurs de *makespan* semblent plus faciles à atteindre, puis certaines valeurs de faibles coûts, tandis que les valeurs minimales de coûts semblent à nous plus difficile à atteindre.

Il est intéressant de noter la différence dans l'évolution de la diversité des vecteurs objectifs. Toutes les stratégies gloutonnes bénéficient d'une diversité qui décroit lentement vers une valeur qui semble stationnaire, contrairement à la stratégie classique qui présente une répartition uniforme et assez faible, comprise entre 0.1 et 0.3.

Cela renforce l'idée que la diversité des vecteurs objectifs dépend largement de la stratégie utilisée, mais également des paramètres utilisés. C'est assez surprenant par ailleurs de voir que sur cette instance, la diversité décroît malgré les paramètres de diversité très

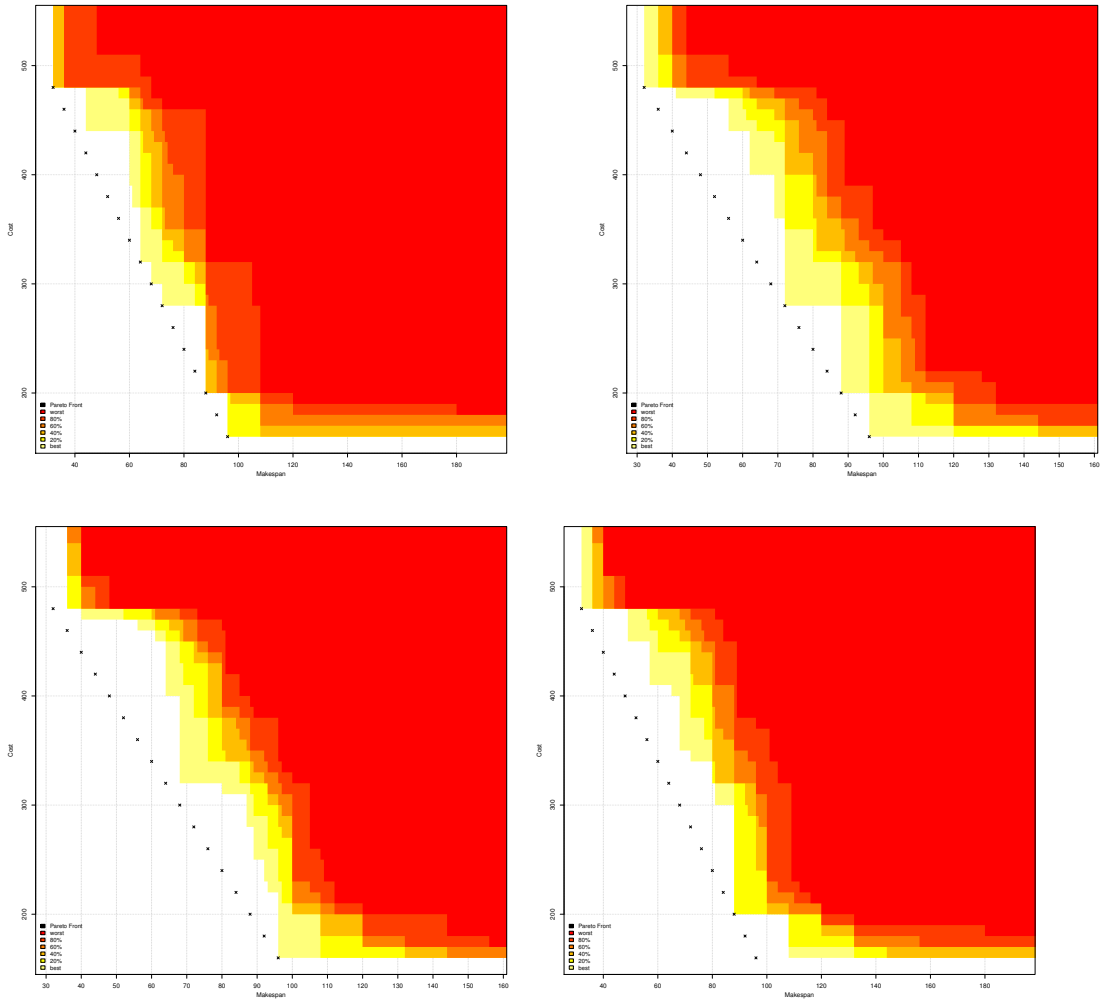
FIGURE 4.153: Populations cumulées (tout temps). Version statique (haut, gauche), gloutonne basique (haut, droite), gloutonne sélection stricte (bas, gauche), gloutonne amélioration relative (bas, droite).



important, contrairement aux instances gloutonnes où elle restait très haute. Vraisemblablement selon provient à la fois de l'instance et de la taille de la population : en considérant des paramètres de diversité très élevés, avec une taille de population plus petite, les probabilités d'avoir deux fois le même plan pour un individu est plus faible qu'avec une large population. Par ailleurs, la zone échantillonnable de l'espace objectifs (i.e. correspondant à des plans) devient de plus en plus restreinte lorsque l'on se rapproche du front exact (sur toutes les instances la zone échantillonnage semble être contenu dans un cône ouvert intersecté avec le front exact à l'une extrémité. Le cône est nécessairement fermé de l'autre côté puisqu'il existe clairement un nombre fini de plans si l'on considère des plans se terminant lorsque tous les passagers ont été transportés à destination). De fait, pour des instances normalisés A et B, si A est plus large que B (i.e, présente une combinatoire plus importante), il est logique qu'à distance également du front il y a plus de probabilité d'obtenir des plans différents avec A qu'avec B. Il

est donc fort probablement que la diversité des vecteurs objectifs diminue à un certain moment, lorsque l'on se trouve suffisamment proche du front.

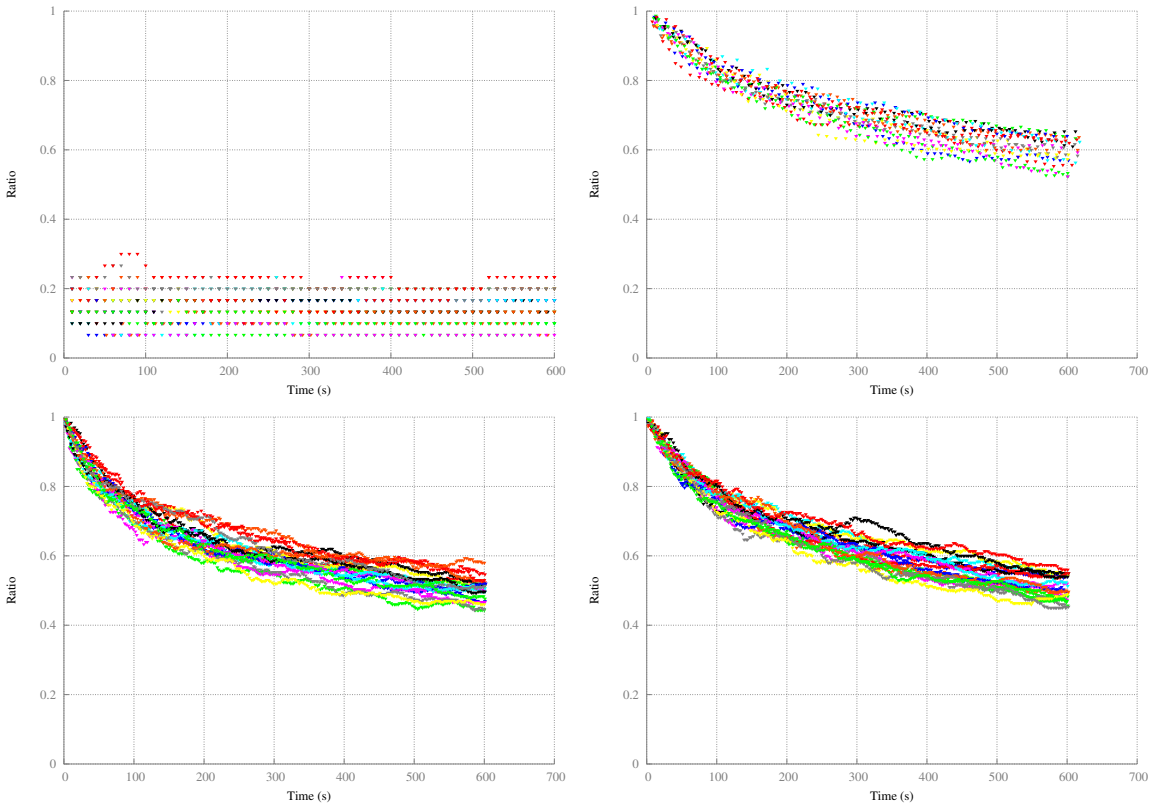
FIGURE 4.154: Surfaces d'atteinte. Version statique (haut, gauche), gloutonne basique (haut, droite), gloutonne sélection stricte (bas, gauche), gloutonne amélioration relative (bas, droite).



La figure 4.156 compare la stratégie statique à la stratégie gloutonne basique et conclut à un avantage pour la stratégie statique. La figure 4.157 compare la version gloutonne basique à la version avec une sélection stricte. Test1 correspond à la version basique et Test2 à la version stricte. Si la mise côte à cote des surfaces d'atteinte mènerait intuitivement à préférer Test2 dont le pire des *runs* est relativement meilleurs que pour Test1, un test statiquement sur les fonctions d'atteinte ne rejette pas l'hypothèse nulle : il n'y a pas assez de différence pour affirmer qu'une version est meilleure que l'autre.

On obtient également les même résultats et les même commentaires en comparant la version utilisant une sélection du meilleur plan par amélioration absolue et la version

FIGURE 4.155: Diversité des vecteurs objectifs au sein de la population. Version statique (haut, gauche), gloutonne basique (haut, droite), gloutonne sélection stricte (bas, gauche), gloutonne amélioration relative (bas, droite).



utilisant une amélioration relative, comme en atteste la figure 4.158. C'est cependant un petit peu surprenant compte tenu du fait que les objectifs ne soient pas normalisés.

FIGURE 4.156: Comparaison des surfaces d'atteinte entre la stratégie statique et gloutonne basique.

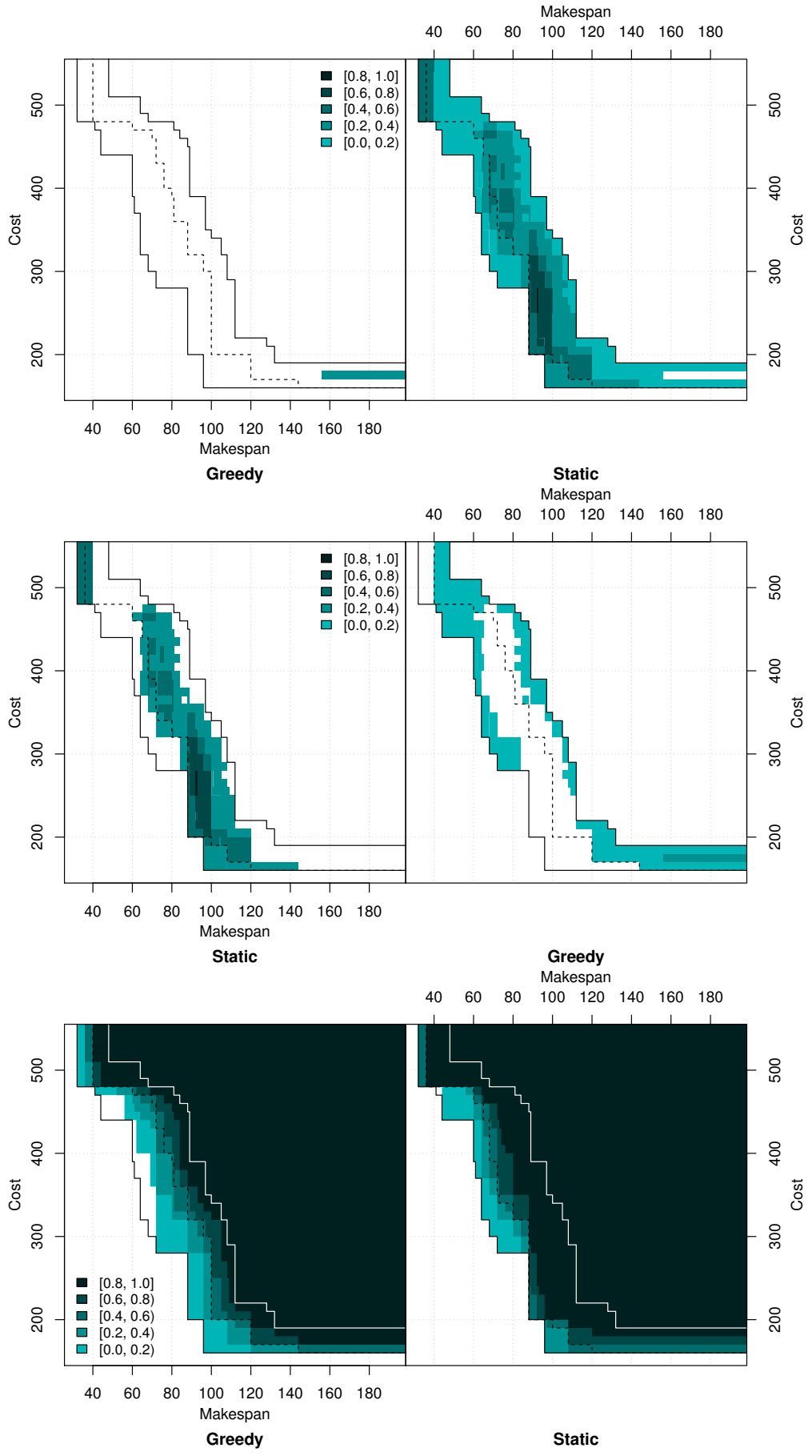


FIGURE 4.157: Comparaison des surfaces d'atteinte entre la stratégie gloutonne basique et gloutonne stricte.

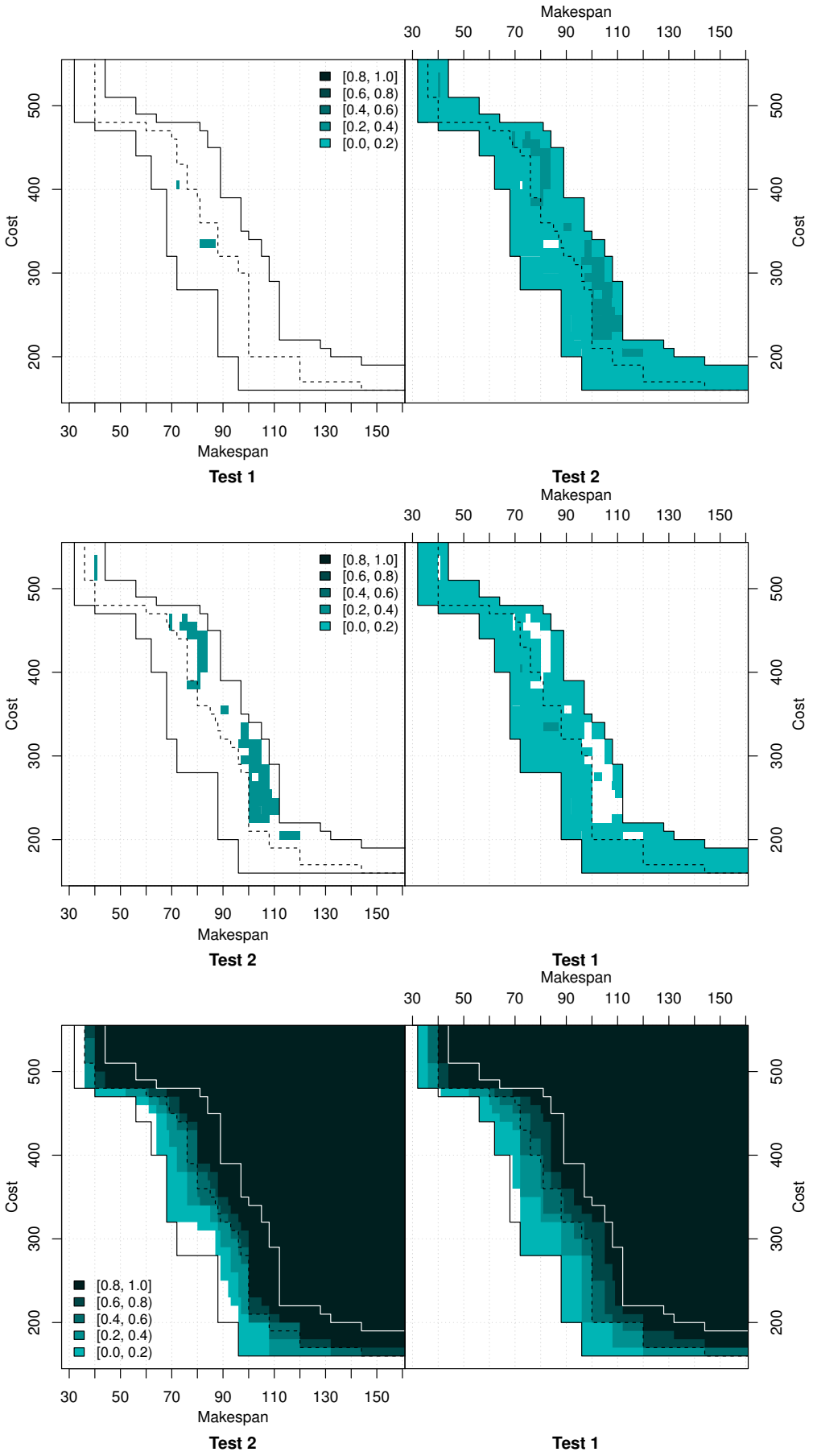
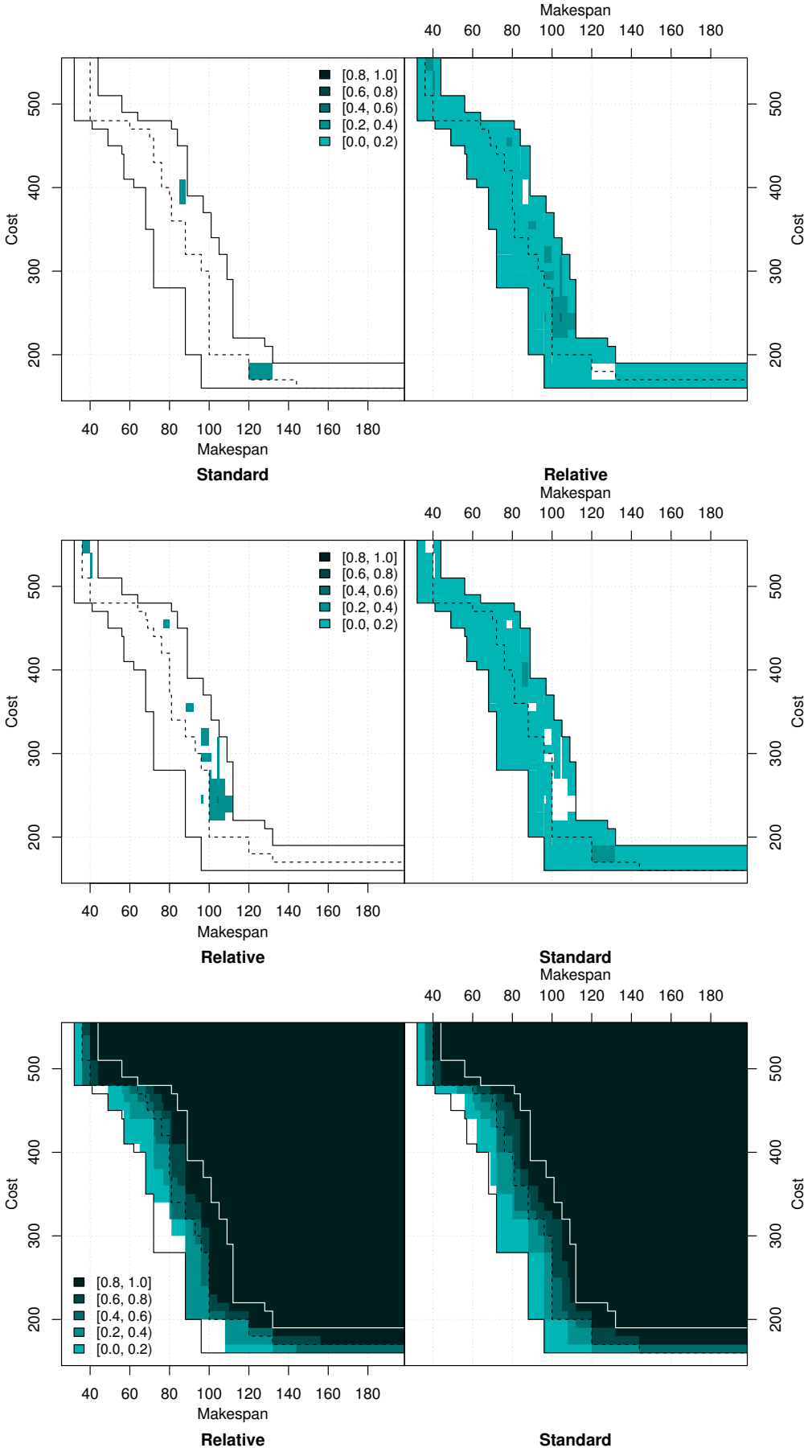


FIGURE 4.158: Comparaison des surfaces d'atteinte entre la stratégie gloutonne basique et gloutonne amélioration relative.



4.6.3 Conclusion

On peut tirer plusieurs conclusions de ces séries d'expérience. Tout d'abord, malgré des résultats statistiquement inférieurs à la stratégie statique, les différences semblent assez peu importantes et encourageantes à continuer dans ce sens.

La comparaison entre la série des paramètres de ParamILS et la série avec une diversité accrue et un b_{max} réduit est très révélatrice. Malgré un faible échantillonnage de l'espace objectif, les résultats semblent globalement similaires. Sur certaines instances très compliquées, notamment la 4, on observe des résultats clairement moins bon et à contrario une baisse plus brutale de l'hypervolume en début de *run*. Le paramètre b_{max} semble donc avoir une influence très intéressante : il est possible d'obtenir des plans faisables avec une valeur très faible, pour peu que l'on réessaye plusieurs fois, mais après un moment, il se peut que la valeur soit trop faible pour trouver un plan meilleur pour l'individu considéré. De même, certaines zones de l'espace objectifs semblent inaccessibles. Ce sont souvent des zones dont la projection orthogonale sur le front de Pareto se trouverait au centre de celui-ci. Cette constation nous amènera a proposer de l'échantillonnage par individu lors de l'évaluation dans la section 4.8. D'autres pistes à explorer seront également une stratégie adaptative ou auto-adaptative du paramètre b_{max} : une valeur faible semble suffisante pour démarrer le processus, et une valeur de plus en plus importante permet de continuer à faire avancer le front.

4.7 Stratégie auto-adaptative

La stratégie auto-adaptive consiste à encoder les paramètres à utiliser directement dans le génotype des individus. Les paramètres vont donc évoluer avec les individus et l'argument expliquant la pertinence de ces stratégies consiste à dire que les paramètres jouent un rôle certain sur la qualité des individus. Ainsi, les meilleurs individus, qui sont ceux qui survivent au sein de la population, sont également ceux qui doivent porter les meilleurs paramètres. À l'aide des opérateurs de croisement, ces bons paramètres vont donc se diffuser au sein de la population.

Comme le génotype des individus n'est plus homogène, c'est à dire qu'on a la part ≪ traditionnelle ≫ qui code une solution au problème, et une part dédiée aux paramètres,

il faut étendre les opérateurs de variations afin qu'ils prennent également en compte les gènes codant les paramètres. C'est tout le travail de l'élaboration d'une telle stratégie.

Dans notre cas nous avons testé une stratégie très simple qui étend les opérateurs de DAE. Surtout, la spécificité ici est que chaque individu ne possède qu'un objectif et non une distribution de probabilité comme c'était le cas jusqu'à présent.

Opérateur de croisement Le croisement de DAE est un croisement à un point et un seul individu est retenu : celui qui conserve le mieux l'ordre chronologique. Ainsi, le premier individu généré à partir des parents conserve l'objectif du premier parent, et le second individu conserve le second parent.

On pourrait éventuellement, dans des futurs travaux, travailler directement sur la distribution de probabilité en gardant le même principe.

Mutation La mutation de l'objectif n'a pas lieu au moment des mutations de l'autre partie du génotype mais après chaque évaluation. Cela revient pratiquement au même puisqu'un individu n'est évalué que celui-ci est modifié, soit via un croisement, soit via une mutation.

À chaque évaluation, l'objectif est modifié avec une probabilité p_o, tiré aléatoirement sur les 3 autres objectifs. La probabilité p_o est un hyper-paramètre que l'on a considéré statique et fixé arbitairement à 0.05. Ce choix est largement contestable et mériterait à être mis dans l'interface et optimisé via ParamILS.

Pour travailler sur une distribution de probabilité, on pourrait imaginer plusieurs opérateurs :

- Ajout d'un bruit gaussien sur chacun des composantes. On peut penser à un bruit centré sur 0.5 et de variance 1, impliquant une renormalisation de la distribution.
- Lissage des probabilités en les recalculant de la sorte : $p'_i(x) = \frac{1-p(x)}{l-1}$ avec l le nombre d'objectifs.

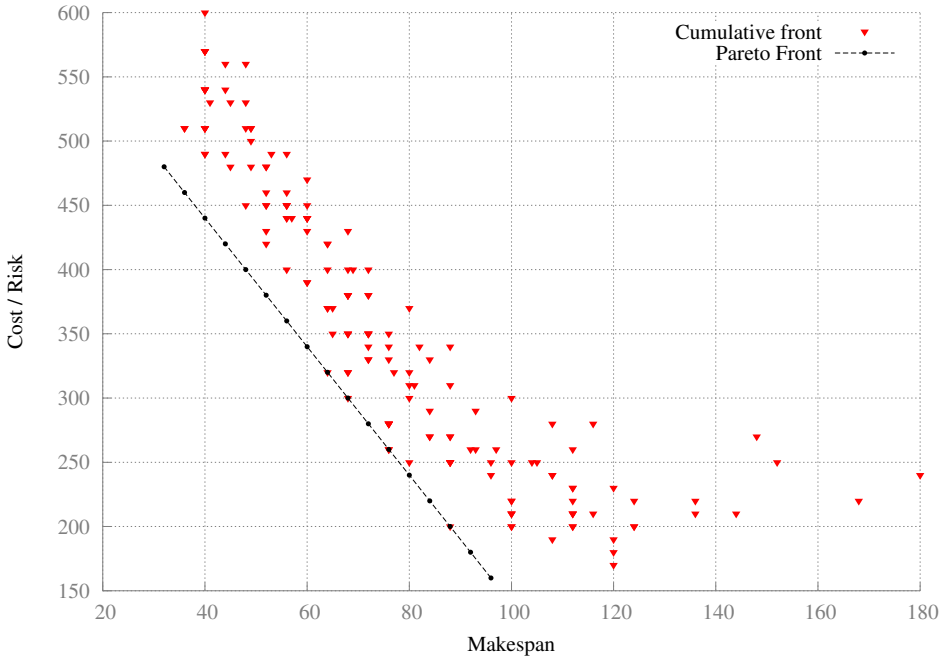
Le lissage ces probabilités provient de la convergence de la suite $x_{n+1} = \frac{1-x_n}{a}$ vers $\frac{1}{a+1}$ si $0 < x_0 < 1$ et $a > 1$ puisqu'en résolvant l'équation de récurrence nous avons $x_n = \frac{1}{a+1} + (-\frac{1}{a})^n x_0$. Cela ne fonctionne pas avec un nombre d'objectifs qui vaut 2 puisqu'alors la série est simplement alternée. Ce n'est cependant pas un problème ici puisque YAHSP propose 4 objectifs.

4.7.1 Résultats empiriques

Par manque de temps, seule l'instance Zeno 9 a fait l'objet de tests sur la stratégie auto-adaptive. Si les *runs* sont toujours au nombre de 20, leur durée est de 600 secondes.

En comparaison avec la stratégie statique, la figure 4.159 indique un front cumulé un peu plus éloigné (à confirmer avec les surfaces d'attente) mais qui semble parfaitement uniformemment distribué sur l'ensemble du front exact.

FIGURE 4.159: Instance Zeno9 : Fronts de Pareto cumulés pour tous les runs, à la fin de chaque run.



La population cumulée montre un échantillonnage de l'espace objectif plus important que pour la stratégie statique, plus dense proche du front, malgré une zone qui semble peu atteinte. On peut émettre une critique vis à vis du fait que contrairement à la stratégie classique, on visite également plus de zones éloignées du front exact.

Les surfaces d'atteinte de la figure 4.162 indique une forme différente de ce que l'on a pu observer avant avec les instances globalement linéaires, à savoir des surfaces d'atteinte présentant un point d'inflexion et de meilleurs résultats sur les extrémités. On a au contraire ici des surfaces d'atteinte sans valeur de coût ou de *makespan* particulièrement atteinte.

FIGURE 4.160: Instance Zeno9 : Population cumulées pour tous les runs et à tout temps.

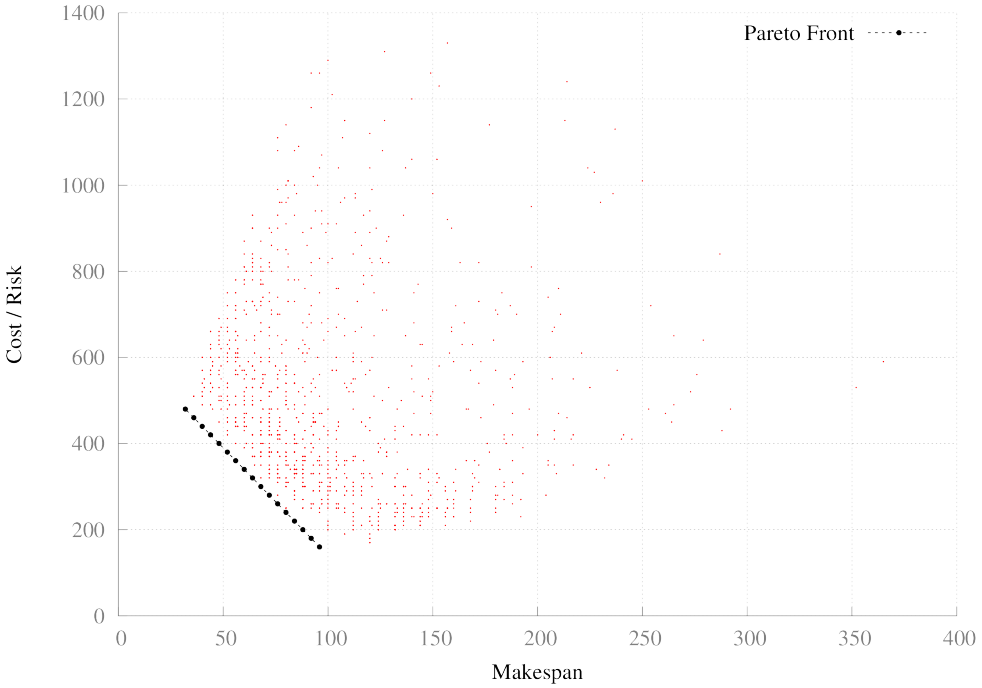
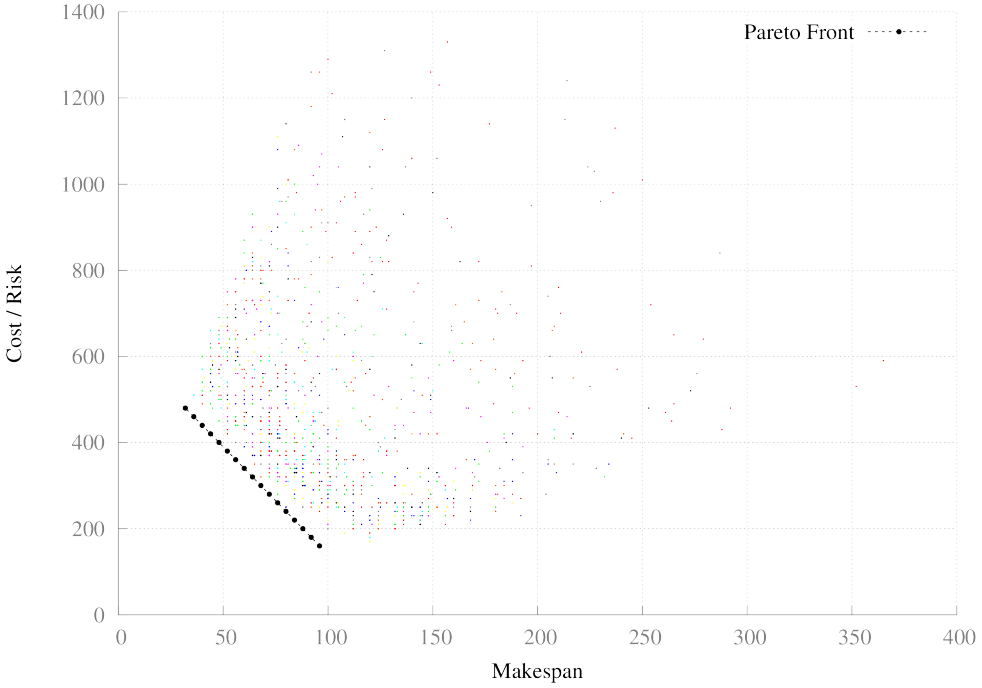


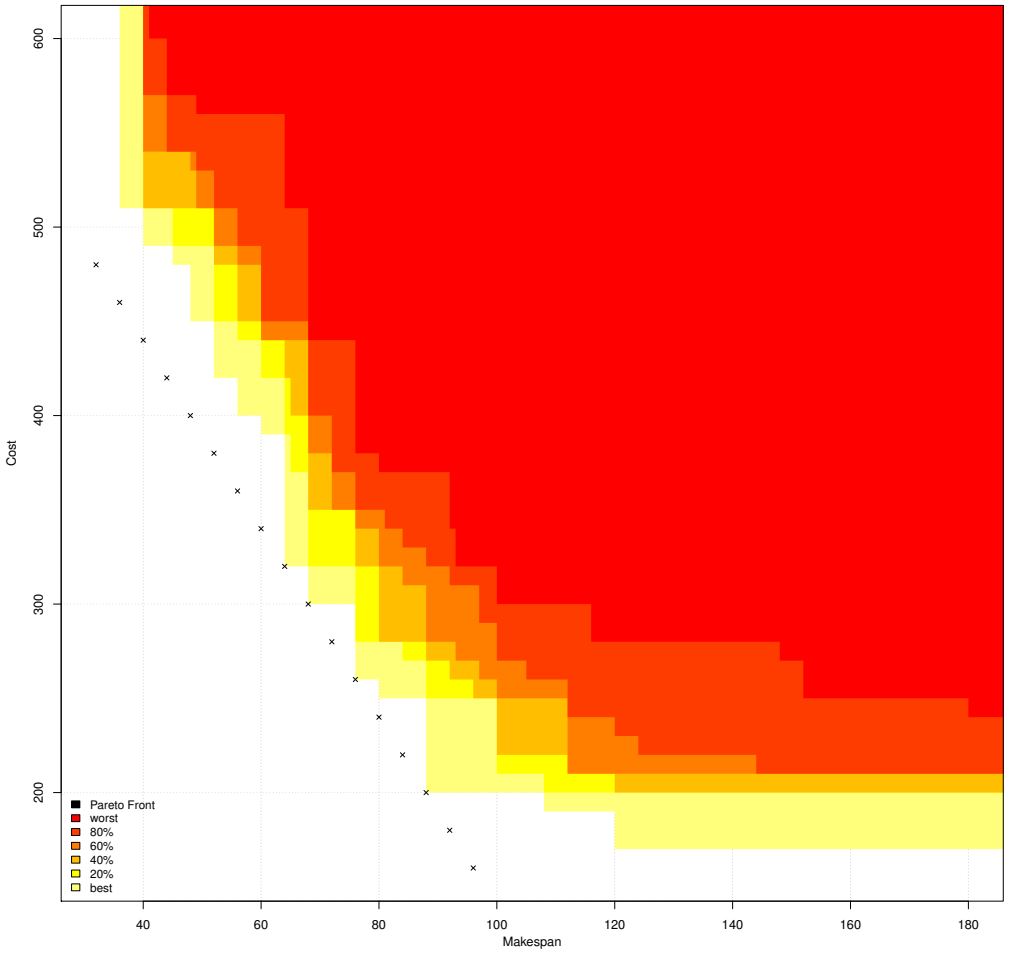
FIGURE 4.161: Instance Zeno9 : Population cumulées pour tous les runs et à tout temps (version colorée).



Il est intéressant de noter les différences au niveau des trajectoires de l'hypervolume par rapport à la stratégie statique ou les autres stratégies. On n'observe pas ici de « saut »dans l'hypervolume, ce qui explique probablement la faible densité des points atteint proche du front. Cela confirme un peu plus l'hypothèse d'une zone relativement difficile à atteindre proche du front, qui se traverse par « saut ». C'est donc un point en défaveur de la stratégie auto-adaptative puisqu'il semblerait qu'on n'arrive pas à retrouver ce comportement (que l'on avait pourtant avec une stratégie adaptative).

A contrario, la diversité des vecteurs objectifs est totalement comparable à la stratégie classique.

FIGURE 4.162: Instance Zeno9 : Surfaces d'atteinte pour tous les runs.



Comparaison avec la stratégie statique La comparaison des surfaces d'atteinte confirme ce qui a pu être dit précédemment. À savoir que la stratégie auto-adaptative

FIGURE 4.163: Instance Zeno9 : Trajectoires de l'hypervolume pour tous les runs.

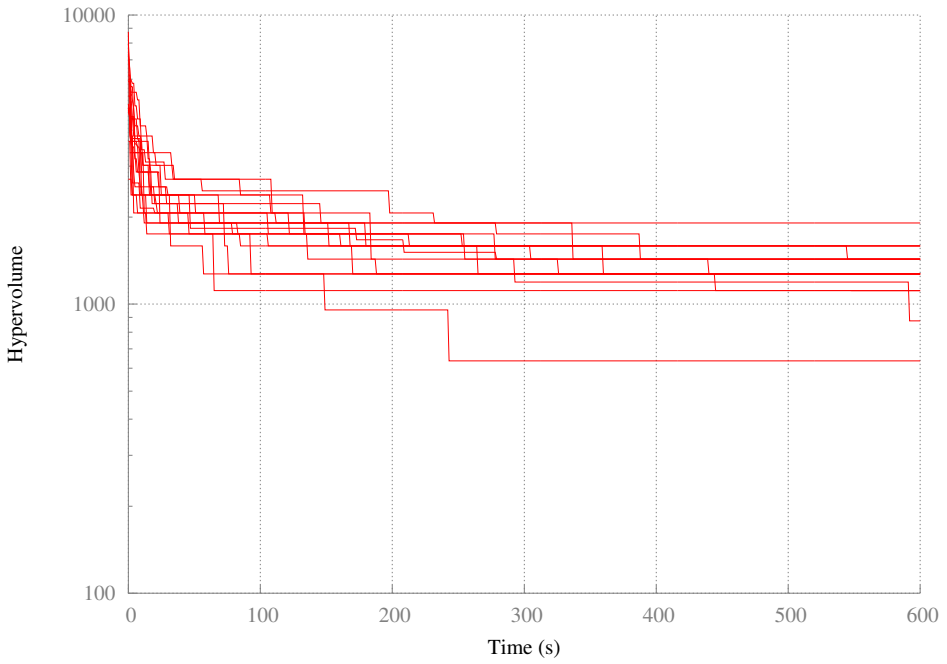


FIGURE 4.164: Instance Zeno9 : Diversité des vecteurs objectifs pour tous les runs.

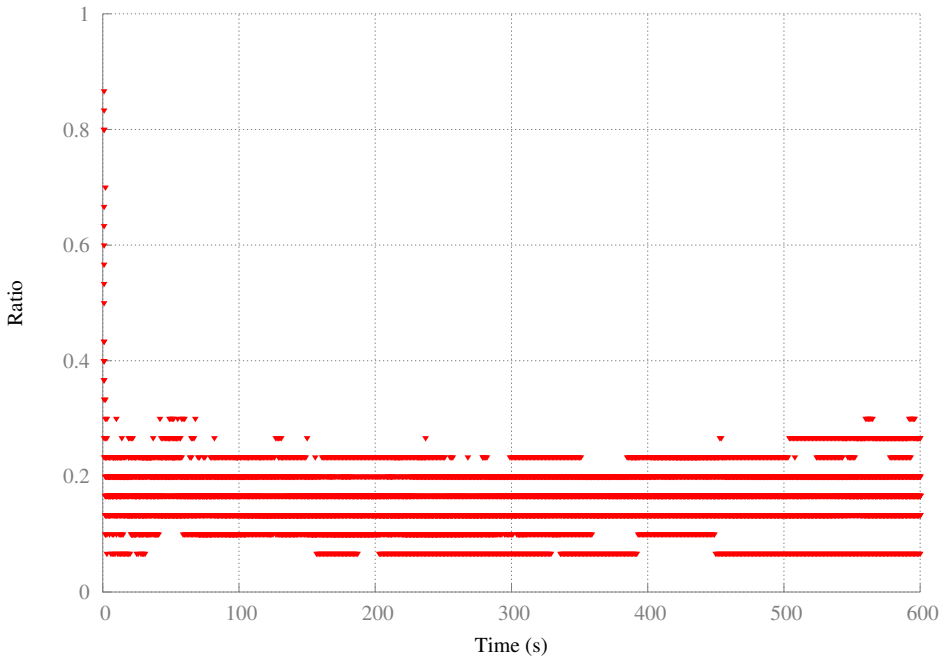
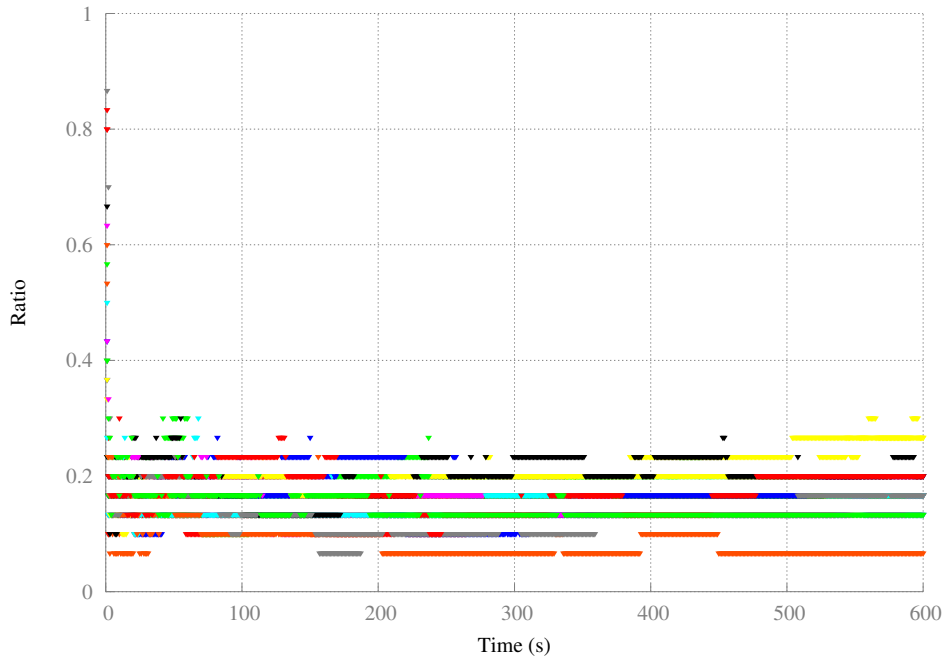


FIGURE 4.165: Instance Zeno9 : Diversité des vecteurs objectifs pour tous les runs (version colorée).



semble moins capable d'atteindre les valeurs extrêmes. A contrario, ses résultats sur les parties centrales semblent aussi bien voire légèrement mieux que pour la stratégie statique.

FIGURE 4.166: Instance Zeno9 : Comparatif des surfaces d'atteinte.

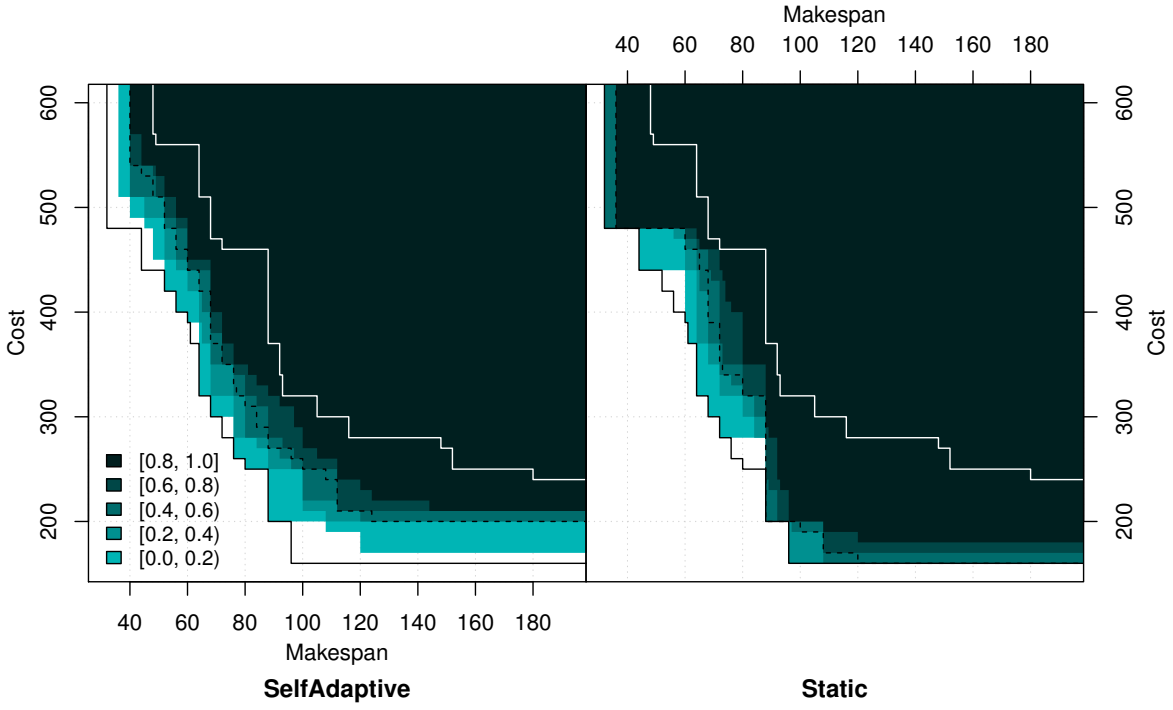


FIGURE 4.167: Instance Zeno9 : AutoAdaptative comparée à Statique.

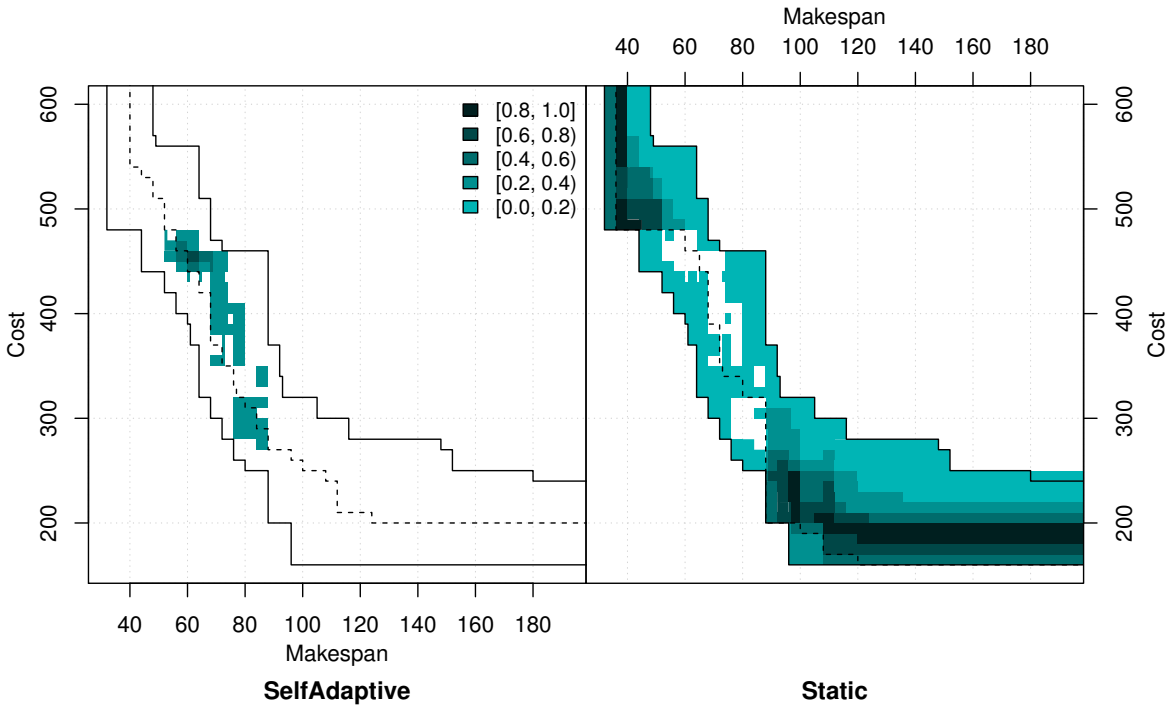
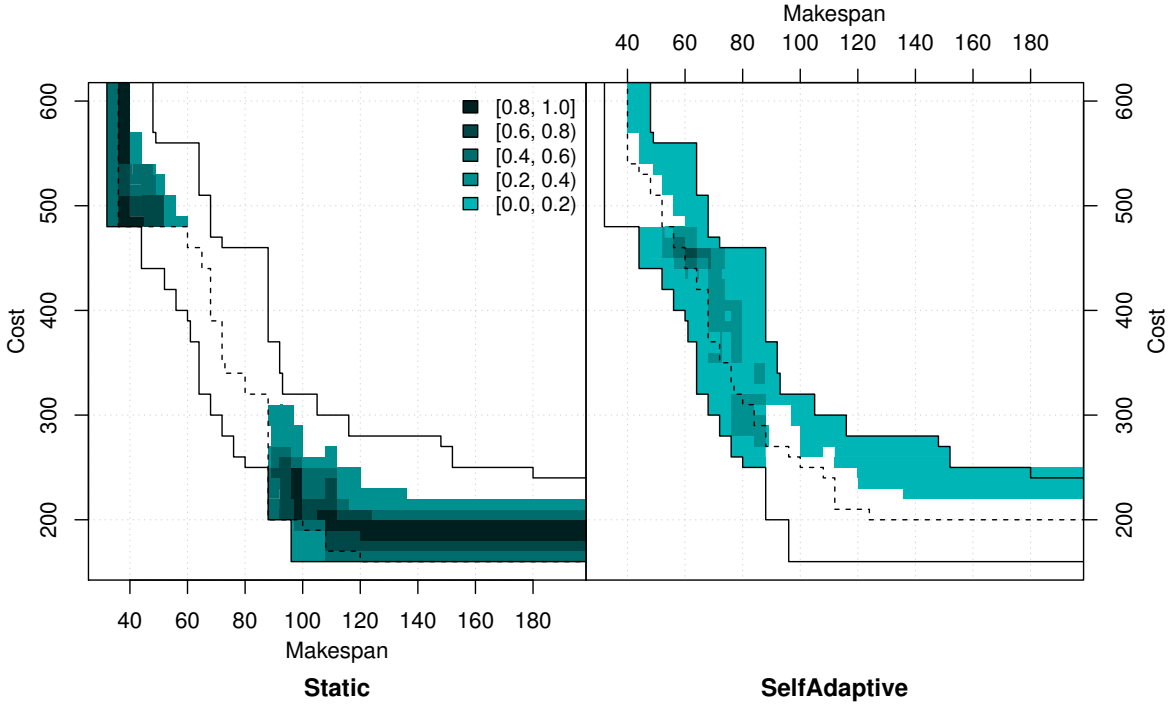


FIGURE 4.168: Instance Zeno9 : Statique comparée à AutoAdaptative.



4.7.2 Conclusion

Les résultats sont trop peu nombreux pour tirer une quelconque conclusion quand aux performances de la stratégie adaptative, d'autant que la stratégie implémentée est extrêmement rudimentaire. Les résultats sont toutefois assez encourageant dans le sens où si statistiquement ils sont moins bon que la stratégie statique et présentent quelques faiblesses apparentes, ils permettent de montrer qu'il est possible d'obtenir des résultats corrects sans l'aide d'aucun paramètre ou hyper-paramètre. De plus, ils confirment un peu plus des observations qui ont pu être faites précédemment et que l'on essayera de discuter dans la section 4.8 qui suit.

4.8 Réflexions sur l'évaluation

4.8.1 De la stochasticité de YAHSP.

Lorsqu'un individu est évalué, le foncteur d'évaluation de DescarwINva appeler YAHSP qui va tenter de résoudre les sous-problèmes que décrit l'individu. Si l'individu est faisable, c'est à dire que YAHSP a réussit à résoudre tous les sous-problèmes et construire en plan global, alors le foncteur d'évaluation affecte le vecteur objectif de l'individu comme les valeurs du *makespan* et du coût (ou risque) correspondant au plan trouvé.

Plusieurs facteurs vont influencer YAHSP, et conduire, pour un même individu, à l'obtention de plans différents.

Premièrement, la séquence pseudo-aléatoire de YAHSP, dépendant de la graine passée à l'initialisation de ce dernier, peut conduire à différents plans pour un même problème à résoudre. En effet, rappelons que YAHSP est un solveur stochastique sous-optimal. Au vu de travaux antérieurs, il apparaît que réinitialiser YAHSP, avant d'évaluer à nouveau un individu est largement bénéfique pour la résolution générale du problème de planification.

Secondement, YAHSP est un solveur mono-objectif et ainsi, la résolution d'un sous-problème va se focaliser sur un objectif particulier, qui sont au nombre de quatre : `length`, `cost`, `makespan_max` et `makespan_add`. Le premier correspond à optimiser la longueur du plan obtenu en nombre d'actions tandis que le second vise à optimiser le

coût du plan. L'objectif `makespan_max` tient compte de la durée de départ des actions ainsi que de la date de départ minimale pour effectuer l'action considérée. Ainsi, si l'on considère un ensemble d'actions A et pour chaque action $a \in A$ la durée t_a et la date de début $\bar{t}_a$, une borne inférieure admissible pour les atomes pouvant être produit par les actions de A sera $\underset{a \in A}{max}(t_a + \bar{t}_a)$. La méthode exacte de sélection est décrite originalement dans l'article de Bonet et Geffner [43]. Enfin, avec `makespan_add` et le même ensemble A, la borne retenue sera la somme : $\sum_{a \in A} (t_a + \bar{t}_a)$. Cette stratégie a tendance à réduire la durée des actions présentes dans le plan, mais tolère des actions de durées importantes. Il apparaît naturelle que des actions de durées importantes puisse être nécessaire dans l'obtention d'un plan final optimal : une action de durée 10 pouvant être totalement effectuée en parallèle est plus intéressante que 10 actions de durées 1 qui doivent être exécutées de manière séquentielle.
En toute logique, pour un même individu, le plan obtenu ne sera vraisemblablement pas identique suivant l'objectif, même en considérant une réinitialisation de YAHSP, avec la même graine, juste avant la résolution.

Si l'on omet le premier point, on peut considérer qu'un individu x peut mener à l vecteurs objectifs, où l est le nombre d'objectifs d'optimisation disponibles au niveau du solveur local (l est donc différent de m, le nombre d'objectifs du problème initial, avec ici $l = 4$ et $m = 2$). Il s'agit alors de trouver l'objectif optimal à utiliser, en un sens à définir [9], la finalité étant toujours d'obtenir le meilleur ensemble de Pareto possible (c'est à dire tel que le front de Pareto soit le plus proche du front exact et le plus uniformément réparti). Il est évident que l'objectif optimal est propre à chaque individu voire à chaque gène définissant un sous-problème.

Si l'on considère le premier point, pour un même objectif d'optimisation o_i, on peut considérer qu'un individu x peut mener à k vecteurs objectifs, où k est le nombre de plans avec des vecteurs objectifs différents que l'on peut obtenir avec YAHSP, en utilisant l'objectif o_i, pour toute les graines envisageables. Il est clair que la combinatoire de cet ensemble peut être énorme. Le problème est donc d'établir une méthode pour évaluer la capacité d'un individu à être optimisé selon l'objectif o_i et c'est ce que dois refléter son vecteur objectif. De manière formelle, la traditionnelle fonction objectif $f : X \to \mathbb{R}^m$,

9. Il semble raisonnable de considérer que l'objectif optimal est celui qui conduit à un vecteur objectif qui domine n'importe quel autre vecteur objectif atteignable au travers d'un autre objectif et, dans le cas où ils sont non-comparables, trouver une métrique du type « amélioration » par rapport à un point de référence, le point de référence étant par exemple ici le vecteur objectif de la génération précédente.

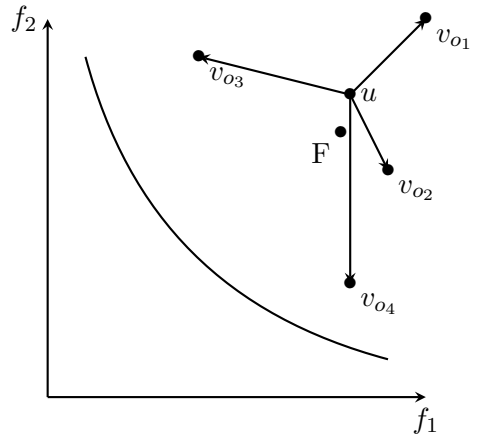


FIGURE 4.169: Un individu x, évalué à u à la génération précédente, peut mener à 4 points selon les 4 objectifs $(o_i)_i$ d'un solveur déterministe. Le point F est le vecteur moyen.

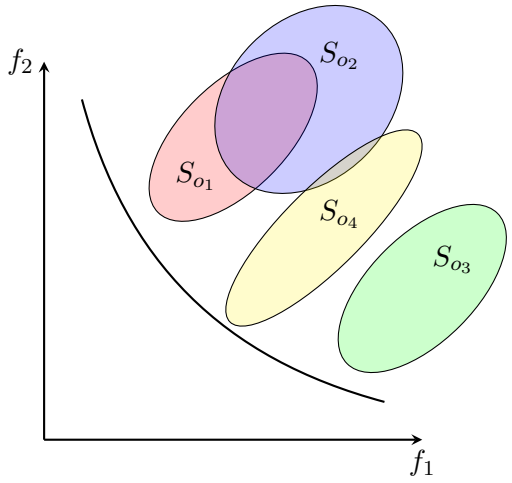


FIGURE 4.170: Les surfaces $(S_{o_i})_i$ représentent les surfaces dans lequelles on peut trouver un plan faisable en utilisant YAHSP sur un même individu x selon l'objectif o_i. On observe empiriquement une distribution des points symétrique par rapport à un axe parallèle à la première bissectrice. Comportement probablement spécifique au solveur embarqué utilisé. La zone accessible de x connaissant la distribution de probabilité sur $(o_i)_i$ est $\underset{i}{\cup} S_{o_i}$.

surjective, est remplacée par une fonction $\tilde{f} : X \times (\Omega, F, \mathbb{P}) \to \mathbb{R}^m$, avec X l'espace des variables de décision (ou génotypique) et $(\Omega, F, \mathbb{P})$ un espace probabilisé dont un élément correspondrait à une initialisation de YAHSP selon une graine particulière. Évidemment, cela peut être négligeable si pour un individu x et son ensemble $(v_j)^i_{1 \leq j \leq k}$ des vecteurs objectifs atteignables selon l'objectif i, la distance $d^i(x) = \underset{n,m}{max}(||v_n - v_m||_2)$ est raisonnablement petite. Nous verrons qu'en pratique ce n'est pas le cas, ce qui peut de fait poser problème pour l'évaluation de l'objectif optimal à utiliser.

En pratique, les deux points sont réunis et ainsi et le problème double :

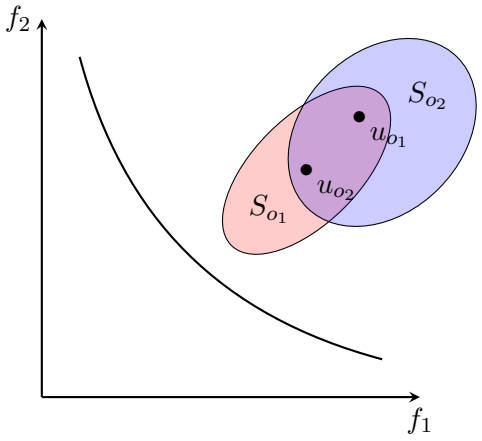


FIGURE 4.171: Il semble plus intéressant d'optimiser en utilisant o_1 plutôt que o_2 car on peut trouver un point $u \in S_{o_1}$ tel que $\forall v \in S_{o_2},\ u \succ v$. Cependant, l'appel à YAHSP utilisant o_1 à retourné u_1 qui est dominé par le point u_2 retourné en utilisant o_2. On peut généraliser le raisonnement aux individus en considérant la zone accessible de x comme $\bigcup_i S_{o_i}$.

— Comment estimer efficacement l'objectif optimal à utiliser, en tenant compte du non-déterminisme de la fonction d'évaluation ?

— Comment affecter un vecteur objectif reflétant au mieux sa capacité à être optimisé, de sorte à influencer le mécanisme de sélection et remplacement ?

L'évaluation classique d'un individu pose deux soucis majeurs. Dans le philosophie de DAE, un individu devrait être évalué selon qu'il présente ou non des sous-problèmes faciles à résoudre et menant à des plans de bonne qualité. Affecter la qualité d'un individu comme le vecteur objectif du plan obtenu par le solveur local a du sens lorsque le solveur est déterministe ou lorsque la zone atteignable de l'espace objectif est pratiquement ponctuel par rapport à l'espace objectif résultant de plans admissibles. Si ce n'est pas le cas, il est très facile de « tromper » l'algorithme génétique comment en témoigne le figure 4.171. Si la zone atteignable est suffisamment grande, alors finalement l'algorithme génétique ne fait qu'apporter de la diversité puisque l'on tue le mécanisme de sélection et tout procédé élitiste à cause de l'aléa de la fonction d'évaluation. On peut alors se demander si l'effet positif de l'algorithme génétique n'est pas nul et que le résultat final obtenu n'est pas uniquement conditionné par les capacités du solveur local, ses paramètres (notamment le b_{max} dans le cas de YAHSP) et la chance d'avoir des mutations favorables (doublées d'une évaluation chanceuse).

Le second soucis, qui découle du premier, est que cette incapacité à évaluer correctement

un individu implique également une incapacité à évaluer correctement la stratégie optimale. Dans le cas d'une stratégie utilisant comme *feedback* l'évaluation unique d'un individu par rapport à l'évaluation, aussi unique, à la génération précédente, on se retrouve avec le même problème de pertinence de l'information illustrée par le schéma 4.171. Dans le cas d'une stratégie auto-adaptative, le problème est encore plus clair puisque ces stratégies reposent sur l'idée que les meilleurs paramètres mènent aux meilleurs individus qui eux seront conservés. À partir du moment où ce ne sont pas les meilleurs individus qui sont conservés la stratégie auto-adaptative est mise à défaut. On peut raisonnablement se demander si ces stratégies sont meilleures qu'un tirage aléatoire du paramètre à utiliser.

4.8.2 Échantillonnage

Pour tenir compte du non-déterminisme de la fonction d'évaluation induite par le comportement stochastique de YAHSP et pour acquérir assez d'information pour décider du meilleur vecteur objectif à considérer, nous proposons de modifier la séquence d'évaluation d'un individu, décrite par l'algorithme 6 et introduirons de l'échantillonnage (algorithme 8). Nous discuterons un moyen de sélectionner ou déterminer un vecteur objectif au sein de l'échantillon.

4.8.2.1 Estimer la capacité à être optimisé

Comme nous l'avons vu, tirer au hasard un vecteur objectif dans la zone atteignable d'un individu ne peut pas traduire la facilité de celui-ci à être optimisé. L'objectif est donc de trouver un mécanisme d'évaluation qui permette de traduire cette difficulté.

On rappelle la définition de la fonction d'évaluation actuelle : $\tilde{f} : X \times (\Omega, F, \mathbb{P}) \to \mathbb{R}^m$.

Algorithm 8 Séquence d'évaluation d'un individu x avec échantillonage et strategie.

x.objVector ← unfeasible
x.objVector ← setObjectiveVector(x) {As we set the state on unfeasible, the objective vector is computed according to that state.}
for k from 1 to k_max **do**
  if StratLevel = Population **then**
    o_i ← strategy()
  else if StratLevel = Individual **then**
    o_i ← x.strategy()
  end if
  YAHSP_Initialization(rand()) {Initialize YAHSP with a random seed. }
  YAHSP_Optimize(o_i)
  s ← feasible
  for all x_i in x **do**
    if StratLevel = Gene **then**
      o_i ← x_i.strategy()
      YAHSP_Initialization(rand()) {Initialize YAHSP with a random seed. }
      YAHSP_Optimize(o_i)
    end if
    x.subplan[i] ← YAHSP_Solve(x_i)
    if x_i not solved in b_{max} **then**
      s ← unfeasible
      break
    end if
  end for
  p ←YAHSP_Compress(x.subplan)
  u ← setObjectiveVector(x)
  updateStrategy(u)
  mutateStrategy()
  if (s = feasible **and** $u \underset{\lambda}{\succ} x$.objVector) **then**
    x.objVector ← u
    x.plan ← p
    x.state ← s
  end if
end for

On propose deux méthodes basées sur l'échantillonnage :

$$\begin{aligned} f_{k,\lambda} : X \times (\Omega, F, \mathbb{P})^k &\to \mathbb{R}^m \\ (x, (\omega_i)_{1 \le i \le k}) &\to \tilde{f}(x, \omega_i), \text{ tel que } \forall j,\ \lambda(\tilde{f}(x, \omega_i)) > \lambda(\tilde{f}(x, \omega_j)) \end{aligned}$$

$$\begin{aligned} \bar{f}_k : X \times (\Omega, F, \mathbb{P})^k &\to \mathbb{R}^m \\ (x, (\omega_i)_{1 \le i \le k}) &\to \bar{f}(x, \omega_i) = \frac{1}{k} \sum_i \tilde{f}(x, \omega_i) \end{aligned}$$

L'objectif de $f_{k,\lambda}$ est d'estimer le meilleur des plans qu'il est possible d'obtenir à partir d'un individu. Cependant, les expériences montrent qu'il est plus difficile d'atteindre les extrémités du nuage de points que les points plus au centre comme en atteste la figure 4.176 ou en générale, tous les résultats obtenus sur les instances larges, ce qui peut fausser la comparaison de la même manière que sur la figure4.171. Enfin, il est également difficile de qualifier le meilleur des vecteurs objectifs obtenus puisqu'ils peuvent ne pas être comparables au sens de la relation de dominance de Pareto.

Pour remédier à ces problèmes, $\bar{f}_k$ est construit pour être un estimateur du vecteur objectif espéré pour un individu donné, c'est à dire $\bar{f}_k \underset{k\to\infty}{\to} E[\tilde{f}(x)]$.

Si l'on considère un individu x et y, $E[\tilde{f}(x)] > E[\tilde{f}(y)]$, pour les même paramètres de b_{max} et le même objectif à optimiser, signifie que la structure de x est plus proprice à obtenir des plans dont le vecteur objectif sera plus faible [10].

Évidemment, $\bar{f}_k$ ne résulte surement pas d'un plan faisable et il ne faut pas mettre l'archive à jour avec ce vecteur ci.

Evaluation classique (1)

---

1. DAE appelle YAHSP sur x
2. x est affecté du vecteur objectif v du plan retourné par YAHSP
3. $A = A \cup \{v\}$

Evaluation échantillon (2)

---

1. Générer un échantillon $(v_i)_{1\leq i\leq k}$
2. x est affecté du vecteur objectif v selon $\bar{f}_k$ ou $f_{k,\lambda}$
3. $A = A \cup ND((v_i))$

En outre, avec (2), on change le problème et l'espace objectif de l'algorithme génétique. Avec (1) l'espace objectif est « l'espace des vecteurs objectifs des plans », le problème étant « trouver les plans Pareto-optimaux » alors qu'avec (2), c'est « l'espace de la

10. On pourrait également penser à utiliser un quantile pour forcer à garder les individus qui obtiennent des vecteurs objectifs qui se concentrent plus proche du front, ou encore la différence entre un quantile, par exemple le médian, et le vecteur espérance.

facilité de résolution d'un individu » et le problème est « trouver des individus qui sont le plus susceptibles de donner de bons plans »[11]. Cela me semble plus en accord avec la philosophie de DAE.

Le problème induit par (1) et que l'on cherche à résoudre (en effet, on cherche à obtenir avant tout les plans Pareto-optimaux), est noté (P) et le problème induit par (2) est noté ($\tilde{\text{P}}$). Il s'agit un problème connexe possédant son propre front de Pareto dont l'interprétation est plus abstraite.

Pour une même instance d'un problème de planification et un même algorithme génétique, le front de ($\tilde{\text{P}}$) dépend du solveur local utilisé.

Dans le cas d'un solveur déterministe et optimal Pour un objectif i d'optimisation du solveur, une seule itération est nécessaire pour l'évaluation exacte de $E[\tilde{f}(x)]_i$. En supposant $k = l$ et que l'on optimise tour à tour selon chaque objectif, on obtient l vecteurs objectifs comme décrit sur la figure 4.169.

v_{o_3} et v_{o_4} sont non-dominés, F est le centre de gravité de la famille des vecteurs objectifs obtenus et c'est le résultat de l'évaluation de x pour ($\tilde{\text{P}}$).

Si le front de Pareto de (P) ne possède pas de zone concave, $\forall x, \forall \tilde{p} \in \text{FP}(\tilde{P}),\ \exists p \in \text{FP}(P),\ p \succeq \tilde{p}$[12].

On a clairement $(P) \implies (\tilde{P})$. L'objectif serait d'étudier sous quelles conditions $(\tilde{P}) \implies P$.Dans un second temps il faudrait étendre l'étude au cas d'un solveur déterministe sous-optimal puis au cas d'un solveur stochastique sous-optimal. À défaut d'avoir le temps d'effectuer cette étude, quelques résultats empiriques seront présentés plus loin, qui semble montrer qu'il est possible d'obtenir des résultats similaires pour (P) en résolvant (P') plutôt que directement (P).

Utiliser un algorithme mono-objectif ? Un individu avec une bonne stratégie conduit à échantillonner au mieux la zone de l'espace objectif qu'il peut atteindre, comme montré sur la figure 4.173. On obtient donc un certain nombre de vecteurs objectifs non-dominés au sein de l'échantillon. L'idée est d'utiliser cette valeur d'hypervolume comme

11. La notion de facilité est ici définit par la valeur moyenne des vecteurs des plans obtenus, mais on pourrait imaginer d'autres métriques.

12. Il suffit de prendre un front concave en « U » pour (P) et considérer deux points sur les extrémités, le vecteur moyen se trouvera « derrière » le front de (P).

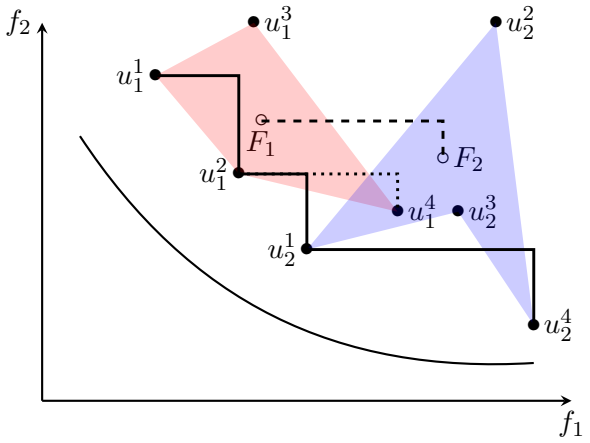


FIGURE 4.172: Correspondance front de (P) / front de ($\tilde{\text{P}}$). F_1 et F_2 sont respectivement les centres de gravité des polygones formés par les familles $(u_1^i)_i = E[\tilde{f}(x)]_i$ et $(u_2^i)_i = E[\tilde{f}(y)]_i$. Ils correspondent à l'évaluation de deux individus x et y. La ligne en pointillés représente le front de ($\tilde{\text{P}}$) en considérant ces deux individus. Le ligne continue est le front du problème (P). La ligne en points menant à u_1^4 montre que ce point faisait partie de l'archive locale lors de l'évaluation de x (c'est à dire les points non-dominés des points formant le polygone rouge).

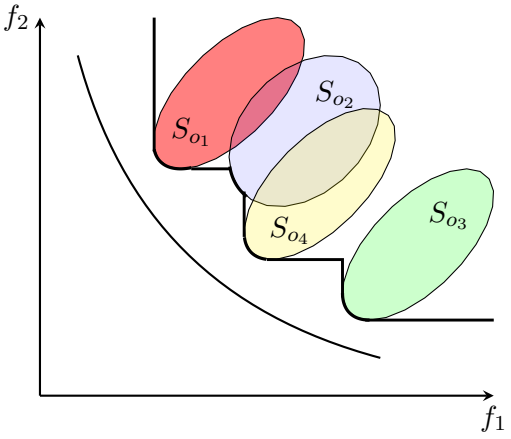


FIGURE 4.173: Étant donné un individu x dont on peut observer l'ensemble des zones atteignables (S_{o_i}), on a tracé en gras le front de Pareto que l'on pourrait théoriquement atteindre (en oubliant le fait que le problèle soit discret). Notons les différentes valeurs d'opacité : plus la valeur est faible moins la surface correspondante impacte l'hypervolume. Une bonne stratégie adaptive viserait à trouver distribution de probabilité p telle que l'on échantillonne en priorité selon les objectifs dont la zone accessible est intéressante.

fitness de l'individu x. Si l'hypervolume évolue positivement l'individu est probablement composé de meilleurs sous-problèmes : plus faciles à résoudre et menant à un plan total de meilleure qualité, ou propose une meilleure stratégie pour mieux échantillonner la zone accessible.

Si un individu est capable de donner de meilleurs vecteurs objectifs, son hypervolume sera meilleur qu'un individu qui structurellement ne peut pas mener à de meilleurs vecteurs objectifs, comme illustré sur la figure 4.174.

Évidemment, cela reste hypothétique puisque je n'ai pas eu le temps de faire de tests pour vérifier si cela pouvait conduire à de bons résultats, l'idée étant apparu en analysant les résultats des expériences réalisées. Ce qui serait intéressant cependant, c'est de pouvoir obtenir des résultats intéressants sur un problème multi-objectif à l'aide d'algorithmes mono-objectifs.

On donne tout de même la séquence d'évaluation (algorithme 4.8.2.1), basée sur celle de 4.8.2.1. Il faut bien évidemment continuer à mettre à jour l'archive au moment de l'évaluation avec les éléments non-dominés de l'archive.

Evaluation échantillon (2')

1. Générer un échantillon $(v_i)_{1 \leq i \leq k}$
2. $f(x) = H^+((v_i)_{1 \leq i \leq k})$
3. $A = A \cup ND((v_i))$

4.8.3 Méthodologie pour comparer l'effet de YAHSP ou de l'algorithme génétique

On cherche ici à estimer l'effet de YAHSP en comparaison à celui de l'algorithme génétique, sur la progression du processus de recherche, compte tenu d'une méthode d'affectation du vecteur objectif d'un individu. Pour cela, on considère l'évolution de l'hypervolume. La difficulté est de tenir compte de la stochasticité du solveur local et de la stochasticité de l'algorithme génétique. Pour ce faire, l'échantillonnage va aider à supprimer autant que possible l'effet de l'aléatoire du solveur local, tandis que de multiples *runs* permettront de tenir compte du comportement incertain de l'algorithme génétique.

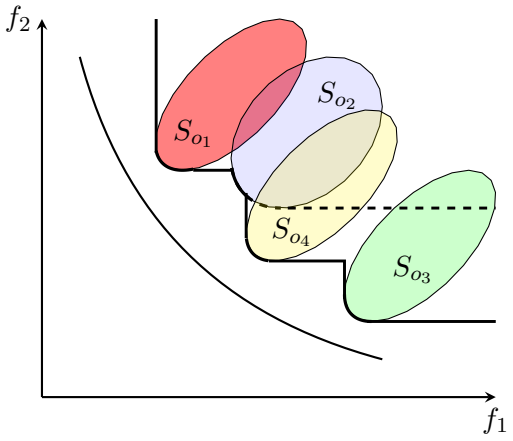


FIGURE 4.174: L'hypervolume tronqué en utilisant la ligne en pointillée correspond à un individu dont la stratégie ne permet d'utiliser que la l'objectif o_1 et o_2, c'est à dire que la probabilité de choisir o_3 ou o_4 est 0. L'individu x avec une telle stratégie aura donc une *fitness* inférieure au même individu utilisant la stratégie menant à l'hypervolume en ligne pleine. On peut généraliser le raisonnement à des individus différents (utilisant ou non la même stratégie.

Pour cela on redéfinit une nouvelle fois le processus d'évaluation de YAHSPet on la décline en deux versions décrites par l'algorithmes 9 et 10. A_g est l'archive globale du processus d'optimisation, A_1 une archive locale, concaténation de l'archive globale et des points non-dominés de l'échantillon. Pour un $k$suffisamment grand, l'échantillon tend à supprimer l'aléatoire du solveur local en tirant parti au maximum de ses capacités. Ainsi, si la différence $H(A_1 - A_g)$ vaut 0 aucun nouveau point s'est ajouté à l'archive globale. Si au contraire la différence est positive, A_1 contient de nouveaux points non-dominés et l'échantillonnage a permis d'améliorer l'archive globale.

A_2 considère uniquement le premier vecteur objectif trouvé et est donc représentatif d'une utilisation minimale du solveur local et à priori plus sensible à ses aléas. Si la différence $H(A_2 - A_g)$ est positive, cela signifie que le vecteur u_1 n'est pas dominé.

On notera $H^1(A_1 - A_g)$ et $H^2(A_1 - A_g)$, la différence obtenue respectivement pour la première et la seconde version. On indicera de la même manière la seconde différence.

Algorithm 9: Version 1

Générer un échantillon $(u_i)_{1 \leq i \leq k}$
$A_1 = A_g \cup ND((u_i))$
$A_2 = A_g \cup \{u_1\}$
Afficher $H(A_1 - A_g)$ et $H(A_2 - A_g)$
$A = A_1$
Affecter le vecteur ojectif de l'individu

Algorithm 10: Version 2

Générer un échantillon $(u_i)_{1 \leq i \leq k}$
$A_1 = A_g \cup ND((u_i))$
$A_2 = A_g \cup \{u_1\}$
Afficher $H(A_1 - A_g)$ et $H(A_2 - A_g)$
$A = A_2$
Affecter le vecteur ojectif de l'individu

En étudiant les courbes $H^1(A_1 - A_g)$ et $H^2(A_1 - A_g)$ au cours du temps, on peut évaluer les capacités du solveur local sur l'amélioration de l'hypervolume. Si le solveur local est capable de faire progresser l'hypervolume fréquemment alors $H^1(A_1 - A_g)$ proposera une fréquence d'amélioration et une amplitude moins importante que $H^2(A_1 - A_g)$ puisque dans le premier cas on utilise pleinement les capacités du solveur local et dans le second, on a une utilisation minimale.

NOTE : Je n'ai pas encore trop réfléchi comment réellement comparer statistiquement ces courbes ni à toutes les interprétation selon les courbes d'hypervolume. Cela fera certainement l'objet d'un travail futur.

4.8.4 Vérifications expérimentales

L'échantillonnage permet d'explorer explicitement certaines zones de l'espace objectif. Comme en témoigne les figures 4.175 et 4.176, la zone accessible d'un individu dans l'espace objectif dépend fortement de l'objectif passé à YAHSP.

Sur la figure 4.175 il est intéressant de noter que la zone échantillonnée est la même avec un $b_{max} = 100000$ ou $b_{max} = 10000$. Cependant, la différence avec un b_{max} très faible n'est pas flagrante, les fronts étant très proches, ce qui confirme qu'il est possible de trouver des plans avec un effort très faible, tout en échantillonnant pratiquement aussi bien.

La figure 4.176 permet d'observer l'influence de la stratégie de YAHSP sur la zone atteignable par un même individu. Il est intéressant de noter que sur cet exemple les zones sont complémentaires et permettent d'échantilloner une grande partie de l'espace objectif.

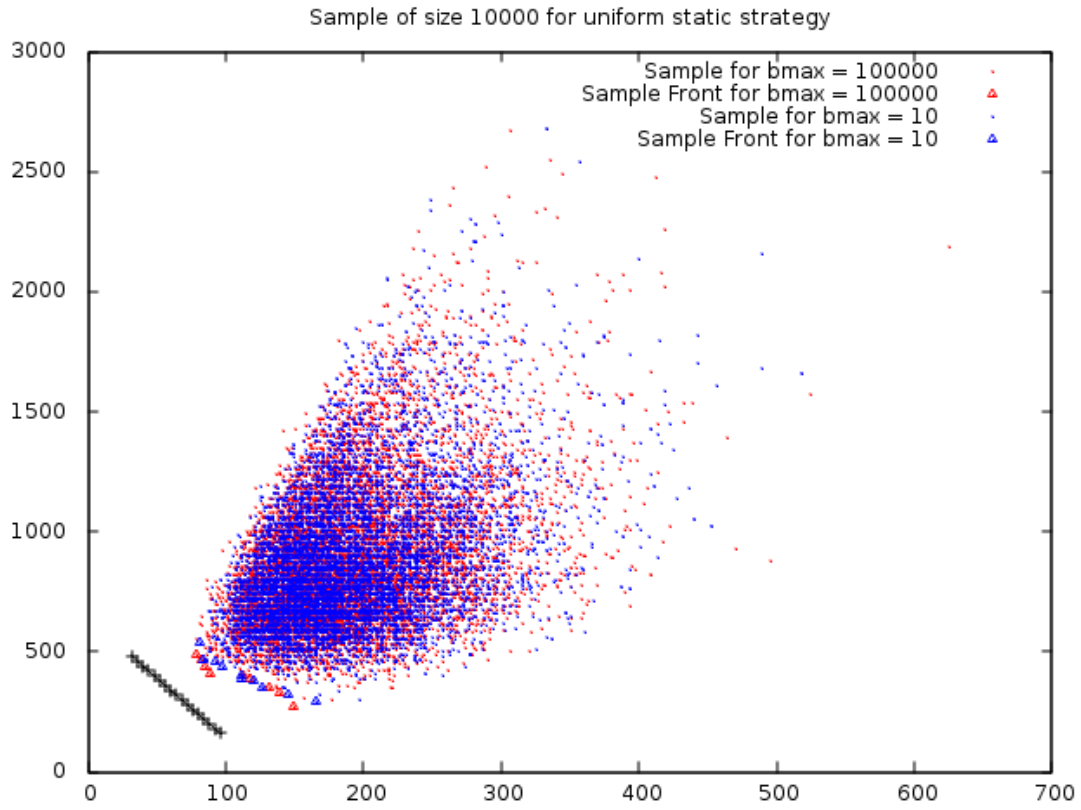


FIGURE 4.175: Zone échantillonnée pour différentes valeurs de b_{max} et pour une stratégie uniforme. L'échantillon est composé de 10000 points.

Sur la figure 4.177 on observe l'échantillon de deux individus. Il semble logique de préférer l'individu 2 à l'invidivu 1 puisqu'il permet d'atteindre de meilleurs plans en règle générale : le centre de gravité du nuage de 1 est dominé par celui de 2.

Sur toutes les figures suivantes, le vecteur objectif assigné à un individu est le meilleur des vecteurs de l'echantillon selon l'indicateur $\lambda^j_{\Delta^+}(x) = \sum_i \Delta f_i^j(x)$. Il est probable que les résultats soient encore plus éloquents en prenant le premier vecteur objectif trouvé lors de l'échantillonnage.

Les graphiques 4.178 montrent l'évolution de la différence d'hypervolume entre l'archive globale et l'archive concaténée. Si le résultat est 0 selon signifie que l'archive globale domine l'archive locale. Lorsque cette différence est négative cela signifie qu'au moins un point de l'archive locale n'est dominé par aucun point de l'archive globale et indique donc que l'échantillon a participé à améliorer le front.

Sur la figure 4.178 on voit la fréquence de découverte de points non-dominés au cours des générations et en fonction de la taille de population. Plus le pic est important

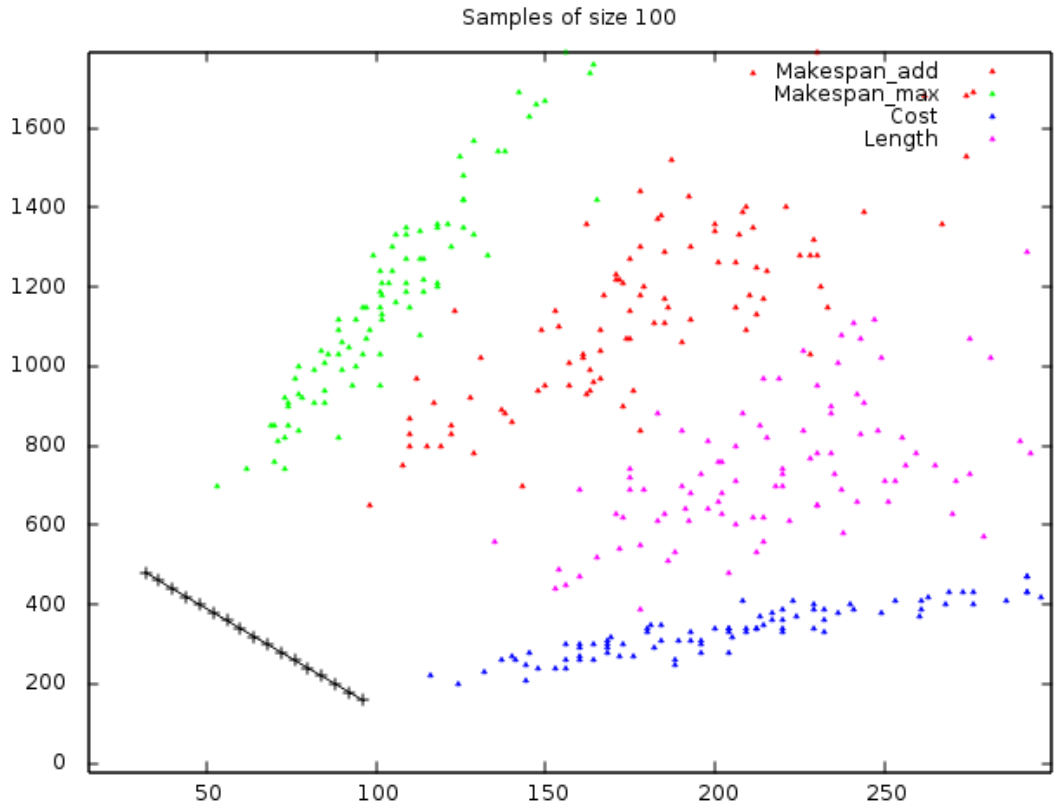


FIGURE 4.176: Différence d'hypervolume entre archive locale+globale et archive globale, sans mise à jour de l'archive globale et pour différente taille de population.

et plus les points non-dominés de l'échantillon sont proches du front réels. L'archive globale n'est pas mise à jour entre deux générations en incluant les points non-dominés de l'archive locale. On remarque que la taille de la population joue un rôle important dans la fréquence et l'amplitude des pics d'hypervolume. Avec un unique individu, les sauts sont fréquents et leur amplitude importante, alors qu'avec 30 individus, ils sont plus rares et d'amplitude plus mesurée. Enfin, pour 300 individus, leur fréquence est quasi-insignifiante.

La figure 4.179 montre toujours la différence d'Hypervolume entre l'archive locale+globale et l'archive locale, d'une part avec mise à jour après chaque évaluation de l'archive et sans mise. Peu importe la taille de la population, la fréquence et l'amplique des pics diminuent avec la mise à jour. Si l'on considère qu'utiliser une valeur assez grande de k_{max} permet de tirer le « meilleur » d'un individu avec une probabilité proche de 1, alors les pics d'hypervolume permettent de mesurer la capacité de convergence de l'algorithme vers le front exact en retirant la composante stochastique de la fonction d'évaluation.

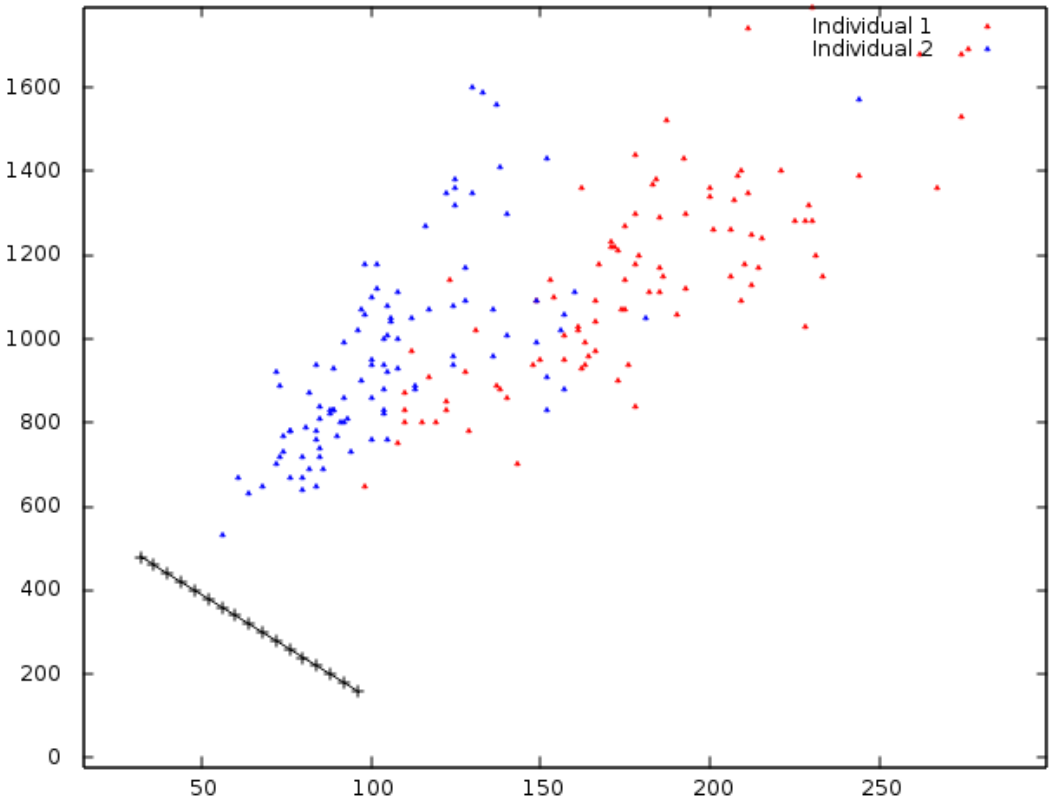


FIGURE 4.177: Echantillons de taille 100 pour deux individus.

Pour une même valeur de k_{max}, la probabilité de découvrir de nouveaux points non-dominés diminuent

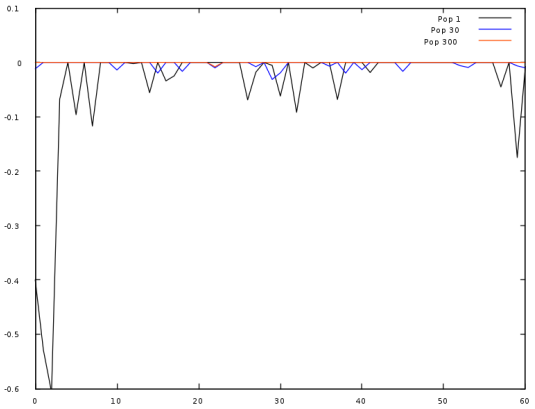


FIGURE 4.178: Différence d'hypervolume entre archive locale+globale et archive globale, sans mise à jour de l'archive globale et pour différente taille de population.

La figure 4.179 est particulièrement intéressante. Notons l'abscisse dont l'unité est la génération et pas le temps pour ne pas biaiser la comparaison. Pour une population réduite à un seul individu, on note que l'effet de l'algorithme génétique se limite à des mutations puisqu'on a désactivé les croisements. Le fait que la courbe orange montrent

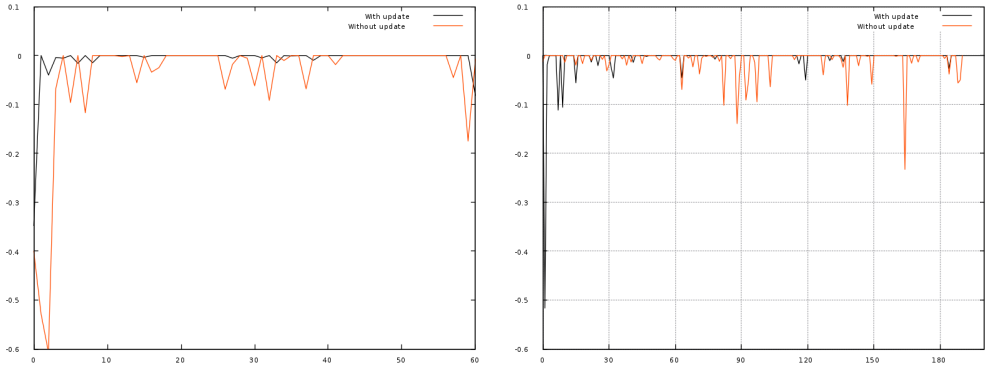


FIGURE 4.179: Différence d'hypervolume entre archive locale+globale et archive globale, avec et sans mise à jour de l'archive globale (gauche population de un individu, 30 à droite).

plus de sauts et d'une plus grande amplitude indique que l'échantillon aurait été susceptible d'améliorer l'archive locale assez souvent et de manière très significative. A contrario, si l'on met à jour l'archive globale avec les individus non-dominés de l'échantillon, on améliore très rarement l'hypervolume. Si l'on estime que la structure d'un individu n'a pas changé radicalement à cause des mutations, cela tendrait à montrer que les mutations ne sont pas assez fortes pour changer la significativement la zone des vecteurs objectifs accessibles d'un individu.

On retrouve le même phénomène à gauche, avec une population de 30 individus, comprenant cette fois des croisements. Là encore, les changements dans l'hypervolume grâce à l'échantillon sont assez importants lorsque l'archive n'est pas mise à jour et d'amplitude et de fréquence faible avec mise à jour. On peut raisonnablement penser que l'effet de YAHSP domine l'effet de l'algorithme génétique puisque lorsque l'on tire pleinement parti de YAHSP, on gardant l'ensemble des solutions non-dominées qu'il trouve, l'hypervolume évolue moins fréquemment et de manière moins visible. Au contraire, si l'on laisse le processus utiliser le moins possible YAHSP, les améliorations de l'hypervolume si l'on avait pleinement tiré parti de YAHSP sont très fréquente et d'amplitude importante.

On observe sur la figure **??** que les sauts d'hypervolume sont d'amplitude moins importante lorsque l'on met à jour l'archive globale avec l'archive locale que si l'on met à jour qu'avec le premier individu. On retrouve également ce constat avec la stratégie auto-adaptive sur la figure 4.181. Plus précisemment, on observe une diminution de la fréquence des sauts au cours des générations avec la mise à jour ce qui tenderait à montrer que l'échantillonnage a un effet très bénéfique sur l'hypervolume, peu importe la

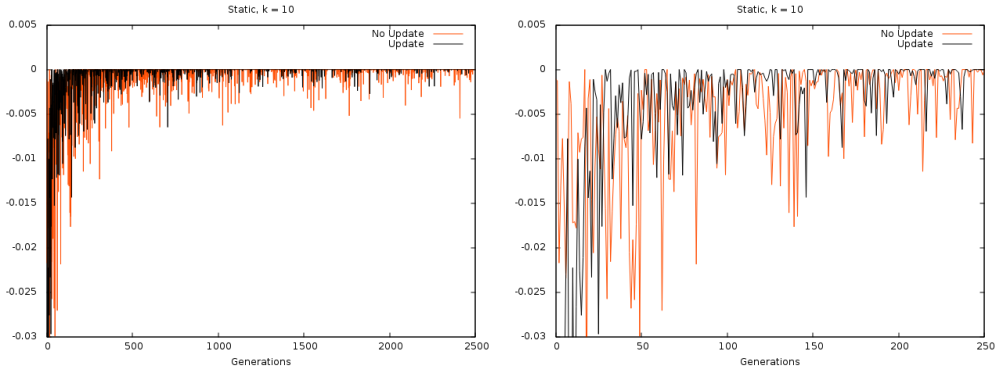


FIGURE 4.180: Moyenne de la différence d'hypervolume entre archive locale+globale et archive globale pour la stratégie statique sur 20 runs.

stratégie. Cette impression se précise un peu plus avec la figure 4.182 où l'on effectue un échantillonnage de taille 100.

Enfin, pour faciliter la comparaison sur la taille de l'échantillon, la figure 4.183 superpose les courbes de différence d'hypervolume avec mise à jour pour une taille d'échantillon de 10 puis 100.

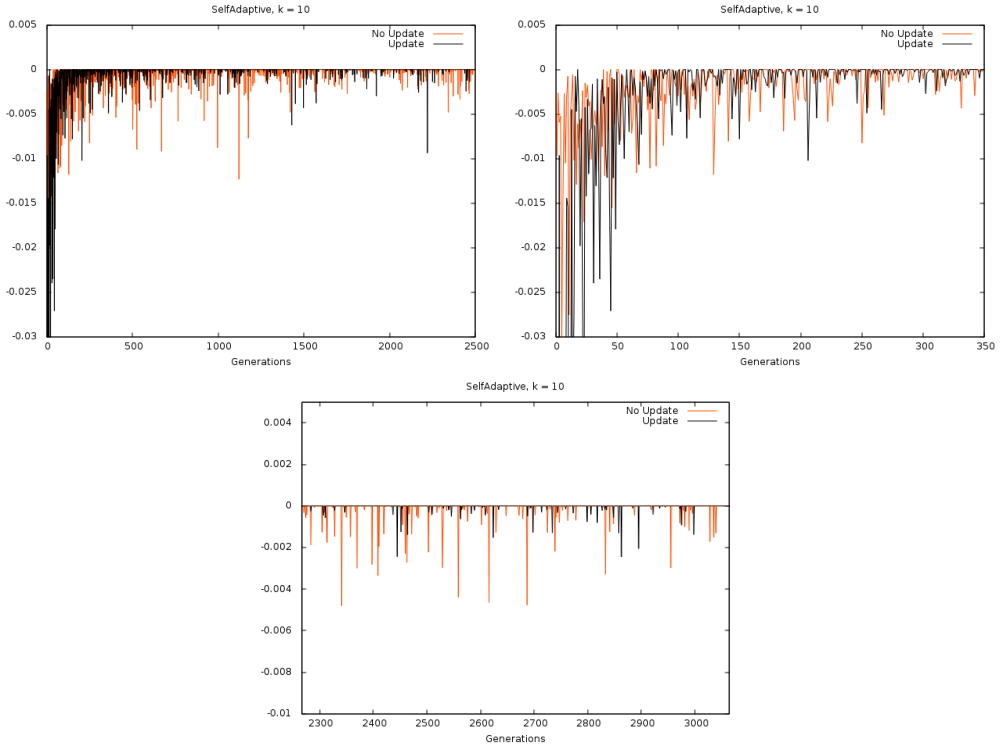


FIGURE 4.181: Moyenne de la différence d'hypervolume entre archive locale+globale et archive globale pour la stratégie auto-adaptative sur 20 runs.

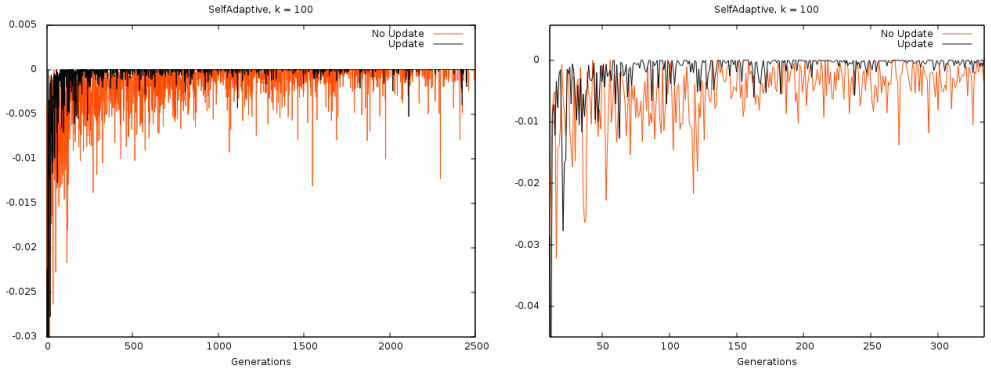


FIGURE 4.182: Moyenne de la différence d'hypervolume entre archive locale+globale et archive globale pour la stratégie auto-adaptative sur 20 runs avec une taille d'échantillon de 100.

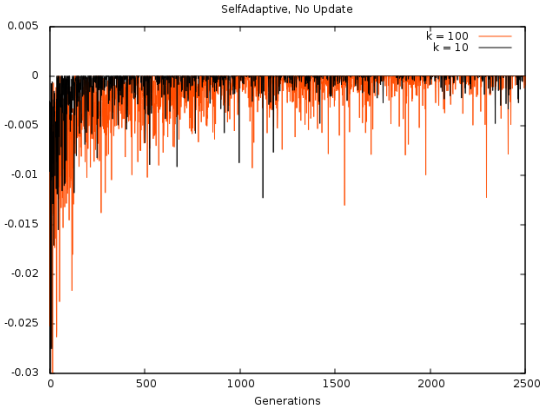


FIGURE 4.183: Moyenne de la différence d'hypervolume entre archive locale+globale et archive globale pour la stratégie auto-adaptative sur 20 runs et pour différentes tailles d'échantillon.

4.8.5 Résultats empiriques

On présente ici quelques résultats préliminaires sur l'échantillonnage et le fait de mettre à jour l'archive. On teste deux versions, l'une avec une stratégie statique, l'autre avec une stratégie auto-adaptative. On fait varier la taille de l'échantillon (10 et 100). Enfin, on présente pour chacune des stratégies et une taille d'échantillon de 10, des *runs* où le vecteur objectif d'un individu est affecté comme le vecteur empirique moyen de l'échantillon.

L'instance de référence est Zeno 9, avec le même protocole que précédemment. Seul le vecteur retenu est affiché et non l'ensemble des vecteurs de l'échantillon.

4.8.5.1 Série 1 : Stratégie statique, 10-échantillon

En comparaison avec les fronts cumulés obtenu avec un 1-échantillon, on obtient une meilleure répartition, totalement uniforme sur l'ensemble du front. On retrouve cette propriété sur la figure 4.185 qui montre la population cumulée et semble bénéficier d'une plus grande densité que pour le 1-échantillon. Cependant, notons que la zone échantillonnée est plus diffuse et les valeurs des vecteurs objectifs trouvés s'éloignent davantage.

Comme pour la stratégie auto-adaptative avec un 1-échantillon, les vecteurs objectifs extrêmes semblent plus difficile à atteindre, tandis que les zones centrales semblent plus favorisées comme en atteste les figures 4.191, 4.192 et 4.193 . On ne retrouve pas non plus le saut d'hypervolume caractéristique, observé sur le 1-échantillon de pour la stratégie statique.

Le test statistique sur les fonctions d'atteinte rejette l'hypothèse nulle : les deux séries ne sont pas identiques.

FIGURE 4.184: Instance Zeno9 : Fronts de Pareto cumulés pour tous les runs, à la fin de chaque run.

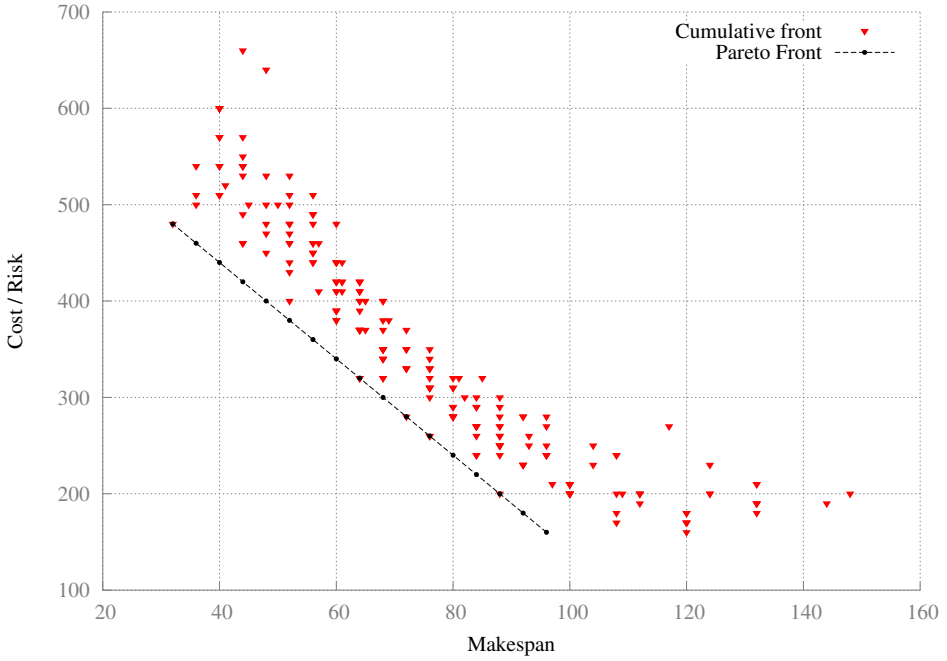


FIGURE 4.185: Instance Zeno9 : Population cumulées pour tous les runs et à tout temps.

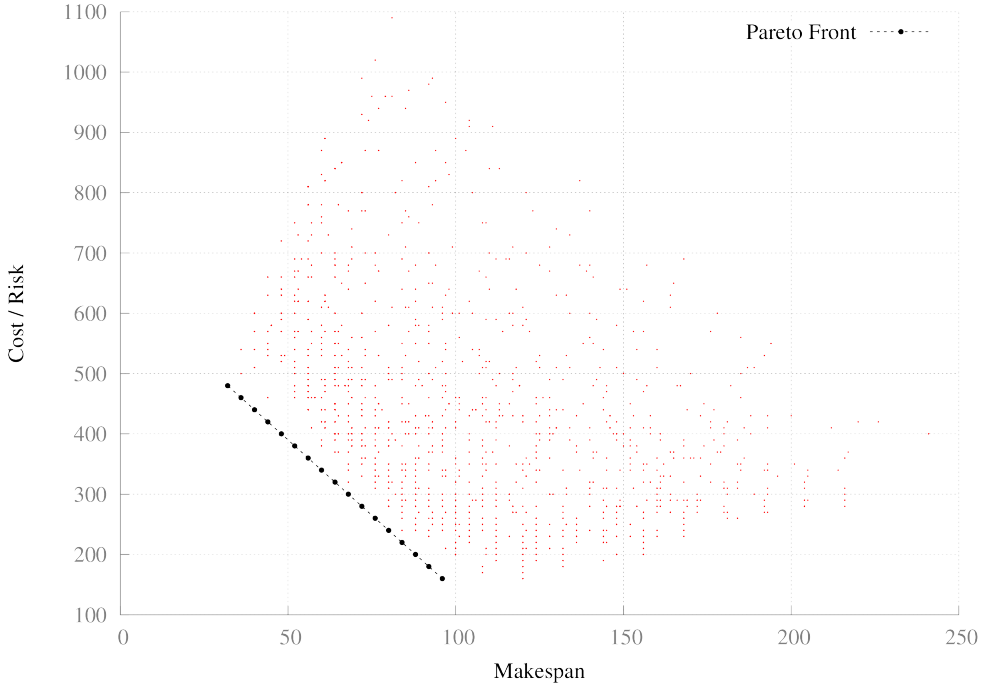


FIGURE 4.186: Instance Zeno9 : Population cumulées pour tous les runs et à tout temps (version colorée).

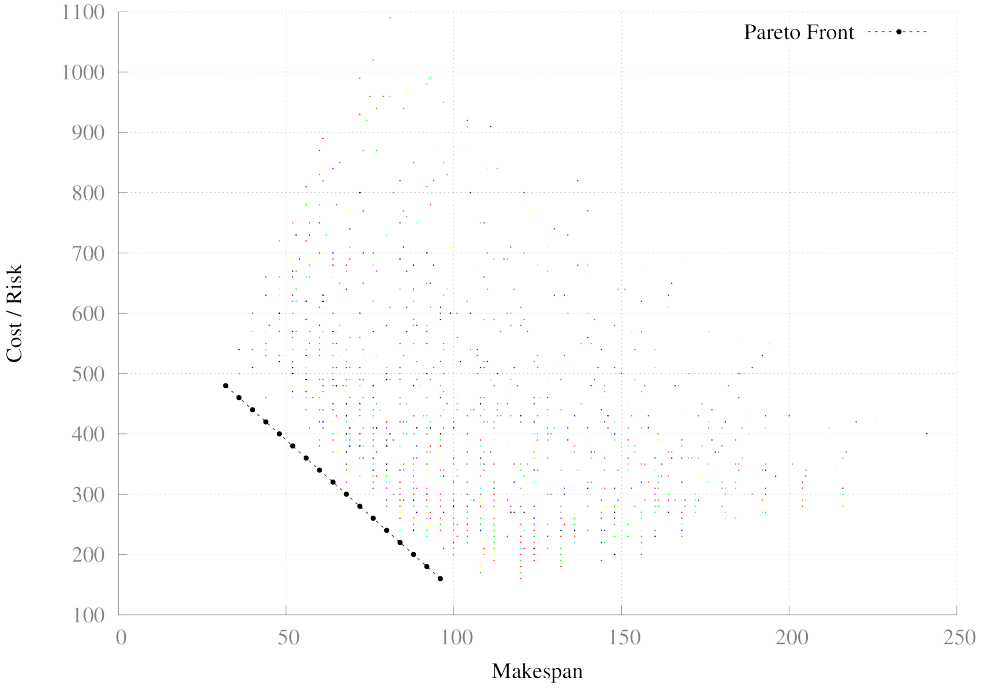


FIGURE 4.187: Instance Zeno9 : Surfaces d'atteinte pour tous les runs.

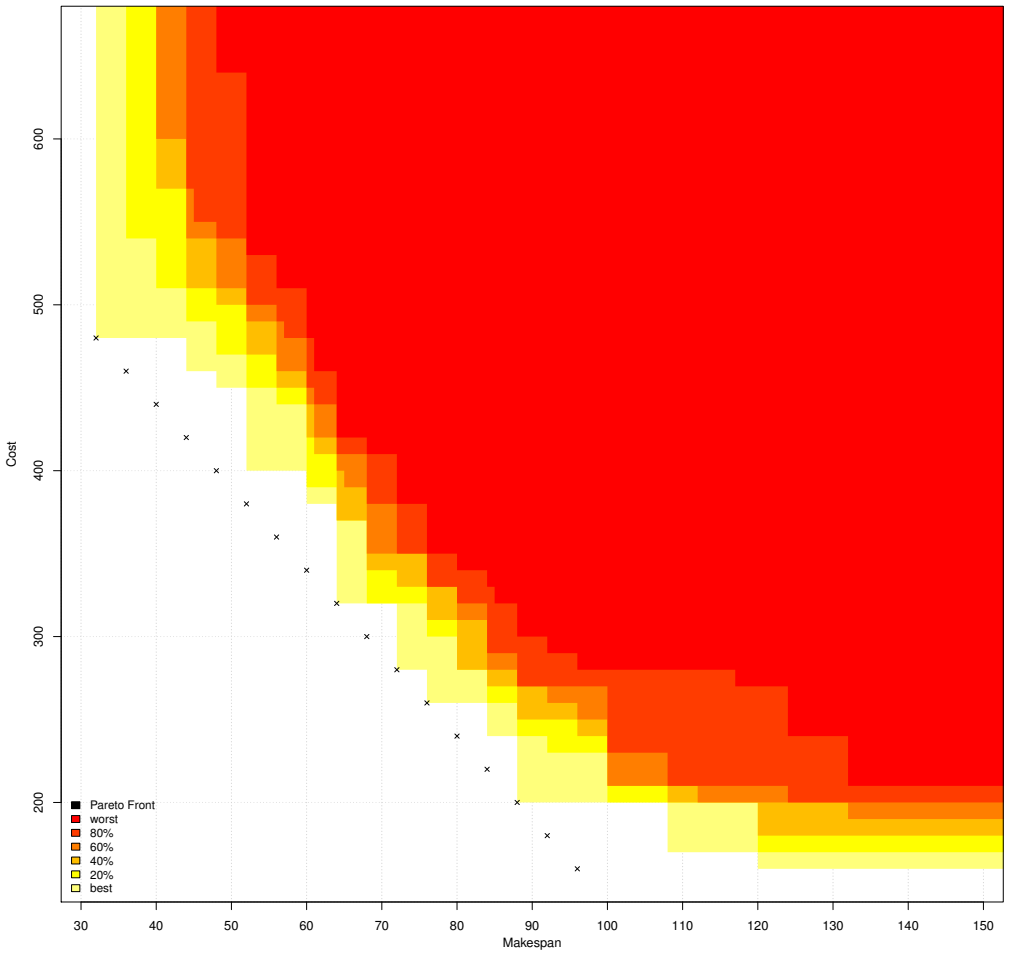


FIGURE 4.188: Instance Zeno9 : Trajectoires de l'hypervolume pour tous les runs.

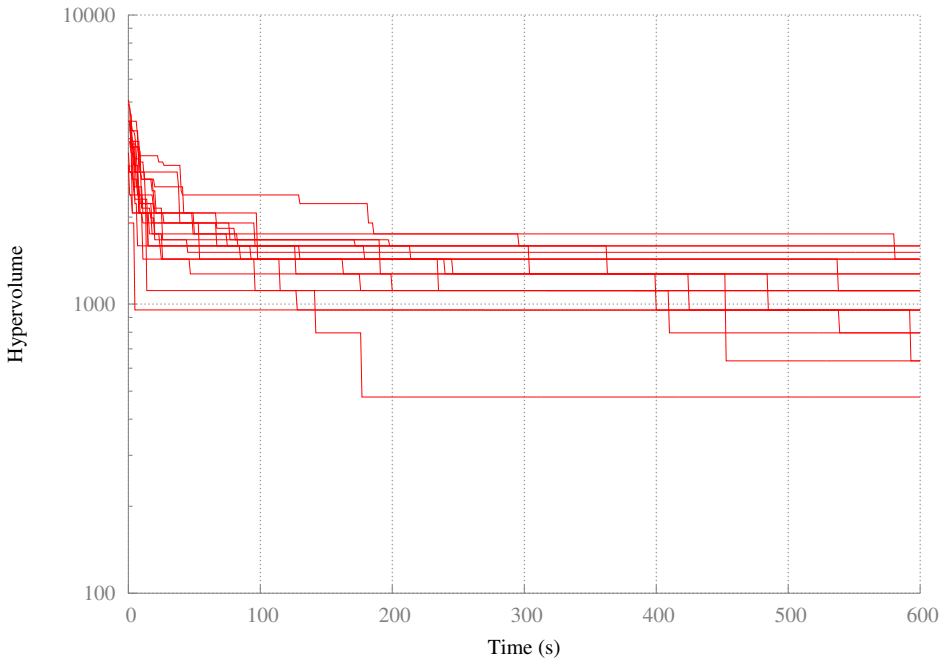


FIGURE 4.189: Instance Zeno9 : Diversité des vecteurs objectifs pour tous les runs.

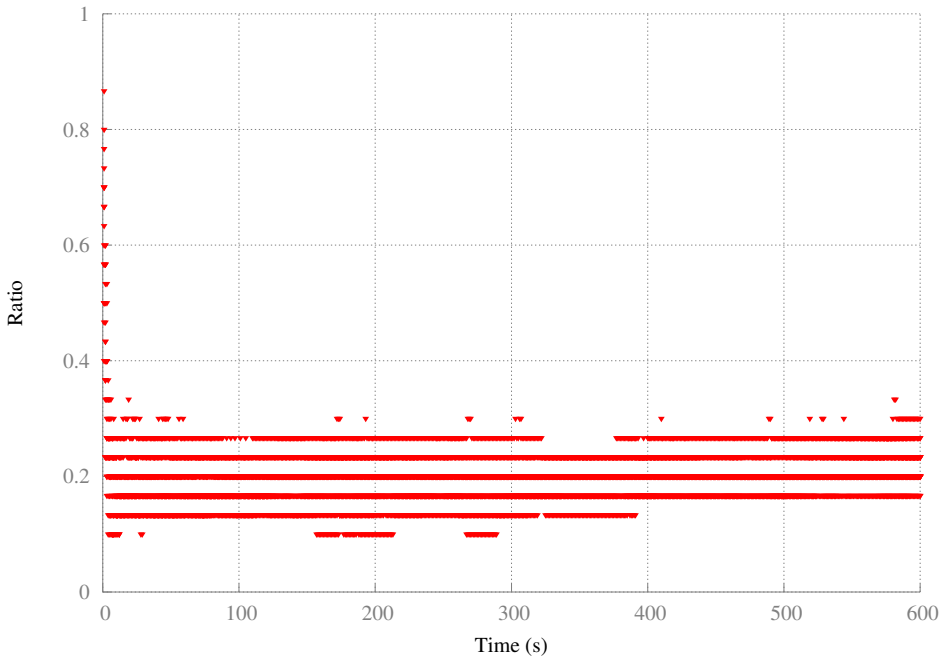


FIGURE 4.190: Instance Zeno9 : Diversité des vecteurs objectifs pour tous les runs (version colorée).

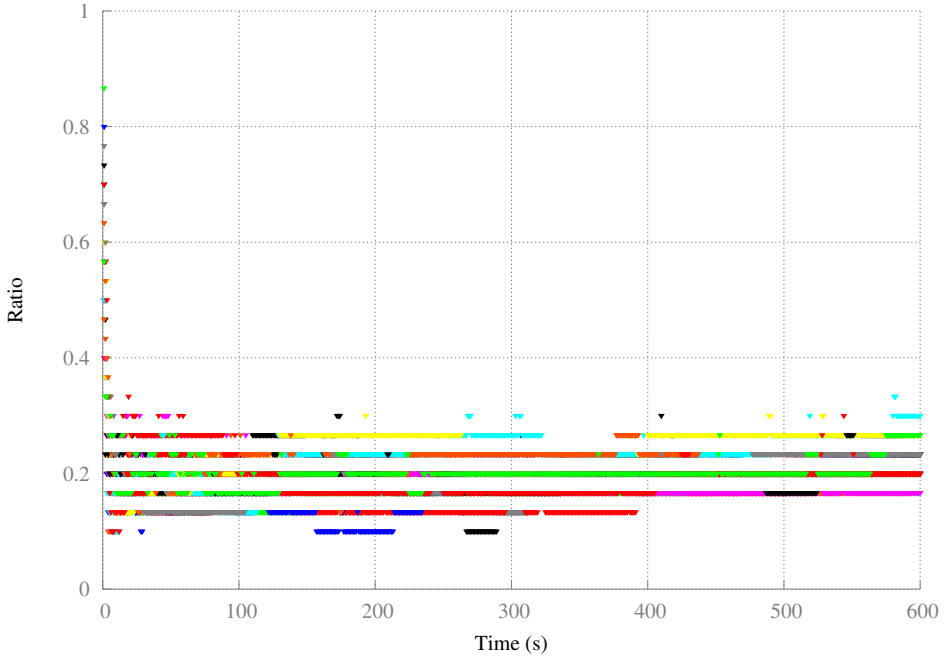


FIGURE 4.191: Instance Zeno9 : Comparatif des surfaces d'atteinte.

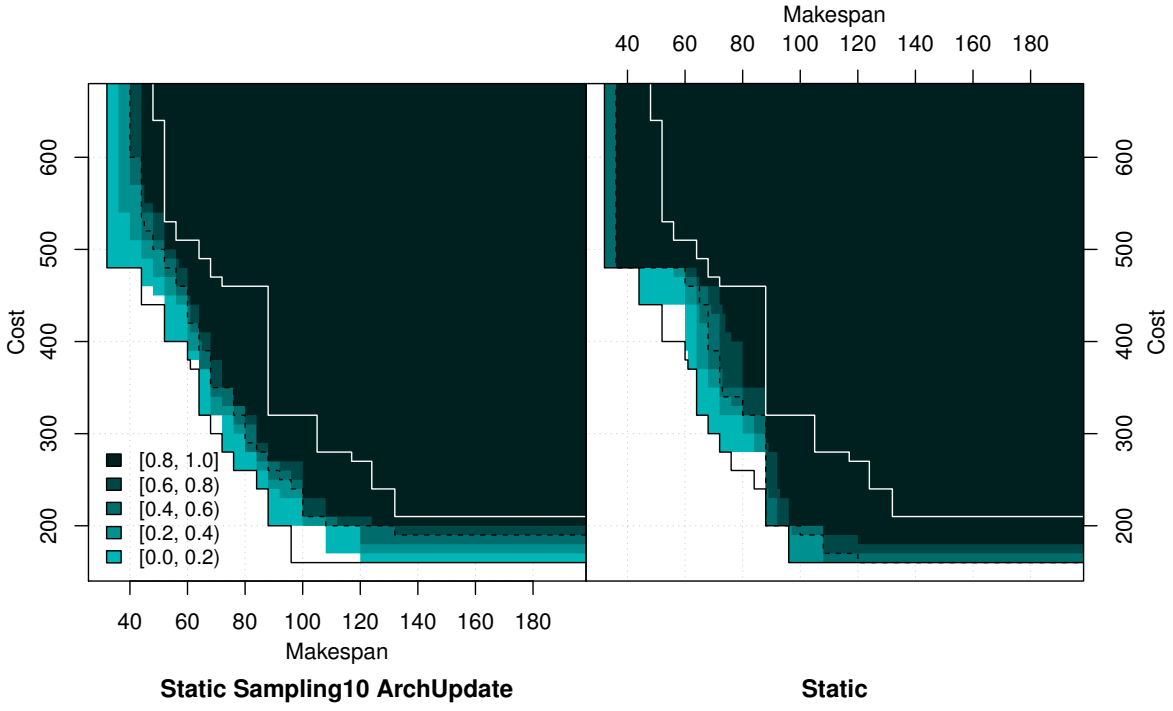


FIGURE 4.192: Instance Zeno9 : Adaptative comparée à Statique.

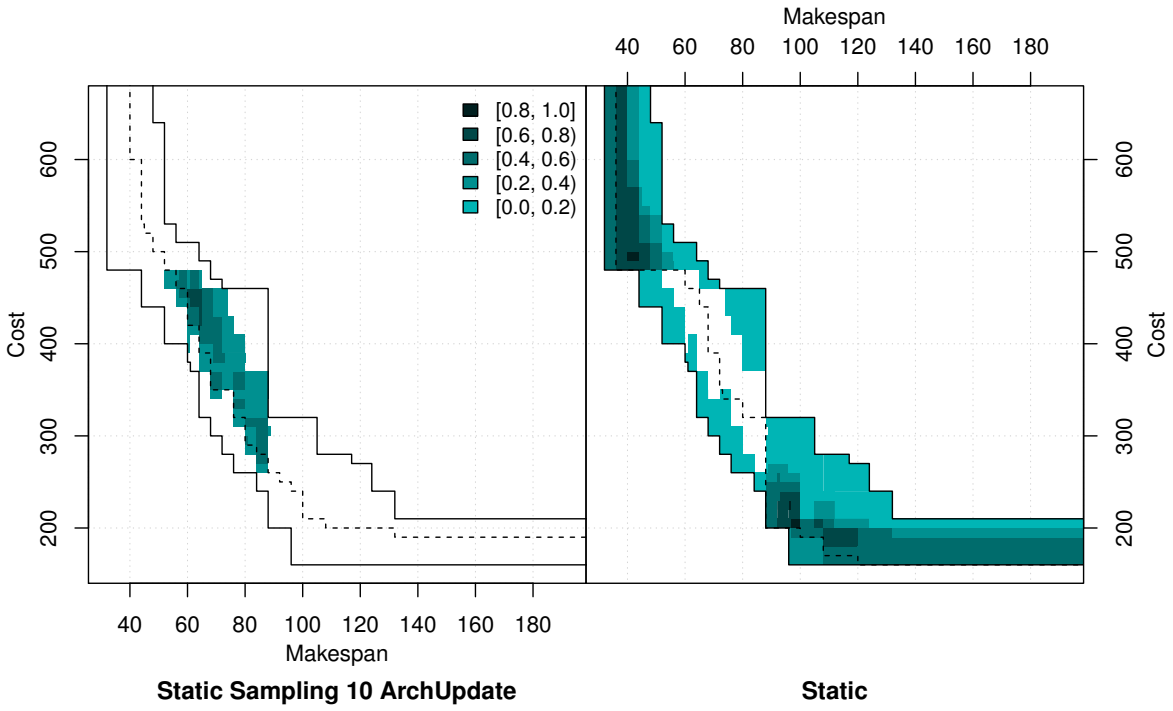
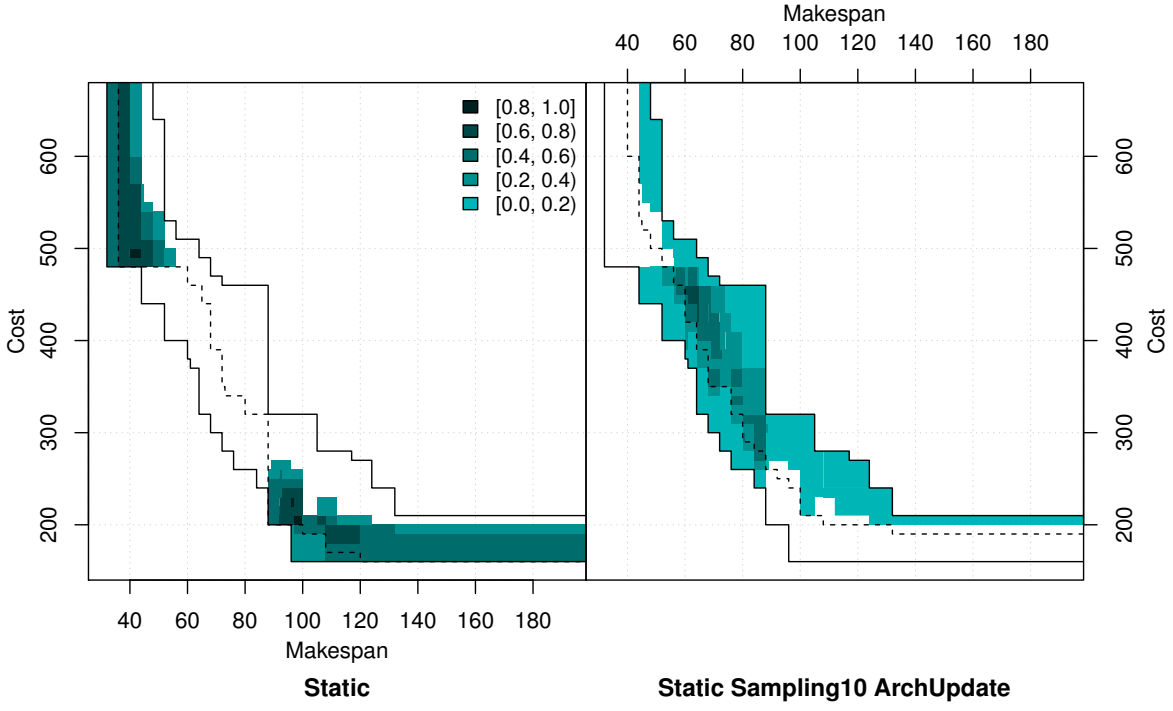


FIGURE 4.193: Instance Zeno9 : Statique comparée à Adaptative.



4.8.5.2 Série 2 : Stratégie statique, 100-échantillon

En comparaison, on a ici de nouveau des fronts cumulés plus uniformes que pour le 1-échantillon. Par rapport au 10-échantillon, on note une plus grande proximité avec le front exact. Notons une faiblesse apparente des valeurs de faible *makespan* sur la figure 4.195 de la population cumulée.

Par rapport au 10-échantillon, les surfaces d'atteinte ne semble pas avoir énormemment évoluées même si l'on note quelques améliorations. L'hypervolume quand à lui descend plus lentement du fait d'une évaluation en moyenne 10 fois plus longue par individu. Il est donc intéressant de constater que malgré un facteur 10 sur le temps d'évaluation, les surfaces d'atteinte et les résultats globaux de la série sont identiques à ceux de 10-échantillon, comme l'en atteste le teste statistique sur les fonctions d'atteinte.

FIGURE 4.194: Instance Zeno9 : Fronts de Pareto cumulés pour tous les runs, à la fin de chaque run.

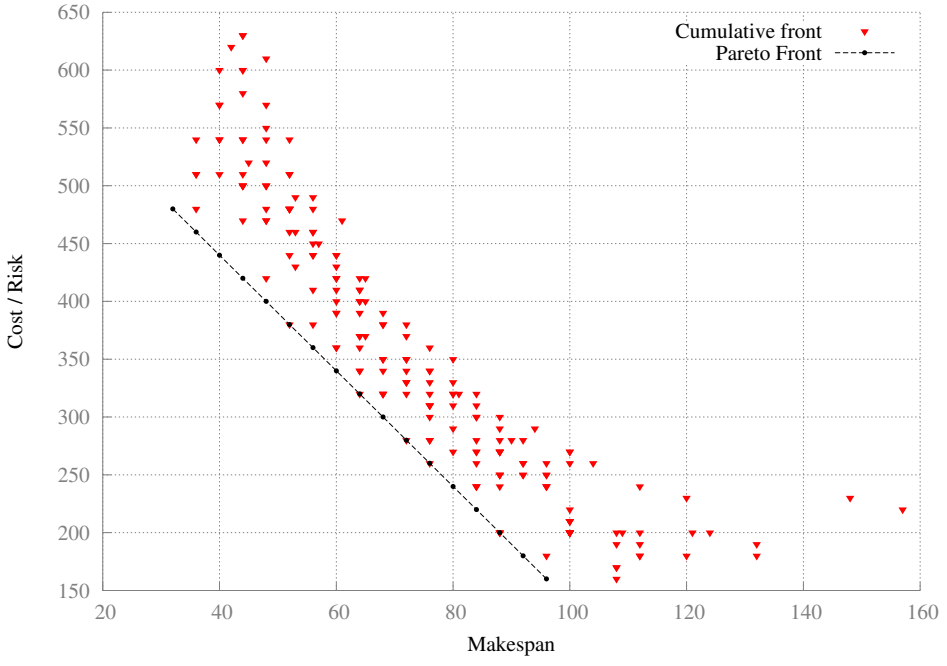


FIGURE 4.195: Instance Zeno9 : Population cumulées pour tous les runs et à tout temps.

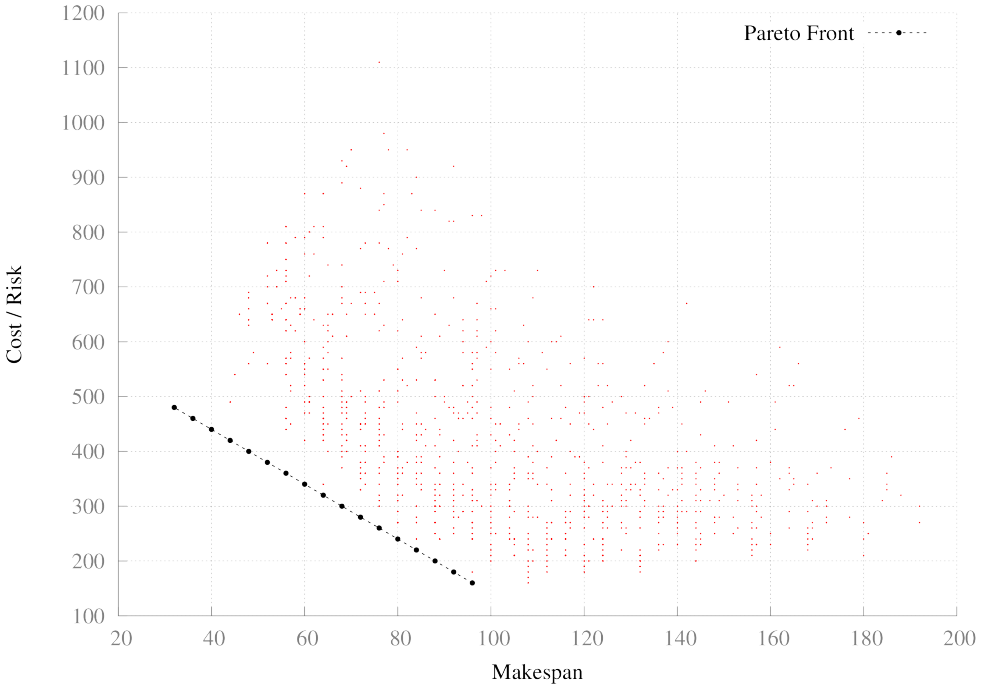


FIGURE 4.196: Instance Zeno9 : Population cumulées pour tous les runs et à tout temps (version colorée).

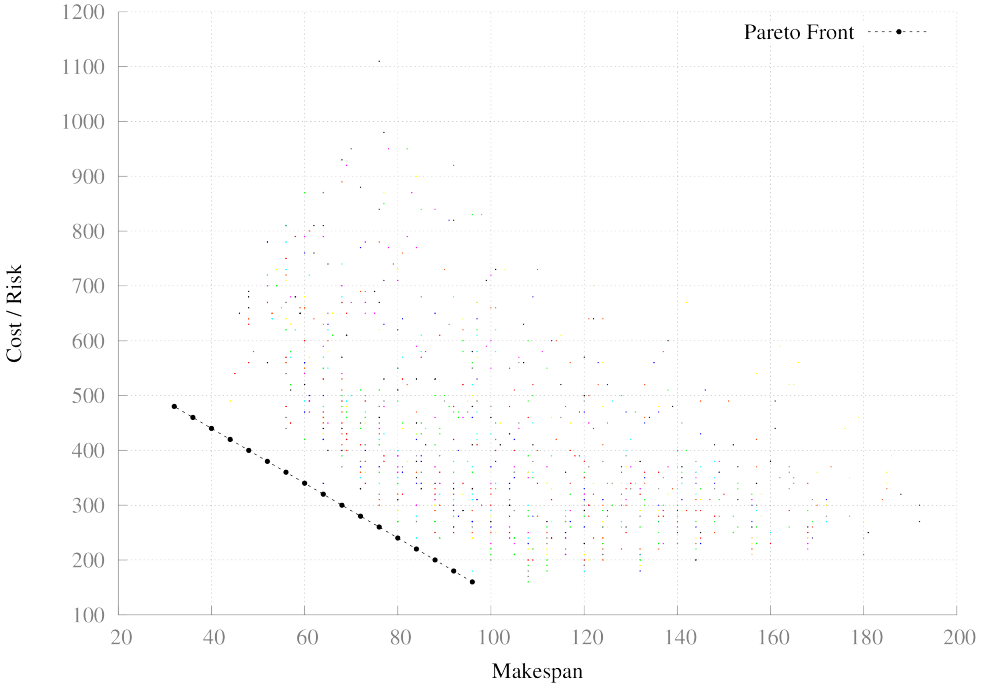


FIGURE 4.197: Instance Zeno9 : Surfaces d'atteinte pour tous les runs.

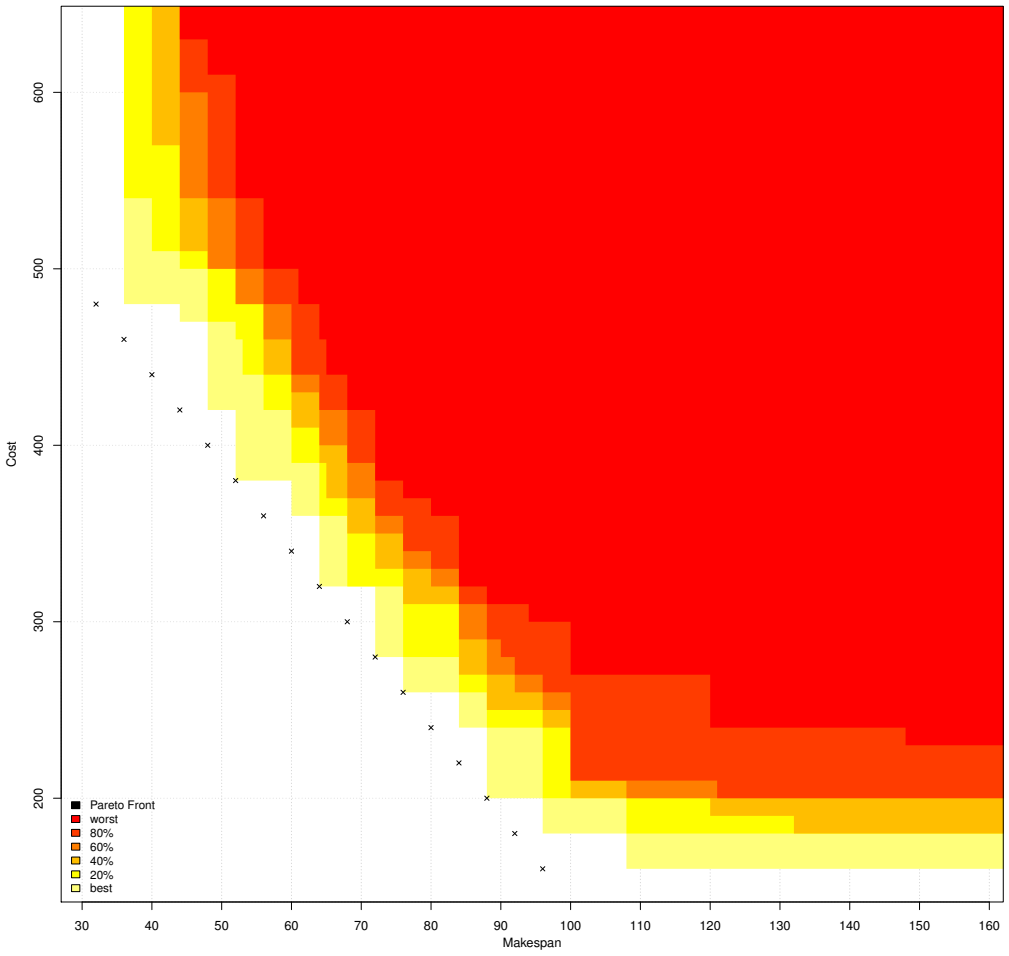


FIGURE 4.198: Instance Zeno9 : Trajectoires de l'hypervolume pour tous les runs.

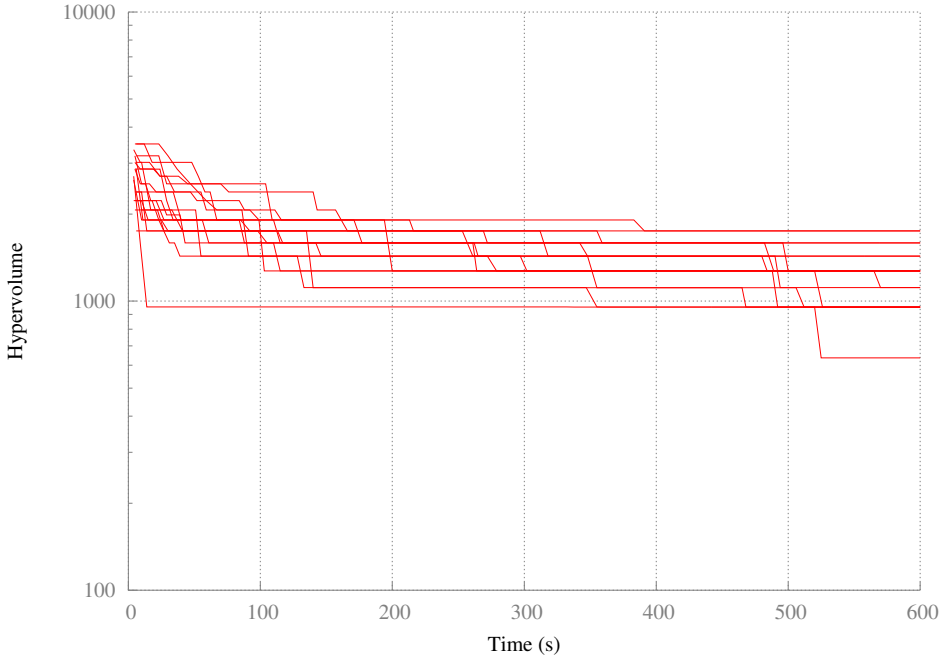


FIGURE 4.199: Instance Zeno9 : Diversité des vecteurs objectifs pour tous les runs.

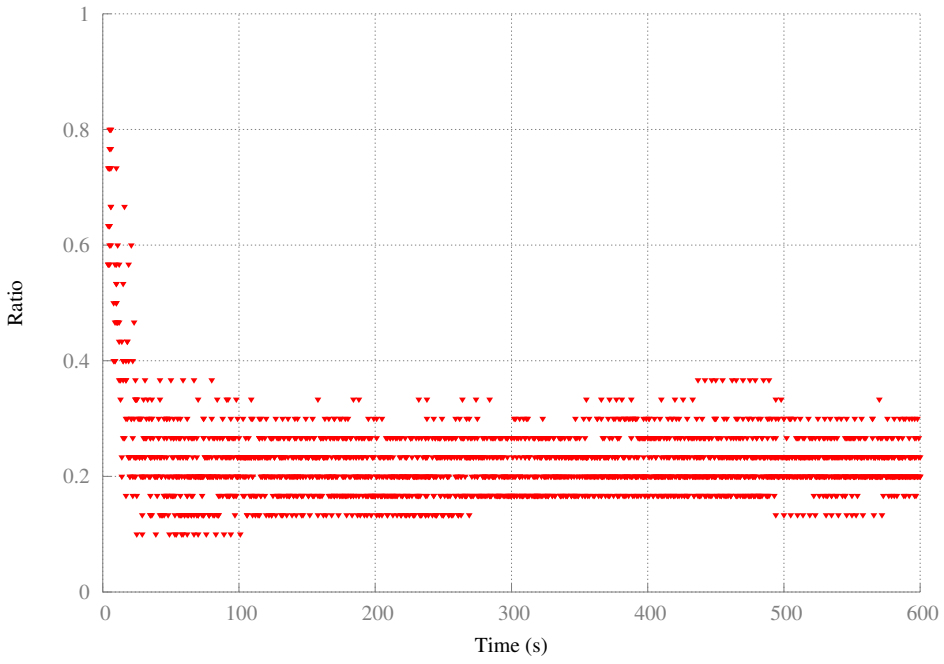


FIGURE 4.200: Instance Zeno9 : Diversité des vecteurs objectifs pour tous les runs (version colorée).

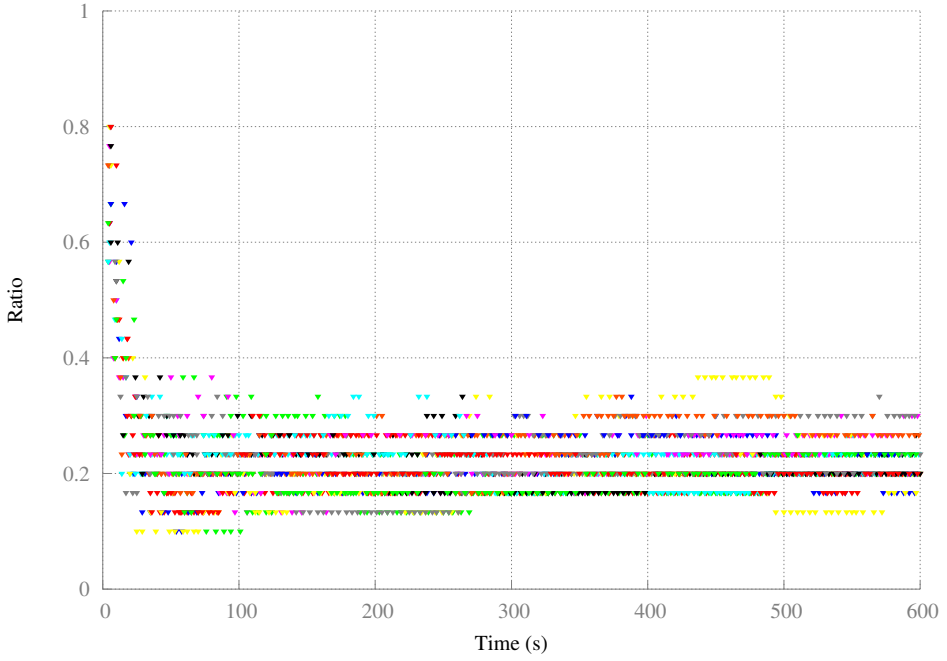


FIGURE 4.201: Instance Zeno9 : Comparatif des surfaces d'atteinte.

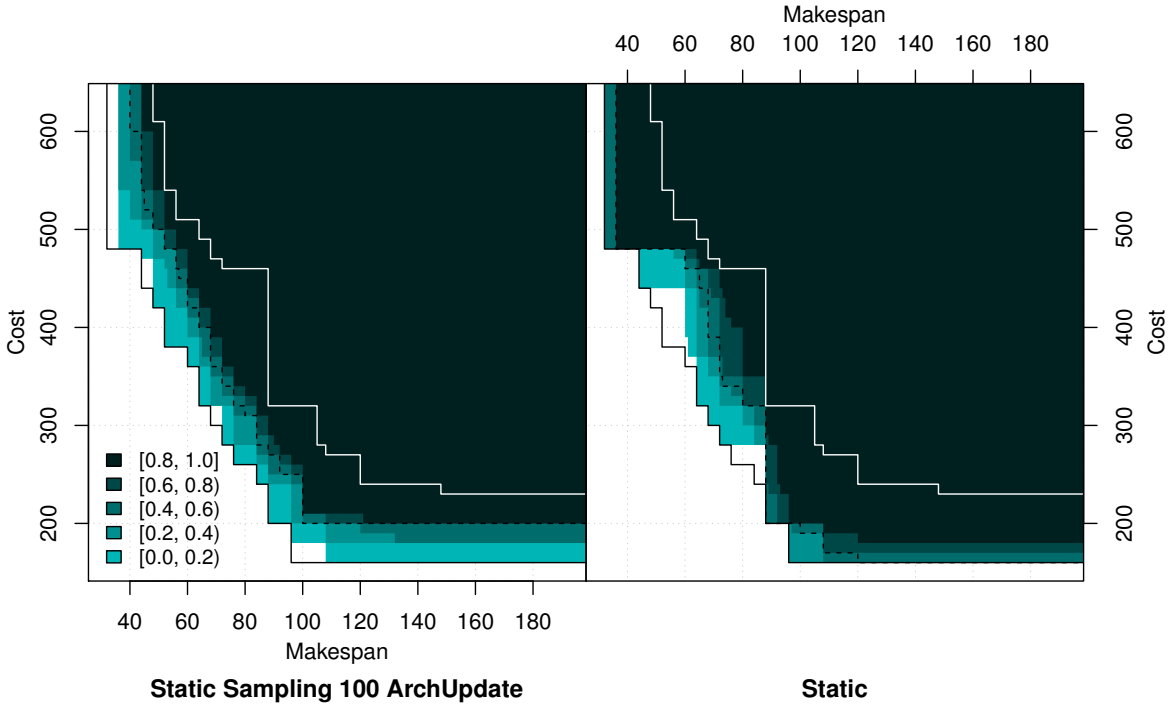


FIGURE 4.202: Instance Zeno9 : Adaptative comparée à Statique.

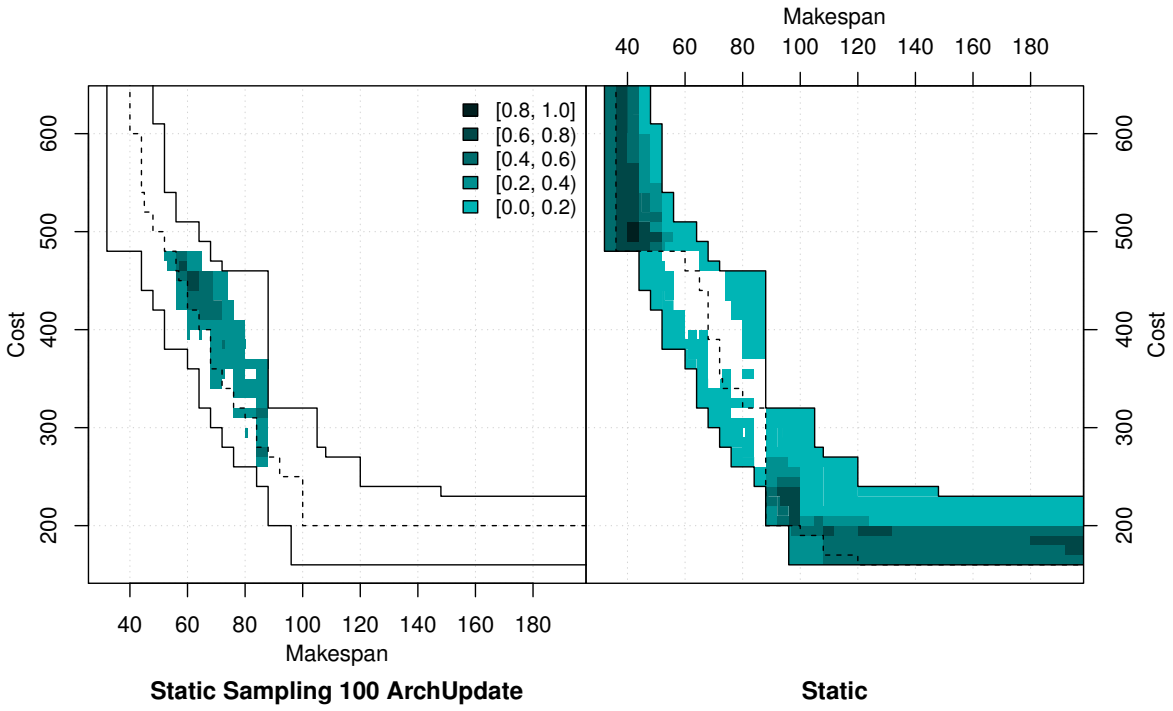
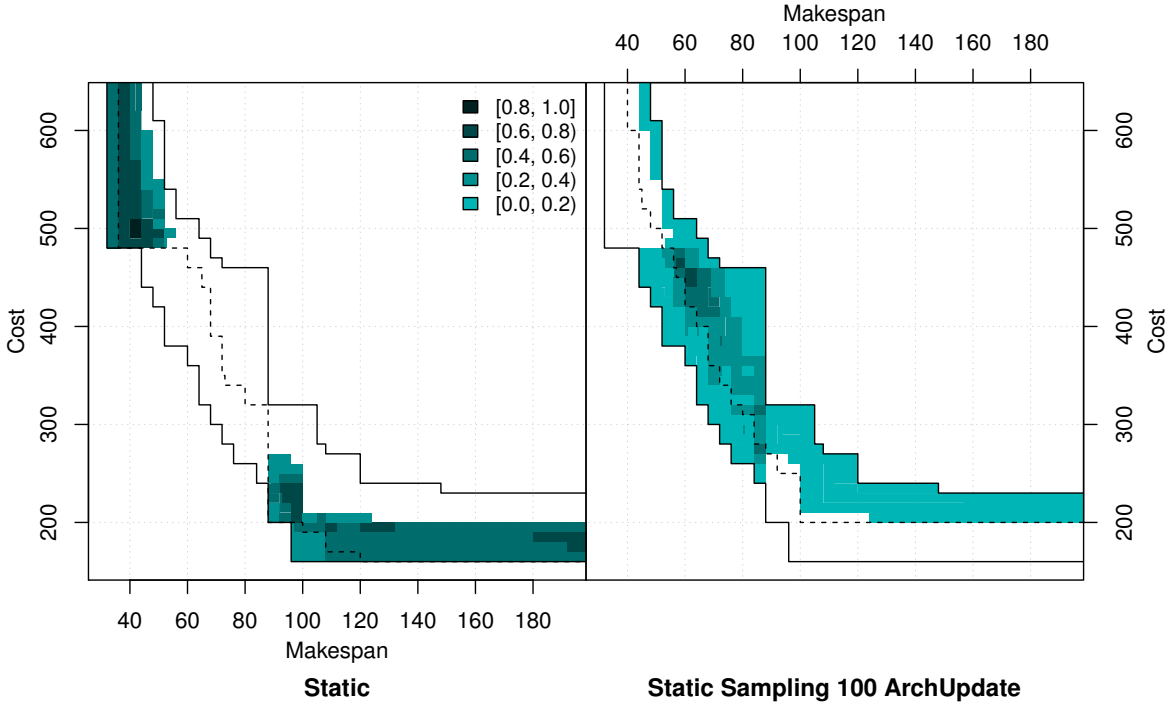


FIGURE 4.203: Instance Zeno9 : Statique comparée à Adaptative.



4.8.5.3 Série 3 : Stratégie auto-adaptative, 10-échantillon

On constate une nouvelle fois une difficulté à atteindre les valeurs extrêmes malgré une bonne répartition des fronts cumulés. En comparaison avec la stratégie auto-adaptative avec un 1-échantillon, l'hypervolume décroit moins rapidemment, toujours à cause du surcoût en calculs induit par l'échantillon.

Il est intéressant de noter qu'un test statistique sur les fonctions d'atteinte ne rejette pas l'hypothèse nulle entre la stratégie adaptative et la stratégie statique pour un 10-échantillon alors qu'il la rejettait pour un 1-échantillon.

FIGURE 4.204: Instance Zeno9 : Fronts de Pareto cumulés pour tous les runs, à la fin de chaque run.

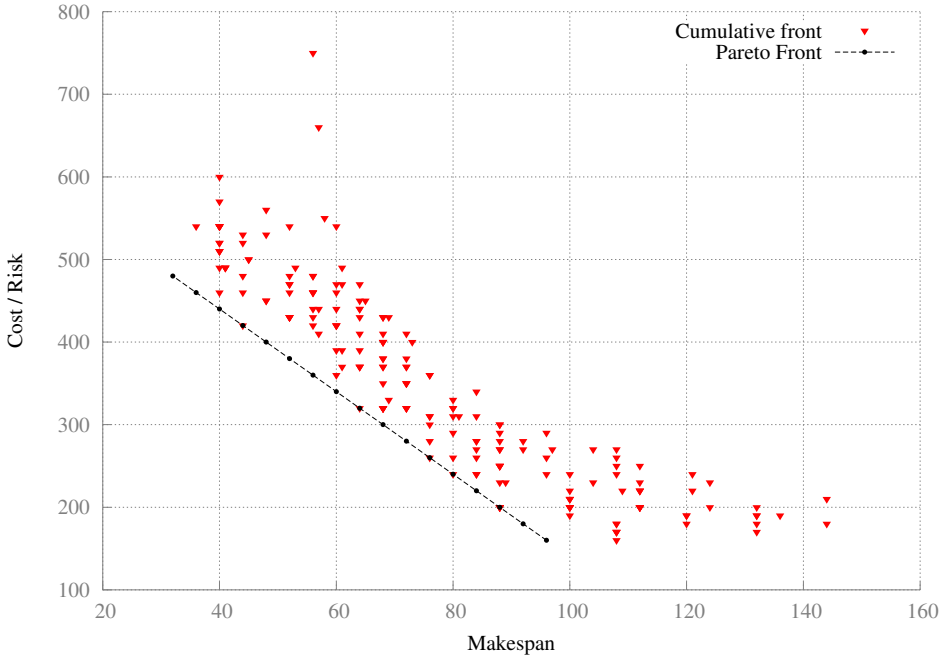


FIGURE 4.205: Instance Zeno9 : Population cumulées pour tous les runs et à tout temps.

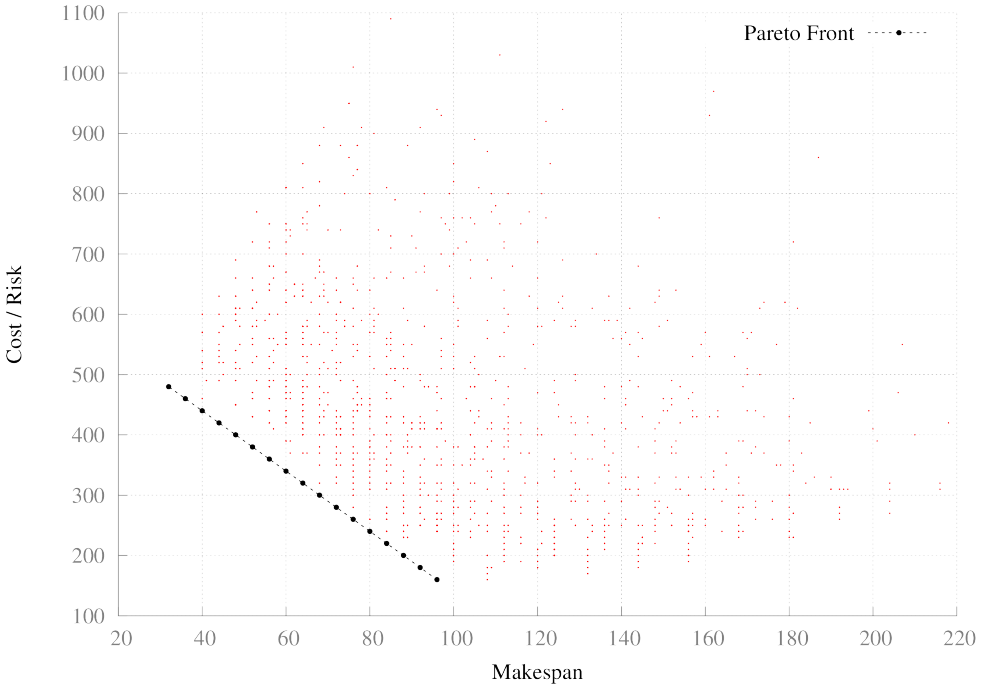


FIGURE 4.206: Instance Zeno9 : Population cumulées pour tous les runs et à tout temps (version colorée).

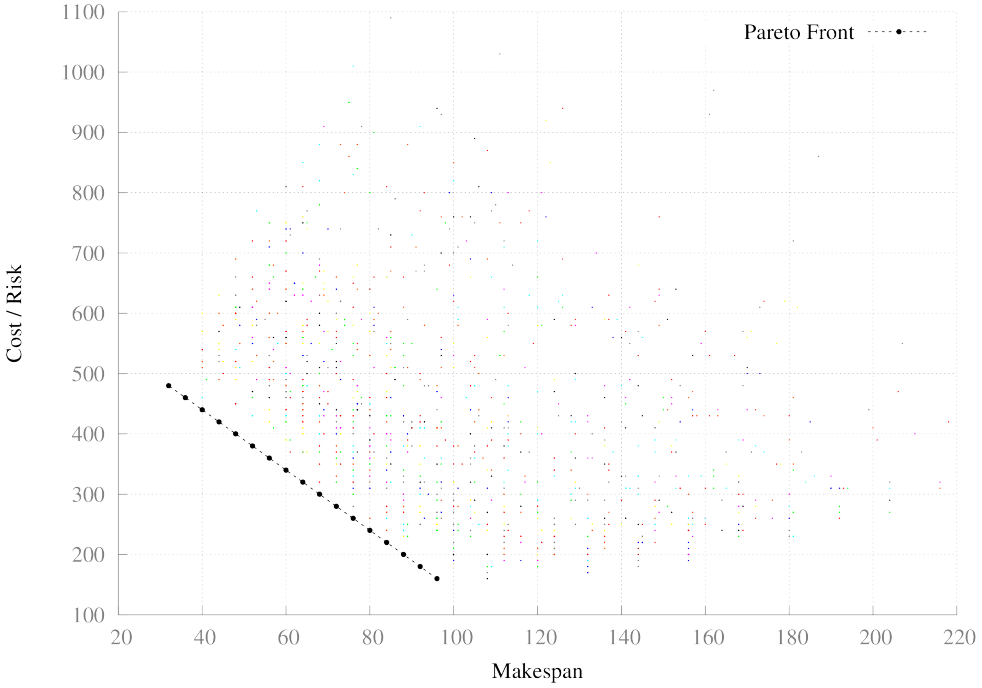


FIGURE 4.207: Instance Zeno9 : Surfaces d'atteinte pour tous les runs.

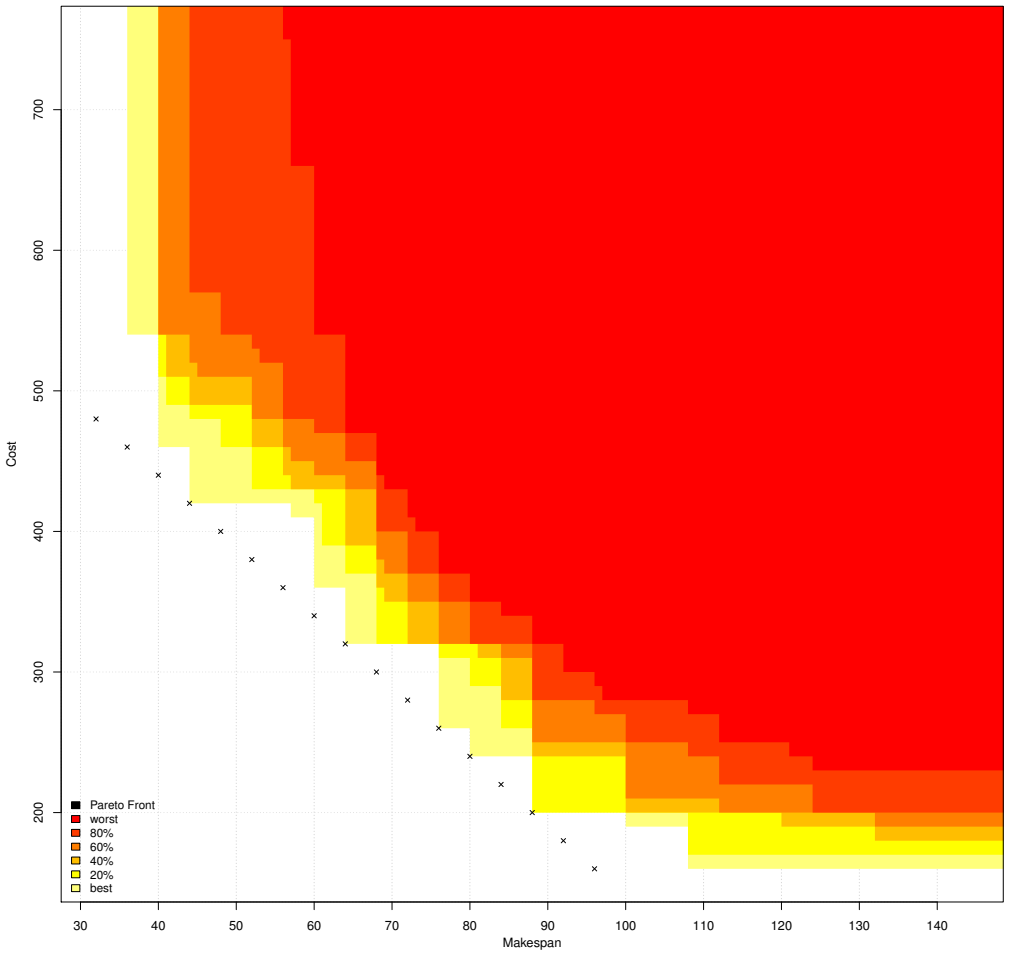


FIGURE 4.208: Instance Zeno9 : Trajectoires de l'hypervolume pour tous les runs.

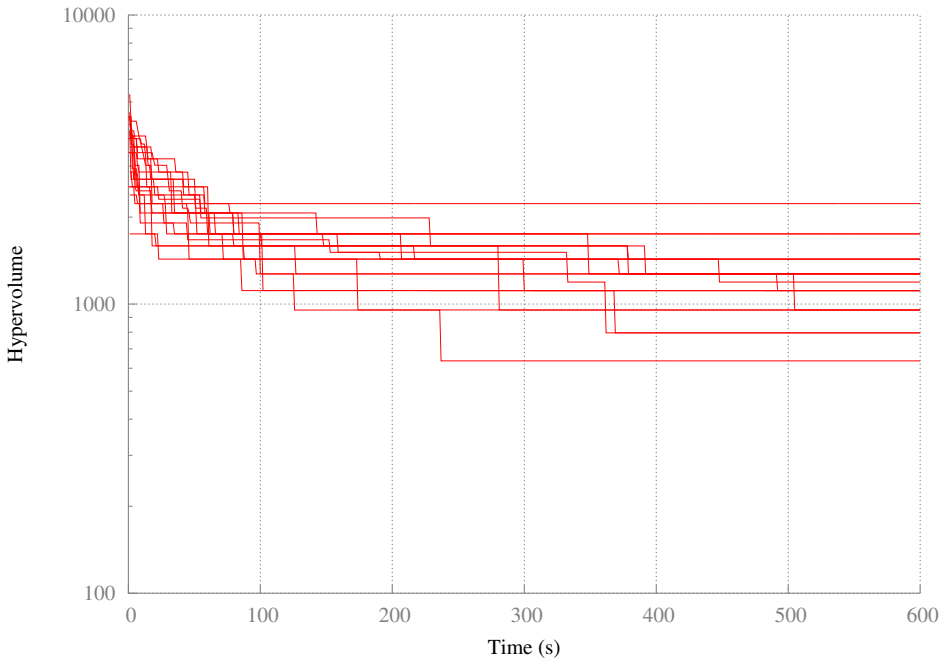


FIGURE 4.209: Instance Zeno9 : Diversité des vecteurs objectifs pour tous les runs.

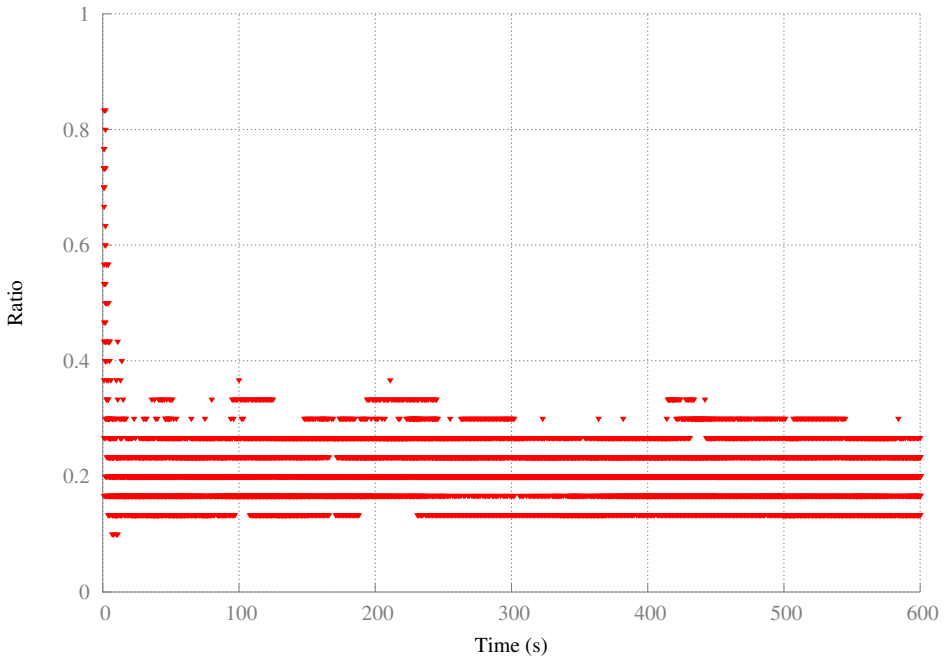


FIGURE 4.210: Instance Zeno9 : Diversité des vecteurs objectifs pour tous les runs (version colorée).

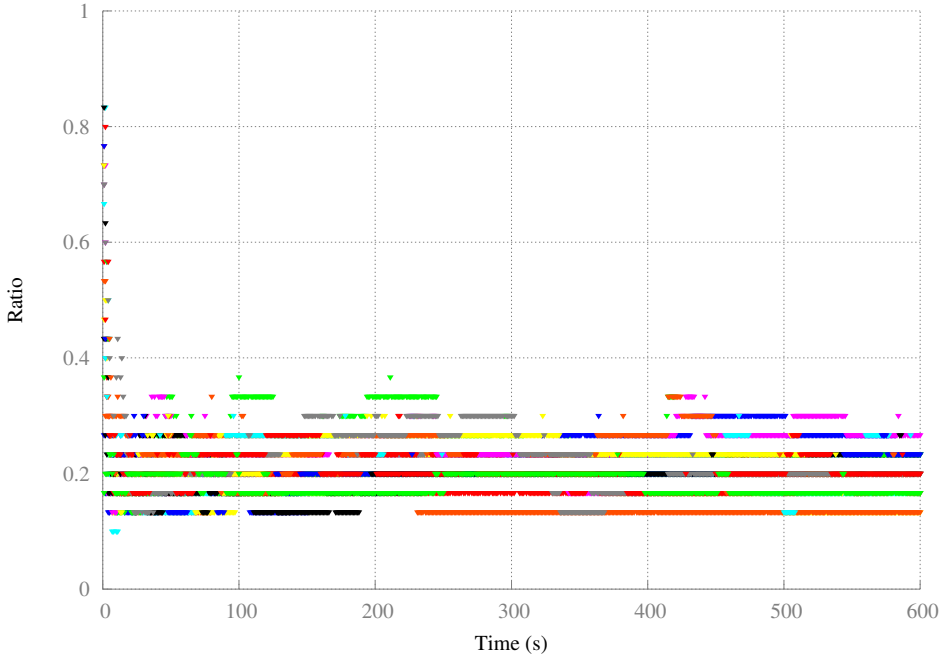


FIGURE 4.211: Instance Zeno9 : Comparatif des surfaces d'atteinte.

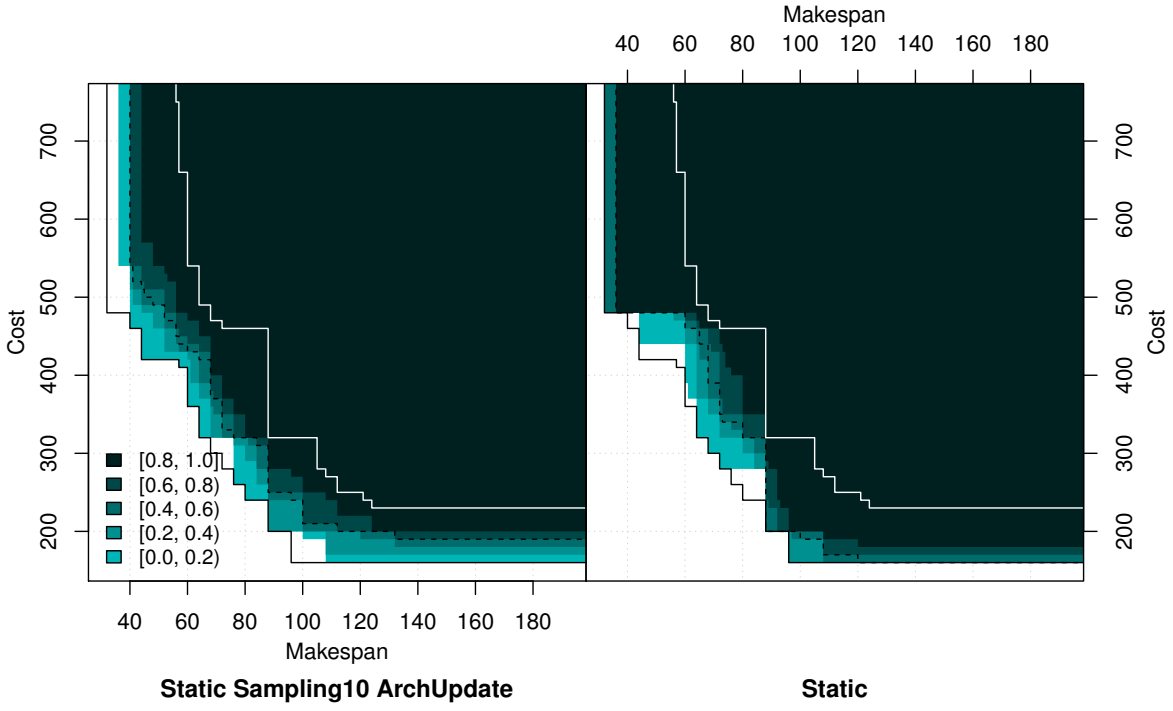


FIGURE 4.212: Instance Zeno9 : AutoAdaptative comparée à Statique.

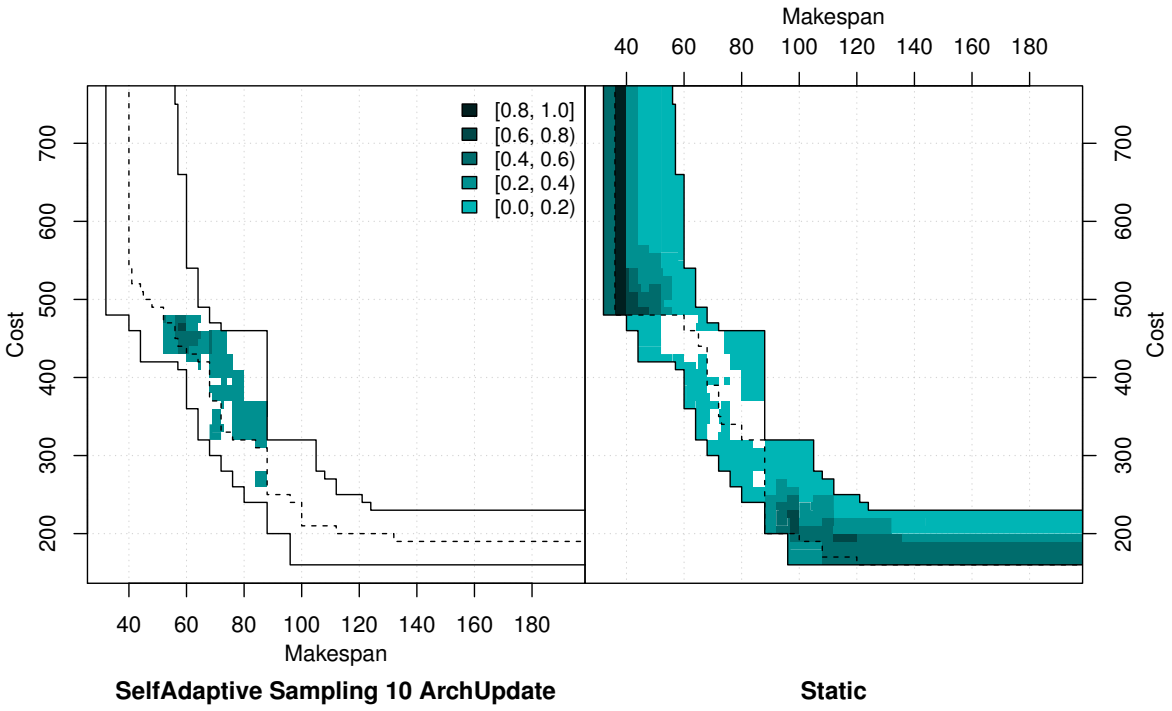
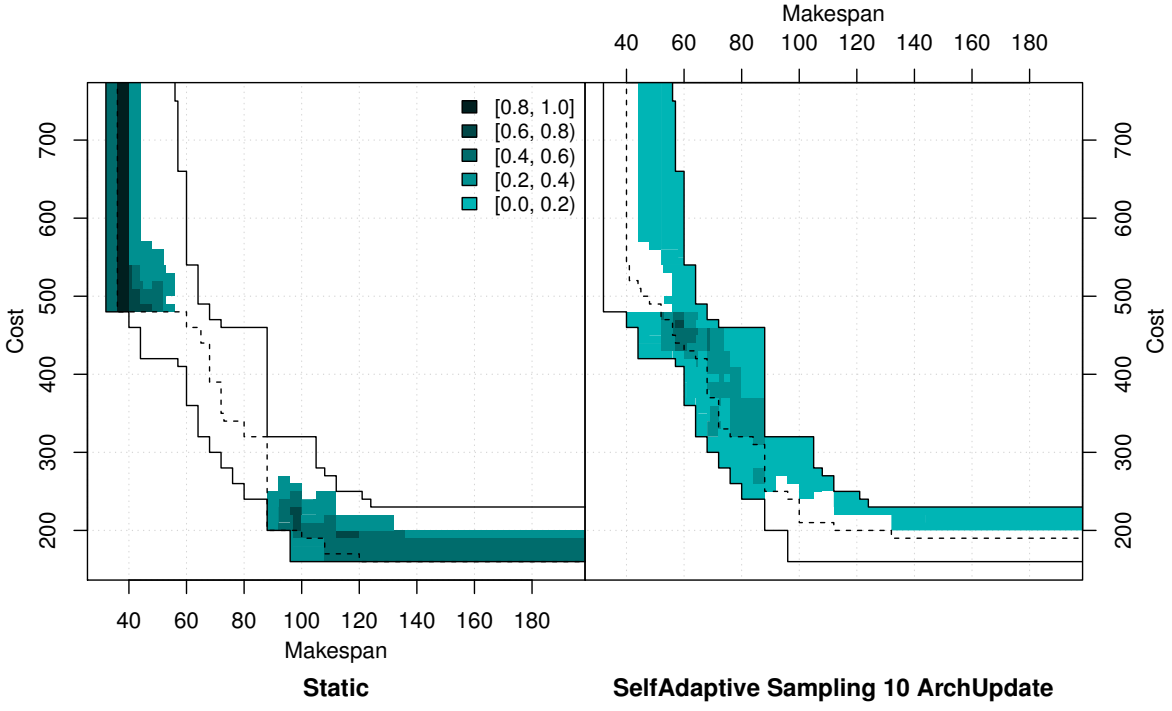


FIGURE 4.213: Instance Zeno9 : Statique comparée à AutoAdaptative.



4.8.5.4 Série 4 : Stratégie auto-adaptative, 100-échantillon

FIGURE 4.214: Instance Zeno9 : Fronts de Pareto cumulés pour tous les runs, à la fin de chaque run.

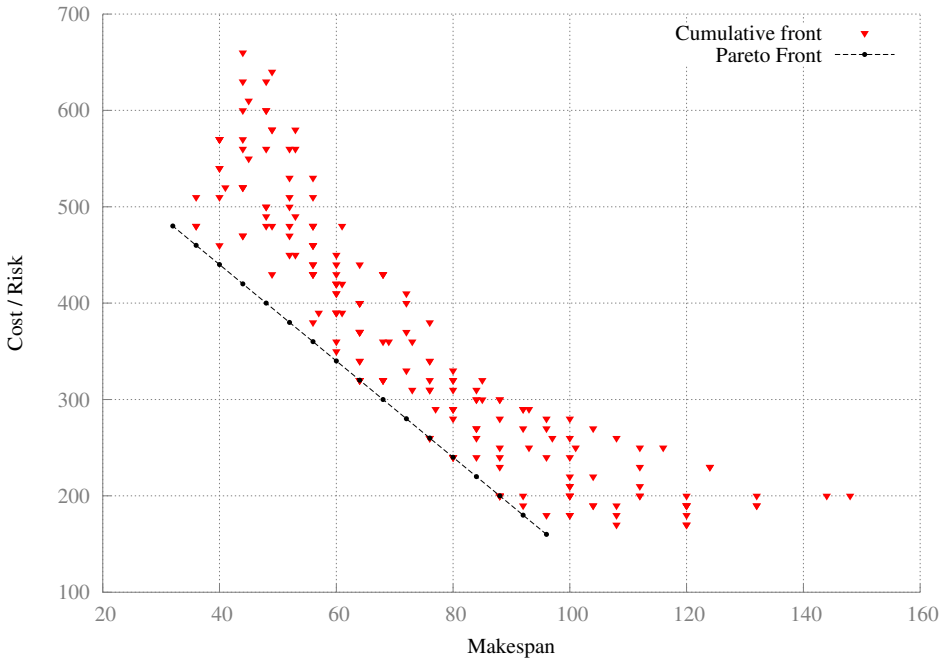


FIGURE 4.215: Instance Zeno9 : Population cumulées pour tous les runs et à tout temps.

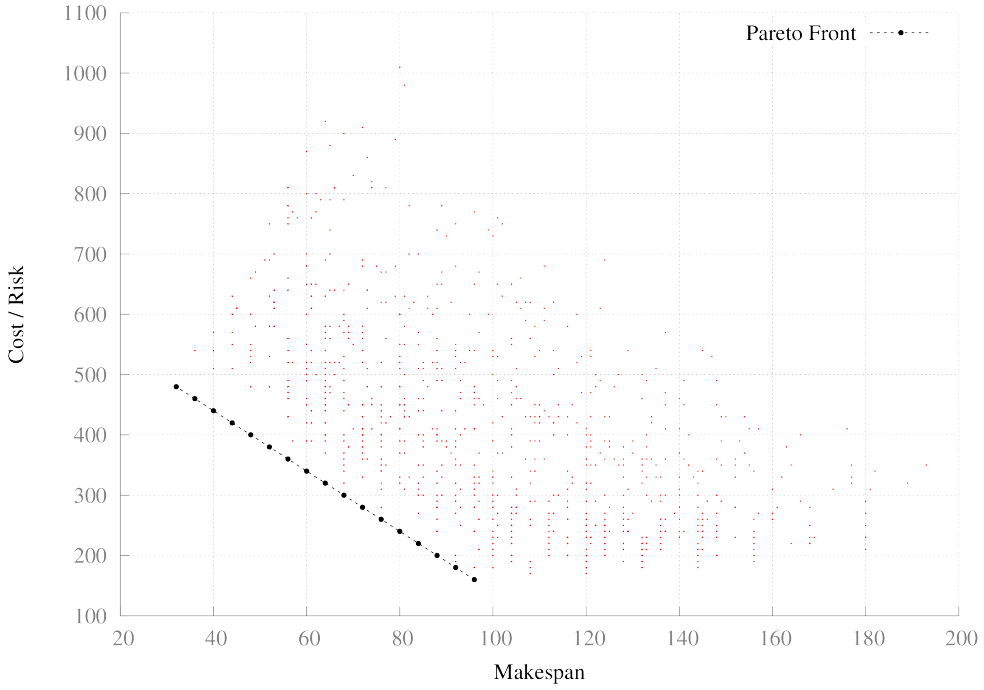


FIGURE 4.216: Instance Zeno9 : Population cumulées pour tous les runs et à tout temps (version colorée).

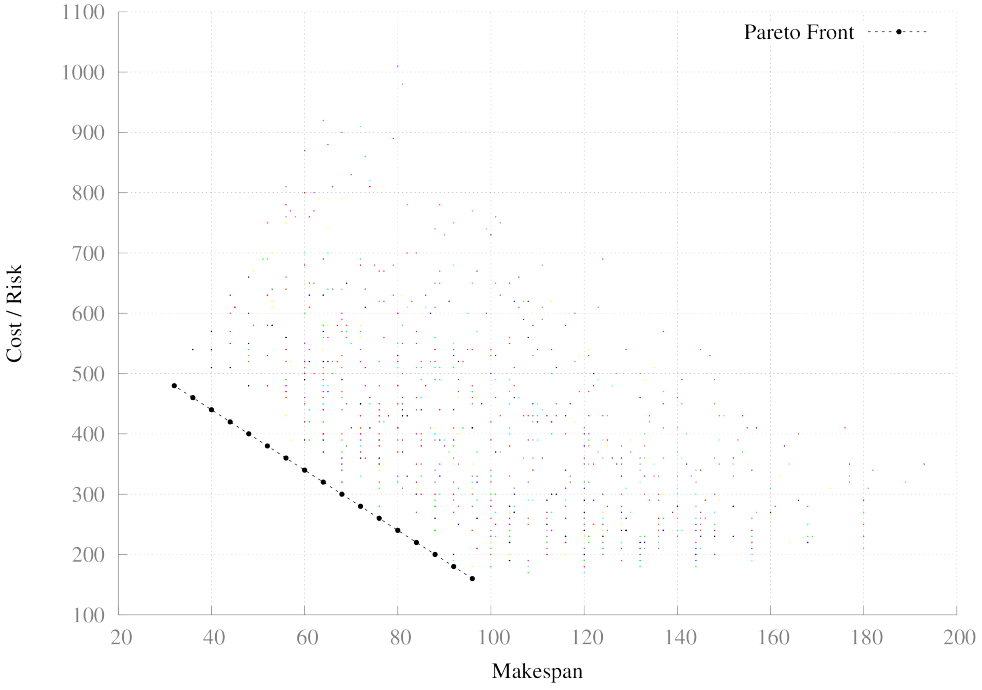


FIGURE 4.217: Instance Zeno9 : Surfaces d'atteinte pour tous les runs.

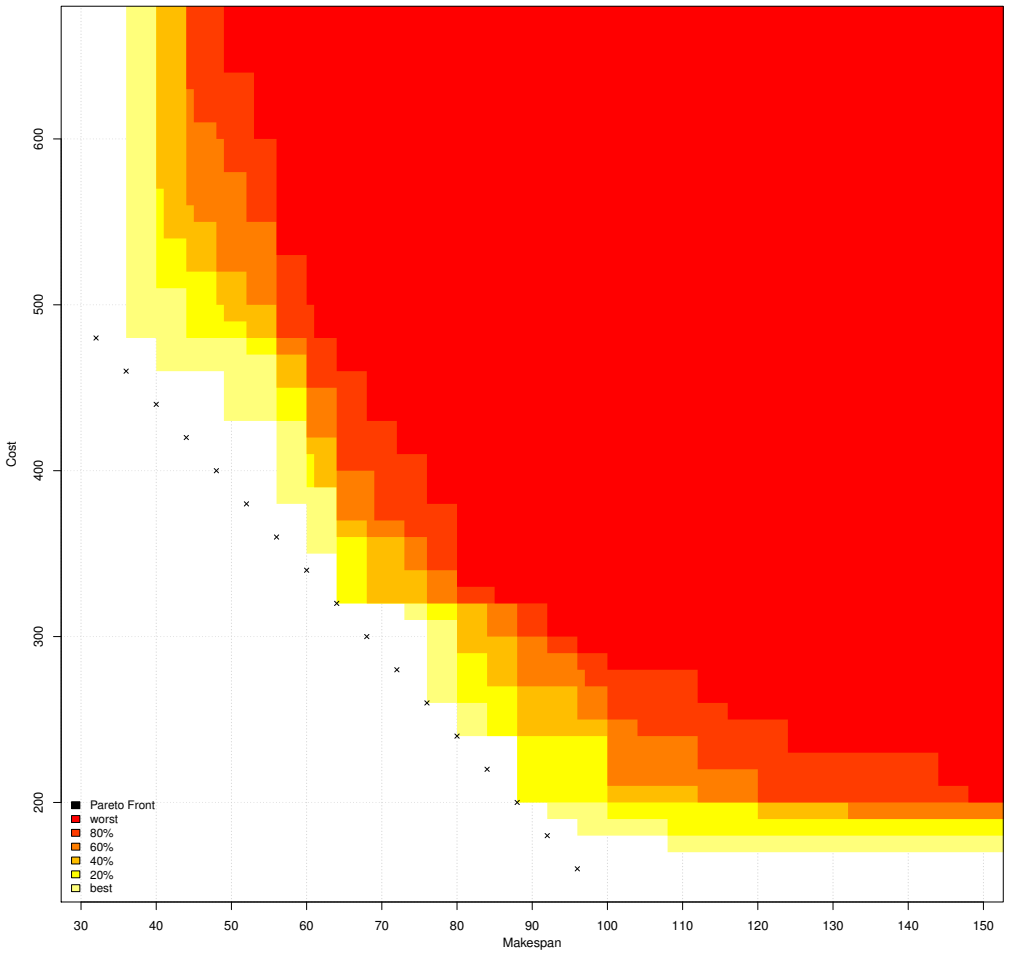


FIGURE 4.218: Instance Zeno9 : Trajectoires de l'hypervolume pour tous les runs.

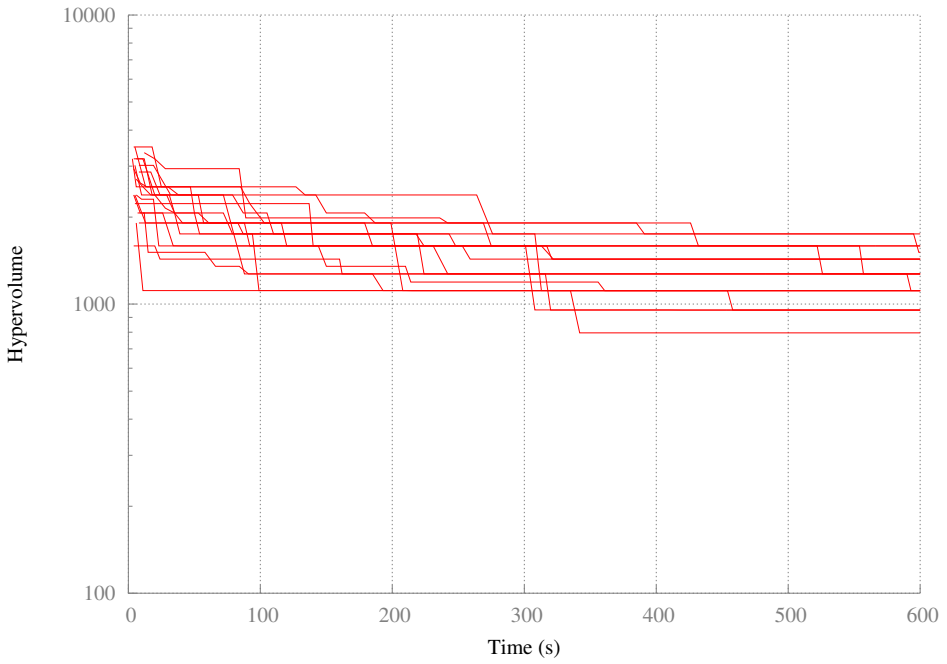


FIGURE 4.219: Instance Zeno9 : Diversité des vecteurs objectifs pour tous les runs.

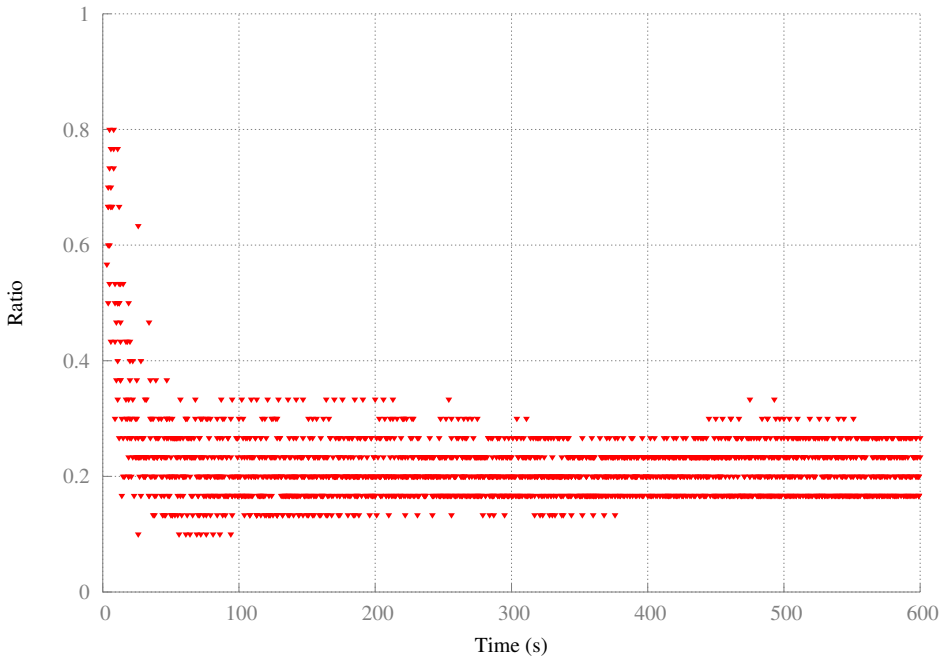


FIGURE 4.220: Instance Zeno9 : Diversité des vecteurs objectifs pour tous les runs (version colorée).

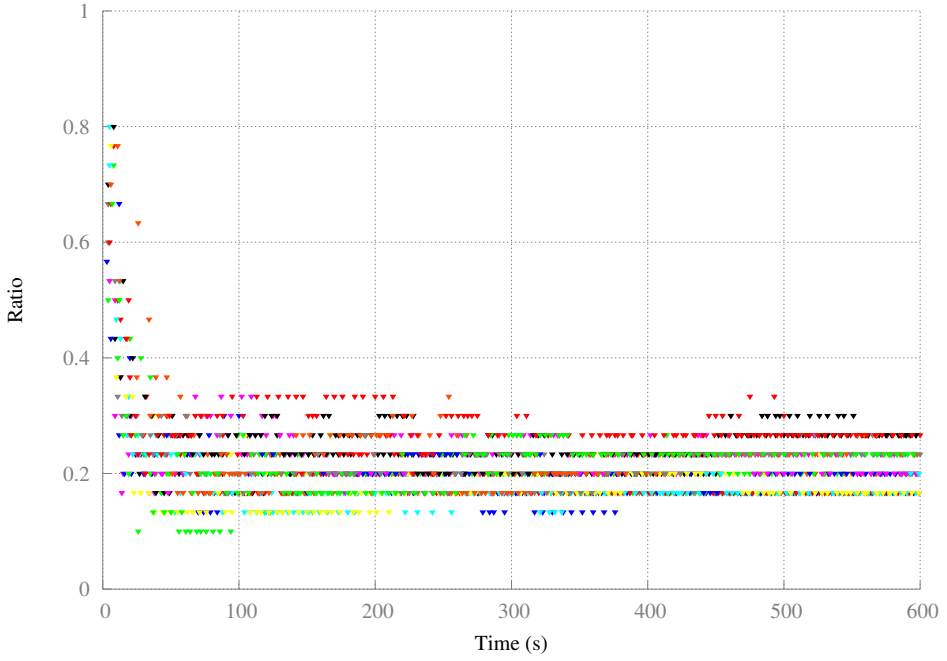


FIGURE 4.221: Instance Zeno9 : Comparatif des surfaces d'atteinte.

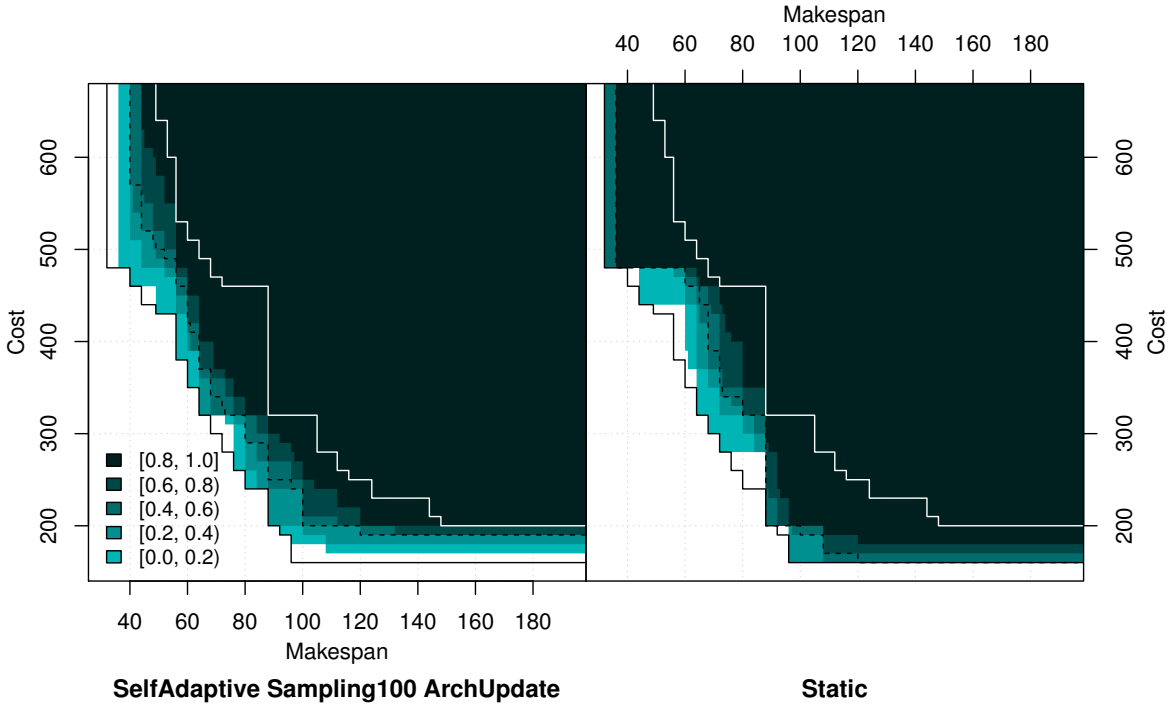


FIGURE 4.222: Instance Zeno9 : AutoAdaptative comparée à Statique.

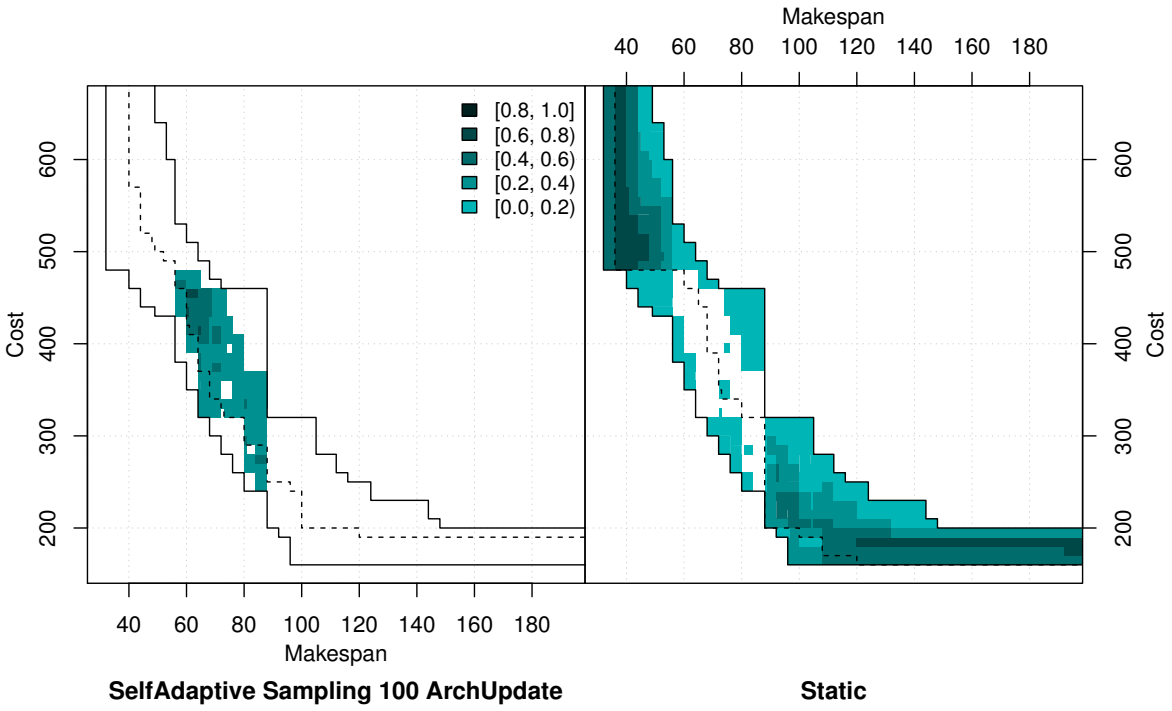
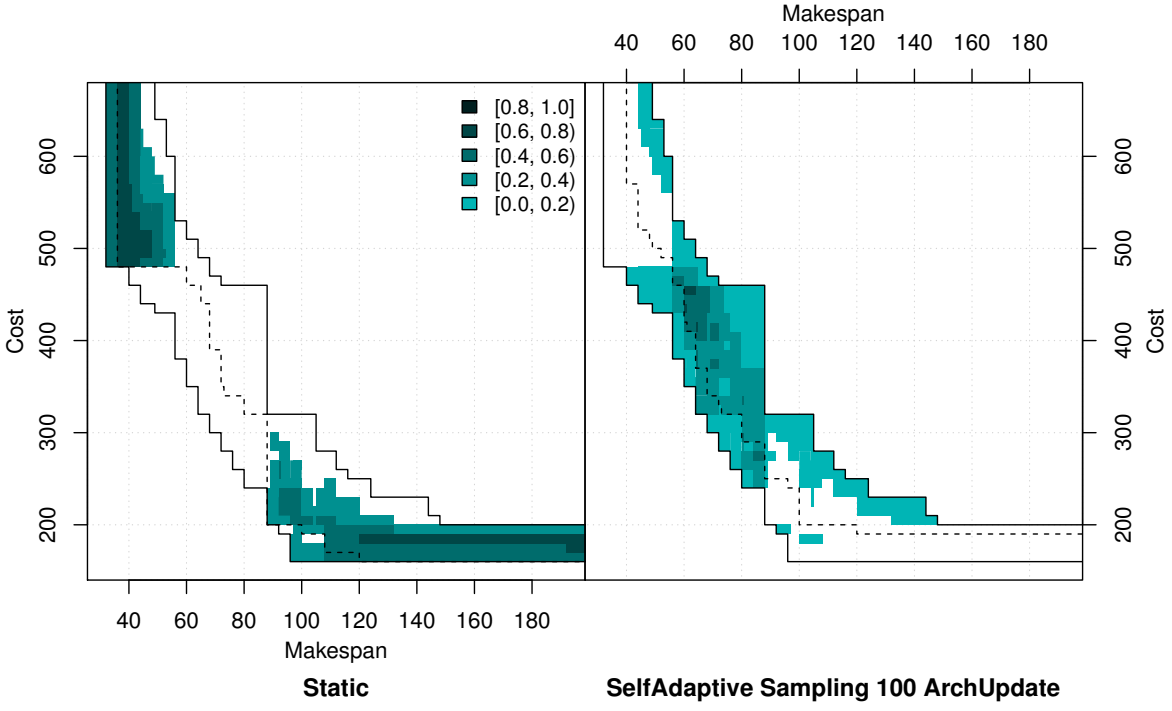


FIGURE 4.223: Instance Zeno9 : Statique comparée à AutoAdaptative.



4.8.5.5 Série 5 : Stratégie statique, 10-échantillon, vecteur moyen

On présente ici une stratégie statique, avec un échantillon de taille 10 et dont le vecteur moyen sert de vecteur objectif pour l'individu évalué. Notons un front uniforme, comme pour la sélection via l'indicateur $\lambda^j_{\Delta^+}(x) = \sum_i \Delta f^j_i(x)$, avec cependant, pour une taille d'échantillon égale, un front cumulé dont les points semblent légèrement plus proches du front exact, au centre de celui-ci.

Il est bien sur impossible de comparer la population cumulée puisque celle-ci ne fait pas référence au problème de planification d'origine, chaque vecteur objectif ne résultat pas d'un plan mais juste de la moyenne au sein de l'échantillon. C'est très logiquement que l'on se situe plus loin du front du problème initial qu'avec une méthode d'évaluation classique et il est intéressant de noter une zone plus dense coté ≪ front exact ≫ du nuage de points.

Les surfaces d'atteinte de la figure 4.227 indique de bons résultats sur l'ensemble du front, avec une plus forte probabilité d'atteinte des zones proches du front qu'avec les séries précédentes. On remarque touours quelques lacunes sur les valeurs extrêmes par rapport à la série de référence, avec la stratégie statique et un 1-échantillon.

FIGURE 4.224: Instance Zeno9 : Fronts de Pareto cumulés pour tous les runs, à la fin de chaque run.

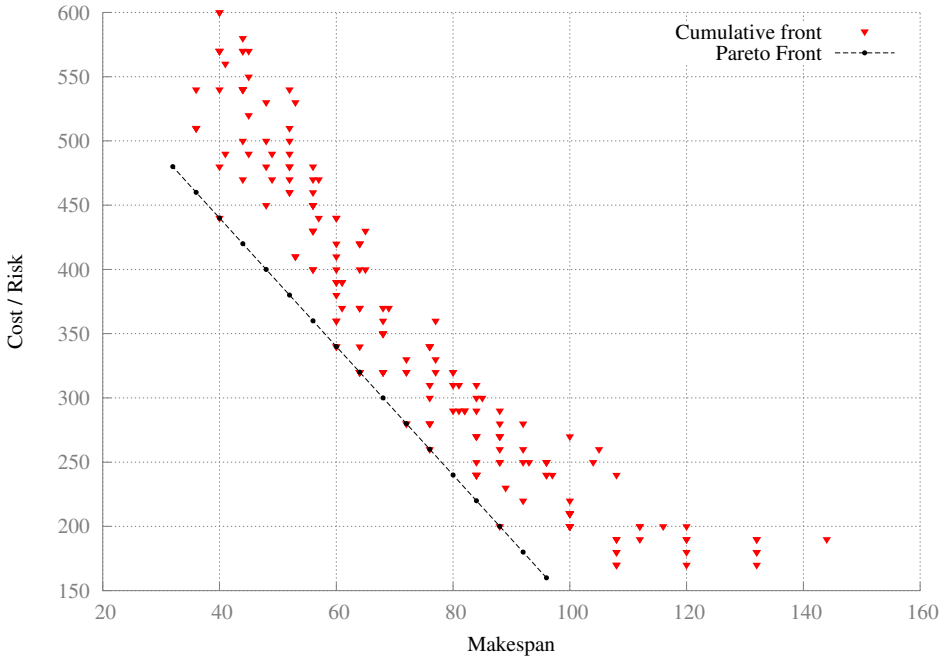


FIGURE 4.225: Instance Zeno9 : Population cumulées pour tous les runs et à tout temps.

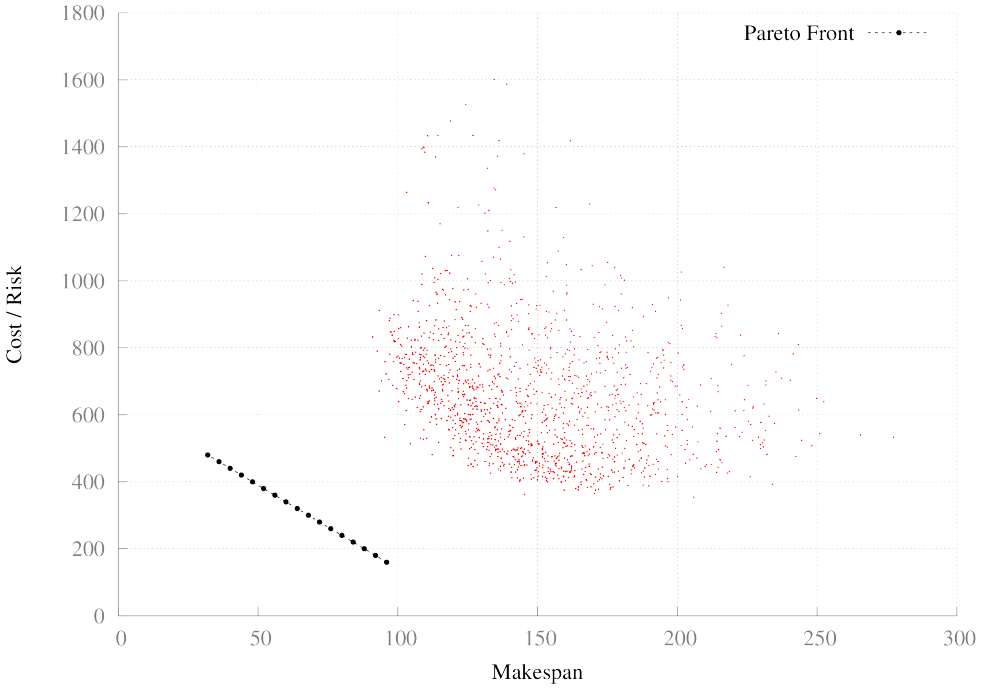


FIGURE 4.226: Instance Zeno9 : Population cumulées pour tous les runs et à tout temps (version colorée).

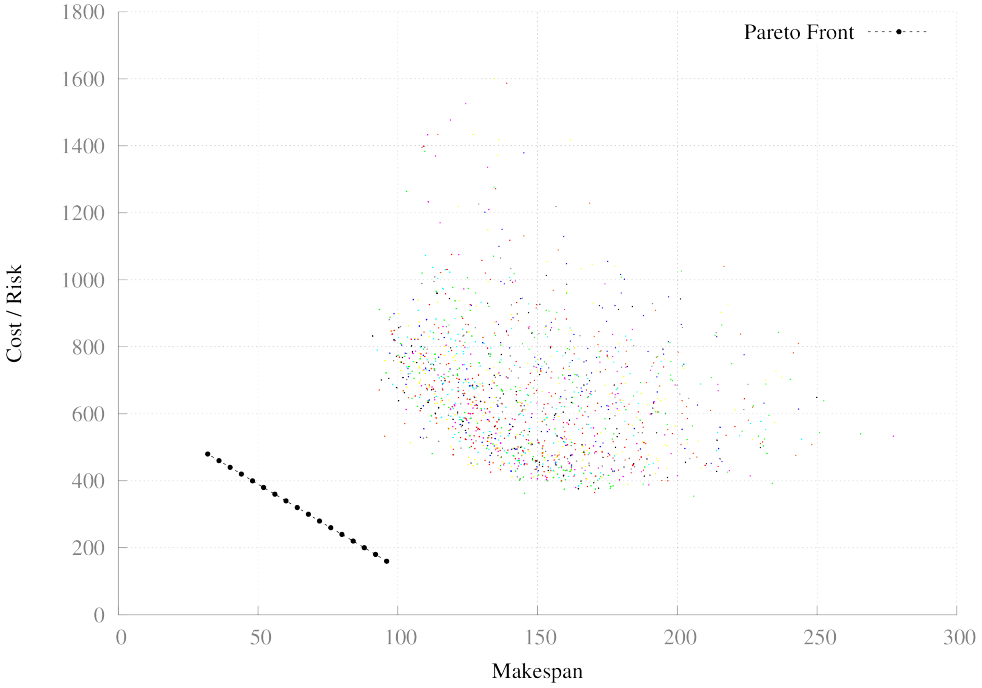


FIGURE 4.227: Instance Zeno9 : Surfaces d'atteinte pour tous les runs.

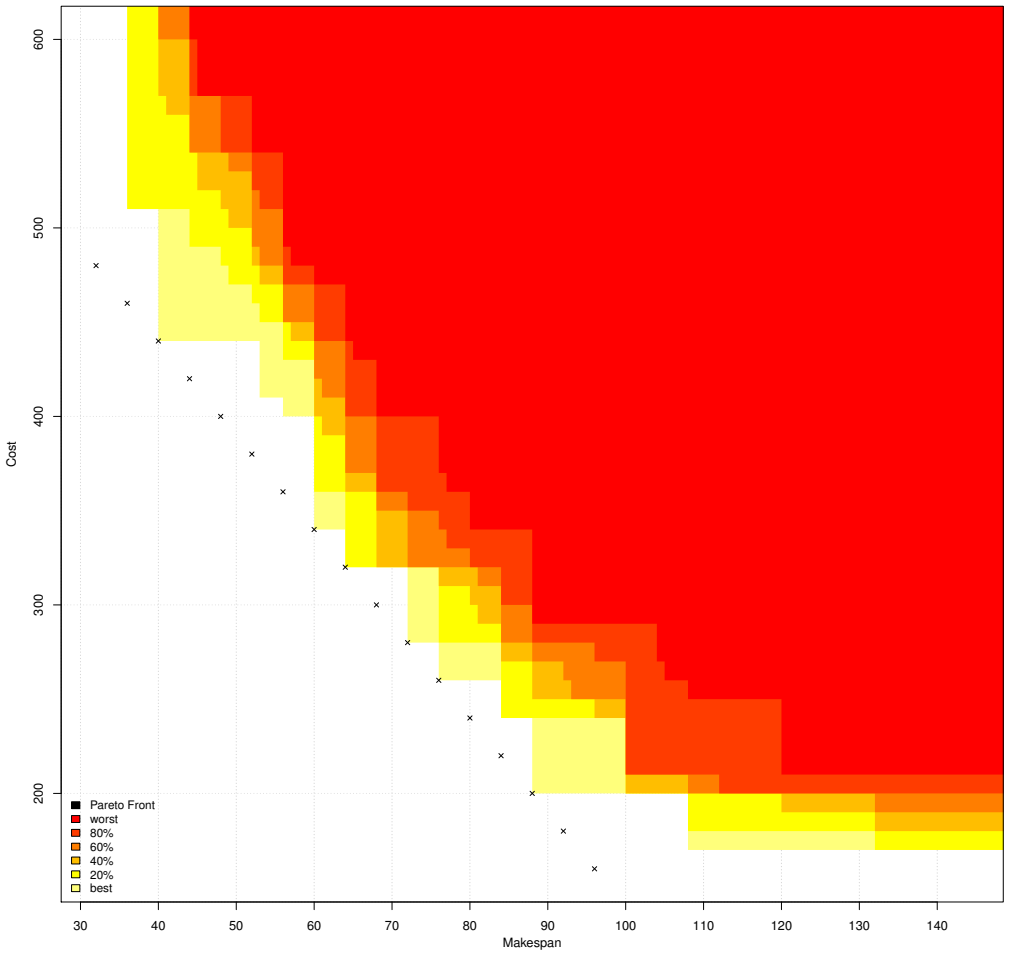


FIGURE 4.228: Instance Zeno9 : Trajectoires de l'hypervolume pour tous les runs.

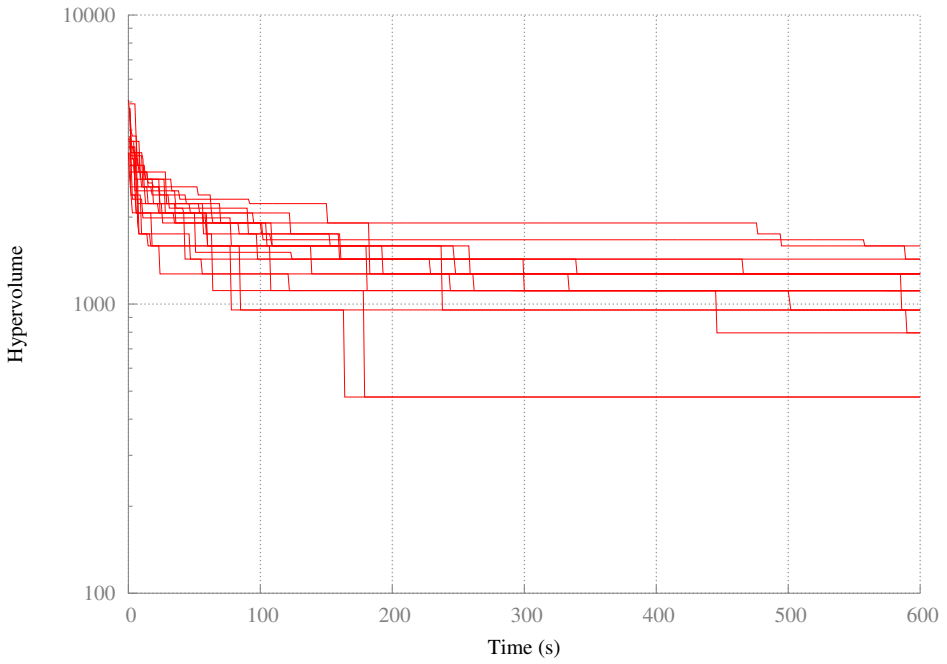


FIGURE 4.229: Instance Zeno9 : Diversité des vecteurs objectifs pour tous les runs.

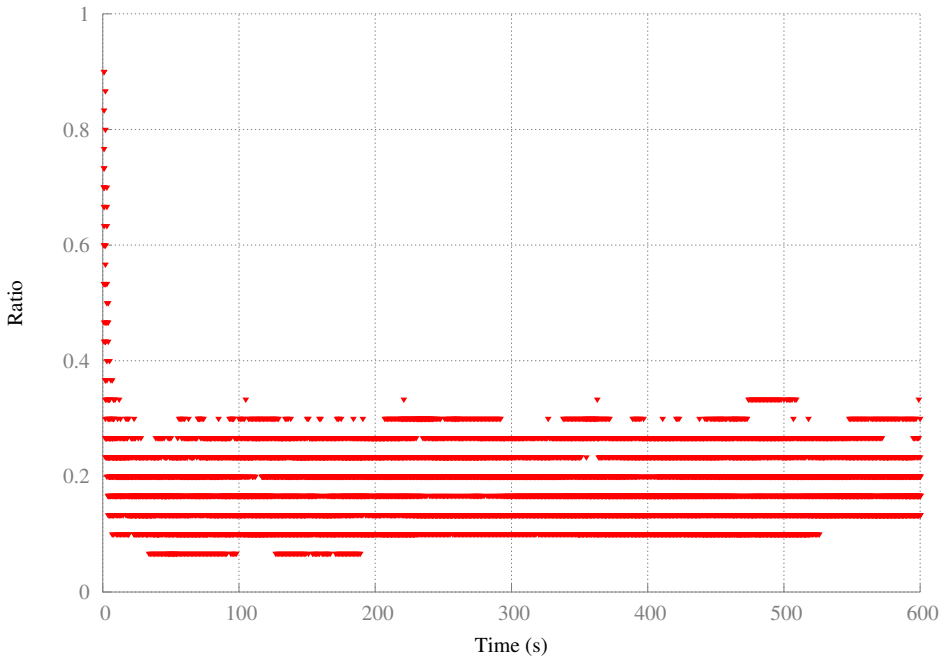


FIGURE 4.230: Instance Zeno9 : Diversité des vecteurs objectifs pour tous les runs (version colorée).

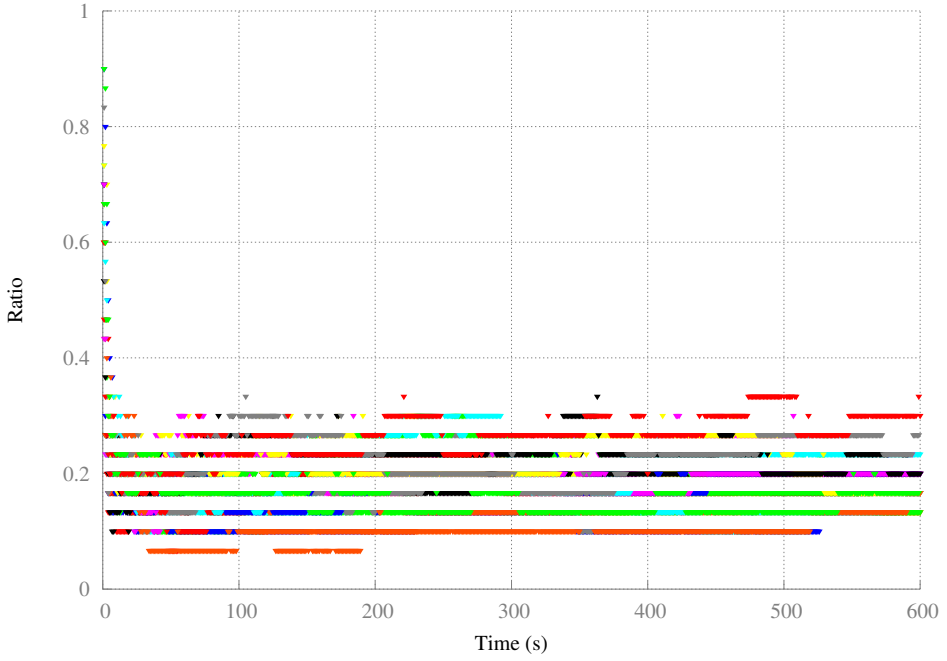


FIGURE 4.231: Instance Zeno9 : Comparatif des surfaces d'atteinte.

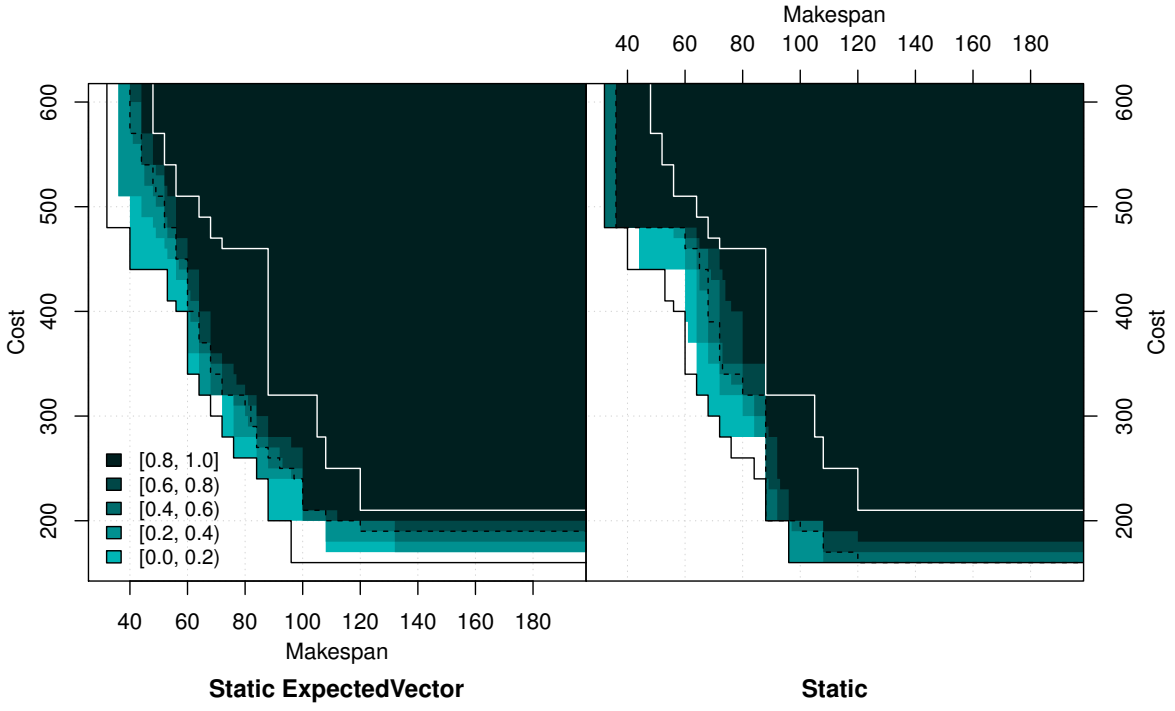


FIGURE 4.232: Instance Zeno9 : Statique avec vecteur moyen comparée à Statique.

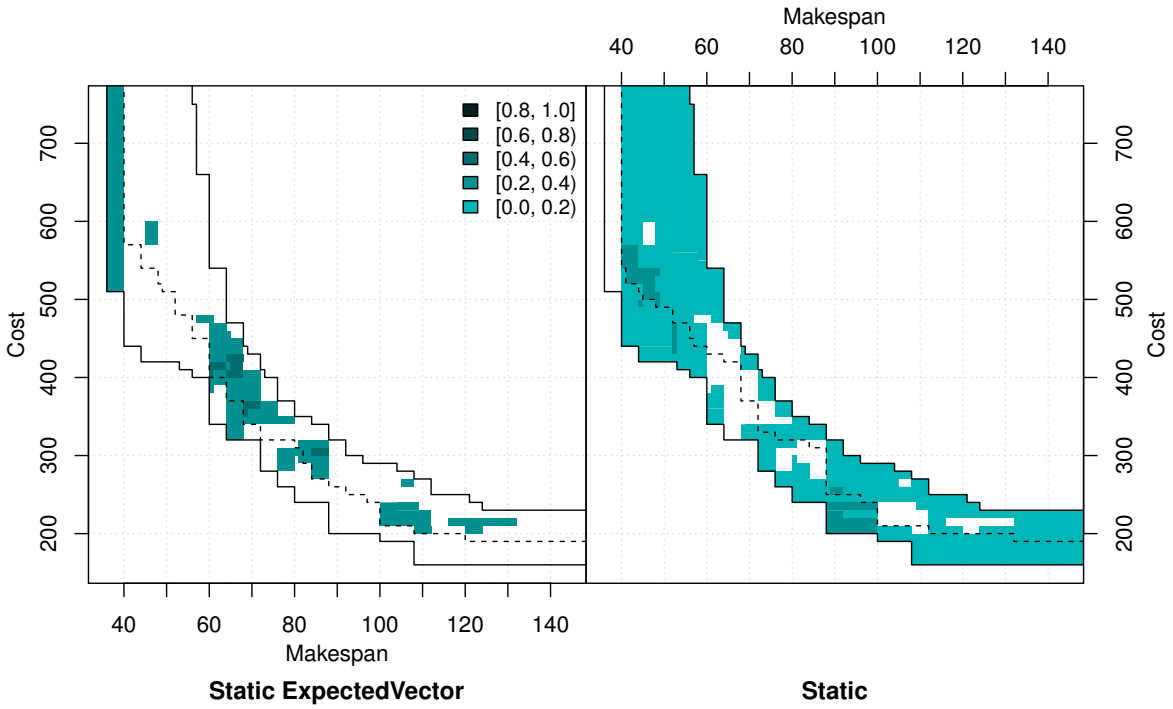
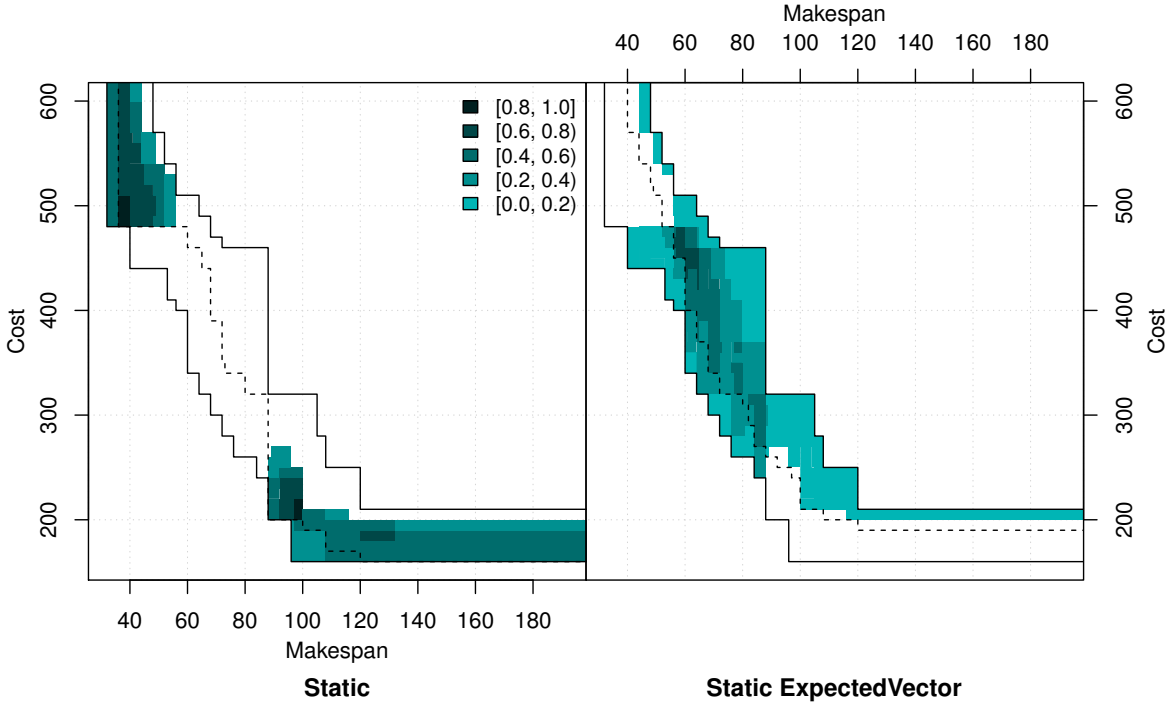


FIGURE 4.233: Instance Zeno9 : Statique comparée à Statique avec vecteur moyen.



4.8.5.6 Série 6 : Stratégie auto-adaptative, 10-échantillon, vecteur moyen

FIGURE 4.234: Instance Zeno9 : Fronts de Pareto cumulés pour tous les runs, à la fin de chaque run.

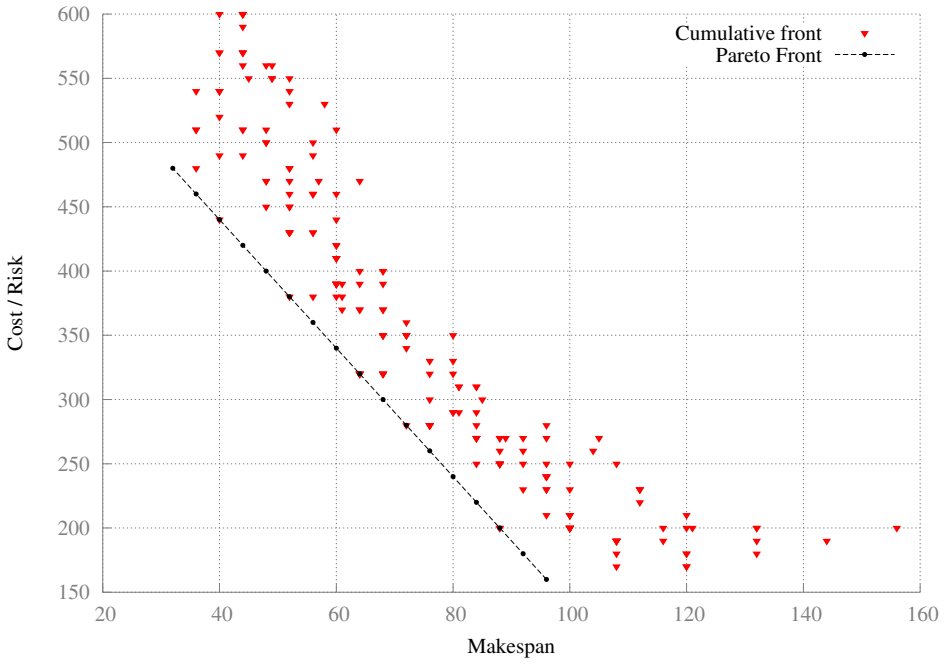


FIGURE 4.235: Instance Zeno9 : Population cumulées pour tous les runs et à tout temps.

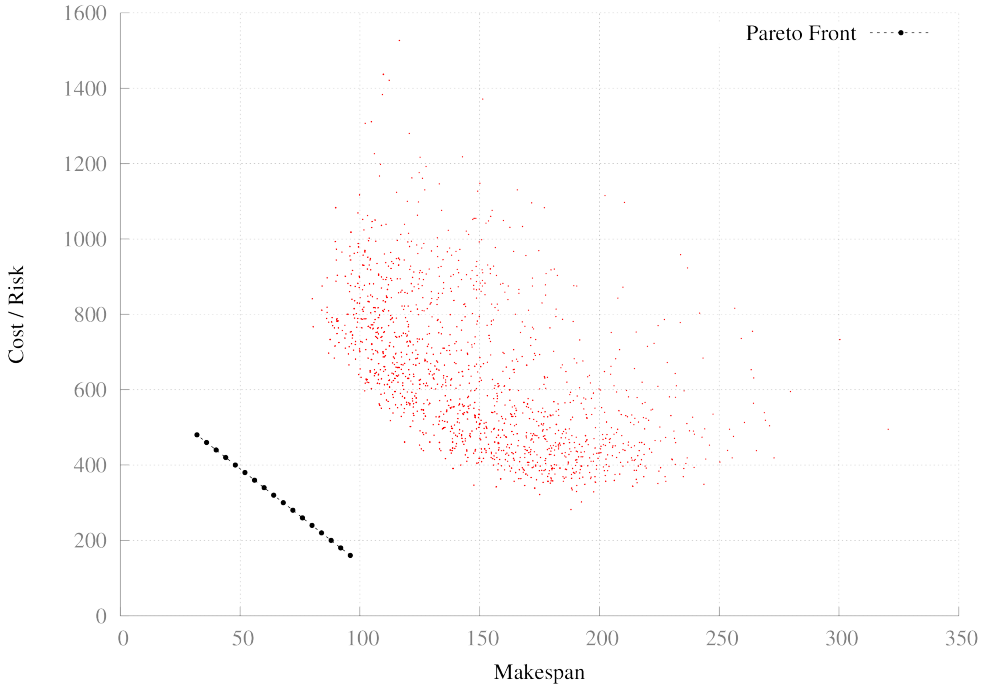


FIGURE 4.236: Instance Zeno9 : Population cumulées pour tous les runs et à tout temps (version colorée).

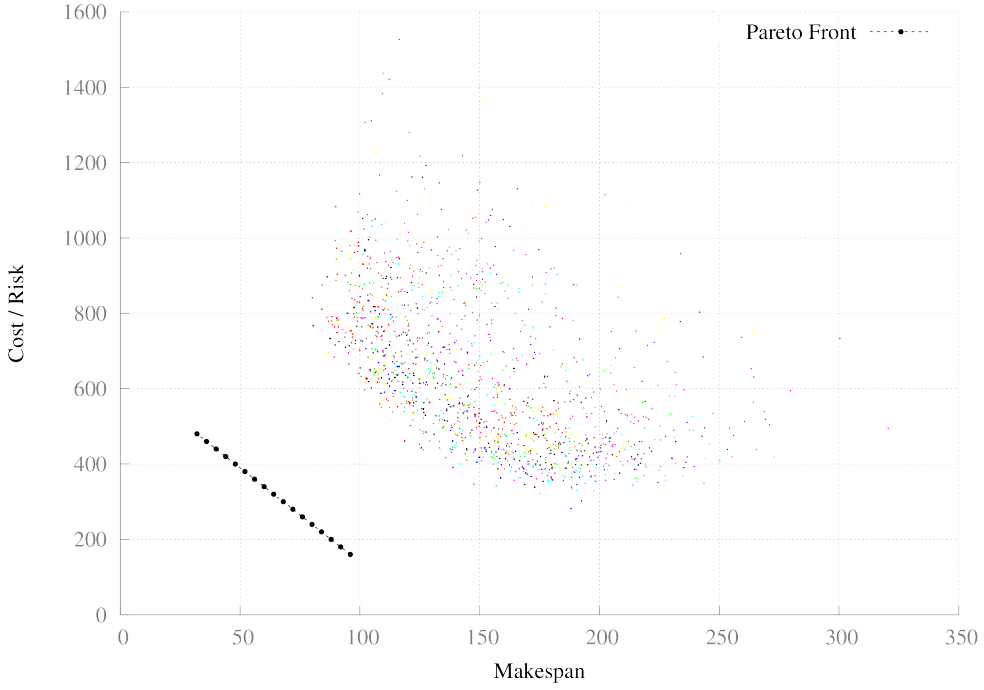


FIGURE 4.237: Instance Zeno9 : Surfaces d'atteinte pour tous les runs.

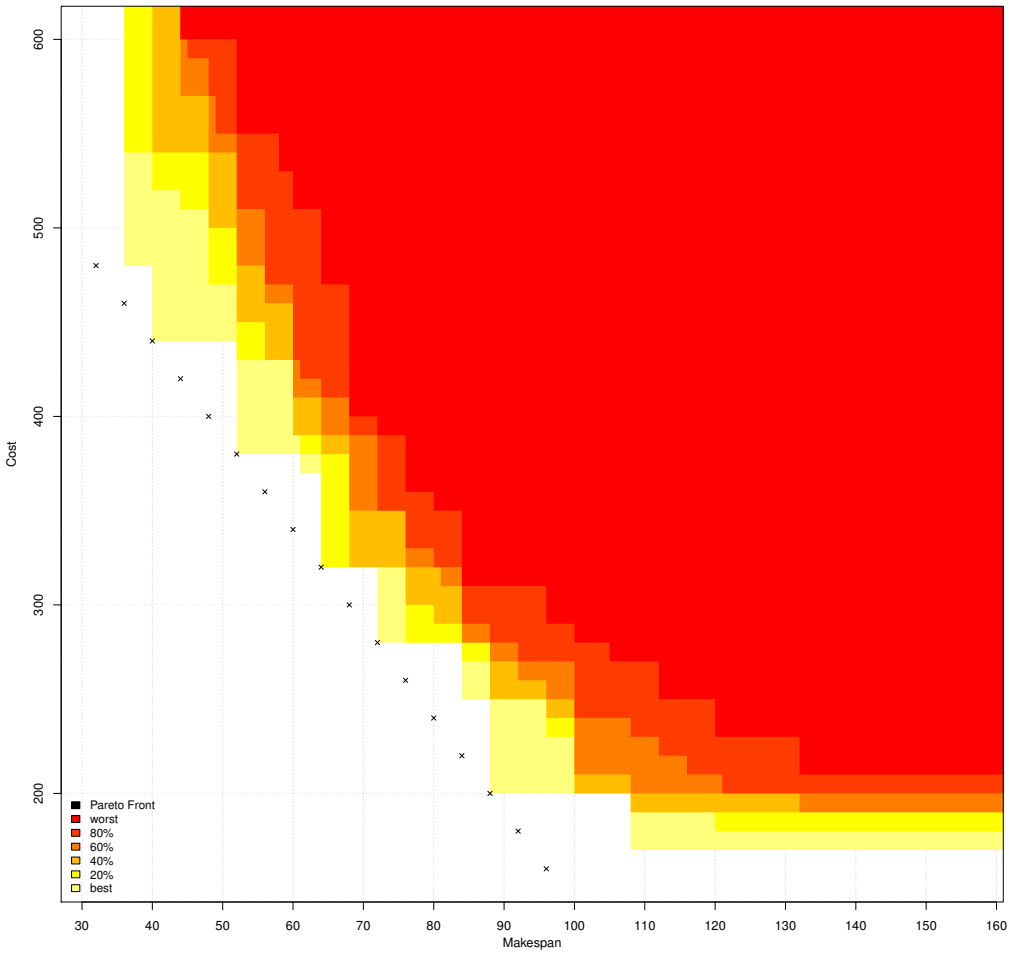


FIGURE 4.238: Instance Zeno9 : Trajectoires de l'hypervolume pour tous les runs.

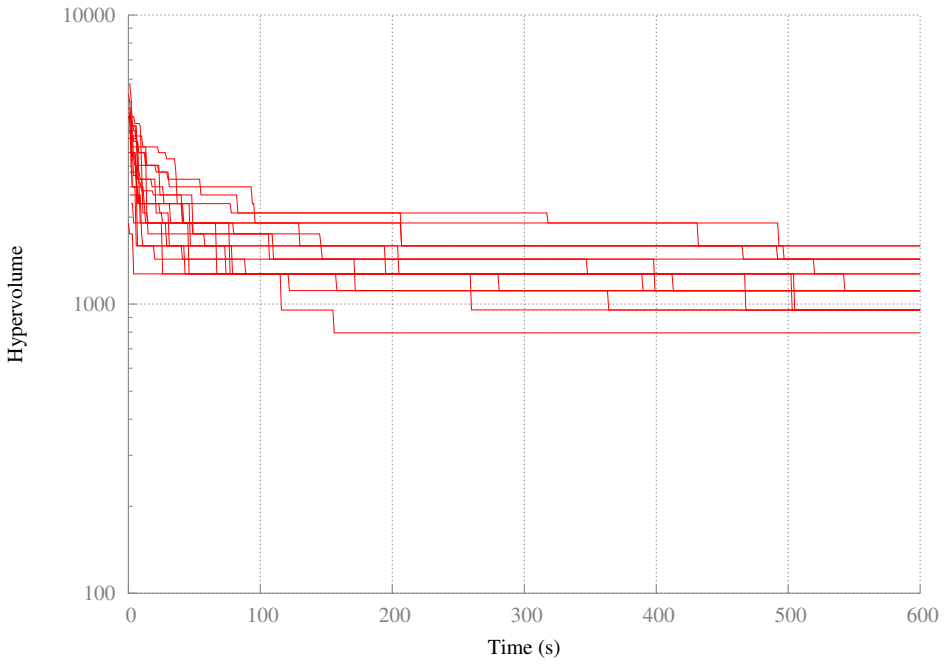


FIGURE 4.239: Instance Zeno9 : Diversité des vecteurs objectifs pour tous les runs.

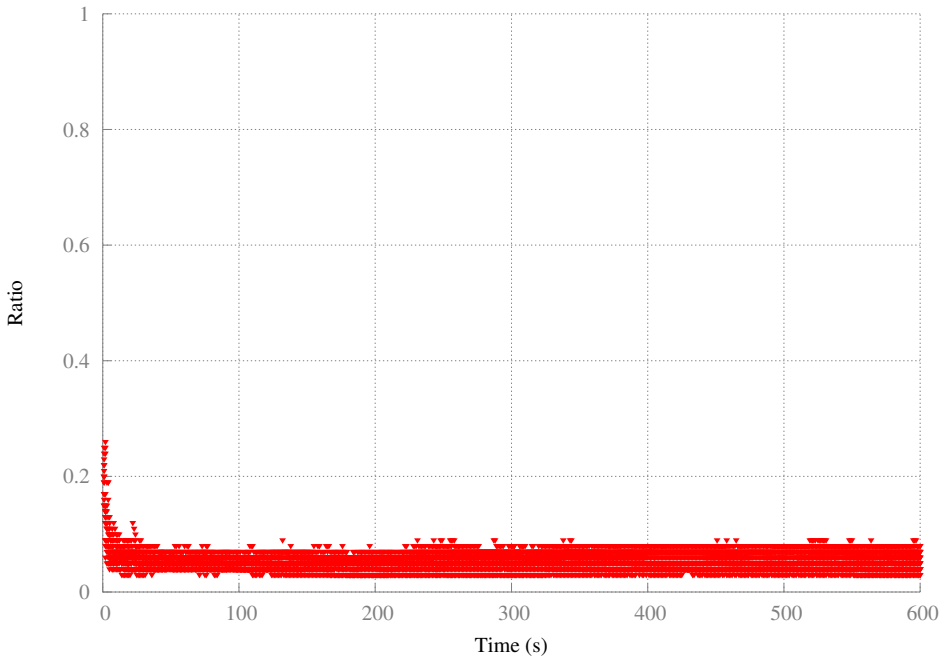


FIGURE 4.240: Instance Zeno9 : Diversité des vecteurs objectifs pour tous les runs (version colorée).

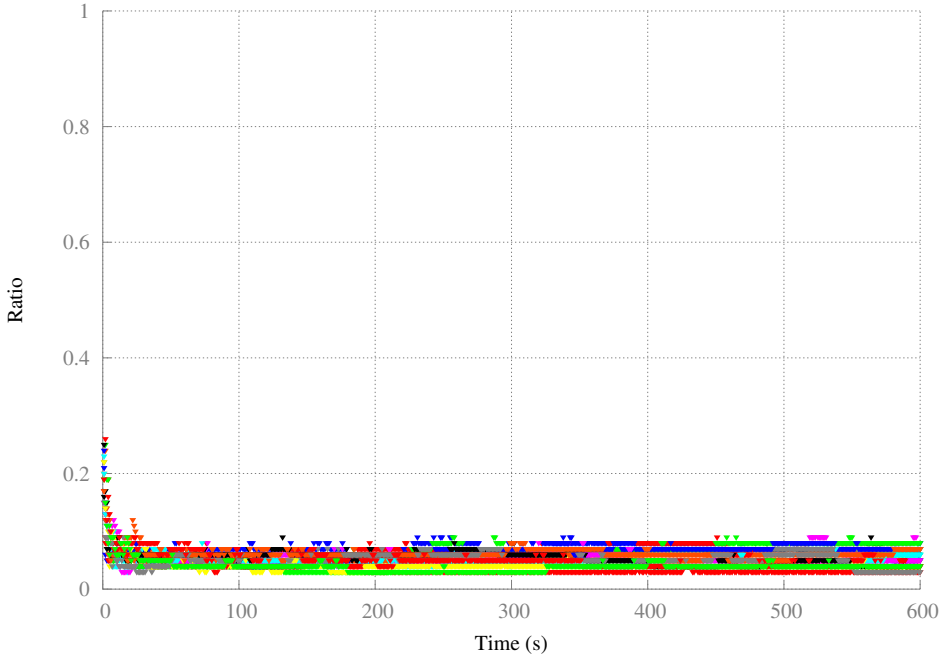


FIGURE 4.241: Instance Zeno9 : Comparatif des surfaces d'atteinte.

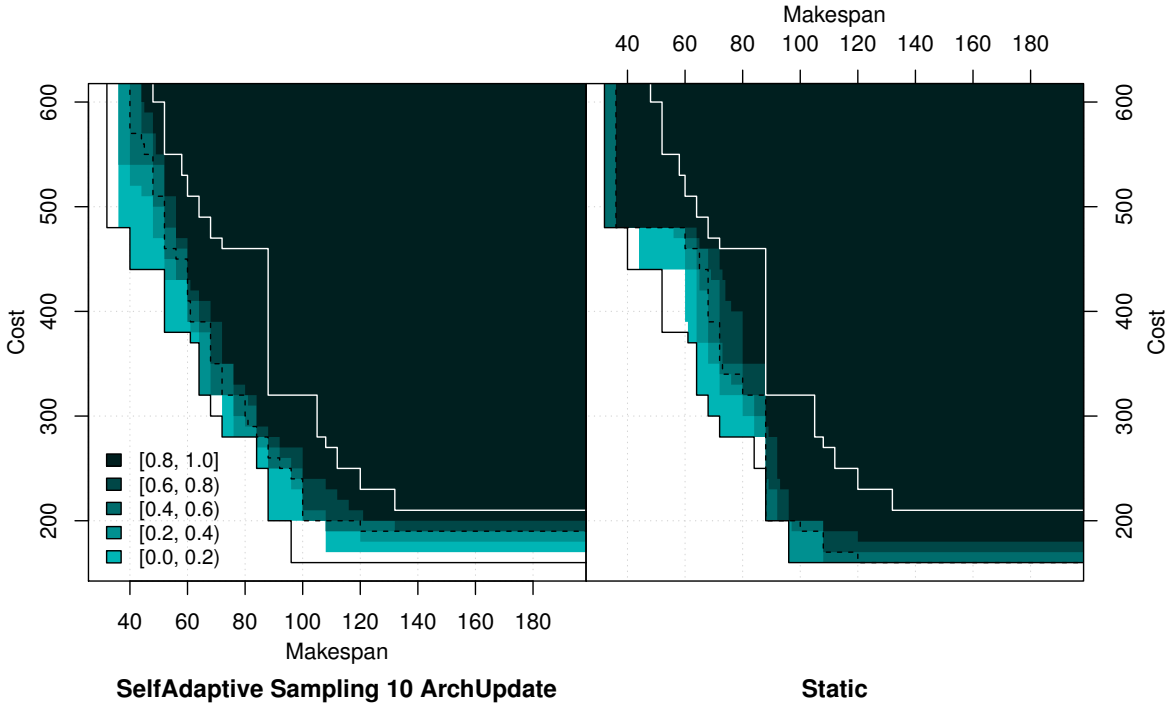


FIGURE 4.242: Instance Zeno9 : AutoAdaptative comparée à Statique.

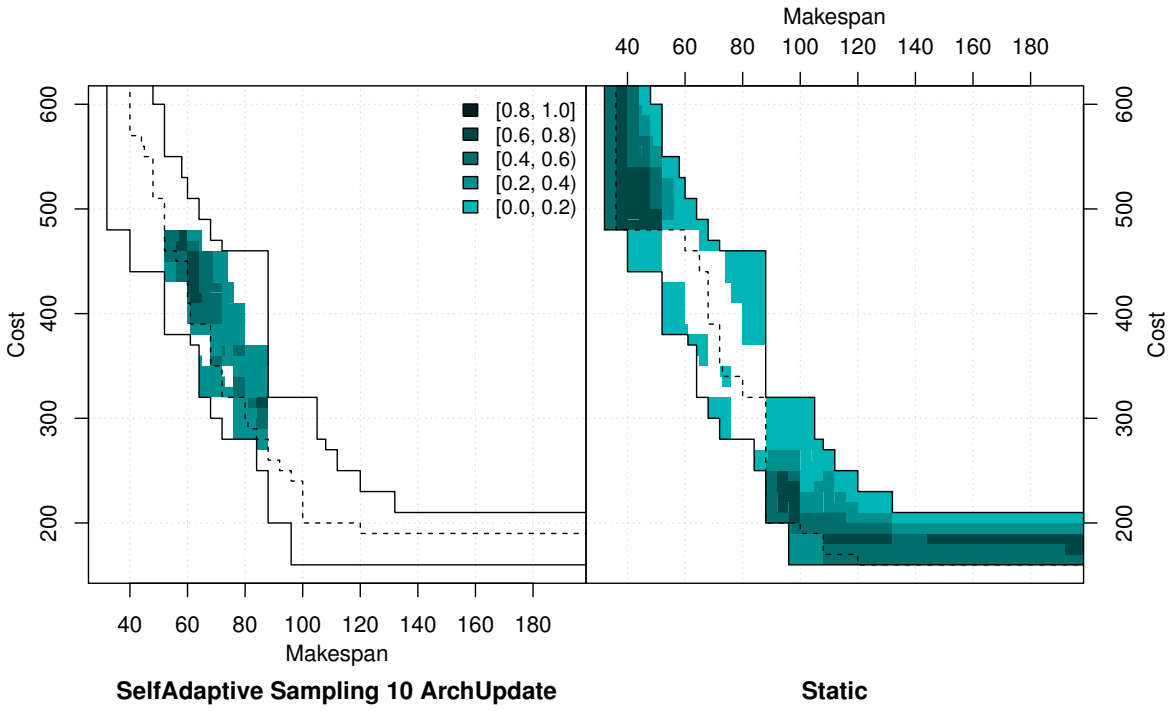
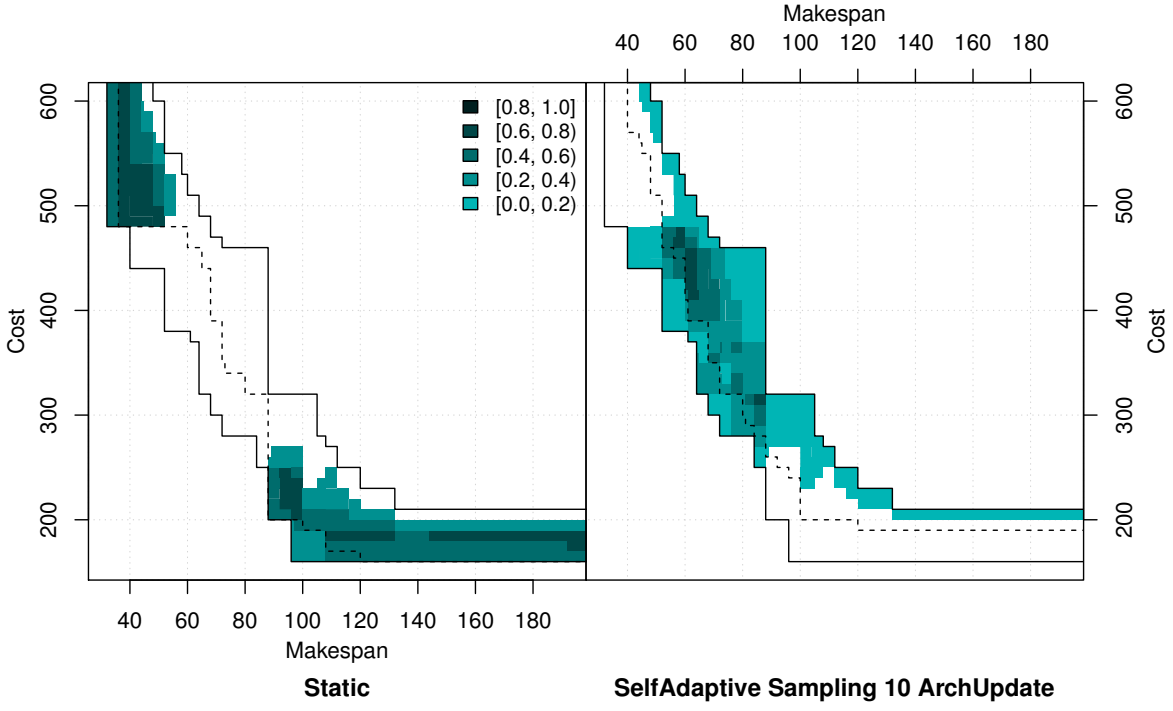


FIGURE 4.243: Instance Zeno9 : Statique comparée à AutoAdaptative.



4.8.6 Conclusion

Il semblerait que l'algorithme génétique apporte peu au processus en général, si ce n'est de la diversité de manière grandement incertaine. Les résultats préliminaires sur l'échantillonnage permettent de montrer qu'il permet d'atteindre des zones normalement plus difficiles (les zones centrales du front), au détriment des valeurs extrêmes. Ceci semble plus dû à la manière dont agit le solveur local : en règle générale YAHSP réussit facilement à trouver des plans de faible *makespan* ou de faible coût, impliquant évidemment une valeur importante pour le second objectif. En échantillonnant on donne plus de chance de trouver des plans aux valeurs « intermédiaires » ce qui doit en partie expliquer les résultats. Un travail futur pourrit s'intéressait à une manière de changer le comportement de l'évaluation (b_{max}, k, stratégie) au cours du temps afin de réussir à obtenir le meilleur de chaque configuration.

Ces résultats montrent également qu'il est possible d'atteindre de bons résultats, statistiquement équivalents, en utilisant un mécanisme d'évaluation qui résout un problème annexe, dont le résultat ne provient pas d'une solution faisable du problème initial.

Évidemment, beaucoup d'hypothèses seraient à confirmer d'une part sur les instances larges de MultiZenoTravel mais aussi d'autre part sur d'autres problèmes afin de discerner le comportement propre au problème MultiZenoTravel et le comportement propre à $\text{DaE}_{\text{YAHSP}}$.

Chapitre 5

Conclusion

Il y a bon nombre de conclusions à tirer de ce projet de fin d'étude. D'une part parce qu'il s'est articulé autour de deux parties distinctes, et parce qu'il pose plus questions qu'il n'offre de réponses, tout en, je pense, offrant quelques perspectives.

Tout d'abord, il fournit une extension multi-objectif du problème de planififcation temporelle ZenoTravel issu de la série *IPC*, nommée MultiZenoTravel, et propose un algorithme de résolution exacte du problème, basé sur la structure des plans optimaux au sens de Pareto. L'étude de cette structure conduit à trouver des moyens efficaces d'énumérer implicitement l'ensemble des solutions. Une implémentation appelée ZenoSolver, a été réalisée en C++, standard 2011, et a permis d'obtenir un large panel d'instances, aux caractéristiques très différentes (convexe, concave, répartition uniforme ou en *cluster*, symétrique, asymétrique,...) et à la difficulté variable.

Nous avons sélectionné et généré des instances dites « Larges » pour servir de représentants pour la suite MultiZenoTravel dans le but d'obtenir un jeu supplémentaire d'instances de référence pour le problème de planification temporelle, avec la particularité de connaître le front exact. Nous fournissons également des résultats préliminaires de DaE$_{\text{YAHSP}}$ sur ces instances, ainsi qu'une comparaison avec une méthode de type « aggrégation » basique pour démontrer les capacités du solveur sur les instances concaves.

La seconde partie dédiée à l'optimisation des paramètres a consisté dans un premier temps à établir des *runs* de références avec DaE$_{\text{YAHSP}}$ sur les instances « Larges » et avec une version du logiciel originale, après une optimisation *offline* des paramètres. La

problématique qui nous a intéressés concerne le choix de l'objectif à passer à YAHSP lors de l'évaluation. Ces *runs* de référence ont servi à la comparaison avec diverses stratégies. Chronologiquement, les premiers développements se sont focalisés sur des stratégie adaptative, utilisant de l'information au cours de la recherche pour inférer sur la meilleure décision à prendre. Les résultats étant quelque peu décevants, un stratégie dite ≪ gloutonne ≫ a été mise en place, où tous les choix sont essayé et le meilleur est retenu. Des résultats intéressants et l'analyse de la dynamique du processus au travers de différentes métriques m'a poussé à chercher à identifier exactement le rôle du processus d'évolution et du solveur local dans la résolution globale du problème. Cela m'a amené à proposer une méthode d'échantillonnage pour réduire à minima l'incertitude sur la qualité intrinsèque d'un individu dû au comportement stochastique de YAHSPet utiliser une méthode d'évaluation différente dont la quantité affectée à un individu ne résulte pas d'une solution au problème de planification mais une espérance représentant sa capacité à fournir de bonnes solutions. En parallèle, quelques essais ont été conduit sur une stratégie auto-adaptative rudimentaire.

Si l'analyse de toutes ces expérience permet d'obtenir des résultats intéressants, il n'a pas été possible d'obtenir un mécanisme qui dépasse en tout point une stratégie statique consistant à fournir des poids pour les objectifs de manière manuelle. Évidemment, le temps étant contraignant, toutes les combinaisons de facteurs ou toutes les idées n'ont pas pû être testées et les résultats obtenus gagneraient à être confirmé ou informés sur d'autres problèmes de planification.

Enfin, un dernier point accompli durant ce projet et qui ne transparait pas dans ce rapport, c'est le génie logiciel et le développement effectué tout du long. En effet, au début du projet, la version courante de DaE n'était pas fonctionnelle et de nombreuses régressions m'ont obligé à reprendre en grande partie le code et valider statistiquement les résultats obtenus par rapport à ceux des articles précédents dans un premier temps, avant de pouvoir lancer mes propres campagnes de tests.

Annexe A

Annexe A : Instance MultiZenoTravel

A.1 Paramétrisation des instances

A.1.1 Génération des instances

Nous avons généré grâce au ZenoSolver, l'ensemble des instances, avec le Front de Pareto associé, selon les paramètres suivants :

1. **n** : 3,4,5,6.
2. **t** : 3,6,9.
3. **p** : 2,3,4,5.
4. **c et d** :
   - Progression arithmétique : $f_1(i) = 2i$.
   - Progression logarithmique : $f_2(i) = log(i)$.
   - Progression racine carrée : $f_3(i) = \sqrt{i}$.
   - Progression exponentielle : $f_4(i) = 2^i$.

Remarque : Attention, $p > t$.

Pour la génération de c et d, nous avons choisi de ne garder que les combinaisons suivantes :

1. Arithmétique / Arithmétique

2. Logarithmique / Logarithmique
3. Exponentielle / Exponentielle
4. Racine / Racine
5. Arithmétique / Exponentielle
6. Exponentielle / Logarithmique

Cela provient du fait que les autres combinaisons génèrent des instances dont le front est proche d'une autre combinaison.

Au final, cela conduit à 48 instances par combinaisons de c et d, et donc 288 instances. Nous pouvons noter que le temps de résolution exacte des 48 instances pour une combinaison donnée est de l'ordre de 2 minutes.

A.1.2 Autres paramétres possibles

ZenoSolver permet de controler un peu plus finement, pour une fonction de génération du vecteur d ou c, la valeur obtenue par un facteur d'échelle et un facteur de translation. Très simplement, on peut paramétrer a et b tels que $d_i = af(i) + b$ avec des valeurs par défaut, respectivement, 1 et 0. Cela permet d'éviter un phénomène d'applatissement par exemple, comme c'est le cas en utilisant une fonction exponentielle avec une progression arithémtique (cependant, la plupart des solveurs renormalisent les objectifs eux-même pour éviter ce problème).

A.1.3 Commentaires sur les instances

A.1.3.1 Artihmétique / Arithmétique

Les précédentes études des instances MultiZenoTravel n'avaient pas fait varier le nombre avion et on constatait à chaque fois un front linéaire. Cependant, la figure A.1 montre que selon certaines valeurs de p le front n'est plus linéaire mais convexe. Il faudra alors faire attention aux instances à choisir comme représentantes du front linéaire.

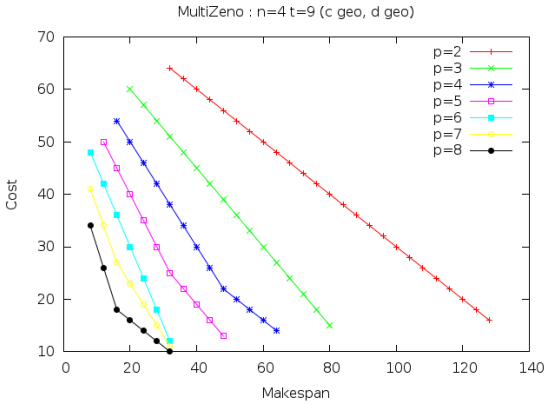


Figure A.1: Front d'instances MultiZeno où le paramètre p varie.

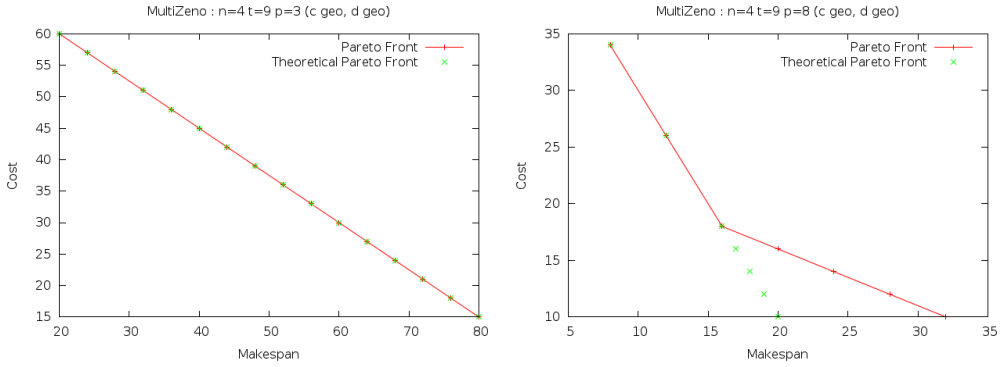


Figure A.2: Front de Pareto vs Front théorique, pour différentes valeur de p

A.1.3.2 Logarithmique / Logarithmique

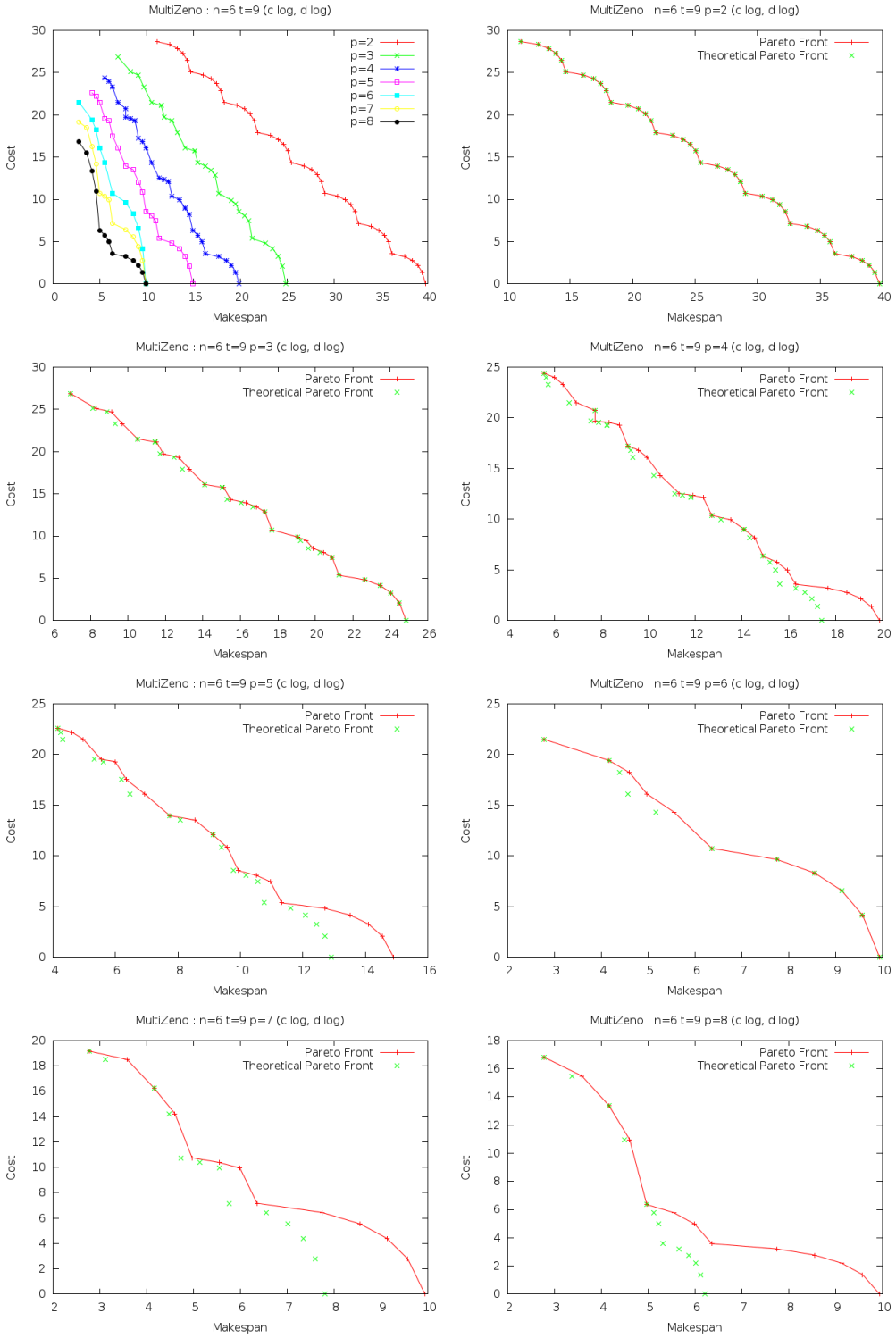


Figure A.3: Génération Logarithmique / Logarithmique

A.1.3.3 Racine / Racine

Si le front est généralement concave par morceau avec une tendance linéaire, on observe d'importantes différences dans la forme du front en fonction de la valeur de p.

Dans le cas où $p = 2$ le front théorique est parfaitement atteint, le front est régulier, de tendance parfaitement linéaire et concave par morceau. Si l'on considère $p = 3$ alors il survient une petite irrégularité concave dans les valeurs élevées du coût mais dans l'ensemble on peut faire la même description que pour le cas $p = 2$.

Avec $p = 4$, une partie du Front de Pareto s'éloigne du front théorique modifiant la tendance qui est alors convexe. L'irrégularité du cas précédent a cependant disparu.

Le cas $p = 5$ est un peu plus suprenant puisqu'il présente une première partie linéaire, puis une seconde partie convexe et enfin une partie clairement concave pour les faibles valeurs du coût. Il est assez difficile de donner la tendance de la courbe mais elle semble tout de même assez linéaire.

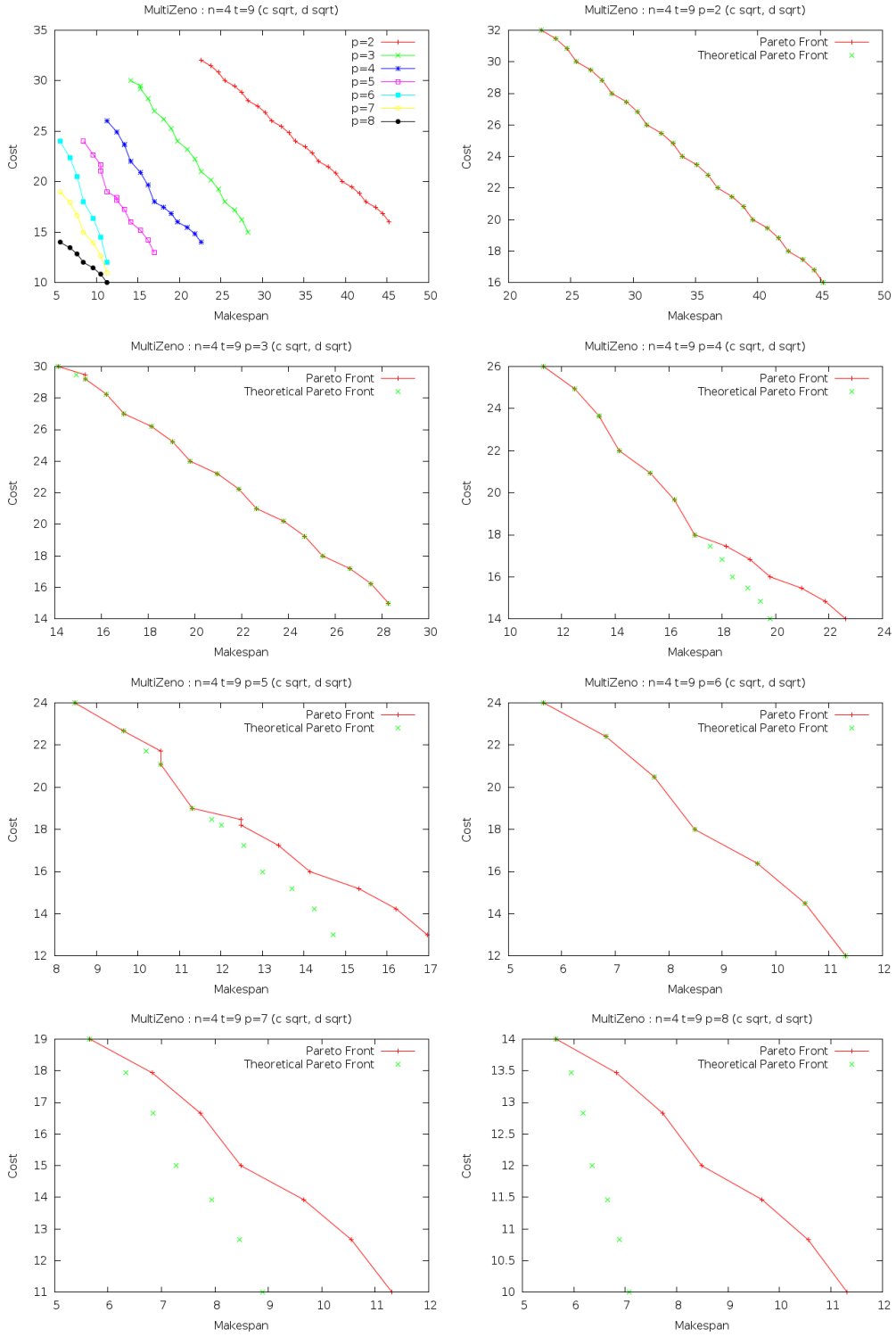


Figure A.4: Génération Racine / Racine

Pour $p = 6, 7, 8$ les choses sont beaucoup plus classiques. Plus aucune irrégularité, seule la distance au front théorique varie.

A.1.3.4 Exponentielle / Exponentielle

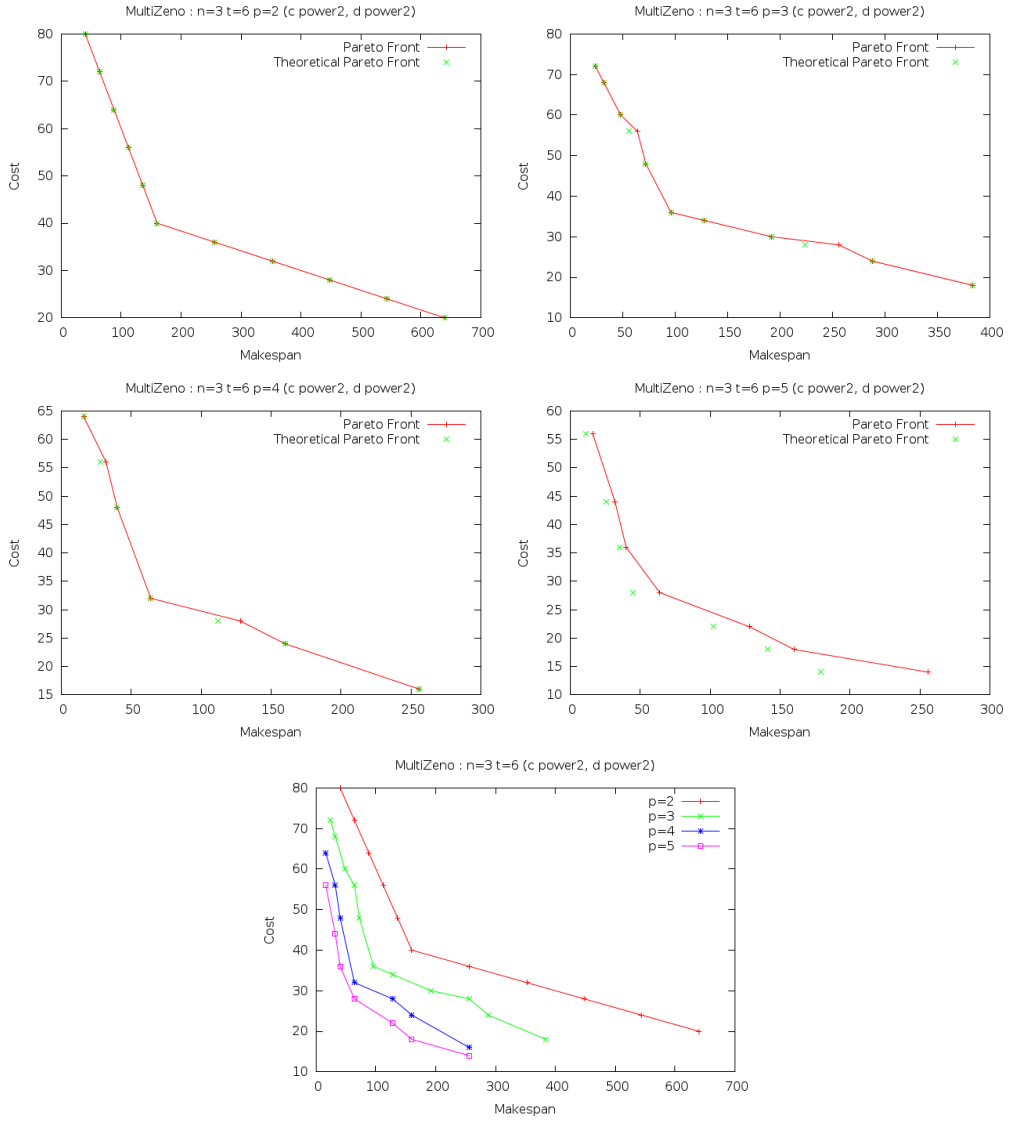


Figure A.5: Génération Exponentielle / Exponentielle

A.1.3.5 Arithmétique / Exponentielle

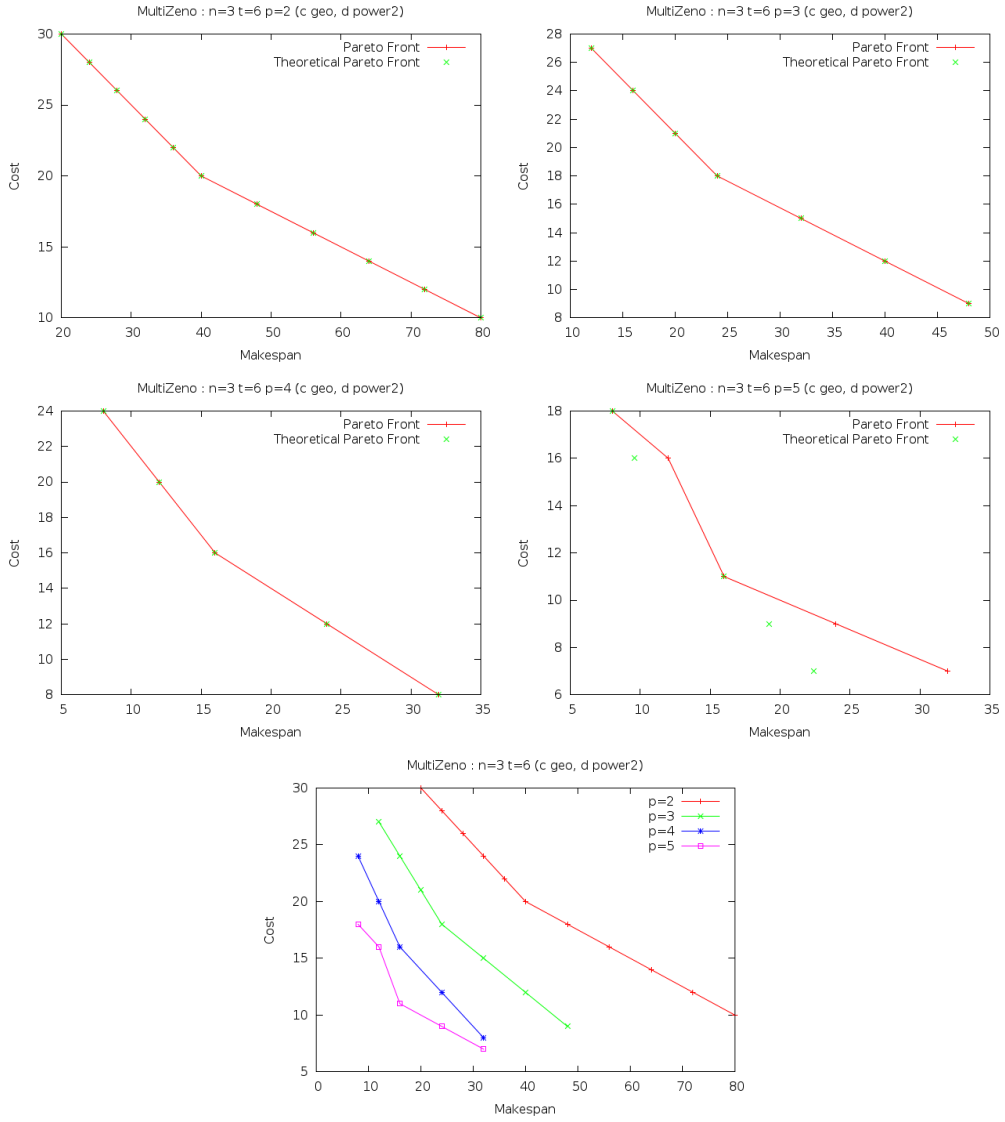


Figure A.6: Génération Arithmétique / Exponentielle

A.1.3.6 Exponentielle / Logarithmique

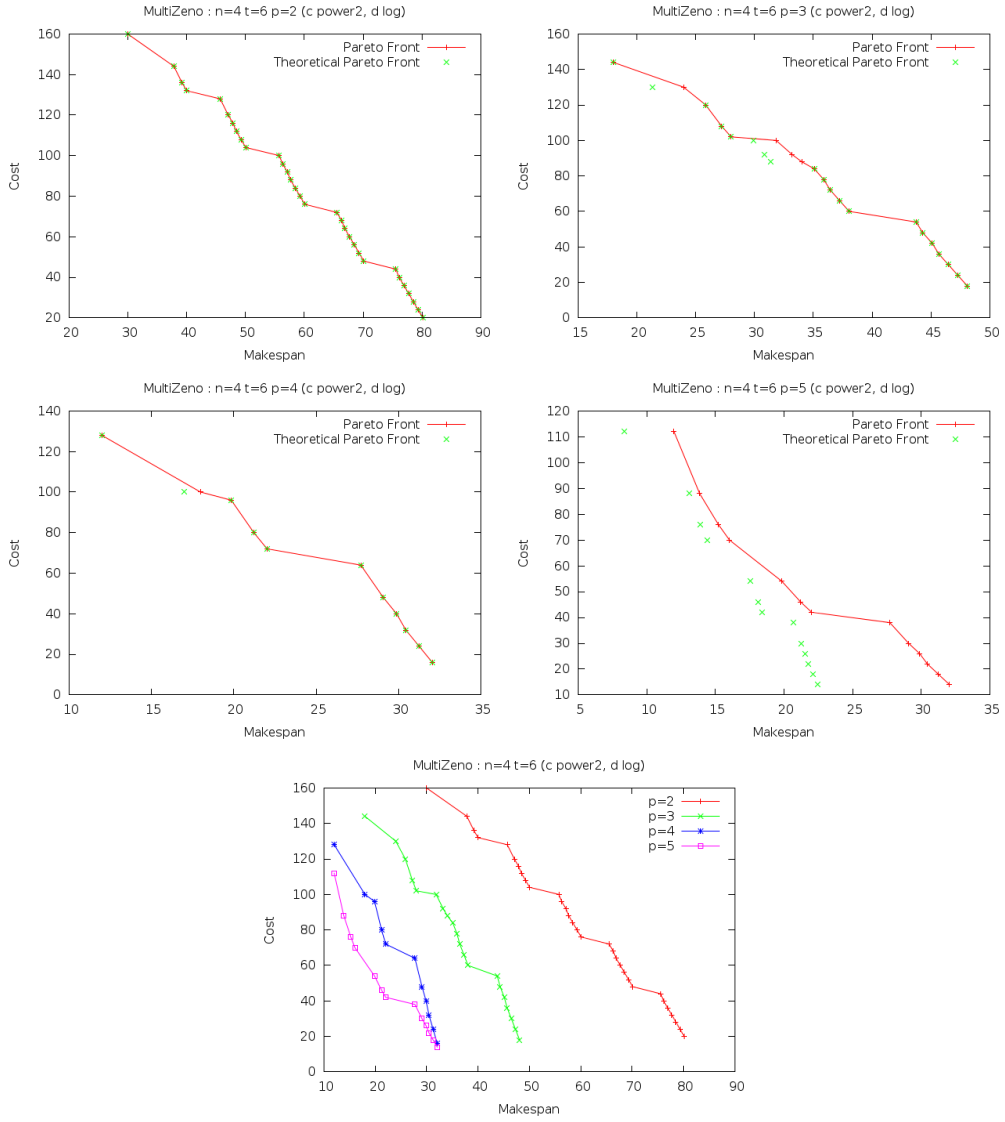


Figure A.7: Génération Exponentielle / Logarithmique

A.1.3.7 Autres instances

Quelques autres instances ont été générées avec une fonction plus particulière : $f(i) = aE[\frac{i}{b}] + \frac{(i \ mod \ b)}{c}$ ou $g(i) = aE[\frac{i}{b}] + \frac{(-1)^{E[\frac{i}{b}]}(i \ mod \ b)}{c}$ ou . Avec a un paramètre contrôlant la taille des discontinuité, b la fréquence des discontinuités et c la pente de chaque segment de droite. La fonction g alternant le sens de variation pour chacun des segments contrairement à f.

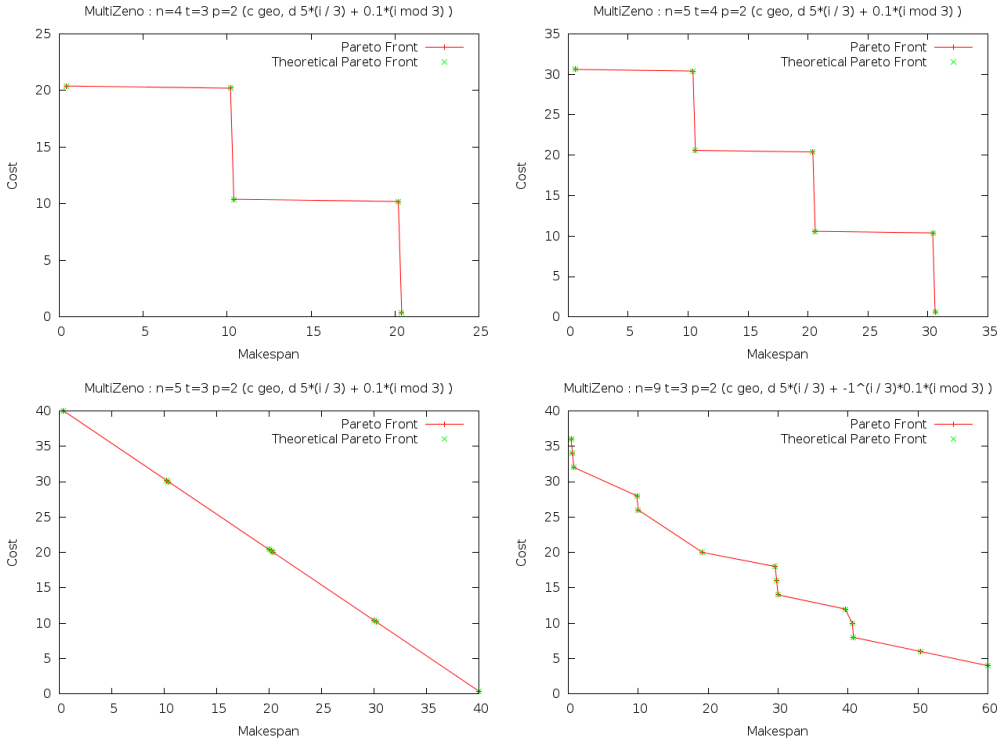


FIGURE A.8: Instances diverses

Table des figures

Liste des tableaux

Bibliographie

[1] P. Haslum and H. Geffner. Admissible Heuristics for Optimal Planning. In *Proc. 5th Int. Conf. on AI Planning and Scheduling (AIPS 2000)*, pages 140–149. AAAI Press, 2000.

[2] Vincent Vidal. A Lookahead Strategy for Heuristic Search Planning. In R. Brafman et al., editor, *Proc. of the* 14^{th} *ICAPS*, pages 150–159. AAAI Press, 2004.

[3] Cristina Bazgan Florian Jamain and Daniel Vanderpooten. Approximation de taille minimale de l'ensemble de pareto de problèmes multicritères., 2013.

[4] M. Grabisch. L'utilisation de l'intégrale de choquet en aide multicritère à la décision. *Newsletter of the European Working Group "Multicriteria Aid for Decisions"*, 3 (14) :143–174, 2006.

[5] J.-L. Marichal. On Choquet and Sugeno integrals as aggregation functions. In M. Grabisch, T. Murofushi, and M. Sugeno, editors, *Fuzzy Measures and Integrals*, pages 247–272. Physica Verlag, Heidelberg, 2000.

[6] M. Grabisch and M. Roubens. Application of the Choquet integral in multicriteria decision making. In M. Grabisch, T. Murofushi, and M. Sugeno, editors, *Fuzzy Measures and Integrals - Theory and Applications*, pages 348–374. Physica Verlag, 2000.

[7] D. Denneberg. Non-additive measure and integral, basic concepts and their role for applications. In M. Grabisch, T. Murofushi, and M. Sugeno, editors, *Fuzzy Measures and Integrals – Theory and Applications*, pages 42–69. Physica Verlag, Heidelberg, 2000.

[8] D. Dubois, H. Prade, and R. Sabbadin. Qualitative decision theory with Sugeno integrals. In M. Grabisch, T. Murofushi, and M. Sugeno, editors, *Fuzzy Measures*

and Integrals – Theory and Applications, pages 314–332. Physica Verlag, Heidelberg, 2000.

[9] David E. Goldberg and Jon Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*, pages 41–49, Hillsdale, NJ, USA, 1987. L. Erlbaum Associates Inc. ISBN 0-8058-0158-8. URL http://dl.acm.org/citation.cfm?id=42512.42519.

[10] Kenneth Alan De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* PhD thesis, Ann Arbor, MI, USA, 1975. AAI7609381.

[11] Tobias Blickle. *Theory of Evolutionary Algorithms and Application to System Synthesis.* PhD thesis, Swiss Federal Institute of Technology, Zurich, November 1996. URL http://www.handshake.de/user/blickle/publications/diss.pdf.

[12] Kalyanmoy Deb. *Multi-Objective Optimization Using Evolutionary Algorithms.* John Wiley & Sons, Inc., New York, NY, USA, 2001. ISBN 047187339X.

[13] Matthew Cary. Towards optimal epsilon-approximate nearest neighbor algorithms in constant dimensions. *J. Algorithms*, 41 :page,, 2001.

[14] Carlos M. Fonseca and Peter J. Fleming. Genetic algorithms for multiobjective optimization : Formulation, discussion and generalization, 1993.

[15] N. Srinivas and Kalyanmoy Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2 :221–248, 1994.

[16] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization : Nsga-ii. pages 849–858. Springer, 2000.

[17] Jeffrey Horn, Nicholas Nafpliotis, and David E. Goldberg. Multiobjective optimization using the niched pareto genetic algorithm. Technical report, 1993.

[18] Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms : A comparative case study and the strength pareto approach, 1999.

[19] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. Spea2 : Improving the strength pareto evolutionary algorithm. Technical report, 2001.

[20] Joshua D. Knowles and David W. Corne. Approximating the nondominated front using the pareto archived evolution strategy. *Evol. Comput.*, 8(2) :149–172, June 2000. ISSN 1063-6560. doi : 10.1162/106365600568167. URL http://dx.doi.org/10.1162/106365600568167.

[21] David Corne, Joshua D. Knowles, and Martin J. Oates. The pareto envelope-based selection algorithm for multi-objective optimisation. In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, PPSN VI, pages 839–848, London, UK, UK, 2000. Springer-Verlag. ISBN 3-540-41056-2. URL http://dl.acm.org/citation.cfm?id=645825.669102.

[22] Eckart Zitzler and Simon Künzli. Indicator-based selection in multiobjective search. In *in Proc. 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII*, pages 832–842. Springer, 2004.

[23] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Combining convergence and diversity in evolutionary multi-objective optimization. *Evolutionary Computation*, 10(3) :263–282, 2002.

[24] K. Deb, M. Mohan, and S. Mishra. A Fast Multi-objective Evolutionary Algorithm for Finding Well-Spread Pareto-Optimal Solutions. KanGAL report 2003002, Indian Institute of Technology, Kanpur, India, 2003.

[25] Viviane Grunert da Fonseca, Carlos M. Fonseca, and Andreia O. Hall. Inferential performance assessment of stochastic optimisers and the attainment function. pages 213–225. Springer-Verlag.

[26] Carlos M. Fonseca, Viviane Grunert da Fonseca, and Luís Paquete. Exploring the performance of stochastic multiobjective optimisers with the second-order attainment function. In *Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization*, EMO'05, pages 250–264, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 3-540-24983-4, 978-3-540-24983-2. doi : 10.1007/978-3-540-31880-4_18. URL http://dx.doi.org/10.1007/978-3-540-31880-4_18.

[27] Joshua Knowles. A summary-attainment-surface plotting method for visualizing the performance of stochastic multiobjective optimizers. In *Proceedings of the 5th International Conference on Intelligent Systems Design and Applications*, ISDA

'05, pages 552–557, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2286-06. doi : 10.1109/ISDA.2005.15. URL http://dx.doi.org/10.1109/ISDA.2005.15.

[28] Mostepha Redouane Khouadjia, Marc Schoenauer, Vincent Vidal, Johann Dréo, and Pierre Savéant. Multi-Objective AI Planning : Evaluating DAE-YAHSP on a Tunable Benchmark. In R.C. Purshouse et al., editor, *Proc. EMO*, pages 36–50. LNCS 7811, Springer Verlag, 2013.

[29] Mostepha Redouane Khouadjia, Marc Schoenauer, Vincent Vidal, Johann Dréo, and Pierre Savéant. Pareto-Based Multiobjective AI Planning. In Francesca Rossi, editor, *Proc. IJCAI*. AAAI Press, 2013.

[30] Donald E. Knuth. *The Art of Computer Programming, Generating All Tuples and Permutations*, volume 4. 2005.

[31] Mostepha Redouane Khouadjia, Marc Schoenauer, Vincent Vidal, Johann Dréo, and Pierre Savéant. Multi-Objective AI Planning : Comparing Aggregation and Pareto Approaches. In Martin Middendorf and Christian Blum, editors, *Proc. EvoCOP*, pages 202–213. LNCS 7832, Springer Verlag, 2013.

[32] Qingfu Zhang and Hui Li. A multi-objective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6) :712–731, 2007.

[33] Michal Sroka and Derek Long. Exploring Metric Sensitivity of Planners for Generation of Pareto Frontiers. In K. Kersting and M. Toussaint, editors, *The Sixth "Starting Artificial Intelligence Research" Symposium (STAIRS 2012)*, pages 306–317. IOS Press, 2012.

[34] Nikolaus Hansen, Nikolaus Hansen, Andreas Ostermeier, and Andreas Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies : The covariance matrix adaptation. pages 312–317. Morgan Kaufmann, 1996.

[35] Raphaël Cerf. *Une théorie asymptotique des algorithmes génétiques*. PhD thesis, March 1994.

[36] F. Hutter, H. H. Hoos, and T. Stützle. Automatic algorithm configuration based on local search. In *Proc. of the Twenty-Second Conference on Artifical Intelligence (AAAI '07)*, pages 1152–1157, 2007.

[37] Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown, and Thomas Stützle. ParamILS : an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36 :267–306, October 2009.

[38] Mostepha Redouane Khouadjia, Marc Schoenauer, Vincent Vidal, Johann Dréo, and Pierre Savéant. Quality Measures of Parameter Tuning for Aggregated Multi-Objective Temporal Planning. In P. Pardalos et al., editor, *Proc. LION7*, pages 341–356. LNCS 7997, Springer, 2013.

[39] J. Knowles, L. Thiele, and E. Zitzler. A tutorial on the performance assessment of stochastic multiobjective optimizers. 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland, February 2006. revised version.

[40] Aldeida Aleti. *An adaptive approach to controlling parameters of evolutionary algorithms*. PhD thesis, 2005.

[41] Dirk Thierens. An adaptive pursuit strategy for allocating operator probabilities. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, GECCO '05, pages 1539–1546, New York, NY, USA, 2005. ACM. ISBN 1-59593-010-8. doi : 10.1145/1068009.1068251. URL http://doi.acm.org/10.1145/1068009.1068251.

[42] Álvaro Fialho, Luis Da Costa, Marc Schoenauer, and Michèle Sebag. Analyzing bandit-based adaptive operator selection mechanisms. *Annals of Mathematics and Artificial Intelligence*, 60(1-2) :25–64, October 2010. ISSN 1012-2443. doi : 10.1007/s10472-010-9213-y. URL http://dx.doi.org/10.1007/s10472-010-9213-y.

[43] Blai Bonet, Gábor Loerincs, and Héctor Geffner. A robust and fast action selection mechanism for planning. In *In Proceedings of AAAI-97*, pages 714–719. MIT Press, 1997.