A Convergent Multi-Objective Evolutionary Algorithm with a Local Mutation Operator

Louis-Claude CANON – Emmanuel JEANNOT

Project-Team Runtime
Labri/INRIA/Nancy Université
Bordeaux – France

Workshop on Evolutionary Algorithms

March 29, 2010
1. Convergence with Local Mutation
2. MOEA
3. Scheduling
4. Conclusion
Convergence with Local Mutation

Mutation Operator & Neighborhood

Search space
**Definition**

Markovian kernel $K$: $K(x, A) = \Pr\{X_{t+1} \in A \mid X_t = x\}$

$A$ is a set of states and $x$ is the state of the markovian chain $X$ at step $t$. 

Theorem (Rudolph, 1996)

An elitist evolutionary algorithm ($K(e, A) = 1, \forall x \in A$) with a global mutation operator ($K_m(x, A) > 0, \forall x \in A$) converges towards the set of optimal solutions.
**Definition**

Markovian kernel $K$: $K(x, A) = \Pr\{X_{t+1} \in A \mid X_t = x\}$

$A$ is a set of states and $x$ is the state of the markovian chain $X$ at step $t$.

**Theorem (Rudolph, 1996)**

*An elitist evolutionary algorithm $(K(x, A_\epsilon) = 1, \forall x \in A_\epsilon)$ with a global mutation operator $(K_m(x, A_\epsilon) > 0 \forall x \in A)$ converges towards the set of optimal solutions.*
Local/Global Equivalence

Notations

- $K_c$: crossover operator
- $K_m$: mutation operator
- $K_s$: selection operator
- $(K_c K_m K_s)$: EA iteration

Theorem (Canon-Jeannot, 2010)

Let $K_c(x, \{x\}) \geq \delta_c$ and $K_s(x, \{x\}) \geq \delta_s$ for each $x \in A$. Then,

$$(K_c K_m K_s)(M)(x, A) \geq (\delta_c \delta_s) M K_m(x, A).$$

Implication

An elitist EA converges if:

crossover and selection are not deterministic ($\delta_c > 0$ and $\delta_s > 0$)
local become global after $M$ iterations:
$K(M)m(x, A)$ is global
Convergence with Local Mutation

Local/Global Equivalence

Notations

- $K_c$: crossover operator
- $K_m$: mutation operator
- $K_s$: selection operator
- $(K_c K_m K_s)$: EA iteration

Theorem (Canon-Jeannot, 2010)

Let $K_c(x, \{x\}) \geq \delta_c$ and $K_s(x, \{x\}) \geq \delta_s$ for each $x \in A$. Then, $(K_c K_m K_s)^{(M)} (x, A) \geq (\delta_c \delta_s)^M K_m^{(M)} (x, A)$. 

Implication

An elitist EA converges if: crossover and selection are not deterministic ($\delta_c > 0$ and $\delta_s > 0$) local become global after $M$ iterations: $K_m^{(M)} (x, A)$ is global.
Convergence with Local Mutation

Local/Global Equivalence

Notations
- $K_c$: crossover operator
- $K_m$: mutation operator
- $K_s$: selection operator
- $(K_c K_m K_s)$: EA iteration

Theorem (Canon-Jeannot, 2010)

Let $K_c(x, \{x\}) \geq \delta_c$ and $K_s(x, \{x\}) \geq \delta_s$ for each $x \in A$. Then,

$$(K_c K_m K_s)^{(M)}(x, A) \geq (\delta_c \delta_s)^M K_m^{(M)}(x, A).$$

Implication

An elitist EA converges if:
- crossover and selection are not deterministics ($\delta_c > 0$ and $\delta_s > 0$)
- local become global after $M$ iterations: $K_m^{(M)}(x, A)$ is global
Outline

1. Convergence with Local Mutation
2. MOEA
3. Scheduling
4. Conclusion
In multiobjective optimization, there may not be a single optimal solution but non-dominated solutions.
Classic approaches

- **PESA-II** [Corne et al, 2001]: usage of hyperbox
- **NSGA-II** [Deb et al, 2002]: non-dominated sorting + crowding distance
- **SPEA2** [Zitzler et al, 2002]: number of dominated solutions + density estimation
- **DEMO** [Robic et al, 2005]: differential evolution + crowding distance
### Related Work (1)

#### Classic approaches
- PESA-II [Corne et al, 2001]: usage of hyperbox
- NSGA-II [Deb et al, 2002]: non-dominated sorting + crowding distance
- SPEA2 [Zitzler et al, 2002]: number of dominated solutions + density estimation
- DEMO [Robic et al, 2005]: differential evolution + crowding distance

#### Principle
- Proximity to the Pareto front
- Uniform distribution of the solutions (good spread)
Specific approach
- IBEA [Zitzler et al, 2004]: indicator-based
Related Work (2)

Specific approach
- IBEA [Zitzler et al, 2004]: indicator-based

Set preference relation [Zitzler et al., 2009]
Generic approach that prune solutions from a set according to some relation given by the decision maker.
Non-dominated Sorting

Perimeter of the cuboid formed by using the nearest neighbors.
NSGA-II Description

Non-dominated Sorting

Crowding distance
Perimeter of the cuboid formed by using the nearest neighbors.
MOEA

NSGA-II Loop

Main loop

1. Combine parent population $P_t$ and offspring $Q_t$ into $R$
2. Keep all first fronts of $R$ (with non-dominating sorting) that fit in $P_{t+1}$
3. For the last front that does not fit into $P_{t+1}$: select the solutions that have the highest crowding distances
4. Generate next offspring $Q_{t+1}$ from $P_{t+1}$
Definition (Algorithm PR (main loop))

1. Generate offspring $B(t + 1)$ from $A(t)$
2. Combine $A(t)$ and $B(t + 1)$ into $A(t + 1)$
3. $t = t + 1$
Rudolph MOEA Conditions

Definition (Algorithm PR (main loop))

1. Generate offspring $B(t + 1)$ from $A(t)$
2. Combine $A(t)$ and $B(t + 1)$ into $A(t + 1)$
3. $t = t + 1$

Theorem (Rudolph, 2000)

Algorithm PR converges if the transition matrix from $A(t)$ to $B(t + 1)$ is positive.
# NSGA-II Modifications

## Main loop

1. Combine parent population $P_t$ and offspring $Q_t$ into $R$
2. Elitism mechanism (keep best solutions of $R$)
3. For each solution, assign a probability to be kept (larger for first fronts and solutions with high crowding distances)
4. Select a subset of solutions of $R$ and out them in $P_{t+1}$ according to these probabilities
5. Generate next offspring $Q_{t+1}$ from $P_{t+1}$
NSGA-II Modifications

Main loop

1. Combine parent population $P_t$ and offspring $Q_t$ into $R$
2. Elitism mechanism (keep best solutions of $R$)
3. For each solution, assign a probability to be kept (larger for first fronts and solutions with high crowding distances)
4. Select a subset of solutions of $R$ and out them in $P_{t+1}$ according to these probabilities
5. Generate next offspring $Q_{t+1}$ from $P_{t+1}$

Theorem (Canon-Jeannot, 2010)

The modified version of NSGA-II converges with a local mutation operator (with crossover probability lower than 1).
Outline

1. Convergence with Local Mutation
2. MOEA
3. Scheduling
4. Conclusion
A Scheduling Problem

Parallel application
- Set of tasks $V$
- Graph of precedence constraints

Heterogeneous platform
- Distinct resources:
  - computation
  - communication
A Scheduling Problem

Parallel application
- Set of tasks $V$
- Graph of precedence constraints

Heterogeneous platform
- Distinct resources:
  - computation
  - communication

Definition

**Scheduling** consists to assign a computation resource to each task and to set their start and end time.
In our stochastic version, each duration is defined by a random variable (RV). The makespan is also a RV.

From the makespan distribution, we measure:

- its central tendency (efficiency)
- its statistical dispersion (robustness)

The mean and standard deviation are relevant metrics.
Matching string [Siegel et al, 1997]

Array of processors on which are executed the tasks. The $i$th task is executed on the processor in the $i$th position in the array.
Scheduling

Chromosome Representation

Matching string [Siegel et al, 1997]
Array of processors on which are executed the tasks. The \(i\)th task is executed on the processor in the \(i\)th position in the array.

Schedule string [Siegel et al, 1997]
Permutation of the tasks that respect the linear extension of the DAG. Determine the order of execution of tasks on each processor.
Matching string [Siegel et al, 1997]

Array of processors on which are executed the tasks. The $i$th task is executed on the processor in the $i$th position in the array.

Schedule string [Siegel et al, 1997]

Permutation of the tasks that respect the linear extension of the DAG. Determine the order of execution of tasks on each processor.

Example

Matching string

```
1 1 2 1
```

Schedule string

```
1 2 3 4
```
Mutation Operator

Matching string
A task is randomly selected and repositioned on an arbitrary processor.

Schedule string
A task is randomly selected and put forward or backward in the permutation (while respecting the constraints of the DAG).
Matching string

Possible to generate any string by considering each task one by one ($n$ steps, with $n$ the number of tasks).
Matching string
Possible to generate any string by considering each task one by one \((n\) steps, with \(n\) the number of tasks).

Schedule string
The maximum number of permutations needed to obtain any linear extension from any other is the linear extension diameter (upper bounded by \(n^2\)).
Matching string

Possible to generate any string by considering each task one by one ($n$ steps, with $n$ the number of tasks).

Schedule string

The maximum number of permutations needed to obtain any linear extension from any other is the linear extension diameter (upper bounded by $n^2$).

Local mutation

Applied $n^2$, the mutation operator is global. The modified version of NSGA-II converges towards the set of optimal solutions for this problems.
Outline

1 Convergence with Local Mutation

2 MOEA

3 Scheduling

4 Conclusion
Contributions [Canon, Jeannot, 2010]

- Convergence proof of a modified NSGA-II.
- A practical application
- The convergence probability is infinitesimal
Contributions [Canon, Jeannot, 2010]

- Convergence proof of a modified NSGA-II.
- A practical application
- The convergence probability is infinitesimal

Perspective

Convergence speed