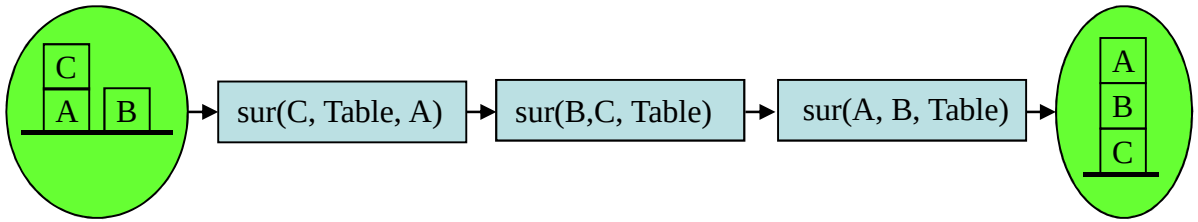
# Planification génétique avec chromosomes de longueur variable

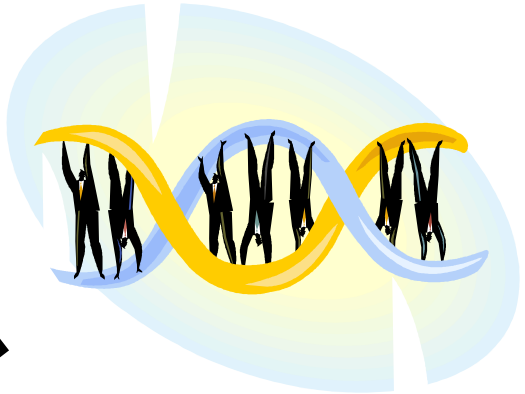
Alexandru HORIA BRIE
*Ecole Polytechnique*
*Palaiseau*

Philippe MORIGNOT
*AXLOG Ingénierie*
*Arcueil*

# 1. Modèle

# Encodage



**Chromosome**
**(*plan*)**


**Gène**
**(*action*)**

# PDDL

```
(define (domain prodigy-bw)
 (:action pick-up
        :parameters (?ob1)
        :precondition (and (clear ?ob1) (on-table ?ob1) (arm-
      empty))
        :effect
        (and (not (on-table ?ob1))
                (not (clear ?ob1))
                (not (arm-empty))
                (holding ?ob1)))
 (:action put-down
        :parameters (?ob)
        :precondition (holding ?ob)
        :effect
        (and (not (holding ?ob))
                (clear ?ob)
                (arm-empty)
                (on-table ?ob)))
 (:action stack
        :parameters (?sob ?sunderob)
        :precondition (and (holding ?sob) (clear ?sunderob))
        :effect
        (and (not (holding ?sob))
                (not (clear ?sunderob))
                (clear ?sob)
                (arm-empty)
                (on ?sob ?sunderob)))
```

```
(:action unstack
        :parameters (?sob ?sunderob)
        :precondition (and (on ?sob ?sunderob) (clear ?sob) (arm-
      empty))
        :effect
        (and (holding ?sob)
                (clear ?sunderob)
                (not (clear ?sob))
                (not (arm-empty))
                (not (on ?sob ?sunderob)))))
```

```
(define (problem bw-large-b)
 (:domain prodigy-bw)
 ;(:length (:parallel 18) (:serial 18))
 (:objects 1 2 3 4 5 6 7 8 9 10 11)
 (:init (arm-empty)
       (on 3 2) (on 2 1) (on-table 1) (on 11 10) (on 10 5) (on 5 4)
       (on-table 4) (on 9 8) (on 8 7) (on 7 6) (on-table 6) (clear
      3)
       (clear 11) (clear 9))
 (:goal (and
       (on 1 5) (on 5 10) (on-table 10) (on 8 9) (on 9 4) (on-
      table 4)
       (on 2 3) (on 3 11) (on 11 7) (on 7 6) (on-table 6) (clear 1)
       (clear 8) (clear 2)
       )))
```

# Formalisme

- $C = (a_i, (p_{i,j}, o_j)_{j \in Param(i)})_{i \in [1,N]}$

avec $\begin{cases} a_i & : \text{indice de la } i\text{-ème action} \\ p_{i,j} & : \text{indice du } j\text{-ème paramètre dans la } i\text{-ème action} \\ o_j & : \text{indice du } j\text{-ème objet dans la liste de paramètre} \end{cases}$

- Typage de $o_j$
- Exemple (anomalie de Sussman) :

$$C = ((1, ((1, 3), (2, 1), (3, 4))),$$
$$(1, ((1, 2), (2, 4), (3, 3))),$$
$$(1, ((1, 1), (2, 4), (3, 2)))$$

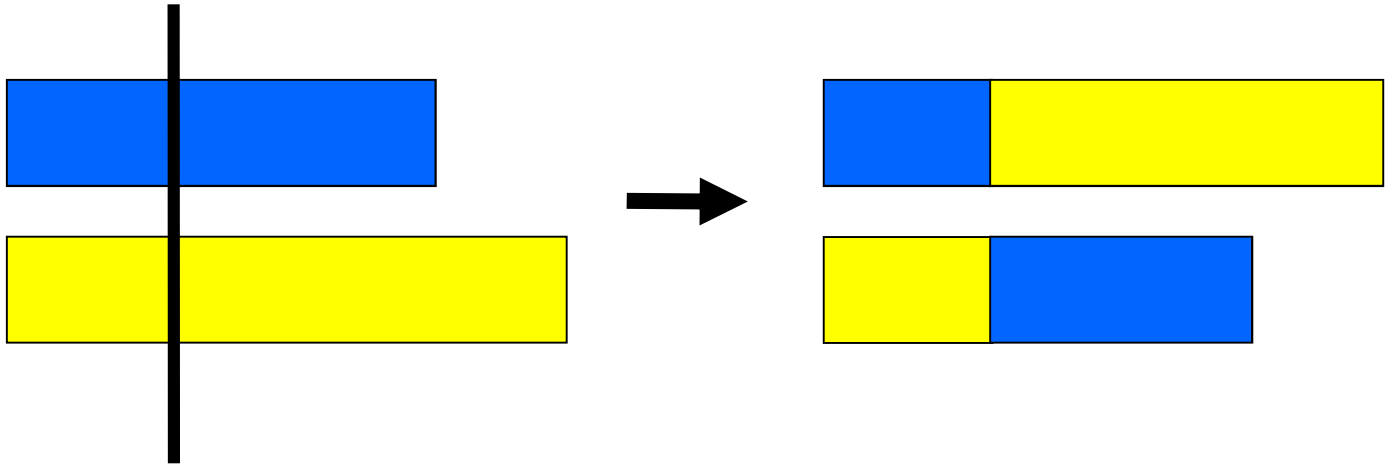
# 2. Algorithme

# Fonction d'adaptation

- Combinaison linéaire des termes :
  - Nombre de pré-conditions non satisfaites
  - Nombre d'actions non exécutables
  - Longueur(C) – position du 1$^{er}$ conflit
  - Longueur de la plus longue sous-séquence correcte
  - Nombre de post-conditions satisfaisant les buts
- C = solution => f_adaptation = 0 (> 0 sinon).
- Simulation de l'exécution du chromosome, en ignorant les actions conflictuelles ou les exécutant quand même.

# Opérateur de croisement

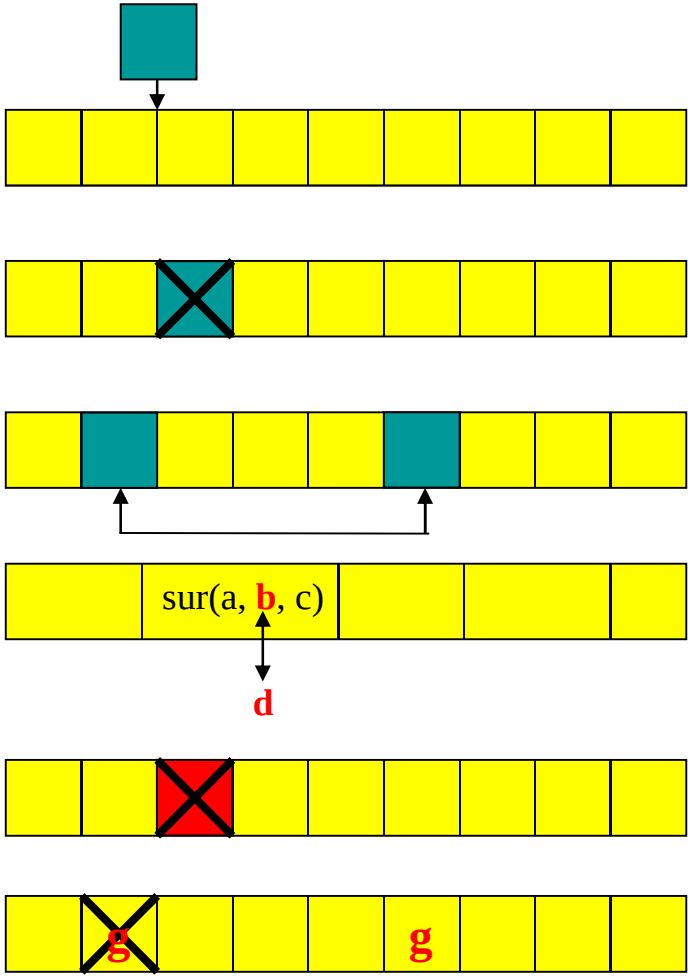
- 1-point uniforme



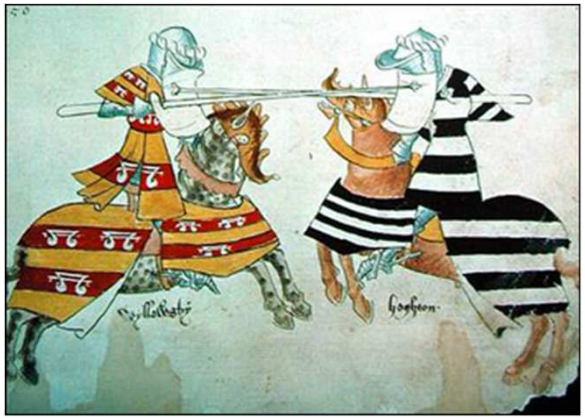
- ~~1-point non-uniforme~~, ~~2-points (non-)uniforme.~~

# Opérateurs de mutation

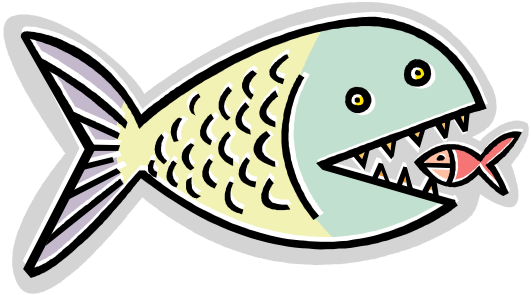
- Ajout aléatoire de gène :
- Retrait aléatoire de gène :
- Permutation aléatoire de deux gènes :
- Remplacement d'un gène :
- Mutations heuristiques :
  - Retrait d'un gène conflictuel :
  - Retrait d'un gène dupliqué :



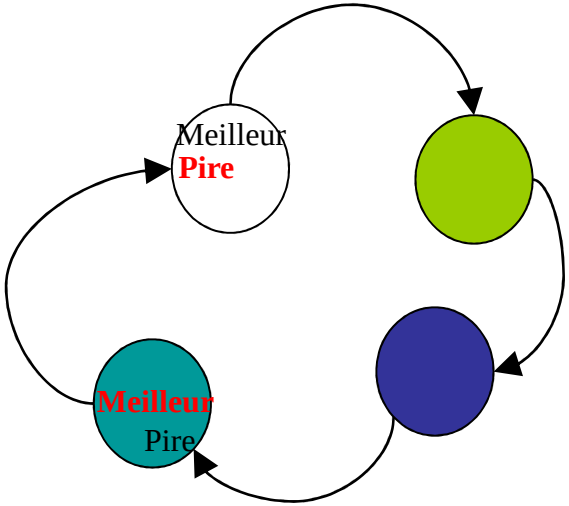
# Imports génétiques



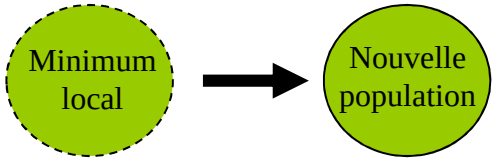
**Sélection par tournoi**



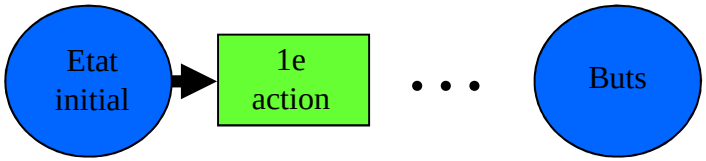
**Croisement / mutation élitiste**




**Populations multiples**




**Ré-initialisation de la population**




**Mimétisme faible**

# Algorithme

1. Parser le domaine et le problème, parser le fichier de configuration ; lier les objets aux types.
2. Initialiser les populations avec des chromosomes aléatoires et de longueur aléatoire (en utilisant le mimétisme faible).
3. TANT QUE (une solution n'est pas trouvée OU un temps limite n'est pas dépassé) :
   A. Sélectionner 1 ou 2 chromosome(s) par sélection par tournoi.
   B. Appliquer le croisement et/ou une mutation.
   C. Calculer la valeur d'adaptation des chromosomes-enfants.
   D. Appliquer la sélection élitiste, si demandé.
   E. Ajouter le(s) résultat(s) à la génération suivante de cette population.
   F. REPETER depuis A JUSQU'À-CE-QUE taille(PopulationSuivante) = constante.
4. Décoder la solution (ou la meilleure solution jusqu'à présent)

# 3. Résultats expérimentaux

# Implémentation

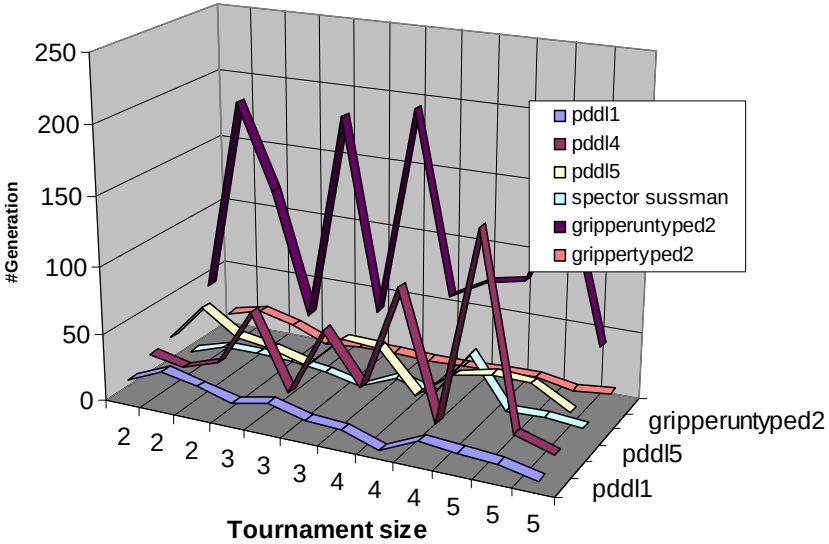
- Domaines PDDL 3.0 :
  - Problèmes dans le monde des blocs non typés de Lee Spector ou des domaines STRIPS.
  - Problèmes « *gripper* » typés ou non.
- *0.5* à *2.0* secondes par génération (temps réel)
  - Sur un ordinateur moyennement chargé (400 MHz, Pentium).
  - Avec une implémentation en C++ (Bison & Flex pour le parser PDDL).
  - Des milliers de chromosomes par population ; une trentaine de gènes par chromosome en moyenne; 5 paramètres changés en moyenne.
- 31 paramètres interdépendants : poids, seuils, probabilités.

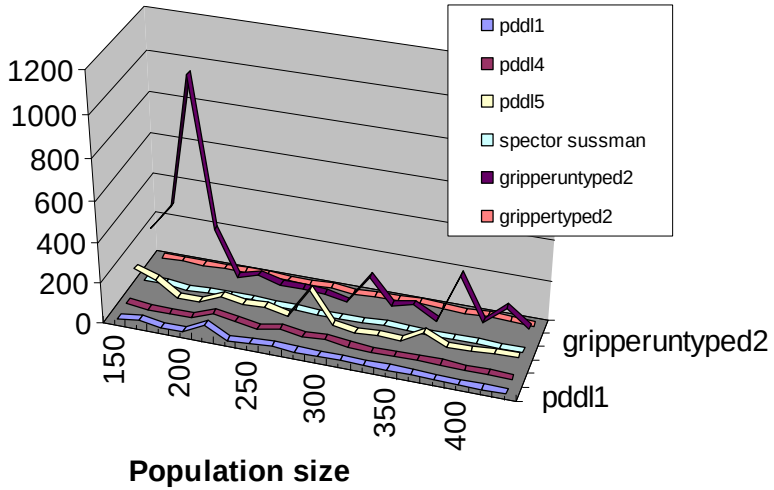
# Paramètres

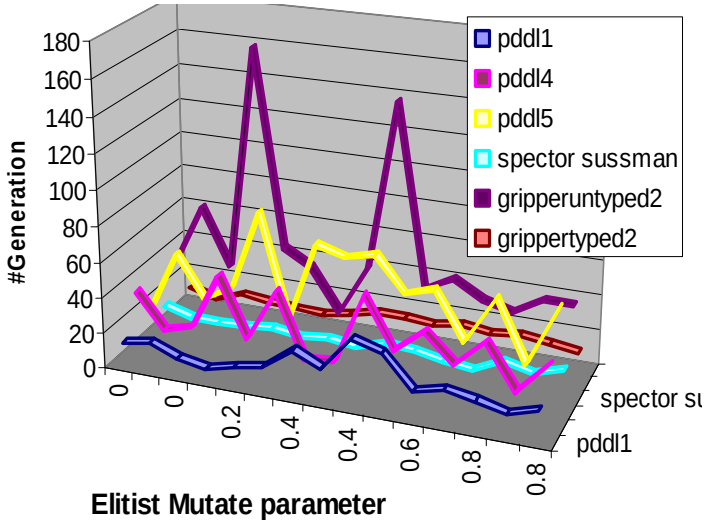
isolation_time 7 // #generation before migration among islands
rand_gene_break_random 0.8 // probability that the mutation / crossover point is random, instead of heuristically chosen
crossover_prob 0.8 // crossover probability
mutate_act_replace_prob 0.07 // probability of replace mutation
mutate_param_replace_prob 0.06 // prob. of parameter mutation
mutate_swap_prob 0.05 // probability of swap mutation
mutate_add_prob 0.07 // probability of growth mutation
mutate_erase_prob 0.06 // probability of shrink mutation
mutate_elim_duplicate 0.03 // probability of duplicate removal
mutate_maturation 0.001 // prob. of conflicting genes removal
probability_of_consecutive_mutations 0.2 // probability of mutating several consecutive genes
elitist_mutate 0.6 // probability of using elitism for mutation
elitist_cross 0.7 // probability of using elitism for crossover
cross_and_mutate 0.5 // probability of mutating the offspring of a crossover

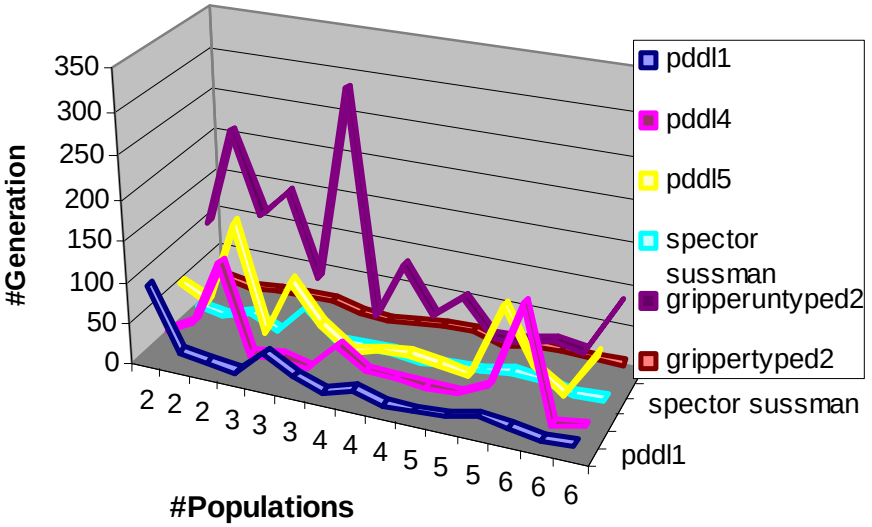
// Eight weights for components of the fitness function
conflict_pound 0.05 // For the conflicts
force_update_even_if_collision 1 // flag to ignore the bad actions in the plan's fitness
conflict_pound 0.05 // For the conflicts
conflicting_actions_pound 0.01 // For the conflicting actions
purpose_distance_pound 0.3 // For the unsolved goals
repeating_actions_pound 0.1 // For duplicated actions
conflict_position_pound 0.02 // For the first conflict pos.
longestsequencepond -0.01 // For the longest correct subplan
number_of_actions_pound -0.001 // For the chromosome size
upper_nactions_limit 10 // threashold below which number_of_actions_pound is valid
time_before_shaking_population 15 // #generation with elite stagnation before reset
shaking_population_total 130 // max value for picking odds relatives
shaking_population_random_chroms 127 // relative odds for adding a randomly seeded chromosome
big_mutate_number_of_mutations 5 // #mutation to be applied on the chosen chromosome
shaking_population_best_sol_mutations 1 // relative odds of mutating a previous chromosome
shaking_population_elite_mutations 1 // relative odds of mutating the population's elite
population_shake 1 // flag for population reset

# Analyse des paramètres



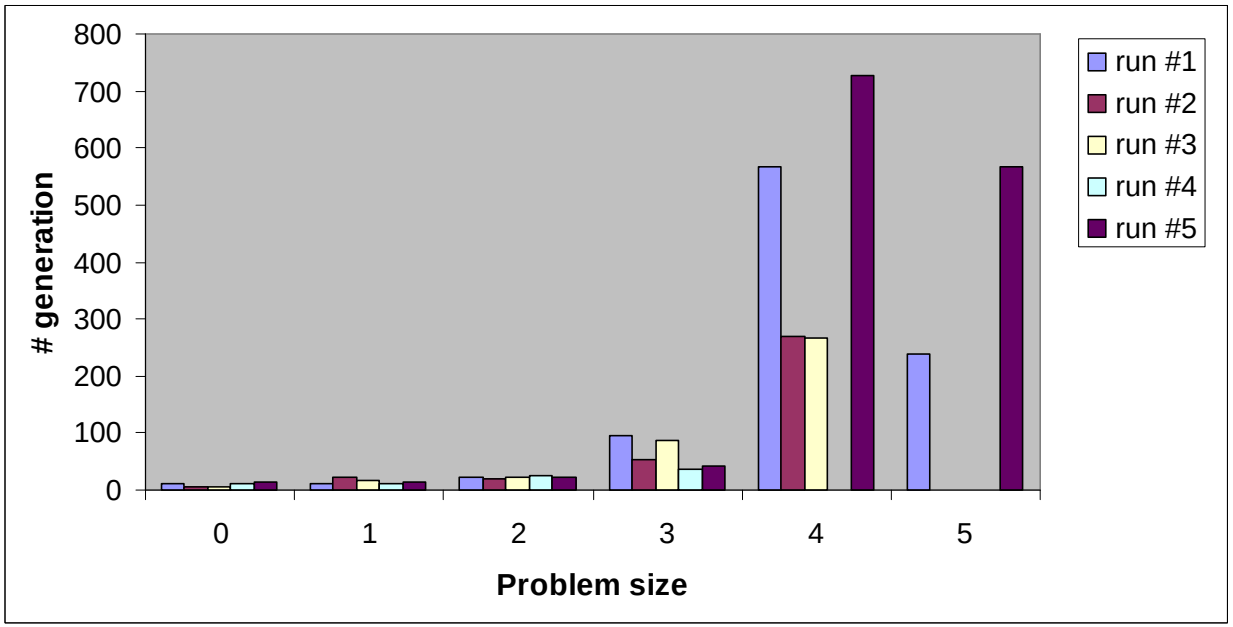



# Passage à l'échelle

- Domaine « gripper » :



# 4. Conclusion & Travaux futurs

# Conclusion

- Encodage intuitif :
  - 1 gène = 1 action instanciée ;
  - 1 chromosome = 1 plan linéaire.
- Chromosomes de longueur variable.
- Croisement 1-point uniforme, mutations (ajout / retrait, permutation, remplacement, 2 heuristiques).
- Fonction d'adaptation = combinaison linéaire de caractéristiques d'interprétation du plan.
- Sous-ensemble STRIPS typé de PDDL version 4.1.
- Analyse des paramètres :
  - Environ 300 individus par population, 3 à 4 populations, 3 à 4 individus par tournoi.
- Langage de description de plan plus étendu que celui de Ion Muslea, Lee Spector, et Henrik Westerberg.
- Passage à l'échelle encore à améliorer …

# Travaux futurs

- Validité de la métaphore ? (chromosomes humains longs, mais de longueur fixe).
- Plus d'expériences (analyse multicritères ?) sur d'autres domaines de PDDL.
- Encoder le parallélisme des actions dans un chromosome.
- Opérateurs stochastiques en PDDL.
- Un planificateur non génétique comme un opérateur dans un algorithme génétique.
  - Meilleur passage à l'échelle ?
- Autres imports : *Fast Messy GA*, *Linkage Learning GA*, ...