Optimising Small-World Properties in VANETs with a Parallel Multi-Objective Coevolutionary Algorithm

Grégoire Danoy, Julien Schleich, Bernabé Dorronsoro and Pascal Bouvry

HPCLATAM 2013
July 29, 2013
Outline

1. Introduction
2. Asynchronous Parallel CCNSGA-II
3. Injection Network Problem
4. Experimental Setup
5. Numerical Results
6. Conclusion and Perspectives
1. Introduction
2. Asynchronous Parallel CCNSGA-II
3. Injection Network Problem
4. Experimental Setup
5. Numerical Results
6. Conclusion and Perspectives
Introduction

Motivation

- Real-world (RW) problems are typically multi-objective
- MOEAs show limitations on such large-scale problems
- Cooperative coevolutionary MOEAs are one promising option, but few works:
  - applied them on RW problems,
  - exploit their parallelization capabilities.

Objectives

- Apply for the first time a parallel asynchronous cooperative coevolutionary MOEA
- Optimize a topology control problem in VANETs
- Analyze its performance (speedup, quality)
1. Introduction

2. Asynchronous Parallel CCNSGA-II

3. Injection Network Problem

4. Experimental Setup

5. Numerical Results

6. Conclusion and Perspectives
Multi-Objective Optimization

- Optimize more than one objective at the same time
  - Objectives are usually conflicting
  - Improving one means worsening the others

- Results in a set of non-dominated solutions
  - Pareto front

- Performance of approximate techniques
  - Convergence
  - Diversity

Bad diversity

Bad convergence

Ideal case
NSGA-II

- Non-dominated Sorting Genetic Algorithm [1]
- Most popular metaheuristic for multi-objective optimization

Based on Potter Cooperative Coevolutionary EA [2]

Subpopulations evaluate part of global solution vector

Cooperate by exchanging local representatives

Parallel Asynchronous CCNSGA-II

- Good parallelization capabilities
  - Subpopulations evolve in parallel
- Asynchronous CCNSGA-II
  - Proposed in [3]
  - Only applied on MO test functions

1. Introduction

2. Asynchronous Parallel CCNSGA-II

3. Injection Network Problem

4. Experimental Setup

5. Numerical Results

6. Conclusion and Perspectives
Vehicular Ad hoc Networks (VANETs)

- Wireless ad hoc network
  - No central entity
  - Nodes act as routers
- Mobility induces topological changes
- Partitioning problem
Injection Network

- Hybrid VANETs
  - Vehicle-to-Vehicle and vehicle-to-Infrastructure

- Injection points
  - Nodes connected to infrastructure
  - Form fully-connected overlay network

- Rely on small-world properties
  - High CC: better broadcasting efficiency
  - Low APL: faster and easier to maintain routing

Multi-objective Optimization Problem

\[
f(s) = \begin{cases} 
\min \{\text{inj}\} \\
\max \{\text{cc}\} \quad ; \quad \text{s. t. component} = 1 \\
\min \{a\text{pl}_{\text{diff}}\}
\end{cases}
\]

With \( a\text{pl}_{\text{diff}} = |a\text{pl} - a\text{pl}_{\text{random}}| \)

- Equivalent random graph
  - same number of nodes and average density
  - averaged over 30 different instances

- Generated using Watts rewiring process [5]
  - with randomness, i.e. \( p = 1 \)

Outline

1. Introduction
2. Asynchronous Parallel CCNSGA-II
3. Injection Network Problem
4. Experimental Setup
5. Numerical Results
6. Conclusion and Perspectives
Experimental Setup

- VehILux
- Realistic Mobility Model
- Screenshots from traces (Graph)
- Equivalent Random Graphs
- Graphstream
- Dynamic Graph Library
- Multi-objective solver
- jMetal
- Solution Set
**Experimental Setup - Algorithms configuration**

- Configuration is the one originally suggested by the authors in [2]

- Solution encoding: binary array
  - 1 for injection point
  - 0 for normal node
  - length: number of devices (cars)

---

**Table: Configuration details**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numb. of subpop.</td>
<td>4</td>
</tr>
<tr>
<td>Cores used</td>
<td>4 (1 for NSGA-II)</td>
</tr>
<tr>
<td>Number of threads</td>
<td>1 per subpopulation</td>
</tr>
<tr>
<td>Population size</td>
<td>100</td>
</tr>
<tr>
<td>Final archive size</td>
<td>100, from all subpops.</td>
</tr>
<tr>
<td>Migration policy</td>
<td>20 random</td>
</tr>
<tr>
<td>Max. evaluations</td>
<td>50,000</td>
</tr>
<tr>
<td>Pop. initialisation</td>
<td>Random</td>
</tr>
<tr>
<td>Selection</td>
<td>Binary tournament</td>
</tr>
<tr>
<td>Recombination</td>
<td>DPX</td>
</tr>
<tr>
<td>Probability</td>
<td>0.9</td>
</tr>
<tr>
<td>Mutation</td>
<td>Bit Flip</td>
</tr>
<tr>
<td>Probability</td>
<td>$\frac{1}{\text{number of variables}}$</td>
</tr>
<tr>
<td>Independent runs</td>
<td>30</td>
</tr>
</tbody>
</table>

* Not applicable for NSGA-II

---

Experimental Setup - Network instances

- VehILux mobility model [3]
  - Realistic road network topology (OpenStreetMaps)
  - Real traffic counting data from the Luxembourg Ministry of Transport

<table>
<thead>
<tr>
<th>Surface</th>
<th>0.6 km²</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coverage radius</td>
<td>100 m</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>6 a.m.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Number</td>
<td>21900 22200 22500</td>
</tr>
<tr>
<td>Number of Nodes</td>
<td>40 62 60</td>
</tr>
<tr>
<td>Partitions</td>
<td>10 8 6</td>
</tr>
<tr>
<td>Solution space size</td>
<td>$1^{12} 4.61^{18} 1.15^{18}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7 a.m.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Number</td>
<td>25500 25800 26099</td>
</tr>
<tr>
<td>Number of Nodes</td>
<td>223 248 301</td>
</tr>
<tr>
<td>Partitions</td>
<td>10 6 7</td>
</tr>
<tr>
<td>Solution space size</td>
<td>$1.34^{67} 4.52^{74} 4.07^{90}$</td>
</tr>
</tbody>
</table>

1. Introduction
2. Asynchronous Parallel CCNSGA-II
3. Injection Network Problem
4. Experimental Setup
5. **Numerical Results**
6. Conclusion and Perspectives
HPC facility of the UL
- Nodes HP Proliant
- 2 Intel L5640 CPUs having 6 cores each at 2.26 GHz.

Computation times
- Increase drastically with the problem size

Super-linear speedup
- For all instances
- Average: 4.55
Solution quality

- **SPREAD - Diversity of solutions**
  - NSGA-II better on half of the instances

- **EPSILON - Convergence**
  - CCNSGA-II better on all instances
Outline

1. Introduction
2. Asynchronous Parallel CCNSGA-II
3. Injection Network Problem
4. Experimental Setup
5. Numerical Results
6. Conclusion and Perspectives
Conclusions and Perspectives

Conclusions
- First application of asynchronous parallel CCNSGA-II on RW problem
  - Injection Network in VANETs
- Analyzed performance of CCNSGA-II on realistic instances
  - Super linear speedup
  - Better convergence

Perspectives
- Analyze scalability of the parallel CCNSGA-II
- Develop decentralised heuristics
- Use CCMOEAP empirical bounds to assess the heuristics performance
Thank you for your attention

Optimising Small-World Properties in VANETs with a Parallel Multi-Objective Coevolutionary Algorithm

Grégoire Danoy, Julien Schleich, Bernabé Dorronsoro and Pascal Bouvry

gregoire.danoy@uni.lu
**Small world Properties**

- **Small Average Path Length (APL)**
  \[
  APL = \frac{1}{n(n-1)} \sum_{i,j} d(v_i, v_j)
  \]

- **High Clustering Coefficient (CC)**
  - **Local**
    \[
    CC_v = \frac{|E(\Gamma_v)|}{k_v(k_v - 1)}
    \]
  - **Global**
    \[
    CC = \frac{1}{n} \sum_v CC_v
    \]
Definition of Watts [1]

\[ APL \approx APL_{random} \]
\[ CC \gg CC_{random} \]
Problem 1: the network can be partitioned

- There exists no path between some pair of nodes

Solution 1: injection points

- A subset of nodes use an additional network interface
- This subset of nodes forms a fully connected overlay network
Problem 2: the topological properties are not optimal

Solution 2: select injection points in order to obtain better small world properties

• High CC: better broadcasting efficiency
• Low APL: faster and easier to maintain routing
Network Instances

Instance 21900

Instance 25800