Are you a Hybrid? Yes, of course, everyone is a Hybrid nowadays!

Christian Blum

**Artificial Intelligence Research Institute (IIIA)**
**Spanish National Research Council (CSIC)**
IIIA-CSIC

CSIC: Spanish National Research Council
► Largest public institution dedicated to research in Spain (created in 1939)
► Third-largest in Europe
► 6% of all research staff in Spain work for the CSIC
► 20% of the scientific production in Spain is from the CSIC

IIIA: Artificial Intelligence Research Institute
► 20 tenured scientists (of three different ranks)
► Around 35 additional staff member (post-docs, PhD students, technicians, administration)
► Three research lines (machine learning, logic and constraint programming, multi-agent systems)
Topic of today: preparing the grounds

Optimization

Function optimization

Combinatorial optimization

Differentiable  Non-differentiable

Integer Programming
Mixed-Integer Linear Programming
Mixed-Integer Nonlinear Programming
Importance of combinatorial optimization problems

Note: Combinatorial optimization problems arise in numerous industrial settings

Examples

- Routing problems
- Assignment problems
- Scheduling problems
Algorithms for combinatorial optimization

- Algorithms
  - Exact Techniques
    - Dynamic programming
    - Branch & Bound
  - Approximate Techniques
    - Greedy Heuristics
    - Metaheuristics
      - Evolutionary Algorithms (Genetic Algorithms)
      - Ant Colony Optimization
      - Tabu Search
My research topics: algorithm-oriented (basic research)
What is swarm intelligence?

In a nutshell: **AI discipline** whose goal is designing intelligent multi-agent systems by taking **inspiration** from the **collective behaviour** of animal societies such as ant colonies, flocks of birds, or fish schools.
Swarm intelligence

**Properties:**

- Consist of a set of simple entities
- **Distributedness:** No global control
- **Self-organization** by:
  - **Direct communication:** for example, by visual or chemical contact
  - **Indirect communication:** Stigmergy (Grassé, 1959)

**Result:** Complex tasks/behaviors can be accomplished/exhibited in cooperation
Swarm Intelligence topics from last years

- **Combinatorial optimization:** adding negative learning to ant colony optimization  
  **Inspiration:** foraging behaviour of ant colonies

- **Distributed optimization:** graph coloring, independent set finding  
  **Inspiration:** self-desynchronization in Japanese tree frogs

- **Distributed problem solving:** duty-cycling in sensor networks  
  **Inspiration:** work-synchronization in ant colonies

More info: On my website

https://www.iiia.csic.es/~christian.blum/
Hybrid metaheuristics: definition

**Definition:** What is a hybrid metaheuristic?

- **Problem:** a precise definition is not possible/desirable

**Possible characterization:**

A technique that results from the combination of a metaheuristic with other techniques for optimization

**What is meant by:** other techniques for optimization?

- **Metaheuristics**
- **Branch & bound**
- **Dynamic programming**
- **Integer Linear Programming (ILP) techniques**
Hybrid metaheuristics: reason of being

Which Algorithm to use?

Problem instance size

Note: Hybrid algorithms that exploit the synergies between exact and approximate algorithms often excel in the context of large-scale problem instances
Hybrid metaheuristics: history

History:

- For a long time the different communities co-existed quite isolated.
- Hybrid approaches were developed already early, but only sporadically.
- Only since about 15 years the published body of research grows significantly:
  1. 1999: CP-AI-OR Conferences/Workshops
  2. 2004: Workshop series on Hybrid Metaheuristics
  3. 2006: Matheuristics Workshops

Consequence: The term hybrid metaheuristics identifies a separate line of research.
Example: decoder-based approaches (1)

General idea:

▶ Represent feasible solutions to the problem in an indirect way
▶ Design a decoder that translates such a solution into an actual solution

Advantages:

▶ This may transform a complex search space into one that is easier to handle
▶ Encapsulates problem complexity into the decoder

Note:

▶ Already very well known in the field of evolutionary algorithms
▶ Example: biased random key genetic algorithms (BRKGAs)
Example: decoder-based approaches (2)

Example: decoder-based approaches (3)

Option 1:

- Define a feasible solution as a set of nodes which contains exactly one node from each cluster
- Obtain an actual solution by applying Prim’s algorithm to a feasible solution

Option 2:

- Define a feasible solution as a spanning tree of the clusters
- Utilize a dynamic programming algorithm to know which node to choose from each cluster
Example: multi-level framework (1)

General idea:

► **First:** Iterative coarsening of the original problem instance

► **Then:** Find a solution to the coarsest level

► **Finally:** Iteratively refine this solution at each level
Example: multi-level framework (2)

The multi-level framework:
Example: multi-level framework (3)
Example: combinations with machine learning (1)

- **Machine learning:**
  - Reinforcement learning

- **Deep learning:**
  - Recurrent neural networks
  - Convolutional neural networks
Example: combinations with machine learning (2)

Note: machine learning may be used in (at least) two ways for improving algorithms for combinatorial optimization:

1. Generate a **fast approximation** for heavy computational tasks
2. **Learning missing expert knowledge** (for example a greedy function)
Example: combinations with machine learning (3)


Used methodology:

- Deep learning architecture structure2vec.
- Training: type of Q-learning

Results: The learned greedy policies ...

- outperform classical greedy algorithms for Minimum Vertex Cover, Maximum Cut and Traveling Salesman problem
- generalize to different problem instance sizes
Example: combinations with machine learning (4)

Note: Such a learned greedy function can be very interesting for problems for which no well-working greedy function is known.

Example: Capacitated minimum dominating set (CapMDS) problem

Classical MDS: Find the smallest subset of nodes such that each node of the graph (1) forms part of the subset or (2) is a neighbor of at least one node in the subset.
Example: combinations with machine learning (5)

CapMDS additional constraint: Each node can only cover a limited number of neighbors

![Graph 1](image1)

Optimal MDS solution

![Graph 2](image2)

Optimal CapMDS solution (when max. 2 neighbors can be covered)
More Detailed Example

Algorithms Based on Problem Instance Reduction

Main idea: apply an exact technique to sub-instances
Subset selection problems

Note: For simplicity, algorithms in this part are explained in the context of subset selection problems.

Keep in mind: These algorithms can be applied to any combinatorial problem.

Definition: Subset Selection Problem \((C, F, f)\)

1. \(C\) is a finite set of \(n\) items.
2. \(F : 2^C \mapsto \{\text{TRUE, FALSE}\}\) indicates for each subset \(S \subseteq C\) if it is feasible solution. Let \(X \subseteq 2^C\) be the set of all feasible solutions.
3. \(f : X \mapsto \mathbb{R}\) is an objective function that assigns a value to each feasible solution.

Note: Many well-known combinatorial optimization problems can be expressed as subset selection problems (Example: TSP).
Standard: Large Neighborhood Search (LNS)

- Small neighborhoods:
  1. **Advantage:** It is fast to find an improving neighbor (if any)
  2. **Disadvantage:** The average quality of the local minima is low

- Large neighborhoods:
  1. **Advantage:** The average quality of the local minima is high
  2. **Disadvantage:** Finding an improving neighbor might itself be $NP$-hard due to the size of the neighborhood

Ways of examining large neighborhoods:

- Heuristically

- **Exact techniques:** for example an ILP solver
Destruction-based large neighborhood search

1. Generate initial solution $S$
2. $S_{\text{partial}} := \text{Destroy } S \text{ partially}$
3. $S_{\text{ILP}} := \text{Apply exact solver to } S_{\text{partial}}$
4. $S := \text{Choose between } S \text{ and } S_{\text{ILP}}$
Alternative: Construct, Merge, Solve & Adapt (CMSA)

Construct, Merge, Solve & Adapt: A new general algorithm for combinatorial optimization

Christian Blum, Pedro Pinacho, Manuel López-Ibáñez, José A. Lozano

A comparative analysis of two matheuristics by means of merged local optima networks

Christian Blum, Gabriela Ochoa
Construct, Merge, Solve & Adapt: Flow Diagram

- **C**: complete set of items
- **C' ⊆ C**: sub-instance
- Set the age of all \( c \in C \) to zero

- Probabilistically generate \( n_a \) solutions
  - \( \hat{C} \): set of items used in these solutions
- \( C' := C' \cup \hat{C} \)
- \( S_{\text{ILP}} := \text{Apply exact solver to } C' \)

- Increment age of all \( c \in C' \)
  - Set age of all \( c \in S_{\text{ILP}} \) to zero

- Remove from \( C' \) all \( c \in C' \setminus S_{\text{ILP}} \)
  - whose age is equal to \( age_{\text{max}} \)
Test Case: Multi-dimensional Knapsack Problem (MDKP)

Given:

- A set of items $C = \{1, \ldots, n\}$
- A set of resources $K = \{1, \ldots, m\}$
- Of each resource $k$ we have a maximum quantity $c_k$ (capacity)
- Each item $i$ requires from each resource $k$ a certain quantity $r_{i,k}$
- Each item $i$ has a profit $p_i$

Valid solutions: Each subset $S \subseteq C$ is a valid solution if

$$\sum_{i \in S} r_{i,k} \leq c_k \quad \forall k \in K$$

Objective function (to be maximized): $f(S) := \sum_{i \in S} p_i$ for all valid $S$
MDKP: Greedy Heuristic

Re-ordering of the items: with respect to utility values

\[ u_i \leftarrow \frac{p_i}{\sum_{k \in K} r_{i,k} / c_k} \quad i \in C. \]

Simple heuristic:

- Consider the list of re-ordered items of left to right
- Add each item that does not violate the resource constraints to the partial solution under construction
MDKP: ILP Model

Standard model:

\[
\text{maximize} \quad \sum_{i \in C} p_i \cdot x_i \\
\text{subject to} \quad \sum_{i \in C} r_{i,k} \cdot x_i \leq c_k \quad \forall k \in K \\
\quad x_i \in \{0, 1\} \quad \forall i \in C
\]  

Application to sub-instances:

- **In LNS:** fix all \( x_i \) to one \( \forall i \in S_{\text{partial}} \).
- **In CMSA:** replace \( C \) by \( C' \)
MDKP: instance tightness

Important parameter: Instance tightness $0 \leq \alpha \leq 1$

- When $\alpha$ close to zero: capacities are low and valid solution only contain very few items
- When $\alpha$ close to one: capacities are very high and solutions contain nearly all items

Plan:

- Apply both LNS and CMSA to instances from the whole tightness range.
- Both algorithms are tuned with irace seperately for instances of each considered tightness.
MDKP: LNS/CMSA comparison (1)

- **X-axis:** instances with increasing tightness (from left to right)
- **Y-axis:** improvement of CMSA over LNS (in percent)
MDKP: LNS/CMSA comparison (2)

- **X-axis:** instances with increasing tightness (from left to right)
- **Y-axis:** improvement of CMSA over LNS (in percent)
MDKP: Comparative visualization of results

- Solution visited by CMSA
- Search transition of CMSA
- Solution visited by LNS
- Search transition of LNS
- Solution visited by both algorithms
- Solution at the start of a run (either algorithm)
- Solution at the end of a run (either algorithm)
- Best solution found across all runs and algorithms

Node Size Proportional to incoming weighted degree
MDKP: Comp. visualization, lowest tightness \((n = 10000)\)
MDKP: Comp. visualization, medium tightness \((n = 10000)\)
MDKP: Comp. visualization, highest tightness ($n = 10000$)
And in comparison to the state of the art? (1)
And in comparison to the state of the art? (2)

- **Benchmark:** 30 instances with 500 items and 10 resources from the OR-Library ([http://people.brunel.ac.uk/~mastjjb/jeb/info.html](http://people.brunel.ac.uk/~mastjjb/jeb/info.html)).
- **Comp. setup:** time limit 200 seconds, 100 runs per instance

**Avg. solution quality:**

<table>
<thead>
<tr>
<th></th>
<th>TPTEA</th>
<th>DQPSO</th>
<th>CMSA</th>
<th>LNS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>212840.70</td>
<td>212841.60</td>
<td><strong>212842.03</strong></td>
<td>212841.73</td>
</tr>
</tbody>
</table>

**Avg. computation time:**

<table>
<thead>
<tr>
<th></th>
<th>TPTEA</th>
<th>DQPSO</th>
<th>CMSA</th>
<th>LNS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3467.04</td>
<td>120.26</td>
<td><strong>76.71</strong></td>
<td>108.06</td>
</tr>
</tbody>
</table>
Summary and Possible Research Directions

Summary: Hybrid metaheuristics

- **Reason of being:** especially due to the large scale of many real-life problems
- **Advantage:** they profit from the synergies obtained by combining different techniques
- **Therefore:** many optimization algorithms nowadays are hybrids

Possible Research Directions:

- **Regarding existing approaches:** contributing to their deeper understanding and their improvement
- **Regarding new approaches:** especially the combination with machine learning seems to offer opportunities
Thanks to colleagues involved in my research on hybrids

Pedro Pinacho  
Günther R. Raidl  
Gabriela Ochoa
Questions?

Recommended literature:
