Algorithms for context prediction in Ubiquitous Systems

Prediction by randomised search approaches

Stephan Sigg

Institute of Distributed and Ubiquitous Systems
Technische Universität Braunschweig

February 3, 2009

Overview and Structure

- Introduction to context aware computing
- Basics of probability theory
- Algorithms
  - Simple prediction approaches: ONISI and IPAM
  - Markov prediction approaches
  - The State predictor
  - Alignment prediction
  - Prediction with self organising maps
  - Stochastic prediction approaches: ARMA and Kalman filter
  - Alternative prediction approaches
    - Simulated Annealing
    - Evolutionary algorithms
    - Neural networks
    - Dempster shafer

Overview and Structure

- Introduction to context aware computing
- Basics of probability theory
- **Algorithms**
  - Simple prediction approaches: ONISI and IPAM
  - Markov prediction approaches
  - The State predictor
  - Alignment prediction
  - Prediction with self organising maps
  - Stochastic prediction approaches: ARMA and Kalman filter
  - **Alternative prediction approaches**
    - **Simulated Annealing**
    - **Evolutionary algorithms**
    - Neural networks
    - Dempster shafer

Outline

Prediction by randomised search approaches

1. Randomised seach approaches
2. Evolutionary approaches
3. Restrictions of evolutionary approaches
4. Design of evolutionary algorithms
5. Context prediction with evolutionary algorithms
6. Properties of evolutionary prediction approaches

Introduction

Randomised search approaches

- Randomised search approaches combine methods that utilise random variables to guide the search for an optimum in a search space
  - Evolutionary algorithms
  - Simulated Annealing
  - Tabu search
  - ...

Introduction

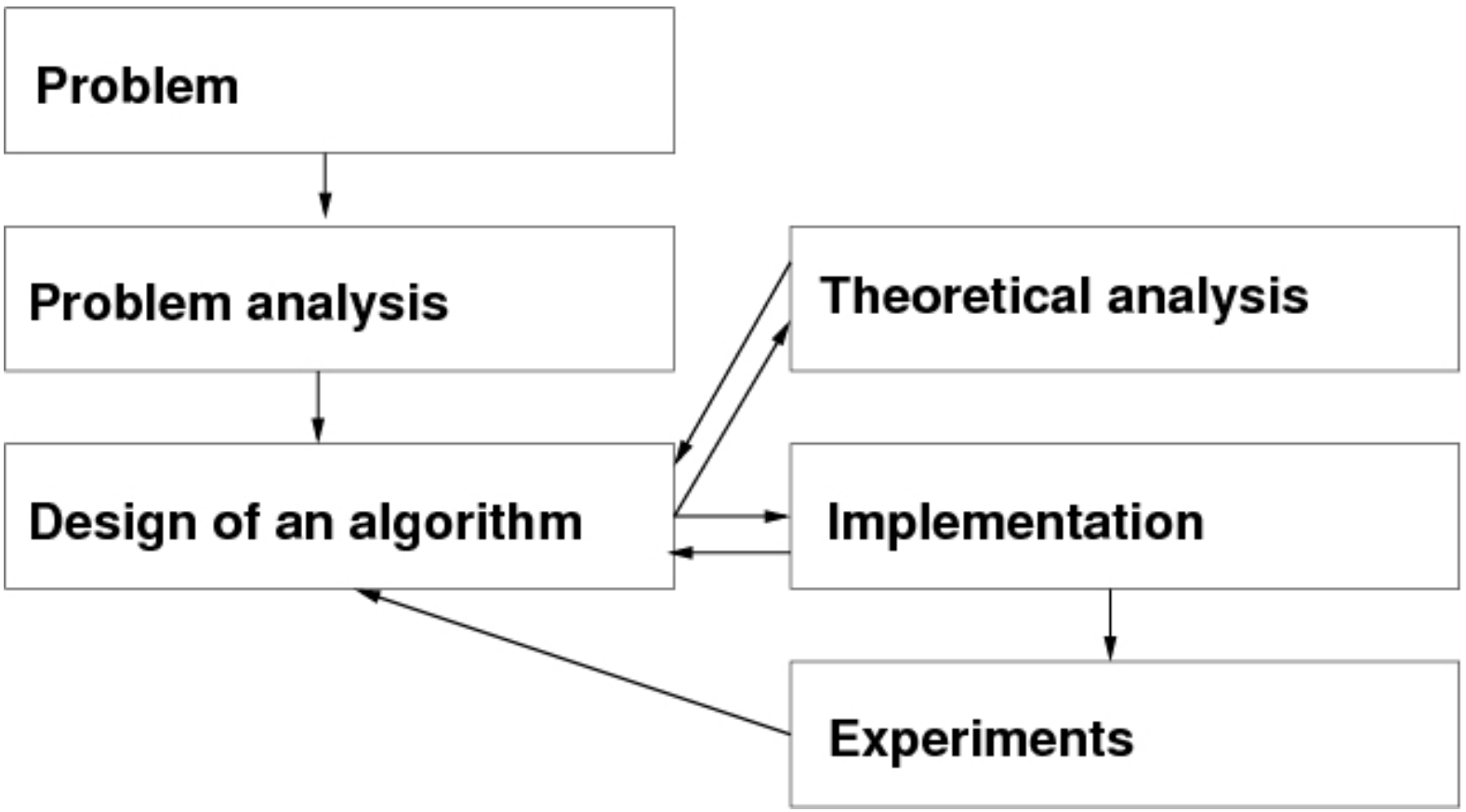
Randomised search approaches

- In contrast to deterministic approaches, random search mechanisms are not necessarily designed for a specific problem
- The algorithm searches for a point in the search space that is considered the optimum regarding a scoring function (fitness function)
- Problem specific modelling of the search space not necessarily required
- Black box optimisation

Introduction

Randomised search approaches

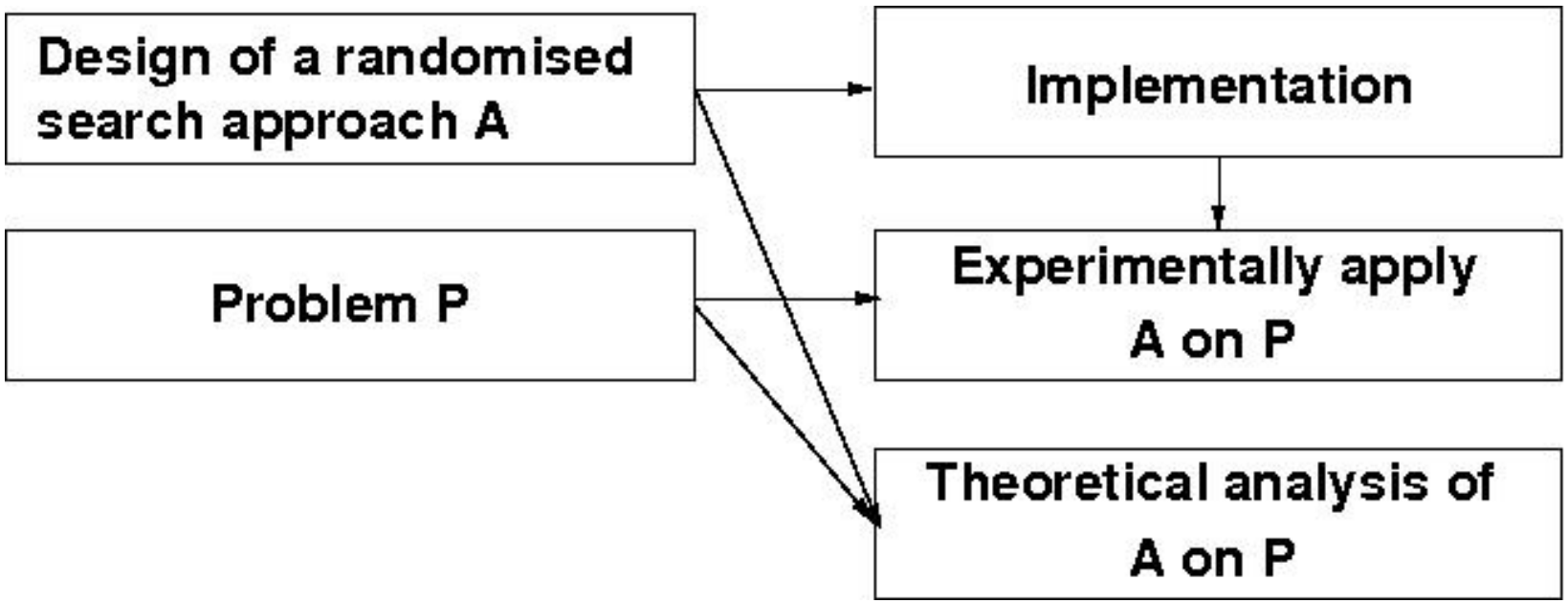
- Classical approach to solve an optimisation problem:



Introduction

Randomised search approaches

- Random approach to solve an optimisation problem:



Introduction

Randomised search approaches

- We distinguish between
  - A search space (Genotype)
  - A feature space (Phenotype)
  - A Genotype-Phenotype-Mapping
  - A scoring function (Fitniss function)
- Example
  - Genotype (binary string): 0110010
  - Phenotype (Real valued): 12

Hill climbing methods

Introduction

- Hill climging strategies
  - Searching manner reminds of intuitive way to climb a mountain (by a sightless climber)
  - Most frequently applied in engineering design
    - Assumptions to state extrema are not fulfilled (e.g. unfriendly/unknown conditions)
    - Difficulties to carry out necessary differentiations
    - Solution to the equations describing all conditions does not always lead to optimum point in the search space
    - Equations to describe conditions are not immediately solvable

Hill climbing methods

Introduction

- Problem formulation either maximisation or minimisation (here max):
  - Problem to solve: $max_x\{F(x)|x \in \mathbb{R}^n\}$
  - Column vector at optimum position required: $(X_1^*, x_2^*, \ldots, x_n^*)^T$
  - As well as Optimum value $F^* = F(x^*)$

Hill climbing methods

Introduction

- Various types of minima (maxima) can be distinguished between:
  - Strong
  - Weak
  - Local
  - Global

Hill climbing methods

Introduction

Local maximum

For a local maximum the following conditions hold:

$$F(x^*) \geq F(x)$$

$$0 \leq ||x - x^*|| = \sqrt{\sum_{i=1}^{n} (x_i - x_i^*)^2} \leq \varepsilon$$

$$x \in \mathbb{R}^n$$

Hill climbing methods

Introduction

- The Maximum is called strong, if $F(x^*) < F(x)$ for $x \neq x^*$.
- If the objective function has only one maximum it is called unimodal
- The highest local maximum of an objective function is called the global maximum.

Hill climbing methods

Local random search

Local random search

For every point x in a search space S, a non-empty neighbourhood $N(x) \subseteq S$ is defined. The local random search approach iteratively draws one sample $x' \in N(x)$. When the fitness of the new value is better than the old one ($F(x) < F(x')$), the new value is utilised as the new best search point. Otherwise it is discarded.

Hill climbing methods

Local random search

- In principle, $N(x) = x$ or $N(x) = S$ is valid, but the original idea is that $N(x)$ is a relatively small set of search points.
- The points $x' \in N(x)$ are expected to be nearer to x than those points $x'' \notin N(x)$
- Typically, $x \in N(x)$

Hill climbing methods

Local random search

- Example: $S = \{0,1\}^n$ and $N_d(x)$ are all points y with Hamming distance smaller than d ($H(x,y) \leq d$)

$$|N_d(x)| = \binom{n}{d} + \binom{n}{d-1} + \cdots + \binom{n}{1} + \binom{n}{0}$$

  - For constant d we obtain: $|N_d(x)| = \Theta(n^d) \ll |S| = 2^n$

Hill climbing methods

Local random search

- Local search belongs to the class of hill climbing search methods since the next search point is never chosen to decrease the fitness function.
- For deterministic random search:
  - $x' = max_{\chi}(N(x))$
  - This implies that always the highest slope is propagated

Hill climbing methods

Local random search

- Problems with local search heuristics:
  - When the neighbourhood is too small, the search will easily result in local optima only
  - When the neighbourhood is too big, the method approximates random search.
  - It can therefore be beneficial to change the neighbourhood radius during dependent on the search progress.
    - Initially, the neighbourhood is big in order to allow huge steps in the search space
    - Later on the neighbourhood is decreased in order to search in the direct neighbourhood of the already found search points
    - The challenging task is not to decrease to neighbourhood too fast since this may force the search method to remain at a local optimum.

Hill climbing methods

Local random search

- In order to solve the problem that the local search might terminate in a local optimum, multistart strategies are applied
  - The local search approach is applied t times on the problem domain.
  - Probability amplification results in respectable search result also when single success probability is low.
    - Assume a success probability of $\delta > 0$ for one interation of the algorithm
    - When the algorithm is applied t times, the overall probability of success is $1 - (1 - \delta)^t$
    - Small polynomial success probabilities are enough for the multistart strategie to obtain very good overall success probabilities

Introduction

Metropolis algorithms

- For the local random search heuristic, only multistart strategies are able to avoid the termination in local optima.
- A Metropolis approach allows to accept also new search points that decrease the fitness value
- If $F(x') < F(x)$ the search point x' is discarded only with probability

$$1 - \frac{1}{e^{(F(x)-F(x'))/T}}$$

Introduction

Metropolis algorithms

- The probability to accept search points with decreasing fitness value is dependent on the degree by which the fitness value is decreased.
- For $T \rightarrow 0$ the Metropolis approach becomes a random search
- For $T \rightarrow \infty$ the Metropolis approach becomes an uncontrolled local search
- The choice of T might have great impact on the performance of the algorithm.
- Knowledge on the problem or the fitness function might impact the choice of T

Introduction
Simulated Annealing

- Since the choice of an optimal value of T is not easy, it has been discussed to change the parameter in the pace of the optimisation
- At the beginning of the optimisation process, T should allow to 'jump' to other regions of the search space that have an increased fitness value
- When the simulation is nearing an end, the process should gradually 'freeze' until a local search approach will propagate the the local optimum in the neighbourhood.
- The name simulated Annealing is chosen since an analogy is seen to natural cooling processes of the creation of crystals
    - In this process, the temperature is gradually decreased so that Molecules that could move freely at the beginning are slowly put into their place

Introduction
Simulated Annealing

- One question regards the optimal choice of the cooling schedule for T
- Non-Adaptive approaches
  - Fixed temperature function $T(t)$
  - Every few steps the original value is multiplied with a factor $\alpha < 1$
- Adaptive approaches
  - React on the optimisation process
  - Probably dependent on the frequency of accepted iterations.

Introduction
Simulated Annealing

- The most crucial problem of the simulated Annealing approach is that to-date no natural problem is known, for which it has been proven that Simulated Annealing is sufficiently more effective than the Metropolis algorithm with the optium stationary temperature.
- However, artificially constructed problems exist, for which it could be shown that Simulated Annealing is superior to the Metropolis algorithm

Introduction

Tabu search

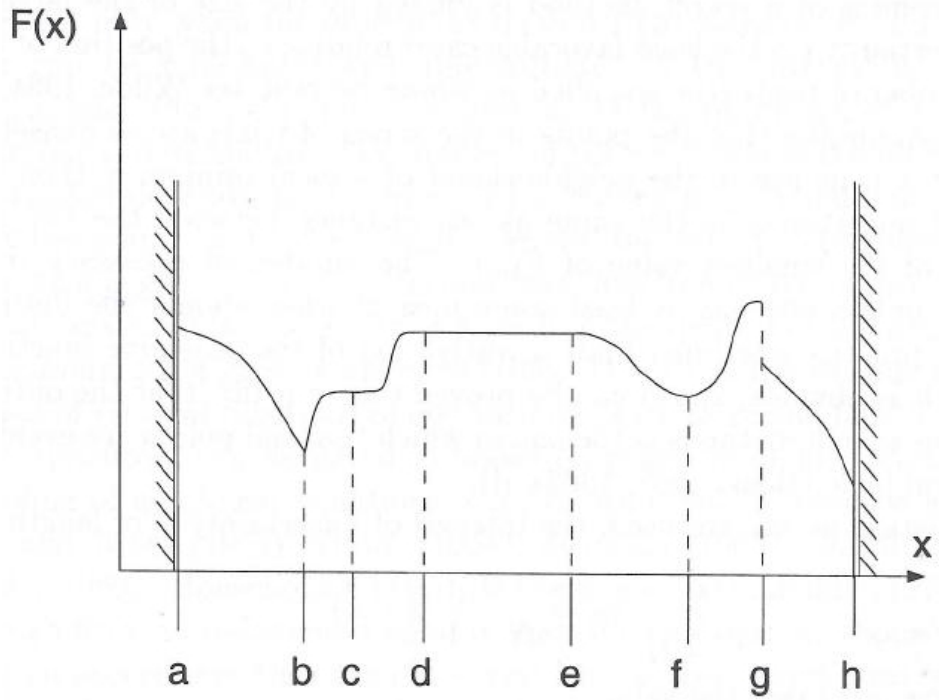
- The algorithms discussed so far only store the actual search point
- For Simulated Annealing and the Metropolis algorithm, also the search point with the best fitness value achieved so far is stored typically.
- However, knowledge about all other points is typically lost
- The algorithms might therefore access suboptimal points in the search space several times
- This increases the optimisation time

Introduction
Tabu search

- Tabu-search approaches also store a list of search points that have recently been accessed.
- Due to memory restrictions the list is typically of finite length
- When the size of the list is as least of the size of the neighbourhood $N(x)$ the method can terminate when the best point in the neighbourhood has been found.

One dimensional search strategies

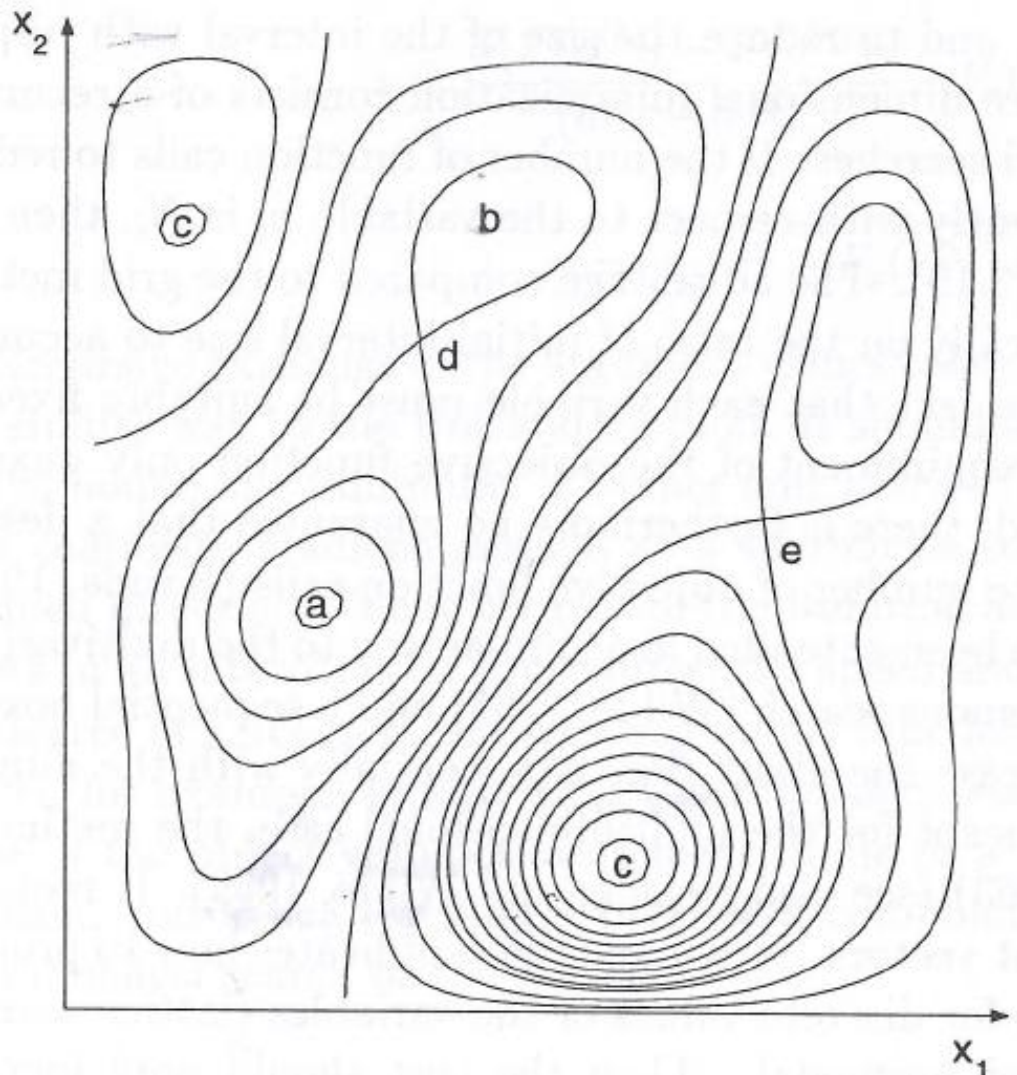
The one-dimensional search problem



- Local maxima/minima: a, b, d, e, f, g, h
- Saddle point: c
- Weak local maxima: d, e
- Global maximum: g

Multi dimensional search strategies

The multi-dimensional search problem



a: Global minimum

b: Local minimum

c: Local maxima

d,e: Saddle points

Multi dimensional search strategies

The multi-dimensional search problem

- The curse of dimensionality
  - When the dimension of the search space increases linary,
  - The number of possible solutions increases exponentionally.
  - A sequential program has therefore a WC-Runtime of $O(c^n)$
    - The constant c depends on the accuracy required

Multi dimensional search strategies

The multi-dimensional search problem

Pareto optimality

Let $\overrightarrow{x} = (x_1, \ldots, x_n)^T$ be a search point in a multi-dimensional search problem and $F_i : \mathbb{R} \rightarrow \mathbb{R}, \forall i$ the objective functions for the respective dimensions. A search point $\overrightarrow{x}$ is said to be pareto optimal with respect to a set of search points $\overrightarrow{x'} \in S$, if for at least one objective function F_i the equation $F_i(x_i) > F_i(x_i'), \forall x' \in S$ holds.

Black-box optimisation approaches

Introduction

- We speak of black-box optimisation, when the Genotype-Phenotype-Mapping is not known.
- However, a method to obtain Phenotype-outputs from Genotype-inputs (the black box) is available.
- The Algorithm then iteratively picks Genotype values, transforms them to Phenotype-outputs and evaluates the result with a scoring/fitness function

Outline

Prediction by randomised search approaches

1. Randomised seach approaches
2. Evolutionary approaches
3. Restrictions of evolutionary approaches
4. Design of evolutionary algorithms
5. Context prediction with evolutionary algorithms
6. Properties of evolutionary prediction approaches

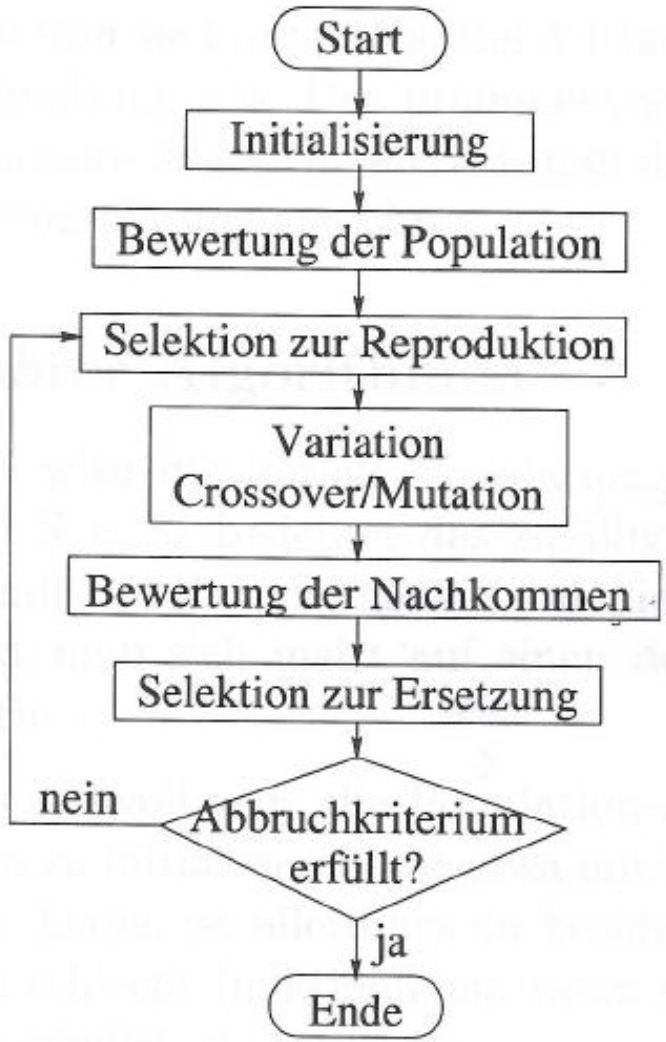
Evolutionary search approaches

Introduction

- Several researchers have studied the use of evolutionary approaches for optimisation purposes
- To-date, evolutionary algorithms combine these different approaches so that no clear distinction can be made
- An overview on various approaches is given in the following

Evolutionary search approaches

Introduction



Evolutionary search aproaches

Genetic algorithms

- Proposed by John Holland [1]
- Binary discrete search spaces: $\{0,1\}^n$
- Fitnessproportional selection
  - For m individuals $x_1, \ldots, x_m$ the probability to choose x_i is $\frac{f(x_i)}{f(x_1)+\cdots+f(x_m)}$.
- Main evolution operator is crossover
  - Originally One-point crossover
- The main goal was not optimisation but the adaptation of an environment

[1] J. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

Evolutionary search aproaches

Genetic algorithms

- One-point crossover
  - An individual x'' is composed of two individuals x and x' according to a uniformly determined crossover position:

$$x''_j = \begin{cases} x_j & \text{if } j \leq i \\ x'_j & \text{if } j > i \end{cases} \tag{1}$$

- K-point crossover
  - Choose k points in the individual strings and change source individual at these positions iteratively
- Uniform crossover
  - Similar to k-point crossover: Randomly decide for every bit position which bit will be adopted in the individual of the offspring population.

Evolutionary search aproaches

Genetic algorithms

- The hope associated with genetic algorithms was that they are able to solve some functions especially well

Separable function

A function is called separable, if the input variables can be divided into disjoint sets $X_1, \ldots, X_k$ with $f(x) = f_1(X_1) + \cdots + f_k(X_k)$

- Since genetic algorithms utilise crossover, it was expected that they are therefore well suited to quickly find the optimum on separable functions

Evolutionary search aproaches

Genetic algorithms

Royal road functions

For royal road functions, k blocks of variables of length l are formed. On each block X_l the identical function f_l is implemented with

$$f_l(X_l) = \begin{cases} 1 & \text{All variables in } X_l \text{ equal 1} \\ 0 & \textit{else}. \end{cases} \quad (2)$$

- It was shown that genetic algorithms do NOT perform well on these functions.[2]
- The reason is that it is highly unlikely to perform crossover exactly at the border of the variable blocks.
- It is better to optimise the single blocks on their own separately by mutation.

[2] T. Jansen and I. Wegener, *Real royal road functions – where crossover provably is essential*, Discrete applied mathematics, Vol. 149, Issue 1-3, 2005.

Evolutionary search aproaches

Evolution strategies

- Proposed by Bienert, Rechenberg and Schwefel[3] [4]
- At first only steady search spaces as $\mathbb{R}^n$
- No Crossover
- Only mutation
  - First mutation operator: Each component x_i is replaced by $x_i + \sigma Z_i$ (Z_i normally distributed, σ^2 Variance)

[3] I. Rechenberg, *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution, 1973.*

[4] H.P. Schwefel, *Evolution and optimum seeking*, 1993

Evolutionary search aproaches

Evolution strategies

1/5 rule

After $10n$ iterations, the variance is adopted every n iterations. When the number of accepted mutations in the last $10n$ steps is greater than $1/5$, σ is divided by 0.85 and else multiplied by 0.85.

- This heuristic is based on an analysis of the fitness function $x_1^2, \ldots, x_n^2$ – the sphere model.

Evolutionary search aproaches

Evolution strategies

Plus and comma strategies

A $(\mu + \lambda)$-strategie has a population size of μ and created λ individuals for the offspring population. The μ indivuduals of the offspring population consist of the best indivuduals from the old μ individuals and the created λ individuals.
In a (μ, λ)-strategie, the offspring population is created from the best of the λ created individuals exclusively.

- Comma-strategies try to avoid local optima

Evolutionary search aproaches

Evolutionary programming

- The approach was proposed by Lawrence J. Fogel[5][6]
- Various similarities to evolution strategies
- Search Space: Space of deterministic finite automata that well adapt to their environment.

[5] L.J. Fogel, *Autonomous automata*, Industrial Research, Vol. 4, 1962.

[6] L.J. Fogel *Biotechnology: Concepts and Applications*, Prentice-Hall, 1963

Evolutionary search aproaches

Genetic programming

- Proposed by John Koza[7]
- Search space: Syntactically correct programs
- Crossover more important than mutation

[7] John Koza *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992

Evolutionary search aproaches

Hybrid approaches

- Since evolutionary approaches are typically slow to initially find a search point with a reasonable fitness value,
- Approaches are combined with fast heuristics that initially search for a good starting point.
- Aferwards the evolutionary approach is applied

Outline

Prediction by randomised search approaches

1. Randomised seach approaches
2. Evolutionary approaches
3. Restrictions of evolutionary approaches
4. Design of evolutionary algorithms
5. Context prediction with evolutionary algorithms
6. Properties of evolutionary prediction approaches

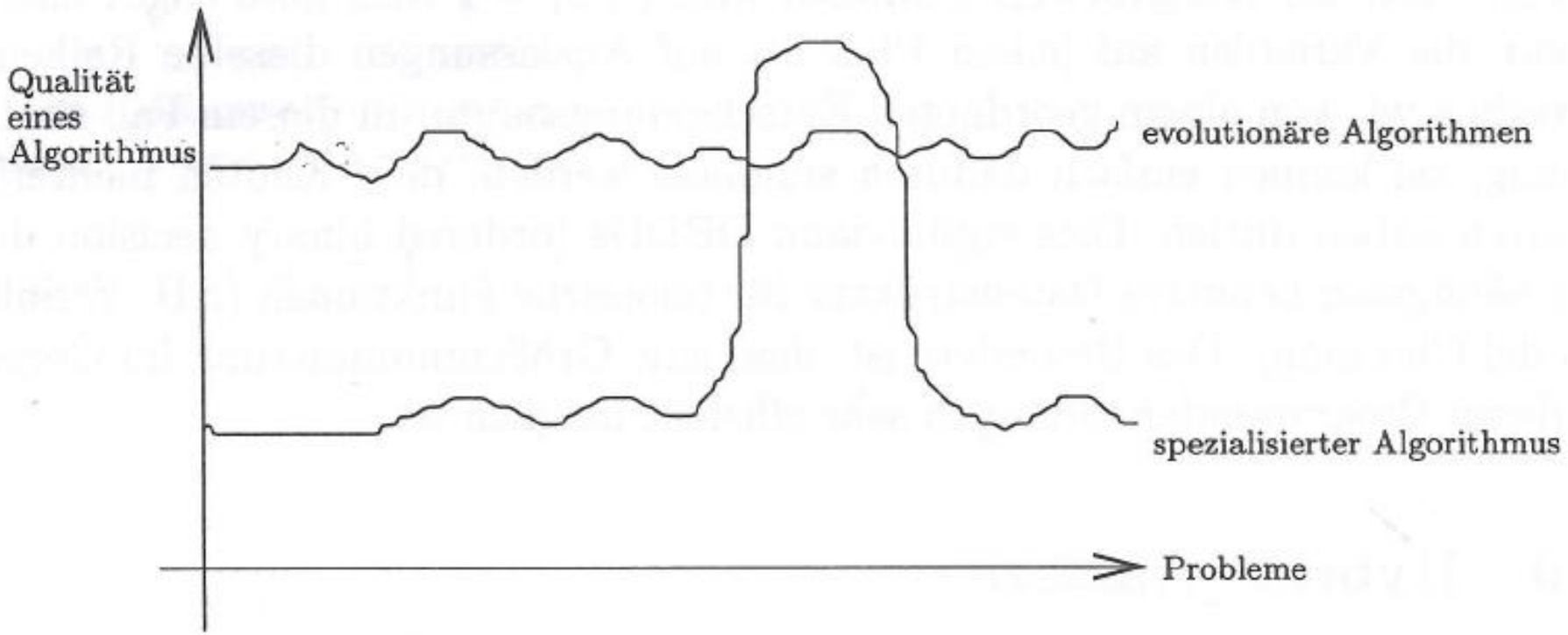
Restrictions of evolutionary approaches

The No-free-lunch theorem

- In the early days of evolutionary algorithm it has been argued that
  - Problem specific algorithms are better than evolutionary algorithms on a very small subset of problems
  - Evolutionary algorithms perform better on average over all problems
- Therefore, evolutionary algorithms have been proposed as a good choice for a general purpose optimisation scheme

Restrictions of evolutionary approaches

The No-free-lunch theorem



Restrictions of evolutionary approaches

The No-free-lunch theorem

- We will shortly discuss this figure
  - Can an algorithm be suited for 'all' problems?
    - Distinct coding of the search space
    - Various fitness functions
  - What does 'all problems' mean?
    - For all possible representations and sizes of the search space
    - All possible fitness functions on the feature space
    - For a given search space and feature space, all possible fitness functions
    - Every single point in the search space is the optimum point in several of these problems
  - Can one algorithm be better on average than another algorithm on 'all' problems?

Restrictions of evolutionary approaches

The No-free-lunch theorem

- To understand this scenario, Wolpert and Macready formalised the assertion[8]
- Assumptions:
  - The set of all functions $f : S \longrightarrow W$ considered is given by F
  - S and W are finite (as every computation on physical computers can only have finite resources)
  - The fitness function is evaluated only once for each search point
  - $A(f)$ is the number of search points requested until the optimum is found

[8] D.H. Wolpert and W.G. Macready, *No Free Lunch Theorems for Optimization*, IEEE Transactions on Evolutionary Computation 1, 67, 1997.

Restrictions of evolutionary approaches

The No-free-lunch theorem

No free lunch theorem

Assume that the average performance of an algorithm in the No Free Lunch Scenario for S and W is given by $A_{S,W}$, the average over all $A(f), f \in F$ Given two algorithms A and A', we obtain $A_{S,W} = A'_{S,W}$

- This means that two arbitrary algorithms perform equally well on average on all problems

Restrictions of evolutionary approaches

The No-free-lunch theorem

Proof of the No Free Lunch Theorem

Proof by induction over $s := |S|$.
W.l.o.g.: $W = \{1, \ldots, N\}$
We consider sets $F_{s,i,N}$ of all functions f on a search space of non-visited search points of size s with at least one x with $f(x) > i$
Observe that for every function f and every permutation π also f_π belongs to $F_{s,i,N}$

Restrictions of evolutionary approaches

The No-free-lunch theorem

Proof of the No Free Lunch Theorem

Induction start: $s = 1$

Every algorithm, since it does not repeat any requests, has to choose the single search point and thus find the optimum point.

Restrictions of evolutionary approaches

The No-free-lunch theorem

Proof of the No Free Lunch Theorem

Induction: $s - 1 \rightarrow s$

We define a function $a : S \rightarrow \mathbb{N}$ so that for every $x \in S$ the share of functions with $f(x) = j$ is exactly $a(j)$.

This is independent of x, since all permutations f_π of a function f also belong to $F_{s,i,N}$,

$a(j)$ is therefore the probability to choose a search point with fitness value j – Independent of the concrete algorithm A

Restrictions of evolutionary approaches

The No-free-lunch theorem

Proof of the No Free Lunch Theorem

Induction: $s - 1 \rightarrow s$

With probability $a(j)$ an algorithm A finds a search point with fitness value j.

If $j > i$, the number of functions $f(x) = j$ is equal to the number of functions $f_\pi(y) = j$, since all permutations of f are also in $F_{s,i,N}$.

The probability to achieve a fitness value $j > i$ is therefore independent of the algorithm.

Restrictions of evolutionary approaches

The No-free-lunch theorem

Proof of the No Free Lunch Theorem

Induction: $s - 1 \rightarrow s$

With probability $a(j)$ an algorithm A finds a search point with fitness value j.

If $j \leq i$, x is not optimal in scenario $F_{s,i,N}$ and the new scenario is $F_{s-1,i,N}$

Restrictions of evolutionary approaches

The No-free-lunch theorem

Proof of the No Free Lunch Theorem

Summary – in other words:
For any two algorithms we can state a suitable permutation of the Problem-function for one problem (i.e. state another problem), so that both algorithms in each iteration request identical search points.

- Especially, since every search point could be optimal, there are always algorithms that request the optimal search point right from the start.

Restrictions of evolutionary approaches

An almost-no-free-lunch-theorem

- The NFL is possible, since ALL algorithms and ALL problems are considered
- It is a reasonable question if an NFL is also valid in smaller, more realistic scenarios.
- In [9] is was prooved, that a similar theorem can be stated also for more realistic problem scenarios.

[9] S. Droste, T. Jansen and I. Wegener, *Perhaps not a free lunch but at least a free appetizer*, Proceedings of the 1st Genetic and Evolutionary Computation Conference, 1999.

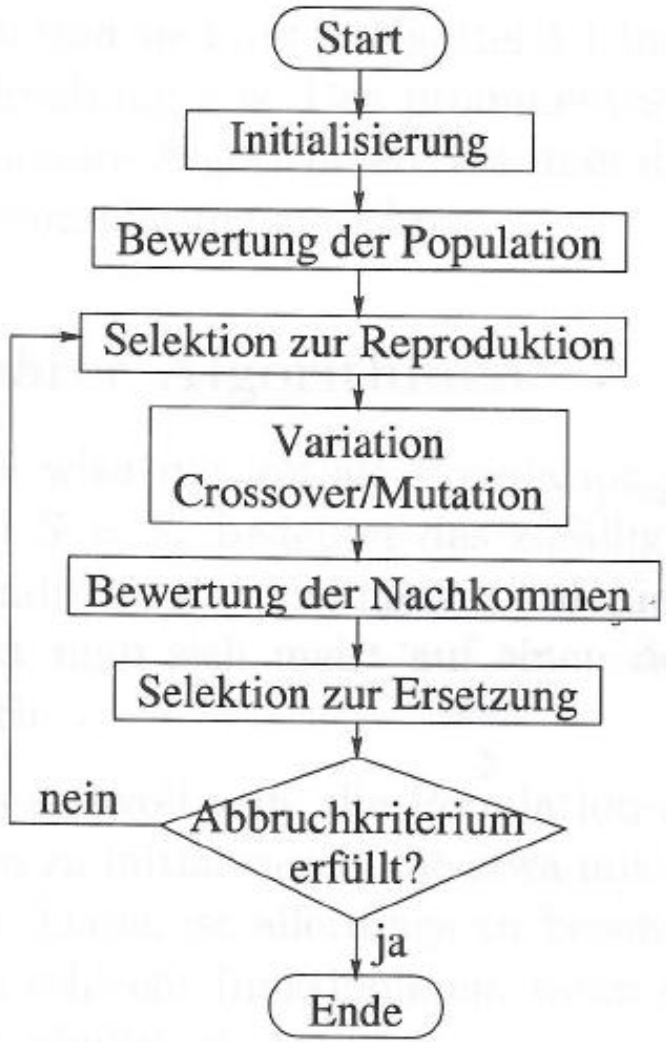
Outline

Prediction by randomised search approaches

1. Randomised seach approaches
2. Evolutionary approaches
3. Restrictions of evolutionary approaches
4. Design of evolutionary algorithms
5. Context prediction with evolutionary algorithms
6. Properties of evolutionary prediction approaches

Design of evolutionary algorithms

Overview



Design of evolutionary algorithms

Search space

- When designing an algorithm for a given problem, the design of the search space has great impact on the performance of the algorithm
- We have to decide which parameters impact the fitness by what amount.
- Parameters might depend on each other so that not all have to be modelled.

Design of evolutionary algorithms

Search space

- It is often natural to represent search points as vectors of finite length
  - With components of the same set $(\mathbb{R}, \mathbb{Z}, \mathbb{N}, \{0, 1\})$
  - This leads to search spaces of the type $S = X^n$
  - Also vectors with components of distinct type possible (multi-type)
- Mutation and crossover operators have to respect these properties of the search space.
- Mutation and crossover often assume that neighbouring search points are related to each other.
- It is important to choose a representation that well reflects the characteristics of the problem at hand.

Design of evolutionary algorithms

Search space

Hamming cliff

- The hamming distance between 2^n and $2^n + 1$ is 1
- The hamming distance between 2^n and $2^n - 1$ is $n + 1$!!!

- A possible solution are Gray Codes
- The hamming distance between neighbouring numbers is always one

Design of evolutionary algorithms

Search space

Gray codes

- For the numbers 0 and 1, the representation is 0 and 1
- When $0, \ldots, 2^n - 1$ are correctly represented by the bitvectors $a_0, \ldots a_{N-1}$ with $N = 2^n$
  - Represent $0, \ldots, 2^{n+1} - 1$ by $0a_0, \ldots, 0a_{N-1}, 1a_{N-1}, \ldots, 1a_0$
- The hamming distance of neighbouring numbers is then 1

- The drawback of this approach is that numbers with greater numerical distance have also to distance 1
- $0a_0$ and $1a_0$ also have hamming distance 1

Design of evolutionary algorithms

Selection principles

- Selection principles decide which individuals are the basis for the next generation.
- The selection is based on the fitness function
- Often: Survival of the fittest

Design of evolutionary algorithms

Selection principles

- Selection strategies
  - Try to optimise the overall fitness of individuals
    - Assume: Individuals with similar fitness values are neighbours in the search space
  - Try to prevail diversity in the search space
- Both strategies are contradictory

Design of evolutionary algorithms

Selection principles

Fitnessproportional selection

- For a population $x_i, \ldots, x_n$ the individual x_i is chosen with probability

$$p(x_i) = \frac{f(x_i)}{f(x_1) + \cdots + f(x_n)} \tag{3}$$

- For the selection a random variable u is drawn from $[0, 1]$ and x_i is considered if

$$p(x_1) + \cdots + p(x_{i-1}) < u \leq p(x_1) + \cdots + p(x_i) \tag{4}$$

- Frequently applied for evolutionary approaches
- Unfortunately, linear modification of the fitness function $(f \longrightarrow f + c)$ results in different behaviour

Design of evolutionary algorithms

Selection principles

SUS – Stochastic Universal Sampling

- The λ individuals are drawn from a uniformly distributed random variable u from $[0, 1/\lambda)$
- A control variale s is initially set to $p(x_i)$ with $i = 1$
- When $u < s$, an Individual of type x_i is considered for the next population and u is increased by $1/\lambda$
- When $u \geq s$, s is increased by $p(x_{i+1})$ and i is increased by one.

- SUS is a selection mechanism especially proposed for evolutionary algorithms
- λ candidates for the offspring population are created in the manner described

Design of evolutionary algorithms

Selection principles

- As some selection approaches have problems with the scaling of the fitness function (e.g. fitness proportional selection), it is possible to only consider an fitness-based ordering of search points.
- The $(\mu + \lambda)$ and (μ, λ) strategies fall into this category.
- Another example: Threshold selection
  - The candidates for the offspring population are drawn uniformly at random from the t highest rated individuals

Design of evolutionary algorithms

Selection principles

Tournament selection

- A tournament size of $q \in \{1, \ldots, n\}$ is defined.
- A set of q individuals is then drawn uniformly at random from the population
- The best individual from this set is considered for the offspring population.

- For $q = 1$ the tournament selection is a random selection
- For $q = n$ it implements a deterministic choice

Design of evolutionary algorithms

Selection principles

- Lifetime of individuals
  - Some strategies define a maximum lifetime of individuals
  - An individual is then replaced when its maximum lifetime is reached
- Most approaches implement unlimited lifetime
- For comma strategies the lifetime is 1 for every individual

Design of evolutionary algorithms

Selection principles

- Since a great number of distinct selection strategies exists, a quality measure for selection strategies is desired.

Design of evolutionary algorithms

Selection principles

Quality measure – Takeover time

The takeover time is the count of generations until an algorithm that exclusively relies on selection (no mutation or crossover) has replaced all indivituals in the population by the best individual

- Very short or very long takeover times are not good
- Algorithm will then either not converge or converge in local optima
- But even when the takeover time is known it is still not clear how to interprete the data

Design of evolutionary algorithms

Selection principles

Quality measure – Selection intensity

To calculate selection intensity, the varinace σ^2 of the fitness values in the population and the mean fitness value is measured before ($\overline{f}$) and after ($\overline{f_{sel}}$) the selection.
The selection intensity is then defined as

$$I = \frac{(\overline{f_{sel}} - \overline{f})}{\sigma} \tag{5}$$

- Measure depends on the variance of the fitness values
- Variance of fitness values dependent on selection method
  - Quality measure therefore depends on selection method that is to be quantified.
- Interpretation of this measure is therefore not trivial

Design of evolutionary algorithms
Mutation

- A mutation creates one new individual from one given individual
- Mutation operators are designed for specific search spaces
- Mutation shall apply only few modifications of individuals on average
- Individuals that are closer to the original individual (regarding the neighbourhood function) shall have a greater probability than those that are farther away

Design of evolutionary algorithms

Mutation

- Search spaces in $\{0,1\}^n$
  - The common mutation operator chooses a mutation probability p and flips every bit with this probability
  - To obtain a search point with hamming distance i the probability is $p^i(1-p)^{n-i}$
  - $p = \frac{1}{2}$ is random search
  - To assure that individuals that are farther away have decreased probability to be constructed, $p \leq \frac{1}{2}$
  - The expectation on the number of bits mutated is np and the variance is $np(1-p)$
  - It is very unlikely to obtain an individual that is far away in the search space
  - A standard choice is $p = \frac{1}{n}$

Design of evolutionary algorithms

Mutation

- Search spaces $A_1 \times \cdots \times A_n$
  - A similar approach as for $\{0, 1\}$ search spaces can be taken
  - With probability p one of $|A_i|$ possible values is taken uniformly at random for position i
  - The probability that position i is not mutated is therefore

$$(1 - p) + p \cdot \frac{1}{|A_i|} \tag{6}$$

Design of evolutionary algorithms
Mutation

- Search space $\mathbb{R}^n$
  - For mutation purposes, a probability vector is typically added to the actual search point
  - The expectation of the vector should be 0 so that no direction is prefered

Design of evolutionary algorithms

Mutation

- Permutations on the search space
  - Example: TSP – k-opting
    - Order of places is unraveled at k positions
    - These k blocks are then again connected randomly
  - Another approach is to change the order of nodes in some blocks

Design of evolutionary algorithms
Mutation

- Mutations of syntax trees (Genetic programming)
  - One of four possible mutation operators is chosen uniformly at random

Grow	Choose a leaf and replace this by random syntax tree
Shrink	Choose an inner node and replace this by a leaf with random value
Switch	Choose random inner node and exchange the position of two randomly chosen children
Cycle	Choose a node at random and change its labelling/value

  - It has to be taken care that the resulting syntax tree remains syntactically correct

Design of evolutionary algorithms

Recombination

- Recombination typically takes two individuals and results in one or two offspring individuals
  - Also recombination of more than two individuals possible
  - Often generalisations of the two-individual case
- Distinct recombination methods for various search spaces
- Crossover parameter p_c specifies the probability with which crossover (and not mutation) is applied for one selected individual
- In some cases (e.g. binary coded numbers) not all positions in the individual string are allowed to apply crossover on

Design of evolutionary algorithms

Recombination in $\{0,1\}^n$

- One-point crossover
- k-point crossover
- Uniform crossover

Design of evolutionary algorithms

Recombination in $\{0,1\}^n$

Shuffle crossover

- Parent-individuals are randomly permutated with π
- Crossover operation is applied
- Resulting individuals are re-permutated with π^{-1}

- For shuffle crossover, neighbouring bits have not a higher probability to have their origin in the same parent individual

Design of evolutionary algorithms

Recombination in $\{0,1\}^n$

Random respectful recombination

- All information that is identical in both parent individuals is copied to the child-individual
- For all other positions, the value is chosen uniformly at rondom

Design of evolutionary algorithms

Recombination in $\mathbb{R}^n$

- Recombination in $\mathbb{R}^n$
  - The same recombination approaches can be applied as in $\{0,1\}^n$

Design of evolutionary algorithms

Recombination in $\mathbb{R}^n$

Alternative recombination approaches in $\mathbb{R}^n$

- When parent individuals have values x_i and y_i at position i
- We can choose position i for the child as

$$x_i + \mu_i(y_i - x_i) \tag{7}$$

- μ_i is drawn uniformly at random from $[0, 1]$

Design of evolutionary algorithms

Recombination for permutations

Order crossover

- Variant of two-point crossover that is suitable for permutations
- Values between both crossover positions are taken from the first individual
- All missing values are filled in the order they occured in the second individual (beginning from the second crossover position)

Parent 1	12	3456	789
Parent 2	84	1593	627
Child	??	3456	???
Child	19	3456	278

Design of evolutionary algorithms

Recombination for permutations

Partially maped crossover (PMX)

- Variant of two-point crossover that is suitable for permutations
- Values between both crossover positions are taken from the first individual
- Missing values are inclued at the same position the value is found in the second individual.
- If this position is already occupied by value x_i, the position of individual x_i is chosen instead (and so on)

Parent 1	12	3456	789
Parent 2	84	1593	627
Child	??	3456	???

Child	89	3456	127

Design of evolutionary algorithms

Recombination for permutations

Order crossover II

- k positions are randomly marked
- All other positions are taken over from the second parent in their occurence order

- Assume that the positions 2,4,6,8 are marked.

Parent 1	12	3456	789
Parent 2	8$\underline{4}$	1$\underline{5}$9$\underline{3}$	6$\underline{2}$7
Child	82	1394	657

Design of evolutionary algorithms

Structures of populations

- The structure of the population has also an impact on the performance of the algorithm
  - Consideration of duplicate individuals
  - Diversity

Design of evolutionary algorithms

Structures of populations

- Creation of niche in the population
  - In order to keep isolated individuals with respectable fitness value
  - The number of individuals in the neighbourhood is also considered for the fitness-based selection

$$f'(x) = \frac{f(x)}{d(x, P)} \tag{8}$$

Design of evolutionary algorithms

Structures of populations

- Consideration of sub-populations
  - Similar individuals are grouped together for optimisation
  - Recombination not over the whole population but between individuals of a sub population
  - Idea:
    - Individuals of distinct sub-populations have good fitness.
    - By crossover operation, an individual inbetween is created that has typically worse fitness value
  - Selection applied on the overall population

Design of evolutionary algorithms

Dynamic and adaptive approaches

- As parameter choices impact the performance of an evolutionary algorithm, adaptation of parameters during simulation might also be beneficial
- Similar approach as for the 'mutation' probability of simulated annealing
- Feasible also for Crossover, mutation, fitness function, population structure

Design of evolutionary algorithms

Comments on the implementation of evolutionary algorithms

- Evolutionary algorithms are easy to implement when compared to some complex specialised approaches
- However, Evolutionary algorithms are computationally complex
- It is therefore beneficial to implement efficient variants to the distinct methods

Design of evolutionary algorithms

comments on the implementation of evolutionary algorithms

- Generation of pseudo random bits is important for many of the theoretic results for evolutionary algorithms to hold
- It is, however possible to reduce the number of random experiments
  - It is more efficient to calculate the next flipping bit in a mutation instead of doing the calculation for every bit independently

Design of evolutionary algorithms

comments on the implementation of evolutionary algorithms

- Most of the computational time is typically consumed by the fitness calculation
- One approach to reduce complexity is to prevent re-calcuation of fitness for individuals
  - Dynamic data structures that support search and insert

Outline

Prediction by randomised search approaches

1. Randomised seach approaches
2. Evolutionary approaches
3. Restrictions of evolutionary approaches
4. Design of evolutionary algorithms
5. Context prediction with evolutionary algorithms
6. Properties of evolutionary prediction approaches

Context prediction with evolutionary algorithms

Prediction approach

- Context prediction is a search problem:
- Given the input sequence, the continuation of this sequence is requested.
- A search algorithm operates therefore on the search space of string sequences
- The output of the algorithm is again a string sequence mapping

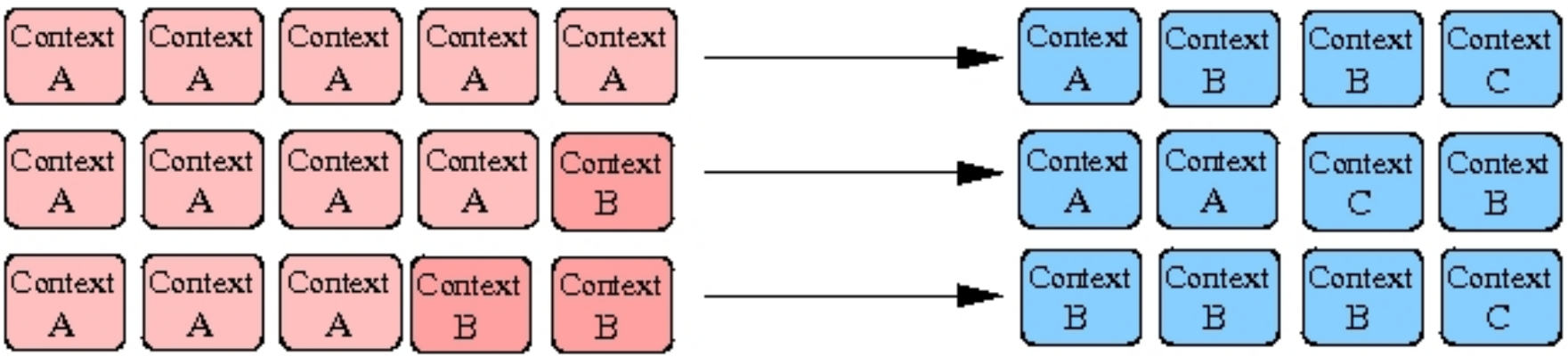
Context prediction with evolutionary algorithms

Prediction approach

- In context prediction the task of the evolutionary algorithm is to find a suitable mapping from input sequences to output sequences
- The search approach adapts to a changing environment by adapting the input-output mapping
- With the mapping computed by the algorithm, a prediction is stated
- The fitness value for a mapping provided is determined by the accuracy of the prediction

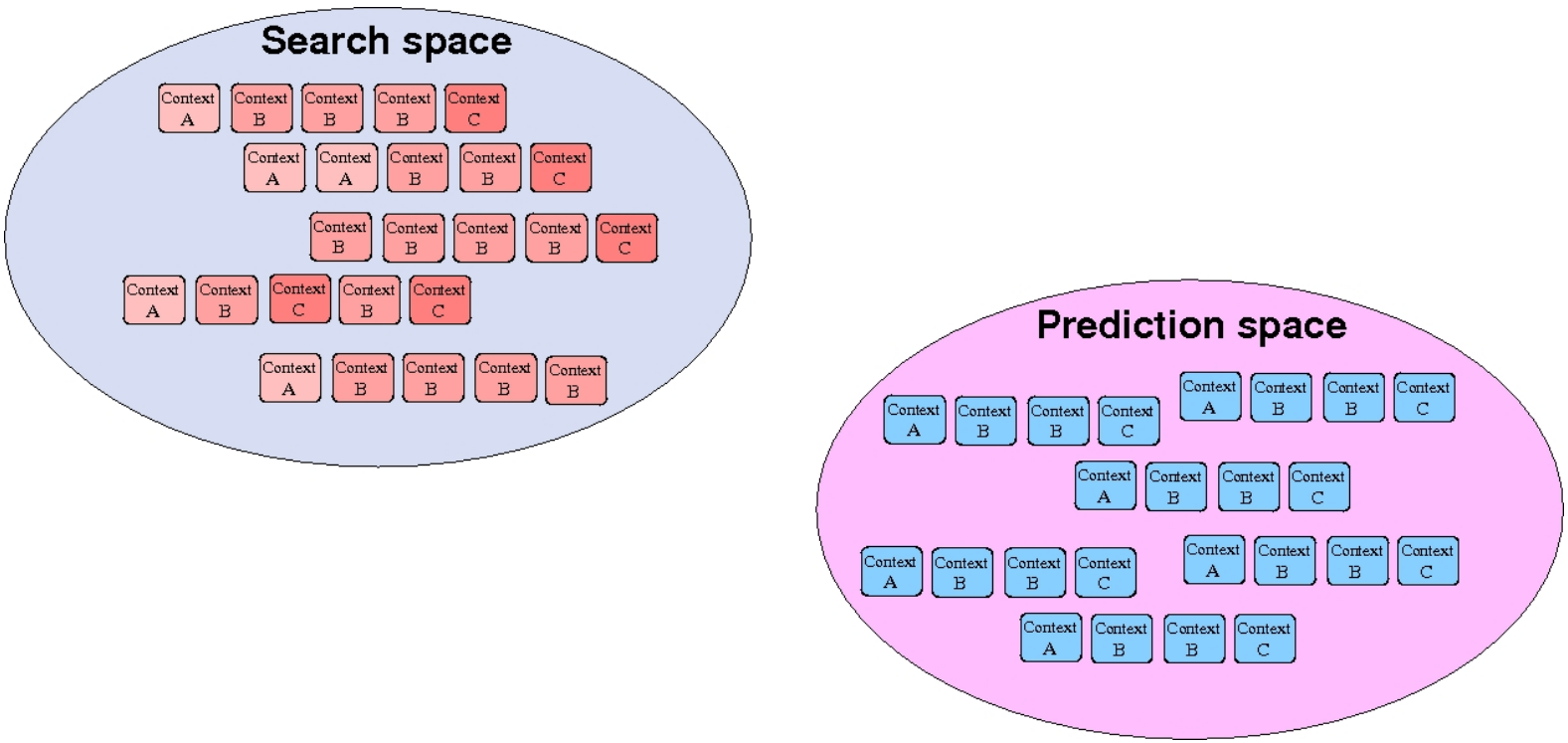
Context prediction with evolutionary algorithms

Prediction approach



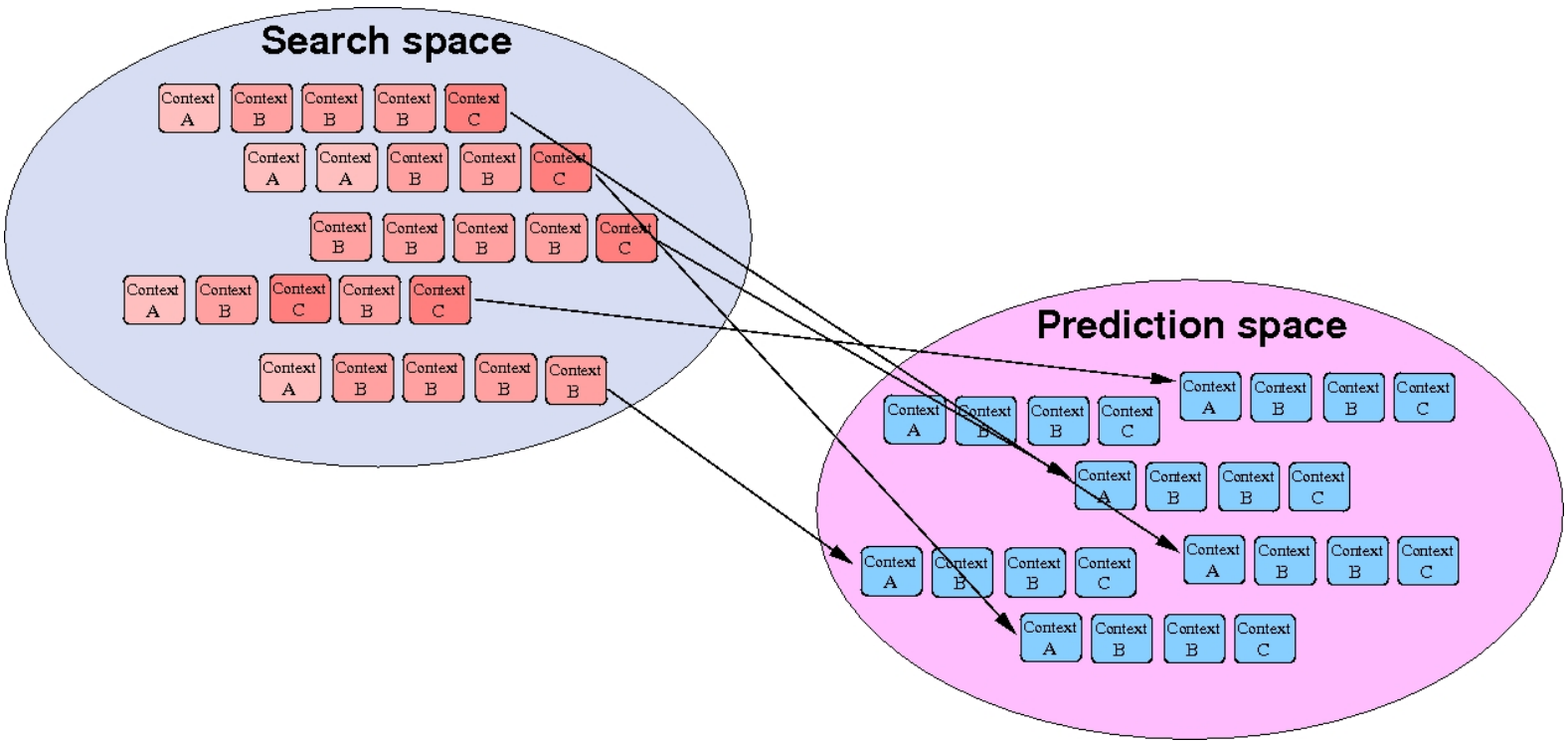
Context prediction with evolutionary algorithms

Prediction approach



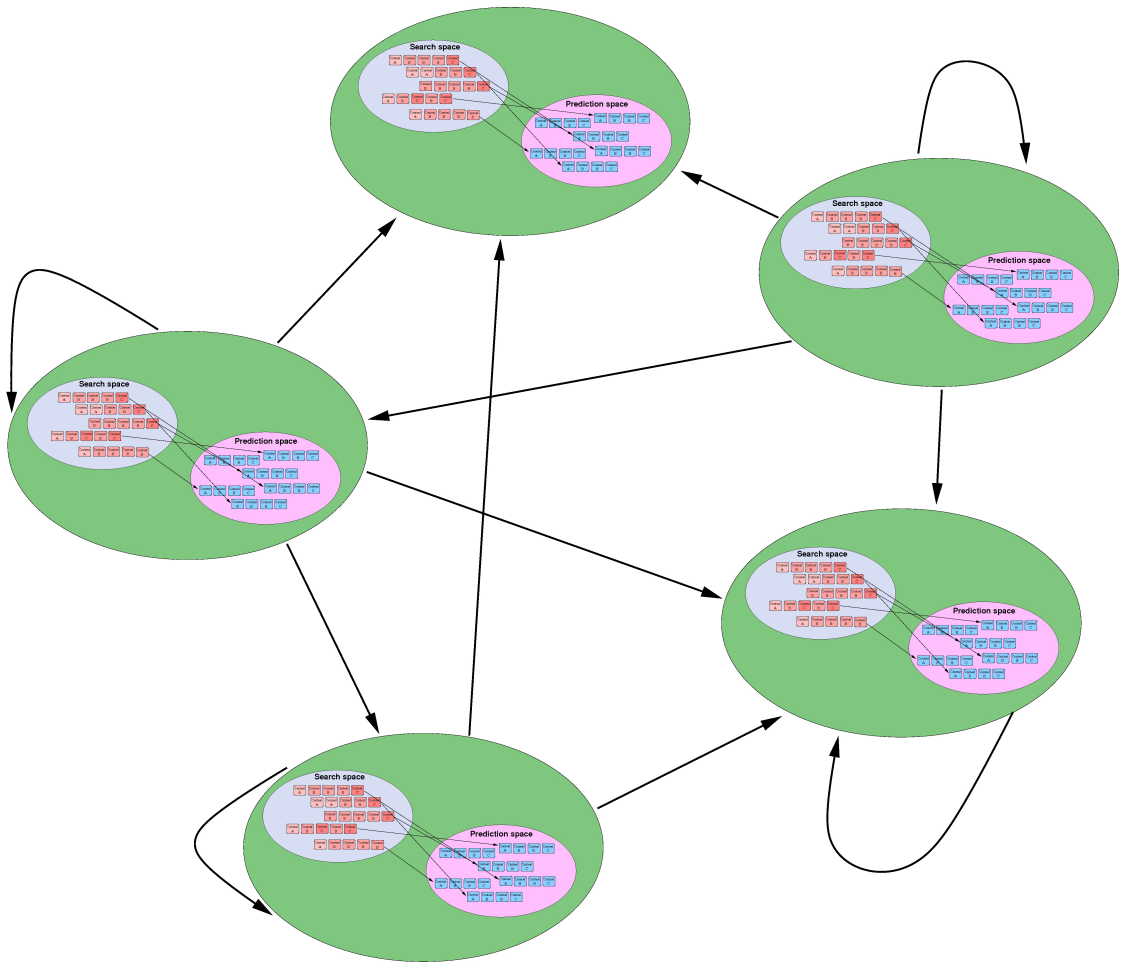
Context prediction with evolutionary algorithms

Prediction approach



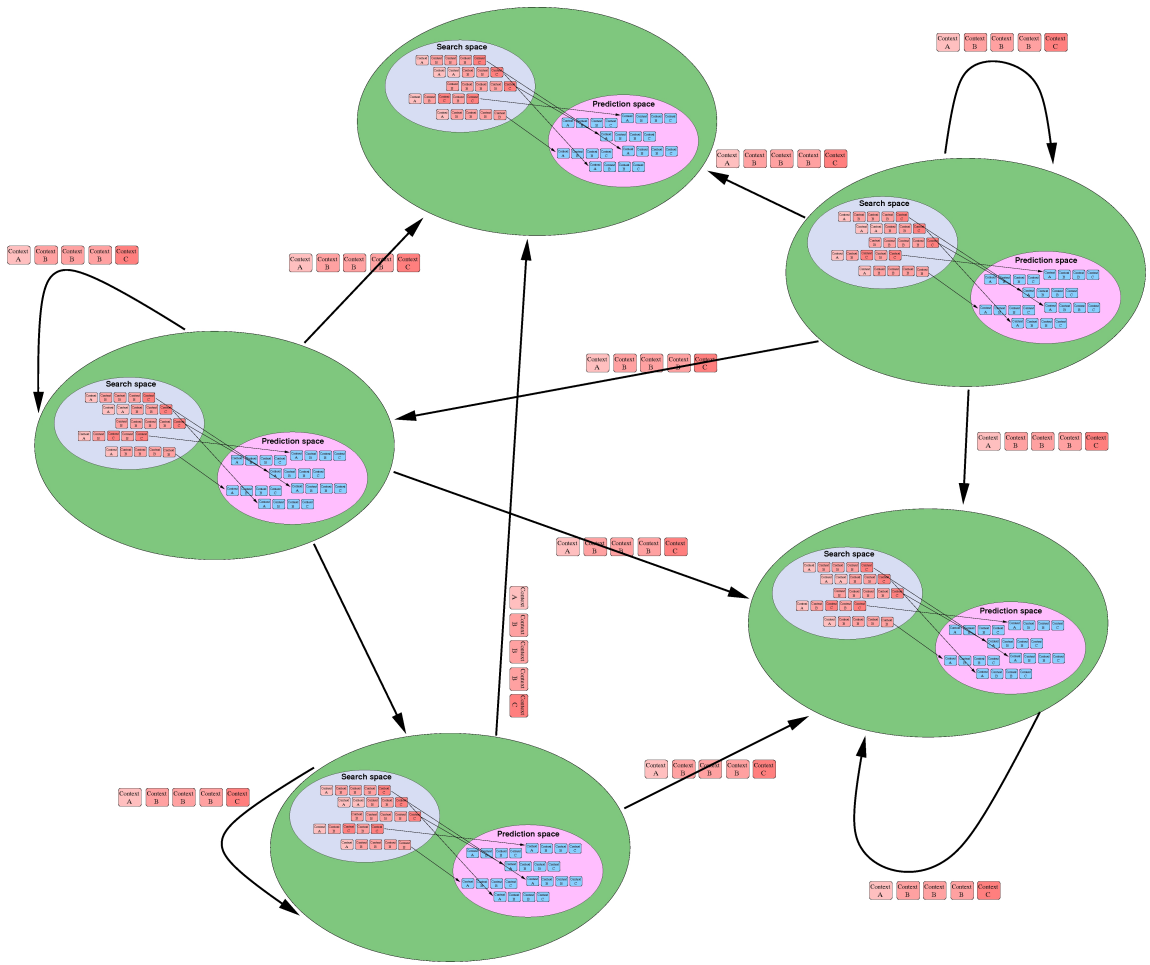
Context prediction with evolutionary algorithms

Prediction approach



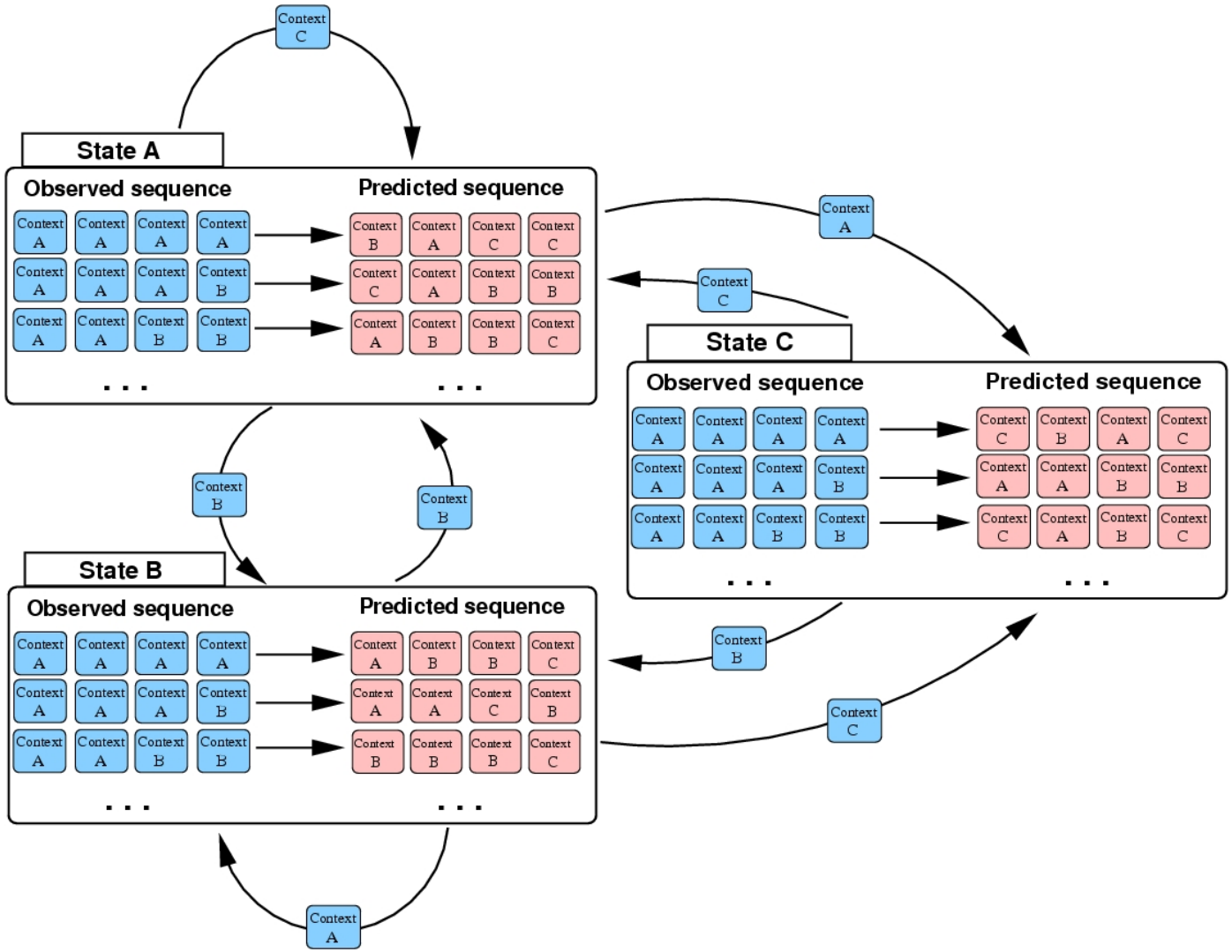
Context prediction with evolutionary algorithms

Prediction approach



Context prediction with evolutionary algorithms

Prediction approach



Outline

Prediction by randomised search approaches

1. Randomised seach approaches
2. Evolutionary approaches
3. Restrictions of evolutionary approaches
4. Design of evolutionary algorithms
5. Context prediction with evolutionary algorithms
6. Properties of evolutionary prediction approaches

Properties of evolutionary prediction approaches

Processing load

- Prediction possible as table-look-up in time $O(log(k))$
- Pre-processing in order to obtain a suitable mapping, however, very expensive.
- Pre-processing time dependent on the problem structure and the algorithmic modelling
  - Search space (binary, real-valued, non-numeric)
  - Crossover and/or mutation
  - ...

Properties of evolutionary prediction approaches

Memory requirements

- Memory requirements
  - Mapping from search-space to prediction space is to be stored
  - $O(k^{|C|})$

Properties of evolutionary prediction approaches

Prediction horizon

- Arbitrary but restricted by the lenght of typical patterns
- No iterative prediction possible.
- Compare: Alignment approach

Properties of evolutionary prediction approaches

Adaptability

- Highly adaptable, since the adaptation is the main task of the search approach
- Also on-line aproach and continuous learning feasible

Properties of evolutionary prediction approaches

Multi-dimensional time series

- Multidimensional time series are handled as any other time series

Properties of evolutionary prediction approaches

Iterative prediction

- Iterative Prediction not feasible since it would exceed the typical pattern length

Properties of evolutionary prediction approaches

Prediction of context durations

- Possible when context duration implicitly stated by sample frequency

Properties of evolutionary prediction approaches

Approximate matching of patterns

- Approximate matching
  - Supported, since various patterns in the search space might be mapped to one individual in the prediction space

Properties of evolutionary prediction approaches

Context data types

- Arbitrary type of context patterns supported

Properties of evolutionary prediction approaches

Pre-processing

- Pre-processing required to achieve mapping of sufficient prediction accuracy
- Changing environments: On-line approach feasible.
- Pre-processing time dependent on the problem structure and the algorithmic modelling
  - Search space (binary, real-valued, non-numeric)
  - Crossover and/or mutation
  - ...

Aspects of prediction algorithms

Summary

	IPAM	ONISI	Markov	CRF
Numeric Contexts	yes	no	no	no
Non-numeric Contexts	yes	yes	yes	yes
Complexity	$O(k)$	$O(k^2)$	$O(C^2)$	$O(C^2)$
Learning ability	(no)	yes	yes	yes
Approximate matching	no	no	no	no
Multi-dim. TS	(no)	(no)	(no)	(no)
Discrete data	yes	yes	yes	yes
Variable length patterns	no	yes	no	(yes)
Multi-type TS	yes	no	(no)	(no)
Continuous data	no	no	no	no
Pre-processing	$O(k)$	–	$O(k)$	$O(k)$
Context durations	no	no	no	no
Continuous time	no	no	yes	yes

Aspects of prediction algorithms

Summary

	SPM	Align	SOM	EA
Numeric Contexts	yes	yes	yes	yes
Non-numeric Contexts	yes	yes	yes	yes
Complexity	$O(1)$	$O(I \cdot k^2)$	$O(\log(\|k\|))$	$O(\log(\|k\|))$
Learning ability	(yes)	yes	yes	yes
Approximate matching	no	yes	yes	yes
Multi-dim. TS	(no)	yes	yes	yes
Discrete data	yes	yes	yes	yes
Variable length patterns	yes	yes	yes	yes
Multi-type TS	no	yes	yes	yes
Continuous data	no	no	no	no
Pre-processing	$O(k)$	$O(k^2)$	$O(k^2)$	–[10]
Context durations	no	yes	yes	yes
Continuous time	no	no	no	no

[10] Preprocessing complexity dependent on problem definition and algorithmic modelling. Presumably high preprocessing required for random search approach

Properties of evolutionary prediction approaches

Conclusion

- Benefits
  - Automatic adaptation
  - Arbitrary time series feasible (length, dimension, context types)
  - Very flexible approach
- Drawbacks
  - Very high complexity for pre-processing