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

Alexandre QUEMY

25 avril 2014

Table de matières

Avant-propos

ANR Descarwin

Projet Descarwin de l'Agence Nationale de la Recherche (ANR).

Les différentes entités du projet

- Équipe TAO, Inria Saclay & LRI Paris-Sud Université, Orsay.
- ONERA-DSCD, Toulouse.
- THALES Research & Technologies, Palaiseau.

Mono-objectif vs multi-objectifs

Problème mono-objectif

$$\min_{x \in \Omega} f(x),\ \forall i\ g_i(x) \leq 0$$

- Ω espace des variables de décision (ex. $\mathbb{R}^m$).
- $f : \Omega \rightarrow \mathbb{K}$, fonction objectif avec $dim\ \mathbb{K} = 1$.
- g_i, contrainte sur Ω.

Une solution unique dans l'espace objectif (ou aucune).

Mono-objectif vs multi-objectifs

Problème mono-objectif

$$\min_{x \in \Omega} f(x), \ \forall i \ g_i(x) \leq 0$$

Problème multi-objectifs

$$\min_{x \in \Omega} f(x), \ \forall i \ g_i(x) \leq 0$$

- $f : \Omega \rightarrow \mathbb{K}$, *vecteur* objectif avec *dim* $\mathbb{K} > 1$.

Comment définir une solution optimale ?

Optimalité au sens de Pareto

Relation de dominance

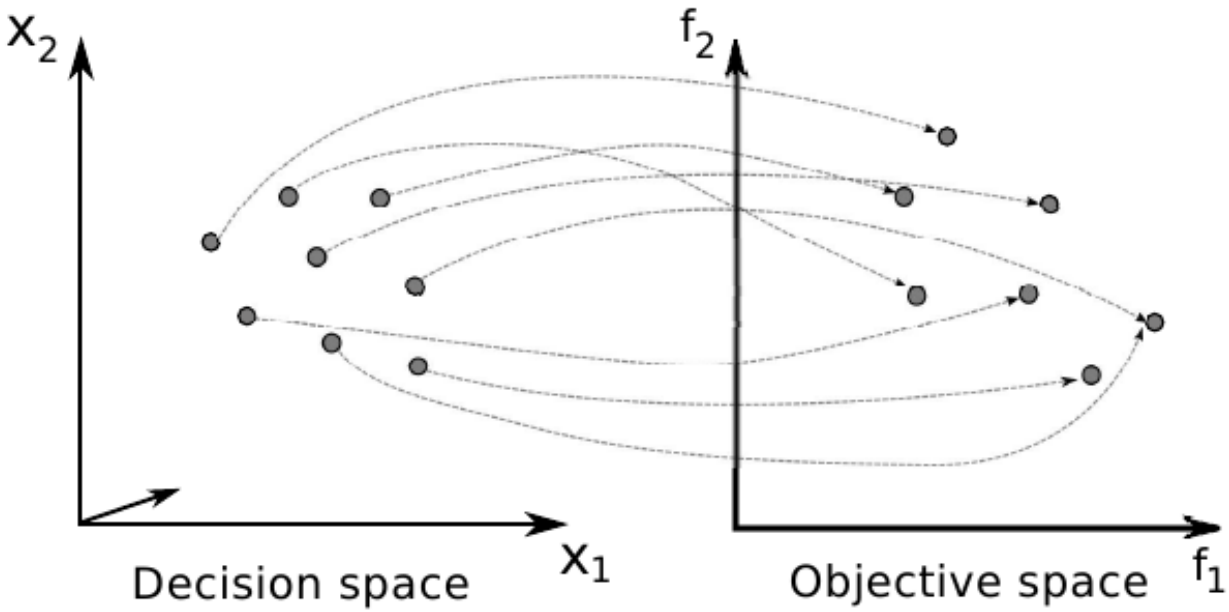
Soit $x, y \in \Omega$, x domine y si $\begin{cases} \forall i \in \{1, \ldots, n\}, f_i(x) \succeq f_i(y) \\ \exists j \in \{1, \ldots, n\}, f_j(x) \succ f_j(y) \end{cases}$

Front de Pareto

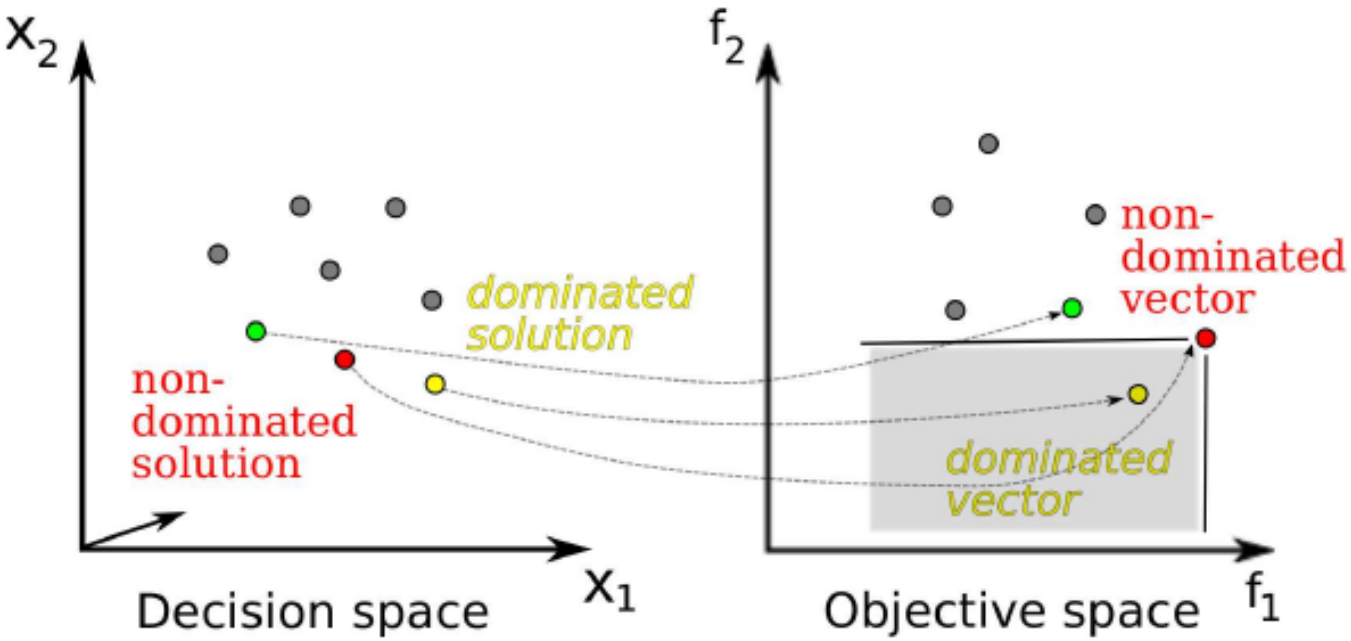
$$x \in FP \Leftrightarrow \nexists y \in \Omega,\ y \succ x$$

- Un ensemble de solutions : ensemble de compromis.

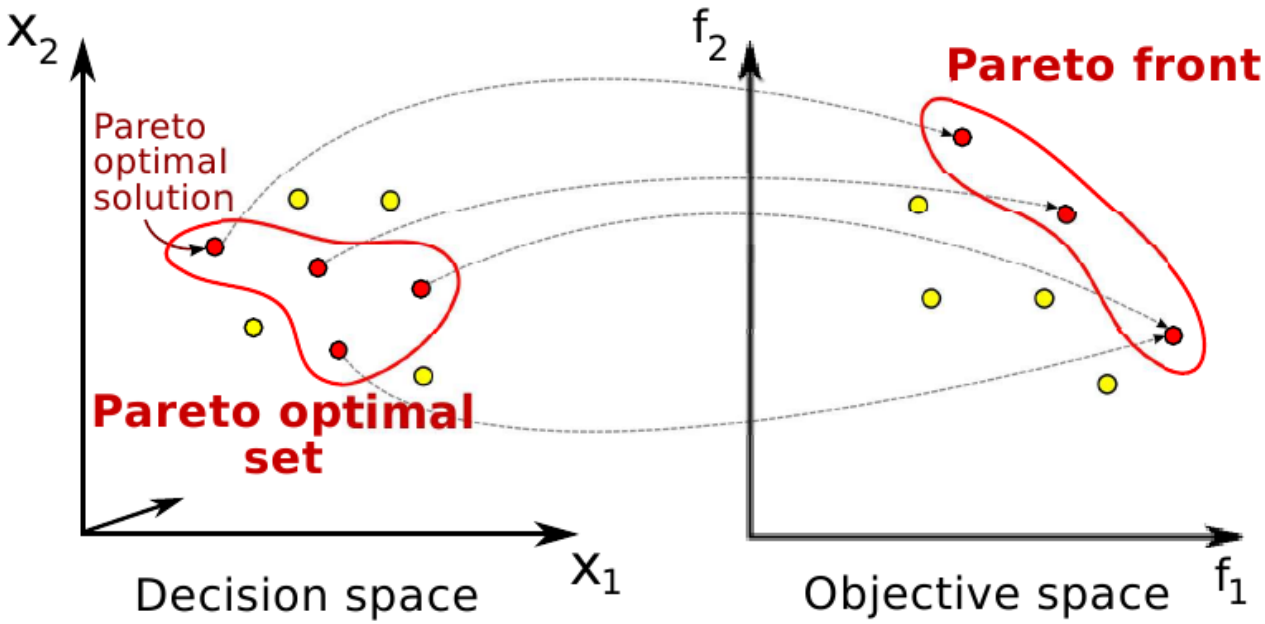
Espace de décisions / Espace des objectifs



Dominance


x2
x1
Decision space
dominated
solution
non-
dominated
solution
f2
f1
Objective space
non-
dominated
vector
dominated
vector

Front de Pareto



Problème de Planification

Modèle d'états

- S ensemble des états.
- s_0, état initial du problème.
- $S_G \subset S$, ensemble d'états à atteindre (buts).
- $A(s)$, ensemble des actions applicables à l'état s.
- $f : A \times S \rightarrow S' \subseteq S$, fonction de transition.

Un *plan* est une suite d'actions transformant s_0 en $s_G \in S_G$.

Problème de Planification Temporel (PPT)

Concurrence des actions

Le PPT étendent le problème classique de planification par des actions ayant des durées et pouvant s'exécuter en parallèle.

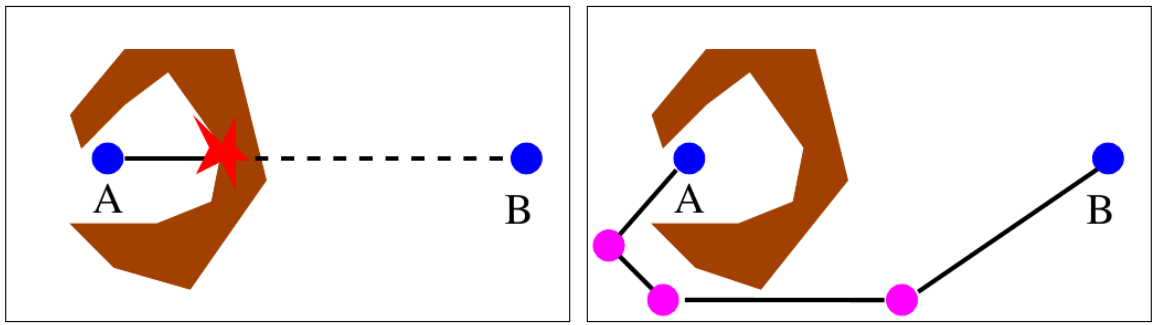
Langage PPDL pour le problèmes de planification

Langage standardisé pour la description de domaines et d'instances du problème de planification.

Divide-And-Evolve

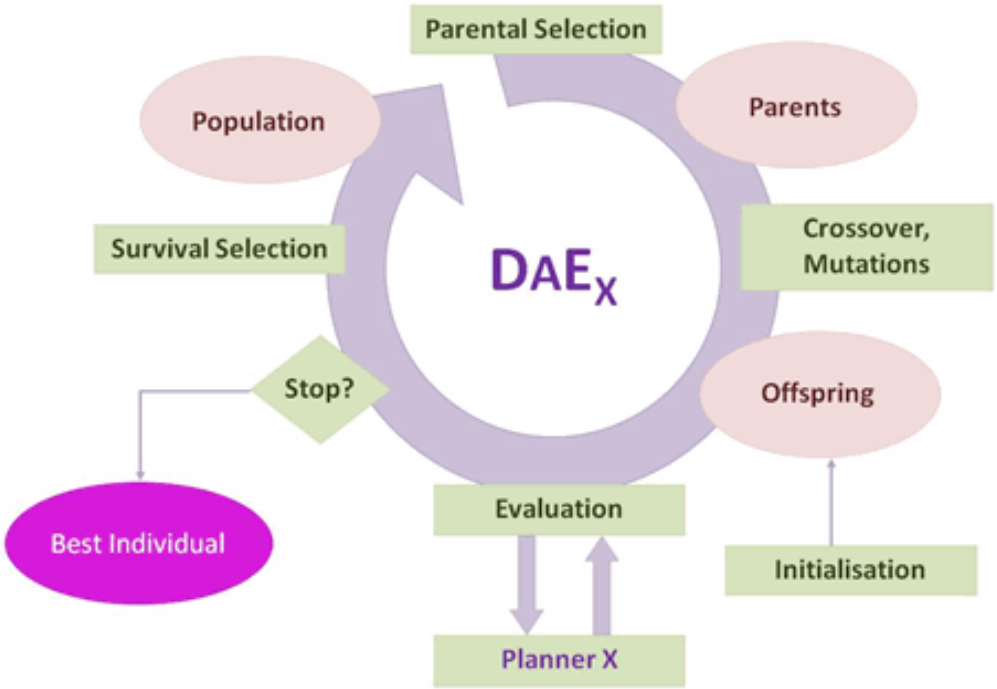
Paradigme DaE

- Approche mémétique : hybridation d'un algorithme génétique avec un algorithme *ad-hoc*.
- Découpe le problème S en sous-problèmes S_i simples.
- S_i résolus par un solveur local (ex. YAHSP pour DAE).



(Variable length) Genotype = (●1, ●2, ●3)

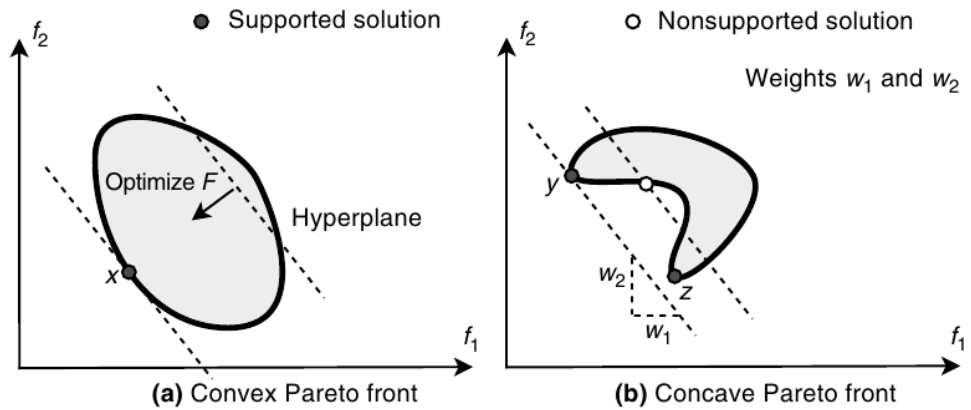
Divide-And-Evolve



Divide-And-Evolve

Quelques remarques

- Pas besoin d'un solveur embarqué optimal.
- Approche efficace en pratique : vainqueur de la session sous-optimale déterministe temporelle (IPC11).
- Premier solveur multi-objectifs de PPT avec approche Pareto.



Les enjeux du projet

Les problématiques

- Pas de problèmes références multi-objectifs pour les PPT.
- Quels sont les paramètres optimaux pour DaE ?
- Cas particulier de la stratégie à passer au solveur local.

Objectifs et travail accompli

Objectifs

Créer des instances dont :

- le front exact est connu.
- la forme du front est variée.
- la difficulté est variable.

Travail accompli

- Généralisation du problème mono-objectif ZenoTravel.
- Résolution du problème général sous hypothèses faibles.
- Développement d'un solveur efficace pour ce problème.

Problème ZenoTravel

Le domaine

- n villes centrales formant une clique. Une ville c_I et c_G.
- p avions.
- t passagers.
- $(d_i j)_{1 \leq i,j \leq n}$ durée entre la ville c_i et c_j.
- $(d_I i)_{1 \leq i \leq n}$ et $(d_i G)_{1 \leq i \leq n}$.

Objectif

Trouver un plan qui amène les t passagers de c_I à c_G en un minimum de temps (*makespan*).

Problème MultiZenoTravel

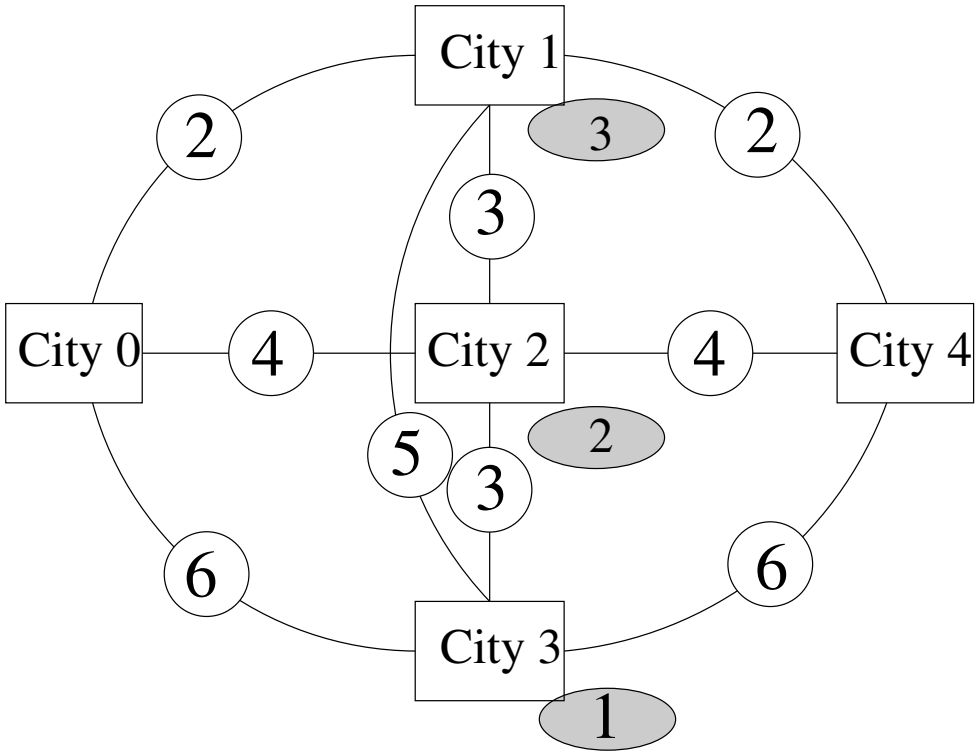
Le domaine

- n villes centrales formant une clique. Une ville c_I et c_G.
- p avions.
- t passagers.
- $(d_i j)_{1 \leq i,j \leq n}$ durée entre la ville c_i et c_j.
- $(d_I i)_{1 \leq i \leq n}$ et $(d_i G)_{1 \leq i \leq n}$.
- $(c_i)_{1 \leq i \leq n}$ coût pour un atterrissage en c_i.

Objectif

Trouver un plan qui amène les t passagers de c_I à c_G en un minimum de temps (*makespan*) et en un *coût* minimum.

Une instance



Les hypothèses

Hypothèse forte

$$\forall (i,j), d_i + d_j < d_{ij}$$

Problème peu réaliste.

Hypothèse faible : inégalité triangulaire

En notant $c_0 = I$ et $c_{n+1} = G$:

$$\forall (i,j,k), d_{ik} \leq d_{ij} + d_{jk}$$

Réaliste mais optimalité non-démontrée... :'(

Propriétés des plans optimaux

Proposition

Un plan Pareto-optimal visite exactement $2t - p$ villes centrales.

Corollaire

Chaque personne visite une seule ville centrale.

Remarque

Les arcs entre villes centrales ne sont jamais utilisés.

Plan Potentiellement Pareto-optimal (PPP)

Un PPP est défini par 2 tuples : $e \in [1, n]^t$ et $w \in [1, n]^{t-p}$.

Propriétés des PPP

PPP *admissible*

Un PPP P est admissible si $P \in E \times W$ avec :

- $E = \{e \in [1, n]^t; \forall i \in [1, t-1], d_{e_i} \geq d_{e_{i+1}}\}$
- $W = \{w \in [1, n]^{t-p}; \forall i \in [1, t-p-1], d_{w_i} \geq d_{w_{i+1}}\}$.

Cardinalité

$$card(E \times W) = \binom{n+t-1}{t}\binom{n+(t-p)-1}{t-p}$$

Mieux que $n^{(2t-p)}$!

Propriétés des PPP

Coût d'un PPP

Soit $(e, w) \in E \times W$, $\text{Cost}((e, w)) = \sum_{e_i \in e} c_{e_i} + \sum_{w_i \in w} c_{w_i}$.

Makespan d'un PPP et bornes

Le makespan d'un PPP est le makespan le plus court parmi tous les plans réalisables à partir de (e, w) et $4t - 2p$ arcs.

$$M_S(C) = 2(\sum_{e_i \in e} d_{e_i} + \sum_{w_i \in w} d_{w_i}) \qquad M_L(C) = \frac{M_S(C)}{p}$$

Propriétés des PPP

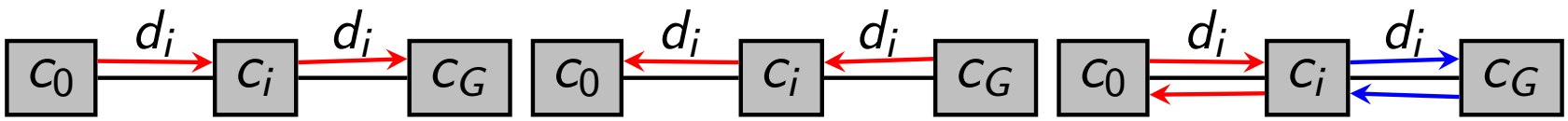
Domination gloutonne au sens de Pareto

Soient 2 PPP C et C', C *domine* C' si :

$$\begin{cases} M_S(C) \leq M_L(C') \\ \text{Cost}(C) \leq \text{Cost}(C') \end{cases}$$

Utile pour élaguer les PPPs ! :)

Makespan optimal d'un PPP



Motifs de vol

- Seuls trois motifs réalisables.
- Seul le motif 3 implique une intéraction entre avions.

Notation pour un PPP donné

- β : nombre de Motifs 3 ($0 \leq \beta \leq t - p$).
- β_{set} : liste de villes pour le motif 3.
- β-powerSet : ensemble des β_{set}.

Makespan optimal d'un PPP

Sous-problème associé à un PPP C

Soit (C, β_{set}) et le PPP C' tel que $e' = e \backslash \beta_{set}$ et $w' = w \backslash \beta_{set}$.
C' est un PPP sans Motif 3, avec $t - \beta$ passagers.

Méthode pour obtenir la borne minimale

Pour chaque couple $((e, w) \in E \times W, \beta_{set} \in \beta\text{-}powerSet)$:

1. Calculer les durées optimales de chaque avion du sous-problème.
2. À partir de ces durées, dispatcher les durées des Motifs 3.

Choisir le makespan minimal obtenu parmi tous les β_{set}.

Les deux étapes peuvent se faire par un algorithme glouton !

Makespan optimal d'un PPP

Preuve (partielle) d'optimalité pour un (C, β_{set})

- Temps incompressible : $T = 4 \sum_{i \in \beta_{set}} d_i + 2 \sum_{i \in \{e', w'\}} d_i$.
- Un plan théorique optimal avec ces Motifs est un plan sans aucun point d'attente.
- La méthode gloutonne donne une répartition optimale du temps entre les p avions. Donc si un plan peut être construit avec un tel makespan, il est optimal pour cette répartition de Motifs.
- Comme il existe une méthode pour construire un tel plan, on conclut que la méthode est optimale.

Complexité de la méthode

Complexité dans le pire des cas

$$2^{t-p}\binom{n+t-1}{t}\binom{n+(t-p)-1}{t-p}$$

En pratique jamais atteinte.

ZenoSolver

ZenoSolver

- Logiciel C++11 pour résoudre MultiZenoTravel.
- Fonctions génératrices des coûts et durées quelconques.
- Facilement parallélisable.
- Faible occupation mémoire (environ la taille du FP).

ZenoSolver

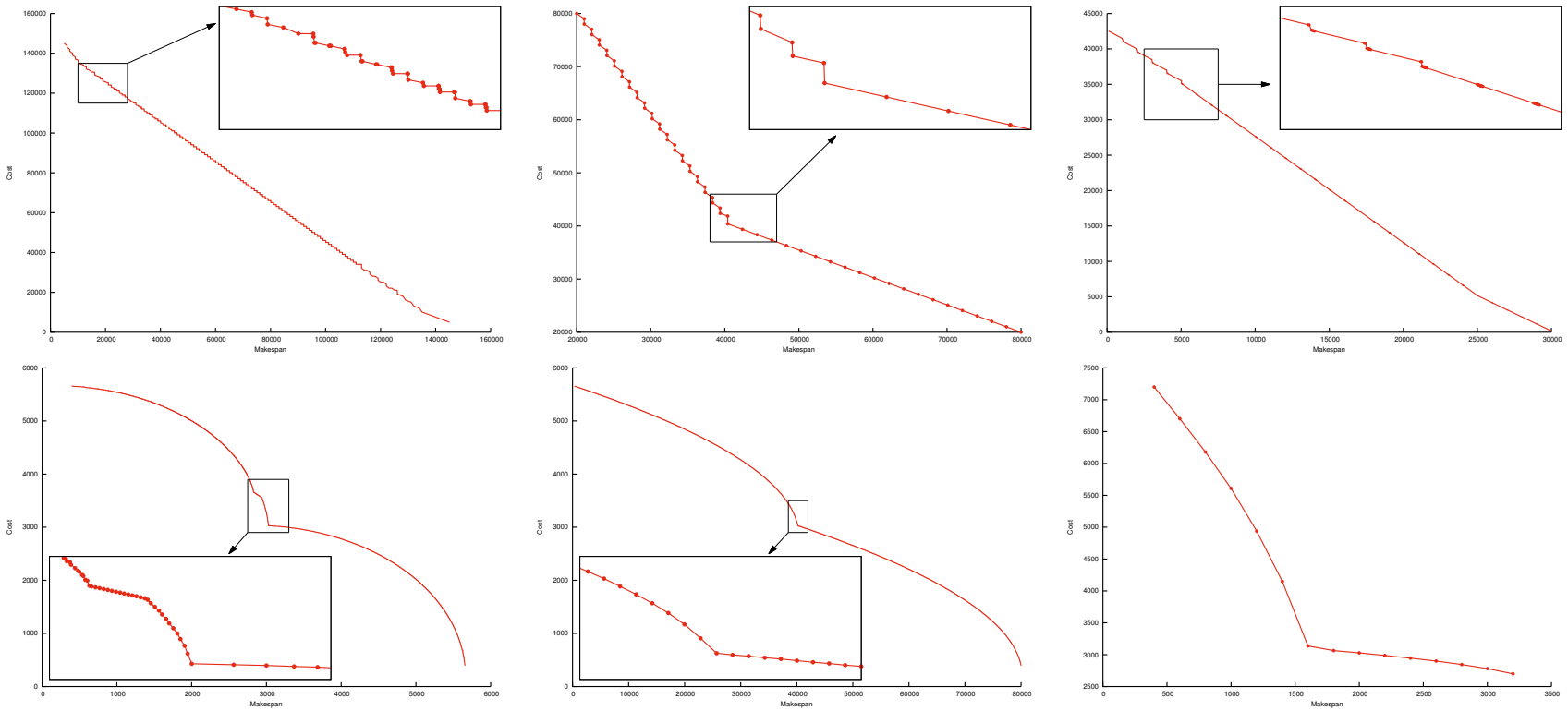
Performances empiriques

Efficacité de l'élagage dépendant de l'instance.

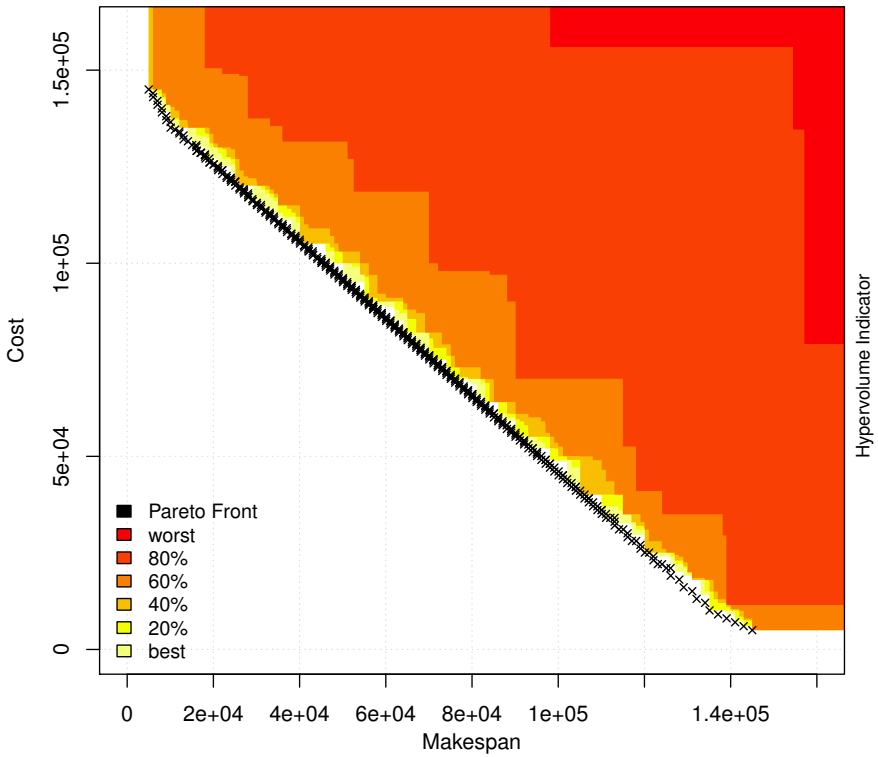
$c_i = \sqrt{i}$ et $d_i = i$

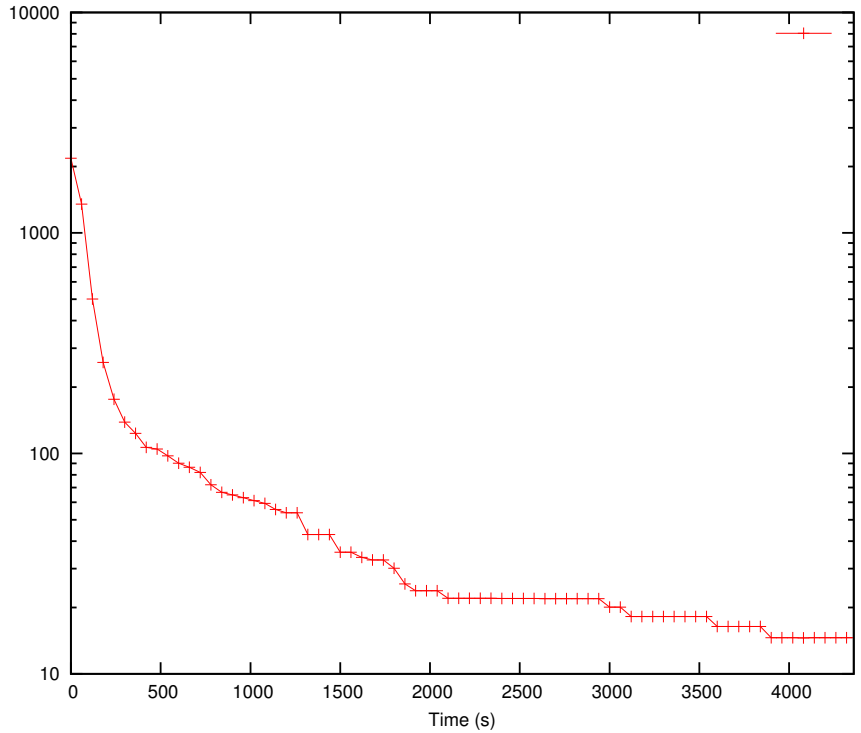
n	t	p	PPP Size	Iterations		S Size	Front Size	Time (ms)	
3	3	2	30	48	36	9	5	0	0
4	4	2	350	770	419	19	10	0	0
5	5	2	4410	12810	3014	33	17	10	3
6	6	2	58212	219912	14739	51	26	223	25
7	7	2	792792	3868032	48512	73	37	6510	163
8	8	2	11042460	69320988	47838	99	65	115977	1709
9	9	2	156434850	1260758070	119686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

ZenoSolver

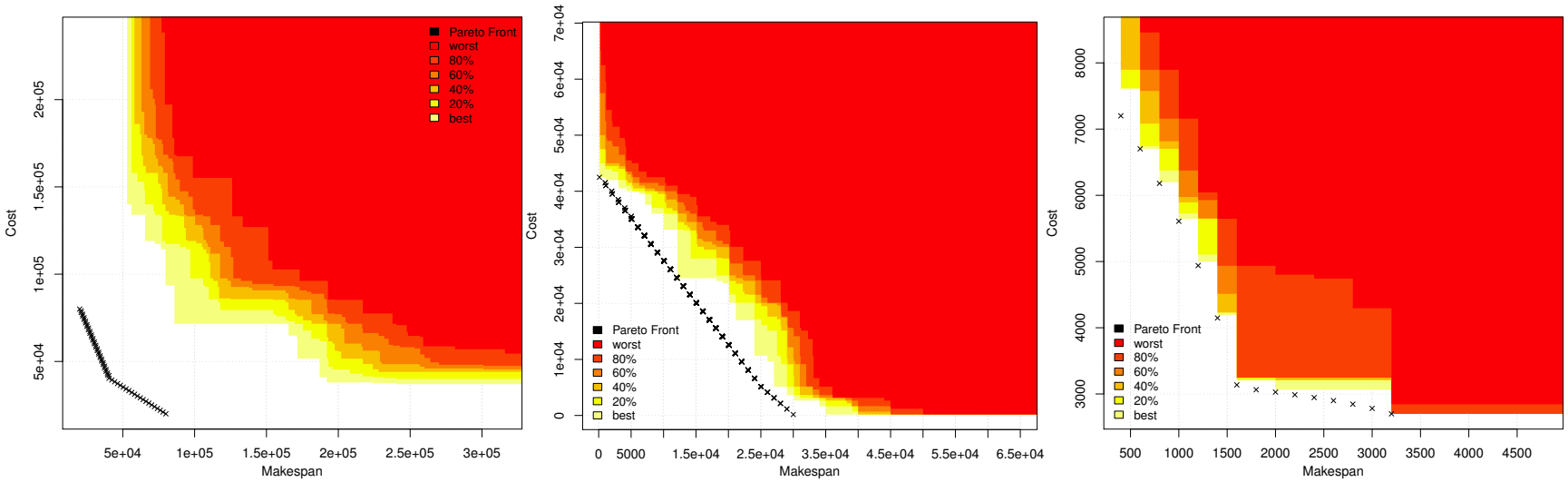


Résultats empiriques sur des instances larges



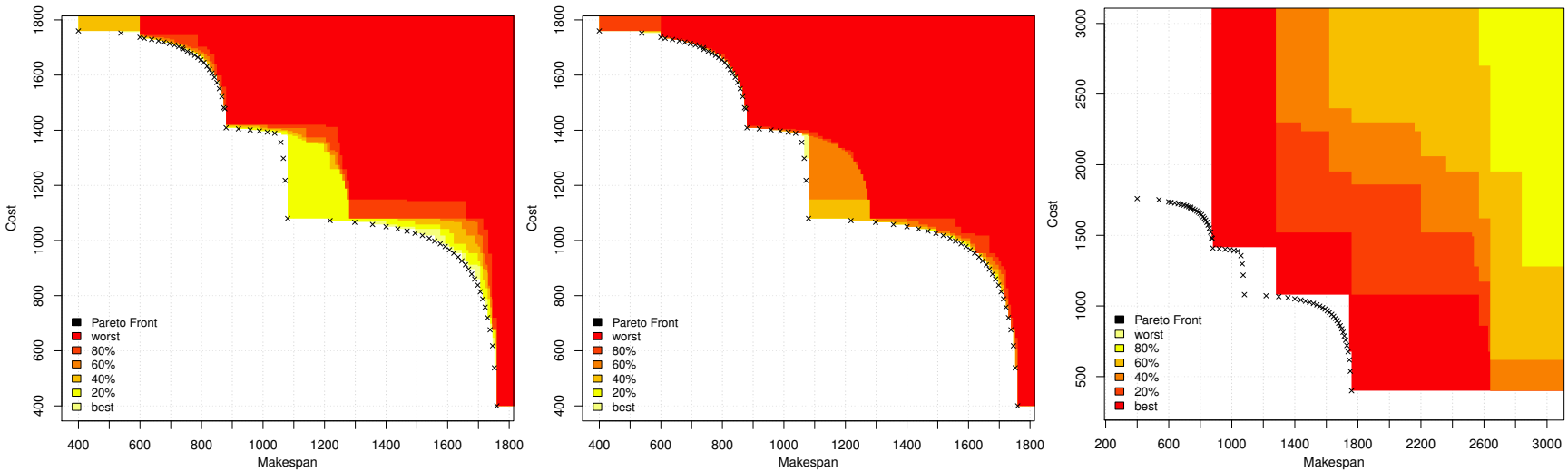


Résultats empiriques sur des instances larges

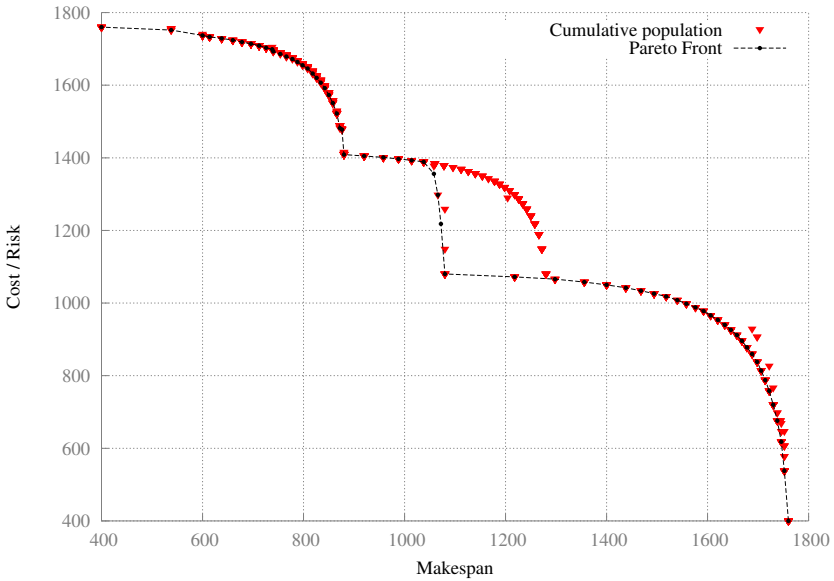


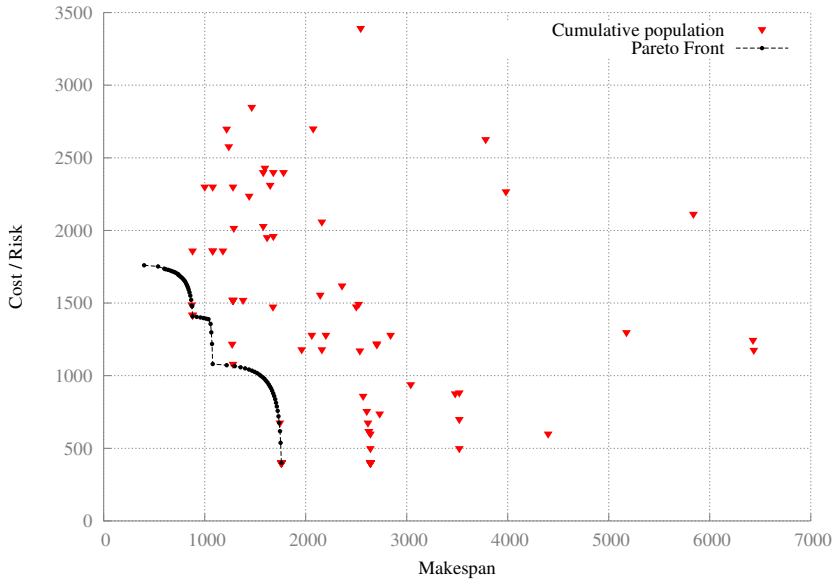
Aggrégation vs. Pareto

Dix fois plus de temps pour l'aggrégation que pour Pareto !



Aggrégation vs. Pareto





Quelques autres résultats

Résultats divers

- Passage à l'échelle des résultats précédents (algorithme génétique, effet de la stratégie, difficulté sur les fronts concaves, difficulté avec un nombre de passagers grandissants).
- Étude statistique approfondie des fronts concaves.
- Étude statistique de l'influence d'autres facteurs (nombre de villes, de passagers, nombre de buts dans la décomposition).

Modélisation

Divide-and-Evolve

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

Processus stochastique $(X_t)_t$:

- Sur un espace probabilisé (Ω, F, P).
- À valeur dans un espace d'état E.
- Conditionné par U un temps d'arrêt.
- Contrôlé par un ensemble de paramètres $p \in \mathcal{P}$.

Modélisation

Objectif

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

Trouver $\Phi : [0, T] \to \mathcal{P}$ tel que
$\lambda(X_t(\Phi_t, \omega) \succ \lambda(X_t(\Phi'_t, \omega)),\ \forall \omega, \forall \Phi'$.
Avec λ un indicateur de qualité.
Dans notre cas, $\lambda = H^-$, indicateur d'hypervolume.

Modélisation

Objectif

0n ne s'intéresse qu'à l'objectif à passer au solveur local.

- l objectfifs : $\mathcal{P} = \{o_i\}_{1 \leq i \leq l}$.
- Tirage des objectifs selon une distribution de probabilité P.
- Pas de modèle analytique : estimation de la distribution optimale P^*.

$m \neq l$

m nombre d'objectifs du problème, l objectifs du solveur local.

Modélisation

Hypothèses

- Plusieurs objectifs doivent être utilisés.
- $\frac{dP_t^*}{dt} \sim 0$.
- Les fréquences de *sauts* très supérieures à dt.

$\Longrightarrow$ distribution stationnaire entre deux *sauts*.

Objectifs

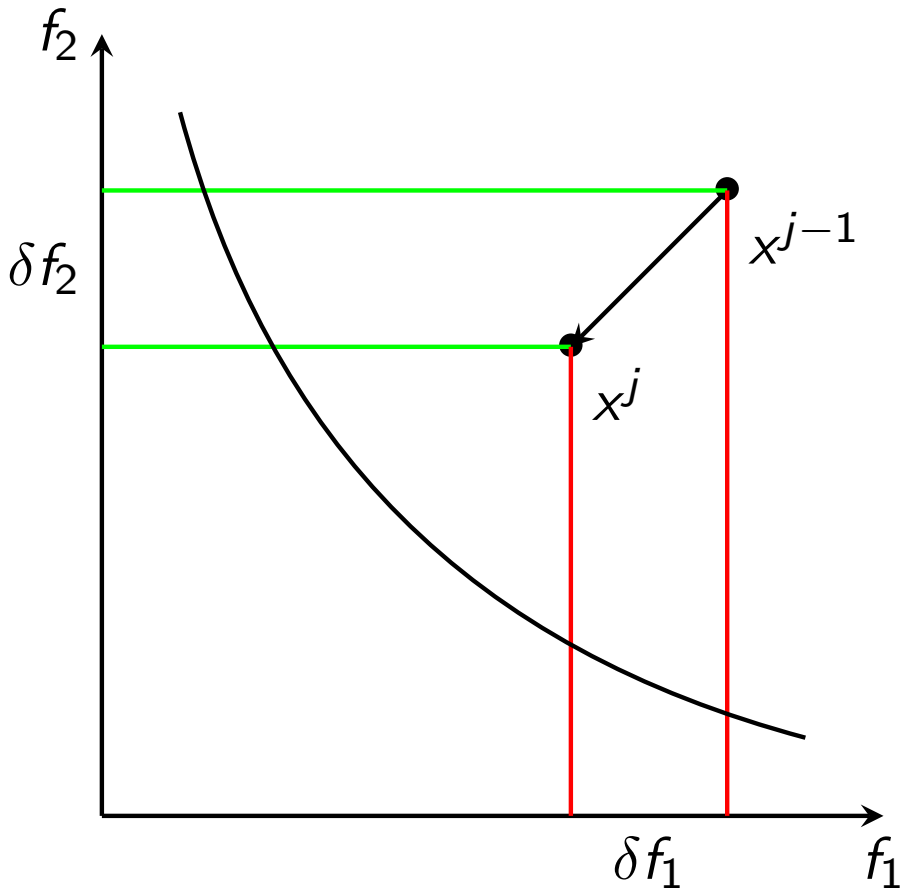
- Estimer (rapidement) la distribution P^*.
- Traquer les changements élémentaires de P^*.
- Détecter les sauts de P^* pour relancer l'estimation.

Stratégie adaptative ϕ

Etapes

- Stratégie de collecte d'information.
- Stratégie d'estimation d'effet des choix.
- Stratégie d'évaluation de la qualité des paramètres.
- Stratégie de sélection des paramètres

Stratégie de collecte d'information



Stratégie de collecte d'information

Évolution relative d'une solution

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

Indicateurs

- $\lambda^j_{\Delta^+}(x) = \sum_i \Delta f_i^j(x)$
- $\lambda^j_{\Delta^\times}(x) = \prod_i \Delta f_i^j(x)$
- $\lambda^j_{\Delta^e}(x) = e^{\sum_i \Delta f_i^j(x)}$
- Au niveau de la population, différence d'hypervolume.

Stratégie d'estimation d'effet des choix

Peu de stratégies adaptatives utilisent de l'estimation.

Marquage

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

Efficacité

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

Stratégie d'évaluation de la qualité des paramètres

On définit une fenêtre temporelle k.

Average Quality Attribution

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

Extreme Quality Attribution

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

Stratégie de sélection des paramètres

Probability Matching

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

Adaptive Pursuit

$$o^* = \underset{j-1-h \leq k \leq j-1}{argmax} (q^k(o_i) p^k(e^+|o_i))$$

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

$$p_{\max} = 1 - (I - 1) p_{\min}$$

Détection des sauts

Test cumulatif de Page-Hinkley

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

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

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

Critère : $m^j - M^j > \alpha$

Validation statistique

Validation statistique

- ANOVA sur les différentes modalités de chaque facteur.
- Surface d'Atteinte, Hypervolume, test de Wilcoxon...

Les résultats ne sont pas encore disponibles !

- Validation sur un large panel d'instances.
- Optimisation pendant 2 à 4 jours des paramètres statiques avec ParamILS.
- *Run* entre 300 et 5400 secondes (20 par instances) !

Bilan

Les points positifs

- Publication pour la conférence PPSN 2014.
- Apports intéressants pour l'optimisation multiobjectifs.
- Génie logiciel et refactoring : Descarwin bien plus performant.
- Expérience enrichissante (thématiques, recherche, distance).

Les points négatifs

- Conditions parfois difficiles du fait de la distance.
- Manque de temps pour concrétiser des idées sur les stratégies.
- Code pas encore parfait !

Questions

Merci pour votre attention ! :-)