HC-ACO: The Hyper-Cube Framework for Ant Colony Optimization

Christian Blum * Andrea Roli * Marco Dorigo *

* IRIDIA - Université Libre de Bruxelles
Av. Roosvelt 50 - Bruxelles (Belgium)
Email: {cblum, aroli, mdorigo}@ulb.ac.be

1 Introduction

Ant Colony Optimization (ACO) [2] is a recently proposed metaheuristic approach for solving hard combinatorial optimization problems. The inspiring source of ACO is the foraging behavior of real ants. In most ACO implementations the hyperspace for the pheromone values used by the ants to build solutions is only implicitly limited. In this paper we propose a new way of implementing ACO algorithms, which explicitly defines the hyperspace for the pheromone values as the convex hull of the set of 0-1 coded feasible solutions of the combinatorial optimization problem under consideration. We call this new implementation the hyper-cube framework for ACO algorithms. The organization of this extended abstract is as follows. In section 2 we briefly present the original Ant System [3] for static combinatorial optimization problems. In section 3 we propose the hyper-cube framework for ACO algorithms and we present pheromone updating rules for Ant System (AS) and $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS). In section 4 we discuss some of the advantages involved with the hyper-cube framework for ACO algorithms, while Section 5 outlines future work.

2 ACO algorithms

Let $\mathcal{O} = \{o_1, ..., o_n\}$ be a finite set of objects. We consider the problem of choosing an optimal subset (ordered or unordered; according to the constraints of the problem) of cardinality $0 \leq l \leq n$ from the finite set $\mathcal{O}$. This cardinality can be fixed or variable. All subsets s fulfilling the constraints of the problem are called feasible solutions of the problem. The set of all feasible solutions is denoted by $\mathcal{S}$. The quality of a solution s is evaluated by an objective function $f : \mathcal{S} \to I\!R$ and is denoted by $f(s)$. We will deal with minimization problems. Therefore the set of optimal solutions $\hat{\mathcal{S}}$ is a subset of $\mathcal{S}$ with $f(s^i) < f(s^j) \quad \forall s^i \in \hat{\mathcal{S}}, s^j \in \mathcal{S} \setminus \hat{\mathcal{S}}$.

In ACO algorithms artificial ants construct a solution by building a path on a *construction graph* $\mathcal{G} = (\mathcal{C}, \mathcal{L})$ where each element from the set $\mathcal{C}$ of solution components corresponds to an object in the set $\mathcal{O}$ and the elements of $\mathcal{L}$ (called *connections*) fully connect the elements of $\mathcal{C}$. To each solution component o_i we have associated a *desirability value* η_i (which is called *heuristic information*) and a *pheromone value* τ_i. In each iteration of the algorithm, each ant (from a set of k ants) generates a feasible solution according to probabilities $p(s[o_l], o_r)$ where $s[o_l]$ is a partial solution with element o_l as last added element. In a particular implementation of ACO algorithms, known as Ant System (AS), these probabilities are determined by the following *state transition rule*:

$$p(s[o_l], o_r) \quad = \quad \begin{cases} \frac{[\eta_r]^\alpha [\tau_r]^\beta}{\sum_{o_u \in J(s[o_l])} [\eta_u]^\alpha [\tau_u]^\beta} & \text{if} \quad o_r \in J(s[o_l]) \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

In this formula α and β are parameters to adjust the relative importance of heuristic information and pheromone values and $J(s[o_l])$ denotes the set of solution components which are allowed to be added to the partial solution $s[o_l]$. Once all ants have constructed a solution the *online delayed pheromone update rule* is applied. This pheromone update rule for ant A_i $(i = 1, ..., k)$ with generated solution s^i $(i = 1, ..., k)$ consisting of solution components $o_l \in \mathcal{O}$ is as follows:

$$\tau_j \leftarrow \rho \cdot \tau_j + \sum_{i=1}^{k} \Delta\tau_j^i \quad \text{where} \quad \Delta\tau_j^i \quad = \quad \begin{cases} \frac{1}{f(s^i)} & \text{if} \quad o_j \in s^i \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $f(s^i)$ is the quality of solution s^i and $0 < \rho < 1$ is a pheromone decay parameter. This pheromone update rule leads to an increase of pheromone on solution components which have been found in better quality solutions than other solution components (where the pheromone value will decrease). Although AS is important, because it was the first ACO algorithm proposed, in the last few years some changes and extensions of AS have been proposed, e.g. $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [6]. In general, ACO algorithms are proven to be a very effective – for some problems like the QAP even the state-of-the-art – metaheuristic method for combinatorial optimization problem solving.

3 HC-ACO

In the AS framework described in the previous section, the pheromone values associated to the solution components are used to probabilistically construct solutions to the optimization problem under consideration. If we regard the set of pheromone values as a vector $\vec{\tau} = (\tau_1, ..., \tau_n)$, this vector is moving in a hyperspace with different limits for different pheromone updating rules. We will denote this hyperspace in the following with $\mathcal{T}$. For AS, the pheromone values τ_i are limited by

$$\lim_{t \to \infty} \tau_i(t) \leq \frac{1}{1-\rho} \cdot \frac{k}{f(s^{opt})} \tag{3}$$

where $s^{opt} \in \hat{\mathcal{S}}$. It is clear that the limits of $\mathcal{T}$ can be very different depending on the pheromone updating rule itself (e.g., AS versus AS using elitist strategies [3]) and the amount of pheromone added in each step, which is a function of the solution quality. The hyper-cube framework described in the following will give a well-defined description of the hyperspace $\mathcal{T}$.

3.1 The hyper-cube framework

For most combinatorial optimization problems a mathematical programming formulation exists where solutions are modeled as binary vectors (0-1 Integer Programming). In this formulation we have a set of decision variables, corresponding to the solution components, which can assume values $\{0, 1\}$. A 1 is indicating that the corresponding solution component is a member of the solution defined by the vector. In this kind of problem modeling the set of feasible solutions $\mathcal{S}$ is a subset of the set of corners of the n-dimensional hyper-cube (in case of n solution components). If we relax the $\{0, 1\}$-constraints, the extended set of feasible solutions $\tilde{\mathcal{S}}$ is the set of all vectors $\vec{v} \in \mathbb{R}^n$ which are expressible as convex combinations of binary vectors $\vec{v_i} \in \mathcal{S}$ (see figure 1a for an example):

$$\vec{v} \in \tilde{S} \Leftrightarrow \vec{v} = \sum_{\vec{v_i} \in S} \alpha_i \vec{v_i}, \quad \alpha_i \in [0, 1] \tag{4}$$

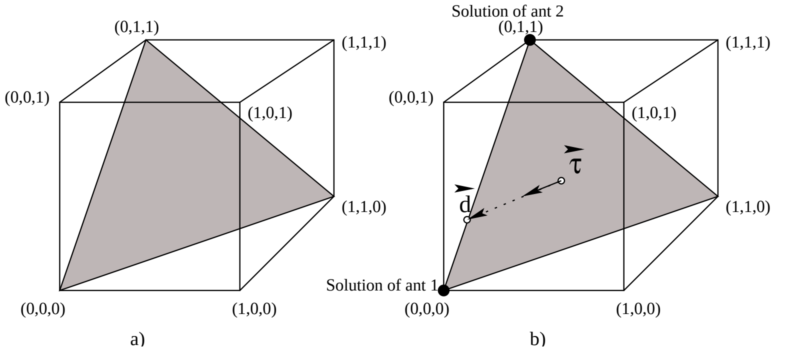

Figure 1: The set $\mathcal{S}$ of feasible solutions consists of the three vectors $(0,0,0)$, $(1,1,0)$ and $(0,1,1)$. The gray shaded area is the set $\tilde{\mathcal{S}}$. In b), two solutions have been created by two ants. $\vec{d}$ is the weighted average of these two solutions ($(0,0,0)$ is of higher quality) and $\vec{\tau}$ will be shifted towards $\vec{d}$

As in the 0-1 Integer Programming (IP) formulation for combinatorial optimization problems, in the hyper-cube framework for ACO algorithms we regard the solutions $s = (s_1, ..., s_n)$ for our problem as binary vectors of length n. Again each position is indicating the presence or absence of the corresponding solution component in the solution described by this vector. Additionally the vector of pheromone values $\vec{\tau} = (\tau_1, ..., \tau_n)$ will be moving in $\tilde{\mathcal{S}}$ and can therefore be seen as a probability distribution over the solution components (in $\vec{\tau}$, τ_i is associated to solution component o_i, $1 \leq i \leq n$).

3.2 Pheromone updating rules for the hyper-cube framework

In the following we rewrite the pheromone updating rules for Ant System (respectively $\mathcal{MM}$AS) for the hyper-cube framework and we will give a new interpretation. Ant System's new updating rule is obtained via a normalization of equation (2):

$$\tau_j \leftarrow \rho \cdot \tau_j + (1-\rho) \cdot \sum_{i=1}^{k} \Delta\tau_j^i \quad \text{where} \quad \Delta\tau_j^i \quad = \quad \begin{cases} \frac{\frac{1}{f(s^i)}}{\sum_{l=1}^{k} \frac{1}{f(s^l)}} & \text{if} \quad o_j \in s^i \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

In $\mathcal{MM}$AS in every iteration just one solution (called s^{upd} in the following) updates the pheromone values so that the left part of equation (5) becomes $\tau_j \leftarrow \rho \cdot \tau_j + (1-\rho) \cdot \Delta\tau_j^{upd}$. Solution s^{upd} can be the iteration best solution generated by ants or the global best solution generated by the ants since the start of the algorithm. In $\mathcal{MM}$AS the right part of equation (5) therefore is reduced to $\Delta\tau_j^{upd} = 1$ if $o_j \in s^{upd}$ and 0 otherwise. The updating rule given in equation (5) can be re-interpreted as a shift of the probability distribution within the hyperspace $\mathcal{T}$ towards the probability distribution given by the weighted average of solutions. The following theorem gives a formal description of that:

Theorem 1 *The pheromone updating rule in the hyper-cube framework given by equation (5) is equivalent to*

$$\vec{\tau} \quad \longleftarrow \quad \vec{\tau} + \mu \cdot (\vec{d} - \vec{\tau}) \tag{6}$$

$$\vec{d} = (d_1, ..., d_n) \quad \text{where} \quad d_j \quad = \quad \sum_{i=1}^{k} \frac{\frac{1}{f(s^i)} \cdot s_j^i}{\sum_{l=1}^{k} \frac{1}{f(s^l)}}, \quad j = 1, ..., n \tag{7}$$

and $s^i \in \mathcal{S}^{upd}$*, the set of solutions used for updating.* $0 < \mu < 1$ *is a parameter called learning rate.*

Vector $\vec{d}$ in Theorem 1 can be interpreted as a weighted average of the set of solutions used for updating. So, the pheromone updating rule for Ant System in the hyper-cube framework computes a probability distribution of the solution components present in the set of solutions generated by the ants. Then the old probability distribution $\vec{\tau}$ which was used by the ants to generate the solutions gets shifted by a factor $0 < \mu < 1$ in the direction of the new probability distribution $\vec{d}$. This fact – as we will see later – opens the door to comparisons with genetic algorithm techniques. An example for Ant System's updating rule is given in figure 1b. It is also important to mention that, if the starting values for the pheromone values describe a vector $\vec{\tau} \in \mathcal{T}$ (the convex hull of all feasible solutions), then by applying these pheromone updating rules $\vec{\tau}$ will always stay in $\mathcal{T}$.

4 Benefits of the hyper-cube framework

The most important aspect involved with the introduction of the hyper-cube framework for ACO algorithms is the fact that this framework favours a mathematical examination of ACO algorithms. This aspect is essential for detailed comparisons with Evolutionary Compuation algorithms. Also, the fact that $\vec{\tau} \in \mathcal{T}$ for the duration of the algorithm can be used for different techniques (i.e., diversification).

A diversification scheme for ACO algorithms: In the following we define the concepts of *global desirability* and *global frequency* for solution components o_j. We denote the set of all solutions generated by the ants since the start of the algorithm by $\mathcal{S}_{ants}$. The *global desirability* of solution components o_j ($j = 1, ..., n$) is given by equation (8a) and the *global frequency* by equation (8b):

$$\text{(a)} \quad v_j^{des} \longleftarrow \max\{\frac{1}{f(s)} : s \in \mathcal{S}_{ants}, s_j = 1\} \qquad \text{(b)} \quad v_j^{fr} \longleftarrow \sum_{s \in \mathcal{S}_{ants}} s_j \tag{8}$$

In the following we will use these vectors, $\vec{v^{des}}$ and $\vec{v^{fr}}$, to re-position the vector of pheromone values $\vec{\tau}$ in $\mathcal{T}$. We use b_1 ants to construct solutions according to the global desirability values and b_2 ants to construct solutions according to the global frequencies using the transition rule:

$$p(s[o_l], o_r) \quad = \quad \begin{cases} \frac{p_r}{\sum_{o_u \in J(s[o_l])} p_u} & \text{if} \quad o_r \in J(s[o_l]) \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where $p_i = \frac{1}{v_i^{fr}}$ (respectively $p_i = v_u^{des}$) for $i = 1, ..., n$, and b_1 and b_2 are parameters. Then we produce the new vector $\vec{\tau}$ of pheromone values using these $b_1 + b_2$ solutions in the same way as we compute vector $\vec{d}$ in the pheromone updating rule of Theorem 1. We compute it as a weighted average of these solutions. This mechanism gives us the possibility to restart ACO algorithms in an intelligently biased way whenever it reaches stagnation. The usage of the global desirability ensures a quite high pheromone value for solution elements which have been found in good quality solutions in the past ($\rightarrow$ exploitation), whereas the usage of global frequencies guides the search to areas in the search space which have not been explored so far ($\rightarrow$ exploration). Preliminary results for the QAP suggest the usefulness of the above diversification scheme.

Comparisons with Evolutionary Computation: The new hyper-cube framework simplifies the examination of similarities and differences between ACO algorithms and Evolutionary Computation. In the field of Evolutionary Computation in the last decade some interesting developments quite similar to ACO algorithms have been proposed. They have in common that they use a probabilistic mechanism for recombination of individuals. This leads to algorithms where the population statistics are kept in probability vectors instead of the population itself as done in usual Genetic Algorithms. In each iteration of the algorithm these probabilities are used to generate new solutions. The new solutions are then used to adapt the probability vector (or even to replace it by the vector of solution component probabilities given by the new set of solutions). The first approach of such a kind was given by the work of Syswerda [7], who replaced the usual *two parent recombination* (TPR) operator by an operator called Bit-Simulated Crossover (BSC). Another approach called Population-Based Incremental Learning (PBIL) has been proposed by Baluja et al. [1]. Harik et al. [4] extended the idea of PBIL and proposed an algorithm called Compact Genetic Algorithm (cGA). Mühlenbein and Voigt

[5] finally presented a new form of recombination called *gene pool recombination* (GPR). Later they generalized their idea of GPR in proposing a class of algorithms called Univariate Marginal Distribution Algorithms (UMDA). The new hyper-cube framework clarifies the relation of ACO algorithms to these algorithms and highlights the great advantage of ACO algorithms as the way of using the distribution of solution component probabilities for solution construction. This provides a structured way of handling constrained problems which the other approaches are lacking.

5 Conclusions

We have presented the hyper-cube framework for ACO algorithms, which provides a well defined hyperspace $\mathcal{T}$ for the pheromone values. Its potential usefulness is indicated by the discussion in section 4. The subject of our current work involves testing the diversification scheme presented in section 4 and doing comparative studies of ACO algorithms and Evolutionary Computation techniques in greater detail. We will experimentally investigate the influence of the new implementation for $\mathcal{MM}$AS, as the hyper-cube framework changes the way of setting values τ_{min} and τ_{max}. In usual $\mathcal{MM}$AS implementations, τ_{min} and τ_{max} are dependent on the upper limit for pheromone values, which gets approximated during the run of the algorithm. This leads to a repeated resetting of these values, which could potentially disturb the dynamics of the system. In the hyper-cube framework this is not the case as we know the limits for pheromone values beforehand. We also intend to investigate the potential usefulness of the hyper-cube framework for combinatorial optimization problems with linear objective functions. In this case the derivation of the objective function could be a basis for a decision about the usefulness of a solution for updating the pheromones. This could also be the start of merging ACO algorithms with techniques for cutting off certain areas of the search space (reducing the search space).

Acknowledgements: This work was supported by the "Metaheuristics Network", a Research Training Network funded by the Improving Human Potential programme of the CEC, grant HPRN-CT-1999-00106. Andrea Roli acknowledges support from the CEC through a "Marie Curie Training Site" (contract HPMT-CT-2000-00032) fellowship. The information provided is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication. Marco Dorigo acknowledges support from the Belgian FNRS, of which he is a Senior Research Associate.

References

[1] S. Baluja and R. Caruana. Removing the Genetics from the Standard Genetic Algorithm. In A. Prieditis and S. Russel, editors, *The International Conference on Machine Learning 1995*, pages 38–46, San Mateo, California, 1995. Morgan Kaufmann Publishers.

[2] M. Dorigo and G. Di Caro. The Ant Colony Optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 11–32. McGraw-Hill, 1999.

[3] M. Dorigo, V. Maniezzo, and A. Colorni. Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics - Part B*, 26(1):29–41, 1996.

[4] G. R. Harik, F. G. Lobo, and D. E. Goldberg. The Compact Genetic Algorithm. *IEEE Transactions on Evolutionary Computation*, 3(4):287–297, 1999.

[5] H. Mühlenbein and H.-M. Voigt. Gene Pool Recombination in Genetic Algorithms. In I.H. Osman and J.P. Kelly, editors, *Proc. of the Metaheuristics Conference*, Norwell, USA, 1995. Kluwer Academic Publishers.

[6] T. Stützle and H. H. Hoos. $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System. *Future Generation Computer Systems*, 16(8):889–914, 2000.

[7] G. Syswerda. Simulated Crossover in Genetic Algorithms. In L.D. Whitley, editor, *Proc. of the second workshop on Foundations of Genetic Algorithms*, pages 239–255, San Mateo, California, 1993. Morgan Kaufmann Publishers.