

Hybrid Metaheuristics

An overview of the integration of metaheuristics and tree search

Hybrid Metaheuristics

An overview of the integration of metaheuristics and tree search

DEIS, Campus of Cesena
Alma Mater Studiorum - Università di Bologna
andrea.roli@unibo.it

DEIS, Campus of Cesena
Alma Mater Studiorum - Università di Bologna
andrea.roli@unibo.it

DEIS, Campus of Cesena
Alma Mater Studiorum - Università di Bologna
andrea.roli@unibo.it



Hybrid metaheuristics (1)

- The focus of research on metaheuristics has shifted from an algorithm-oriented point of view to a problem-oriented point of view
- Research aims at combining different algorithmic components for the design of algorithms
- HM denotes all the combinations of metaheuristics components and metaheuristics with AI and OR techniques

Hybrid metaheuristics (2)

- Open nature of locution ‘hybrid metaheuristics’ helps creative thinking and the exploration of new research directions
- HM aim at exploiting advantages of the individual pure strategies
- Goal: combining complementary strengths (*synergy*)

Essential suggested bibliography

- C. Blum, M. Blesa, A. Roli and M. Sampels (eds.). Hybrid Metaheuristics: An Emerging Approach to Optimization. Springer, 2008.
- P. Van Hentenryck and M. Milano (eds.), Hybrid Optimization: the 10 years of CPAIOR. Springer, 2010.
- V. Maniezzo, T. Stützle, S. Voß. Matheuristics: Hybridizing metaheuristics and mathematical programming. Springer, 2009.

Outline

- 1 Hybridization of Metaheuristics with (Meta-)Heuristics
- 2 Hybridization of Metaheuristics with tree search and CP
 - Large neighborhood search
 - ACO & CP
 - ACO-Beam search
 - Solution-Guided Multi-Point Constructive Search
- 3 On the relation between complete and incomplete search
 - Climbing discrepancy search
 - Incomplete dynamic backtracking search
- 4 HM design and engineering guidelines
- 5 Bibliography

Outline

- 1 Hybridization of Metaheuristics with (Meta-)Heuristics
- 2 Hybridization of Metaheuristics with tree search and CP
 - Large neighborhood search
 - ACO & CP
 - ACO-Beam search
 - Solution-Guided Multi-Point Constructive Search
- 3 On the relation between complete and incomplete search
 - Climbing discrepancy search
 - Incomplete dynamic backtracking search
- 4 HM design and engineering guidelines
- 5 Bibliography

Metaheuristics ‘self’ hybridization

- Notable examples in population-based techniques combined with local search (e.g., memetic algorithms and ACO)
- Hyper-heuristics
- Population-based iterated local search
- Multi-level techniques

Outline

- 1 Hybridization of Metaheuristics with (Meta-)Heuristics
- 2 Hybridization of Metaheuristics with tree search and CP
 - Large neighborhood search
 - ACO & CP
 - ACO-Beam search
 - Solution-Guided Multi-Point Constructive Search
- 3 On the relation between complete and incomplete search
 - Climbing discrepancy search
 - Incomplete dynamic backtracking search
- 4 HM design and engineering guidelines
- 5 Bibliography

Metaheuristics combined with tree search and CP

Main approaches for the integration of metaheuristics and tree search:

- 1 Metaheuristics are applied before tree search, providing a valuable input, or vice versa
- 2 Metaheuristics, mainly local search, use tree search to efficiently explore the neighborhood of the current solution
- 3 Tree search applies a metaheuristic in order to improve a solution (i.e., a leaf of the tree) or a partial solution (i.e., an inner node)
- 4 Metaheuristic concepts can be used to obtain incomplete but efficient tree exploration strategies

Large neighborhood search

- LNS: a family of techniques in which local search uses tree search/CP for exploring a, typically very large, neighborhood
- LNS tries to combine the advantage of a large neighborhood with an exhaustive but fast tree search/CP exploration
- LNS was first proposed by Shaw in [20] and similar ideas were presented in [1, 16, 17].

Large neighborhood search

- The core principle consists in viewing the exploration of a neighborhood as the solution of a sub-problem
- In other words,
 - the neighborhood is defined as a sub-problem
 - and tree search exhaustively explores it

Example: VRP

LNS example of high-level scheme

Input: Problem $\mathcal{P}(X, D, C, f)$ defined by variables $X = \{x_1, \dots, x_n\}$, domains $D = \{D_1, \dots, D_n\}$, constraints $C = \{c_1, \dots, c_m\}$, objective function f to be minimized

Output: Best solution found

$s \leftarrow \text{InitialSolution}();$ let $s \equiv \{(x_1 = v_1), \dots, (x_n = v_n)\}$

while termination condition not met **do**

$K \leftarrow \text{Select}(n, k)$

 Let $\mathcal{P}_{|K}$ be the problem in which variables x_i with $i \in K$ are free
 and $x_i = v_i$ if $i \notin K$

$s' \leftarrow \text{SolveCP}(\mathcal{P}_{|K})$

if $f(s') < f(s)$ **then**

$s \leftarrow s'$

end if

end while

Observations (1)

- Crucial part of the algorithm is the choice of the sub-problem (the k variables in the example)
- This choice is usually made on the basis of both heuristic and stochastic criteria
- The sub-problem/neighborhood size should be large enough to diversify the search, but the complexity of solving it should be rather low

Observations (2)

- When CP is applied to extend a partial assignment into a complete one, the propagation of constraints is usually effective and deep, due to the variables that are already instantiated
- In general, more elaborated local search strategies than first improvement can be used
- Also mathematical programming techniques can be used as inner solvers. See [7] and [8] for a survey.

ACO & CP

- ACO and CP are constructive techniques with complementary strengths:
 - ACO is characterized by a (reinforcement) learning capability
 - CP is efficient in handling constraint

► News: recent book on ACO & CP by Christine Solnon [21].

Integration of ACO and CP

Several ways for integrating ACO and CP. In this overview we will consider two of them:

- 1 CP used as solution construction method for artificial ants
- 2 ACO used for initializing variable/value heuristics in CP

CP inside ACO

Main idea: artificial ants build solutions by using CP in which variable and value heuristics are the ones of ACO.

- 1 Useful in ‘strongly’ constrained optimization problems
- 2 Instead of penalizing infeasible solutions or hard-coding constraint checks, CP is used
- 3 The solving process consists of iteratively building a set of solutions by CP and updating pheromone values

CP-with-ACO high-level algorithm

Input: Variables $x_1, \dots, x_n$, domains $D_1, \dots, D_n$, constraints $C_1, \dots, C_m$

Output: A feasible assignment, optimal w.r.t. a given objective function or **nil** if no feasible solution exists

Setup initial domains for variables $x_1, \dots, x_n$

Post initial constraints

while search not completed **do**

for each ant **do**

 Propagate_And_Label($x_1, \dots, x_n$)

end for

 Update_Probabilistic_Model()

if new best solution found **then**

 Post upper bound constraint

end if

end while

Observations

- 1 Alternative perspective: CP uses reinforcement learning for updating variable/value heuristics
- 2 In general, reinforcement learning can also be used to tune other algorithm parameters, such as the order of constraints to propagate, when to propagate, when to restart.

More details also in [14], in which an application to a machine scheduling problem with sequence-dependent setup times is illustrated.

Two phases scheme

Main idea: ACO is run before CP so as to initialize the weights in value heuristics

Two phases algorithm:

- 1 ACO is performed and the final pheromone matrix is saved
- 2 CP solver performs a complete search and uses the pheromone matrix as a heuristic information for value selection

Observations

- Besides ACO, other mechanisms can be used to initialize heuristic weights.
E.g., sampling, diving, evolutionary computation techniques
- General scheme: first phase with a metaheuristics, second phase with CP/tree search.

ACO-Beam search

Disadvantages of heuristic greedy construction mechanisms:

- might mislead search
(see also Discrepancy search algorithms)
- usually, no search space reduction mechanisms—such as bounding—are used

→ B&B mechanisms used inside greedy construction.

See [6, 11, 12, 13, 3, 4, 10].

ACO-Beam search

Main ideas:

- Allow the extension of partial solutions in parallel, in potentially more than one way
- Non-independent and parallel probabilistic construction of several solutions at each iteration
- Two types of information used: greedy and bounding information

Beam search

- Breadth-first search in which only a limited number of partial paths is kept, at most k_{bw}
- At each level, at most μ extensions of these paths are generated
- Only the *best* k_{bw} are kept in memory
- Children nodes are evaluated on the basis of bounding information
- k_{bw} is said the *beam width*

Probabilistic beam search

- Partial solutions are expanded by means of a probabilistic procedure into μ extensions (at most)
- At most k_{bw} extensions are chosen on the basis of bounding information

Beam-ACO

- Solution construction in ACO is PBS (i.e., not just a simple heuristic construction)
- After all the artificial ants have built a solution, the pheromone values are updated

See [3, 6]

Observations

- Partial solution extension is based on ACO transition probabilities and heuristic
- Solutions are built in parallel and not independently

Solution-Guided Multi-Point Constructive Search

- Framework for combining any tree search/CP technique with search characteristics typical for metaheuristics
- Makes it possible for tree search to exploit intensification/diversification metaheuristic mechanisms

Solution-Guided Multi-Point Constructive Search

Main working:

- A set of elite solutions E is built; solutions in E will be used to guide tree search
- One elite solution is randomly chosen and used as a reference for the value heuristic of the search process
- Elite set E updated

SGMPCS high-level algorithm

initialize elite solution set E

while termination criteria unmet **do**

if $\text{rand}[0, 1) < p$ **then**

 set upper bound on cost function; set fail limit, l

$s \leftarrow \text{search}(\emptyset, l)$

if $s \neq \emptyset \wedge s$ is better than $\text{worst}(E)$ **then**

 replace $\text{worst}(E)$ with s

end if

else

$r \leftarrow$ randomly chosen element of E

 set upper bound on cost function; set fail limit, l

$s \leftarrow \text{search}(r, l)$

if $s \neq \emptyset \wedge s$ is better than r **then**

 replace r with s

end if

end if

end while

return $\text{best}(e)$

Observations

- Initial elite set is composed of solutions returned by one or multiple runs of a metaheuristic algorithm
- Basic inspiration as previous methods: guide tree search by means of information collected by a metaheuristic technique
- Despite its simplicity the technique is quite effective (see [2])

Further examples

- F. Focacci, F. Laburthe and A. Lodi. Local Search and Constraint Programming: LS and CP illustrated on a transportation problem. In M. Milano (ed.), *Constraint and Integer Programming. Towards a Unified Methodology*, Kluwer Academic Publishers, 2004.
- M. Dell'Amico, and A. Lodi, On the integration of metaheuristic strategies in constraint programming. In C. Rego and B. Alidaee (eds.), *Metaheuristic Optimization via Memory and Evolution*, Springer, 2002.

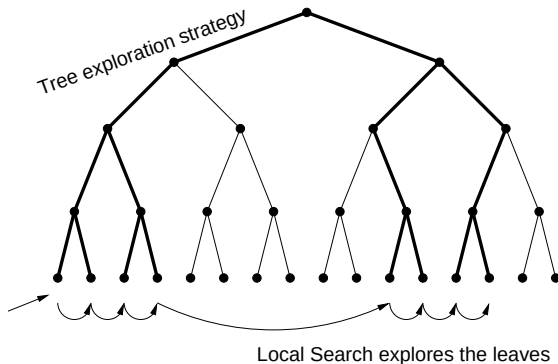
Outline

- 1 Hybridization of Metaheuristics with (Meta-)Heuristics
- 2 Hybridization of Metaheuristics with tree search and CP
 - Large neighborhood search
 - ACO & CP
 - ACO-Beam search
 - Solution-Guided Multi-Point Constructive Search
- 3 On the relation between complete and incomplete search**
 - Climbing discrepancy search
 - Incomplete dynamic backtracking search
- 4 HM design and engineering guidelines
- 5 Bibliography

Complete tree search vs. incomplete local search

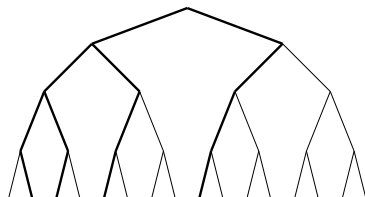
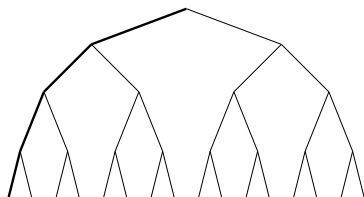
- Each type of search beats the other on some problem classes
- Local search exploits gradients in the search space, whilst tree search usually does not
- Local search can not prove unsatisfiability (with the exception of few special cases)
- The supposed disadvantage of incompleteness is often illusory: very large problems are too hard to be solved to optimality
- Local search usually does not make use of a *structured* memory, but it rather exploits search history statistics

A pictorial view



Climbing discrepancy search

- **Discrepancy search** explores a (binary) search tree by means of a heuristic that suggests which branch to take at each decision point
- It explores the leaves (solutions) in increasing values of *discrepancy* (k) from the *reference solution*



Climbing discrepancy search

Possible weakness of DS: it does not take into account the objective function and the correlation among solution values

A way for handling this drawback:

- Exploit correlation among states in the search landscape
- For example, as soon as a better solution is found it is accepted as reference solution

See [15].

Generalized CDS high-level algorithm

```
 $k \leftarrow 1$  { $k$  is the discrepancy}  
 $s^* \leftarrow \text{InitialSolution}()$  { $s^*$  is the reference solution proposed by the  
heuristics}  
while Termination condition not met do  
   $s' \leftarrow \text{Choose}(\text{Leaves}(s^*, k), \text{history})$  {Choose a leaf at  
  discrepancy  $k$  from  $s^*$ }  
  if  $\text{Accept}(s')$  then {if the new solution is accepted}  
     $s^* \leftarrow s'$  {the new solution substitutes the reference one}  
     $k \leftarrow 1$   
  end if  
   $k \leftarrow k + 1$   
end while
```

Observations

- Depending on the implementation of functions
Choose($Leaves(s^*, k), history$) and Accept(s') variants can
be designed
E.g., simulated annealing-like CDS

Incomplete dynamic backtracking search

- Performs a local search over feasible partial assignments
- Objective: minimizing the distance from a complete feasible assignment (i.e., maximizing the length of the partial feasible assignment)

See [18, 19].

Incomplete dynamic backtracking search

Input: Variables $x_1, \dots, x_n$, domains $D_1, \dots, D_n$, constraints $C = \{c_1, \dots, c_m\}$

Output: A feasible assignment A or **nil** if no feasible solution exists

$A \leftarrow \emptyset$; $V \leftarrow \{x_1, \dots, x_n\}$

while $V \neq \emptyset$ **do**

$v_i \leftarrow \text{random-member}(V)$

$d \leftarrow \text{heuristic-choice}(D_i)$ such that $v_i = d$ is consistent with A under C

if $d = \text{null}$ **then**

for $i = 1, \dots, \min(b, |A|)$ **do**

$(v_j = d') \leftarrow \text{random-member}(A)$

$A \leftarrow A \setminus \{v_j = d'\}; V \leftarrow V \cup \{v_j\}$

end for

else

$A \leftarrow A \cup \{v_i = d\}; V \leftarrow V - \{v_i\}$

end if

end while

Return A

Observations

- Parameter b tunes the amount of ‘diversification’
- Variants of IDB search depending on the criteria for choosing unassigned variables and values and variables from the partial assignment
- Similarities with Local Search with Constraint Propagation and Conflict-Based Heuristics [9]

Outline

- 1 Hybridization of Metaheuristics with (Meta-)Heuristics
- 2 Hybridization of Metaheuristics with tree search and CP
 - Large neighborhood search
 - ACO & CP
 - ACO-Beam search
 - Solution-Guided Multi-Point Constructive Search
- 3 On the relation between complete and incomplete search
 - Climbing discrepancy search
 - Incomplete dynamic backtracking search
- 4 HM design and engineering guidelines
- 5 Bibliography

Design guidelines

- 1 Clearly define goals and requisites of the solving task
- 2 Collect all the possible pieces of information on the problem at hand
- 3 Choose the solving method
- 4 Follow an engineering approach
- 5 Make use of scientific testing

Some questions to answer

- Is optimality important?
- Is computation time an issue?
- Do I need an *anytime* solver?
- Will be the problem model subjected to changes and refinements?

Some erroneous assertions

‘Hybrid metaheuristics are always *better* than any pure technique’

Some erroneous assertions

‘Hybrid metaheuristics are always *better* than any pure technique’

‘The problem is NP-hard, so we have to resort to incomplete algorithms’

Some erroneous assertions

‘Hybrid metaheuristics are always *better* than any pure technique’

‘The problem is NP-hard, so we have to resort to incomplete algorithms’

‘Complete techniques are always better than incomplete ones’

Some erroneous assertions

‘Hybrid metaheuristics are always *better* than any pure technique’

‘The problem is NP-hard, so we have to resort to incomplete algorithms’

‘Complete techniques are always better than incomplete ones’

‘Incomplete techniques are always better than complete ones’

Outline

- 1 Hybridization of Metaheuristics with (Meta-)Heuristics
- 2 Hybridization of Metaheuristics with tree search and CP
 - Large neighborhood search
 - ACO & CP
 - ACO-Beam search
 - Solution-Guided Multi-Point Constructive Search
- 3 On the relation between complete and incomplete search
 - Climbing discrepancy search
 - Incomplete dynamic backtracking search
- 4 HM design and engineering guidelines
- 5 Bibliography



D. Applegate and W Cook.

A computational study of the job-shop scheduling problem.
ORSA Journal on Computing, 3(2):149–156, 1991.



J. Christopher Beck.

Solution-guided multi-point constructive search for job shop scheduling.
Journal of Artificial Intelligence Research, 29:49–77, 2007.



C. Blum.

Beam-ACO—hybridizing ant colony optimization with beam search: an application to open shop scheduling.
Computers and Operations Research, 32:1565–1591, 2005.



C. Blum.

Beam-ACO for simple assembly line balancing.

INFORMS Journal on Computing, 20(4):618–627, 2008.



C. Blum, M. J. Blesa Aguilera, A. Roli, and M. Sampels, editors.

Hybrid Metaheuristics – An Emerging Approach to Optimization, volume 114 of *Studies in Computational Intelligence*.

Springer Verlag, Berlin, Germany, 2008.



C. Blum, C. Cotta, A. J. Fernández, and J. E. Gallardo. A probabilistic beam search algorithm for the shortest common supersequence problem.

In C. Cotta and J. I. van Hemert, editors, *Proceedings of EvoCOP 2007 – Seventh European Conference on Evolutionary Computation in Combinatorial Optimisation*, volume 4446 of *Lecture Notes in Computer Science*, pages 36–47. Springer Verlag, Berlin, Germany, 2007.



C. Blum, J. Puchinger, G. R. Raidl, and A. Roli.

Hybrid metaheuristics.

In M. Milano and P. van Hentenryck, editors, *Hybrid Optimization: 10 Years of CPAIOR*. Springer Verlag, Berlin, Germany, 2010.

In press.



M. Chiarandini, I. Dumitrescu, and T. Stützle.

Very large-scale neighborhood search: Overview and case studies on coloring problems.

In Blum et al. [5], pages 117–150.



N. Jussien and O. Lhomme.

Local search with constraint propagation and conflict-based heuristics.

Artificial Intelligence, 139:21–45, 2002.



M. López-Ibáñez and C. Blum.

Beam-ACO for the travelling salesman problem with time windows.

Computers & Operations Research, 37(9):1570–1583, 2010.



V. Maniezzo.

Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem.

INFORMS Journal on Computing, 11(4):358–369, 1999.



V. Maniezzo and A. Carbonaro.

An ANTS heuristic for the frequency assignment problem.

Future Generation Computer Systems, 16:927–935, 2000.



V. Maniezzo and M. Roffilli.

Very strongly constrained problems: an ant colony optimization approach.

Cybernetics and Systems, 39(4):395–424, 2008.



B. Meyer and A. Ernst.

Integrating ACO and constraint propagation.

In M. Dorigo et al., editors, *Proceedings of ANTS 2004 – Fourth International Workshop on Ant Colony Optimization and Swarm Intelligence*, volume 3172 of *Lecture Notes in Computer Science*, pages 166–177. Springer Verlag, Berlin, Germany, 2004.



M. Milano and A. Roli.

On the relation between complete and incomplete search: an informal discussion.

In N. Jussien and F. Laburthe, editors, *Proceedings of CP-AI-OR'02 – Fourth Int. Workshop on Integration of AI*

and OR techniques in Constraint Programming for Combinatorial Optimization Problems, pages 237–250, 2002.



G. Pesant and M. Gendreau.

A view of local search in Constraint Programming.

In E. Freuder, editor, *Principles and Practice of Constraint Programming - CP'96*, volume 1118 of *Lecture Notes in Computer Science*, pages 353–366. Springer-Verlag Berlin Heidelberg, Germany, 1996.



G. Pesant and M. Gendreau.

A Constraint Programming Framework for Local Search Methods.

Journal of Heuristics, 5:255–279, 1999.



S. Prestwich.

Combining the scalability of local search with the pruning techniques of systematic search.

Annals of Operations Research, 115:51–72, 2002.



S. Prestwich.

The Relation Between Complete and Incomplete Search, chapter 3, pages 63–83.

Volume 114 of Blum et al. [5], 2008.



P. Shaw.

Using constraint programming and local search methods to solve vehicle routing problems.

In M. Maher and J.-F. Puget, editors, *Principle and Practice of Constraint Programming – CP98*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431. Springer Verlag, Berlin, Germany, 1998.



C. Solnon.

Ant Colony Optimization and Constraint Programming.
Wiley-ISTE, 2010.