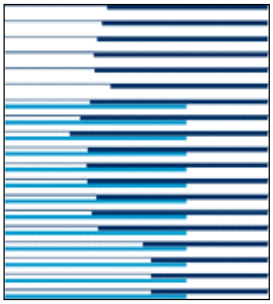
# The Sequential Ordering Problem

**Roberto Montemanni**

Dalle Molle Institute for Artificial Intelligence

University of Applied Sciences of Southern Switzerland

Email: roberto@idsia.ch Tel: +41 58 666 666 7



## Contributors

- D. Anghinolfi, University of Genoa, IT
- G. Di Caro, IDSIA, CH
- L.M. Gambardella, IDSIA, CH
- F. Gomez, IDSIA, CH
- M. Mojana, University of Lugano, CH
- M. Paolucci, University of Genoa, IT
- A.E. Rizzoli, IDSIA, CH
- D.H. Smith, University of Glamorgan, UK
- N.E. Toklu, IDSIA, CH
- D. Weyland, IDSIA, CH

# Outline

- Introduction
- Problem description
- A Genetic Algorithm
- A Hybrid Ant System
- A Heuristic Manipulation Technique
- A Particle Swarm Optimization
- An Enhanced Ant Colony System
- A Shared Incumbent Environment
- Bibliography

# Outline

- Introduction
- Problem description
- A Genetic Algorithm
- A Hybrid Ant System
- A Heuristic Manipulation Technique
- A Particle Swarm Optimization
- An Enhanced Ant Colony System
- A Shared Incumbent Environment
- Bibliography

# Real world applications


## Freight transportation

Escudero L.F., Guignard M., Malik K. [1994]. *A Lagrangean relax-and-cut approach for the sequential ordering problem with precedence relationships*, Annals of Operations Research, 50: 1219-237.

# Real world applications

## Crane scheduling in port terminals

Montemanni, R., Smith, D.H., Rizzoli, A.E. and Gambardella, L.M. [2009]. *Sequential Ordering Problems for Crane Scheduling in Port Terminals*, International Journal of Simulation and Process Modelling 5(4): 348–361.

# Real world applications

## Flexible manufacturing systems

Ascheuer, N. [1995]. *Hamiltonian path problems in the online optimization of flexible manufacturing systems*, PhD Thesis, Technische Universität Berlin.

# Real world applications

## Helicopter scheduling

Timlin M.T., Pulleyblank W.R. [1992]. *Precedence constrained routing and helicopter scheduling: heuristic design*, Interfaces 22(3): 100-111.

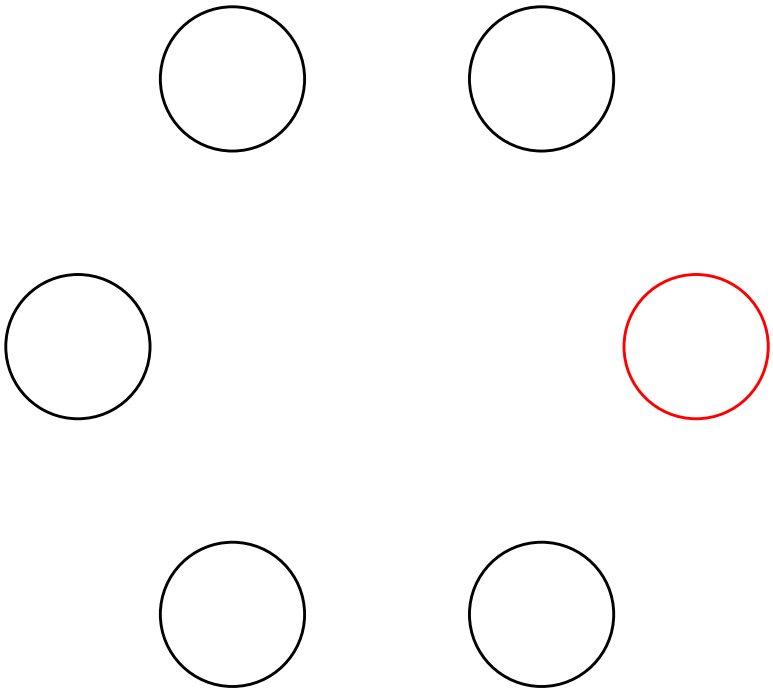
## Real world applications

**Automotive paint shops**

Spieckermann, S., Gutenschwager, K. and Voß, S. [2004]. *A sequential ordering problem in automotive paint shops*, International Journal of Production Research 42(9): 1865–1878.

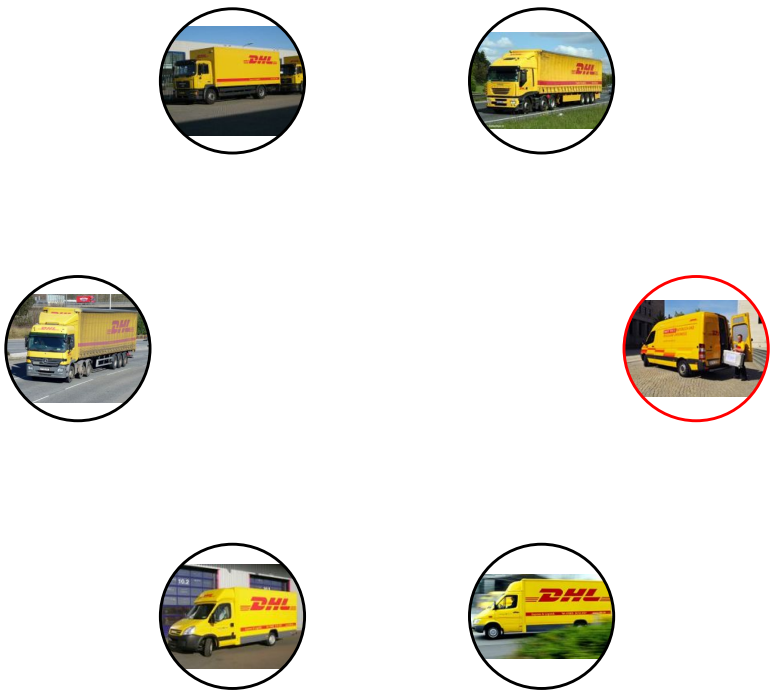
# Outline

## Informal problem description



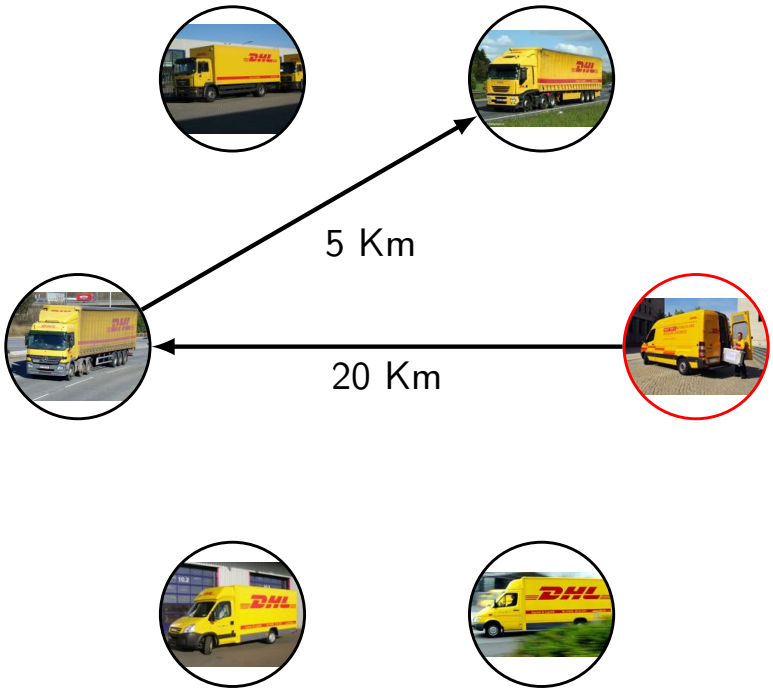
- Nodes = customers

## Informal problem description



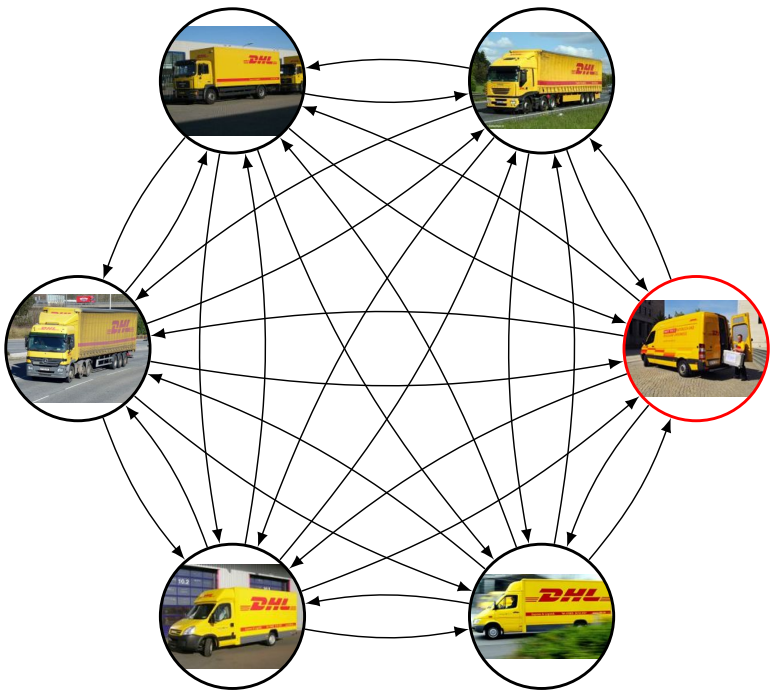
- Nodes = customers

# Informal problem description



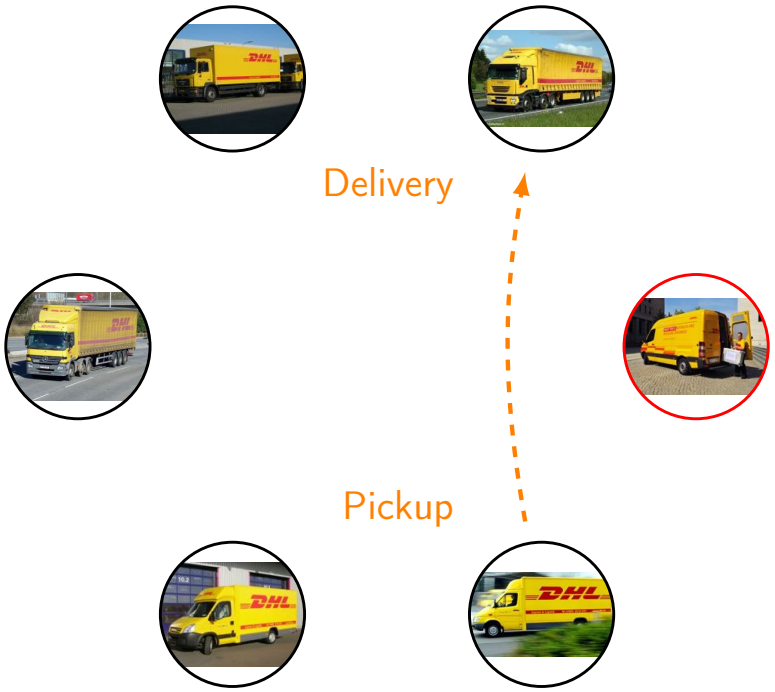

- Nodes = customers
- Arcs = travel times

# Informal problem description



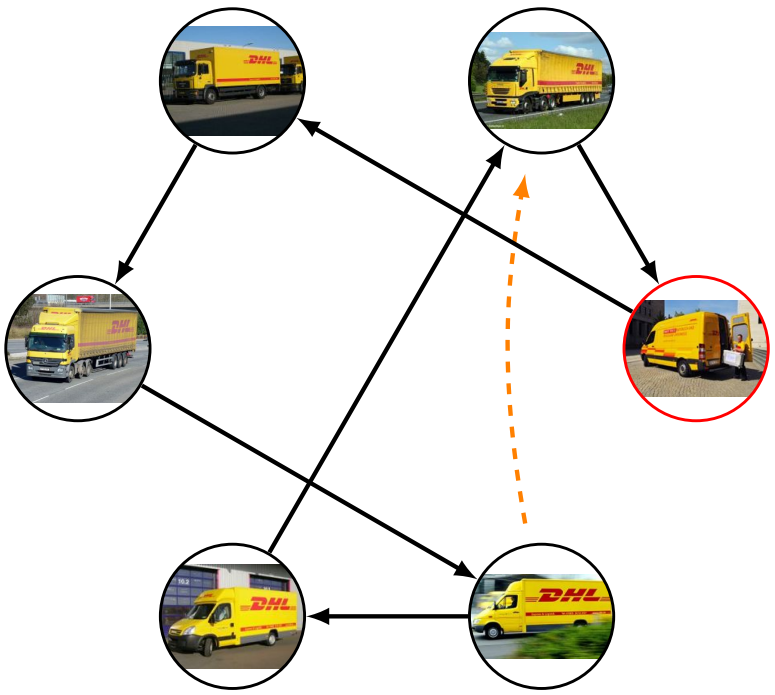
- Nodes = customers
- Arcs = travel times
- Complete digraph

## Informal problem description



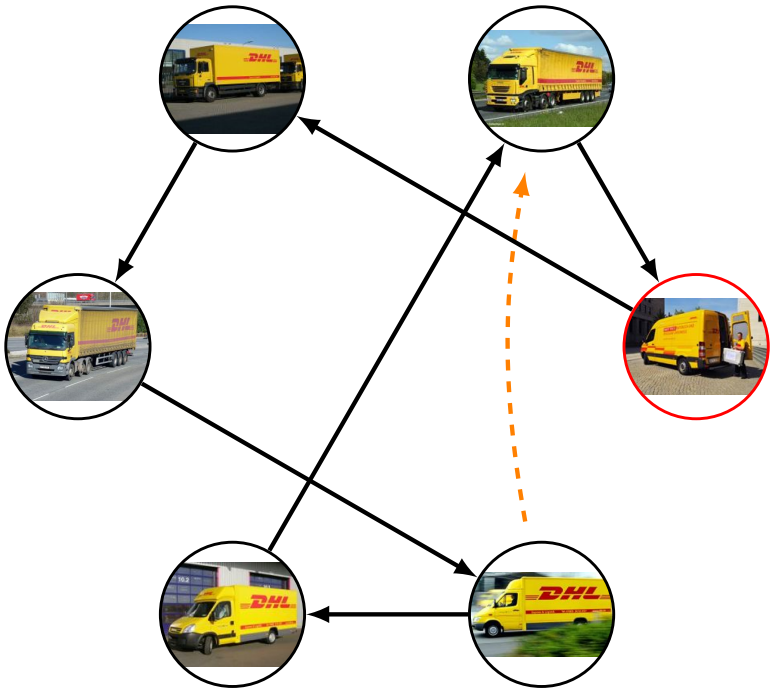

- Nodes = customers
- Arcs = travel times
- Complete digraph
- Precedences

## Informal problem description



- Nodes = customers
- Arcs = travel times
- Complete digraph
- Precedences
- Solution = tour

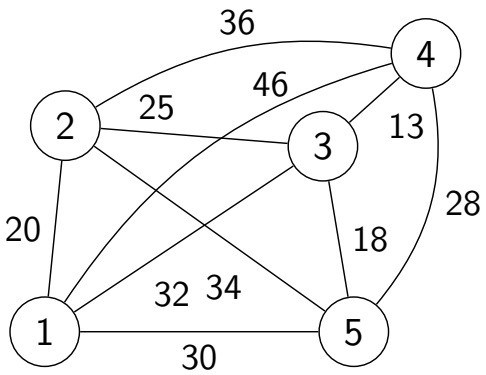
# Informal problem description



- Nodes = customers
- Arcs = travel times
- Complete digraph
- Precedences
- Solution = tour
- NP-Hard

# Formal problem description

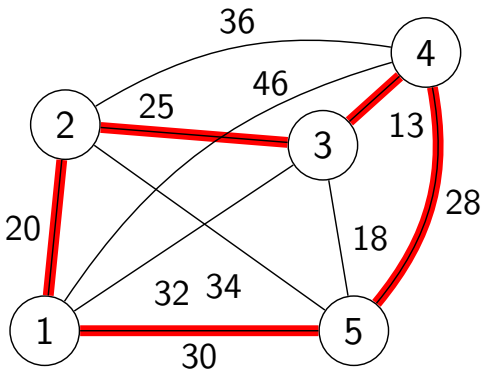
- Given: set of cities + pairwise distances




TSP

## Formal problem description

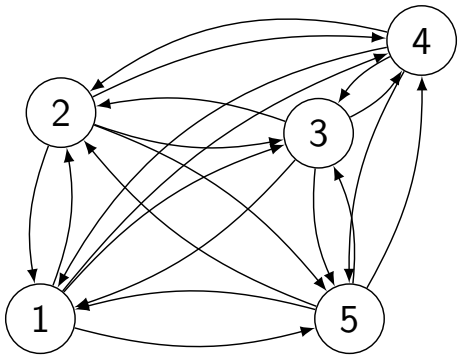
- Given: set of cities + pairwise distances
- Required: shortest tour




TSP

## Formal problem description

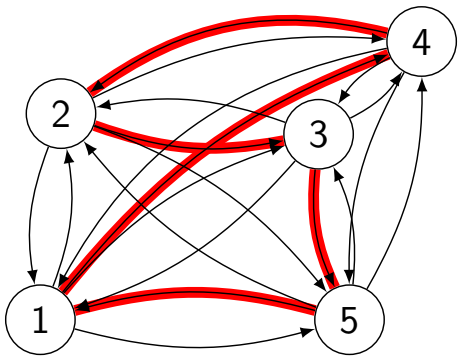
- Given: set of cities + asymmetric distances




TSP ATSP

## Formal problem description

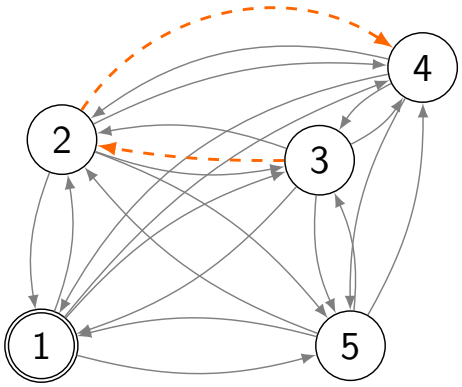
- Given: set of cities + asymmetric distances
- Required: shortest tour




TSP ATSP

## Formal problem description

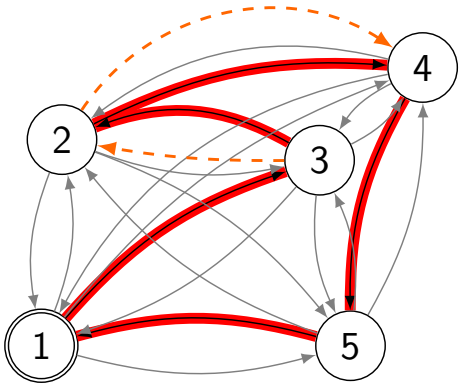
- Given: set of cities + asymmetric distances + precedence set




TSP ATSP SOP

## Formal problem description

- Given: set of cities + asymmetric distances + precedence set
- Required: shortest tour that fulfills the precedences




TSP → ATSP → SOP

## Coincise problem description

**Sequential Ordering Problem (SOP)**

Find a minimum cost Hamiltonian path on a directed graph, subject to precedence constraints among the nodes.

# Outline

- Introduction
- Problem description
- A Genetic Algorithm
- A Hybrid Ant System
- A Heuristic Manipulation Technique
- A Particle Swarm Optimization
- An Enhanced Ant Colony System
- A Shared Incumbent Environment
- Bibliography

## A Genetic Algorithm

S. Chen, S. Smith
Commonality and genetic algorithms.
Technical Report CMU-RI-TR-96-27, The Robotic Institute, Carnegie Mellon University, 1996

## Sequential Ordering Problem

- Escudero (1988)
- *General ATSP Problem*
  - Precedence Constrained ATSP Polytope (Balas, Fischetti, Pulleyblank, 1995).
  - Branch and Cut (Ascheuer, 1996)
  - Maximum Partial Order/Arbitrary Insertion GA (Chen and Smith, 1996)
- *Pick-Up and Delivery*
  - Lexicographic search with labeling Procedure (Savelsbergh, 1990).

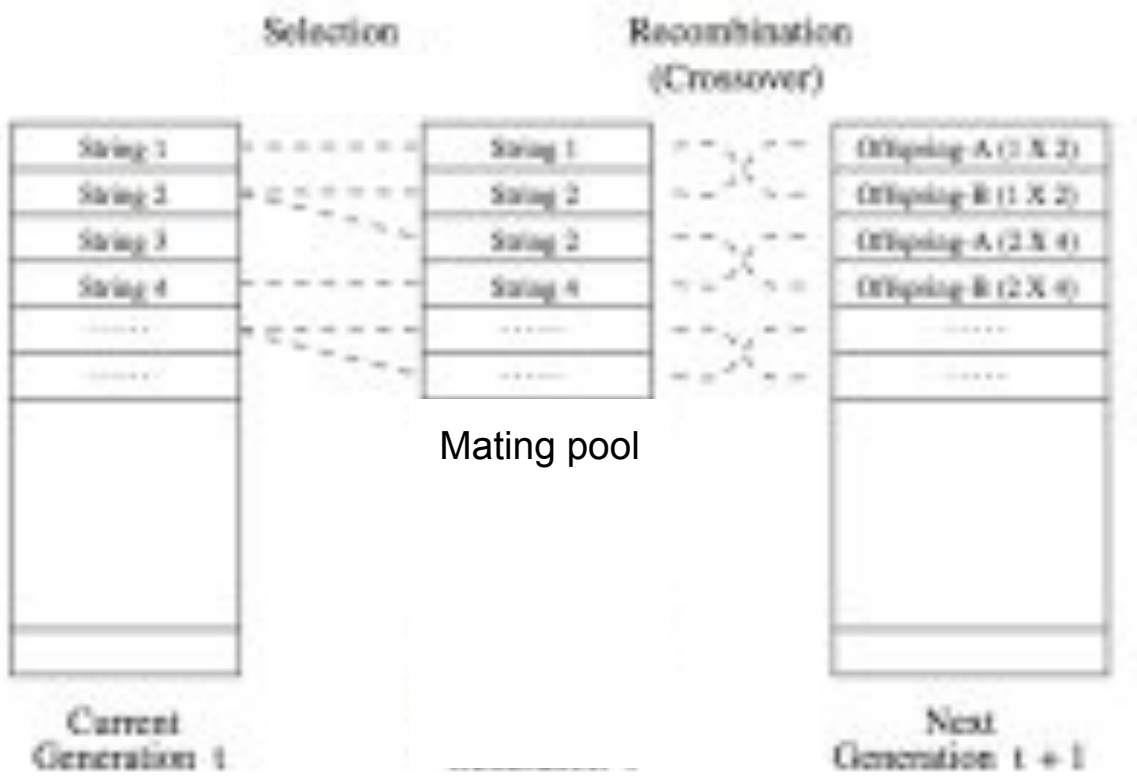
# Genetic Algorithms (GAs)

- Inspired by Darwinian principle of *natural selection*
- Different from other EC methods primarily due to emphasis in sexual reproduction (crossover)
- GAs search the problem space by trying to correctly combine genetic *building blocks* from different individuals in the population

# GA: Basic Procedure

1. **Initialize** random population of candidate solutions
2. **Evaluate** solutions on problem and assign a fitness score
3. **Select** some solutions for mating
4. **Recombine**: create new solutions from selected ones by exchanging structure
5. **IF** good solution not found: **Goto 2**

The cycle from **2** to **5** is known as a *generation*

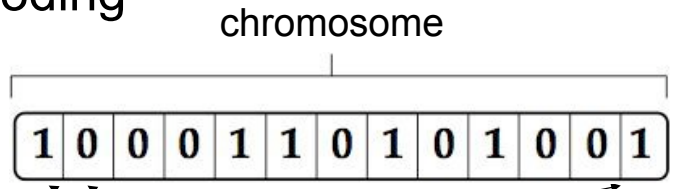


# GA Terminology

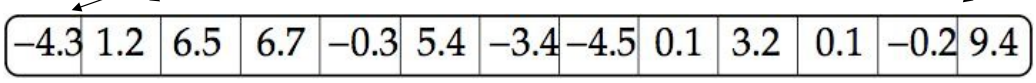
- Solutions are encoded in strings called ***chromosomes***
- Each chromosome consists of some number of ***genes***
- Each gene can take an a value or ***allele*** from some specified alphabet, e.g.
  - Binary {0,1}
  - Real numbers (infinite alphabet)

# Genotype Encoding

- Binary encoding




- Real values



# Selection: fitness proportional

1. Calculate a genotype's probability of being selected in proportion to its fitness

$$p_i = \frac{f_i}{\sum f_j}$$

2. Then select some number of genotypes for mating according to probabilities $p_i$

Genotypes that are more fit are more likely to be selected

# Selection: linear ranking

1. Sort the genotypes by fitness
2. Compute probability of being selected by

$$p_i = 2 - SP + 2 * (SP - 1) * (\text{rank}(i) - 1) / (N - 1)$$

where SP is the ***selective pressure*** **[**1.0,2.0],
and rank denotes the genotype's position in the sorted population, rank(1) is most fit, rank(N) is least fit

3. then select some number of genotypes for mating according to probabilities

# Selection: Tournament

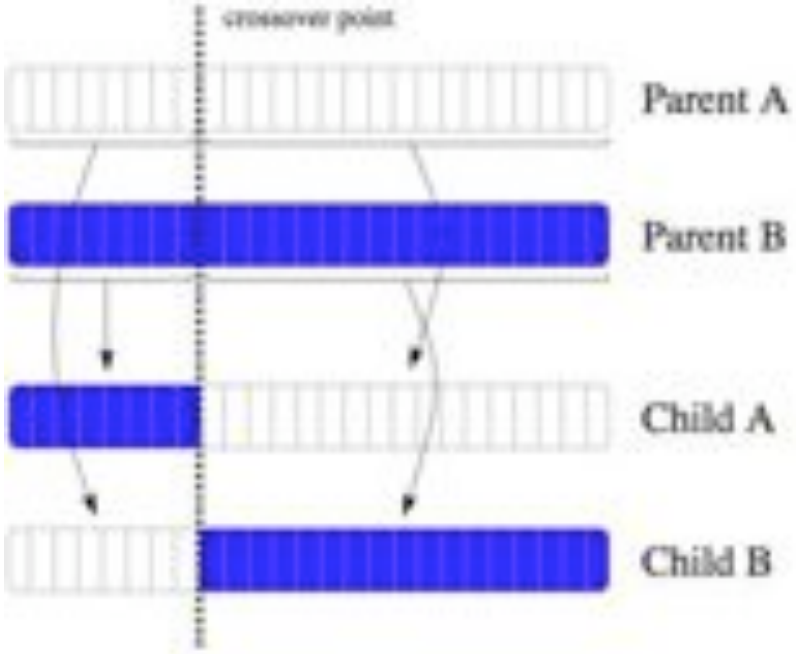
1. Let T be the tournament size (between 2 and the size of the population, N)
2. Select T genotypes at random from the population and take the most fit as the tournament "winner"
3. Put the winner in the mating pool
4. Goto 1 until we have enough genotypes in mating pool

Larger values of T increase selective pressure

# Reproduction

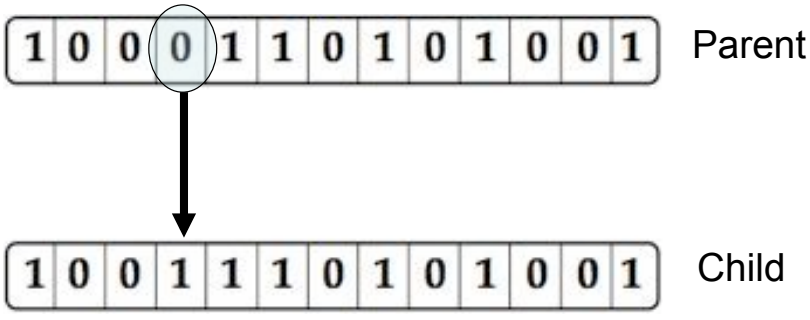
- Generate new individuals (search points) by mixing or altering the genotypes of the *selected* members of the population
- **Crossover**: select alleles from two *parent* chromosome to form two *children* (*recombination* type genetic operator)
- **Mutation**: random perturb some of the alleles of a parent

# Genetic Operators: 1-point crossover




Select random crossover points and exchange substrings

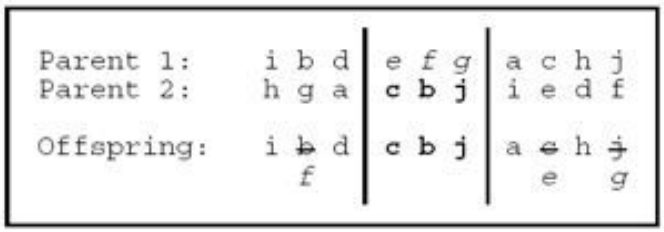
# Mutation: binary




Randomly flip a bit with probability *mutation rate*

# Sequence-based crossover operators

Partially Mapped Crossover (PMX) [Goldberg and R. Lingle., 1985] has the form of two-point crossover.

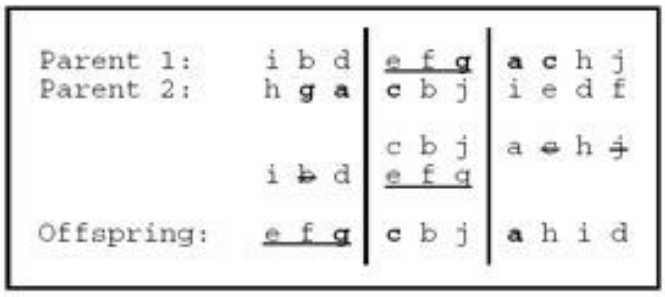



2: Example of PMX. The mappings are c-e, b-f, and j-g.

The offspring takes the cities from Parent 2 between the cut-points, and it takes the cities in the first and last sections from Parent 1. However, if a city in these outer sections has already been taken from Parent 2, its "mapped" city is taken instead. The mappings are defined between the cut-points--the city of Parent 2 is mapped to the corresponding city of Parent 1.

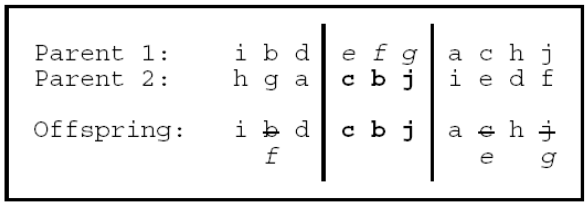
# Sequence-based crossover operators

Order Crossover (OX) [Davis 85] has the form of two-point crossover.

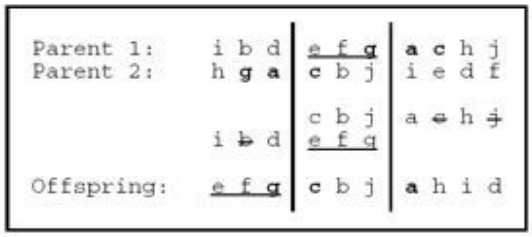



The offspring starts by taking the cities of Parent 2 between the cut-points. Then, starting from the second cut-point, the offspring takes the cities of Parent 1 ("wrapping around" from the last segment to the first segment). When a city that has been taken from Parent 2 is encountered, it is skipped--the remaining cities are appended in the order they have in Parent 1.

# Sequence-based crossover operators



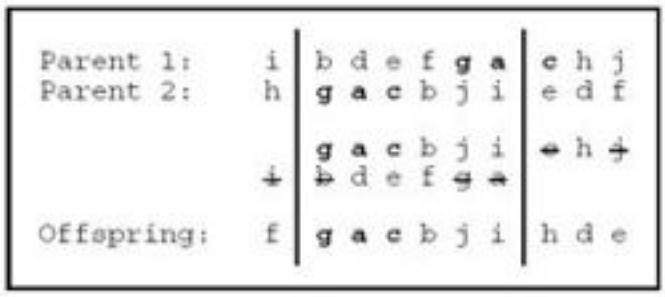

2: Example of PMX. The mappings are c-e, b-f, and j-g.




Comparison between PMX and OX. OX can be less disruptive to sub-tours. For example, the sub-tour e-f-g in Parent 1 is now transmitted to the offspring. However, the common sub-tour g-a-c is (still) disrupted.

# Sequence-based crossover operators

Maximal Sub-Tour (MST)--the longest (undirected) sub-tour that is common to both parents. Thus, OX is modified to preserve the MST.

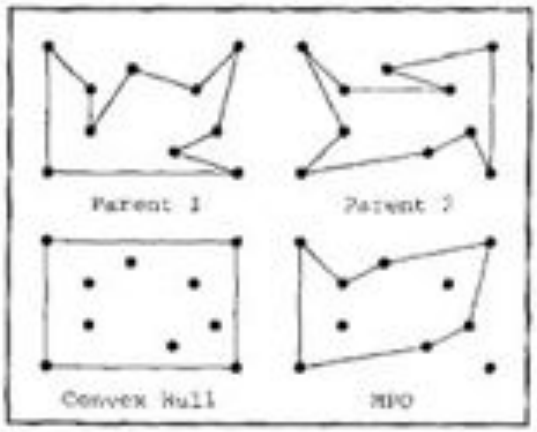



7: Example of MST-OX. The Maximal Sub-Tour g-a-c is now preserved.

Scanning both parents to identify the Maximal Sub-Tour, the first cut-point occurs to the immediate left of the MST in Parent 2. The second cut-point is then made a random distance to the right of the MST1. After OX is applied

## Sequence-based crossover operators

The longest common partial order is the Maximum Partial Order (MPO). Using Arbitrary Insertion to complete this partial solution, the overall process defines the Maximum Partial Order/Arbitrary Insertion heuristic operator.




Maximum Partial Order (MPO).

## Sequence-based crossover operators

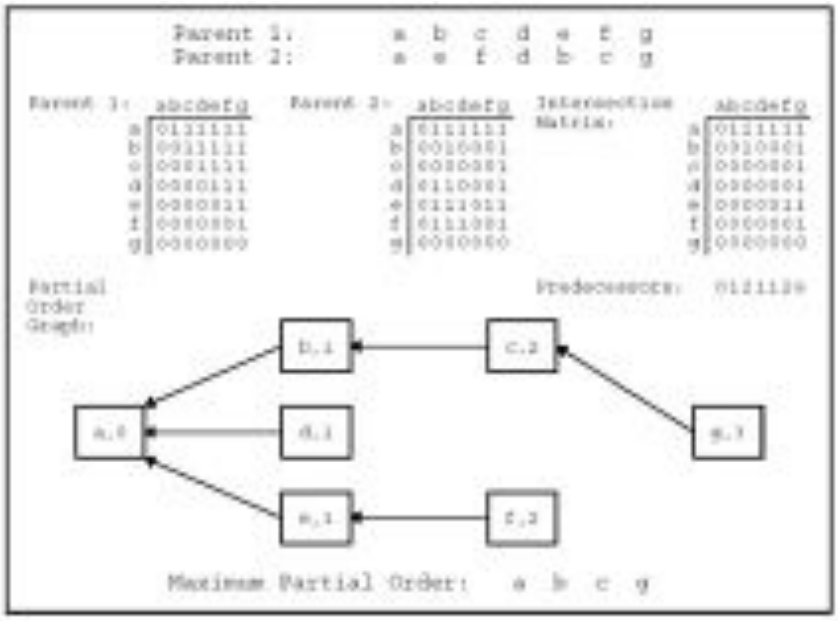
Maximum Partial Order (MPO).

- Convert parent sequences to Boolean matrices
- Intersect matrices
- Sum columns to get each city's predecessors
- Build partial order graph
  - Find city with fewest number of predecessors
  - Attach city to the predecessor with the most *ordered* predecessors
- Find longest path in graph

3: Pseudo-code to find the Maximum Partial Order of two parent sequences.

# Sequence-based crossover operators

Maximum Partial Order (MPO).




City g is preceded by all of the cities in the graph, but it can only be attached to cities c and f because those cities have the most ordered predecessors (2).

# MPO/AI performance

Table 6.3 Results for MPO/AI in GENIE. Population size is 400, and the initial population is 400 AI solutions started from the convex hull. Values are percent surplus from known optimum for average of 5 runs (50 generations each).

| TSPLIB Instance | Avg. Best Convex Hull/AI Start Tour | Avg. Best MPO/AI Tour |
|---|---|---|
| d198 | + 3.05 % | + 1.24 % |
| lin318 | + 6.04 % | + 1.75 % |
| fl417 | + 1.91 % | + 0.58 % |
| pcb442 | + 8.97 % | + 3.48 % |
| u574 | + 8.45 % | + 2.59 % |
| average | + 5.68 % | + 1.93 % |

Is the Common Good? A New Perspective Developed in Genetic Algorithms, PHD Thesis, Stephen Chen, 1999

## MPO/AI for SOP

| SOP Instance | Size | Constraints | Average Best AI Start | Average Best MPOAI | Overall Best BPOAI | Original Branch and Cut Bounds [Ascent] | Time to Bound (s) | Runtime (s) |
|---|---|---|---|---|---|---|---|---|
| R70.1 | 71 | 17 | 41808 | 39615 | 39545 | **39313** | NA | 76 |
| R70.2 | 71 | 35 | 43466 | 40435 | **40422** | [38726, 41770] | 5.4 | 73 |
| R70.3 | 71 | 86 | 46731 | 42568 | ***42535** | [41309, 44732] | 2.6 | 48 |
| R70.4 | 71 | 86 | 59692 | 53583 | **53582** | [52269, 53582] | 2.8 | 47 |
| krs124p.1 | 131 | 25 | 45758 | 40080 | **40186** | [37722, 42945] | 5.4 | 136 |
| krs124p.2 | 131 | 49 | 43698 | 42576 | **41677** | [38834, 41664] | 8.2 | 94 |
| krs124p.3 | 131 | 97 | 92768 | 91083 | **90876** | [89967, 90684] | 3.8 | 103 |
| krs124p.4 | 131 | 131 | 87975 | 76183 | ***76183** | [64658, 80753] | 2.0 | 76 |
| rbg323a | 325 | 2412 | 3408 | 3161 | **3157** | [3136, 3221] | 207.6 | 1868 |
| rbg341a | 343 | 2582 | 3168 | 2683 | **2597** | [2543, 2894] | 70.0 | 2308 |
| rbg358a | 360 | 3239 | 3165 | 2636 | **2589** | [2516, 2756] | 533.0 | 4491 |
| rbg378a | 380 | 3089 | 3428 | 2943 | **2833** | [2781, 3142] | 67.6 | 6364 |
| ry48p.1 | 49 | 11 | 16802 | 16813 | ***16805** | [15226, 16805] | 2.6 | 22 |
| ry48p.2 | 49 | 23 | 18071 | 16670 | ***16666** | [15524, 17071] | 1.6 | 32 |
| ry48p.3 | 49 | 42 | 22074 | 19085 | ***18854** | [18156, 20081] | 7.6 | 28 |
| ry48p.4 | 49 | 88 | 32891 | 31640 | 31640 | [29667, 31640] | 5.0 | 18 |

# Outline

- Introduction
- Problem description
- A Genetic Algorithm
- A Hybrid Ant System
- A Heuristic Manipulation Technique
- A Particle Swarm Optimization
- An Enhanced Ant Colony System
- A Shared Incumbent Environment
- Bibliography

# A Hybrid Ant System

L.M. Gambardella, M. Dorigo.
HAS-SOP: an Hybrid Ant System for the sequential ordering problem.
Technical Report IDSIA-11-97, IDSIA, Lugano, Switzerland, 1997

L.M. Gambardella, M. Dorigo
An Ant Colony System Hybridized with a New Local Search for the Sequential Ordering Problem.
INFORMS Journal on Computing, 12(3), 237-255, 2000

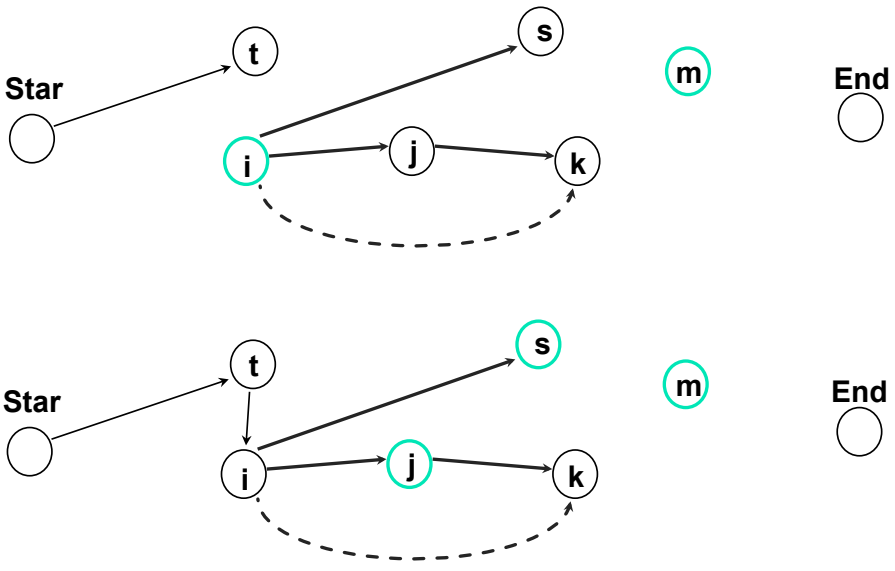
## HAS-SOP: Hybrid Ant System for SOP

- Dorigo, Gambardella 2000
- Constructive phase based on ACS
- Trail updating as ACS
- New local search SOP-3_exchange strategy based on a combination between lexicographic search and a new labeling procedure.
- New data structure to drive the search
- First in literature that uses a local search edge-exchange strategy to directly handle multiple constraints without any increase in computational time.

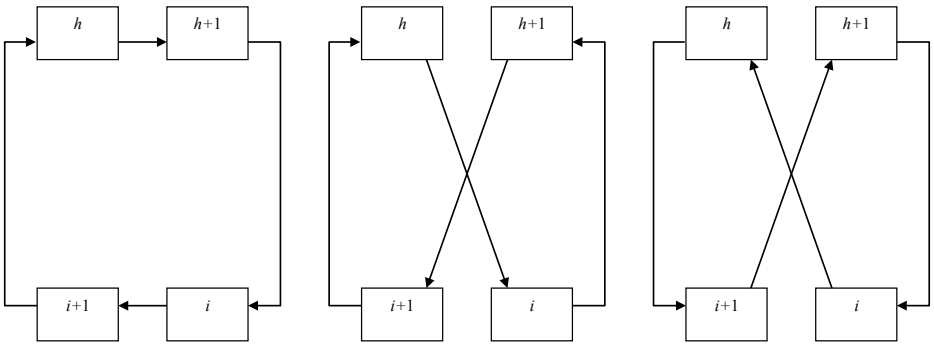
# Ants for SOP

- Each ant iteratively starts from node 0 and adds new nodes until all nodes have been visited and node *n* is reached.
- When in node *i*, an ant chooses probabilistically the next node *j* from the set *F*(*i*) of feasible nodes.
- *F*(*i*) contains all the nodes *j* still to be visited and such that all nodes that have to precede *j*, according to precedence constraints, have already been inserted in the sequence

# Feasable Ant Sets

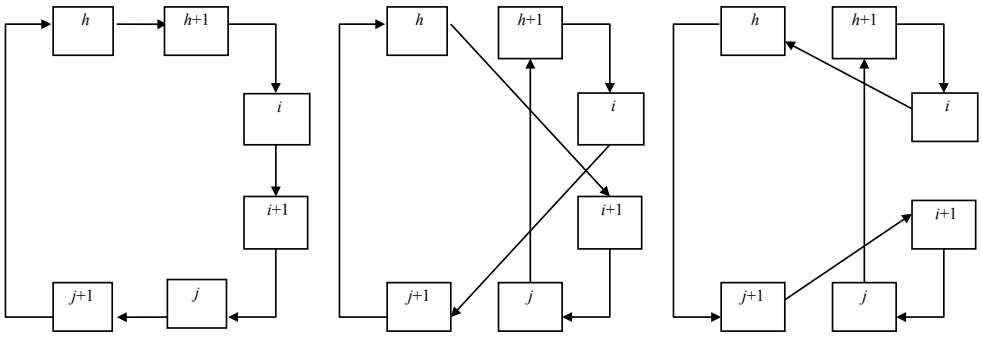


# Local Search




A 2-exchange always inverts a path.

# Local Search

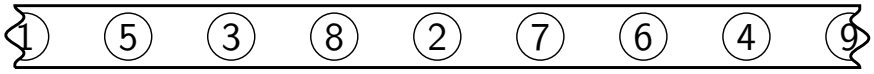



**a** **b** **c**

A 3-exchange without (b) and with (c) path inversion

## SOP-3-exchange local search

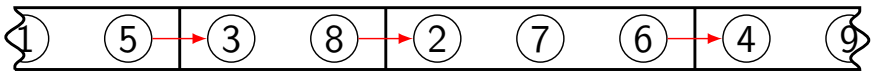
- HAS-SOP has two components: construction heuristic + local search




Gambardella, L. M. and Dorigo, M. [2000]. *An Ant Colony System hybridized with a new local search for the sequential ordering problem*, INFORMS Journal on Computing 12(3): 237–255.

## SOP-3-exchange local search

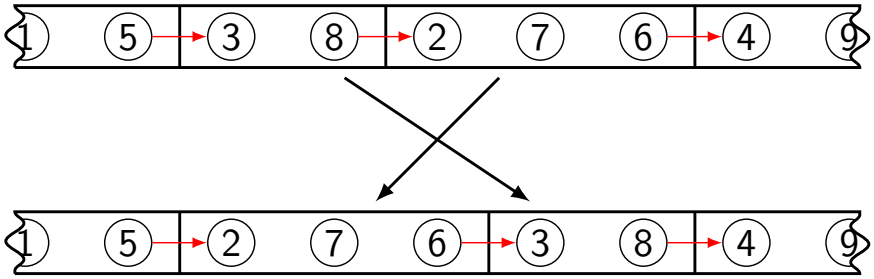
- HAS-SOP has two components: construction heuristic + local search
- 3-opt: three edges are replaced without path inversion




Gambardella, L. M. and Dorigo, M. [2000]. *An Ant Colony System hybridized with a new local search for the sequential ordering problem*, INFORMS Journal on Computing 12(3): 237–255.

## SOP-3-exchange local search

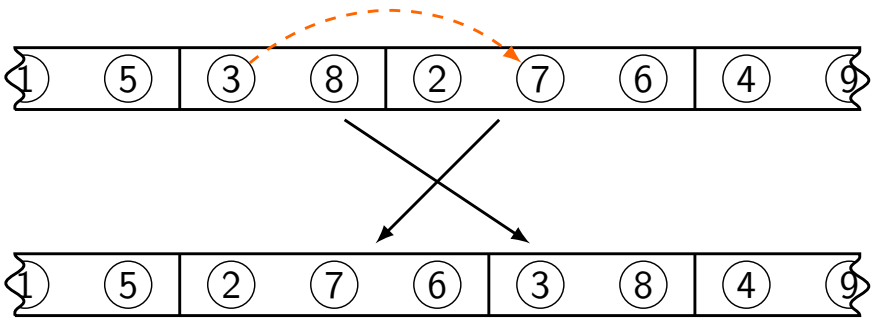
- HAS-SOP has two components: construction heuristic + local search
- 3-opt: three edges are replaced without path inversion




Gambardella, L. M. and Dorigo, M. [2000]. *An Ant Colony System hybridized with a new local search for the sequential ordering problem*, INFORMS Journal on Computing 12(3): 237–255.

## SOP-3-exchange local search

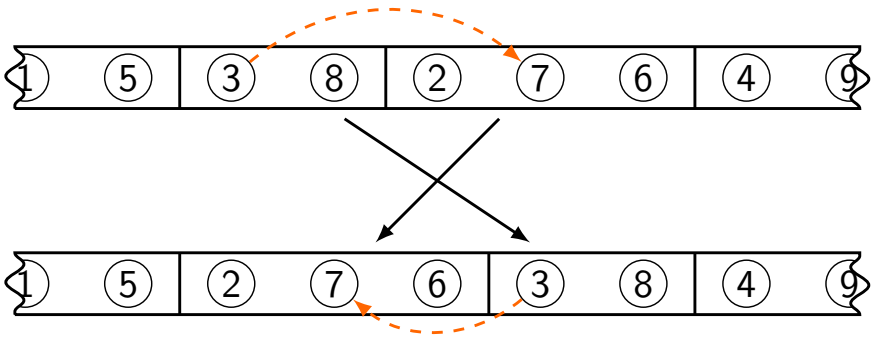
- HAS-SOP has two components: construction heuristic + local search
- 3-opt: three edges are replaced without path inversion
- Feasible move: no precedences between left and right nodes




Gambardella, L. M. and Dorigo, M. [2000]. *An Ant Colony System hybridized with a new local search for the sequential ordering problem*, INFORMS Journal on Computing 12(3): 237–255.

## SOP-3-exchange local search

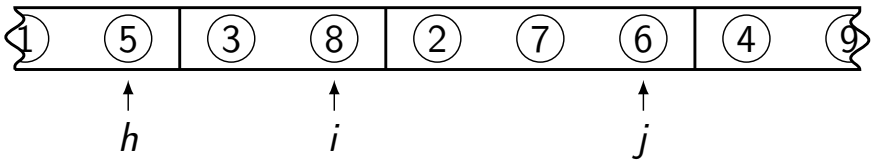
- HAS-SOP has two components: construction heuristic + local search
- 3-opt: three edges are replaced without path inversion
- Feasible move: no precedences between left and right nodes




Gambardella, L. M. and Dorigo, M. [2000]. *An Ant Colony System hybridized with a new local search for the sequential ordering problem*, INFORMS Journal on Computing 12(3): 237–255.

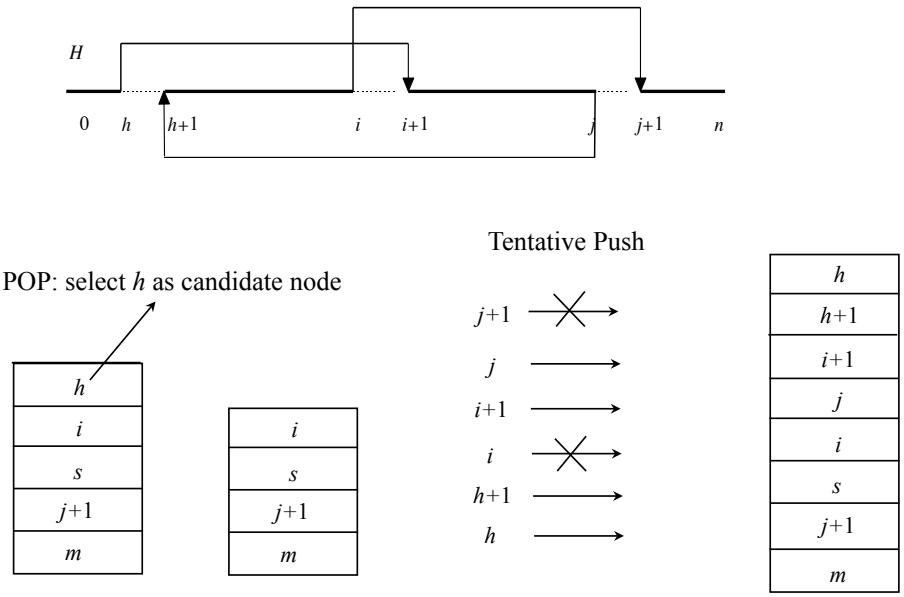
## SOP-3-exchange local search

- HAS-SOP has two components: construction heuristic + local search
- 3-opt: three edges are replaced without path inversion
- Feasible move: no precedences between left and right nodes
- No increase in complexity $\mathcal{O}(n^3)$: lexicographic order of $(h, i, j)$




Gambardella, L. M. and Dorigo, M. [2000]. *An Ant Colony System hybridized with a new local search for the sequential ordering problem*, INFORMS Journal on Computing 12(3): 237–255.

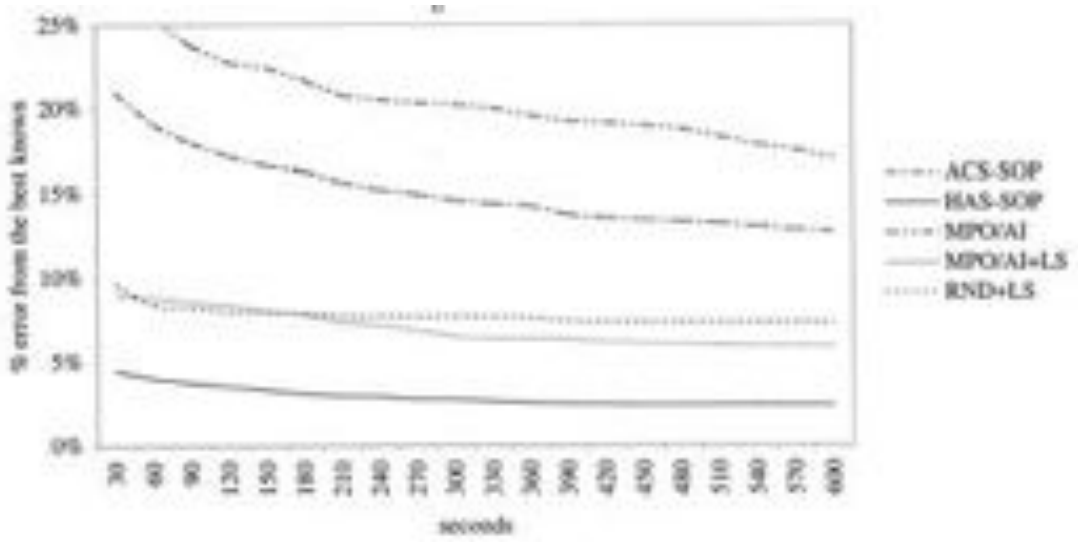
# Selection of node h with the Don't Push Stack




# Local search contribution

| | RND | MPO/AI | ACS-SOP | RND+LS | MPO/AI+LS | HAS-SOP |
|---|---|---|---|---|---|---|
| **prob.100** | 1440.17% | 134.66% | 40.62% | 50.07% | 47.58% | **17.46%** |
| **rbg109a** | 64.57% | 0.33% | 1.93% | 0.08% | 0.06% | **0.00%** |
| **rbg150a** | 37.85% | 0.19% | 2.54% | 0.08% | 0.13% | **0.00%** |
| **rbg174a** | 40.86% | 0.01% | 2.16% | 0.15% | **0.00%** | 0.08% |
| **rbg253a** | 45.85% | 0.03% | 2.68% | 0.21% | **0.00%** | **0.00%** |
| **rbg323a** | 80.14% | 1.08% | 9.60% | 1.27% | **0.08%** | 0.21% |
| **rbg341a** | 125.46% | 3.02% | 12.64% | 4.41% | **0.96%** | 1.54% |
| **rbg358a** | 151.92% | 7.83% | 20.20% | 4.98% | 2.51% | **1.37%** |
| **rbg378a** | 131.58% | 5.95% | 22.02% | 4.17% | 1.40% | **0.88%** |
| **avg** | 235.38% | 17.01% | 12.71% | 7.27% | 5.86% | **2.39%** |

# Sequential Ordering Problems




# HAS-SOP

| PROB | TSPLIB Bounds | MPO/AI Best | MPO/AI Avg | MPO/AI Time (sec) | HAS-SOP Best | HAS-SOP Avg | HAS-SOP Time (sec) |
|---|---|---|---|---|---|---|---|
| ft70.1.sop | 39313 | 39545 | 39615 | 120 | 39313 | 39313.0 | 29.8 |
| ft70.2.sop | [39739,40422] | 40422 | 40435 | 120 | 40419 | 40433.5 | 114.1 |
| ft70.3.sop | [41305,42535] | 42535 | 42558 | 120 | 42535 | 42535.0 | 64.4 |
| ft70.4.sop | [52269,53562] | 53562 | 53583 | 120 | 53530 | 53566.5 | 38.2 |
| kro124p.1.sop | [37722,40186] | 40186 | 40996 | 240 | 39420 | 39420.0 | 115.2 |
| kro124p.2.sop | [38534,41677] | 41667 | 42576 | 240 | 41336 | 41336.0 | 119.3 |
| kro124p.3.sop | [40967,50876] | 50876 | 51085 | 240 | 49499 | 49648.8 | 262.8 |
| kro124p.4.sop | [64858,76103] | 76103 | 76103 | 240 | 76103 | 76103.0 | 57.4 |
| rbg323a.sop | [3136,3157] | 3157 | 3161 | 2760 | 3141 | 3146.0 | 1685.5 |
| rbg341a.sop | [2543,2597] | 2597 | 2603 | 3840 | 2580 | 2591.9 | 2149.6 |
| rbg358a.sop | [2518,2599] | 2599 | 2636 | 6120 | 2555 | 2561.2 | 2169.3 |
| rbg378a.sop | [2761,2833] | 2833 | 2843 | 8820 | 2817 | 2834.3 | 2640.3 |

We tested and compare our algorithms on a set of problems in TSPLIB using a SUN Ultra SPARC 1 (167Mhz)

| PROB | TSPLIB Bounds | NEW Lower Bounds | NEW Upper Bounds | HAS-SOP All Best | Avg Result | Std.Dev. | Avg Time (sec) |
|---|---|---|---|---|---|---|---|
| ESC63.sop | 62 | | | 62 | 62.0 | 0 | 0.1 |
| ESC78.sop | 18230 | | | 18230 | 18230.0 | 0 | 6.9 |
| ft53.1.sop | [7438,7570] | | 7531 | 7531 | 7531.0 | 0 | 9.9 |
| ft53.2.sop | [7630,8335] | | 8026 | 8026 | 8026.0 | 0 | 18.4 |
| ft53.3.sop | [9473,10935] | | 10262 | 10262 | 10262.0 | 0 | 2.9 |
| ft53.4.sop | 14425 | | | 14425 | 14425.0 | 0 | 0.4 |
| ft70.1.sop | 39313 | | | 39313 | 39313.0 | 0 | 29.8 |
| ft70.2.sop | [39739,40422] | 39803 | 40419 | 40419 | 40433.5 | 24.6 | 114.1 |
| ft70.3.sop | [41305,42535] | 41305 | | 42535 | 42535.0 | 0 | 64.4 |
| ft70.4.sop | [52269,53562] | 53072 | 53530 | 53530 | 53566.5 | 7.6 | 38.2 |
| kro124p.1.sop | [37722,40186] | 37761 | 39420 | 39420 | 39420.0 | 0 | 115.2 |
| kro124p.2.sop | [38534,41677] | 38719 | 41336 | 41336 | 41336.0 | 0 | 119.3 |
| kro124p.3.sop | [40967,50876] | 41578 | 49499 | 49499 | 49648.8 | 249.7 | 262.8 |
| kro124p.4.sop | [64858,76103] | | | 76103 | 76103.0 | 0 | 57.4 |
| prob.100.sop | [1024,1385] | 1027 | 1190 | 1190 | 1302.4 | 39.4 | 1918.7 |
| rbg109a.sop | 1038 | | | 1038 | 1038.0 | 0 | 14.6 |
| rbg150a.sop | [1748,1750] | | | 1750 | 1750.0 | 0 | 159.1 |
| rbg174a.sop | 2033 | | | 2033 | 2034.7 | 1.4 | 99.3 |
| rbg253a.sop | [2928,2987] | 2940 | 2950 | 2950 | 2950.0 | 0 | 81.5 |
| rbg323a.sop | [3136,3157] | 3137 | 3141 | 3141 | 3146.0 | 1.4 | 1685.5 |
| rbg341a.sop | [2543,2597] | 2543 | 2574 | 2574 | 2591.9 | 11.8 | 2149.6 |
| rbg358a.sop | [2518,2599] | 2529 | 2545 | 2545 | 2561.2 | 5.2 | 2169.3 |
| rbg378a.sop | [2761,2833] | 2817 | 2817 | 2817 | 2834.3 | 10.7 | 2640.3 |

Norbert Ascheuer (1997) has run his branch&cut SOP program starting from our best solutions. He could not improve them within 24-CPU hours on a SUN SPARC Station 4 (110Mhz) but he proves optimality for rbg378a and computes the new reported lower bounds.

# Outline

# A Heuristic Manipulation Technique

R. Montemanni, D.H. Smith and L.M. Gambardella.
Ant colony systems for large sequential ordering problems.
Proceedings of the IEEE Swarm Intelligence Symposium (SIS 2007), Honolulu, U.S.A., 1-5 April 2007

R. Montemanni, A.E. Rizzoli, D.H. Smith and L.M. Gambardella.
Sequential ordering problems for crane scheduling in port terminals.
*Proceedings of HMS 2008*– The 11th Intermodal Workshop on Harbor, Maritime and Multimodal Logistic Modeling and Simulation, Bruzzone et al. eds., pages 180-189, Campora San Giovanni, Italy, 17-19 September 2008

R. Montemanni , D.H. Smith and L.M. Gambardella.
A heuristic manipulation technique for the sequential ordering problem.
Computers and Operations Research, 35(12), 3931-3944, December 2008

R. Montemanni, D.H. Smith, A.E. Rizzoli and L.M. Gambardella.
Sequential Ordering Problems for Crane Scheduling in Port Terminals.
International Journal of Simulation and Process Modelling, 5(4), 348-361, 2009

# SOP - Literature: Heuristic methods

•Chen, Smith. *Commonality and genetic algorithms*. Technical report CMU-RI-TR-96-27, Carnagie Mellon University, 1996.

•Gambardella and Dorigo. *An Ant Colony System hybridized with a new local search for the sequential ordering problem*. INFORMS Journal on Computing 12(3): 237-255, 2000.

•Seo, Moon. *A hybrid genetic algorithm based on complete graph representation for the sequential ordering problem*. Proc. of GECCO 2003 669-680, 2003.

•Guerriero, Mancini. *A cooperative parallel rollout algorithm for the sequential ordering problem*. Parallel Computing 29(5): 663-677, 2003.

# HAS-SOP: Hybrid Ant System for SOP

- **Ant Colony System** (ACS), ie the behaviour of real ants is mimic to retrieve good solutions to the optimization problem
- **Best known** heuristic algorithm
- More recent heuristics **did not** manage **to outperform** it, even if tests are run on better hardware
- Why does HAS-SOP perform well?
  - ACS is used to identify promising solution
  - A very effective and efficient local search is used to take promising solutions to a local optimum
  - Very efficient implementation

**HAS-SOP: ACS based algorithm (best-known in literature)**

**Gambardella L.M, Dorigo M., An Ant Colony System Hybridized with a New Local Search for the Sequential Ordering Problem, *INFORMS Journal on Computing*, vol.12(3), pp. 237-255, 2000**

# HAS-SOP$_{APC}$: The idea

- Recent heuristics **did not manage** to outperform HAS-SOP
- **Instead of** a complete new method...

  **IDEA**: Heuristic manipulation technique working **on top** of HAS-SOP
- **Artificial Precedence Constraints (APCs)** are iteratively **added and removed** and the resulting modified problem is solved by HAS-SOP

  \+ The **search space** is like to be **reduced** => easier problem

  — **Optimal solutions** might be **hidden** by APCs => dynamic set of active APCs; ad-hoc strategies to add/ remove APCs

# HAS-SOP$_{APC}$: The algorithm (1/2)

- Variables $m_{ij}$: they contain an indication on "**how good**" was in the past to have node **i visided before** node **j**
- Each time HAS-SOP produces a **solution** $S_k$ with **cost** $L_k$, **matrix m** is **updated** as follows:

  $m_{ij} \mathrel{+}= L_1/L_k \ \forall i, j \in V, p_k(i) < p_k(j) < p_k(i) + 5, (i, j) \text{ not} \in R$

  $m_{ji} \mathrel{-}= L_1/L_k \ \forall i, j \in V, p_k(i) < p_k(j) < p_k(i) + 5, (i, j) \text{ not} \in R$

  – $L_1$ is the **cost** of the **first heuristic solution** generated by HAS-SOP

  – $p_k(i)$ is the **position** of node **i** in solution $S_k$

  – **R** is the set of the **active** precedence constraints

# HAS-SOP$_{APC}$: The algorithm (2/2)

- **After** the first **100 solutions** have been produced by HAS-SOP, **20 APCs** are **added**:
  – the non active constraints with the **highest values** of **m** are **added**
- **Every 50** new **solutions** are produced by HAS-SOP, **5 APCs** are **substituted**:
  – the active constraints with the **lower values** of **m** are **drop**
  – the non active constraints with the **highest values** of **m** are **added**

# Computational results: benchmarks

- **TSPLIB** problems are rather easy for modern heuristics (and solved *almost* to proven optimality)
- No significant difference between HAS-SOP and HAS-SOP$_{APC}$ on TSPLIB problems
- New **(larger) random problems** were generated (they are publicly available)
- Problem **n-r-p** has the following characteristics:
  - Number of nodes = *n*
  - Costs such that $0 \leq c_{ij} \leq r$ for all arcs *(i,j)*
  - Approximately *p% of the arcs brings a* precedence constraint
- Values considered for parameters:
  - $n \in \{200, 300, 400, 500, 600, 700\}$
  - $r \in \{100, 1000\}$
  - $p \in \{1, 15, 30, 60\}$

# Computational results: experimental settings

- **Comparison** between HAS-SOP and HAS-SOP$_{APC}$
- **10 runs** are considered for each possible combination problem/method
  - **Average** results are analyzed
  - **Best** results are analyzed
  - **Worst** results are analyzed
- **Computer used:** AMD Opteron 250 2.4GHz / 4GB
- **600 seconds** available for each run
  - Enough to reach a steady state

# Computational results: Comments

- HAS-SOP$_{APC}$ is **never worse** than HAS-SOP, both in terms of **average** and **best** results
- The **improvements** guaranteed by APC **decreases** as the number of **precedence constraints** in the original problem **increases** (the search space is already narrow)
- Larger problems **=>** Larger improvements
- **Average improvement** of HAS-SOP$_{APC}$ over HAS-SOP is:
  - **Average** results over 10 runs: **1.30%**
  - **Best** results over 10 runs: **2.11%**
  - **Worst** results over 10 runs: **1.41%**
- Response of **Statistical tests** on the difference in the results of HAS-SOP$_{APC}$ and HAS-SOP: **extremely significant**

# Conclusions and future work

- A **Heuristic manipulation technique**, based on the creation of **Artificial Precedence Constraints** (APC) has been proposed for the SOP
- The **APC** technique has been implemented **on top of HAS-SOP** (Ant System)
- Computational **results** indicate that the technique induces **improvements** on large and difficult problems
- The novel technique can be used on top of **different algorithms** and for **different problems** (i.e. not only ACO, not only SOP)

# Outline

- Introduction
- Problem description
- A Genetic Algorithm
- A Hybrid Ant System
- A Heuristic Manipulation Technique
- A Particle Swarm Optimization
- An Enhanced Ant Colony System
- A Shared Incumbent Environment
- Bibliography



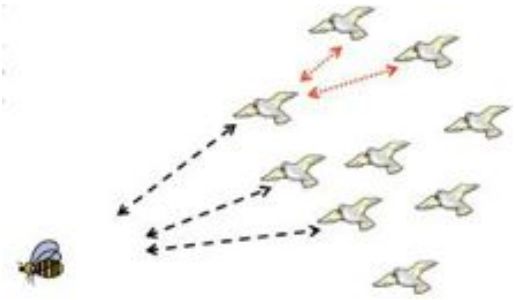
# A Particle Swarm Optimization Approach

D. Anghinolfi, R. Montemanni, M. Paolucci and L.M. Gambardella.
A Particle Swarm Optimization approach for the Sequential Ordering Problem
Proceedings of the VIII Metaheuristic International Conference (MIC 2009), Hamburg, Germany, 13-16 July 2009

D. Anghinolfi, R. Montemanni, M. Paolucci and L.M. Gambardella.
A Particle Swarm Optimization approach for the Sequential Ordering Problem
Computers and Operations Research, 38(7), 2076-1085, 2011

## PSO: natural/social background (1)

- Early work on simulation of bird flocking aimed at understanding the underlying rules of bird flocking [Reynolds, 1984] and roosting behavior [Heppner & Grenader, 1990]



- The notion of change in human social behavior/psychology is seen as the analogous of change in spatial position in birds
- Rules assumed to be simple and based on social behavior: sharing of information and reciprocal respect of the occupancy of physical space
- Social sharing of information among conspeciates seems to offer an evolutionary advantage



## PSO: natural/social background (2)

- Initial simulation work [Eberhart & Kennedy, 1995]
- A population of $N >> 1$ agents is initialized on a toroidal 2D pixel grid with random position and velocity, $(\bar{x}_i, \bar{v}_i),\ i = 1, \ldots, N$
- At each iteration loop, each agent determines its new speed vector according to that of its nearest neighbors
- A random component is used in order to avoid fully unanimous, unchanging, flocking
- Roosting behavior: looks like a dynamic force such that attracts the swarm to land on a specific location. The roost could be the equivalent of the optimum in a search space!

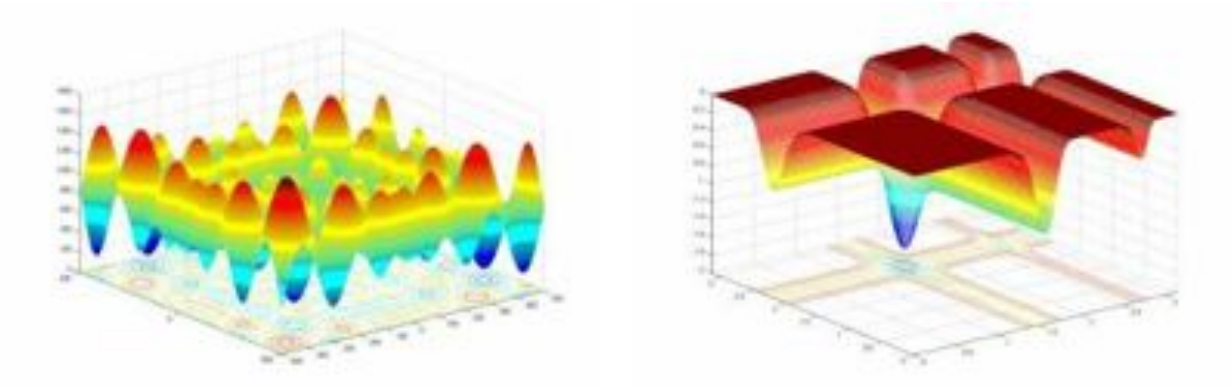
## PSO: natural/social background (3)

- Birds explore the environment in search for food
- Agents = solution hunters that socially share knowledge while they move across a solution space
- An agent that has found a “good” point leads its neighbors there
- . . . and eventually all the agents “flock” toward the best point in the solution space
- Compared to CAs:
  - the neighborhood is not ’physical’ anymore
  - the agents are individual points in the solution space
  - they are mobile, that is, they are not constrained to a certain location
  - the whole system is less interdependent
  - it targets (multi-agent) optimization tasks



## PSO: the particles and the task

- Mainly Optimization of continuous functions $f(\vec{x}) : \mathbb{R}^n \rightarrow \mathbb{R}$
  - A growing number of applications to combinatorial optimization
- For convex functions gradient methods can be effective used, but for non-convex ones . . .
- An agent is an $n$-dimensional particle moving over function’s domain
- A particle $p$ has an internal state consisting of: $\{\vec{x}, \vec{v}, \vec{x}_{pbest}, \mathcal{N}(p)\}$ and makes use of a simple rule to update its velocity and position



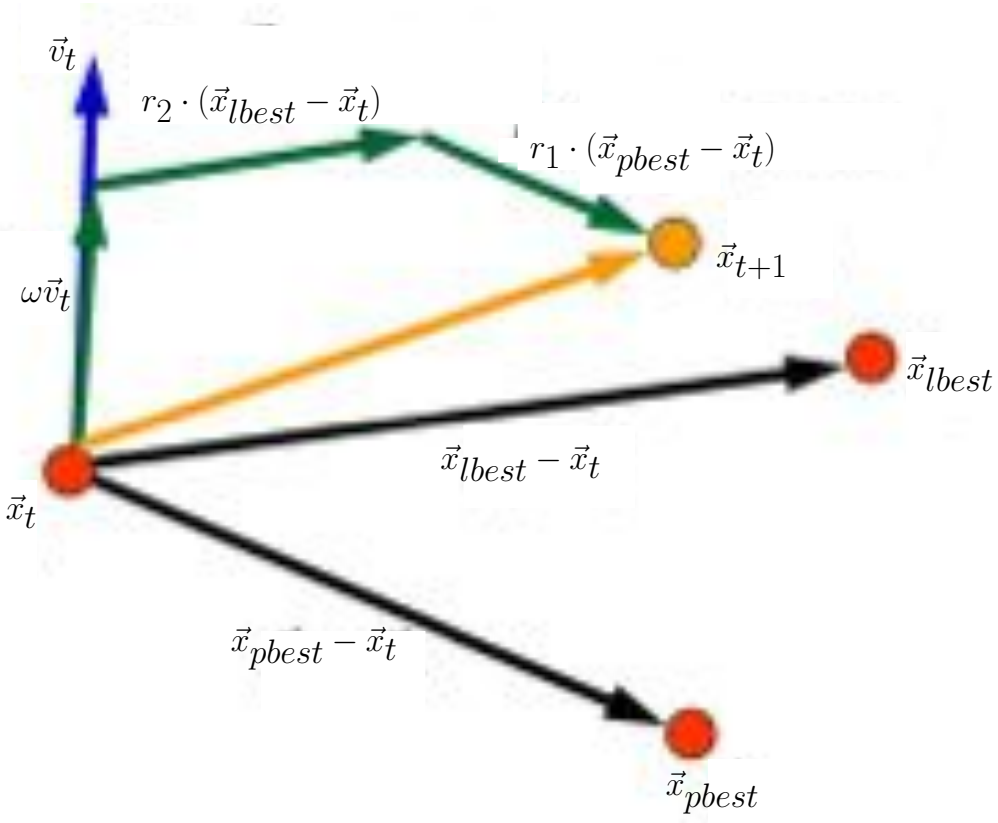

- A. Banks, J. Vincent, C. Anyakoha, *A review of particle swarm optimization. Part I: background and development*, Natural Computing, 6:467–484, 2007
- A. Banks, J. Vincent, C. Anyakoha, *A review of particle swarm optimization. Part II: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications*, Natural Computing, 7:109–124, 2008

## PSO: pseudo-code

**procedure** Particle_Swarm_Optimization_for_Minimization($f(x)$)
  **foreach** *particle* $p \in ParticleSet$ **do**
    $(\vec{x}, \vec{v}) \leftarrow$ init_positions_and_velocity();
    $\mathcal{N}(p) \leftarrow$ selection_of_the_ neighbor_set();
    $\vec{x}_{pbest} \leftarrow \vec{x}$; $\vec{x}_{gbest} = \infty$; / * *init personal and global best positions* * /
  **end foreach**
  **while** ($\neg$ *stopping_criterion*)
    **foreach** *particle* $p \in ParticleSet$ **do**
      $\vec{x}_{pbest} \leftarrow \arg\max\big(f(\vec{x}), f(\vec{x}_{pbest})\big)$;
      $\vec{x}_{lbest} \leftarrow$ get_best_so_far_position_from_neighbors$\big(\mathcal{N}(p)\big)$;
      $\vec{\Delta}_{individual} \leftarrow \vec{x}_{pbest} - \vec{x}$;
      $\vec{\Delta}_{social} \leftarrow \vec{x}_{lbest} - \vec{x}$;
      $(\vec{r}_1, \vec{r}_2) \leftarrow$ random_uniform();
      $\vec{v} \leftarrow \omega\vec{v} + w_1\vec{r}_1 \circ \vec{\Delta}_{individual} + w_2\vec{r}_2 \circ \vec{\Delta}_{social}$;
      $\vec{x} \leftarrow \vec{x} + \vec{v}$;
      **if** $f(\vec{x}) < f(\vec{x}_{pbest})$
        $\vec{x}_{pbest} \leftarrow \vec{x}$;
      **if** $f(\vec{x}) < f(\vec{x}_{gbest})$
        $\vec{x}_{gbest} \leftarrow \vec{x}$;
    **end foreach**
  **end while**
**return** $f(\vec{x}_{gbest})$;

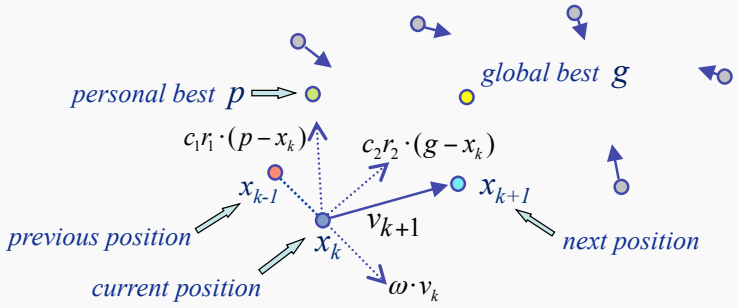


## Vector combination of multiple information



# Particle Swarm Optimization

- Population-based metaheuristic inspired by social behaviour of composed organisms (bird flocking, fish schooling) (Kennedy and Eberhart,1995)
- *Swarm Intelligence* concept: agents' (particles) exploration for optimum is improved by social interaction (sharing experience)
- Particles explore the solution space
  - particles change their positions (velocity)
  - velocity combines directions towards the current personal, local or global best positions




# Discrete PSO

- Originally developed for continuous optimization
- Many applications to combinatorial problems (TSP, VRP, Scheduling)
- DPSO approaches features
  - types of discrete solution-particle mappings
    - binary (e.g., Kennedy and Eberhart,1997)
    - real-valued (e.g.,Tasgetiren et al., 2004, 2007)
    - permutation (e.g., Lian et al., 2006)
  - types of velocity models
    - real-valued (e.g., Parsopoulos and Vrahatis, 2006)
    - stochastic (e.g., Allahverdi and Al-Anzi, 2006)
    - based on a list of moves (e.g., Clerc, 2004)

# The proposed DPSO

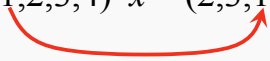
- A set of $m$ particles each associated with a solution
- **Solution** ⇔ permutation of $n$ nodes $x = (\pi_1, \ldots, \pi_n)$ with cost $Z(x_i)$
- **Velocity** ⇔ list of moves obtained as $v = x - y$
- **Insertion Move** (IM) ⇔ $(i, d)$ where $i$=node, $d$=displacement

Example $y = (1,2,3,4)$ $x = (2,3,1,4)$ $v = x - y = \{(1,2),(2,-1),(3,-1)\}$

# The proposed DPSO

- A set of $m$ particles each associated with a solution
- **Solution** ⇔ permutation of $n$ nodes $x = (\pi_1, \ldots, \pi_n)$ with cost $Z(x_i)$
- **Velocity** ⇔ list of moves obtained as $v = x - y$
- **Insertion Move** (IM) ⇔ $(i, d)$ where $i$=node, $d$=displacement

Example $y = (1,2,3,4)$ $x = (2,3,1,4)$ $v = x - y = \{(1,2),(2,-1),(3,-1)\}$



node 1 moved +2 places

# The proposed DPSO

- A set of $m$ particles each associated with a solution
- **Solution** ⇔ permutation of $n$ nodes $x = (\pi_1, \ldots, \pi_n)$ with cost $Z(x_i)$
- **Velocity** ⇔ list of moves obtained as $v = x - y$
- **Insertion Move** (IM) ⇔ $(i, d)$ where $i$=node, $d$=displacement

Example $y = (1,2,3,4)\ \ x = (2,3,1,4)\ \ v = x - y = \{(1,2),(2,-1),(3,-1)\}$

node 2 moved -1 place

# The proposed DPSO

- A set of $m$ particles each associated with a solution
- **Solution** ⇔ permutation of $n$ nodes $x = (\pi_1, \ldots, \pi_n)$ with cost $Z(x_i)$
- **Velocity** ⇔ list of moves obtained as $v = x - y$
- **Insertion Move** (IM) ⇔ $(i, d)$ where $i$=node, $d$=displacement

Example $y = (1,2,3,4)\ \ x = (2,3,1,4)\ \ v = x - y = \{(1,2),(2,-1),(3,-1)\}$

Different from usual insertion moves: inserted node is enqueued at the target position

Example $y' = y \oplus (1,2) = (-, 2, [3,1], 4)$

empty sequence place

ordered (left to right) list of nodes

# The proposed DPSO

- **Position-velocity sum** ⇔ IM iteratively applied to permutation + sequence completion procedure (SCP) $\rho(x)$

$$x = y + v \qquad v = \{(j, d)^k,\ k = 1, \ldots, p\}$$

$$s^k = s^{k-1} \oplus (j,d)^k \qquad k = 1, \ldots, p \qquad s^0 = y \qquad x + v = \rho(s^p)$$

Example $y = (1, 2, 3, 4)$ $v = \{(3,-2), (2,2)\}$

$s^1 = ([1,3], 2, -, 4)$ $s^2 = ([1,3], -, -, [4,2])$

$x=\rho(s^2)$ :

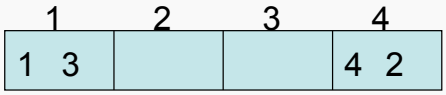
1. $p^0 = ([1,3],-,-,[4,2])$ $h=1$ (found list in place $h$: extract nodes to next empty places)
2. $p^1 = (3,1,-,[4,2])$ $h=2$ (single node in place $h$)
3. $p^2 = (3,1,-,[4,2])$ $h=3$ ($h$ is an empty place: fill $h$ with node in next non empty place)
4. $p^3 = (3,1,4,2)$ $h=4$ (single node in place $h$)

# The proposed DPSO

- The sequence completion procedure

Example: $\sigma' = \sigma + v$ $\sigma = (1,2,3,4)$ $v = \{(3,-2),(2,2)\}$

$\pi(1)=\{1, 3\}$ $\pi(2)=\pi(3)=\varnothing$ $\pi(4)=\{4, 2\}$

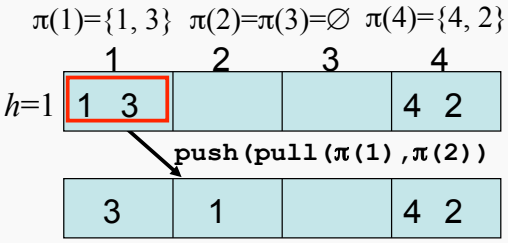



```
for each h=1,...,n do
      if |π(h)|=1 then skip;
      else if |π(h)|=0 then do
            repeat
                  k=h+1;
             while k<n and |π(k)|=0
             push(pull(π(k),π(h));
         done
      else if |π(h)|>1 then do
             while |π(h)|>1 do
                push(pull(π(h),π(h+1));
             done
      done
      endif
done
```

# The proposed DPSO

- The sequence completion procedure

Example: $\sigma' = \sigma + v$ $\sigma = (1,2,3,4)$ $v = \{(3,-2),(2,2)\}$

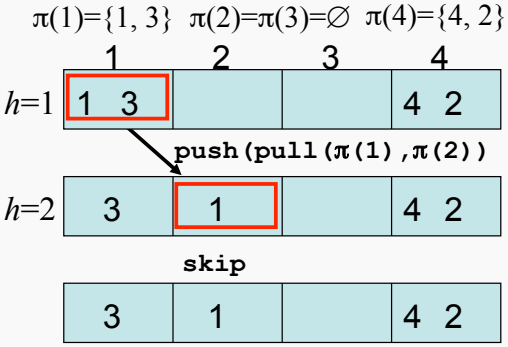



```
for each h=1,...,n do
      if |π(h)|=1 then skip;
      else if |π(h)|=0 then do
            repeat
                  k=h+1;
             while k<n and |π(k)|=0
             push(pull(π(k),π(h));
         done
      else if |π(h)|>1 then do
             while |π(h)|>1 do
                push(pull(π(h),π(h+1));
             done
      done
      endif
done
```


# The proposed DPSO

- The sequence completion procedure

Example: $\sigma' = \sigma + v$ $\sigma = (1,2,3,4)$ $v = \{(3,-2),(2,2)\}$

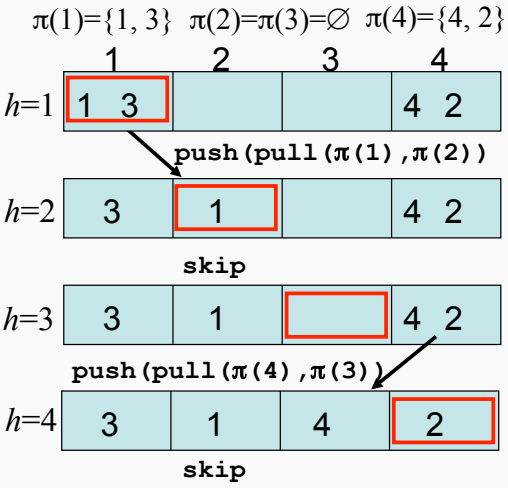


```
for each h=1,...,n do
      if |π(h)|=1 then skip;
      else if |π(h)|=0 then do
            repeat
                  k=h+1;
             while k<n and |π(k)|=0
             push(pull(π(k),π(h));
         done
      else if |π(h)|>1 then do
             while |π(h)|>1 do
                push(pull(π(h),π(h+1));
             done
      done
      endif
done
```

# The proposed DPSO

- The sequence completion procedure

Example: $\sigma' = \sigma + v$ $\sigma = (1,2,3,4)$ $v = \{(3,-2),(2,2)\}$




```
for each h=1,...,n do
      if |π(h)|=1 then skip;
      else if |π(h)|=0 then do
            repeat
                  k=h+1;
            while k<n and |π(k)|=0
            push(pull(π(k),π(h));
         done
      else if |π(h)|>1 then do
             while |π(h)|>1 do
                push(pull(π(h),π(h+1));
             done
      done
      endif
done
```

# The proposed DPSO

- **Constant-velocity multiplication** $w = c \cdot v \quad v = \{(j_1,d_1),...,(j_s,d_s)\}$
  $\Rightarrow w = \{(j_1, c \cdot d_1),...,(j_s, c \cdot d_s\}$

- **Velocity update** (iteration $k$) $\Rightarrow$ global best (**gbest**) model

$$v_i^k = w \cdot v_i^{k-1} + c_1 r_1 \cdot (p_i - \sigma_i^{k-1}) + c_2 r_2 \cdot (g - \sigma_i^{k-1})$$

$w$ = inertia parameter

$c_1$ = cognitive parameter

$c_2$ = social parameter

Velocity components (inertial, towards personal and global best) are summed one at a time

# The proposed DPSO

- Generated positions (solutions) may be not feasible for SOP ⇒ **fixing procedure**: change the node order in permutation to satisfy $R$

```
Input: x permutation, R set of precedences
Output: y feasible permutation
k = n - 1
{
  j = π_k  (node in position k)
  f = 0
  For h∈{1,..., n}:(i, j)∈ P and i = π_h
        if (h > f) then f = h;
  if (f < k) then k = k - 1;
  else
  {
        insert j in position f in x
        k = f - 2;
  }
} While k ≥ 1
y = x
```

# The proposed DPSO

- Generated positions (solutions) may be not feasible for SOP ⇒ **fixing procedure**: change the node order in permutation to satisfy $R$

Example *fixing* $(x, R)$ $x = (1, 2, 3, 4, 5, 6)$ $R = \{(4,2),(4,5),(6,3)\}$

start permutation $x = (1, 2, 3, 4, 5, 6)$

$k$=5: 5 must follow 4 ⇒ ok

$k$=4: ok

$k$=3: 3 must follow 6 ⇒ insert 3 after 6, k=4

$x = (1, 2, 4, 5, 6, 3)$

$k$=4, 3: ok

$k$=2: 2 must follow 4 ⇒ insert 2 after 4, k=1

$x = (1, 4, 2, 5, 6, 3)$

$k$=1: ok

fixed permutation = $(1, 4, 2, 5, 6, 3)$

# The proposed DPSO

- The overall algorithm

```
Input: Digraph D =(V,A), C cost matrix, R set of precedences
Output: x feasible permutation, Z(x) cost of x

Initialization of particles and velocities
While <termination condition not met>
{
        For each particle x_i
        {
                Compute tentative velocity v^t_i
                Compute tentative position x^t_i
                Fix updated position x_i and velocity v_i
                Compute x_i fitness
        }
        Intensification phase
        Udate best references
}
```

# The proposed DPSO

- **Intensification phase** ⇒ *SOP-3-exchange local search*
  - based on a combination between lexicographic search and a new labeling procedure (details in (Gambardella and Dorigo, 2000))
  - executed from the best solution found in an iteration

- **Initial sequences and velocities** are randomly generated:
  - a seed permutation $x_s$ is generated
  - *m* tentative random velocities $v^t_i$ are generated
  - tentative permutation: $x^t_i = x_s + v^t_i$
  - initial permutation: $x^0_i = fixing\ (x^t_i, R)$
  - initial velocity (fixed): $v^0_i = x^0_i - x_s$

# The proposed DPSO

- **Parameter adaptation and stagnation avoidance**
  - Particles are randomly restarted each time they coincide with $g$ (same cost)
  - $nr_k$ counter of number of restarts in an iteration (initially $nr_k$=-1)
  - parameter $c_2$ at iteration $k$+1 is adapted with rule

$$c_2^{k+1} = c_2^k - 0.01 \cdot nr_k$$

Behaviour:

Single restart ⇒ stable $c_2$

No restart ⇒ $c_2$ increased (so velocity of particles towards $g$ )

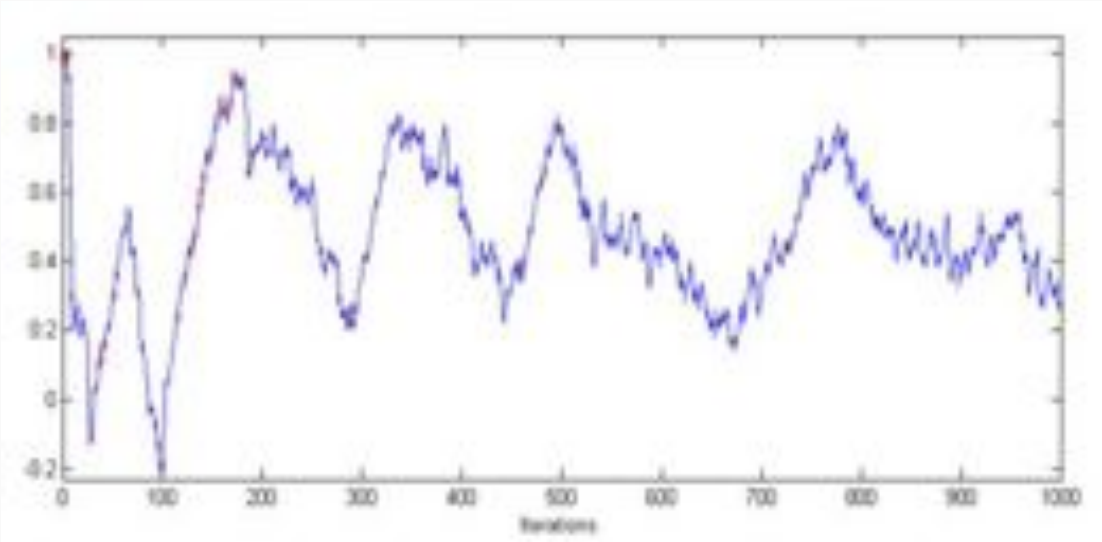
Many restarts ⇒ $c_2$ decreased (even becoming negative)

Coefficient 0.01 was obtained experimentally

# The proposed DPSO

- **Parameter adaptation and stagnation avoidance**

Example of typical oscillations of adapted $c_2$



# Experimental analysis

- DPSO for SOP coded in C++
- Tests performed on a Dual AMD Opteron 250 2.4GHz/4GB PC
- The benchmark: SOPLIB (www.idsia.ch/~roberto/SOPLIB06.zip) used in (Montemanni, Smith, Gambardella, 2007, 2008) for testing HAS-SOP and APC+HAS-SOP algorithms
  - 48 random instances denoted as $n - r - p$
  - $n \in \{200, 300, 400, 500, 600, 700\}$ number of nodes
  - $r \in \{100, 1000\}$ upper bound of cost range $c_{ij} \sim U[0, r]$
  - $p$ approximate % of precedence constraints

# Experimental analysis

- Tuning $w$ and $c_1$ parameters
  - Fixed $w = c_1 \in \{1, 2, 4, 6, 8, 10\}$
  - Detailed neighbour of configuration with best average $w = c_1 = 4$
  - Experimented sample configurations with $w \neq c_1$

**Results**

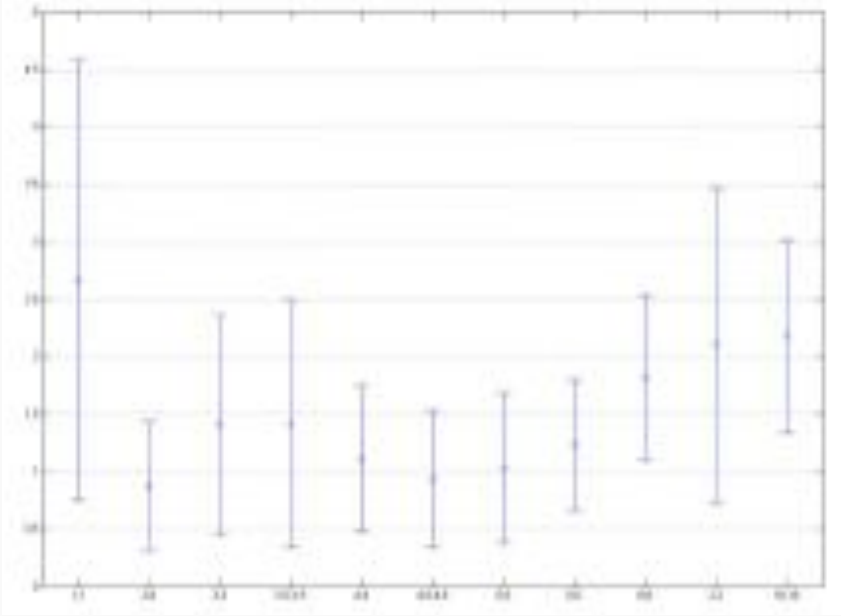
Outliers:
2 instances with p=1

Best with $w \neq c_1$:
$w=2$, $c_1=6$
Avg dev=0.85%
Conf=0.53%

| Overall | | | | Without outliers | | | |
|---|---|---|---|---|---|---|---|
| $w$ | $c_1$ | Avg dev | Conf | $w$ | $c_1$ | Avg dev | Conf |
| 6 | 6 | 1.52% | 0.67% | 4.5 | 4.5 | 0.91% | 0.56% |
| 5 | 5 | 1.75% | 1.19% | 5 | 5 | 1.00% | 0.61% |
| 8 | 8 | 1.85% | 0.67% | 4 | 4 | 1.08% | 0.60% |
| 4.5 | 4.5 | 1.93% | 1.50% | 6 | 6 | 1.21% | 0.54% |
| 4 | 4 | 2.09% | 1.53% | 3 | 3 | 1.37% | 0.91% |
| 10 | 10 | 2.34% | 0.83% | 3.5 | 3.5 | 1.38% | 1.02% |
| 3.5 | 3.5 | 2.37% | 1.68% | 8 | 8 | 1.77% | 0.68% |
| 3 | 3 | 2.60% | 1.89% | 2 | 2 | 2.06% | 1.30% |
| 2 | 2 | 3.94% | 3.00% | 10 | 10 | 2.12% | 0.80% |
| 1 | 1 | 4.47% | 3.11% | 1 | 1 | 2.60% | 1.83% |

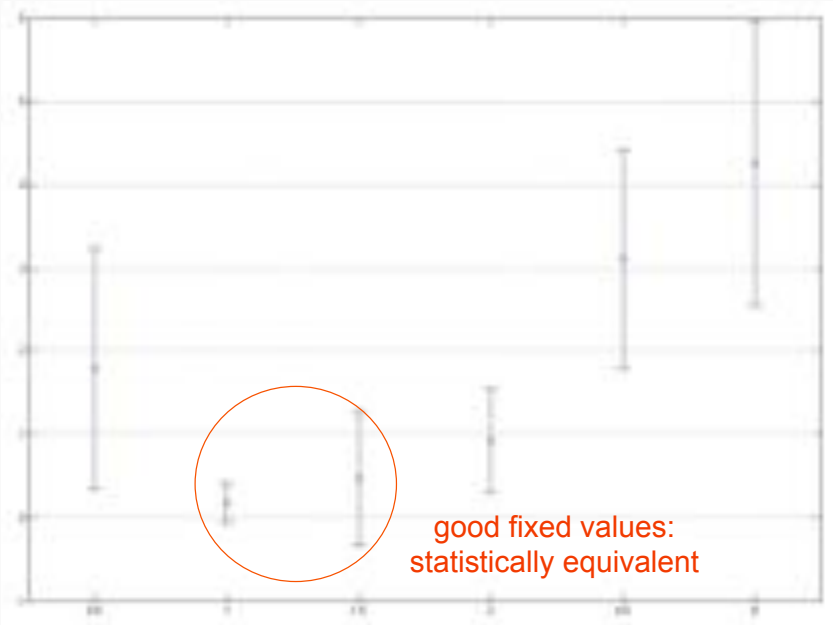
# Experimental analysis

- No configuration is statistically dominant: DPSO not very sensitive to parameters



# Experimental analysis

- Effectiveness of parameter adaptation: average deviations from DPSO with adaptation when using fixed $c_2$



# Experimental analysis

- Comparison with HAS-SOP and APC+HAS-SOP

  Same experimental condition of (Montemanni, Smith, Gambardella, 2007, 2008): 10 run for instance, stop after 600 seconds

Average deviations of DPSO (comparing average results)

| | | HAS | Conf. Int. (95%) | | APC+HAS | Conf. Int. (95%) | |
|---|---|---|---|---|---|---|---|
| *r* | 100 | -15.31% | -22.65% | -7.97% | -14.02% | -21.10% | -6.94% |
| | 1000 | -5.32% | -7.38% | -3.25% | -4.53% | -6.36% | -2.71% |
| *p* | 1 | -26.14% | -37.26% | -15.03% | -24.45% | -35.27% | -13.63% |
| | 15 | -12.81% | -14.92% | -10.70% | -11.18% | -13.30% | -9.06% |
| | 30 | -2.22% | -3.24% | -1.21% | -1.42% | -2.19% | -0.65% |
| | 60 | -0.08% | -0.12% | -0.04% | -0.05% | -0.09% | -0.02% |
| *n* | 200 | -5.63% | -10.87% | -0.38% | -5.04% | -9.89% | -0.18% |
| | 300 | -10.49% | -21.40% | 0.42% | -9.49% | -19.73% | 0.75% |
| | 400 | -11.53% | -22.75% | -0.31% | -10.45% | -21.16% | 0.26% |
| | 500 | -12.85% | -25.29% | -0.40% | -11.92% | -24.52% | 0.68% |
| | 600 | -11.31% | -22.35% | -0.27% | -10.13% | -20.70% | 0.45% |
| | 700 | -10.07% | -19.54% | -0.61% | -8.63% | -17.14% | -0.12% |
| Overall | | -10.31% | -14.35% | -6.28% | -9.28% | -13.14% | -5.41% |

# Experimental analysis

Average deviations of DPSO (comparing best results)

| | | HAS | Conf. Int. (95%) | | APC+HAS | Conf. Int. (95%) | |
|---|---|---|---|---|---|---|---|
| *r* | 100 | -17.78% | -27.33% | -8.24% | -15.72% | -24.72% | -6,71% |
| | 1000 | -5.76% | -8.10% | -3.43% | -4.89% | -6.86% | -2,92% |
| *p* | 1 | -33.22% | -47.50% | -18.93% | -30.09% | -43.78% | -16,41% |
| | 15 | -12.21% | -14.43% | -9.98% | -10.13% | -12.20% | -8,05% |
| | 30 | -1.61% | -2.55% | -0.67% | -0.98% | -1.52% | -0,43% |
| | 60 | -0.06% | -0.11% | -0.02% | 0.00% | -0.04% | 0,03% |
| *n* | 200 | -5.92% | -11.80% | -0.04% | -4.67% | -9.51% | 0,17% |
| | 300 | -10.41% | -22.41% | 1.58% | -8.82% | -19.30% | 1,66% |
| | 400 | -12.58% | -26.07% | 0.92% | -11.29% | -23.86% | 1,27% |
| | 500 | -14.74% | -31.04% | 1.57% | -13.75% | -30.15% | 2,64% |
| | 600 | -13.19% | -26.96% | 0.58% | -11.82% | -25.11% | 1,47% |
| | 700 | -13.80% | -28.87% | 1.26% | -11.45% | -24.45% | 1,55% |
| Overall | | -11.77% | -16.93% | -6.62% | -10.30% | -15.11% | -5.49% |

# Experimental analysis

Percentage of best known solutions found by DPSO

| | Improved | Equal | Worse |
|---|---|---|---|
| HAS-SOP | 89.58% | 10.42% | 0.00% |
| APC+HAS-SOP | 79.17% | 22.92% | 2.08% |

- Further tests:
  - Multi-start LS: overall Avg dev from DPSO = 20.58% (conf.=9.69%)
  - Random DPSO (positions updated with random velocities): overall Avg dev from DPSO = 19.67% (conf.=10.27%)

- TSPLIB benchmark (best results over 10 runs)
  - All best known found apart from *prob.100* (1.93%), *rbg358a* (0.20%) and *rbg378a* (0.04%)
  - Improved best known for *rbg323a* (-0.03%) (resisting since ’90)

# Conclusions

- We introduced a DPSO for SOP which incorporates a parameter adaptation mechanism and avoids stagnation
- Effectiveness shown by experimental tests
- DPSO appeared very effective in guiding an underlying LS procedure as a diversification device
- DPSO with parameter adaptation mechanism appeared not very sensitive to parameter values

- Future development: analyse this method on similar combinatorial problems (i.e., problems sharing the same combinatorial structure) for which a powerful LS procedure is available

# Outline

- Introduction
- Problem description
- A Genetic Algorithm
- A Hybrid Ant System
- A Heuristic Manipulation Technique
- A Particle Swarm Optimization
- An Enhanced Ant Colony System
- A Shared Incumbent Environment
- Bibliography

## An Enhanced Ant Colony System

L.M. Gambardella, R. Montemanni.
An Enhanced Ant Colony System for two Transportation Problems.
Proceedings of Tristan VII, pages 292-295, Tromsø, Norway, 20-25 June 2010

L.M. Gambardella, R. Montemanni, D.A. Weyland.
An Enhanced Ant Colony System for the Sequential Ordering Problem.
Proceedings of OR 2011, Zurich, Switzerland, 30 August-1 September 2011

L.M. Gambardella, R. Montemanni, D.A. Weyland.
Coupling Ant Colony Systems with strong Local Searches.
European Journal of Operational Research, to appear

## ACS Guiding Principles

ACS analysis over many combinatorial optimization problems.

ACS increases pheromone trail on edges belonging to high quality solutions. Pheromone drives the search towards promising regions of the search space.

Efficient combination of constructive procedure and local search.

The constructive phase can be seen as a diversification process. The algorithm works when new solutions are in the neighborhood of the best solution computed so far.

The local search is considered as an intensification phase.

## Ant Colony drawback

One known drawback of the ACS approach is the long total running time required to build new solutions by each artificial ant.

Usually the constructive process takes time O(|V|) for each of the |V| steps required.

This is acceptable in case of small problems, but it is too expensive in case of larger problems.

In fact, ACO algorithms did not show the same performance as in case of small instances when dealing with large routing and scheduling instances.

# Enhanced Ant Colony System

**We propose to modify ACS in two directions:**

1. More efficient constructive procedure

2. Better integration between constructive procedure and local search

   We will present results of these new directions on the Sequential Ordering Problem (SOP)

# Enhanced Ant Colony

**Constructive procedure**

we directly considers the **best solution** computed so far already during the constructive phase.

In node *r* with probability $q_0$, the selected edge is the edge outgoing from node *r* in the best solution computed so far (in case this edge is not feasible, the classic mechanism described above is applied).

Since probability $q_0$ is usually grater than 0.9, the new approach drastically reduces the running time required to select the next edge to visit (typically from O(|V|) to O(1)) and to build a new solution.

# Enhanced Ant Colony

## Local search

1) We apply the local search procedure only on a **promising subset of the solutions** generated by ants, where the subset usually depends on the problem under investigation, and on the running history of the algorithm.

2) the local search is (probabilistically) applied only on those **solution** which have not been already optimized in recent iterations (in order to avoid searching the neighborhood of the same solutions over and over again).

Notice that the local search enhancements are again in the direction of reducing the total running time

# ACS for SOP: Hybrid Ant System

- Dorigo, Gambardella 2000
- Constructive phase based on ACS
- Trail updating as ACS
- New local search SOP-3_exchange strategy based on a combination between lexicographic search and a new labeling procedure.
- New data structure to drive the search
- First in literature that uses a local search edge-exchange strategy to directly handle multiple constraints without any increase in computational time.

## ACS for SOP

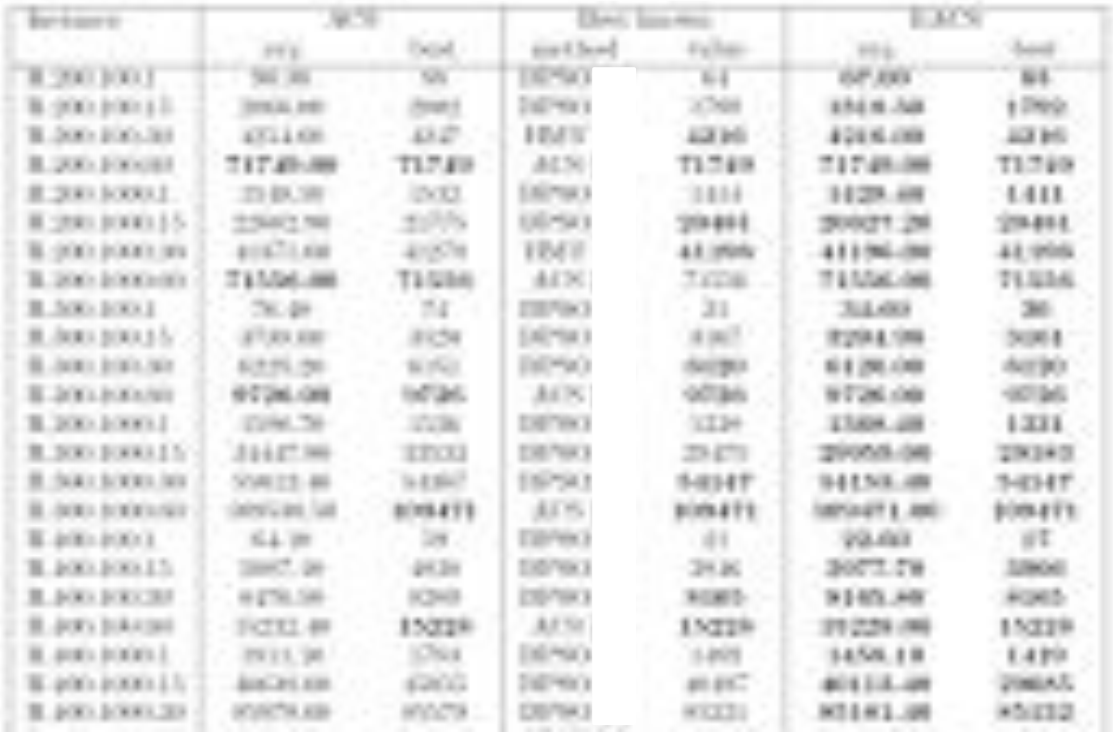
- Each ant iteratively starts from node 0 and adds new nodes until all nodes have been visited and node n is reached.
- When in node i, an ant chooses probabilistically the next node j from the set F(i) of feasible nodes.
- F(i) contains all the nodes j still to be visited and such that all nodes that have to precede j, according to precedence constraints, have already been inserted in the sequence

## Experimental Results

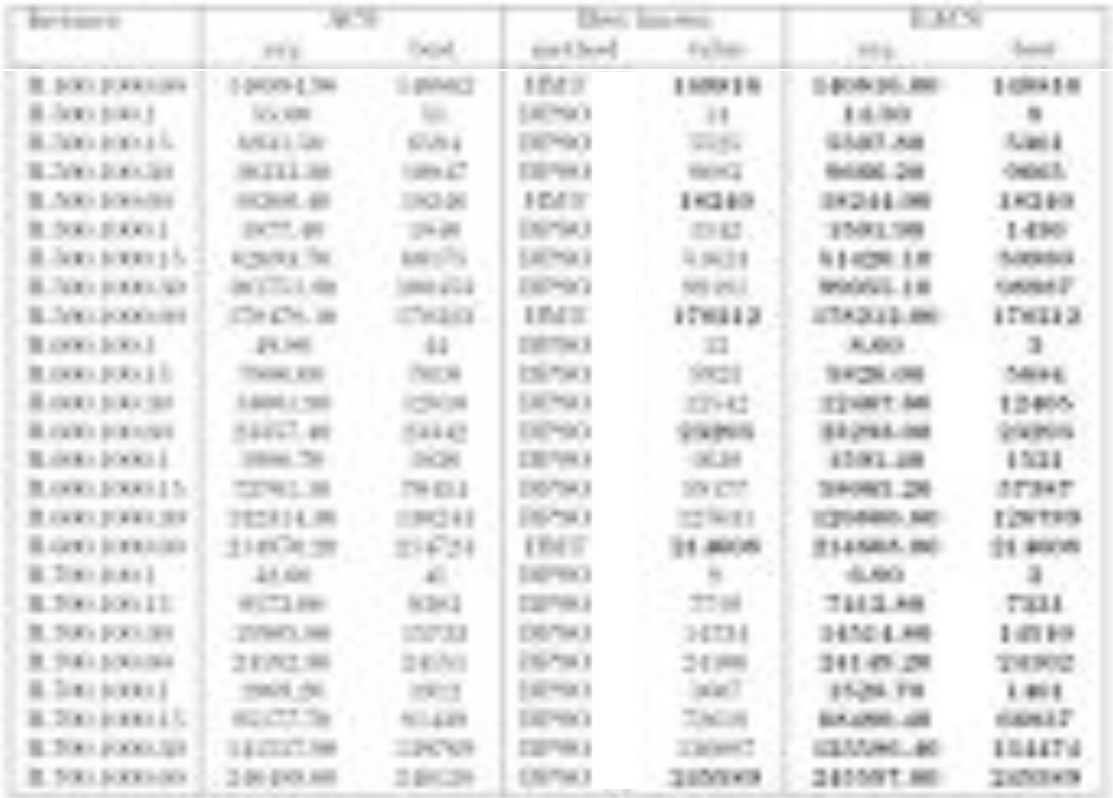
- SOP problems are in SOPLIB2006
- Each instance has the following structure R **n-r-p** where
- n is the number of nodes of the problem
- r is the cost range, i.e., cij $\in$ [0, r] $\forall$i, j $\in$ V
- p is the approximate percentage of precedence constraints, since the number of precedence constraints imposed for an instance is about (p/100)(n (n − 1)/2).
- The benchmark is made of 48 instances generated by combining the following values for the considered parameters, n $\in$ {200 300 400 500 600 700}, r $\in$ {100 1000}, p $\in$ {1 15 30 60}.

- Experiments are run for 600 sec. each and are averaged over 10 trials
- Comparisons are against the original ACS and the current best know algorithms

# EACS for SOP



**ACS** (Gambardella, Dorigo, 2000)
**HMT** (Montemanni, Smith, Gambardella, 2008)
**DPSO** (Anghinolfi, Montemanni, Paolucci, Gambardella, 2009)



**ACS** (Gambardella, Dorigo, 2000)
**HMT** (Montemanni, Smith, Gambardella, 2008)
**DPSO** (Anghinolfi, Montemanni, Paolucci, Gambardella, 2009)

## Experimental Results

- EACS for SOP
  - Enhanced constructive procedure
  - Enhanced integration between Ants and Local search

- 48 instances.
- 48 best known found (16 equal to the previous best known solutions [optimal?]).
- 32 best results improved.

# Conclusions

- EACS is a modification of the original Ant Colony System paradigm, aiming at overcoming its main drawbacks:
  - Slow constructive phase
  - Bad integration with local search
- EACS vs ACS:
  - A faster construction phase using the best known solution retrieved so far
  - Local search is called only when it is likely it can help
  - All together: faster
- Results on SOP show that the approach is promising
- Investigations using other combinatorial optimization problems are running.

# Outline



## A Shared Incumbent Environment

M. Mojana, R. Montemanni, G. Di Caro, L.M. Gambardella,
An Algorithm combining Linear Programming and an Ant Colony System for the Sequential Ordering Problem.
Proceedings of ATAI 2011, pages 80-85, Singapore, 24-25 November 2011.

## Two-Commodity Network Flow Formulation

- Mixed integer linear program (MILP)
- 2-commodity flow formulation
- Compact model
- No experimental study on SOP instances available

Moon, C., Kim, J., Choi, G. and Seo, Y. [2002]. *An efficient genetic algorithm for the traveling salesman problem with precedence constraints*, European Journal of Operational Research 140(3): 606–617.

## Two-Commodity Network Flow Formulation

$$\text{Min.} \quad \sum_{i \in V} \sum_{j \in V} \frac{1}{|V|-1} c_{ij} \left( y_{ij}^p + y_{ij}^q \right) \tag{1}$$

$$\text{s.t.} \quad \sum_{j \in V} y_{ij}^p - \sum_{j \in V} y_{ji}^p = \begin{cases} |V|-1 & \text{for } i = 1, \\ -1 & \text{otherwise} \end{cases} \quad \forall i \in V \tag{2}$$

$$\sum_{j \in V} y_{ij}^q - \sum_{j \in V} y_{ji}^q = \begin{cases} 1-|V| & \text{for } i = 1, \\ 1 & \text{otherwise} \end{cases} \quad \forall i \in V \tag{3}$$

$$\sum_{j \in V} y_{ij}^p + y_{ij}^q = |V|-1 \quad \forall i \in V \tag{4}$$

$$y_{ij}^p + y_{ij}^q = (|V|-1)\, y_{ij} \quad \forall i,j \in V \tag{5}$$

$$\sum_{j \in V} y_{uj}^p - \sum_{j \in V} y_{vj}^p \geq 1 \quad \forall (u,v) \in R \tag{6}$$

$$y_{ij}^p \geq 0, y_{ij}^q \geq 0 \quad \forall i,j \in V \tag{7}$$

$$y_{ij} \in \{0,1\} \quad \forall i,j \in V \tag{8}$$

## Two-Commodity Network Flow Formulation

$$\text{Min.} \quad \sum_{i \in V} \sum_{j \in V} \frac{1}{|V|-1} c_{ij} \left( y_{ij}^p + y_{ij}^q \right)$$

$$\text{s.t.} \quad \sum_{j \in V} y_{ij}^p - \sum_{j \in V} y_{ji}^p = \begin{cases} |V|-1 & \text{for } i = 1, \\ -1 & \text{otherwise} \end{cases} \quad \forall i \in V$$

$$\sum_{j \in V} y_{ij}^q - \sum_{j \in V} y_{ji}^q = \begin{cases} 1-|V| & \text{for } i = 1, \\ 1 & \text{otherwise} \end{cases} \quad \forall i \in V$$

$$\sum_{j \in V} y_{ij}^p + y_{ij}^q = |V|-1 \quad \forall i \in V$$

$$y_{ij}^p + y_{ij}^q = (|V|-1)\, y_{ij} \quad \forall i,j \in V$$

$$\sum_{j \in V} y_{uj}^p - \sum_{j \in V} y_{vj}^p \geq 1 \quad \forall (u,v) \in R$$

$$y_{ij}^p \geq 0, y_{ij}^q \geq 0 \quad \forall i,j \in V$$

$$y_{ij} \in \{0,1\} \quad \forall i,j \in V$$

If $y_{ij} = 1$: arc $(i,j)$ carries flow

## Two-Commodity Network Flow Formulation

$$\text{Min.} \quad \sum_{i \in V} \sum_{j \in V} \frac{1}{|V|-1} c_{ij} \left( y_{ij}^p + y_{ij}^q \right)$$

$$\text{s.t.} \quad \sum_{j \in V} y_{ij}^p - \sum_{j \in V} y_{ji}^p = \begin{cases} |V|-1 & \text{for } i = 1, \\ -1 & \text{otherwise} \end{cases} \quad \forall i \in V$$

$$\sum_{j \in V} y_{ij}^q - \sum_{j \in V} y_{ji}^q = \begin{cases} 1-|V| & \text{for } i = 1, \\ 1 & \text{otherwise} \end{cases} \quad \forall i \in V$$

$$\sum_{j \in V} y_{ij}^p + y_{ij}^q = |V|-1 \quad \forall i \in V$$

$$y_{ij}^p + y_{ij}^q = (|V|-1)\, y_{ij} \quad \forall i,j \in V$$

$$\sum_{j \in V} y_{uj}^p - \sum_{j \in V} y_{vj}^p \geq 1 \quad \forall (u,v) \in R$$

$$y_{ij}^p \geq 0, y_{ij}^q \geq 0 \quad \forall i,j \in V$$

Flow of commodity $p$ and $q$ on arc $(i,j)$

$$y_{ij} \in \{0,1\} \quad \forall i,j \in V$$

If $y_{ij} = 1$: arc $(i,j)$ carries flow

## Two-Commodity Network Flow Formulation

$$\text{Min.} \quad \sum_{i \in V} \sum_{j \in V} \frac{1}{|V|-1} c_{ij} \left( y_{ij}^p + y_{ij}^q \right)$$

$$\text{s.t.} \quad \sum_{j \in V} y_{ij}^p - \sum_{j \in V} y_{ji}^p = \begin{cases} |V|-1 & \text{for } i = 1, \\ -1 & \text{otherwise} \end{cases} \quad \forall i \in V$$

$$\sum_{j \in V} y_{ij}^q - \sum_{j \in V} y_{ji}^q = \begin{cases} 1-|V| & \text{for } i = 1, \\ 1 & \text{otherwise} \end{cases} \quad \forall i \in V$$

$$\sum_{j \in V} y_{ij}^p + y_{ij}^q = |V|-1 \quad \forall i \in V$$

$$y_{ij}^p + y_{ij}^q = (|V|-1)\, y_{ij} \quad \forall i,j \in V$$

$$\sum_{j \in V} y_{uj}^p - \sum_{j \in V} y_{vj}^p \geq 1 \quad \forall (u,v) \in R$$ Enforces the precedence relation

$$y_{ij}^p \geq 0, y_{ij}^q \geq 0 \quad \forall i,j \in V$$ Flow of commodity $p$ and $q$ on arc $(i,j)$

$$y_{ij} \in \{0,1\} \quad \forall i,j \in V$$ If $y_{ij} = 1$: arc $(i,j)$ carries flow

## Two-Commodity Network Flow Formulation

$$\text{Min.} \quad \sum_{i \in V} \sum_{j \in V} \frac{1}{|V|-1} c_{ij} \left( y_{ij}^p + y_{ij}^q \right)$$

$$\text{s.t.} \quad \sum_{j \in V} y_{ij}^p - \sum_{j \in V} y_{ji}^p = \begin{cases} |V|-1 & \text{for } i = 1, \\ -1 & \text{otherwise} \end{cases} \quad \forall i \in V$$

$$\sum_{j \in V} y_{ij}^q - \sum_{j \in V} y_{ji}^q = \begin{cases} 1-|V| & \text{for } i = 1, \\ 1 & \text{otherwise} \end{cases} \quad \forall i \in V$$

$$\sum_{j \in V} y_{ij}^p + y_{ij}^q = |V|-1 \quad \forall i \in V$$

$$y_{ij}^p + y_{ij}^q = (|V|-1)\, y_{ij} \quad \forall i,j \in V$$ Constant flow on active arcs

$$\sum_{j \in V} y_{uj}^p - \sum_{j \in V} y_{vj}^p \geq 1 \quad \forall (u,v) \in R$$ Enforces the precedence relation

$$y_{ij}^p \geq 0, y_{ij}^q \geq 0 \quad \forall i,j \in V$$ Flow of commodity $p$ and $q$ on arc $(i,j)$

$$y_{ij} \in \{0,1\} \quad \forall i,j \in V$$ If $y_{ij} = 1$: arc $(i,j)$ carries flow

## Two-Commodity Network Flow Formulation

$$\text{Min.} \quad \sum_{i \in V} \sum_{j \in V} \frac{1}{|V|-1} c_{ij} \left( y_{ij}^p + y_{ij}^q \right)$$

$$\text{s.t.} \quad \sum_{j \in V} y_{ij}^p - \sum_{j \in V} y_{ji}^p = \begin{cases} |V|-1 & \text{for } i = 1, \\ -1 & \text{otherwise} \end{cases} \quad \forall i \in V$$

$$\sum_{j \in V} y_{ij}^q - \sum_{j \in V} y_{ji}^q = \begin{cases} 1-|V| & \text{for } i = 1, \\ 1 & \text{otherwise} \end{cases} \quad \forall i \in V$$

$$\sum_{j \in V} y_{ij}^p + y_{ij}^q = |V|-1 \quad \forall i \in V$$ Nodes total outgoing flow

$$y_{ij}^p + y_{ij}^q = (|V|-1)\, y_{ij} \quad \forall i,j \in V$$ Constant flow on active arcs

$$\sum_{j \in V} y_{uj}^p - \sum_{j \in V} y_{vj}^p \geq 1 \quad \forall (u,v) \in R$$ Enforces the precedence relation

$$y_{ij}^p \geq 0, y_{ij}^q \geq 0 \quad \forall i,j \in V$$ Flow of commodity $p$ and $q$ on arc $(i,j)$

$$y_{ij} \in \{0,1\} \quad \forall i,j \in V$$ If $y_{ij} = 1$: arc $(i,j)$ carries flow

## Two-Commodity Network Flow Formulation

$$\text{Min.} \quad \sum_{i \in V} \sum_{j \in V} \frac{1}{|V|-1} c_{ij} \left( y_{ij}^p + y_{ij}^q \right)$$

$$\text{s.t.} \quad \sum_{j \in V} y_{ij}^p - \sum_{j \in V} y_{ji}^p = \begin{cases} |V|-1 & \text{for } i = 1, \\ -1 & \text{otherwise} \end{cases} \quad \forall i \in V$$

$$\sum_{j \in V} y_{ij}^q - \sum_{j \in V} y_{ji}^q = \begin{cases} 1-|V| & \text{for } i = 1, \\ 1 & \text{otherwise} \end{cases} \quad \forall i \in V$$ $q$ net production

$$\sum_{j \in V} y_{ij}^p + y_{ij}^q = |V|-1 \quad \forall i \in V$$ Nodes total outgoing flow

$$y_{ij}^p + y_{ij}^q = (|V|-1)\, y_{ij} \quad \forall i,j \in V$$ Constant flow on active arcs

$$\sum_{j \in V} y_{uj}^p - \sum_{j \in V} y_{vj}^p \geq 1 \quad \forall (u,v) \in R$$ Enforces the precedence relation

$$y_{ij}^p \geq 0, y_{ij}^q \geq 0 \quad \forall i,j \in V$$ Flow of commodity $p$ and $q$ on arc $(i,j)$

$$y_{ij} \in \{0,1\} \quad \forall i,j \in V$$ If $y_{ij} = 1$: arc $(i,j)$ carries flow

## Two-Commodity Network Flow Formulation

$$\text{Min.} \quad \sum_{i \in V} \sum_{j \in V} \frac{1}{|V|-1} c_{ij} \left( y_{ij}^p + y_{ij}^q \right)$$

$$\text{s.t.} \quad \sum_{j \in V} y_{ij}^p - \sum_{j \in V} y_{ji}^p = \begin{cases} |V|-1 & \text{for } i = 1, \\ -1 & \text{otherwise} \end{cases} \quad \forall i \in V$$

$p$ net production

$$\sum_{j \in V} y_{ij}^q - \sum_{j \in V} y_{ji}^q = \begin{cases} 1-|V| & \text{for } i = 1, \\ 1 & \text{otherwise} \end{cases} \quad \forall i \in V$$

$q$ net production

$$\sum_{j \in V} y_{ij}^p + y_{ij}^q = |V|-1 \quad \forall i \in V$$

Nodes total outgoing flow

$$y_{ij}^p + y_{ij}^q = (|V|-1)\, y_{ij} \quad \forall i,j \in V$$

Constant flow on active arcs

$$\sum_{j \in V} y_{uj}^p - \sum_{j \in V} y_{vj}^p \geq 1 \quad \forall (u,v) \in R$$

Enforces the precedence relation

$$y_{ij}^p \geq 0, y_{ij}^q \geq 0 \quad \forall i,j \in V$$

Flow of commodity $p$ and $q$ on arc $(i,j)$

$$y_{ij} \in \{0,1\} \quad \forall i,j \in V$$

If $y_{ij} = 1$: arc $(i,j)$ carries flow

## Two-Commodity Network Flow Formulation

$$\text{Min.} \quad \sum_{i \in V} \sum_{j \in V} \frac{1}{|V|-1} c_{ij} \left( y_{ij}^p + y_{ij}^q \right)$$

Minimize total tour cost

$$\text{s.t.} \quad \sum_{j \in V} y_{ij}^p - \sum_{j \in V} y_{ji}^p = \begin{cases} |V|-1 & \text{for } i = 1, \\ -1 & \text{otherwise} \end{cases} \quad \forall i \in V$$

$p$ net production

$$\sum_{j \in V} y_{ij}^q - \sum_{j \in V} y_{ji}^q = \begin{cases} 1-|V| & \text{for } i = 1, \\ 1 & \text{otherwise} \end{cases} \quad \forall i \in V$$

$q$ net production

$$\sum_{j \in V} y_{ij}^p + y_{ij}^q = |V|-1 \quad \forall i \in V$$

Nodes total outgoing flow

$$y_{ij}^p + y_{ij}^q = (|V|-1)\, y_{ij} \quad \forall i,j \in V$$

Constant flow on active arcs

$$\sum_{j \in V} y_{uj}^p - \sum_{j \in V} y_{vj}^p \geq 1 \quad \forall (u,v) \in R$$

Enforces the precedence relation

$$y_{ij}^p \geq 0, y_{ij}^q \geq 0 \quad \forall i,j \in V$$

Flow of commodity $p$ and $q$ on arc $(i,j)$

$$y_{ij} \in \{0,1\} \quad \forall i,j \in V$$

If $y_{ij} = 1$: arc $(i,j)$ carries flow

## Shared Incumbent Environment: Idea (1)

- Combine metaheuristic and MILP approach

## Shared Incumbent Environment: Idea (1)

- Combine metaheuristic and MILP approach
- Both have strengths and weaknesses

## Shared Incumbent Environment: Idea (1)

- Combine metaheuristic and MILP approach
- Both have strengths and weaknesses
- Building blocks: HAS-SOP and IBM ILOG CPLEX solving MILP

## Shared Incumbent Environment: Idea (1)

- Combine metaheuristic and MILP approach
- Both have strengths and weaknesses
- Building blocks: HAS-SOP and IBM ILOG CPLEX solving MILP
- Both algorithms generate feasible solutions

## Shared Incumbent Environment: Idea (1)

- Combine metaheuristic and MILP approach
- Both have strengths and weaknesses
- Building blocks: HAS-SOP and IBM ILOG CPLEX solving MILP
- Both algorithms generate feasible solutions
- Run in parallel and take the best:
  - CPU time $\times 2$ cores
  - HAS-SOP: SOP-centered vision (+), UB progression (+), stagnation (-)
  - CPLEX: MILP-centered vision (+), explores non-promising subtrees (-)


## Shared Incumbent Environment: Idea (2)

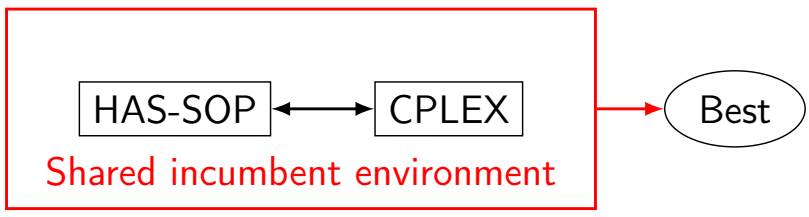
- Solution: a tighter integration

## Shared Incumbent Environment: Idea (2)

- Solution: a tighter integration
- Cooperate instead of compete

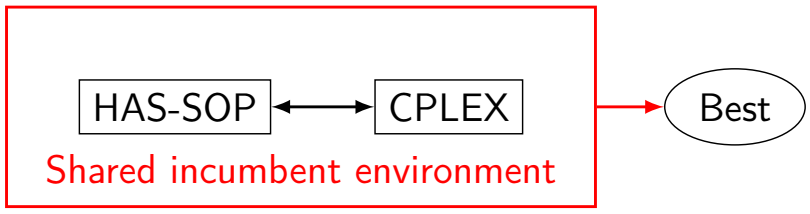
## Shared Incumbent Environment: Idea (2)

- Solution: a tighter integration
- Cooperate instead of compete
- Share incumbent solutions (both directions)



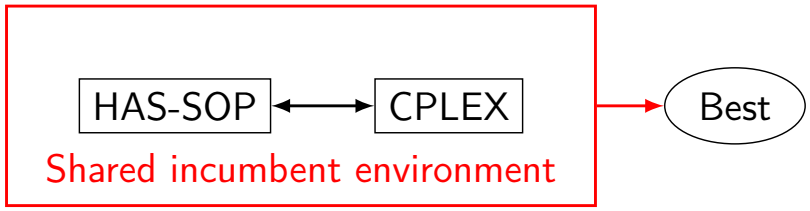
## Shared Incumbent Environment: Idea (2)

- Solution: a tighter integration
- Cooperate instead of compete
- Share incumbent solutions (both directions)
- HAS-SOP → CPLEX
  - Prune more nodes → search space reduction
  - Repeat presolve → model size reduction




## Shared Incumbent Environment: Idea (2)

- Solution: a tighter integration
- Cooperate instead of compete
- Share incumbent solutions (both directions)
- HAS-SOP → CPLEX
  - Prune more nodes → search space reduction
  - Repeat presolve → model size reduction
- CPLEX → HAS-SOP
  - Escape from stagnation
  - Global view of the problem



## Benchmark Instances / Experimental settings

- SOPLIB2006
- Reference instances for the recent SOP literature
- 48 instances
- No instance solved to optimality yet (mainly lack of lower bounds)

## Benchmark Instances / Experimental settings

- SOPLIB2006
- Reference instances for the recent SOP literature
- 48 instances
- No instance solved to optimality yet (mainly lack of lower bounds)
- CPLEX 12.1 (2 cores) vs Shared Incumbent Environment (1+1 cores) vs Metaheuristics (1 core)
- Maximum wall time: 600 s
- Computer: Dual AMD Opteron 250 2.4 GHz/4GB

# SIE vs CPLEX (1)

| Instance | CPLEX [LB; UB] | Closed | SIE [LB; UB] | Closed | SIE vs CPLEX LB | UB |
|---|---|---|---|---|---|---|
| R.200.100.1 | [ 61 ; 61 ] | ⋆ | [ 61 ; 61 ] | ⋆ | = | = |
| R.200.100.15 | [ 1013 ; ∞ ] | | [ 1091 ; 2200 ] | | + | − |
| R.200.100.30 | [ 3635 ; ∞ ] | | [ 3693 ; 4216 ] | | + | − |
| R.200.100.60 | [ 71749 ; 71749 ] | ⋆ | [ 71749 ; 71749 ] | ⋆ | = | = |
| R.200.1000.1 | [ 1404 ; 1404 ] | ⋆ | [ 1404 ; 1404 ] | ⋆ | = | = |
| R.200.1000.15 | [ 12673 ; ∞ ] | | [ 13364 ; 22636 ] | | + | − |
| R.200.1000.30 | [ 35777 ; ∞ ] | | [ 37103 ; 41201 ] | | + | − |
| R.200.1000.60 | [ 71556 ; 71556 ] | ⋆ | [ 71556 ; 71556 ] | ⋆ | = | = |
| R.300.100.1 | [ 26 ; ∞ ] | | [ 26 ; 26 ] | ⋆ | = | − |
| R.300.100.15 | [ 1712 ; ∞ ] | | [ 1839 ; 3817 ] | | + | − |
| R.300.100.30 | [ 5103 ; ∞ ] | | [ 5259 ; 6214 ] | | + | − |
| R.300.100.60 | [ 9726 ; 9726 ] | ⋆ | [ 9726 ; 9726 ] | ⋆ | + | − |
| R.300.1000.1 | [ 1292 ; 1296 ] | | [ 1292 ; 1605 ] | | = | + |
| R.300.1000.15 | [ 17888 ; ∞ ] | | [ 19265 ; 36796 ] | | + | − |
| R.300.1000.30 | [ 45213 ; ∞ ] | | [ 46665 ; 54480 ] | | + | − |
| R.300.1000.60 | [ 109471 ; 109471 ] | ⋆ | [ 109471 ; 109471 ] | ⋆ | = | = |
| R.400.100.1 | [ 13 ; ∞ ] | | [ 13 ; 13 ] | ⋆ | = | − |
| R.400.100.15 | [ 2358 ; ∞ ] | | [ 2494 ; 5132 ] | | + | − |
| R.400.100.30 | [ 6769 ; ∞ ] | | [ 7117 ; 8408 ] | | + | − |
| R.400.100.60 | [ 15228 ; 15228 ] | ⋆ | [ 15228 ; 15228 ] | ⋆ | = | = |
| R.400.1000.1 | [ 1342 ; ∞ ] | | [ 1336 ; 1932 ] | | − | − |
| R.400.1000.15 | [ 23595 ; ∞ ] | | [ 25518 ; 46653 ] | | + | − |
| R.400.1000.30 | [ 69542 ; ∞ ] | | [ 72280 ; 86854 ] | | + | − |
| R.400.1000.60 | [ 140816 ; 140816 ] | ⋆ | [ 140816 ; 140816 ] | ⋆ | = | = |

# SIE vs CPLEX (2)

| Instance | CPLEX [LB; UB] | Closed | SIE [LB; UB] | Closed | SIE vs CPLEX LB | UB |
|---|---|---|---|---|---|---|
| R.500.100.1 | [ 4 ; ∞ ] | | [ 4 ; 60 ] | | = | − |
| R.500.100.15 | [ 3030 ; ∞ ] | | [ 3111 ; 6959 ] | | + | − |
| R.500.100.30 | [ 7590 ; ∞ ] | | [ 7792 ; 10244 ] | | + | − |
| R.500.100.60 | [ 18240 ; 18240 ] | ⋆ | [ 18240 ; 18240 ] | ⋆ | = | = |
| R.500.1000.1 | [ 1315 ; ∞ ] | | [ 1313 ; 2016 ] | | − | − |
| R.500.1000.15 | [ 27501 ; ∞ ] | | [ 29304 ; 61143 ] | | + | − |
| R.500.1000.30 | [ 82621 ; ∞ ] | | [ 83568 ; 102336 ] | | + | − |
| R.500.1000.60 | [ 178212 ; 178212 ] | ⋆ | [ 178212 ; 178212 ] | ⋆ | = | = |
| R.600.100.1 | [ 1 ; ∞ ] | | [ 0 ; 54 ] | | − | − |
| R.600.100.15 | [ 2993 ; ∞ ] | | [ 3071 ; 7723 ] | | + | − |
| R.600.100.30 | [ 10285 ; ∞ ] | | [ 10400 ; 13029 ] | | + | − |
| R.600.100.60 | [ 23293 ; 23293 ] | ⋆ | [ 23293 ; 23293 ] | ⋆ | = | = |
| R.600.1000.1 | [ 1337 ; ∞ ] | | [ 1336 ; 2012 ] | | − | − |
| R.600.1000.15 | [ 31488 ; ∞ ] | | [ 32411 ; 72568 ] | | + | − |
| R.600.1000.30 | [ 105322 ; ∞ ] | | [ 105986 ; 129158 ] | | + | − |
| R.600.1000.60 | [ 214608 ; 214608 ] | ⋆ | [ 214608 ; 214608 ] | ⋆ | = | = |
| R.700.100.1 | [ 1 ; ∞ ] | | [ 0 ; 52 ] | | − | − |
| R.700.100.15 | [ 3789 ; ∞ ] | | [ 3872 ; 9431 ] | | + | − |
| R.700.100.30 | [ 12057 ; ∞ ] | | [ 12237 ; 15880 ] | | + | − |
| R.700.100.60 | [ 24069 ; 24102 ] | | [ 24102 ; 24102 ] | ⋆ | + | = |
| R.700.1000.1 | [ 1229 ; ∞ ] | | [ 0 ; 2021 ] | | − | − |
| R.700.1000.15 | [ 34056 ; ∞ ] | | [ 35270 ; 83966 ] | | + | − |
| R.700.1000.30 | [ 108184 ; ∞ ] | | [ 107041 ; 144585 ] | | − | − |
| R.700.1000.60 | [ 245589 ; 245589 ] | ⋆ | [ 245589 ; 245589 ] | ⋆ | = | = |

## SIE vs CPLEX (3)

- SIE in general improves the lower bounds

## SIE vs CPLEX (3)

- SIE in general improves the lower bounds
- SIE always provides heuristic solutions (upper bounds)

# SIE vs CPLEX (3)

- SIE in general improves the lower bounds
- SIE always provides heuristic solutions (upper bounds)
- SIE closes 16 instances (vs 13 oc CPLEX) over 48

# SIE vs Metaheuristics (1)

| Instance | Best Known | HAS-SOP | SIE | SIE vs Best Known | SIE vs HAS-SOP |
|---|---|---|---|---|---|
| R.200.100.1 | 63 | 88 | 61 | − | − |
| R.200.100.15 | 1792 | 2002 | 2200 | + | + |
| R.200.100.30 | 4216 | 4247 | 4216 | = | − |
| R.200.100.60 | 71749 | 71749 | 71749 | = | = |
| R.200.1000.1 | 1411 | 1532 | 1404 | − | − |
| R.200.1000.15 | 20481 | 21775 | 22636 | + | + |
| R.200.1000.30 | 41196 | 41278 | 41201 | + | − |
| R.200.1000.60 | 71756 | 71756 | 71556 | − | − |
| R.300.100.1 | 31 | 74 | 26 | − | − |
| R.300.100.15 | 3162 | 3520 | 3817 | + | + |
| R.300.100.30 | 6120 | 6151 | 6214 | + | + |
| R.300.100.60 | 9726 | 9726 | 9726 | = | = |
| R.300.1000.1 | 1331 | 1536 | 1605 | + | + |
| R.300.1000.15 | 29248 | 33533 | 36796 | + | + |
| R.300.1000.30 | 54147 | 54367 | 54480 | + | + |
| R.300.1000.60 | 109471 | 109471 | 109471 | = | = |
| R.400.100.1 | 21 | 59 | 13 | − | − |
| R.400.100.15 | 3925 | 4838 | 5132 | + | + |
| R.400.100.30 | 8165 | 8289 | 8408 | + | + |
| R.400.100.60 | 15228 | 15228 | 15228 | = | = |
| R.400.1000.1 | 1456 | 1783 | 1932 | + | + |
| R.400.1000.15 | 39612 | 45055 | 46653 | + | + |
| R.400.1000.30 | 85192 | 85579 | 86854 | + | + |
| R.400.1000.60 | 140816 | 140862 | 140816 | = | − |

# SIE vs Metaheuristics (2)

| Instance | Best Known | HAS-SOP | SIE | SIE vs Best Known | SIE vs HAS-SOP |
|---|---|---|---|---|---|
| R.500.100.1 | 11 | 51 | 60 | + | + |
| R.500.100.15 | 5431 | 6584 | 6959 | + | + |
| R.500.100.30 | 9665 | 10047 | 10244 | + | + |
| R.500.100.60 | 18240 | 18246 | 18240 | = | − |
| R.500.1000.1 | 1501 | 1840 | 2016 | + | + |
| R.500.1000.15 | 51091 | 60175 | 61143 | + | + |
| R.500.1000.30 | 99018 | 100453 | 102336 | + | + |
| R.500.1000.60 | 178212 | 178323 | 178212 | = | − |
| R.600.100.1 | 6 | 44 | 54 | + | + |
| R.600.100.15 | 5798 | 7610 | 7723 | + | + |
| R.600.100.30 | 12465 | 12810 | 13029 | + | + |
| R.600.100.60 | 23299 | 23342 | 23293 | − | − |
| R.600.1000.1 | 1534 | 1936 | 2012 | + | + |
| R.600.1000.15 | 57812 | 70454 | 72568 | + | + |
| R.600.1000.30 | 126789 | 130244 | 129158 | + | − |
| R.600.1000.60 | 214608 | 214724 | 214608 | = | − |
| R.700.100.1 | 5 | 41 | 52 | + | + |
| R.700.100.15 | 7380 | 9573 | 9431 | + | − |
| R.700.100.30 | 14513 | 15733 | 15880 | + | + |
| R.700.100.60 | 24102 | 24151 | 24102 | = | − |
| R.700.1000.1 | 1579 | 1912 | 2021 | + | + |
| R.700.1000.15 | 67510 | 81439 | 83966 | + | + |
| R.700.1000.30 | 134474 | 139769 | 144585 | + | + |
| R.700.1000.60 | 245589 | 246128 | 245589 | = | − |

# SIE vs Metaheuristics (3)

- SIE does not consistently outperform HAS-SOP but...

## SIE vs Metaheuristics (3)

- SIE does not consistently outperform HAS-SOP but...
- SIE improved 16 HAS-SOP solutions

## SIE vs Metaheuristics (3)

- SIE does not consistently outperform HAS-SOP but...
- SIE improved 16 HAS-SOP solutions
- SIE different behavior w.r.t. HAS-SOP

## SIE vs Metaheuristics (3)

- SIE does not consistently outperform HAS-SOP but...
- SIE improved 16 HAS-SOP solutions
- SIE different behavior w.r.t. HAS-SOP
- SIE improved 6 best known heuristic solutions

## Conclusions

- Heuristic and exact algorithms are complementary

## Conclusions

- Heuristic and exact algorithms are complementary
- A network flow formulation for the SOP has been considered

## Conclusions

- Heuristic and exact algorithms are complementary
- A network flow formulation for the SOP has been considered
- HAS-SOP, an effective metaheuristic algorithm for SOP, has been considered

## Conclusions

- Heuristic and exact algorithms are complementary
- A network flow formulation for the SOP has been considered
- HAS-SOP, an effective metaheuristic algorithm for SOP, has been considered
- The Shared Incumbent Environment (combining CPLEX and HAS-SOP) outperforms CPLEX alone on SOP instances

## Conclusions

- Heuristic and exact algorithms are complementary
- A network flow formulation for the SOP has been considered
- HAS-SOP, an effective metaheuristic algorithm for SOP, has been considered
- The Shared Incumbent Environment (combining CPLEX and HAS-SOP) outperforms CPLEX alone on SOP instances
- The Shared Incumbent Environment sometimes improves the integer solutions provided by other metaheuristics

## Conclusions

- Heuristic and exact algorithms are complementary
- A network flow formulation for the SOP has been considered
- HAS-SOP, an effective metaheuristic algorithm for SOP, has been considered
- The Shared Incumbent Environment (combining CPLEX and HAS-SOP) outperforms CPLEX alone on SOP instances
- The Shared Incumbent Environment sometimes improves the integer solutions provided by other metaheuristics
- Communication CPLEX $\rightarrow$ HAS-SOP to be redesigned?

## Conclusions

- Heuristic and exact algorithms are complementary
- A network flow formulation for the SOP has been considered
- HAS-SOP, an effective metaheuristic algorithm for SOP, has been considered
- The Shared Incumbent Environment (combining CPLEX and HAS-SOP) outperforms CPLEX alone on SOP instances
- The Shared Incumbent Environment sometimes improves the integer solutions provided by other metaheuristics
- Communication CPLEX $\rightarrow$ HAS-SOP to be redesigned?
- General results? Future: apply similar frameworks to other optimization problems

# Outline

- Introduction
- Problem description
- A Genetic Algorithm
- A Hybrid Ant System
- A Heuristic Manipulation Technique
- A Particle Swarm Optimization
- An Enhanced Ant Colony System
- A Shared Incumbent Environment
- Bibliography

# References 1/3


- Pulleyblank W.R., Timlin M.T. (1991). Precedence constrained routing and helicopter scheduling: heuristic design. Technical Report RC17154 (#76032), IBM T.J. Watson Research Center, USA.
- Timlin M.T., Pulleyblank W.R.,(1992). Precedence constrained routing and helicopter scheduling: heuristic design. Interfaces 22(3): 100-111.
- Ascheuer N., Escudero L.F., Grötschel M., Stoer M. (1993). A cutting plane approach to the sequential ordering problem (with applications to job scheduling in manufacturing). SIAM Journal on Optimization, 3:25-42.
- Escudero L.F., Guignard M., Malik K. (1994). A Lagrangean relax-and-cut approach for the sequential ordering problem with precedence relationships. Annals of Operations Research, 50: 1219-237.
- Balas E., Fischetti M., Pulleyblank W.R. (1995). The precedence-constrained asymmetric traveling salesman polytope. Mathematical Programming, 65: 241-265.
- Ascheuer N. (1995). Hamiltonian path problems in the on-line optimization of flexible manufacturing systems. PhD thesis, Technische Universität Berlin, Germany.
- Chen S., Smith S. (1996). Commonality and genetic algorithms. Technical Report CMU-RI-TR-96-27, The Robotic Institute, Carnegie Mellon University, USA.

# References 2/3


- Gambardella L.M., Dorigo M. (1997). HAS-SOP: an Hybrid Ant System for the sequential ordering problem. Technical Report IDSIA-11-97, IDSIA, Lugano, Switzerland.
- Gambardella L.M., Dorigo M. (2000). An ant colony system hybridized with a new local search for the sequential ordering problem. INFORMS Journal on Computing, 12(3): 237-255.
- Guerriero F., Mancini M. (2003). A cooperative parallel rollout algorithm for the sequential ordering problem. Parallel Computing, 29(5): 663-677.
- Seo D.I., Moon B.R. (2003). A hybrid genetic algorithm based on complete graph representation for the sequential ordering problem. Proceedings of GECCO 2003, pages 669-680.
- Hernàdvölgyi I.T. (2003). Solving the sequential ordering problem with automatically generated lower bounds. In Proceedings of Operations Research 2003, pages 355-362.
- Hernàdvölgyi I.T. (2004). Automatically Generated Lower Bounds for Search. PhD thesis, University of Ottawa, Canada.
- Montemanni R., Smith D.H., Gambardella L.M. (2007). Ant Colony Systems for large Sequential Ordering Problems. In Proceedings of IEEE SIS 2007.
- Montemanni R., Rizzoli A.E., Smith D.H., Gambardella L.M. (2008). Sequential ordering problems for crane scheduling in port terminals. Proceedings of HMS 2008, pages 180-189.


# References 3/3


- Montemanni R., Smith D.H., Gambardella L.M. (2008). A Heuristic Manipulation Technique for the Sequential Ordering Problem. Computers and Operations Research, 35(12): 3931-3944.
- Montemanni R., Smith D.H., Rizzoli A.E., Gambardella L.M. (2009). Sequential Ordering Problems for Crane Scheduling in Port Terminals. International Journal of Simulation and Process Modelling 5(4), 348-361.
- Anghinolfi D., Montemanni R., Paolucci M., Gambardella L.M. (2009). A Particle Swarm Optimization approach for the Sequential Ordering Problem. Proceedings of MIC 2009.
- Gambardella L.M., R. Montemanni (2010). An Enhanced Ant Colony System for two Transportation Problems. Proceedings of Tristan VII, pages 292-295.
- Anghinolfi D., Montemanni R., Paolucci M., Gambardella L.M. (2011). A Hybrid Particle Swarm Optimization approach for the Sequential Ordering Problem. Computers and Operations Research 38(7), 1076-1085.
- Gambardella L.M., R. Montemanni, D.A. Weyland (2011). An Enhanced Ant Colony System for the Sequential Ordering Problem. Proceedings of OR 2011.
- Mojana M., Montemanni R., Di Caro G., Gambardella L.M. An Algorithm combining Linear Programming and an Ant Colony System for the Sequential Ordering Problem. Proceedings of ATAI 2011.
- Gambardella L.M., R. Montemanni, D.A. Weyland (to appear). Coupling Ant Colony Systems with Strong Local Searches. European Journal of Operational Research.

# Exercises

1. In the context of the Genetic Algorithm approach described at P10, consider the following two parents:
   1. **H E U R A L G 1 0**
   2. **H E A U L G 0 1 R**

   Show the result of the application of the following crossover operators:
   a. **Partially Mapped Crossover (PMX)**
   b. **Order Crossover (OX)**
   c. **Maximal Sub-Tour (MST)**
   d. **Maximal Partial Order (MPO)**

   In case of missing information, make random choices.

2. In the context of the Particle Swarm Optimization approach described at P50, consider the following individual:

   **1 2 3 4 5**

   show the solution obtained by applying the following velocity vectors to it (after the application of the sequence completion procedure described at P55):
   a. **v1 = {(1, 4), (4,-2), (5,-1)}**
   b. **v2 = {(2, 1), (3,-1), (4,-2), (5,-3)}**
   c. **v3 = {(4,-2), (1,2), (2,3)}**