Optimization by unconventional ant algorithms

Nicolas Zufferey
GSEM – University of Geneva
Switzerland

UNIVERSITÉ DE GENÈVE
GENEVA SCHOOL OF ECONOMICS AND MANAGEMENT

References (papers)

- D. Schindl, N. Zufferey, *Solution Methods for Fuel Supply of Trains*, **Information Systems and Operational Research** 51 (1), 22 – 29, 2014
- N. Zufferey, *Metaheuristics: some Principles for an Efficient Design*, **Computer Technology and Applications** 3 (6), 446 – 462, 2012
- N. Zufferey, *Optimization by Ant Algorithms: Possible Roles for an Individual Ant*, **Optimization Letters** 6 (5), 963 – 973, 2012
- M. Plumettaz, D. Schindl, N. Zufferey, *Ant Local Search and its Efficient Adaptation to Graph Colouring*, **Journal of the Operational Research Society** 61, 819 – 826, 2010

CHAPTERS

A. Graph **coloring** problem

B. Location-distribution problem in a **railway** network

C. Order acceptance and **scheduling**

D. Enlargement of the ant **paradigm**

CHAPTER (A)

Graph Coloring

Part I

Constructive Ant Systems

CAS

Constructive Ant System

While a time limit is not reached, do:

For i = 1 to N, do:

(1) **build** a solution with ant i

(2) let s_i be the resulting solution

Update the trails:

- by the use of a subset of $\{s_1, \ldots, s_N\}$

Ant = constructive heuristic

(Dorigo, 1992)

How to build a solution s_i with ant i?

At each step: add an element to the current partial solution

Each decision (or move) **m** is based on:

- **Greedy force GF(m)**: short term profit
- **Trail Tr(m)**: history of the search, collaboration between ants

Updating the trail system

(at the end of each generation)

$$Tr(m) = \rho \, Tr(m) + \Delta Tr(m)$$

- $\rho \in]0, 1[$ — **evaporation** coefficient
- $\Delta Tr(m)$ — **reinforcement** term

$$\Delta Tr(m) = \sum_{k \in A} Tr_k(m)$$

- A = **all** the ants of the current generation / **best** ants of the current generation

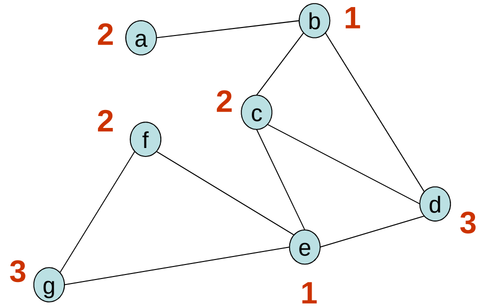
Selection of a move

- Compute the **probability** of each move m
- **Normalize** GF and Tr
- **Tune** parameters α and β

$$p_k(m) = \frac{GF(m)^\alpha \cdot Tr(m)^\beta}{\sum_{m' \in M_k(adm)} GF(m')^\alpha \cdot Tr(m')^\beta}$$

Graph Coloring Problem (GCP)

- **k-coloring** give a color c(x) to each vertex x where c(x) ∈ {1, …, k}
- **conflict** if c(x) = c(y) and x linked to y
- **GCP** find a **conflict-free** k-coloring with the **smallest** possible k



CAS for graph coloring
Costa and Hertz (1997)

- **ANT-DSAT** algorithm
- **Role** of each ant: constructive heuristic
  (select a vertex + assign a color)
- **Trail system**: matrix Tr(x,y), proportional to:
  - the <u>number of times</u> vertices x and y have the same color in the solutions provided by the ants
  - the <u>quality</u> of the solutions where c(x) = c(y)

Part II

Ant Decision Systems

ADS

Ant Decision System

Generalization of (Hertz & Zufferey, 2006)

Initialization:

- Generate an **initial** solution s

While a time limit is not reached, do:

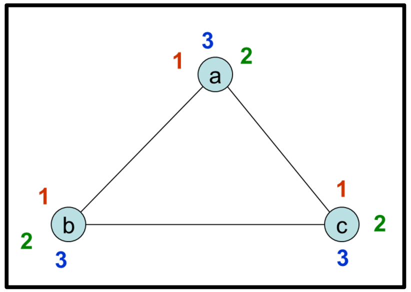
(1) **Some** ants **modify** the solution s
(let D be the set of associated decisions)

(2) **Update the trails** by the use of D

ADS for k-GCP

(Hertz & Zufferey, 2006)

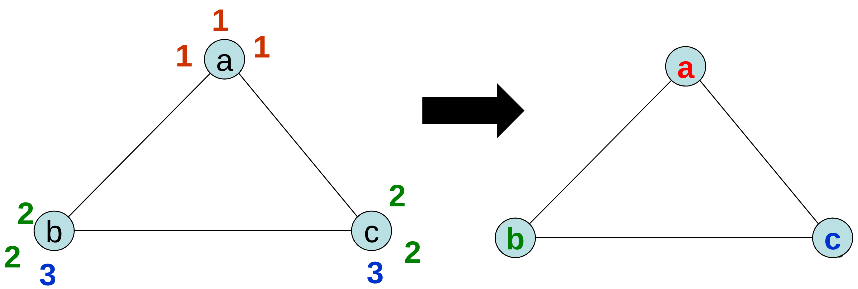
- **Role** of each ant: contribute to give a color to a vertex
- **Fixed k strategy (conflicts)**
- **Associate**
  - a color in {1, …, k} with each ant
  - k ants to each vertex
- **Initial distribution of the ants**
  - put one ant of each color on each vertex
- **Move:** change the distribution of the ants
  - sequentially exchange the position of two ants of different colors, which are located on two vertices x and y



Coloring procedure: at each step

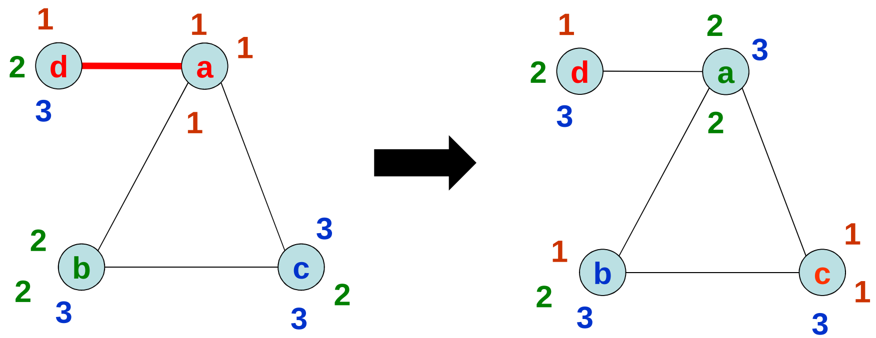
(1) **Select** a non colored vertex x

(2) **Assign** color c to vertex x:
- c must be represented on x by at least one ant of color c
- break ties with c minimizing the number of conflicts
- break ties with c which is the most represented on x
- break ties randomly



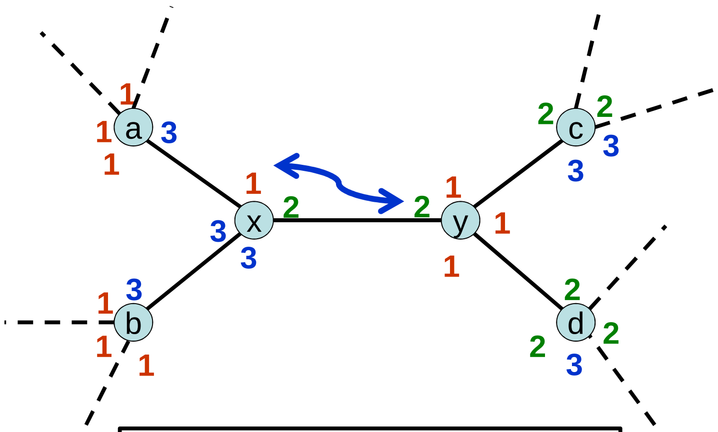
Goal of an iteration: remove a conflict

change the color of vertex a
→ remove color 1 from a



1 conflict **0 conflict**

Greedy forces &Trails



Exchange (x, 1) with (y, 2)
Reinforce trails: (x, 2) and (y, 1)
Remove trails: (x, 1) and (y, 2)

Part III

Ant Local Search

ALS

Ant Local Search

(Plumettaz, Schindl & Zufferey, 2010)

While a time limit is not reached, do:

For i = 1 to N, do:

(1) apply the **local search** associated with ant i

(2) let s_i be the resulting solution

Update the trails:

- by the use of a subset of $\{s_1, \ldots, s_N\}$

Tabu search

(Glover, 1986)

- Start from an **initial** solution s
- A **neighbor** solution s' is generated from the **current** solution s by performing a **move**
- A **tabu list** is used to forbid the reverse of a recently performed move
- At each iteration, go to the **best non tabu** solution

Tabu search for k-GCP

(Bloechliger & Zufferey, 2008)

Solution space: legal partial k-coloring

- s — $(C_1, C_2, \ldots, C_k;$ **OUT**$)$
- C_i — set of vertices with color i (conflict free)
- **OUT** — set of uncolored vertices

Objective function: size of **OUT**

Tabu search for k-GCP

(Bloechliger & Zufferey, 2008)

Move $m = (v, C_i)$

move a vertex v from **OUT** to C_i

put in **OUT** every vertex of C_i adjacent to v

ALS for k-GCP: greedy force

move $m = (v, C_i)$

GF(m): number of vertices in C_i that will be uncolored by v

ALS for k-GCP: trail system

Based on “Friendship” between pairs { x, y } of vertices

Fs(x, y) = $|C_i|^2$ if $x,y \in C_i$ (same color) (0 otherwise)

Part IV

Results

Compared methods

DSAT (Brélaz, 1979)
- constructive method with restarts

CAS (Costa & Hertz, 1997)
- each ant is a DSAT-algorithm with a trail system
- trail(x, y) indicates if it is a good idea to have color(x) = color(y)

ADS (Hertz & Zufferey, 2006)
- guided DSAT at each iteration

ALS (Plumettaz, Schindl & Zufferey, 2010)
- each ant is a tabu search like (Bloechliger & Zufferey, 2008)
- trail(x, y) indicates if it is a good idea to have color(x) = color(y)

Benchmark graphs

(time limit = 1 hour)

Graph	n	density	OPT	BEST	DSAT	CAS	ADS	ALS
DSJC1000.1	1000	0.1	?	20	25	29	25	20
DSJC1000.5	1000	0.5	?	83	112	122	104	86
DSJC1000.9	1000	0.9	?	224	293	313	255	225
DSJC500.1	500	0.1	?	12	15	17	15	12
DSJC500.5	500	0.5	?	48	62	68	56	48
DSJC500.9	500	0.9	?	126	158	167	135	127
DSJR500.1c	500	0.97	?	85	87	97	88	85
DSJR500.5	500	0.47	?	122	129	136	130	125
flat1000_50_0	1000	0.49	50	50	111	120	101	50
flat1000_60_0	1000	0.49	60	60	111	121	102	60
flat1000_76_0	1000	0.49	76	82	111	120	103	85
flat300_28_0	300	0.48	28	28	39	43	36	29
le450_15c	450	0.17	15	15	23	28	18	15
le450_15d	450	0.17	15	15	23	28	18	15
le450_25c	450	0.17	25	25	28	33	29	26
le450_25d	450	0.17	25	25	28	33	29	26

Conclusions

- **Ranking** — CAS < DSAT < ADS < ALS ≈ BEST
- **DSAT > CAS** — the way to **select a decision** in CAS is cumbersome, which lead to a slow method
- **ADS > CAS** — an ant as a **decision helper** could be better than an ant as a **constructive heuristic**
- **ALS ≈ BEST** — an ant as a **local search** leads to the best results

CHAPTER (B)

Location-distribution problem in a railway network

(Schindl & Zufferey, 2014)

Introduction

- Optimize the **refueling costs** of a fleet of locomotives over a railway network
- One source of fuel:
  - fueling trucks, located at yards
- Determine a solution
  - Satisfying all the constraints
  - Minimizing the costs

Motivation

After the 2010 INFORMS optimization contest

Two type of decisions

- Choose the number of trucks contracted at each yard

→ **truck assignment problem** → **HEURISTICS**

- Determine the refueling plan of each locomotive
  (i.e. the quantity of fuel that must be dispensed into each locomotive at every yard)

→ **fuel distribution problem** → **FLOW MODEL**

Constraints

- The capacity of the tank of each locomotive is limited
- The maximum amount of fuel a truck can provide the same day is limited
- It is forbidden to run out of fuel
- A locomotive cannot be refueled at its destination yard
- There is a maximum number of times (which is two) a train can stop to be refueled (excluding the origin)

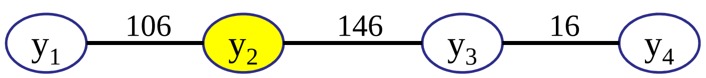
Costs

- Weekly **operating** cost of each fueling truck
- **Fuel** price per gallon associated with each yard

  (which can vary from yard to yard because of the differences in distribution, marketing costs and other factors)
- **Fixed** cost associated with each refueling

A train schedule

Train	Yards	Sequence	Day of Journey	Station Type
t_1	y_1	1	1	Origin
	y_2	2	1	Intermediate
	y_3	3	1	Intermediate
	y_4	4	1	Destination
t_2	y_4	1	1	Origin
	y_2	2	1	Intermediate
	y_1	3	1	Destination



3.25 $/g   3.05 $/g   3.15 $/g   3.15 $/g

Illustration: beginning of the solution

	Locomotive z_1				Locomotive z_2			
Stop No.	**Yard**	**Station**	**Day**	**Gallons**	**Yard**	**Station**	**Day**	**Gallons**
1	y_1	Or.	1	0	y_4	Or.	1	0
2	y_2	Int.	1	1,870	y_2	Int.	1	0
3	y_3	Int.	1	0	y_1	Or.	2	0
4	y_4	Or.	2	0	y_2	Int.	2	0
5	y_2	Int.	2	0	y_3	Int.	2	0
6	y_1	Or.	3	0	y_4	Or.	3	0
7	y_2	Int.	3	4,500	y_2	Int.	3	4,500
8	y_3	Int.	3	0	y_1	Or.	4	0
9	y_4	Or.	4	0	y_2	Int.	4	0
10	y_2	Int.	4	0	y_3	Int.	4	0
11	y_1	Or.	5	0	y_4	Or.	5	0
12	y_2	Int.	5	0	y_2	Int.	5	0
13	y_3	Int.	5	0	y_1	Or.	6	0
14	y_4	Or.	6	0	y_2	Int.	6	0
15	y_2	Int.	6	3,010	y_3	Int.	6	0
16	y_1	Or.	7	0	y_4	Or.	7	0
17	y_2	Int.	7	0	y_2	Int.	7	0
18	y_3	Int.	7	0	y_1	Or.	8	0
19	y_4	Or.	8	0	y_2	Int.	8	4,494
20	y_2	Int.	8	0	y_3	Int.	8	0
21	y_1	Or.	9	0	y_4	Or.	9	0

Illustration: end of the solution

	Locomotive z_1				Locomotive z_2			
Stop No.	Yard	Station	Day	Gallons	Yard	Station	Day	Gallons
21	y_1	Or.	9	0	y_4	Or.	9	0
22	y_2	Int.	9	0	y_2	Int.	9	0
23	y_3	Int.	9	0	y_1	Or.	10	0
24	y_4	Or.	10	0	y_2	Int.	10	0
25	y_2	Int.	10	**3,752**	y_3	Int.	10	0
26	y_1	Or.	11	0	y_4	Or.	11	0
27	y_2	Int.	11	0	y_2	Int.	11	**386**
28	y_3	Int.	11	0	y_1	Or.	12	0
29	y_4	Or.	12	0	y_2	Int.	12	0
30	y_2	Int.	12	0	y_3	Int.	12	0
31	y_1	Or.	13	0	y_4	Or.	13	0
32	y_2	Int.	13	0	y_2	Int.	13	**3,752**
33	y_3	Int.	13	0	y_1	Or.	14	0
34	y_4	Or.	14	0	y_2	Int.	14	0
35	y_2	Int.	14	0	y_3	Int.	14	0

Illustration: evaluation of the solution

- Fuel costs — **80,105.5$** — (26,264 gallons)
- Truck costs — 8,000$ — (4000 $/week)
- Stop costs — 2,000$ — (8 x 250 $)
- **Total costs** — 90,105.5$

Descent local search for TAP

Move

- **Add** a truck to a yard y
- **Drop** a truck from a yard y
- Evaluation: flow algorithm

One iteration

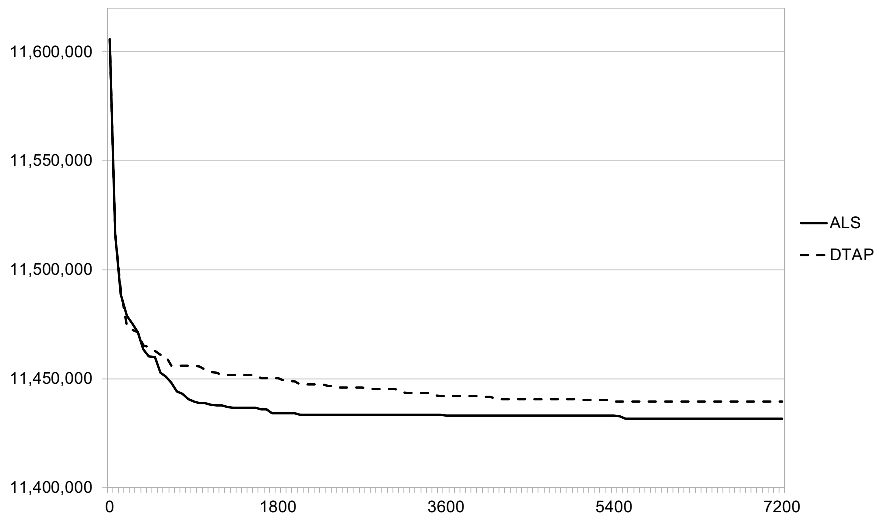
- Evaluate a random set of 5 add moves
- Evaluate a random set of 5 drop moves
- Perform the best of these 10 moves

ALS for the TAP

- Each ant — descent local search
- **Move (x→s)** — add/drop a truck on yard x of solution s
- **GF(x→s)** — objective function (flow algorithm)
- Trail tr(x,y) — proportional to the quality of the solutions with trucks on yards x and y
- **TR(x→s)** — proportional to the tr(x,y)'s with y in s
- **Decision** — perform the best GF move among the 10 best TR moves.

ALS vs Descent

(120 minutes)



Performance of ALS

- 0.23% above a **lower bound**.
- Better than the 3rd team of the INFORMS contest.
- ALS is a **flexible** metaheuristics and can be adapted to nonlinear instances, extensions, etc.
- If several trucks from the same company are contracted for the same yard, the company is likely to propose **discounted prices** for that yard.

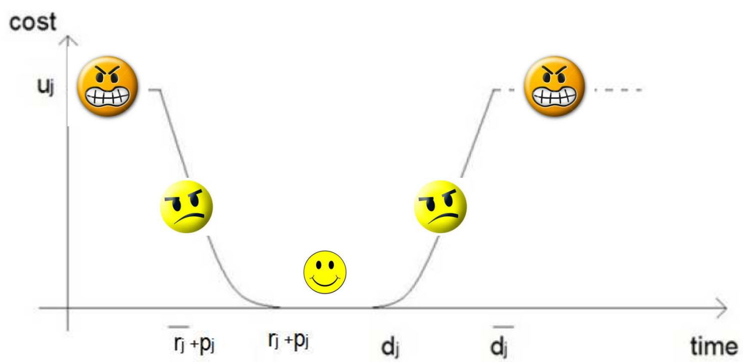
CHAPTER (C)

Order acceptance and scheduling

(Thevenin, Zufferey & Widmer, 2013)

Description of the problem

- Schedule n jobs on one machine.
- Setup times/costs.
- Earliness and tardiness penalties (linear/quadratic).



Solution representation

Solutions are **modeled** by:

- a sequence of jobs: 1-2-3-4
- a rejected set: { 5, 6 }

Timing algorithm

- Compute the completion time of each job
- With **regular** cost functions: **ASAP** rule.
- With **non regular** cost functions: inserting **idle time** can decrease the costs.

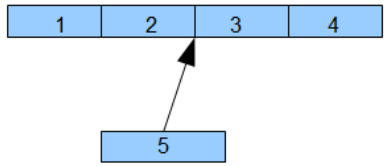
CAS

GF(m) = resulting objective function value

Tr(x, y) based on:

- the number of times job x was processed before job y during the search
- the quality of the associated solutions

Tr(m)
= Tr(1, 5) + Tr(2, 5)
+ Tr(5, 3) + Tr(5, 4)



CAS: selection of a move

CAS

$$p_k(m) = \frac{GF(m)^\alpha \cdot Tr(m)^\beta}{\sum_{m' \in M_k(adm)} GF(m')^\alpha \cdot Tr(m')^\beta}$$

CAS-WP among the **q moves** leading to the best greedy forces,
perform the one associated with the **best trail**
→ **sequential** use of GF and Tr

Tabu search

Joint use of 4 different **types** of move

Add a job
Drop a job
Swap two jobs
Reinsert a job

While the solution is not feasible, drop the job whose removal leads to the min cost.

ALS

- Each **ant** is a tabu search
- Same definitions for GF and Tr
- Every 20 moves of tabu search, an insertion is made by **taking into account the trails** (as in CAS-WP)

Results (10 runs, stop after 30n seconds)

n	α	type	Best	Greedy	Tabu	CAS	CAS-WP	ALS	GA
50	1	LL	40878	71.06	19.36	253.99	10.86	7.91	4.01
		LQ	73879	41.31	7.28	155.38	13.17	0.49	0.92
		QL	86042	22.64	14.02	110.00	5.90	0.68	0.90
		QQ	144888	9.46	0.84	69.98	8.61	0.73	0.68
		Mix	67057	49.96	11.07	181.00	32.95	5.96	7.08
	2	LL	9157	298.58	11.25	1074.55	9.42	6.11	4.71
		LQ	6381	208.30	5.80	2090.34	20.13	0.00	0.00
		QL	7673	194.77	0.19	1452.85	45.94	1.05	0.92
		QQ	26424	64.26	9.60	709.82	96.06	10.30	9.53
		Mix	6133	165.96	13.05	2167.45	69.47	13.25	1.68
	0.5	LL	67911	43.33	5.42	125.77	7.71	4.77	2.45
		LQ	125170	14.85	9.48	60.93	4.20	0.09	0.20
		QL	150014	8.03	5.75	41.85	5.15	0.03	0.03
		QQ	235732	8.09	0.00	13.31	0.59	0.00	0.00
		Mix	137559	6.47	0.53	69.29	11.27	0.41	0.30
Average				**80.47**	**7.58**	**571.77**	**22.76**	**3.45**	**2.23**

CHAPTER (D)

Enlargement of the Ant Paradigm

(Zufferey, 2014)

Part I

What **defines** an ant algorithm?

Evolutionary algorithm

- A **population** of N ants
- Each ant is able of **self adaptation** (independently of the other ants)
- The ants are able to **collaborate** (exchange information)
- At each **generation**, solutions are provided based on the ants activity
- Output: **best** encountered solution

Two main ingredients

How to select a decision m at time t?

Greedy force GF(m)

- **Short term profit**
- Also called: visibility, heuristic information

Trail system Tr(m)

- Information obtained from the **other ants** (history of the search)
- Large value if m was **often** performed in previous **good** solutions

Definition of a generation

Ant	**Constructive heuristic**	**Decision helper**	**Local search**
Generation	Each ant builds a solution	D decisions have been performed	Each ant provides a solution
Method	Constructive Ant System (**CAS**)	Ant Decision System (**ADS**)	Ant Local Search (**ALS**)

Part II

Selection of a decision

Most used formula

$$p_k(m) = \frac{GF(m)^\alpha \cdot Tr(m)^\beta}{\sum_{m' \in M_k(adm)} GF(m')^\alpha \cdot Tr(m')^\beta}$$

Cumbersome

- Compute the probability of **each** decision m
- GF and Tr have to be **normalized**
- Parameters α and β have to be **tuned**

Observations

- GF and Tr are **jointly** used
- Often used in CAS and ADS

Alternative technique

(1) Select a set **D** of decisions with the largest **greedy force** values
(2) Then, select in **D** the decision with the largest **trail** value

Observations

- GF and Tr are **sequentially** used
- The size of **D** is a sensitive **parameter**
- Used in ALS

Part III

Which is the best ant algorithm?

Only my opinion…

Considered problems

- **Graph coloring problem**
  (Plumettaz & Schindl & Zufferey, 2010), (Zufferey, 2012)
- **Job scheduling with abandon costs**
  (Zufferey, 2012)
- **Location-distribution problem in a railway network**
  (Schindl & Zufferey, 2014)
- **Order acceptance and scheduling problem**
  (Thevenin & Zufferey & Widmer, 2013)
- **Truck loading problem**
  (Respen & Zufferey, 2014)

Evaluation of an algorithm

- **Efficiency** quality of the obtained results
- **Speed** time needed to get good results
- **Robustness** sensitivity to variations in problem characteristics and data quality
- **Flexibility** ability to take advantage of the problem structure
- **Simplicity** ease of adaptation

Difficult to design a solution method having a good **overall** performance

CAS: performance

CAS

\+ **Local search techniques**

→ Competitive results

Evaluation of ant algorithms

	CAS	ADS	ALS	TS
Efficiency	-3	-1	+1	0
Speed	-3	-2	-1	0
Robustness	+1	-1	+1	0
Flexibility	0	0	0	0
Simplicity	-1	-2	-1	0
SCORE	-6	-6	0	0