Challenges in applying Evolutionary Algorithms to real-world problems

Professor Xiaodong Li
School of Science (Computer Science and Software Engineering)
RMIT University, Melbourne, Australia
Email: xiaodong.li@rmit.edu.au
Why this talk?

• Personal experience when talking to optimization researchers from math and operations research backgrounds;

• Dealing with practitioners who need solutions to their real-world problems;

• Growing popularity of nature-inspired optimization techniques;

• Clear gaps between existing methods and practical problems to be solved;

• Human-in-the-loop approach to optimization;

• Personal views on research needs and motivations.
My pathway to become an EC researcher

- Graduated from Xidian University with a bachelor in Information Science;
- Master and PhD in Artificial Intelligence, Otago University, New Zealand;
- First academic job as a lecturer at Charles Sturt University, then Monash, and finally at RMIT University;
- In early years interested in artificial life, complexity, and swarming behaviour;
- After a PhD on “Connectionist Learning Architecture Based on an Optical Thin-Film Multilayer Model“, looked for new research ideas...
- Developed a fire-spread simulation model using cellular automata (i.e., an artificial life model), which led to my interests in swarm intelligence;
- Attended Swarm Fest in 2001, and first time at CEC in 2002.
- Attended first GECCO in 2003; my first ever GECCO paper, on a multiobjective PSO algorithm won the ACM SIGEVO Impact Award in 2013 for the highest citations among all GECCO’03 paper.
- Visiting research fellow to Prof. Xin Yao at the University of Birmingham in 2008.

My pathway to become an EC researcher

CEC'17 conference, San Sebastián, Spain, receiving 2017 IEEE CIS TEVC Outstanding Paper Award.
We study and develop **nature-inspired** computational models and algorithms, especially in the areas of evolutionary computation and machine learning, and apply them to real-world problems. The group takes an inter-disciplinary approach drawing its inspirations from mathematical programming, meta-heuristics, and operations research.

**Staff:** Prof. Xiaodong Li (Group leader), A/Prof. Vic Ciesielski, Dr. Andy Song, Dr. Jeffrey Chan, A/Prof. Fabio Zambetta  
**Students:** more than 15 PhD candidates, plus several master and honours students  
**Teaching:** Artificial Intelligence, Machine Learning, Evolutionary Computing, and Data Mining

**Further information:** [https://titan.csit.rmit.edu.au/~e46507/ecml/](https://titan.csit.rmit.edu.au/~e46507/ecml/)
ECML members
My research activities so far have been largely focussed on algorithmic development or enhancement.

Next few slides are some examples…

But my interests have gradually shifted towards solving more practically relevant problems.
Divide-and-Conquer: Large Scale Global Optimization with variable grouping techniques

Multi-modal Optimization using Niching Methods


Preference-based Evolutionary Multiobjective Optimization


Shifting towards solving more practical problems

• When interacting with operations research community;
• NICTA/Data61 optimization summer schools;
• People in engineering and mathematical programming (RMITOpt group, AMSI Optimise 2017, Data61 talk series);
• PhD projects with more practical problems;
• Personal communication with Prof. Zbigniew Michalewicz; his company SolveIT (specialised in integrated planning and supply chain optimization) winning big contracts with some of the largest companies in Australia such as BHP, Rio Tinto.

• Changing from more algorithms focused to more problem focused.

Strengths of nature-inspired optimization methods

• Pros
  – Robust and generic solution methods;
  – No gradient information is required;
  – Less demanding on rigorous math formulation;
  – Usually work with a population of candidate solutions (implicit parallelism);
  – Strong global search capability, i.e., less prone to getting stuck on local optima, and work well on non-convex problems;
  – Fewer assumptions

• Cons
  – Weak theoretical foundation; less established;
  – Computationally more expensive, as compared with conventional methods;
  – Little consideration on solution constructive approaches;
  – Difficult to analyse population dynamics;
  – Difficult to apply to complex large-scale problems with multitudes of components inter-dependent to each other.
Mathematical programming methods

• **Pros**
  – Strong theoretical foundation, more established;
  – Work well on linear and integer programming problems; if problems are not linear, and approximation can be still good enough for certain problems;
  – Work well on problems with convex shapes;
  – “Construct and search” approach can be effective;
  – Some effective problem reduction techniques;
  – Usually single-solution search methods.

• **Cons**
  – Often have strong assumptions, e.g., convexity;
  – Non-convex and nonlinear problems are much harder to deal with; unfortunately many real-world problems belong to this class;
  – MIP solvers such as CPLEX or GUROBI often can only solve small or medium sized problem instances, e.g., the branch-and-bound methods.
Mathematical programming methods

- **Lagrangian relaxation:**

  \[
  \begin{align*}
  \text{Minimize} & \quad c^T x \\
  \text{subject to} & \quad Ax = b \\
  & \quad x \in X
  \end{align*}
  \]

  \[
  \begin{align*}
  \text{Minimize} & \quad c^T x + \mu^T (Ax - b) \\
  \text{subject to} & \quad x \in X
  \end{align*}
  \]

  • Now the original problem is transformed into the following relaxed problem:

  \[
  \begin{align*}
  \text{Minimize} & \quad c^T x + \mu^T (Ax - b) \\
  \text{subject to} & \quad x \in X
  \end{align*}
  \]

  • Now the task is to find the greatest lower bound for the above relaxed problem.

  • Other well known methods include LP relaxation, Dantzig-Wolfe decomposition, column generations, dynamic programming, Benders’ decomposition, etc.
Hybrid methods – new potential?

• **Synergy** - bring together the bests from both paradigms - EC and math programming methods;

• Focusing more on solving problems that are practically relevant, rather than purely for new algorithms;

• Fertile grounds for **new research ideas!!**

• EC methods can leverage on the strong theoretical foundation of the exact methods, from the field of operations research;

• Exact methods can be enhanced to solve non-convex and nonlinear problems for large problem instance sizes, and can be made more robust, with fewer assumptions.

Constrained Pit (CPIT) problems in mining

- Open-pit mining is an important industry in Australia.
- Small increases, or decreases, in efficiency can have a large effect on profit.
- The two most critical tasks in planning an open pit mine is deciding what to mine, and also the order in which to mine it.
- The CPIT problem combines these two tasks, allowing the mine operator to estimate the total value of the mine over its life and also to identify the most valuable areas for excavation.

- Properties:
  - very large-scale
  - Few side constraints, but many variables and precedence constraints;
  - Current MIP solvers cannot handle without first using decomposition.
The problem can be modelled as a network flow problem, and the goal is to find the maximum closure, which gives the maximum profit.


CPIT problem modelling

The MIP formulation for the CPIT problem is given below:

\[
\begin{align*}
\text{max} & \quad \sum_{b \in B} \sum_{t \in T} p_{bt} (x_{bt} - x_{bt-1}), \\
\text{s.t.} & \quad x_{bt} \leq x_{at} \quad \forall (a, b) \in \mathcal{P}, t \in T, \\
& \quad x_{bt} \leq x_{b,t+1} \quad \forall b \in B, t \in T, \\
& \quad \sum_{b \in B} q_{br} (x_{bt} - x_{bt-1}) \leq \overline{R}_{rt} \quad \forall r \in R, t \in T, \\
& \quad x_{bt} \in \{0, 1\} \quad \forall b \in B, t \in T.
\end{align*}
\]

Here, \( B \) is the set of blocks; \( T \) the set of periods; \( p_{bt} \) the profit made from mining block \( b \) at time \( t \); \( \mathcal{P} \) is the set of precedences, where \( (a, b) \in \mathcal{P} \) if block \( a \) must be mined immediately before block \( b \); \( R \) is the set of resources; \( q_{br} \) the amount of resource \( r \) consumed by mining \( b \); \( \overline{R}_{rt} \) the total amount of \( r \) available at \( t \); and, \( x_{bt} \) is a binary decision variable that is 1 if \( b \) is mined at \( t \) or earlier and 0 otherwise.

Merge search – from a population perspective

**Principle**: if a variable takes the same value across many solutions to the same problem, then it is likely to take the same value again (if another solution is generated). As the population size increases, the probability of this being true also increases. The nature of CPIT problem makes it a perfect fit to test this idea.

- The merge operation is a **problem reduction technique** that identifies groups of variables that can be removed from main problem.
- Once the merge has occurred, a MIP solver can be used to find a solution to this restricted problem.
- This solution can then be used to generate another population of neighbouring solutions, and the cycle continues.

Figure 1: By overlaying multiple solutions, regions of fixed and free variables can be identified to produce a reduced version of the problem.

Merge search – from a population perspective

Figure: a) Parallel merge search; b) Time expanded problem graph for two time periods. The cumulative variables ensure that once a block is mined, it stays mined in subsequent periods.

- Each block has a cone of precedence blocks that must be mined before it
- A population of solutions is produced by swapping these cones of blocks between periods

Figure: A block and its predecessor cone is swapped to create a neighbouring solution.

Merge search – from a population perspective

Table 1: Characteristics of minelib [7] datasets.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Blocks</th>
<th>Precedences</th>
<th>Periods</th>
<th>Variables</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>newman1</td>
<td>1,060</td>
<td>3,922</td>
<td>6</td>
<td>6,360</td>
<td>29,904</td>
</tr>
<tr>
<td>zuck_small</td>
<td>9,400</td>
<td>145,640</td>
<td>20</td>
<td>188,000</td>
<td>3,100,840</td>
</tr>
<tr>
<td>kd</td>
<td>14,153</td>
<td>219,778</td>
<td>12</td>
<td>169,836</td>
<td>2,807,196</td>
</tr>
<tr>
<td>zuck_medium</td>
<td>29,277</td>
<td>1,271,207</td>
<td>15</td>
<td>439,155</td>
<td>19,507,290</td>
</tr>
<tr>
<td>marvin</td>
<td>53,271</td>
<td>650,631</td>
<td>20</td>
<td>1,065,420</td>
<td>14,078,080</td>
</tr>
<tr>
<td>zuck_large</td>
<td>96,821</td>
<td>1,053,105</td>
<td>30</td>
<td>2,904,630</td>
<td>34,497,840</td>
</tr>
</tbody>
</table>

Initial solution construction: cones of blocks are computed and ranked according to their value and resource usage and then mined heuristically until the resource limits were reached for each period.

Table 2: Results on minelib dataset instances. Mean and standard deviation not reported for minelib [7], as the only information given was a single objective value. The subscripts $RS$ and $CS$ stand for random search and cone search respectively, and denote the algorithm used to construct the initial solutions.

<table>
<thead>
<tr>
<th>Instance</th>
<th>LP UB</th>
<th>minelib</th>
<th>RandomSearch</th>
<th>ConeSearch</th>
<th>SerialMergeRS</th>
<th>SerialMergeCS</th>
<th>ParallelMergeCS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mean</td>
<td>std. dev</td>
<td>mean</td>
<td>std. dev</td>
<td>mean</td>
<td>std. dev</td>
<td>mean</td>
</tr>
<tr>
<td>newman1</td>
<td>2.449E+07</td>
<td>2.348E+07</td>
<td>2.322E+07</td>
<td>9.356E+04</td>
<td>2.355E+07</td>
<td>1.021E+05</td>
<td>2.411E+07</td>
</tr>
<tr>
<td>zuck_small</td>
<td>8.542E+08</td>
<td>7.887E+08</td>
<td>5.676E+08</td>
<td>9.165E+06</td>
<td>7.868E+08</td>
<td>1.246E+07</td>
<td>7.832E+08</td>
</tr>
<tr>
<td>kd</td>
<td>4.095E+08</td>
<td>3.969E+08</td>
<td>3.508E+08</td>
<td>9.307E+05</td>
<td>3.915E+08</td>
<td>1.273E+06</td>
<td>3.845E+08</td>
</tr>
<tr>
<td>zuck_medium</td>
<td>7.106E+08</td>
<td>6.154E+08</td>
<td>4.587E+08</td>
<td>4.405E+06</td>
<td>6.083E+08</td>
<td>1.494E+07</td>
<td>6.390E+08</td>
</tr>
<tr>
<td>marvin</td>
<td>8.639E+08</td>
<td>8.207E+08</td>
<td>5.925E+08</td>
<td>9.839E+06</td>
<td>7.985E+08</td>
<td>6.507E+06</td>
<td>7.955E+08</td>
</tr>
<tr>
<td>zuck_large</td>
<td>5.739E+07</td>
<td>5.678E+07</td>
<td>4.136E+07</td>
<td>7.470E+04</td>
<td>5.042E+07</td>
<td>2.073E+05</td>
<td>4.616E+07</td>
</tr>
</tbody>
</table>

Further improvement on merge search

Table 3: Difference in problem size between the original and the reduced sub-problem produced by the merge operation.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Original</th>
<th>ParallelMerge</th>
<th>ImprovedMerge</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reduced</td>
<td>%</td>
<td>Reduced</td>
</tr>
<tr>
<td>newman1</td>
<td>6.36E+03</td>
<td>1.16E+03</td>
<td>18.2</td>
</tr>
<tr>
<td>zuck_small</td>
<td>1.88E+05</td>
<td>1.50E+04</td>
<td>8.0</td>
</tr>
<tr>
<td>kd</td>
<td>1.70E+05</td>
<td>1.42E+04</td>
<td>8.4</td>
</tr>
<tr>
<td>zuck_medium</td>
<td>4.49E+05</td>
<td>3.99E+04</td>
<td>9.1</td>
</tr>
<tr>
<td>marvin</td>
<td>1.07E+06</td>
<td>1.38E+04</td>
<td>1.3</td>
</tr>
<tr>
<td>zuck_large</td>
<td>2.90E+06</td>
<td>1.88E+05</td>
<td>6.5</td>
</tr>
</tbody>
</table>

Figure 4: By selecting different combinations of variable groups in the reduced sub-problem, new solutions can be generated.

Minimum cost network flow problems
Minimum cost integer flow problem (MCFP)

\[
z = \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} x_{ij}
\]

(1)

s.t. \[\sum_{j=1}^{n} x_{ij} - \sum_{k=1}^{n} x_{kj} = \begin{cases} q, & \text{if } i = 1 \\ 0, & \text{if } i = (2, 3, \ldots, n-1) \\ -q, & \text{if } i = n \end{cases} \quad (2)\]

\[l_{ij} \leq x_{ij} \leq u_{ij} \quad (i, j = 1, \ldots, n) \quad (3)\]

\[x_{ij} \in \mathbb{Z} \quad (i, j = 1, \ldots, n) \quad (4)\]

Figure 2: A chromosome for priority based encoding method.

Figure 3: An example of priority based decoding paths

• Using GA to evolve representation schemes for solving MCFP;
• Using non-convex cost functions.

Minimum cost integer flow problem (MCFP)

A set of 35 single-source single-sink MCFP instances is randomly generated with different number of nodes ($n = \{5, 10, 20, 40, 80, 120, 160\}$). Each instance has $n$ nodes and $m$ arcs. Five different networks are randomly generated for each node size $n$.

Minimum cost integer flow problem (MCFP)


<table>
<thead>
<tr>
<th>No.</th>
<th>n</th>
<th>m</th>
<th>PTGA-R t mean</th>
<th>std</th>
<th>PTGA-O t mean</th>
<th>std</th>
<th>PTGA-M t mean</th>
<th>std</th>
<th>PrGA t mean</th>
<th>std</th>
<th>LINDOGlocal t^* OBJ</th>
<th>AlphaECP t^* OBJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>8</td>
<td>32.2126</td>
<td>0.00E+00</td>
<td>32.2126</td>
<td>0.00E+00</td>
<td>32.2126</td>
<td>0.00E+00</td>
<td>32.2126</td>
<td>0.00E+00</td>
<td>32.2126</td>
<td>0.00E+00</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>8</td>
<td>33.0507</td>
<td>7.29E-15</td>
<td>33.0507</td>
<td>7.29E-15</td>
<td>33.0507</td>
<td>7.29E-15</td>
<td>33.0507</td>
<td>7.29E-15</td>
<td>33.0507</td>
<td>7.29E-15</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>8</td>
<td>33.1016</td>
<td>7.29E-15</td>
<td>33.1016</td>
<td>7.29E-15</td>
<td>33.1016</td>
<td>7.29E-15</td>
<td>33.1016</td>
<td>7.29E-15</td>
<td>33.1016</td>
<td>7.29E-15</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>8</td>
<td>30.3756</td>
<td>2.19E-14</td>
<td>30.3756</td>
<td>2.19E-14</td>
<td>30.3756</td>
<td>2.19E-14</td>
<td>30.3756</td>
<td>2.19E-14</td>
<td>30.3756</td>
<td>2.19E-14</td>
</tr>
</tbody>
</table>

Table: results on cost function F1. Three possible ways to send the flow over the generated path: randomly (R), one-by-one (O), or by a maximum possible amount (M).
Human-in-the-loop for EMO

- The field of evolutionary multiobjective optimisation has traditionally involved the approximation of the entire Pareto Front of the objective space.
- The computational effort required to find these solutions is significant and the number of solutions found can be considerable.
- By incorporating user preferences the search for solutions can be directed or focussed toward a region of interest.
- It is often assumed that the user has a set of predetermined preferences and an implicit value function that can evaluate potential solutions.
- To aid preference articulation in optimisation, the technique of progressive interactivity has been incorporated from Operations Research.
- This allows the user to learn about the problem, explore options and formulate their preferences while reducing the search space and computation time.

An interactive approach for EMO

- Decision maker interacts with an EMO algorithm during its optimization run
- DM can be educated and can give intermediate feedback. This means preference information can be adjusted during the run.
- The algorithm is more adaptive to changing needs.
- Search space can be significantly reduced, since effort is more targeted to regions of interests. **Machine learning** can be used to learn and model preference information.
An interactive approach for EMO

Figure 1: Preference Region hypervolume for a single generation
An interactive approach for EMO

Figure 2: Average hypervolume by generation - 30 runs with problems ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6.


Truss structural optimization
Truss structural optimization

- Finding an optimal design for a truss structure involves optimizing its **topology, size, and shape**.
- A truss design problem is usually **multimodal**, meaning that the problem offers multiple optimal designs in terms of topology and/or size of the members, but they are evaluated to have similar or equally good objective function values.

![Illustration of (a) 11-member, 6-node ground structure and (b), (c), and (d) its three different design solutions.](image)
Bilevel formulation of the truss problem

A bilevel formulation for the truss problem:

- We treat the topology optimization as the upper level optimization task, and the size and shape optimization as the lower level optimization task.
- The goal is to obtain multiple truss designs by considering both its topology and size simultaneously.

\[
\begin{align*}
\min_{\vec{x}_u \in X_u, \vec{x}_l \in X_l} \quad & F(\vec{x}_u, \vec{x}_l) \\
\text{s.t.} \quad & \vec{x}_l \in \arg\min_{\vec{x}_l \in X_l} \{f(\vec{x}_u, \vec{x}_l) : g_j(\vec{x}_u, \vec{x}_l) \leq 0, j = 1 \ldots J \} \\
\quad & G_k(\vec{x}_u, \vec{x}_l) \leq 0, k = 1 \ldots K,
\end{align*}
\]

We then can apply a **niching method** at the upper level, to obtain multiple designs in terms of topology as well as the size of the truss problem.

Truss solutions found by applying niching

Figure: Found truss solutions for 39-member, 12-node Ground structure found by applying niching to the upper level.


Take-home message …

• **Many challenges remain** when tackling real-world problems using EC methods;

• For practitioners, they are most interested in solving the problems at hand, NOT how well your methods perform on simple test functions;

• Real-world problems are far more challenging, and may require a combination of techniques in order to be effective;

• Important to study carefully the properties/characteristics of the problem under consideration, and try to incorporate the domain-specific knowledge into the design of the solution method;

• Rich ideas beyond just computer science; there are actually many others also do optimization, and we can learn a lot from them, e.g., many mature ideas in the operations research field;

• Do not be afraid of doing things differently; try NOT to follow the crowd;
Acknowledgement

• Support from our two ARC Discovery Grants (with international partner investigators Professor Kalyanmoy Deb and Professor Xin Yao).

• Professor Andreas Ernst from School of Mathematics, Monash University, for his extensive operations research expertise, and his significant role in research collaboration and PhD supervision support.

• Our PhD students, Angus Kenny, Kendall Taylor, Md Jakirul Islam, and Behrooz Ghasemishabankareh for their PhD works.

• Associate Professor Melih Ozlen (Math, RMIT), and Dr Jeffrey Chan for PhD supervision support.

• Help from Dr Yuan Sun, postdoc research fellow, supported by our 2018 ARC Discovery Grant.
Any questions?

Thank you