Diplomarbeitspräsentationen der Fakultät für Informatik

Ant Colony Optimization for Tree and Hypertree Decompositions

Thomas Hammerl

FAKULTÄT FÜR !INFORMATIK

Masterstudium:
Software & Internet Computing

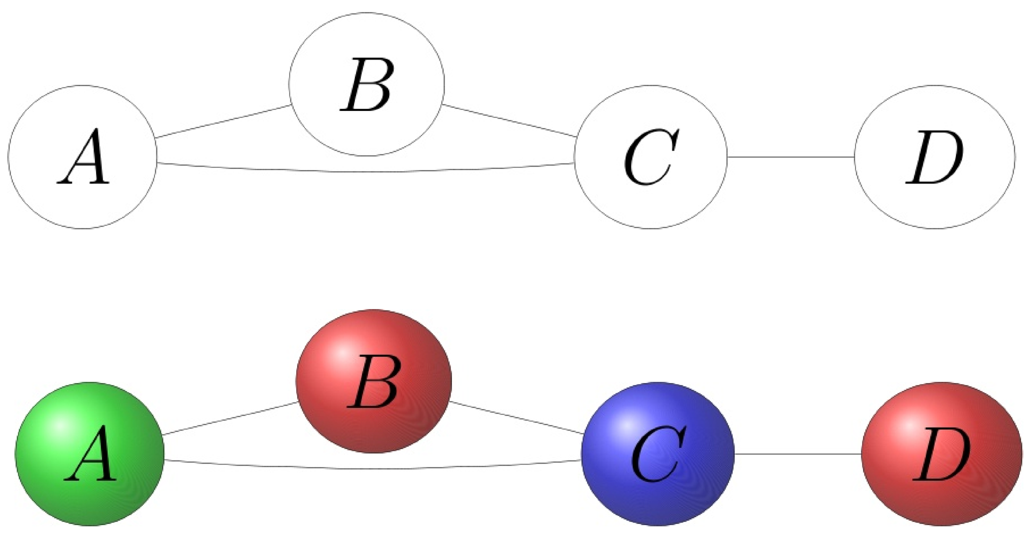
Technische Universität Wien
Institut für Informationssysteme
Abteilung für Datenbanken und Artificial Intelligence
Betreuer: Priv.-Doz. Dr. Nysret Musliu

Constraint Satisfaction Problems

Many instances of constraint satisfaction problems can be solved efficiently if they are representable as a tree respectively generalized hypertree decomposition of small width.

For instance, one such constraint satisfaction problem is the graph coloring problem:

Find a coloring for a given graph using just three colors such that no two adjacent vertices share the same color.

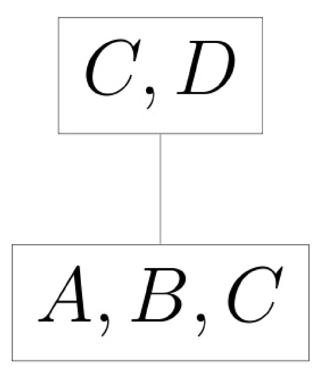


Variables: A, B, C, D

Domains: red, green, blue for each variable

Constraints: $A \neq B$, $A \neq C$, $B \neq C$, $C \neq D$

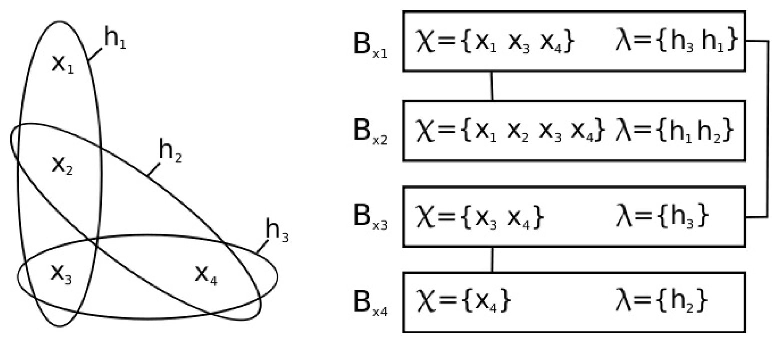
Tree and Hypertree Decompositions



Tree Decomposition

A (hyper)tree decomposition divides the original problem into subproblems. The smaller the biggest subproblem the more efficiently the original problem can be solved. A decomposition can be obtained from a so-called elimination ordering.

An elimination ordering is an ordering of the vertices of the constraint graph (e.g. [A,D,C,B]).

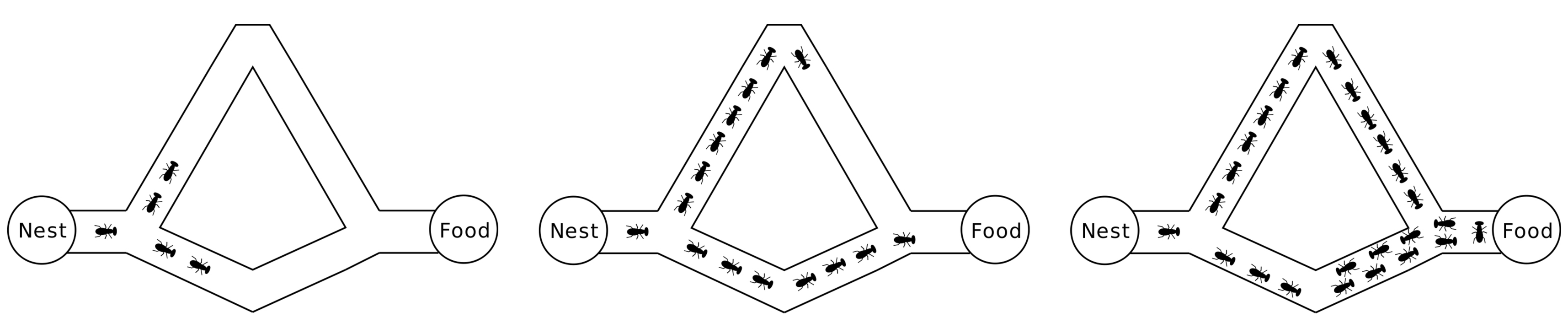


Hypergraph and a possible Hypertree Decomposition.

It is guaranteed that there is an elimination ordering that results in the optimal tree respectively hypertree decomposition.

Ant Colony Optimization (ACO)

The "Double Bridge Experiment":



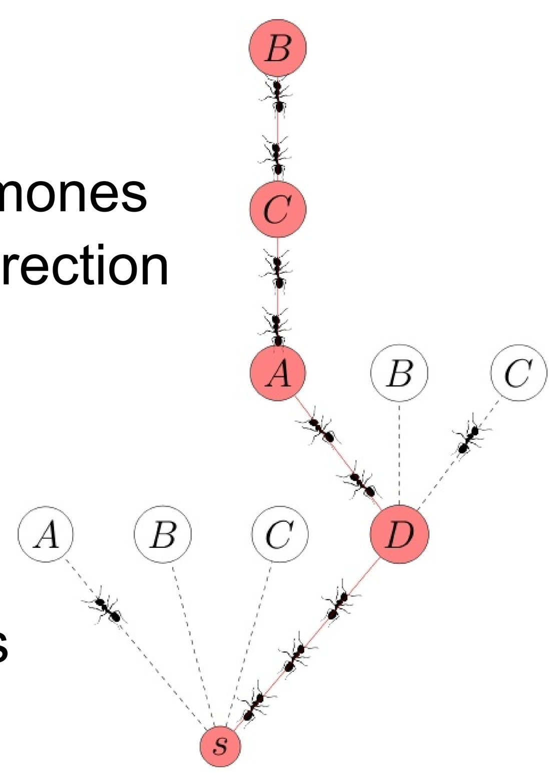
a) Ants start exploration.

b) Ants on shorter branch reach food source.

c) Ants more likely to return on shorter branch.

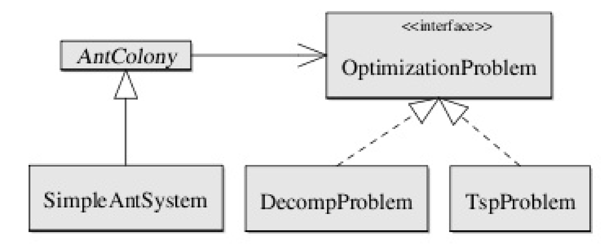
- Nature-inspired algorithm
- Ants find shortest path by depositing pheromones
- Pheromone → ants more likely to choose direction

- Adoption of principle for (hyper)tree decomposition
- Artificial ants construct elimination orderings



A C++ Library for the ACO Metaheuristic: libaco

- Library of ant algorithms
- Five different ACO variants
- Solves combinatorial problems
- Open Source License (LGPL)
- http://code.google.com/p/libaco/



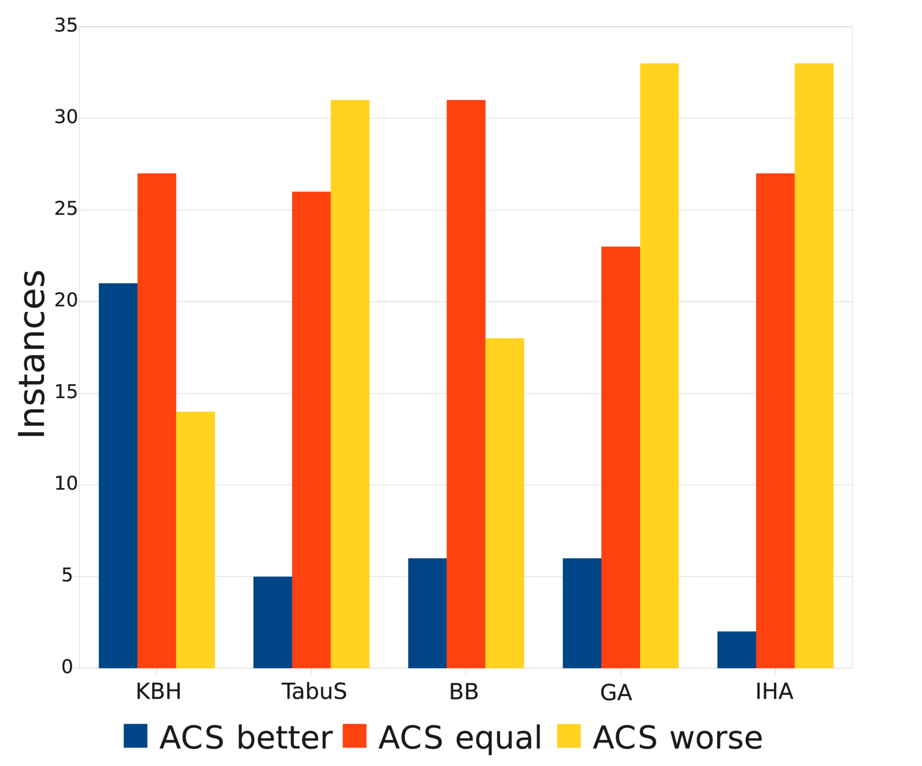

```
void main() {
    AntColonyConfiguration config;
    TspProblem problem = create_tsp_problem();
    SimpleAntColony colony(tsp, config);
    colony.run(); // run one iteration
    std::vector<unsigned int> tour = colony.get_best_tour();
    double length = colony.get_best_tour_length();
}
```

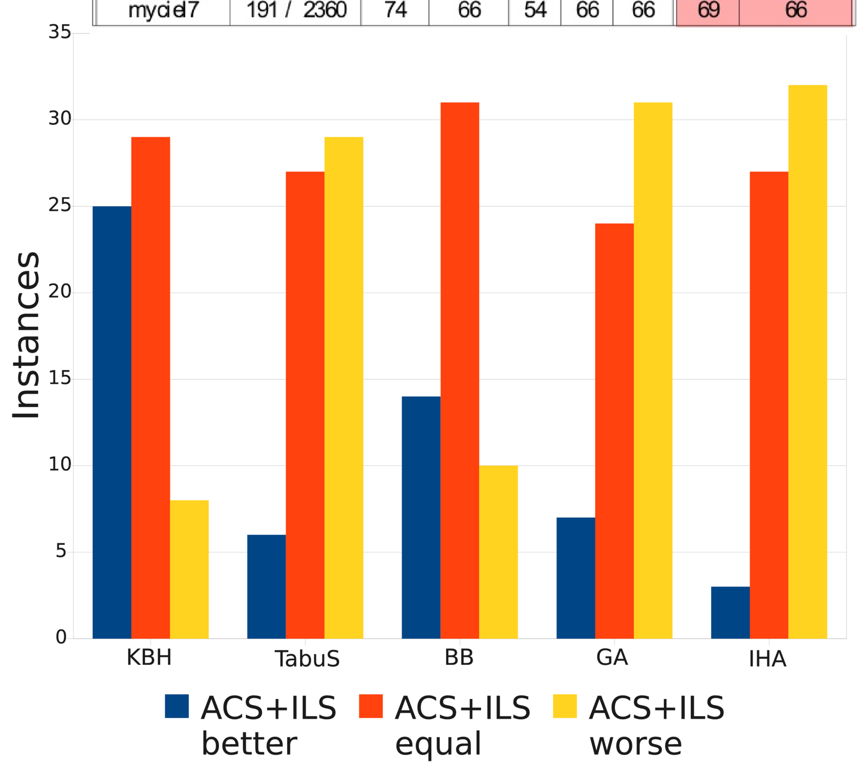
Computational Results

- Tree Decomposition
  - Results for 62 DIMACS instances
  - Two ACO Algorithms:
    - Ant Colony System (ACS)
    - Hybridization with Local Search
  - New upper bound for homer.col
  - Optimal results for many instances

Instance	|V|/ |E|	KBH	TabuS	BB	GA	IHA	ACS	ACS+ILS
anna	138 / 986	12	12	12	12	12	12	12
david	87 / 812	13	13	13	13	13	13	13
huck	74 / 602	10	10	10	10	10	10	10
homer	561 / 3258	31	31	31	31	31	31	30
jean	80 / 508	9	9	9	9	9	9	9
games120	120 / 638	37	33	-	32	32	37	37
queen5_5	25 / 160	18	18	18	18	18	18	18
queen6_6	36 / 290	26	25	25	26	25	25	25
queen7_7	49 / 476	35	35	35	35	35	35	35
queen8_8	64 / 728	46	46	46	45	45	47	46
queen9_9	81 / 1056	59	58	59	58	58	60	59
queen10_10	100 / 1470	73	72	72	72	72	75	73
queen11_11	121 / 1980	89	88	89	87	87	92	89
queen12_12	144 / 2596	106	104	110	104	103	110	109
queen13_13	169 / 3328	125	122	125	121	121	132	128
queen14_14	196 / 4186	145	141	143	141	140	154	150
queen15_15	225 / 5180	167	163	167	162	162	178	174
queen16_16	256 / 6320	191	186	205	186	186	206	201
fpsol2.i.1	269 / 11654	66	66	66	66	66	66	66
fpsol2.i.2	363 / 8691	31	31	31	32	31	31	31
fpsol2.i.3	363 / 8688	31	31	31	31	31	31	31
inithx.i.1	519 / 18707	56	56	56	56	56	56	56
inithx.i.2	558 / 13979	35	35	31	35	35	31	31
inithx.i.3	559 / 13969	35	35	31	35	35	31	31
miles1000	128 / 3216	49	49	49	50	49	52	50
miles1500	128 / 5198	77	77	77	77	77	77	77
miles250	125 / 387	9	9	9	10	9	9	9
miles500	128 / 1170	22	22	22	24	22	25	25
miles750	128 / 2113	37	36	37	37	36	38	38
mulsol.i.1	138 / 3925	50	50	50	50	50	50	50
mulsol.i.2	173 / 3885	32	32	32	32	32	32	32
mulsol.i.3	174 / 3916	32[1]	32	32	32	32	32	32
mulsol.i.4	175 / 3946	32	32	32	32	32	32	32
mulsol.i.5	176 / 3973	31	31	31	31	31	31	31
myciel3	11 / 20	5	5	5	5	5	5	5
myciel4	23 / 71	11	10	10	10	10	10	10
myciel5	47 / 236	20	19	19	19	19	19	19
myciel6	95 / 755	35	35	35	35	35	35	35
myciel7	191 / 2360	74	66	54	66	66	69	66

- Hypertree Decomposition
  - Results for 19 instances of the CSP Hypergraph Library
  - ACS+Set Cover Algorithm





Contributions

- ACO algorithms for Tree and Hypertree Decomposition.
- Hybridization of algorithms with local search methods:
  - Iterated Local Search
  - Hill Climbing
- Proposal of two pheromone update strategies.
- Incorporation of two guiding heuristics:
  - Min-Fill
  - Min-Degree
- Implementation of stagnatation measures:
  - Average Lamda Branching Factor
  - Variation Coefficient
- Comparison of ACO with other decomposition techniques.
- Improved upper bound for DIMACS instance homer.
- A C++ library for the ACO metaheuristic: libaco

Kontakt: thomas.hammerl@gmail.com