A Constructive Hybrid Algorithm for Crew Pairing Optimization

Broderick Crawford
Pontificia Universidad Católica de Valparaíso
Chile

Dr. Carlos Castro
Universidad Técnica Federico Santa María
Chile

Dr. Eric Monfroy
Universidad de Nantes
Francia

Overview

- Introduction
- Set Partitioning Problem (SPP)
- Ant Colony Optimisation (ACO)
- ACO for Set Covering and Set Partitioning Problems
- ACO with Forward Checking and Full Lookahead Procedures
- Experiments and Results
- Conclusions and future work

Introduction

- Set Partitioning Problem (SPP) is a type of problem that can model different real life situations: for example **Crew Pairing**

- There exist problems for which the pure **ACO Constructive Metaheuristic** is of limited effectiveness: very strongly constrained problems like SPP

- We explore the addition of a lookahead mechanism to the ACO algorithm for solving SPP. Applying these **Hybrid Algorithms** for solving standard benchmarks from Beasley`s ORLIB (NorthWest Airline instances)

Set Partitioning Problem

- SPP is the problem of partitioning a given set into mutually independent subsets while minimizing a cost function defined as the sum of the costs associated to each of the eligible subsets

- Its importance derives from the fact that many real life situations can be modeled as SPP, and in fact many combinatorial optimization problems (such as, crew scheduling, vehicle routing, project scheduling, and warehouse location problems, to name a few) can be modeled as SPP with maybe some additional constraints

- In SPP we are given a m × n matrix A = (a_{ij}) in which all the matrix elements are either zero or one. Additionally, each column is given a non negative cost c_j

- We say that a column j covers a row i if $a_{ij} = 1$

- Let x_j denotes a binary variable which is one if column j is chosen and zero otherwise

Set Partitioning Problem Example

	x_1	x_2	x_3	x_4
1	1	1	0	1
2	0	1	0	0
3	0	0	1	1

Where

$C_1 = 10$

$C_2 = 20$

$C_3 = 30$

$C_4 = 40$

Minimize

$$10x_1 + 20x_2 + 30x_3 + 40x_4$$

Subject to

$$x_1 + x_2 + x_4 = 1$$

$$x_2 = 1$$

$$x_3 + x_4 = 1$$

Set Partitioning Problem Formulation

$$Min(z) = \sum_{j=1}^{n} c_j x_j$$

Subject to :

$$\forall i \rightarrow \sum_{j=1}^{n} a_{ij} x_j = 1$$

$$\forall j \rightarrow x_j \in \{0,1\}$$

A pairing is a sequence of flights to be covered by a single crew over a 2 to 3 day period

Pairing j	Flight Sequence	Cost $
1	101-203-406-308	2900
2	101-203-407	2700
3	101-204-305-407	2600
4	101-204-308	3000
5	203-406-310	2600
6	203-407-109	3150
7	204-305-407-109	2550
8	204-308-109	2500
9	305-407-109-212	2600
10	209-109-212	2050
11	402-204-305	2400
12	402-204-310-211	3600
13	406-308-109-211	2550
14	406-310-211	2650
15	407-109-211	2350

Defining the decision variable x_j equal to 1 if paring j is chosen and 0 otherwise. Each flight must be covered.

Minimize $2900x_1 + 2700x_2 + 2600x_3 + 3000x_4 + 2600x_5 + 3150x_6 + 2550x_7 + 2500x_8 + 2600x_9 + 2050x_{10} + 2400x_{11} + 3600x_{12} + 2550x_{13} + 2650x_{14} + 2350x_{15}$

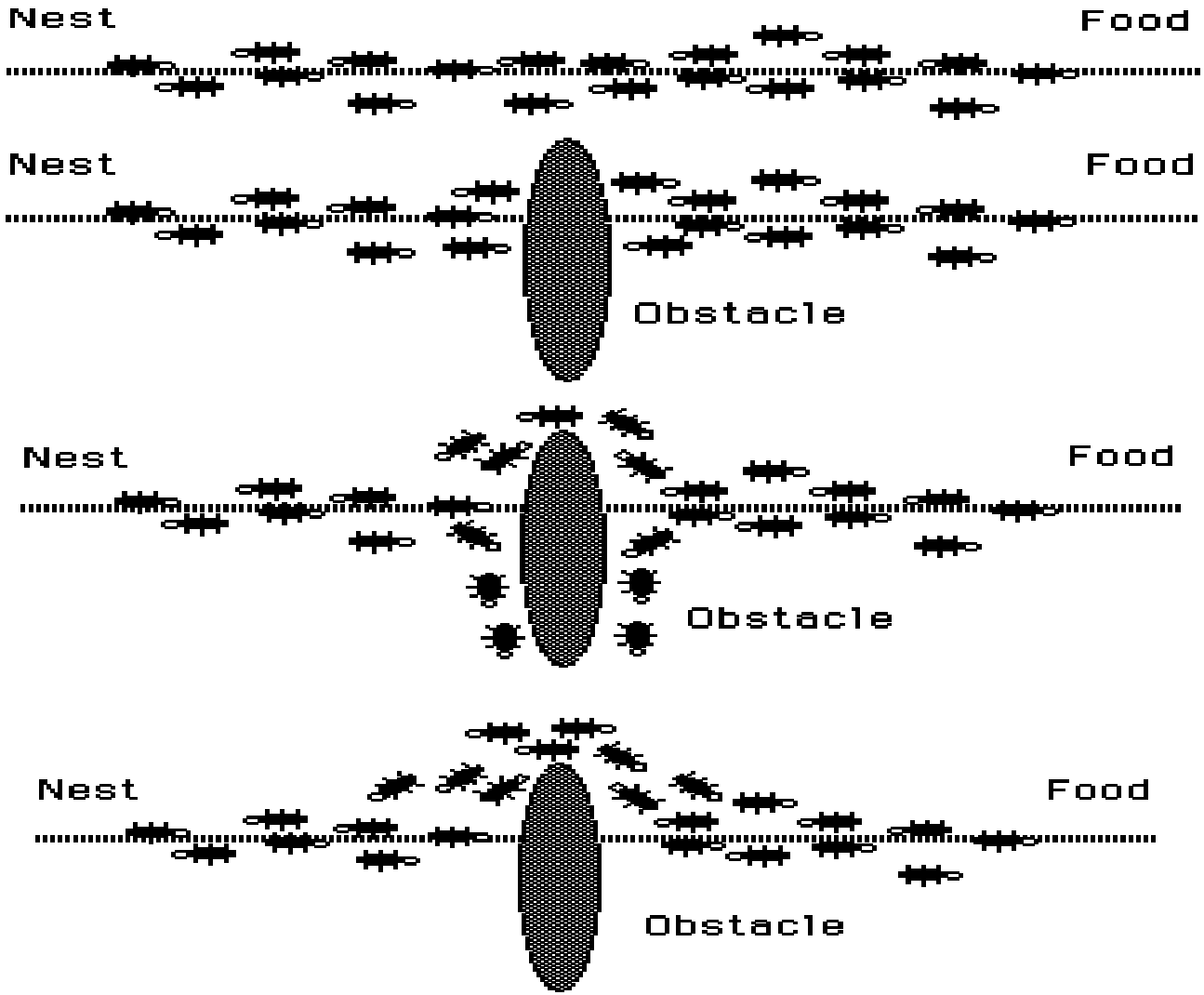
Subject to

$$x_1 + x_2 + x_3 + x_4 = 1 \text{ (flight 101)}$$
$$x_6 + x_7 + x_8 + x_9 + x_{10} + x_{13} + x_{15} = 1 \text{ (flight 109)}$$
$$x_1 + x_2 + x_5 + x_6 = 1 \text{ (flight 203)}$$
$$x_3 + x_4 + x_7 + x_8 + x_{11} + x_{12} = 1 \text{ (flight 204)}$$
$$x_{12} + x_{13} + x_{14} + x_{15} = 1 \text{ (flight 211)}$$
$$x_9 + x_{10} = 1 \text{ (flight 212)}$$
$$x_3 + x_7 + x_9 + x_{11} = 1 \text{ (flight 305)}$$
$$x_1 + x_4 + x_8 + x_{10} + x_{13} = 1 \text{ (flight 308)}$$
$$x_5 + x_{12} + x_{14} = 1 \text{ (flight 310)}$$
$$x_{11} + x_{12} = 1 \text{ (flight 402)}$$
$$x_1 + x_5 + x_{13} + x_{14} = 1 \text{ (flight 406)}$$
$$x_2 + x_3 + x_6 + x_7 + x_9 + x_{15} = 1 \text{ (flight 407)}$$
$$x_j = 0 \text{ or } 1; \forall j = 1,...,15$$

Ant Colony Optimisation (ACO)

- A metaheuristic search process based on the foraging behaviour of ant colonies
  - A number of agents, called (artificial) ants, build up solutions to the problem at hand
  - They deposit pheromone on each element they choose/visit
  - At each step they choose from the available elements based on their associated *probabilities*
  - Element probability is a function of heuristic information (eg, estimated cost) and pheromone information (from other ants)
  - Pheromone biases search towards solution characteristics that have proved valuable in the past

Foraging behaviour of Ant Colonies



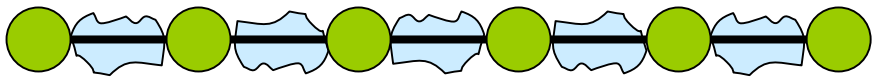
Ant Colony Optimisation (ACO)

ACO *learns* based on solutions encountered in the past

Learning occurs through pheromone, so choosing an appropriate pheromone representation is crucial to the success of an ACO algorithm

Application of ACO

- Ant System (AS, the first ACO algorithm) associate pheromone and heuristic information with *links*

A solution: 

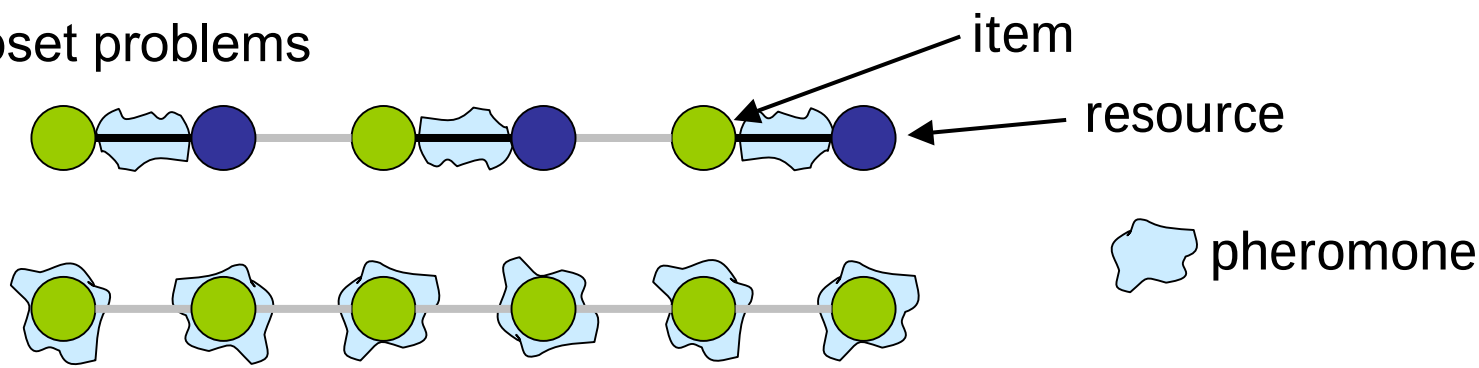
 pheromone

- High degree of similarity between real ants on the ground and artificial ants on a graph

- There are improved ways to apply ACO, like ACS (Ant Colony System)

Application of ACO

- Initial ACO applications on problems that could be modelled as graphs
  - pheromone associated with *edges* of these graphs
- ACO increasingly applied to problems that don't have a natural graph representation
  - Assignment problems
  - Subset problems



ACO Techniques

	AS	ACS
Pheromone Update	At the end of the solution, each ant perform an update of each column choosen by this ant.	Two phases: - Local Update: The pheromone of the choosen column is modified. - Global Update: The pheromone level of the columns founded in the best solution is modified.
Transition Rule	It is based in the probability of each column.	Has two points of decision: - Use of a list of candidates, which contains the best columns to be choosen according to some creterion. - Use of a parameter q, which indicates if it is used the exploitation of the knowledge or if it is used the exploration of the neighborhood.

ACO for SPP

- ACO can be applied in a very straightforward way to the SPP. The columns are chosen as the solution components and have associated a cost and a pheromone trail.
- Constraints say that each column can be visited by an ant once and only once and that a final solution has to cover all rows.
- A walk of an ant over the graph representation corresponds to the iterative addition of columns to the partial solution obtained so far.
- Each ant starts with an empty solution and adds columns until all rows are covered. A column to be added is chosen with a probability that depends of pheromone trail and the heuristic information.

Construction Phase

Each ant k chooses the next column j using the probability given by

$$p_j^k = \frac{[\tau_j]*[n_j]^\beta}{\sum_{l \in N^k} [\tau_l]*[n_l]^\beta} \qquad si \;\; j \in N^k$$

And updates the chosen columns, according to the Evaporation Rule

$$\tau_j = (1-\rho)\tau_j + \rho\tau_0$$

Pheromone Update (τ)

- Update when ant chooses element j

$$\tau_j = (1-\rho)\tau_j + \rho\tau_0$$

- Update at end of iteration for each column in the best solution

$$\tau_j = (1-\rho)\tau_j + \rho\Delta\tau_j$$

where $\Delta\tau_j$ is the frequency of the column j in the solutions of the ants

ACO Procedure for SPP

```
1   Procedure ACO_for_SPP
2   Begin
3     initParameters();
4     While (remain iterations) do
5       For k := 1 to h := nants do
6         While (solution have no completed) do
7           Choose next Column j with Transition Rule Probability
8           addToTabuList(j);
9           For each Row i covered by j do
10            AddColumnToTabuList(j);
11          EndFor
12        EndWhile
13      EndFor
14      updateOptimum();
15      updatePheromone();
16    EndWhile
17    return best_solution_founded
18  End.
```

Heuristic Information (η)

- To use the actual knowledge (as a transition rule), it is used a dynamic heuristic information that depends on the partial solution of an ant. It is defined as

$$\eta_j = \frac{e_j}{c_{c_j\, j}}$$

- Where e_j is the number of additional rows covered adding a column j and c_j is the cost of column j.

Pure ACO with limited effectiveness

	x_1	x_2	x_3	x_4
1	1	1	0	1
2	0	1	0	0
3	0	0	1	1

Constraints

$$x_1 + x_2 + x_4 = 1$$

$$x_2 = 1$$

$$x_3 + x_4 = 1$$

Pure ACO with limited effectiveness

- 1st try.

  $x_1 = 1 \rightarrow x_2 = 0$

  $x_4 = 0$

  → Doesn't respect Constraint 2

  → Solution cannot be completed

- 2nd try.

  $x_2 = 1 \rightarrow$ x1 and $x_4 = 0$ (applying constraint 1)

  $x_3 = 1$ (applying constraint 3)

  Solution Found : $x_1 = 0$

  $x_2 = 1$

  $x_3 = 1$

  $x_4 = 0$

Constraint Programming (CP)

- CP is used in the variable selection process. It means that now, the column to be added in the solution, will be chosen only if doesn't cause a conflict with the next column to select.

- CP in ACO: **Forward Checking**

ACO with FC

- Some efforts have been done in order to integrate CP techniques to ACO algorithms.
- Forward Checking (FC) seems to be the easiest way to prevent future conflicts.
- Adding Forward Checking to ACO means that columns are chosen if they do not produce any conflict with respect to the next column to be chosen.
- Forward checking checks only the constraints between the current variable and the future variables: Arc consistency to the not yet instantiated variables.

Constraint Programming (CP)

```
1   Procedure ACO+CP_for_SPP
2   Begin
3     initParameters();
4     While (remain iterations) do
5       For k := 1 to h := nants do
6         While (solution have no completed) do
7           Choose next Column j with Transition Rule Probability
8           For each Row i covered by j do          /*Constraint with j*/
9            feasible(i):=Posting(j);               /*Constraint Propagation*/
10          EndFor
11          If feasible(i) for all i then
12            AddColumnToSolution(j)
13          Else
14           Backtracking(j)                        /*Set j uninstantiated*/
15        AddColumnToTabuList(j);
16        EndWhile
17      EndFor
18      updateOptimum();
19      updatePheromone();
20    EndWhile
21    return best_solution_founded
22  End.
```

Experiments and Results

- Adding FC and FLA techniques to the basic ACO algorithms for solving standard benchmarks taken from the Beasley`s ORLIB.

- Considering several tests and published experimental results we use the following parameters for the algorithms: $\rho = 0.4$, number of iterations = 160, number of ants = 120, $\beta = 0.5$, for ACS Qo = 0.5

- Algorithms were implemented using ANSI C, GCC 3.3.6, under Microsoft Windows XP Professional version 2002.

Experiments and Results

Table 2. ACO with Forward Checking

Problem	Rows	Columns	Optimum	Density	Beasley	Levine	Kotecha	AS	ACS	AS+FC	ACS+FC
sppnw06	50	6774	7810	18.17	7810	-	-	9200	9788	8160	8038
sppnw08	24	434	35894	22.39	35894	37078	36068	X	X	35894	36682
sppnw09	40	3103	67760	16.20	67760	-	-	70462	X	70222	69332
sppnw10	24	853	68271	21.18	68271	X	68271	X	X	X	X
sppnw12	27	626	14118	20.00	14118	15110	14474	15406	16060	14466	14252
sppnw15	31	467	67743	19.55	67743	-	-	67755	67746	67743	67743
sppnw19	40	2879	10898	21.88	10898	11060	11944	11678	12350	11060	11858
sppnw23	19	711	12534	24.80	12534	12534	12534	14304	14604	13932	12880
sppnw26	23	771	6796	23.77	6796	6796	6804	6976	6956	6880	6880
sppnw32	19	294	14877	24.29	14877	14877	14877	14877	14886	14877	14877
sppnw34	20	899	10488	28.06	10488	10488	10488	13341	11289	10713	10797
sppnw39	25	677	10080	26.55	10080	10080	10080	11670	10758	11322	10545
sppnw41	17	197	11307	22.10	11307	11307	11307	11307	11307	11307	11307

Experiments and Results

- The effectiveness of hybridization is showed to some instances of SPP solving with AS+FC

- Strongly constrained problem characteristic of SPP does the stochastic behavior of ACO improved with FC techniques in the construction phase, so that almost only feasible solutions are induced

- In the original ACO implementation the SPP solving derives in a lot of unfeasible labeling of variables, and many ants can not complete solutions

Conclusions

- Performance of ACO is possible to improve with some classes of hybridization
- Using only the transition rule of pure ACO doesn't allow, in some problems, to complete feasible solutions
- In a restricted problem like SPP, ACO with Lookahead techniques showed an interesting performance, we demonstrated that this integration improves the process, mainly with respect to success costs instead running times. But the trade off in all cases is very convenient

Future Work

- To study the pheromone representation and will try to incorporate it into Lookahead techniques

- Considering that the ant´s solutions may contain redundant components which can be eliminated by a fine tuning after the solution, then we will explore Post Processing procedures, which consist in the identification and replacement of the columns of the ACO solution in each iteration by more effective columns

- To study available local search techniques in order to reduce the input problem and improve the solutions given by the ants