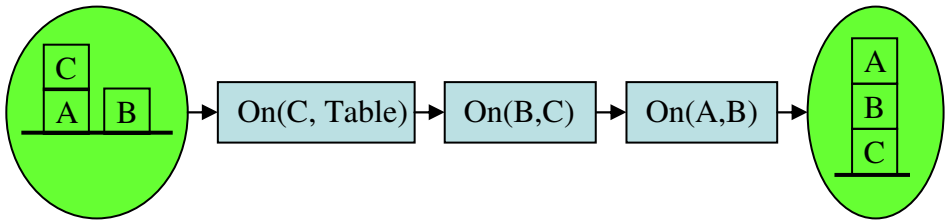
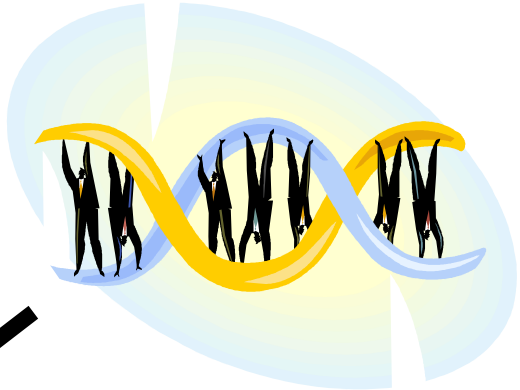
Genetic Planning using Variable Length Chromosomes

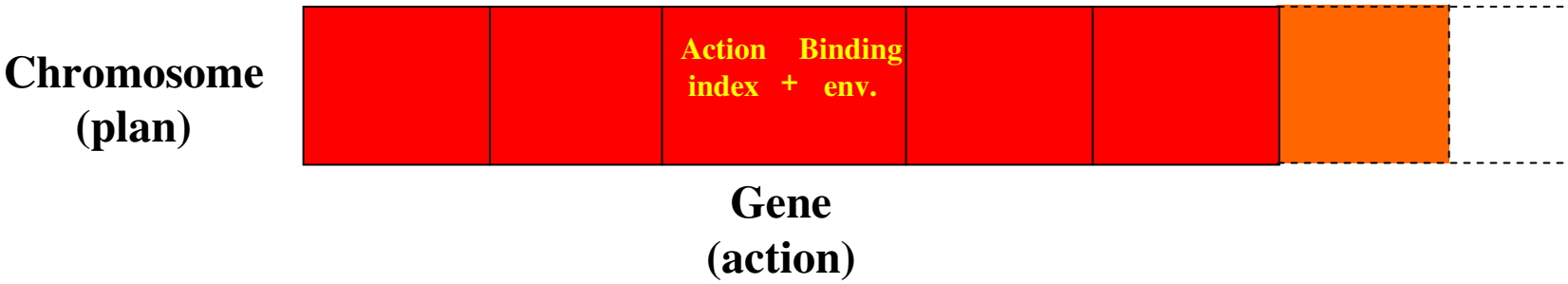
Alexandru HORIA BRIE
Ecole Polytechnique
Palaiseau, France

Philippe MORIGNOT
AXLOG Ingéniérie
Arcueil, France

Encoding



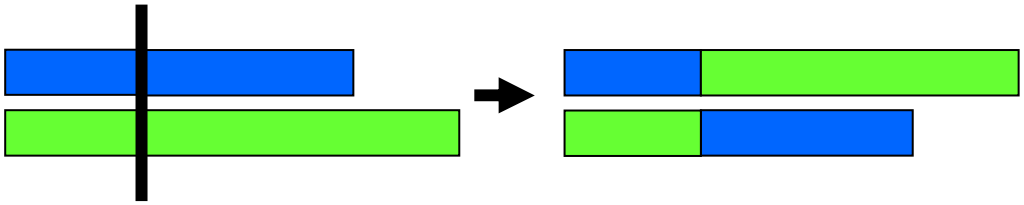




$$C = (a_i, (p_{i,j}, o_j)_{j \in Param(i)})_{i \in [1,N]} \quad \text{where} \begin{cases} a_i & : \text{index of } i\text{ - th action} \\ p_{i,j} & : \text{index of } j\text{ - th parameter in } i\text{ - th action} \\ o_j & : \text{index of } j\text{ - th object in the parameter list} \end{cases}$$

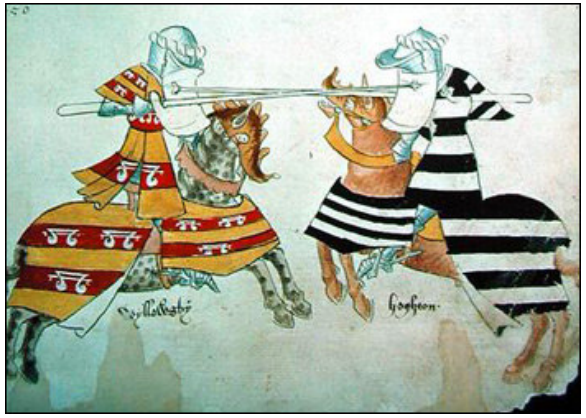
Components

- Fitness function :
  - Number of conflicts
  - Number of unexecutable actions
  - Position of the first conflict
  - Chromosome size, size of the longest correct sub-sequence
  - #collisions wrt. goals
- Operators :
  - Crossover : one-point uniform.

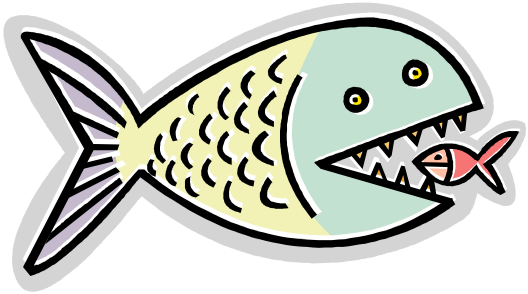


  - Mutation :
    - Growth / shrink
    - Swap
    - Gene replace, action parameter replace
    - Heuristic : removal of conflicting genes, of duplicated genes, etc.
- A chromosome is a variable-length linked list of genes; A gene is a limited-length vector of parameters.
  - The parameter typing comes from a binding environment (« World Model »).

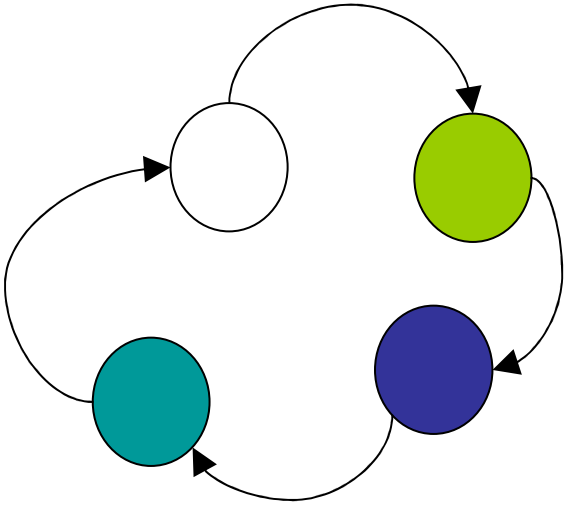
Genetic Techniques



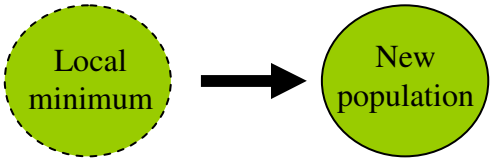
Tournament selection



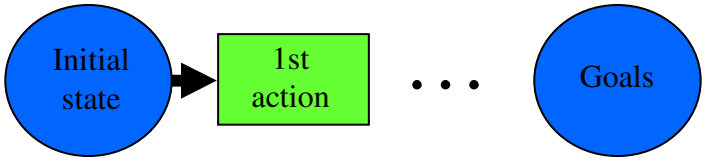
Elite crossover / mutation



Multi-populations



Population reset



Weak memetism (in seeding)

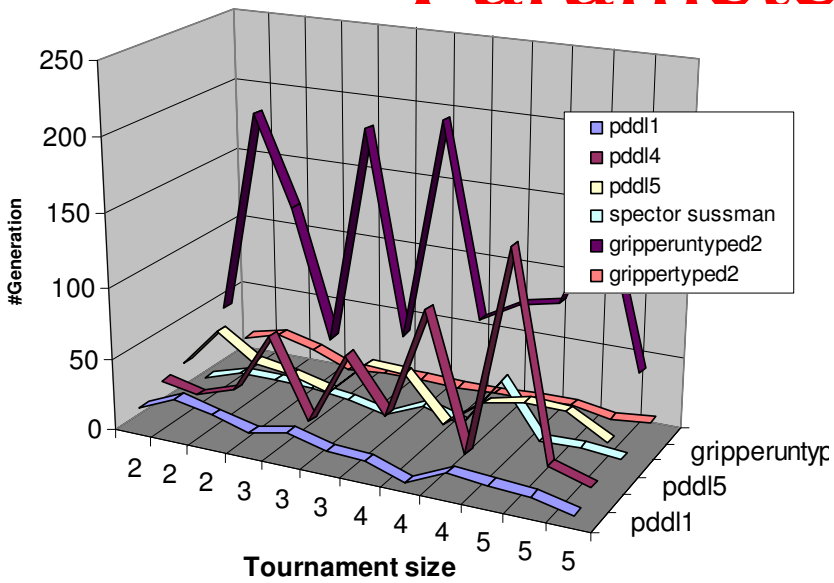
Algorithm

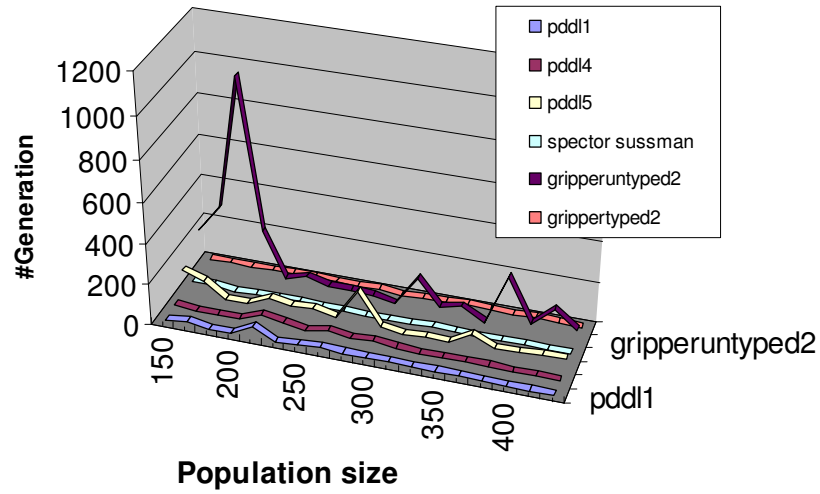
1. Parse the domain and problem, parse the configuration file; bind objects to types and record the typing of possible actions.
2. Initialize the populations with random chromosomes of random lengths (using weak memetism in seeding).
3. WHILE (solution not found OR timeOut not exceeded)
   A. Select 1 or 2 chromosome(s) using tournament selection.
   B. Apply crossover and/or mutation.
   C. Compute the offspring's fitness value.
   D. Apply elitism selection if required.
   E. Add the result(s) to the next generation of this population.
   F. REPEAT from A UNTIL size(NextPopulation) = constant.
4. Decode the solution (or the best solution found so far)

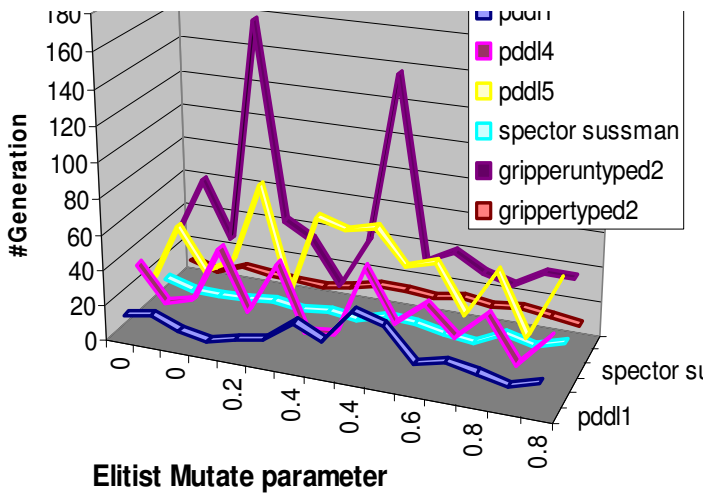
Implementation & Method

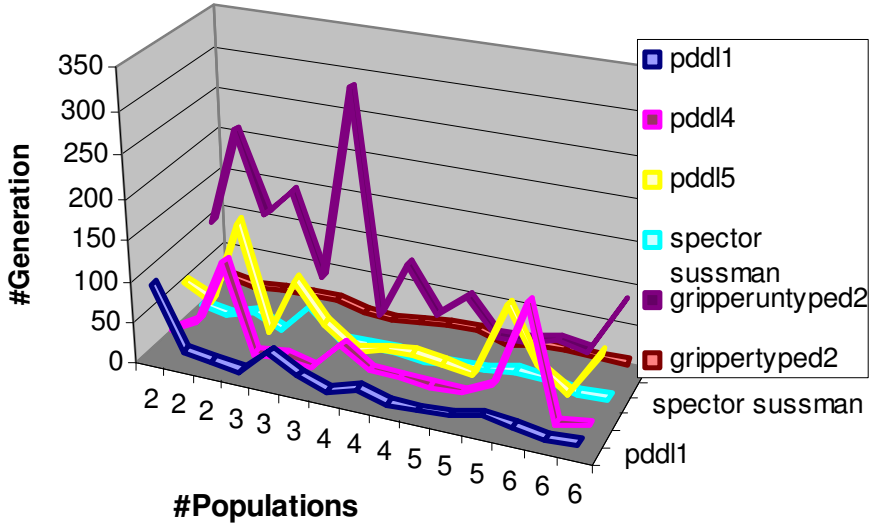
- PDDL domains :
  - Untyped blocks world problems in Spector's or STRIPS domain.
  - (Un-) typed gripper problems.
- *0.5s* to *2.0s* in real time per generation
  - On an average loaded computer (400 MHz, Pentium).
  - With a C++ implementation (Bison & Flex for the PDDL parser).
  - 1000s of chromosomes; 30 genes on average; 5 parameters on average.
- *Several* runs of the *same* example (*average* results).
- Many *inter-dependent* parameters (e.g., weights, thresholds, probabilities).
  - Searching for the most relevant parameters and their « optimal » value, considering the other ones as set to their default value.
- Pure performances (*convergence* speed).

Parameters Analysis

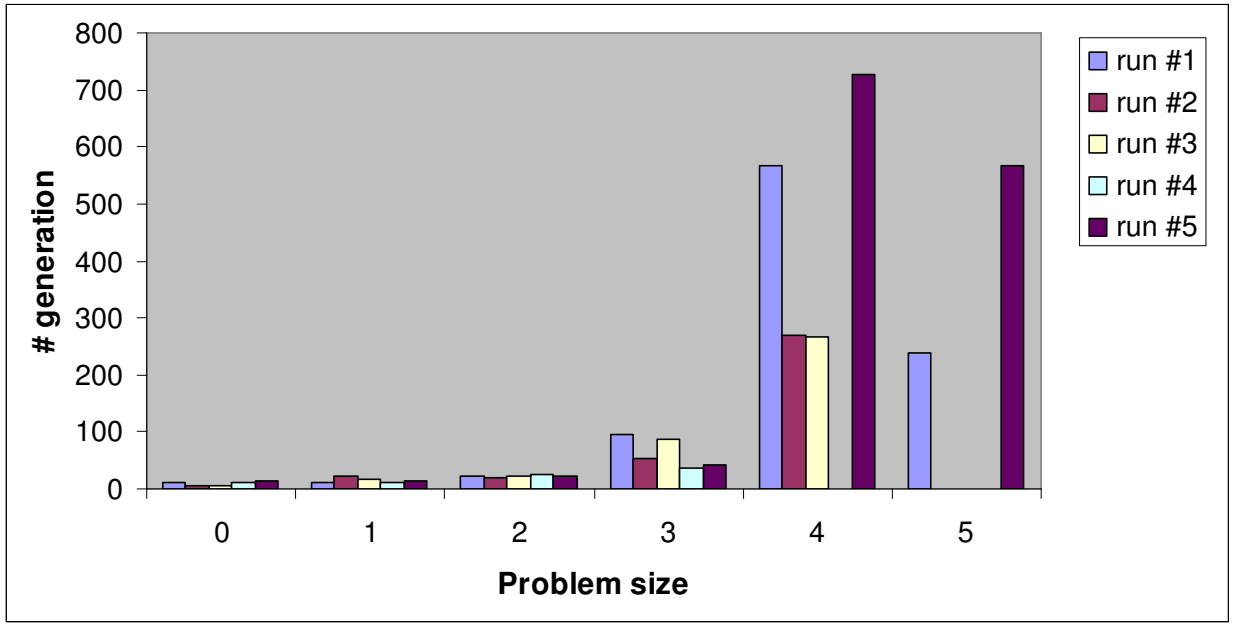








Scaling up


800
700
600
500
400
300
200
100
0
generation
0
1
2
3
4
5
Problem size
run #1
run #2
run #3
run #4
run #5

Conclusion

- Intuitive encoding, complex fitness function, built-in and extensible typing system.
  - Slightly more powerful than the predecessors' planning description language.
- Parameter analysis :
  - Approx. 300 individuals per population, 3 to 4 populations, 3 to 4 individuals in tournament.
- Work to make the model scale up.
- Research directions :
  - Hybridation A.I. planner / genetic planner.
  - Stochastic operators for PDDL.
  - *Fast Messy GA, Linkage Learning GA, ...*