Introduction of Statistics in Optimization

Fabien Teytaud

Joint work with
Tristan Cazenave, Marc Schoenauer, Olivier Teytaud

Université Paris-Dauphine - HEC Paris / CNRS
Université Paris-Sud Orsay - INRIA

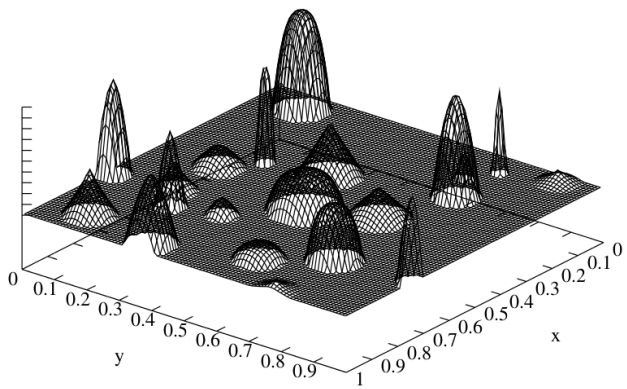
INRA Toulouse, 10 Février 2012

Outline

Outline

Problem

- Data
  - Search Space Ω (set of possible solutions)
  - Objective function (quality criterion)
- Goal
  - Find the best solution (according to the objective function)
- Formally
  - Consider $\mathcal{F} : \Omega \to \mathbb{R}$
  - $x^* \in \Omega / x^* = \text{argmax}(\mathcal{F})$
- Additional properties
  - Black-box optimization
  - Continuous ($\Omega \subset \mathbb{R}^{\mathbb{N}}$)



$\Longrightarrow$ Evolution Strategy

Outline

Evolution Strategy

[Rechenberg, 73][Schwefel, 74]

Initialize y,σ,C
while we have time **do**
  for $i \leftarrow 1 \dots \lambda$ **do**
    $x_i \leftarrow y + \sigma \mathcal{N}_i(0, C)$
    compute $\mathcal{F}(x_i)$
  end for
  update y,σ,C
end while
Return y

$\left\{ \begin{array}{l} y \in \mathbb{R}^N \text{ represents the current search point} \\ \lambda \text{ is the population size} \\ C \in \mathbb{R}^{N \times N} \text{ is the covariance matrix} \\ \sigma \in \mathbb{R}_+ \text{ is the step-size} \end{array} \right.$

Evolution Strategy

[Rechenberg, 73][Schwefel, 74]

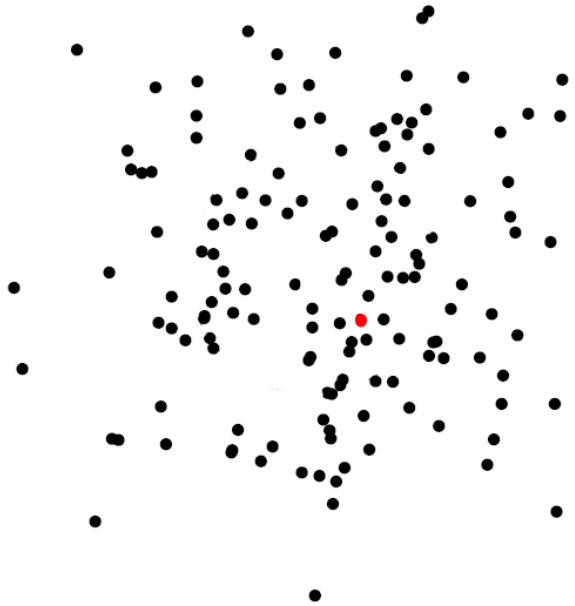
Initialize y,σ,C
while we have time **do**
  for $i \leftarrow 1 \ldots \lambda$ **do**
    $x_i \leftarrow y + \sigma \mathcal{N}_i(0, C)$
    compute $\mathcal{F}(x_i)$
  end for
  update y,σ,C
end while
Return y

$\left\{ \begin{array}{l} y \in \mathbb{R}^N \text{ represents the current search point} \\ \lambda \text{ is the population size} \\ C \in \mathbb{R}^{N \times N} \text{ is the covariance matrix} \\ \sigma \in \mathbb{R}_+ \text{ is the step-size} \end{array} \right.$

Evolution Strategy

[Rechenberg, 73][Schwefel, 74]

Initialize y,σ,C
while we have time **do**
  for $i \leftarrow 1 \dots \lambda$ **do**
    $x_i \leftarrow y + \sigma \mathcal{N}_i(0, C)$
    compute $\mathcal{F}(x_i)$
  end for
  update y,σ,C
end while
Return y

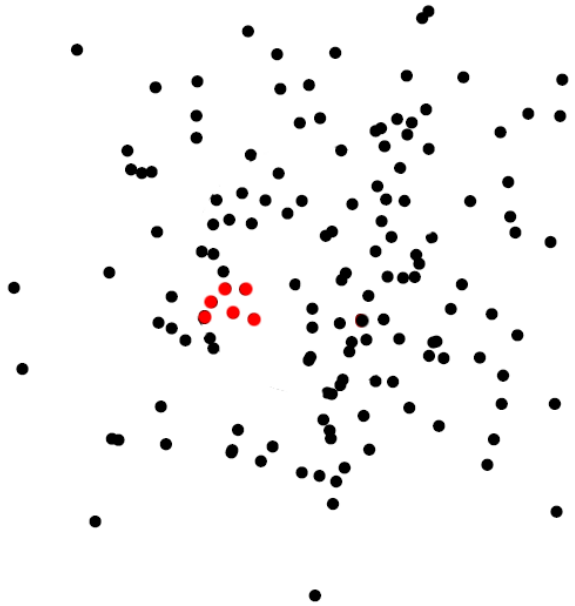


$$\begin{cases} y \in \mathbb{R}^N \text{ represents the current search point} \\ \lambda \text{ is the population size} \\ C \in \mathbb{R}^{N \times N} \text{ is the covariance matrix} \\ \sigma \in \mathbb{R}_+ \text{ is the step-size} \end{cases}$$

Evolution Strategy

[Rechenberg, 73][Schwefel, 74]

Initialize y,σ,C
while we have time **do**
  for $i \leftarrow 1 \dots \lambda$ **do**
    $x_i \leftarrow y + \sigma \mathcal{N}_i(0, C)$
    compute $\mathcal{F}(x_i)$
  end for
  update y,σ,C
end while
Return y

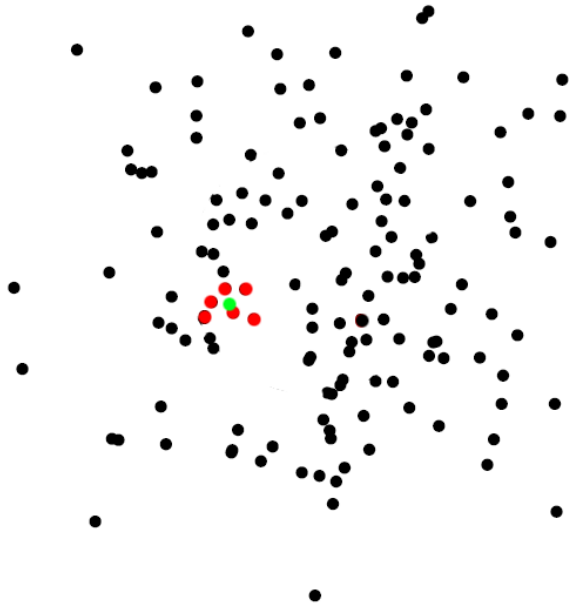


$$\begin{cases} y \in \mathbb{R}^N \text{ represents the current search point} \\ \lambda \text{ is the population size} \\ C \in \mathbb{R}^{N \times N} \text{ is the covariance matrix} \\ \sigma \in \mathbb{R}_+ \text{ is the step-size} \end{cases}$$

Evolution Strategy

[Rechenberg, 73][Schwefel, 74]

Initialize y,σ,C
while we have time **do**
  for $i \leftarrow 1 \dots \lambda$ **do**
    $x_i \leftarrow y + \sigma \mathcal{N}_i(0, C)$
    compute $\mathcal{F}(x_i)$
  end for
  update y,σ,C
end while
Return y



$$\begin{cases} y \in \mathbb{R}^N \text{ represents the current search point} \\ \lambda \text{ is the population size} \\ C \in \mathbb{R}^{N \times N} \text{ is the covariance matrix} \\ \sigma \in \mathbb{R}_+ \text{ is the step-size} \end{cases}$$

Evolution Strategy

Initialize y,σ,C
while we have time **do**
  for $i \leftarrow 1 \dots \lambda$ **do**
    $x_i \leftarrow y + \sigma \mathcal{N}_i(0, C)$
    compute $\mathcal{F}(x_i)$
  end for
  update y,σ,C
end while
Return y

Evolution Strategy

Initialize y,σ,C
while we have time **do**
  for $i \leftarrow 1 \dots \lambda$ **do**
    $x_i \leftarrow y + \sigma \mathcal{N}_i(0, C)$
    compute $\mathcal{F}(x_i)$
  end for
  update y,σ,C
end while
Return y

Updating y: $y \leftarrow \sum_{i \leftarrow 1}^{\mu} w_{(i)} x_{(i)}$

$(\mu/\mu, \lambda)$-ES

Fitness Function

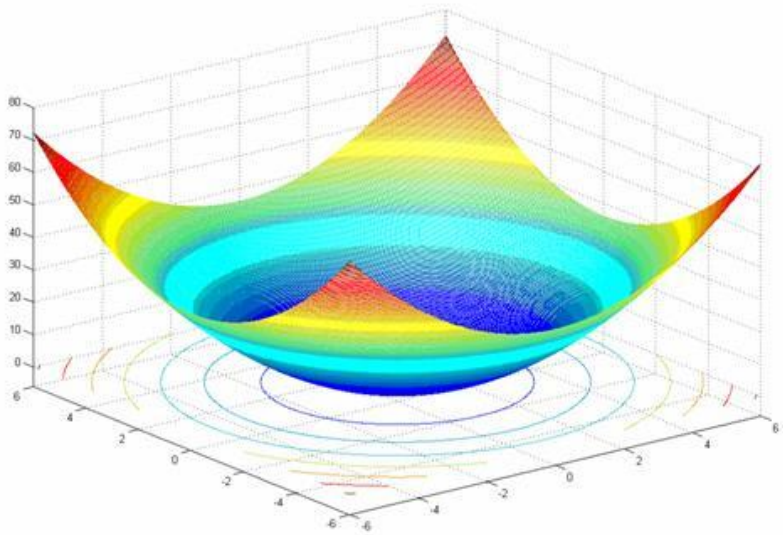
- Sphere function : $x \rightarrow ||x||$

- Minimization

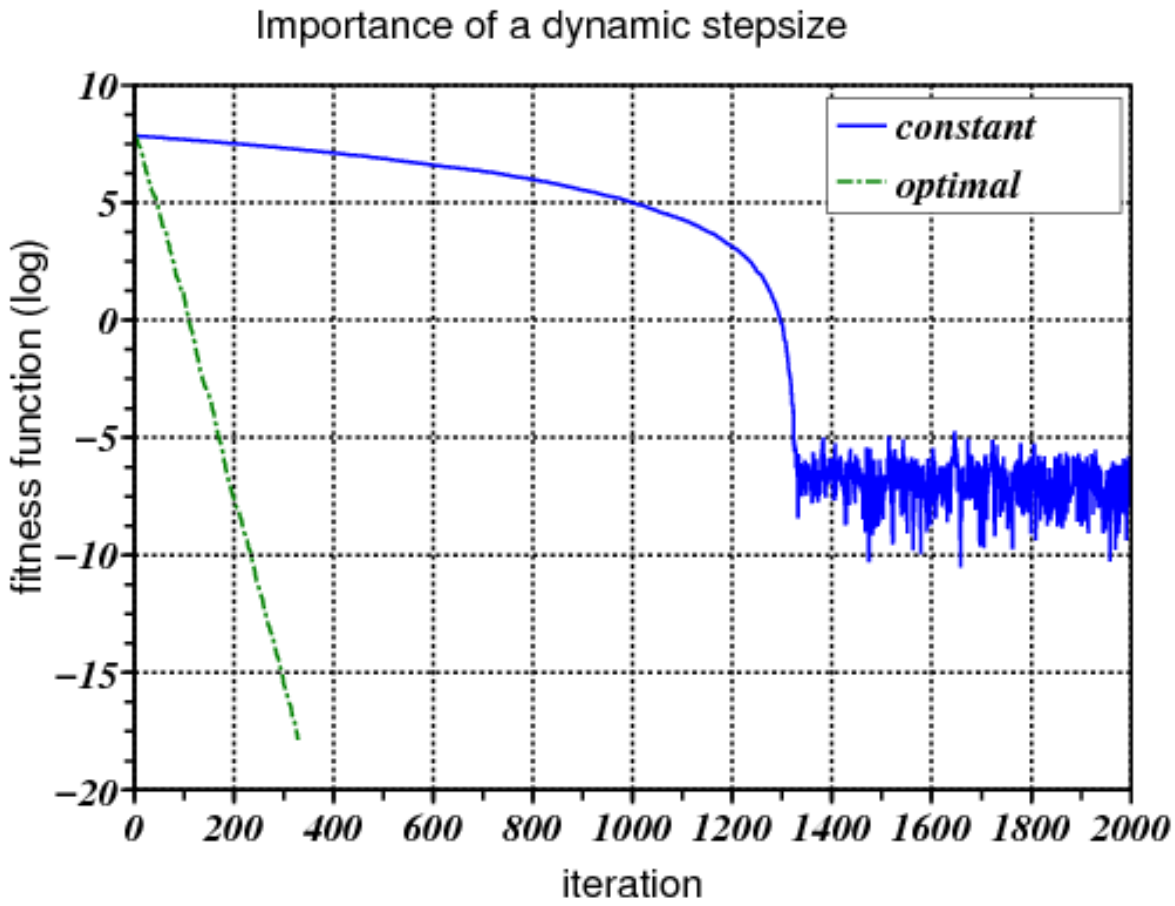
  In all experiments

  $\Longrightarrow$

  The lower the better



Why updating the step-size ?



Scale invariant : $\sigma \leftarrow \alpha ||y||$

Evolution Strategies

- Covariance matrix : Identity or diagonal
  - $\frac{1}{5^{th}}$ rule [Rechenberg, 73]
  - Self-Adaptation [Rechenberg, 73][Schwefel, 74]
  - Path Length Control (aka CSA) [Hansen & Ostermeier, 96, 01]
- Full Covariance matrix - self adaptation [Schwefel, 81]

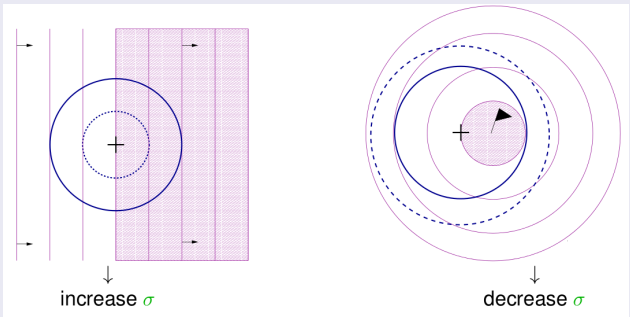
State of the art algorithms

- Covariance Matrix Adaptation (CMA) [Hansen & Ostermeier, 01]
- Covariance Matrix Self-Adaptation (CMSA) [Beyer & Sendhoff, 08]

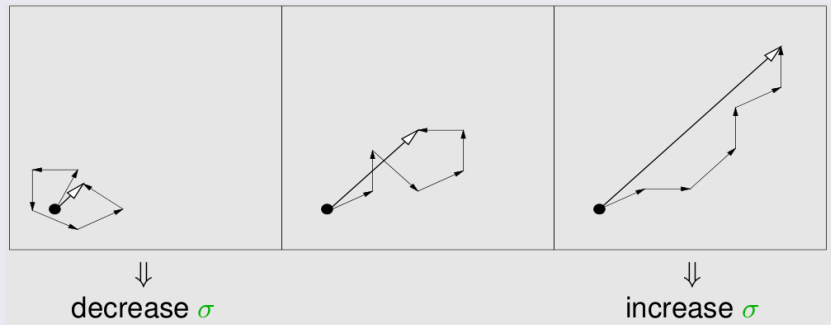
Evolution Strategies

AUGER & HANSEN, CMA-ES TUTORIAL

$1/5^{th}$ rule



CSA



Self-Adaptation

$\sigma_i \leftarrow \hat{\sigma} e^{\tau \mathcal{N}_i(0,1)}$

with $\tau \leftarrow 1/\sqrt{N}$ and N the dimension

Estimation of Multivariate Normal Algorithm (EMNA)

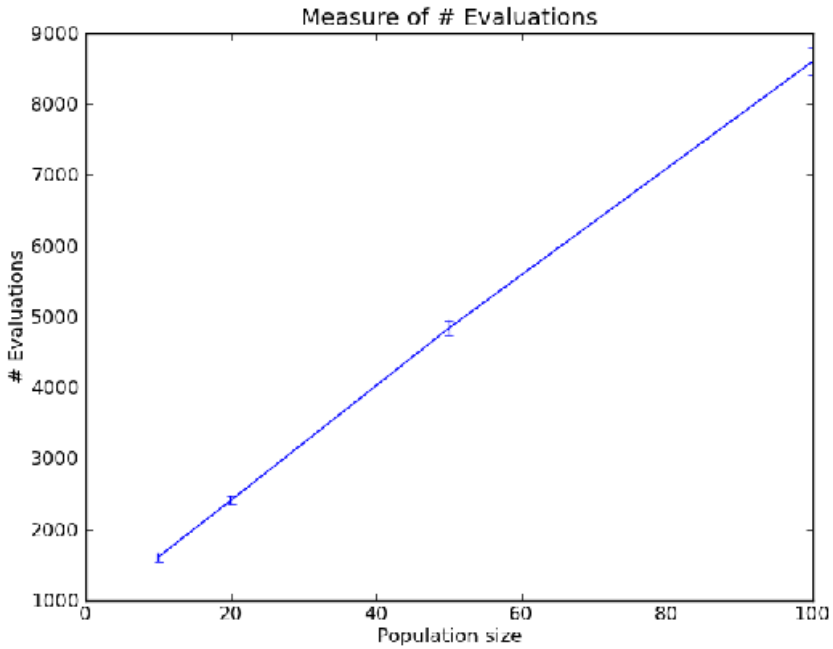
[Larranaga & Lozano, 01]

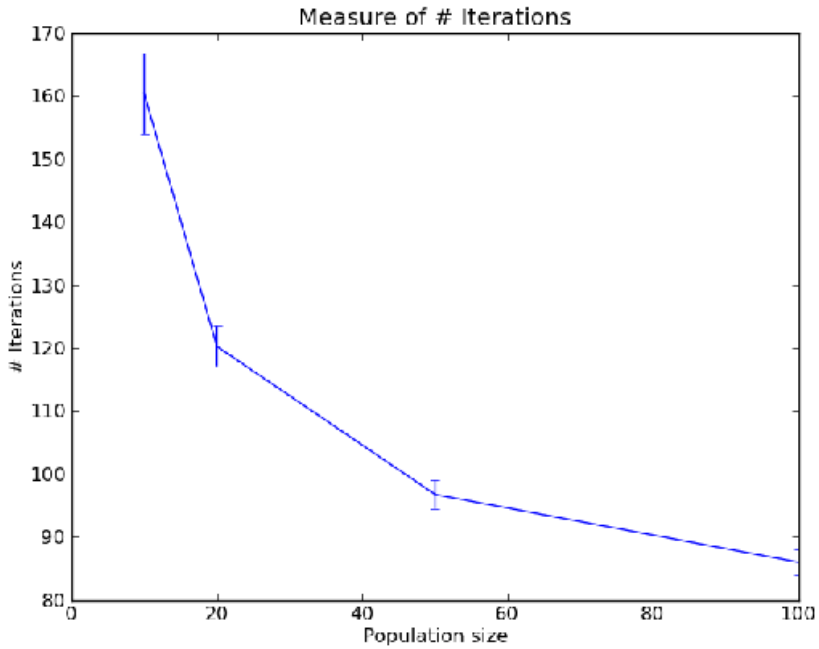
- EDA : evolution of a parameterized probability distribution
  - Sample the domain with the current distribution
  - Evaluate the population
  - Select the best points
  - Update the parameters of the distribution
- EMNA
  - Example of EDA
  - Gaussian distribution (σ, C)
  - Close to ES

Outline

Motivation for parallelism

CMA, Sphere function, $f_{target} = 10^{-10}$, $N = 10$





Theoretical bounds

[Teytaud & Fournier, 10]

Speed-up for p processors

$S_p \leftarrow \frac{T_1}{T_p}$

$(1, \lambda)$-ES

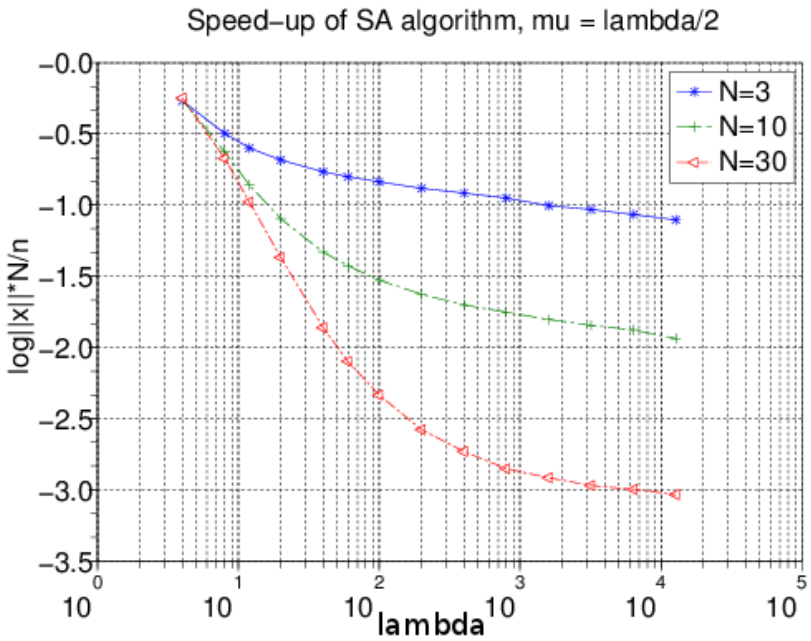
Speed-up logarithmic for all λ

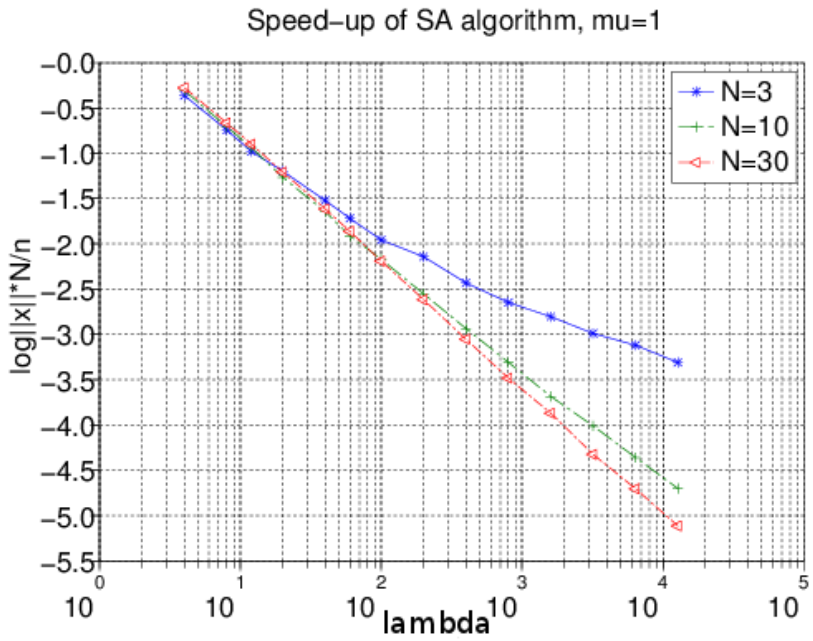
$(\mu/\mu, \lambda)$-ES

Speed-up linear until $\lambda < N$ and then logarithmic

Issues with large population sizes

[Evonum, 09]





$\Longrightarrow \mu = 1$: empirically, a better choice than $\frac{\lambda}{2}$ or $\frac{\lambda}{4}$

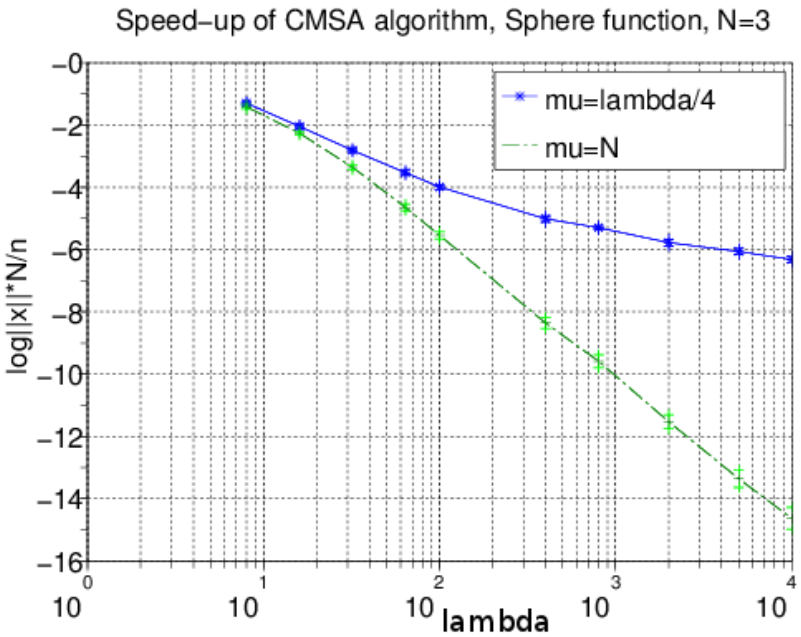
New selection ratio

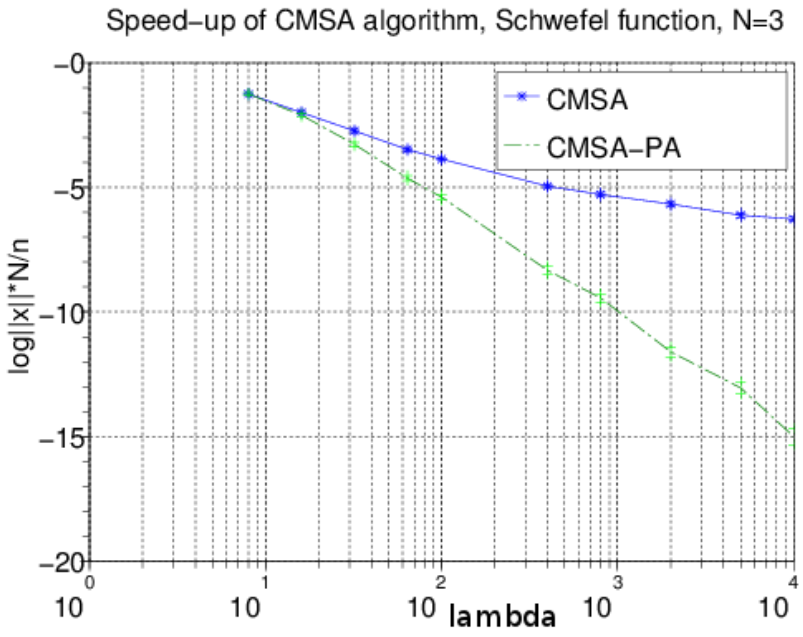
[Evonum, 10]

- New selection ratio (aka PA)
  - Idea: Having a bounded selection ratio
  - Rule: $\mu \leftarrow \min(N, \frac{\lambda}{4})$
- Experiments
  - CMSA, Sphere and Schwefel functions

New selection ratio

Results





Log-λ modification

[EA 09, GECCO 09, PPSN 10]

Problems

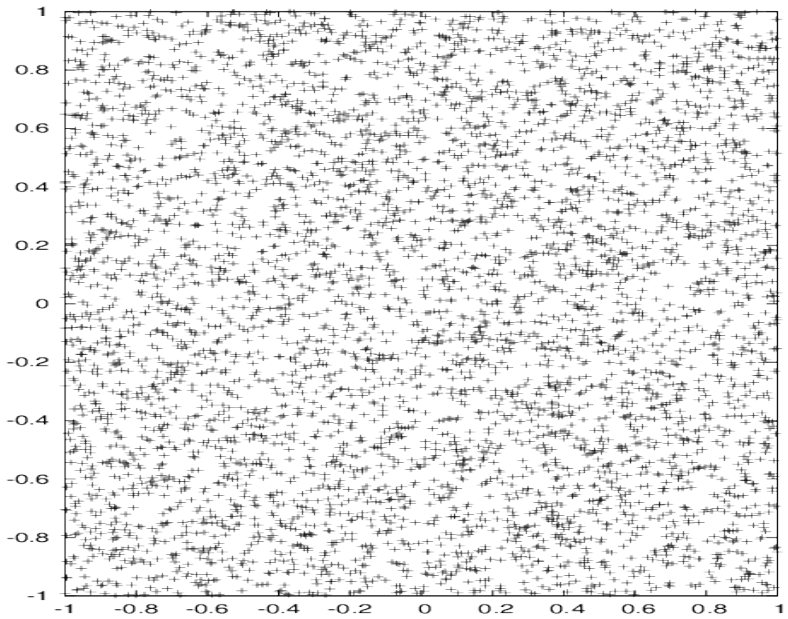
- The weighting trouble (bias . . .)
- The limited behavior for large population size (. . . and then variance)
- Possible redundant mutations

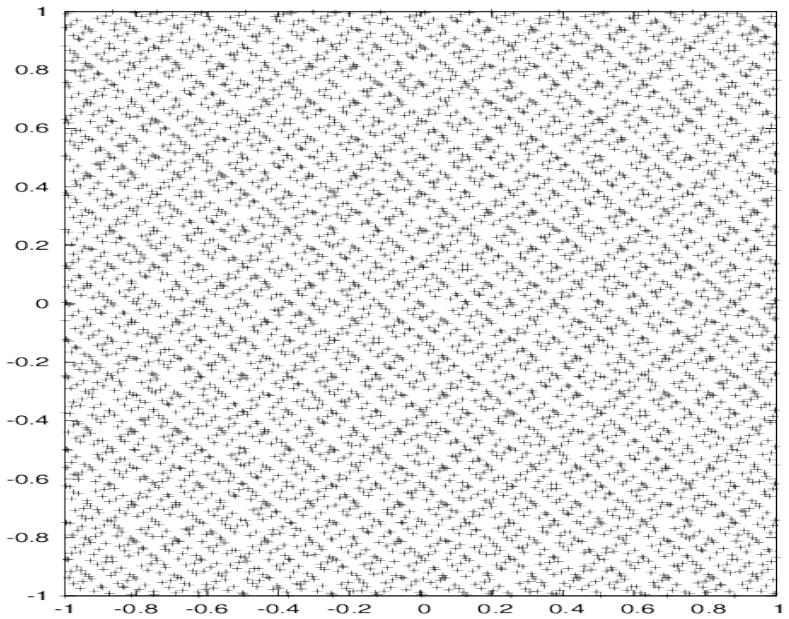
Ideas

- Reweighting
- A faster decrease of the step-size σ
- Quasi-random mutations

Quasi-Random mutations

[Teytaud, 08]

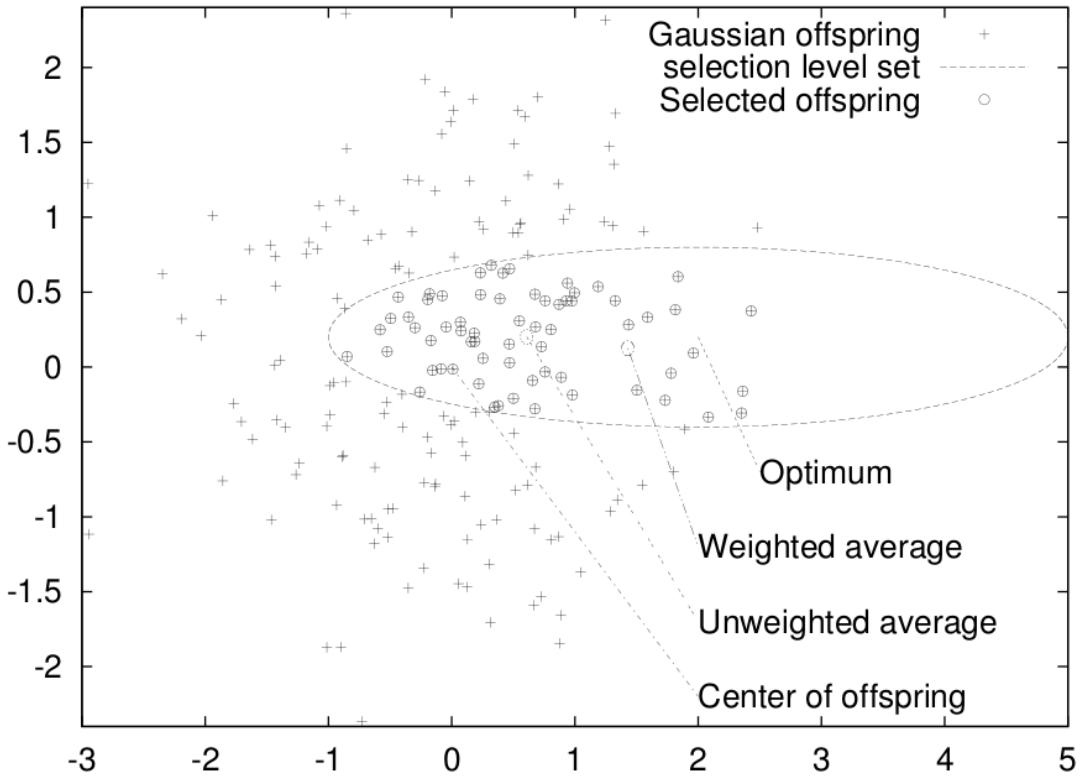




Correcting the bias

[Gecco, 09]

Compute: weight $\leftarrow$ 1/density of distribution



Log-λ modification

Experiments

- EMNA with
  - Faster decrease of σ
  - Reweighting
  - Quasi-Random mutations
- Function
  - Sphere function
- σ initialization
  - Not tuned
- All differences statistically signifiant (with 95% confidence)

Log-λ modification

Results

- Sphere function
- Initial σ not tuned

Dimension, lambda	Baseline	+QR	+ weight	+LB (IEMNA)
2,20	-0.000	-0.001	**-0.001**	-0.001
2,60	-0.001	-0.001	**-0.014**	-0.005
2,200	-0.001	-0.001	-1.501	**-2.106**
2,600	-0.001	-0.001	-1.748	**-2.640**
2,2000	-0.001	-0.001	-1.853	**-2.952**
3,30	-0.001	-0.001	**-0.003**	-0.002
3,90	-0.002	-0.003	**-0.165**	-0.096
3,300	-0.003	-0.003	-1.821	**-2.437**
3,900	-0.003	-0.003	-1.995	**-2.945**
4,40	-0.002	-0.003	**-0.005**	-0.004
4,120	-0.004	-0.004	-0.395	**-0.508**
4,400	-0.004	-0.005	-1.970	**-2.693**
4,1200	-0.005	-0.005	-2.131	**-3.086**
5,50	-0.003	-0.004	-0.007	**-0.008**
5,150	-0.005	-0.006	-0.609	**-0.779**
5,500	-0.006	-0.007	-2.087	**-2.786**
5,1500	-0.007	-0.007	-2.288	**-3.250**

Discussion

- New selection ratio ($\frac{\mu}{\lambda}$)
  - Positive improvement
  - No new parameter
  - Better as λ increases
- Faster decrease of σ
  - Improve the speed-up of EMNA
  - Can be dangerous if no proper reweighting/initialization
  - (Quasi-Random always good)

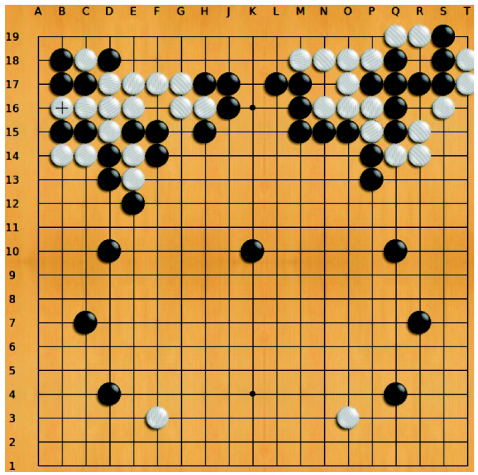
Outline

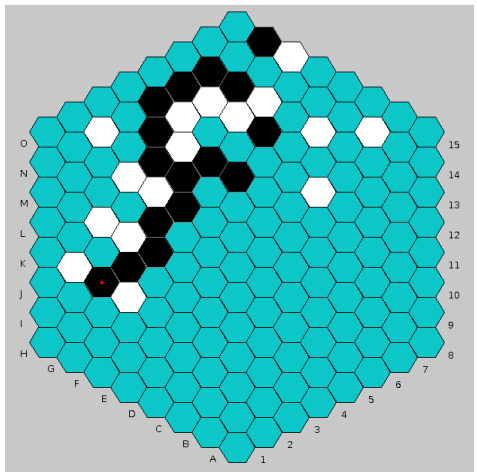
Problem

- Making decisions in an environment which is
  - Discrete
  - Fully observable
  - With finite horizon
  - Reward at the end
  - With a large number of states
- Goal
  - Find the best decision for each state

Games

- Why games ?
  - Well-designed
  - Simple and practical
  - Nice challenge for artificial intelligence
- Which games ?
  - Go
  - Havannah
- Goals
  - Improving the algorithm
  - Keeping the generality of the algorithm





Outline

State of the art algorithms

- Two-player games
  - Min-Max
  - Alpha-Beta
  - Monte-Carlo Tree Search
- One-player games
  - Dynamic programming
  - Nested Monte-Carlo Search
  - Nested Rollout Policy Adaptation

Monte-Carlo Tree Search (MCTS)

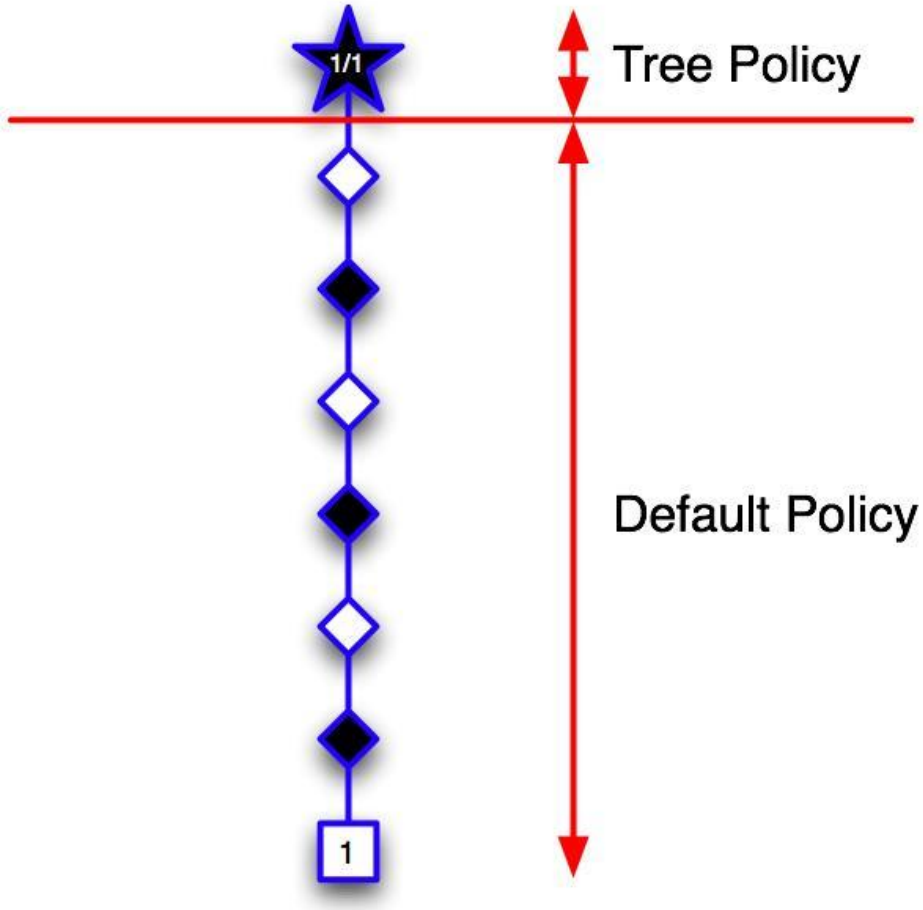
- Recent developments
  [Coulom 06][Chaslot et al., 06][Kocsis & Szepesvari, 06]

- Numourous applications
  - Active learning [Rolet & al., 09]
  - Non-linear Optimization [Auger & Teytaud, 10]
  - Feature Selection [Gaudel & Sebag, 09]
  - Planning [Xie et al., 11]
  - Games
    - Go,
    - Havannah (First use [ACG 09])
    - Hex,
    - Amazon (combined with an evaluation function),
    - . . .

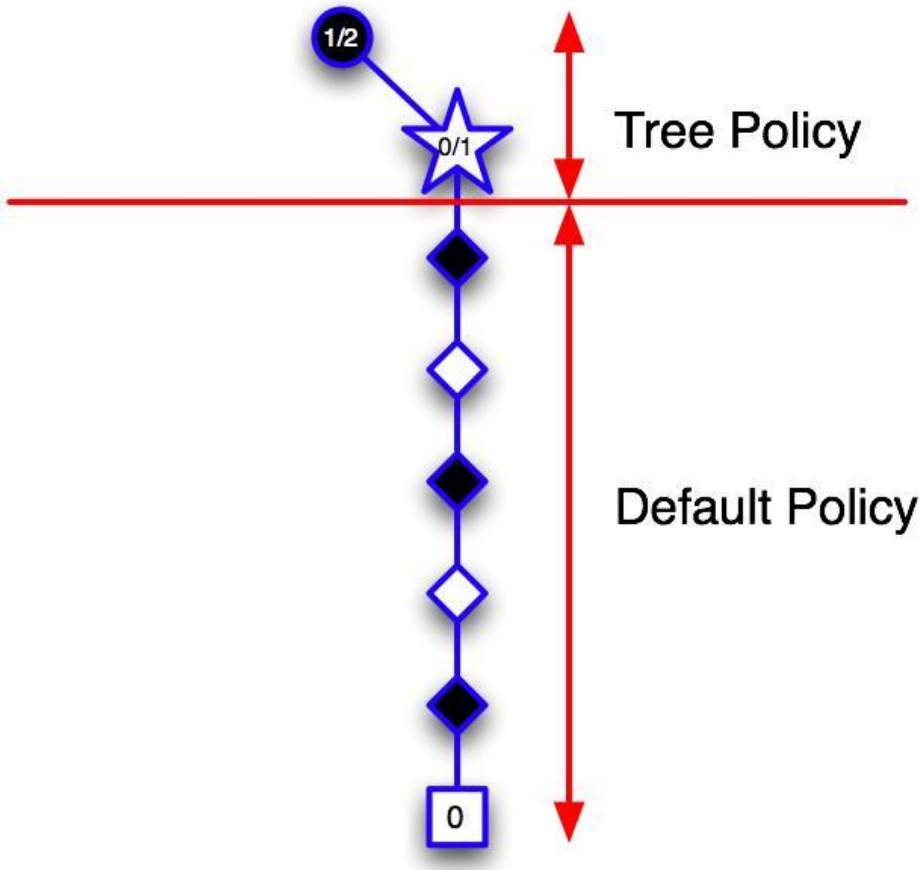
Monte-Carlo Tree Search (MCTS)

- Principle
  - Construction of an imbalance subtree of possible futures
  - Evaluation through Monte-Carlo simulations
  - Use of a bandit formula to bias the subtree
- 3 main steps
  - Descent in the subtree
  - Evaluation of the subtree
  - Growth and update of the subtree

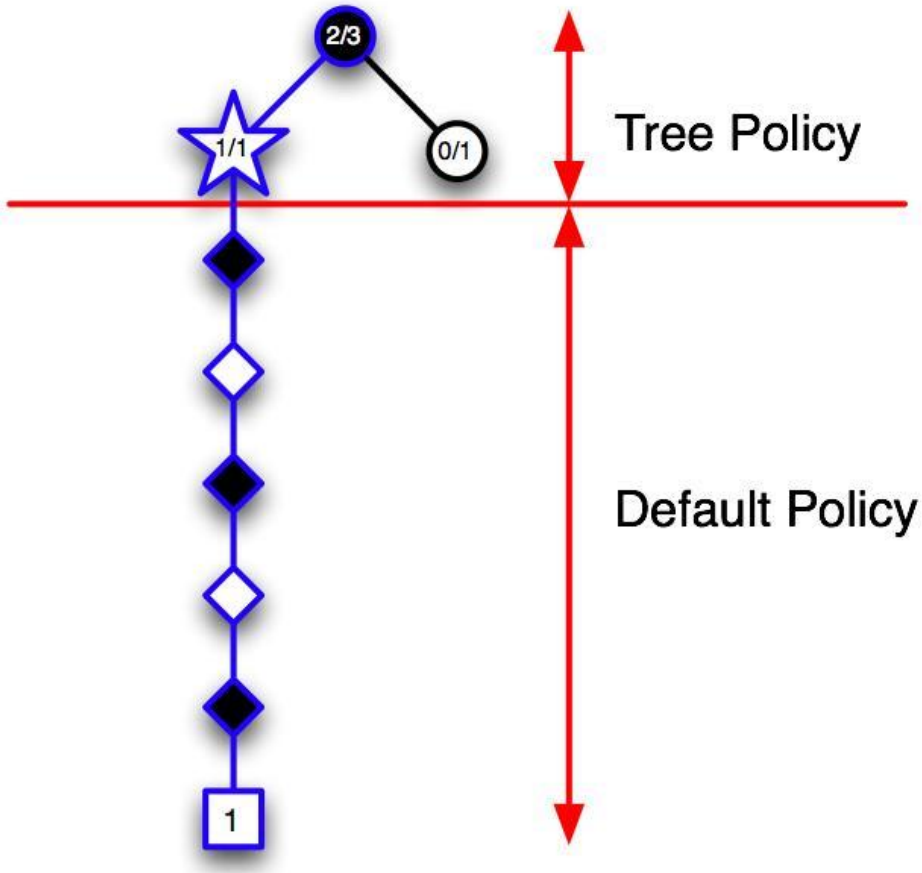
Monte-Carlo Tree Search (MCTS)



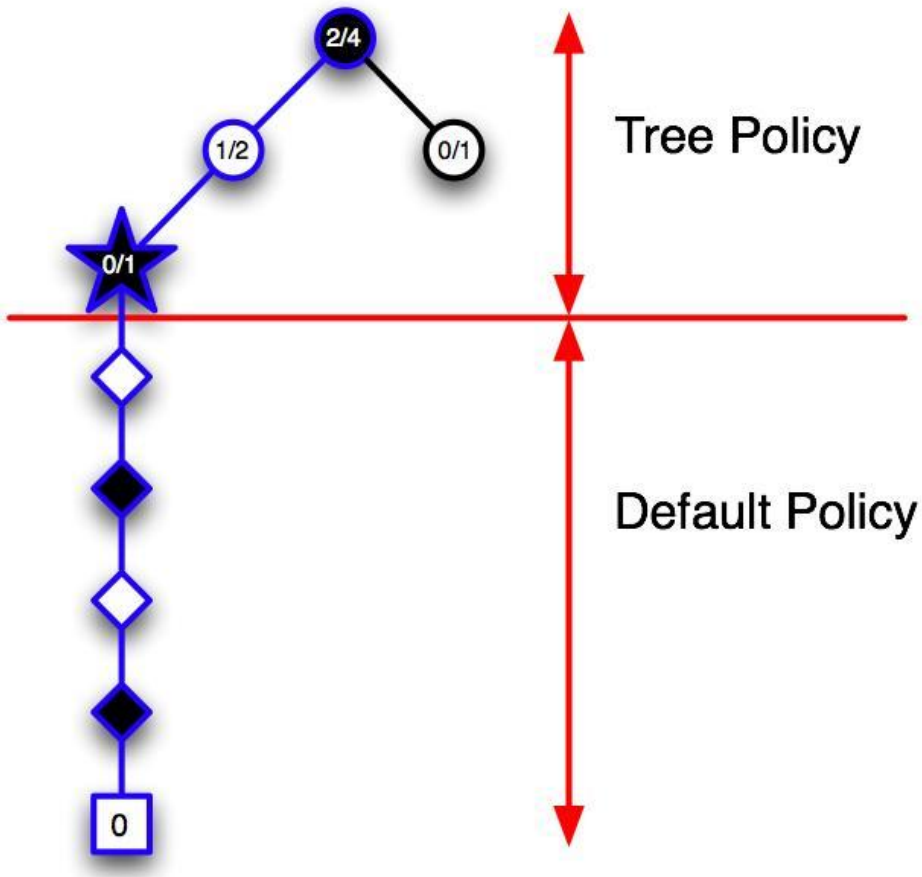
Monte-Carlo Tree Search (MCTS)



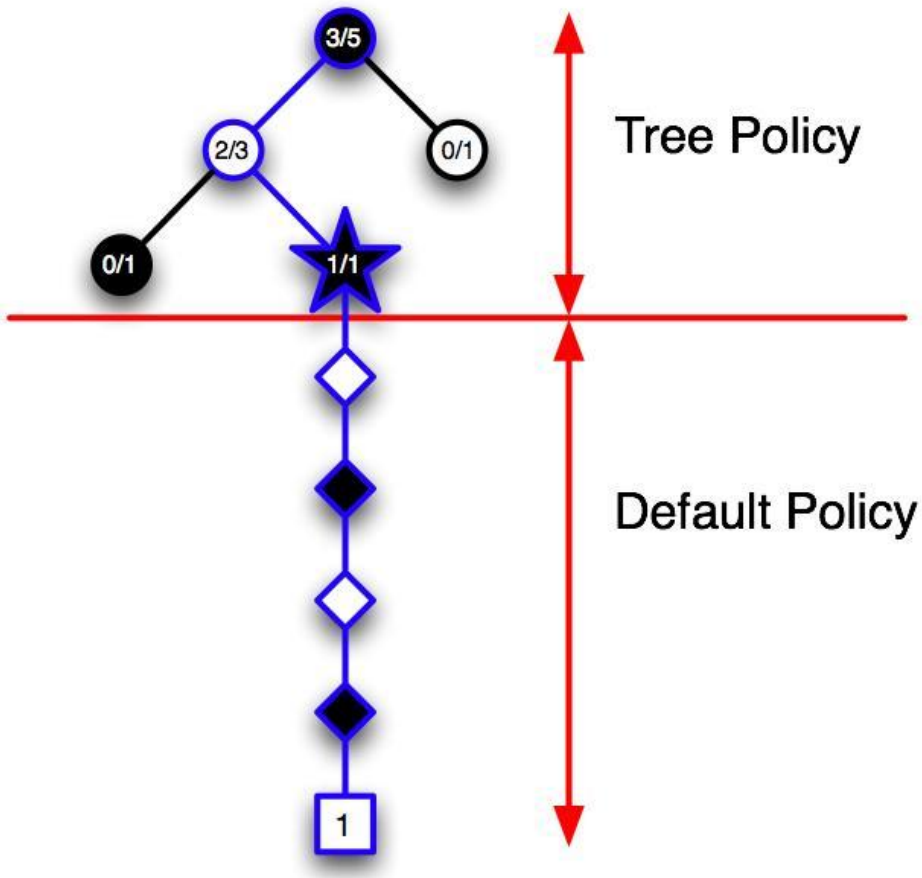
Monte-Carlo Tree Search (MCTS)



Monte-Carlo Tree Search (MCTS)



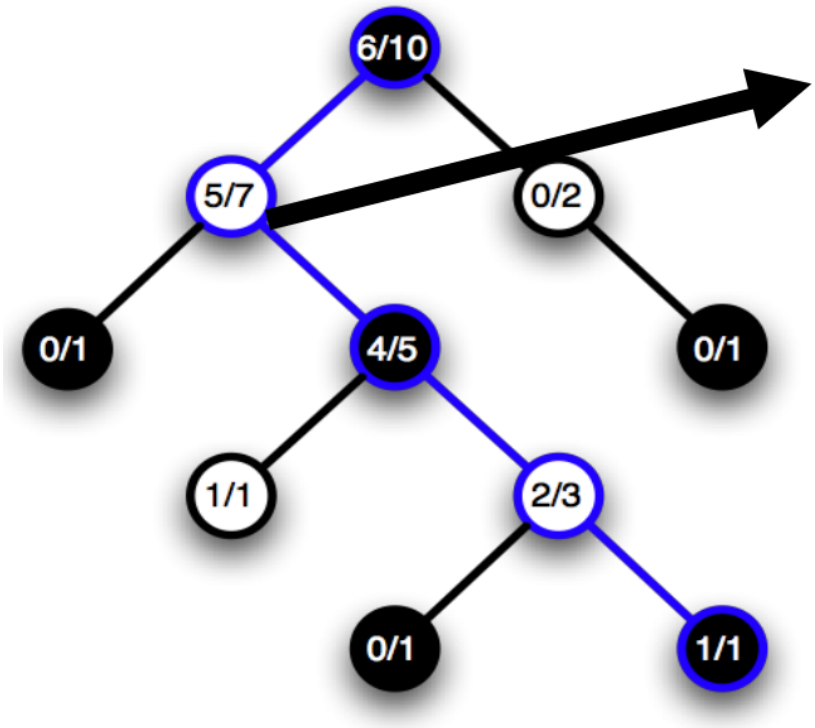
Monte-Carlo Tree Search (MCTS)



Monte-Carlo Tree Search (MCTS)

- Tree policy
  - UCB1 formula [Auer et al., 02]
  - Play arm i that maximizes $\hat{X} + p\sqrt{\frac{\log T}{T_i}}$
    - $\hat{X}_i$: Empirical average reward for move i
    - T : Total number of trials
    - T_i : Number of trials for move i
- Default policy
  - Monte-Carlo Simulation
    $\Rightarrow$ Random choice until the end

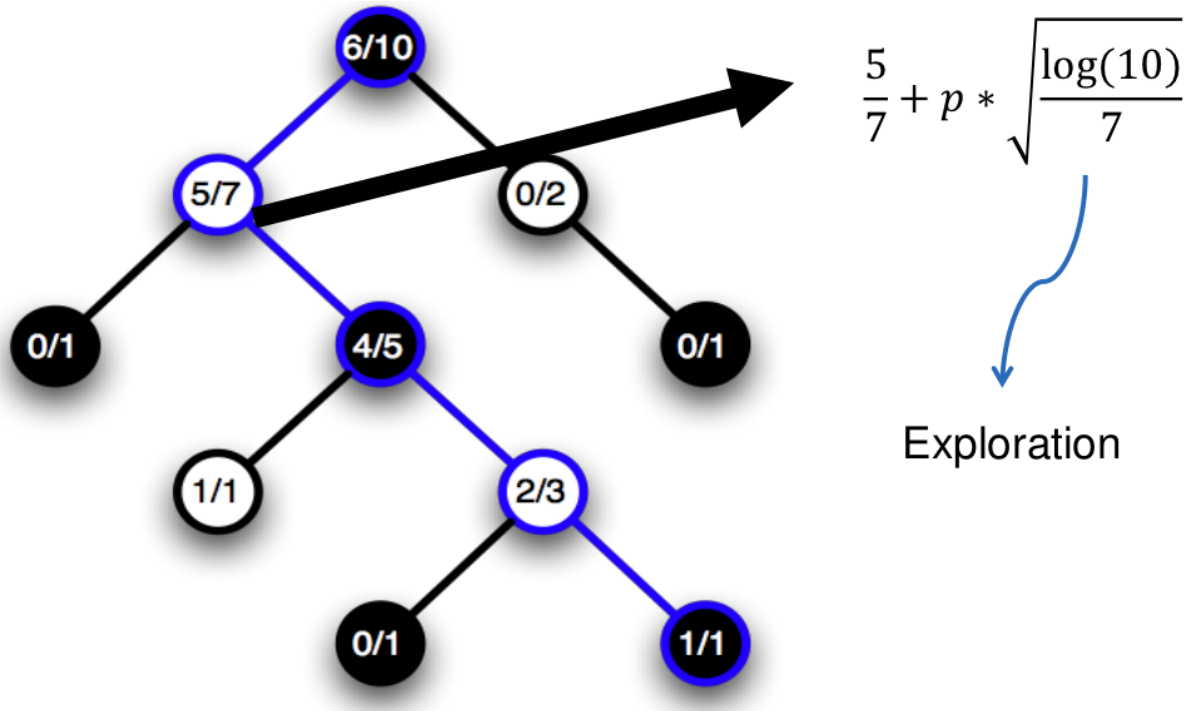
Monte-Carlo Tree Search (MCTS)



$$\frac{5}{7} + p * \sqrt{\frac{\log(10)}{7}}$$

Exploitation

Monte-Carlo Tree Search (MCTS)

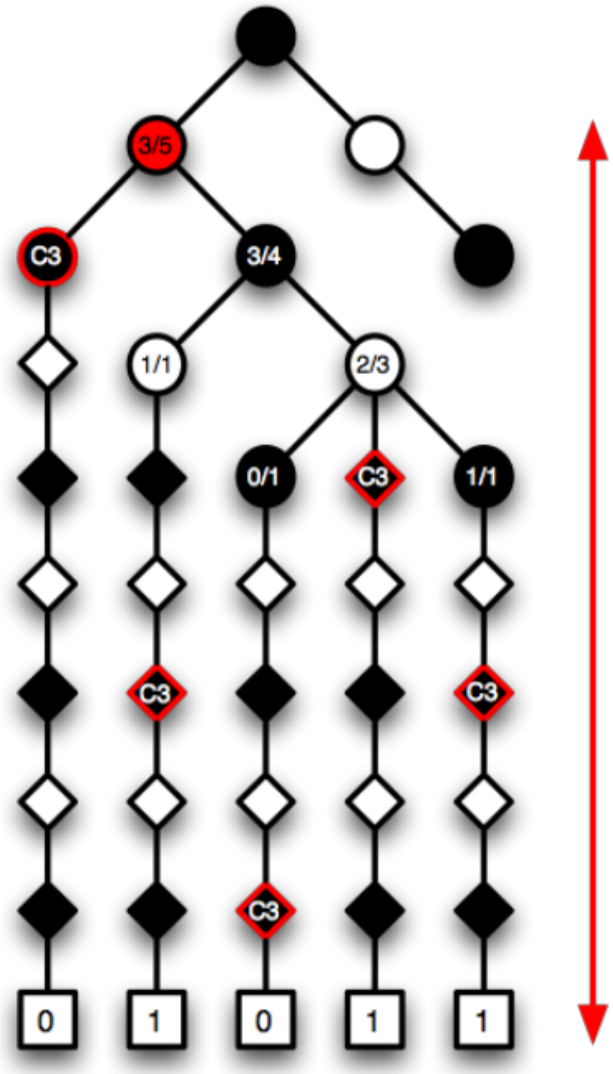


Rapid Action Value Estimate (RAVE)

[Gelly & Sylver, 07]

- Keep for each node n and each move i :
  the number of wins and losses where i has been played *after* n
- Compute a score $V_{RAVE}(i)$:
  empirical score when i has been played *after* n
- NewScore $\leftarrow (1-\alpha)\hat{X}(i) + \alpha V_{RAVE}(i)+$ exploration
  with $\alpha \leftarrow \sqrt{\frac{R}{R+n}}$
- RAVE scores are biased but have a small variance

Rapid Action Value Estimate (RAVE)



Monte-Carlo Tree Search (MCTS)

Pros

- Efficient
- Evaluation function not needed
- Generic
- Anytime

Cons

- Can we do better than pure Monte-Carlo ?
  $\Longrightarrow$ Improving the default policy

Outline

Generic improvements

3 generic rules

- poolRave
- Contextual Monte-Carlo
- Decisive moves and anti-decisive moves

Generic improvements

3 generic rules

- poolRave
- Contextual Monte-Carlo
- Decisive moves and anti-decisive moves [CIG 2010]

PoolRave

PoolRave [ACG 10]

- Based on RAVE
- Use RAVE values for biasing the next Monte-Carlo simulations
  - Compute a pool of good moves according to RAVE
  - When a decision has to be made in the default policy
    - Play a move in the pool with probability p
    - Play a random move with probability 1-p

CMC [EvoGames 2010]

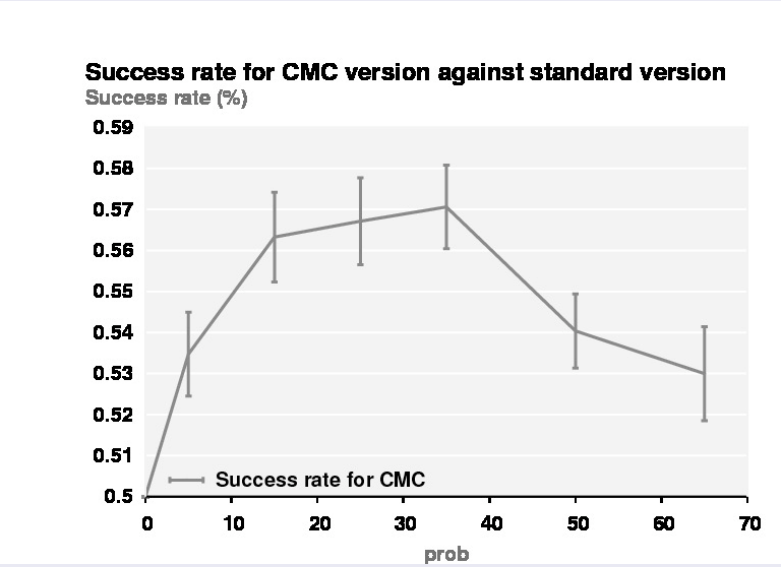
- Keep for each node n and for each moves A and B: The number of wins and losses where A and B have been played after n by the current player
- Compute a score $V_{CMC}(A, B)$: empirical score when A and B have been played after n
  - If B is the last move played by the current player
  - Find the move A which maximizes $V_{CMC}(A, B)$
    - Play A with probability p
    - Play a random move with probability $1 - p$

Results

Poolrave

Application	# of simulations	Best Score against the baseline
Havannah	1000	54.32% ±0.46%
Havannah	10000	54.45% ±0.75%
Havannah	20000	54.42% ±0.89%
Go (9x9)	1000	62.7% ±0.9%
Go (9x9)	10000	64.4% ±0.4%

CMC



Discussion

- All rules are generic
- A small but significative improvement
- PoolRave also tested against a baseline with handcrafted expert-knowledge
  $\rightarrow 51.7\%$
- Comparison with (recent) existing methods:
  - Last Reply [Drake, 09]
  - N-grams [Stankiewicz et al, 11]

Outline

Traveling Salesman Problem (TSP)

Data

- List of cities
- Distances between all cities

Goal

- Find a path visiting each city exactly once
- The path must be as short as possible

Formalization

Data

- $G(N, A)$ is a complet non directed graph
- $N = 0, 1, \ldots, n$ is a set of nodes
- $A = N * N$ is a set of vertices
- $C : A \rightarrow R$ is a cost function
- A solution is a sequence $P = (p_0, \ldots, p_n, p_{n+1})$ where p_0 and p_{n+1} correspond to the node 0 and $p_0, \ldots, p_n$ is a permutation of N

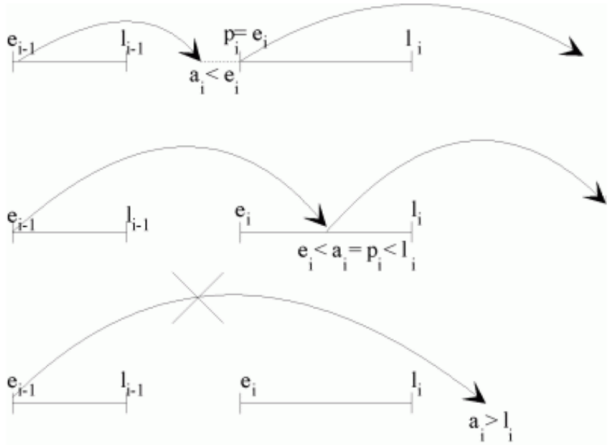
Goal

$$\text{cost}(P) \leftarrow \sum_{k=0}^{n} c(a_{p_k, p_{k+1}})$$

Traveling Salesman Problem with Time Windows(TSPTW)

- Additionnal property: Time windows
  $\Rightarrow$ A city must be visited within a certain time interval

- Some problems have no solution
- Finding a path is NP-hard



TSPTW

- The function to optimize is the same
- Each node i is associated to an interval $[e_i, l_i]$
- These constraints must be satisfied:
  - $\forall p_k, r_{p_k} < l_{p_k}$
  - Departure time : $d_{p_k} \leftarrow \max(r_{p_k}, e_{p_k})$
    where r_{p_k} is the arrival time at node p_k
- NewCost(P) $\leftarrow$cost(P)$+10^6 \times \Omega(P)$
  where $\Omega(P)$ is the number of violated constraints

Outline

Nested Monte-Carlo Search (NMC)

[Cazenave, 09]

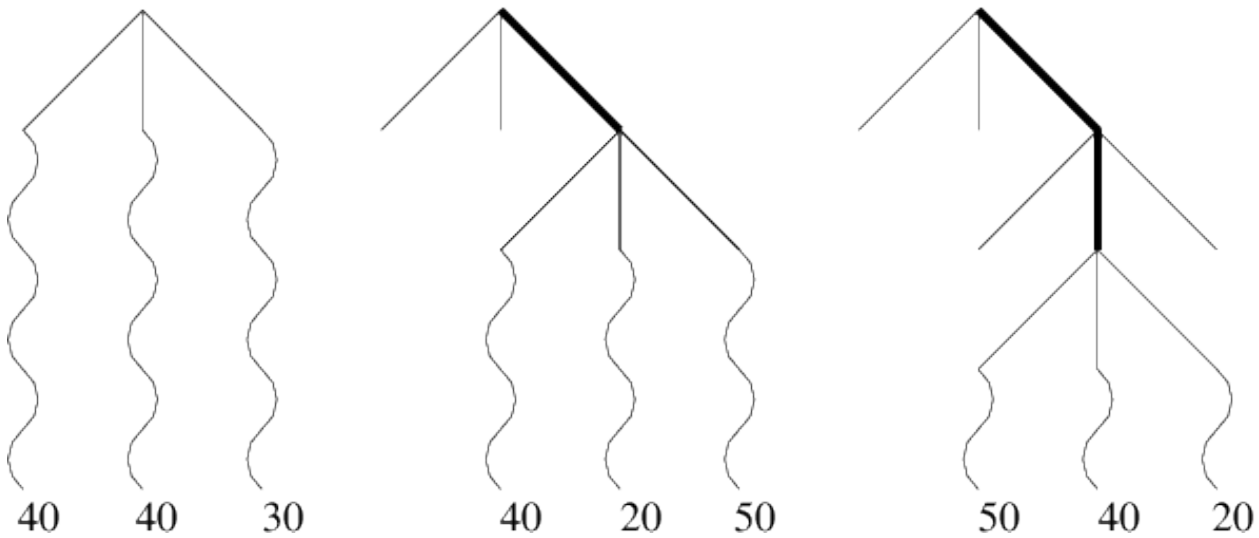
- Tree exploration algorithm
- Evaluation with Monte-Carlo Simulations
- Particulary efficient for one player games and when late decisions are as important as early ones

- NMC plays a whole game and returns the associated score
- NMC takes for parameters the level and the current position (recursive algorithm)

Nested Monte-Carlo Search (NMC)

- Level 0
  - Monte-Carlo policy
  - Each city is randomly chosen
- Level $>$ 0
  - Launch NMC(current level-1)
  - The action with the highest score is chosen

NMC(level 1) example



Adding heuristics

[EvoTranslog 11]

- The algorithm can be improve by biasing the Monte-Carlo policy
- Instead of uniformly random, decisions are chosen accordingly to the probability (using a Boltzmann softmax policy):

$$\pi_\sigma(p,a) \leftarrow \frac{e^{\phi(p,a)^T\sigma}}{\sum_b e^{\phi(p,b)^T\sigma}}$$

$\phi(p,a)$: heuristics vector

σ : weights of the heuristics

- The added heuristics are :
  - The distance to the last node
  - The waiting time (related to e_i)
  - The remaining time before the end of the time window (related to l_i)
- Tuning of the weights done with an evolution strategy (SA-ES)

Nested Rollout Policy Adaptation (NRPA)

[C. Rosin, 11]

- NMC can be improved by modifying the Monte-Carlo simulations
- Instead of random playouts, a policy is learned:
  - Increase weights of best decisions
  - Decrease weights of other decisions
  - For each city, compute a probability proportional to the exponential of its weight

NRPA

- Level 0
  - Adapted policy
  - Each city is chosen accordingly to its probability
- Level > 0
  - Launch N iterations of NRPA(current level-1)
  - Update:
    - The scores
    - The sequences
    - The policy

Adding heuristics

[LION 6, 2012]

Guiding the rollout using domain-specific knowledge

- Force to visit cities as soon as they go after their window end
- Avoid visiting a city it it makes another city go after its window end
- Consider all cities if no city avalible after these two tests
- Important point : Optimal moves are not pruned

Experiments and results

Problem	# cities	State of the art	NMC_EK score	NRPA score	NRPA_EK score
rc206.1	4	117.85	**117.85**	**117.85**	**117.85**
rc207.4	6	119.64	**119.64**	**119.64**	**119.64**
rc202.2	14	304.14	**304.14**	**304.14**	**304.14**
rc205.1	14	343.21	**343.21**	**343.21**	**343.21**
rc203.4	15	314.29	**314.29**	**314.29**	**314.29**
rc203.1	19	453.48	**453.48**	**453.48**	**453.48**
rc201.1	20	444.54	**444.54**	**444.54**	**444.54**
rc204.3	24	455.03	**455.03**	**455.03**	**455.03**
rc206.3	25	574.42	**574.42**	**574.42**	**574.42**
rc201.2	26	711.54	**711.54**	**711.54**	**711.54**
rc201.4	26	793.64	**793.64**	**793.64**	**793.64**
rc205.2	27	755.93	**755.93**	**755.93**	**755.93**
rc202.4	28	793.03	**793.03**	800.18	**793.03**
rc205.4	28	760.47	**760.47**	765.38	**760.47**

Experiments and results

Problem	# cities	State of the art	NMC_EK score	NRPA score	NRPA_EK score
rc202.3	29	837.72	**837.72**	839.58	839.58
rc208.2	29	533.78	536.04	537.74	**533.78**
rc207.2	31	701.25	707.74	702.17	**701.25**
rc201.3	32	790.61	**790.61**	796.98	**790.61**
rc204.2	33	662.16	675.33	673.89	664.38
rc202.1	33	771.78	776.47	775.59	772.18
rc203.2	33	784.16	**784.16**	**784.16**	**784.16**
rc207.3	33	682.40	687.58	688.50	**682.40**
rc207.1	34	732.68	743.29	743.72	738.74
rc205.3	35	825.06	828.27	828.36	**825.06**
rc208.3	36	634.44	641.17	656.40	650.49
rc203.3	37	817.53	837.72	820.93	**817.53**
rc206.2	37	828.06	839.18	829.07	**828.06**
rc206.4	38	831.67	859.07	831.72	**831.67**
rc208.1	38	789.25	797.89	799.24	793.60
rc204.1	46	868.76	899.79	883.85	880.89

Discussion

Pros

- Successful optimization for the nested
- 77% of state of the art scores found for NRPA_EK
- Better understanding of the NRPA algorithm (convergence to local optima)

Cons

- No new record
- Difficulties for large number of nodes

Conclusion

- Evolutionary optimization
  - Auto-adaptative solutions for black-box optimization
  - Continuous search space (but exists also in discrete)
- Making decisions under incertainty
  - Discrete case (but exists also in continuous)
  - Combinatorial optimization

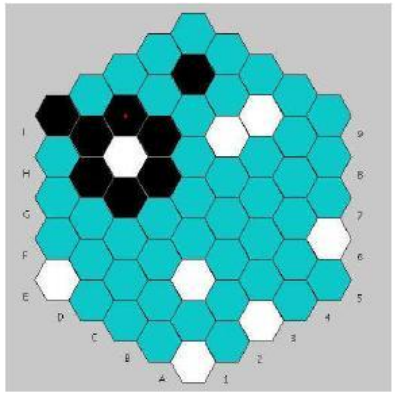
Some other publications

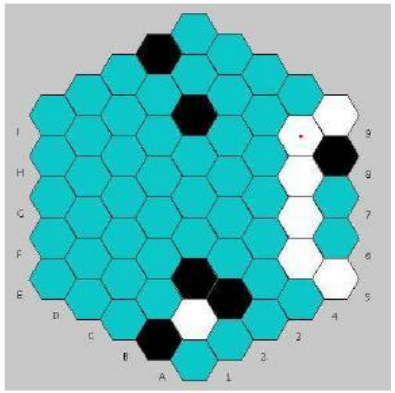
- A cognitive science perspective (random positions in Go, Blind Go) [CIG 11]
- Using Quasi-Random restarts and decreasing σ for multimodal optimization [EA 11]
- Towards a solution of 7x7 Go with Meta-MCTS [ACG 11]
  1st serie of 20 wins over 20 games against pro players in 7x7 Go

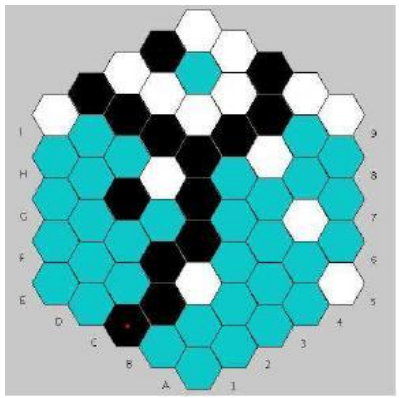
Thank you for your attention

Havannah

- Created by Christian Freeling
- Two-player board game
- Hexagonal board of hexagonal locations
- Connection game
- To win, a player has to realize
  - A ring
  - A bridge
  - A fork

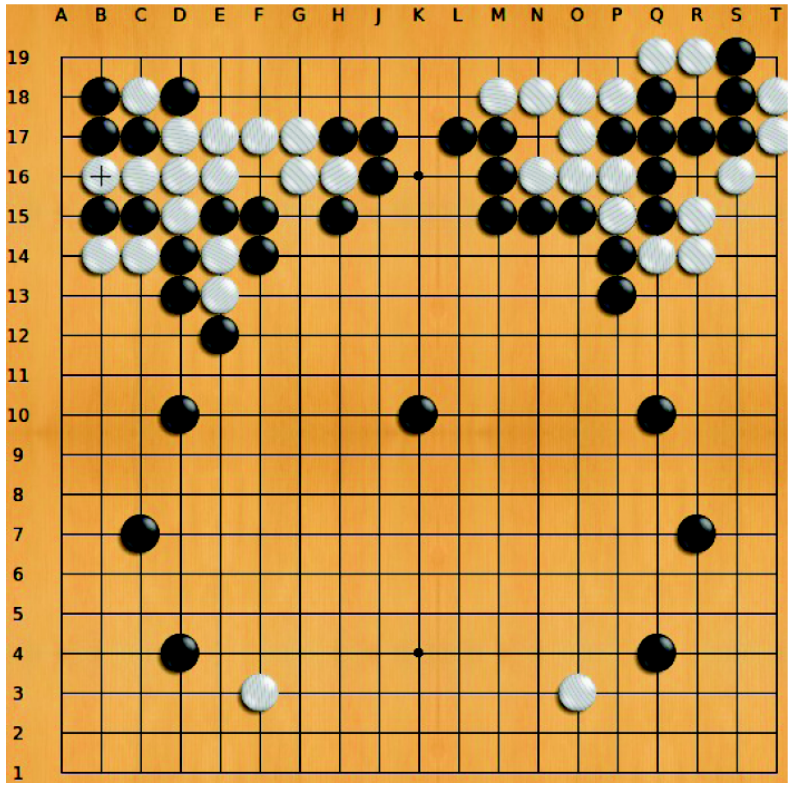






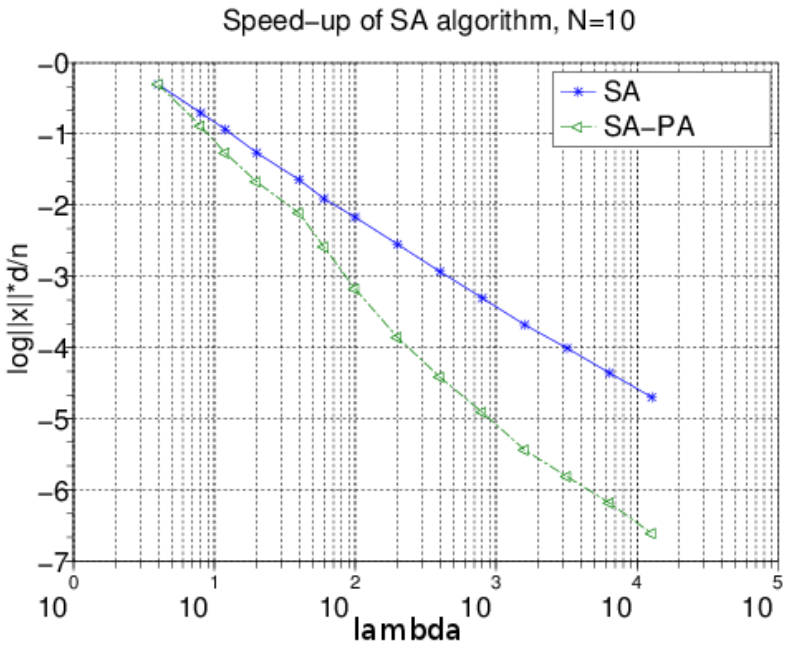
Go

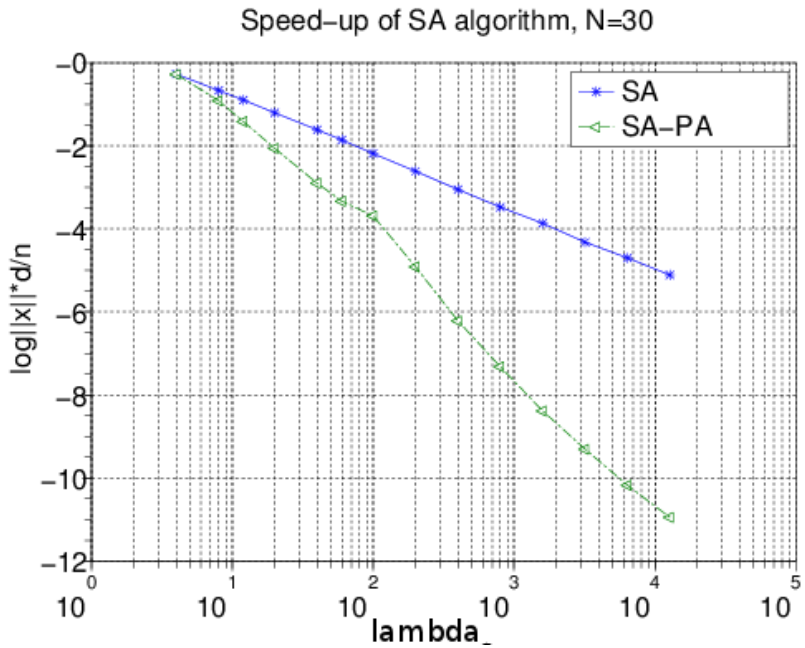
- Two-player board game
- Board 9x9, 19x19
- Each player puts alternatively a stone of the board
- A stone is removed if it has no more liberties
- The player with the biggest area wins



New selection ratio

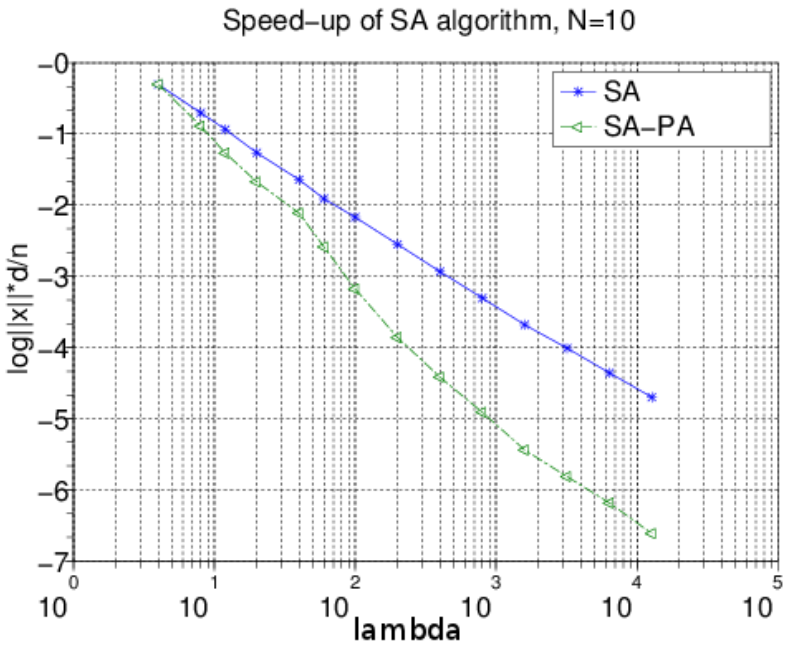
Results

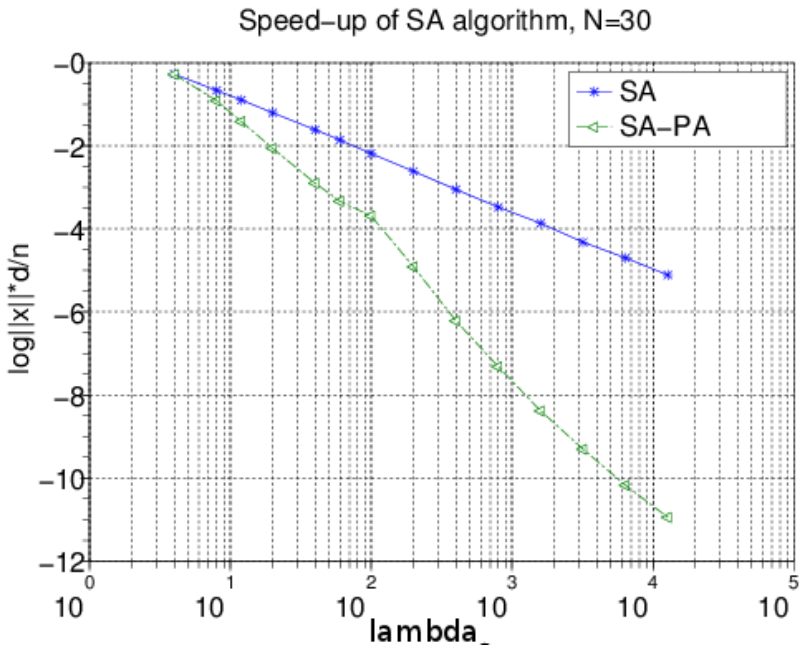




New selection ratio

Results





Log-λ modification

Results

- Sphere function
- Initial σ tuned

Dimension, lambda	Baseline	+QR	+ weight	+LB (IEMNA)
2,20	-0.345	-1.252	-1.743	**-2.103**
2,60	-1.967	-2.086	-2.022	**-2.713**
2,200	-2.050	-2.089	-2.112	**-3.004**
2,600	-2.080	-2.101	-2.061	**-3.190**
2,2000	-2.103	-2.188	-2.111	**-3.434**
3,30	-0.697	-2.299	-2.277	**-2.398**
3,90	-2.330	-2.392	-2.282	**-3.047**
3,300	-2.340	-2.404	-2.293	**-3.302**
3,900	-2.369	-2.443	-2.320	**-3.519**
3,3000	-2.404	-2.476	-2.367	**-3.726**
4,40	-0.749	-2.541	-2.397	**-2.578**
4,120	-2.543	-2.627	-2.443	**-3.271**
4,400	-2.601	-2.658	-2.480	**-3.547**
4,1200	-2.642	-2.717	-2.519	**-3.764**
5,50	-1.330	-2.885	-2.677	**-2.730**
5,150	-2.790	-2.858	-2.622	**-3.488**
5,500	-2.828	-2.908	-2.673	**-3.750**
5,1500	-2.886	-2.964	-2.718	**-3.975**

Log-λ modification

Results

- Multimodal function
- Initial σ tuned

Dimension, lambda	Baseline	+QR	+ weight	+LB (IEMNA)	P-value for QR
2,20	-0.709	**-1.301**	-1.105	-0.529	0
2,60	-1.139	**-1.181**	-0.655	-1.157	2.315e-07
2,200	**-1.104**	-1.074	-0.402	-0.822	1
2,600	-1.100	**-1.133**	-0.119	-0.534	0
2,2000	-1.124	**-1.146**	-0.124	-0.210	3.187e-09
2,6000	-1.144	**-1.162**	-0.173	-0.181	7.199e-11
3,30	-0.971	**-1.332**	-0.799	-0.537	1.721e-09
3,90	**-1.231**	-1.229	-0.481	-0.619	0.559
3,300	-1.210	**-1.243**	-0.178	-0.233	1.415e-05
3,900	-1.240	**-1.252**	-0.181	-0.179	0.031
3,3000	-1.269	**-1.307**	0.081	-0.031	6.342e-12
4,40	-1.204	**-1.388**	-0.713	-0.858	6.262e-06
4,120	**-1.357**	-1.353	-0.344	-0.480	0.624
4,400	-1.352	**-1.368**	-0.205	-0.183	0.013
4,1200	-1.389	**-1.427**	0.224	0.093	7.638e-10
5,50	-1.359	**-1.520**	-0.702	-0.445	0.000
5,150	-1.477	**-1.503**	-0.351	-0.391	0.010
5,500	-1.495	**-1.518**	-0.145	-0.161	0.001
5,1500	-1.539	**-1.579**	0.726	0.715	2.300e-08