DAMS: Distributed Adaptive Metaheuristic Selection

JET (Bond) 23, Paris

Bilel Derbel[1,2] Sébastien Verel[1,3]

[1]INRIA Lille-Nord Europe
[2]Université Lille 1
[3]Université Nice Sophia Antipolis
France

http://www.i3s.unice.fr/∼verel

November 23, 2012

Position of the work

- One EA key point :
  Exploitation / Exploration tradeoff
- One main practicle difficulty :
  Choose operators, design, value of parameters, representation
- Parameters setting (Lobo et al. 2007) :
  - Off-line before the run : *parameter tuning*,
  - On-line during the run : *parameter control*.
- Increasing number of computation ressources (GPU, ...)
  but, add parameters

Position of the work

- One EA key point :
  Exploitation / Exploration tradeoff
- One main practicle difficulty :
  Choose operators, design, value of parameters, representation
- Parameters setting (Lobo et al. 2007) :
  - Off-line before the run : *parameter tuning*,
  - On-line during the run : *parameter control*.
- Increasing number of computation ressources (GPU, ...)
  but, add parameters

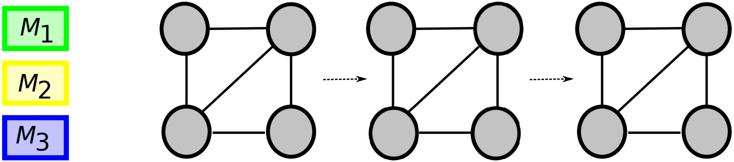
Distributed Adaptative Metaheuristic Selection (DAMS)

- **Control** the execution of different **metaheuristics** in a **distributed environment**

Execution of metaheuristics in a distributed environment

- 3 metaheuristics $\{M_1, M_2, M_3\}$
- 4 nodes of computation

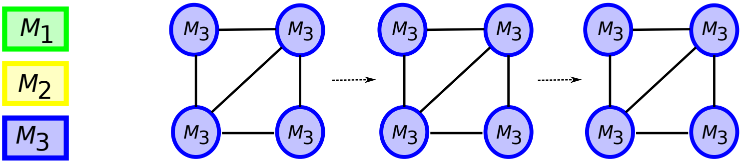
Which algorithm can we design ?



Execution of metaheuristics in a distributed environment

- 3 metaheuristics $\{M_1, M_2, M_3\}$
- 4 nodes of computation

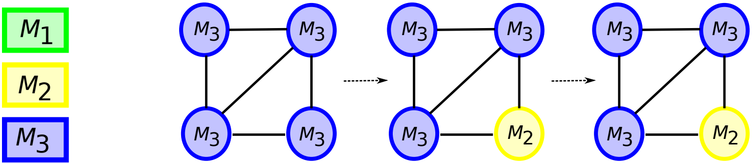
Which algorithm can we design ?



Execution of metaheuristics in a distributed environment

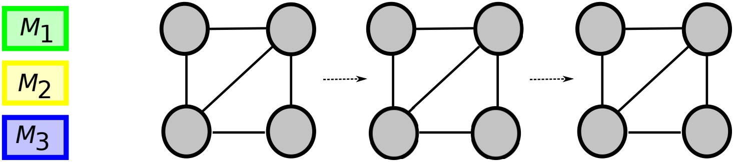
- 3 metaheuristics $\{M_1, M_2, M_3\}$
- 4 nodes of computation

Which algorithm can we design ?



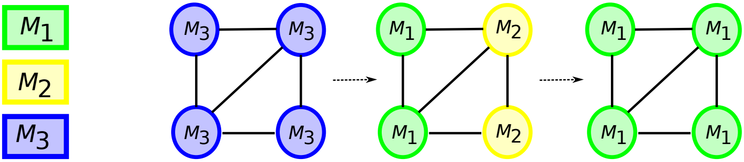
Execution of metaheuristics in a distributed environment

- 3 metaheuristics $\{M_1, M_2, M_3\}$
- 4 nodes of computation



Execution of metaheuristics in a distributed environment

- 3 metaheuristics $\{M_1, M_2, M_3\}$
- 4 nodes of computation

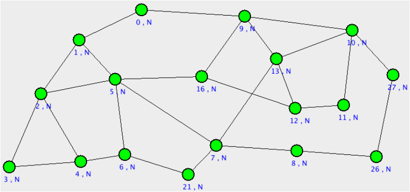


Goal of control : Optimal strategy

The strategy that gives the best overall performances :

- At first, execution of M_3
- then, M_1 and M_2, and then, M_1.

Point of view

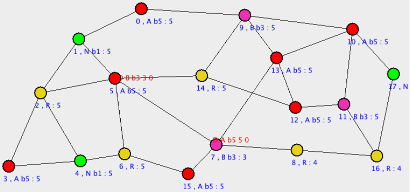


Keys

- Distributed environment = unique system to control
- Attach a metaheuristic to each computational node
- Modify the topology if necessary

$\Rightarrow$ Toward heterogenous parallel EA (island model)

Point of view



Keys

- Distributed environment = unique system to control
- Attach a metaheuristic to each computational node
- Modify the topology if necessary

$\Rightarrow$ Toward heterogenous parallel EA (island model)

Previous works

- Fix strategy :
  ex. 20% nodes 'explorative' EA, 80% nodes 'exploitative' EA

Previous works

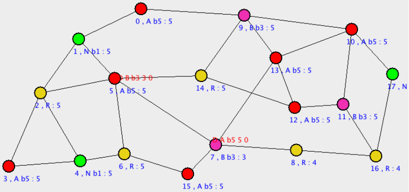
- Fix strategy :
  ex. 20% nodes 'explorative' EA, 80% nodes 'exploitative' EA
- Control of communication, topology :
  - Gossip protocol, newcast protocol [Laredo et al., 10]
  - Heterogenous island model : [Candan et al., 12]
    Migration rate control by the quality of operators
- Control of parameters sharing global information :
  - Master/worker to self-adapt pop. size [Bonnaire et al., 05]
  - Self-adapt pop. size, crossover [Srinivasa et al., 07]
- Control of parameters, distributed environment :
  - In each island : [Tongchim et al., 02]
    Testing 2 possible parameter settings on half population (EA)
    Send/receive (and mutate) the best parameters setting

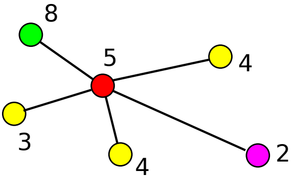
Control : Local strategy

m^n attachements with n nodes and m metaheuristics

Control : Local strategy

m^n attachements with n nodes and m metaheuristics





Local distributed strategy :

For each computational node :

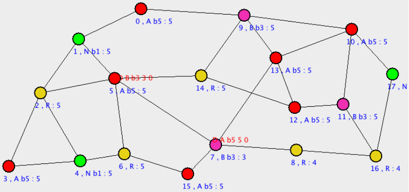
- Measure the efficiency of neighboring metaheuristics
- Select a metaheuristic according to the local information

Trade-off between :

Exploitation of the most promizing metaheuristics

Exploration of news ones

Select Best and Mutate strategy (SBM)



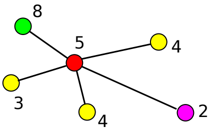
For each nodes :

$k \leftarrow$ initMeta()
$P \leftarrow$ initPop()
$reward \leftarrow 0$
while Not stop **do**
  Migration of populations
  Send (k, $reward$)
  Receive $\forall i$ (k_i, $reward_i$)

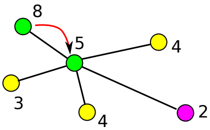
  $k \leftarrow$ best meta from k_i's
  if $rnd(0,1) < p_{mut}$ **then**
    $k \leftarrow$ random meta
  end if

  $P_{new} \leftarrow \text{meta}_k(\text{P})$
  $reward \leftarrow \text{Reward}(P,\ P_{new})$
  $P \leftarrow P_{new}$
end while

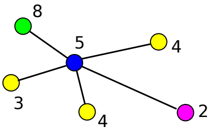
Select Best and Mutate strategy (SBM)



Select 'best' one (rate $1 - p_{mut}$)



Select random one (rate p_{mut})



For each nodes :

$k \leftarrow$ initMeta()
$P \leftarrow$ initPop()
$reward \leftarrow 0$
while Not stop **do**
  Migration of populations
  Send (k, $reward$)
  Receive $\forall i$ (k_i, $reward_i$)

  $k \leftarrow$ best meta from k_i's
  if $rnd(0,1) < p_{mut}$ **then**
    $k \leftarrow$ random meta
  end if

  $P_{new} \leftarrow \text{meta}_k(\text{P})$
  $reward \leftarrow \text{Reward}(P,\ P_{new})$
  $P \leftarrow P_{new}$
end while

Experimental design

From 'Adaptive Operator Selection'

Dynamical Multi-Armed Bandit (DAMS) [Fiahlo et al., GECCO 2008, 2009]

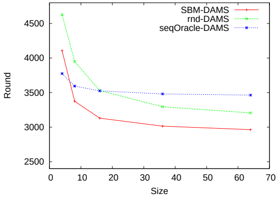
- 'Drosophilia' problems : *OneMax* problem ($l = 10^4$)
- Initial solution 000....0
- 4 mutations operators :
  - $5-$bit, $3-$bit, $1-$bit, $1/l$ bit-flip
- $(1 + \lambda)-$EA with $\lambda = 50$

SBM-DAMS

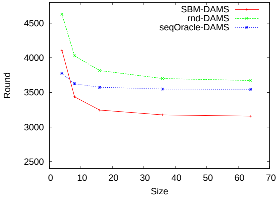
- One iteration of $(1 + \lambda)-$EA
- Reward : fitness gain $f(x_{new}) - f(x)$
- Three topologies : complete, circle, grid graph.
- $p_{mut} = 10^{-3}$, $\{0.0005, 0.005, 0.001, 0.01, 0.05, 0.1, 0.2, 0.3\}$
- 20 independent runs

Performances, adaptation properties

Grid topologies :



Circle topologies :



No distributed adaptive algorithm in the litterature

Algorithms :

- Selection of 'good' meta important ?
  $\Rightarrow$ Random selection of metaheuristic (Rnd-DAMS)
- Optimal independent strategy ?
  $\Rightarrow$ Selection with sequential oracle (seqOracle-DAMS)

SBM outperforms ($\forall$ topo., $\forall$ n) :

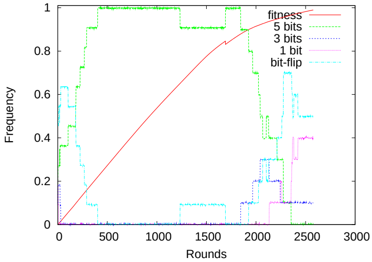
Random selection

Sequential oracle

$\Rightarrow$ Adaptation of SBM very efficient

Dynamics of SBM-DAMS

Average frequency and fitness



- First, $1/l$ bit-flip mutation (opposite of sequential oracle)
- Then 5-bits, 3-bits, 1-bit, $1/l$ bit-flip

Effeciency of local communication

For n nodes around fitness 0 :

$1/l$ bit-flip mutation $>$ 5-bit mutation

For fitness 0,

$$\mathbb{P}(\text{gain bit-flip} > 5) = 1 - \alpha^{\lambda \cdot n}$$

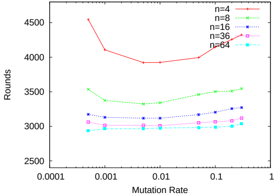
α prob at most 5 bits are flipped

Whatever the topology :

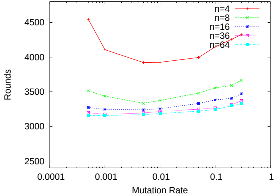
- Decentralized decision (but not independent)
- Global property : correct mutation operator selection

Robustness of metaheuristic mutation parameter p_{mut}

Grid topologies :



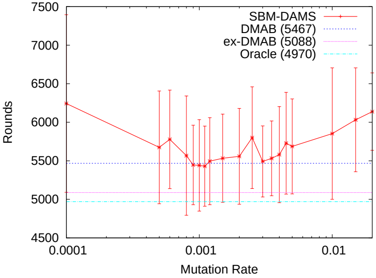
Circle topologies :



- Range of performances small according to metaheuristic mutation rate
- $p_{mut} \in [10^{-3}, 10^{-1}]$

$\Rightarrow$ Quite robust parameter for this problem

Parallel properties : SBM-DAMS *vs.* sequential AOS



- Sequential Adaptive Operator Selection :
  DMAB running $(1+50)$-EA
- SBM-DAMS :
  50 nodes (complete graph) running $(1+1)$-EA

- Performaces (number of evaluations) :
  oracle $<$ ex-DMAB $<$ SBM $<$ DMAB
- More information for AOS :
  DMAB : update operator every mutation
  SBM : update every n mutations
- But, SBM time could be divided by n

Distributed Adaptive Metaheuristic Selection

A generic framework

$M \leftarrow \text{INIT\_META}()$
$P \leftarrow \text{INIT\_POP}()$
$\mathcal{S} \leftarrow \text{INIT\_STATE}()$
$\mathsf{I} \leftarrow \{M, P, \mathcal{S}\}$
repeat

Distributed Adaptive Metaheuristic Selection

A generic framework

$M \leftarrow \text{INIT\_META}()$
$P \leftarrow \text{INIT\_POP}()$
$\mathcal{S} \leftarrow \text{INIT\_STATE}()$
$\mathsf{I} \leftarrow \{M, P, \mathcal{S}\}$
repeat
  */** Distributed Level **/*
  $c \leftarrow \text{LOCAL\_COMMUNICATION}(\mathsf{I})$
  $P \leftarrow \text{UPDATE\_POPULATION}(\mathsf{I}, c)$
  $\mathcal{S} \leftarrow \text{UPDATE\_LOCAL\_STATE}(\mathsf{I}, c)$

Distributed Adaptive Metaheuristic Selection

A generic framework

$M \leftarrow \text{INIT\_META}()$
$P \leftarrow \text{INIT\_POP}()$
$\mathcal{S} \leftarrow \text{INIT\_STATE}()$
$\mathsf{I} \leftarrow \{M, P, \mathcal{S}\}$
repeat
  */** Distributed Level **/*
  $c \leftarrow \text{LOCAL\_COMMUNICATION}(\mathsf{I})$
  $P \leftarrow \text{UPDATE\_POPULATION}(\mathsf{I}, c)$
  $\mathcal{S} \leftarrow \text{UPDATE\_LOCAL\_STATE}(\mathsf{I}, c)$

  */** Metaheuristic Selection Level **/*
  $M \leftarrow \text{SELECT\_META}(\mathsf{I})$

Distributed Adaptive Metaheuristic Selection
A generic framework

$M \leftarrow \text{INIT\_META}()$
$P \leftarrow \text{INIT\_POP}()$
$\mathcal{S} \leftarrow \text{INIT\_STATE}()$
$\mathsf{I} \leftarrow \{M, P, \mathcal{S}\}$
repeat
    /** *Distributed Level* **/
    $c \leftarrow \text{LOCAL\_COMMUNICATION}(\mathsf{I})$
    $P \leftarrow \text{UPDATE\_POPULATION}(\mathsf{I}, c)$
    $\mathcal{S} \leftarrow \text{UPDATE\_LOCAL\_STATE}(\mathsf{I}, c)$

    /** *Metaheuristic Selection Level* **/
    $M \leftarrow \text{SELECT\_META}(\mathsf{I})$

    /** *Atomic Low Level* **/
    $(P, \mathcal{S}) \leftarrow \text{APPLY\_META}(M, P)$
until $\text{STOPPING\_CONDITION}(\mathsf{I})$

Distributed Adaptive Metaheuristic Selection

A generic framework

$M \leftarrow$ INIT_META()
$P \leftarrow$ INIT_POP()
$\mathcal{S} \leftarrow$ INIT_STATE()
$\mathsf{I} \leftarrow \{M, P, \mathcal{S}\}$
repeat
    */** Distributed Level **/*
    $c \leftarrow$ LOCAL_COMMUNICATION(I)
    $P \leftarrow$ UPDATE_POPULATION(I, c)
    $\mathcal{S} \leftarrow$ UPDATE_LOCAL_STATE(I, c)

    */** Metaheuristic Selection Level **/*
    $M \leftarrow$ SELECT_META(I)

    */** Atomic Low Level **/*
    $(P, \mathcal{S}) \leftarrow$ APPLY_META(M, P)
until STOPPING_CONDITION(I)

$k \leftarrow$ initMeta()
$P \leftarrow$ initPop()
$reward \leftarrow 0$
while Not stop **do**
    Migration of populations
    Send (k, $reward$)
    Receive $\forall i$ (k_i, $reward_i$)

    $k \leftarrow$ best meta from k_i's
    if $rnd(0,1) < p_{mut}$ **then**
        $k \leftarrow$ random meta
    end if

    $P_{new} \leftarrow$ meta$_k$(P)
    $reward \leftarrow$ Reward(P, P_{new})
    $P \leftarrow P_{new}$
end while

Distributed Adaptive Metaheuristic Selection

A generic framework

$M \leftarrow$ INIT_META()
$P \leftarrow$ INIT_POP()
$\mathcal{S} \leftarrow$ INIT_STATE()
$\mathsf{I} \leftarrow \{M, P, \mathcal{S}\}$
repeat
  */** Distributed Level **/*
  $c \leftarrow$ LOCAL_COMMUNICATION(I)
  $P \leftarrow$ UPDATE_POPULATION(I, c)
  $\mathcal{S} \leftarrow$ UPDATE_LOCAL_STATE(I, c)

  */** Metaheuristic Selection Level **/*
  $M \leftarrow$ SELECT_META(I)

  */** Atomic Low Level **/*
  $(P, \mathcal{S}) \leftarrow$ APPLY_META(M, P)
until STOPPING_CONDITION(I)

$k \leftarrow 0$
$P \leftarrow$ initPop()

while Not stop **do**
  Migration of populations

  $P_{new} \leftarrow \text{meta}_k(\text{P})$

  $P \leftarrow P_{new}$
end while

Conclusions and Futur Works

- **On-line control** of **metaheuristics** execution in **distributed environment**
- 'Select Best and Mutate' strategy :
  efficient adaptation and parallel properties

Future works

- **Theoretical works** on running-time (in progress)
- Study the performances on NP-hard **multimodal problems**
- Use together **different metaheuristics** (EA, memetic EA, ant, etc.)
- Study addition and deletion of computational nodes

Future works

$M \leftarrow$ INIT_META()
$P \leftarrow$ INIT_POP()
$\mathcal{S} \leftarrow$ INIT_STATE()
$\mathrm{I} \leftarrow \{M, P, \mathcal{S}\}$
repeat
  /** Distributed Level **/
  $c \leftarrow$ LOCAL_COMMUNICATION(I)
  $P \leftarrow$ UPDATE_POPULATION(I, c)
  $\mathcal{S} \leftarrow$ UPDATE_LOCAL_STATE(I, c)

  /** *Metaheuristic Selection Level* **/
  $M \leftarrow$ SELECT_META(I)

  /** Atomic Low Level **/
  $(P, \mathcal{S}) \leftarrow$ APPLY_META(M, P)
until STOPPING_CONDITION(I)

Metaheuristic Selection Level :

- Design selection metaheuristic methods (UCB strategy)

Distributed Level :

- Rewards : cumulative measures
- Information, state : Communication of rewards
- Asynchronous method
- Binary rewards (crossover)
- Define an adaptive migration policy

Environnement Level :

- grid5000