Automation, Control and Intelligent Systems
2013; 1(3): 53-58
Published online June 30, 2013 (http://www.sciencepublishinggroup.com/j/acis)
doi: 10.11648/j.acis.20130103.13

Ant colony optimization with genetic operations

Matej Ciba, Ivan Sekaj

Institute of Control and Industrial Informatics, Bratislava, Slovakia

Email address:
bigmato@centrum.sk(M. Ciba), ivan.sekaj@stuba.sk(I. Sekaj)

To cite this article:
Matej Ciba, Ivan Sekaj. Ant Colony Optimization with Genetic Operations. *Automation, Control and Intelligent Systems*. Vol. 1, No. 3, 2013, pp. 53-58. doi: 10.11648/j.acis.20130103.13

Abstract: This paper attempts to overcome stagnation problem of Ant Colony Optimization (ACO) algorithms. Stagnation is undesirable state which occurs at a later phases of the search process. Excessive pheromone values attract more ants and make further exploration hardly possible. This problem has been addressed by Genetic operations (GO) incorporated into ACO framework. Crossover and mutation operations have been adapted for use with ant generated strings which still have to provide feasible solutions. Genetic operations decrease selection pressure and increase probability of finding the global optimum. Extensive simulation tests were made in order to determine influence of genetic operation on algorithm performance.

Keywords: Ant Colony Optimization, Genetic Operations, Crossover, Mutation, Minimal Path Search

1. Introduction

Wide range of problems like Routing problem, Assignment problem, Scheduling problem and others can be transformed into graph representation. Exact algorithms for instance Dijkstra or Bellman-Ford appear to be slow and inefficient on large scale graphs. In this case some heuristic information which guide search process is useful. One of the well-known graph search algorithm that utilizes a heuristic is A* search [1] or ACO algorithm.

Ant colony optimization represents an efficient tool for optimization and design of graph oriented problems. It is a multi-agent meta-heuristic approach and was first purposed by M. Dorigo et al. [2] as Ant system (AS) algorithm.

During the search process each ant sets off from ant colony (start position) and moves to search food (destination). The aim is to find the shortest path. As ants are passing the terrain (graph) they mark used routes (arcs of the graph) by chemical substance called pheromone. On their way back they use the same way from which abundant loops has been removed, but the amount of pheromone (1) $\Delta\tau^k_{ij}(t)$ they produced is inversely proportional to the tour length $L^k(t)$.

$$\Delta\tau^k_{ij}(t)=\begin{cases} Q/L^k(t) & if\,(i,j)\in T^k(t) \\ 0 & if\,(i,j)\notin T^k(t) \end{cases} \quad (1)$$

$T^k(t)$ is the tour generated by ant k, Q is a constant and tuple (i, j) denotes beginning and termination node of an arc.

All pheromone tracks (2) are preserved by arcs of the graph

$$\tau_{ij}(t+1)=(1-\rho)\tau_{ij}(t)+\sum_{k=1}^{m}\Delta\tau^k_{ij}(t) \quad (2)$$

where $\rho \in (0,1)$ is the pheromone persistence (1 - ρ is evaporation rate) and m is the number of ants. Evaporation rate is a user adjusted parameter and affects pheromone durability; i.e. how long the acquired information will be available. Too high values causes random search, too low values get algorithm stock in local optimum.

An ant in each node has to make a decision which arc to take. At the beginning when no pheromone values are available heuristic values η_{ij} takes dominance. Later the ant uses probability selection rule to choose the next arc according to

$$p^k_{ij}(t)=\frac{p_{ij}(t)}{\sum_{j\in N^k_i}p_{ij}(t)} \quad (3)$$

where $p^k_{ij}(t)$ is probability the ant k chooses the arc (i, j) from the neighborhood N^k_i of node i except the node visited previously. The more pheromone is located on particular arc, the more attractive it appears. The probability $p_{ij}(t)$ of choosing the particular arc (i, j) depends on pheromone $\tau_{ij}(t)$ and the heuristic η_{ij} values which are associated with the arc (4).

$$p_{ij}^{k}(t)=\frac{\tau_{ij}^{\alpha}(t)+\eta_{ij}^{\beta}}{\sum_{j\in N_i^k}\tau_{ij}^{\alpha}(t)+\eta_{ij}^{\beta}} \quad (4)$$

Symbols α and β are weight parameters and represents balance between ant's gathered knowledge and the user preferred area. Heuristic values η_{ij} affect probability only at the beginning when pheromone values are low.

2. Modification of ACO Algorithm

Disadvantages of ACO algorithms are (i) many user parameters and (ii) the selection pressure. The first point is a property of the algorithm, while the second point has had many papers devoted to it. Let's mention ant colony system (ACS) with *pseudo-random proportional* rule [3] in which random uniformly distributed variable $q \in (0,1)$ is compared with a tunable parameter $q_0 \in \langle 0,1\rangle$. If $q > q_0$ then

$$p_{ij}^{k}(t)=\begin{cases}1 & if \quad j=\arg\max p_{ij}\\ 0 & otherwise\end{cases} \quad (5)$$

else the probability selection rule (3) is applied; random selection applied to AS_{rank} [4] where *random selection rate r* is the probability of random selection and represents a user parameter which adjusts the balance between exploration and exploitation; prevention of quick convergence (i) and stagnation avoidance (ii) mechanisms applied to AS [5].

The mechanism for prevention of quick convergence (i) is based on *pseudo-random proportional* rule [3], but the tunable parameter q_0 is dependent on algorithm iteration (6)

$$q_0=\frac{\log_e(NC)}{\log_e(N_{\max})} \quad (6)$$

where NC is the current iteration and N_{max} is the termination iteration.

The stagnation avoidance mechanism (ii) is based on the comparison of a randomly generated quantity $q \in (0,1)$ with probability $p^k_{ij}(t)$ of selected arc. If $q \geq p^k_{ij}(t)$, then choose the next node randomly. This occurs in later stages of the search process, where pheromone values tend to be high, and thus the chance of further exploration is low.

3. Genetic Operations Applied on ACO

Genetic algorithms (GA) were proposed by Holland (1975). The original GA is known as simple genetic algorithm (SGA). GA belongs to adaptive stochastic optimization class and is typically used for combinatorial problems. The four main components of GA are representation (i), mutation (ii), crossover (iii) and selection (vi) mechanism. Each component is adapted in order to provide feasible solution for ACO algorithm.

In ACO algorithms representation (i) of genotype space is sequence of nodes:

$$n^L; \quad n \in N \quad (7)$$

where gene n is graph node and L is path length. The population size is given by the number of genomes, i.e. the number of ants m which generate the set of paths within one generation.

Mutation (ii) mimics random gene changes. In each genome each gene is changed with the equal probability. The simplest form is one point mutation on Fig. 1. In ACO adaptation the first and the last node is excluded from mutation. For feasibility reason the replacement node n_r (new gene) is such a node from the node n_i neighborhood N_i, to which an arc from n_i predecessor n_p to n_i successor n_s exists (Fig. 2). If more such nodes occur, random selection is applied. If no such node exists, another gene is randomly picked up from the list.

In ACO algorithm crossover position is represented by a common node of parental strings except the first and the last node (Fig. 3). If more of such nodes exist, random selection is applied. Crossover operation makes sense only if both child strings differ from their parents.

In GA many selection (vi) mechanisms are available, like roulette-wheel selection, tournament selection, stochastic universal sampling or reward-based selection [7]. Since optimization process is primary done by ants cooperative behavior, the selection process has purely random concept and genetic operations serve just for selection pressure decrease.

On both figures survivor strategy where parents are replaced by their children is shown. No string is allowed to take the same genetic operation more than once.

Figure 1. *One point mutation.*

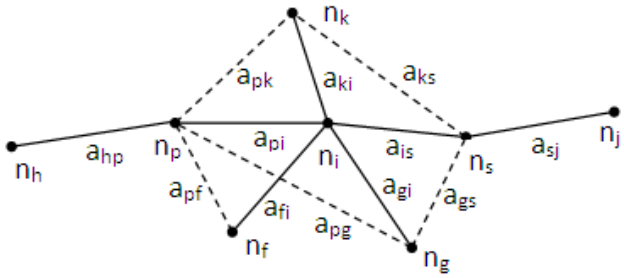

Figure 2. *Candidates which can replace node n_i are n_g and n_k only.*

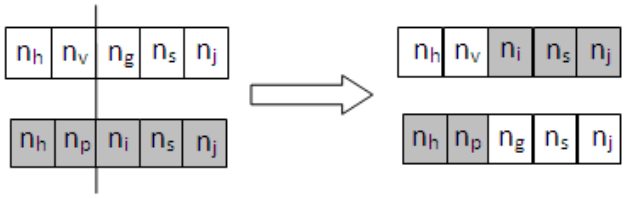

Figure 3. *One point crossover.*

4. ACO Algorithm with Genetic Operations

The above described genetic operations have been applied to one of the best performing ACO algorithms of Kumar, Tiwari and Shankar (ACO_{KTS}) [5]. At the end of each cycle *t*, when all the ants finish their tours $T^k(t)$, genetic operations are applied on the $T^k(t)$ strings which represents the list of nodes. Prior the genetic operations all loops are removed from the tours.

At first mutation is applied. It is applied on random selected tour $T^k(t)$ in random selected node. If mutation is not feasible, another node is chosen. If more candidates by which the selected node can be replaced occur, the new node is random chosen from the candidates. If mutation fails on all nodes of the tour, another tour is chosen.

After all mutation operations are performed, crossover operations are applied. Parent strings are random selected. If crossover operation is not feasible, another second string is selected. If no tour has common node with the first selected tour, another first tour is selected and the random selection process is repeated.

Since genetic operations may produce strings with loops, in ACO framework prior and immediately after each genetic operation a loop removal procedure is performed. After all genetic operations are executed fitness evaluation and pheromone update are scheduled.

Genetic operations do not have to be necessarily feasible. Feasibility of genetic operations depends on the graph and generated tours. For this purpose ACO_{GO} algorithm has embedded user feedback which represents a ratio between accomplished and required genetic operations. Two of such rates are provided; one for each genetic operation.

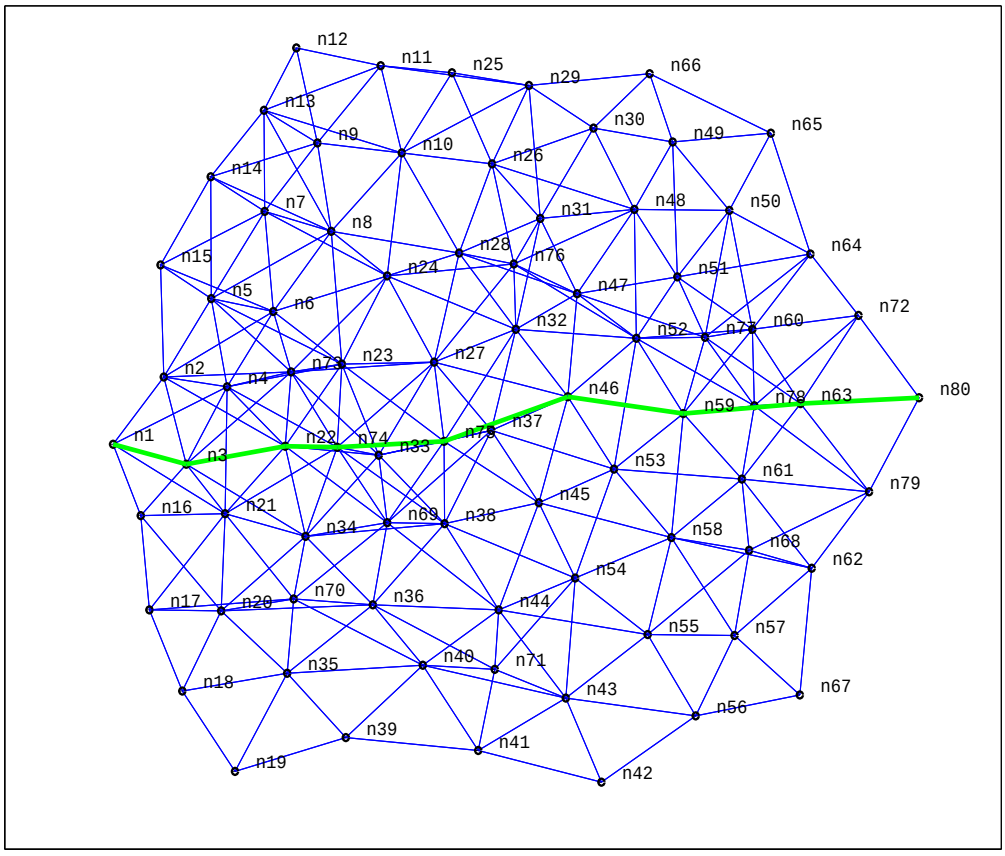

Figure 4. *The 80 node graph with dashed minimal path*

5. Case Study

The above described ACO_{GO} algorithm has been tested on a random generated graph. Common ACO parameter values were set in accordance with [8] and are listed in the Table 1.

The value for the number of cycles represents three macro cycles of ACO_{MC} [9] for the same graph and parameters.

Test graph is a symmetrical multi-graph with 80 nodes and 300 arcs (Fig. 4). Node coordinates *x,y* fall in range *<0,1>* and arc's values c_{ij} represent the arc lengths. The task is to find the shortest path between start node $n_s = 1$ and end node $n_e = 80$.

Variable parameters were set to determine the influence of the genetic operations quantity on algorithm performance and effect of distribution of mutation operations between paths. For each setting 500 trials were performed. For test reconciliation probability *n [%]* of finding the global optimum (T^* = [1 3 22 74 75 46 59 63 80]) was evaluated.

Table 1. *Common ACO parameters settings*

Parameter name	Value
Initial pheromone value $\tau_{ij}(0)$	0.1
Weight of pheromone information α	0.5
Heuristic values η_{ij}	0.1
Weight of heuristic information β	0.1
Pheromone persistence ρ	0.05
Number of ants m	10
Number of cycles	200

6. Results

Simulation results were divided into three groups according to number of crossover pairs and are listed in the Table 2.

Table 2. *Simulation results for various GO settings*

Mutation paths	Mutations per paths	Probability of finding the global optimum [%]		
		No crossover operation	One crossover pair	Two crossover pairs
0	0	5.6	4	6.2
1	1	6.4	6	6
	2	6.8	8.8	3.6
	3	6	6	7
	4	7.6	5.6	7
	5	6.2	7.2	7.4
2	1	6.4	8.6	6.2
	2	6.4	7.4	7.2
	3	7.8	7	7.2
	4	8.8	6.2	8
	5	7.6	10	9.6
3	1	7.4	5.6	6.4
	2	8.2	5.8	8.4
	3	9.2	7.4	7.8
	4	9	9	9.4
	5	8.2	7.4	9.6
4	1	6.4	8.2	7.6
	2	8.8	8.8	9
	3	10.4	13	8.4
	4	10	9.6	10.2
	5	9.4	8.2	11.6
5	1	8.8	8	6.6
	2	9.6	8.6	9
	3	11.2	8	9.2
	4	11.2	10.2	12
	5	11.2	9	10

6.1. Mutation effect

The reference value of *n [%]* was received without any genetic operation and is 5.6 (Table 2, row 1). The results received with GO are better almost in any case. It can be seen that the higher number of mutation operations, the better the performance is (Tables 2).

For better results representation three graphs are provided. Their color map was set to show green - blue when the results are worse than reference value and yellow to red otherwise (Fig. 5 – 7).

The outcome with different mutation distribution is asymmetric. Results received without crossover operation have higher values along with the Mutation paths axis (Fig. 5). However, results received with two crossover pairs have higher values along with the Mutations per path axis (Fig. 7). I.e. with no crossover pair certain amount of mutation operations should be spread out among more paths, but with 2 crossover pairs concentration of mutation operation on less paths tends to perform better. This behavior may be caused by the execution order of the GO: crossover is applied after mutation, thus crossover may re-distribute mutation substrings between more paths.

The results for one crossover pair show different behavior. It does not have the highest value on edges of the surface where the highest amount mutation operation is. The highest value 13% was received with four mutation paths with three mutation operation per path.

Genetic operations where nearly always feasible; ratio accomplished / required mutation operations is 100% and for the crossover operation over 99%.

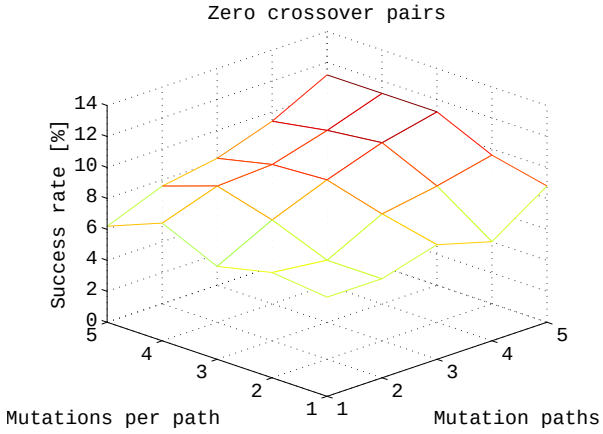

Figure 5. *Zero crossover pairs results*

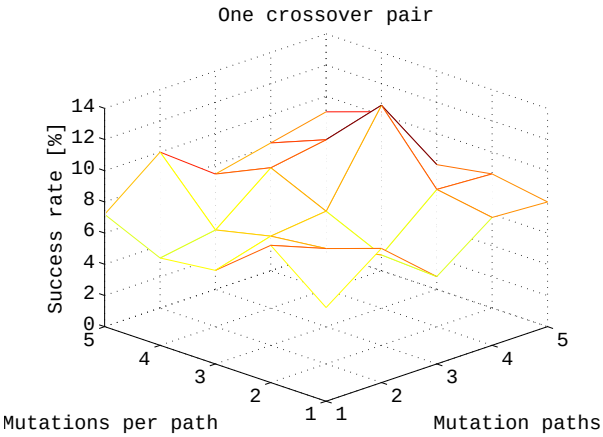

Figure 6. *One crossover pair results*

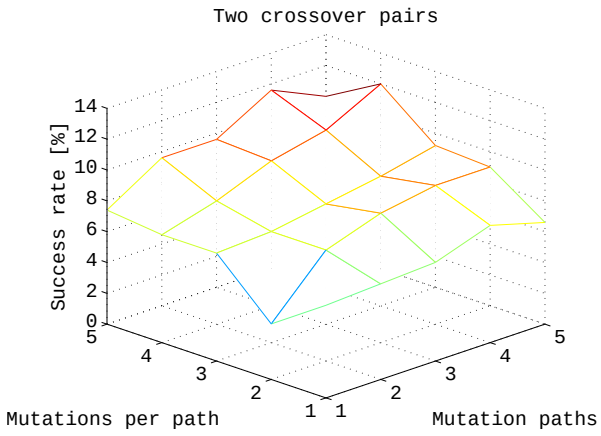

Figure 7. *Two crossover pairs results*

6.2. Crossover effect

In order to determine the effect of crossover operation crossover rate was let to grow up to 100% (Table 3). To prevent interference, no mutation operation was allowed.

The results vary (Fig. 8); the highest output was gained for 60% of crossover rate. Beyond 60% threshold ants foraging behavior is suppressed by crossover overload. As the crossover rate increases, ratio accomplished / required operation decreases (Fig. 9). This is caused by the search space dimension. It is too large for ten ants to meet.

GO does not affect the length of the search process. The mean value of the cycle when the best value was found is 109.081 with standard deviation 2.617.

Table 3. *Crossover operation results*

Crossover rate [%]	Probability of finding global optimum [%]	Valid crossover operations [%]
0	*5.6*	*N/A*
20	*4*	*0.99953*
40	*6.2*	*0.99498*
60	*7.4*	*0.96859*
80	*5*	*0.88374*
100	*5.8*	*0.73246*

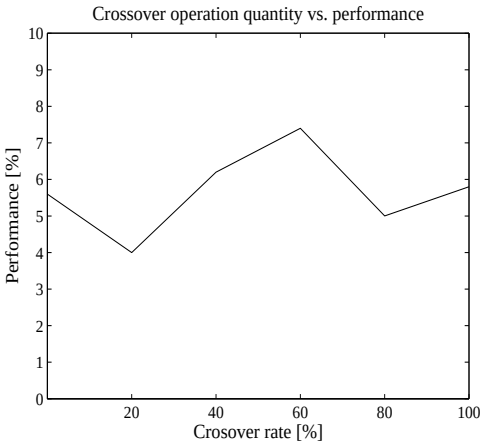

Figure 8: *Crossover operation quantity vs. performance*

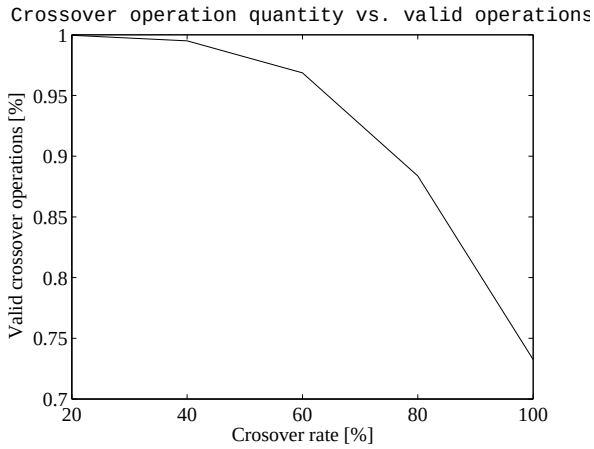

Figure 9: *Crossover operation quantity vs. valid operations*

7. Conclusion

It has been proved that genetic operations increase ACO algorithm performance. Even small number of any genetic operation causes positive effect.

Limit of crossover is 60% of crossover rate. The higher the crossover rate, the lower the accomplished / required ratio. Mutation operation causes better results than crossover operation. This can be explained by the nature of the mutation operation which creates new paths whilst crossover operation can only combine already existing solutions.

The higher amount of mutation operations the higher the performance gain is. No limit for amount of mutation operation was found during the simulation. Without crossover operation distributed mutation operation has better performance, but with two crossover pairs concentrated mutation operation on less paths tends to perform better. The impact of the GO execution order on the mutation operation distribution needs to be verified.

The results are promising; GO improves ACO algorithm performance more than twice. Further research and more experiments are needed to determine the distribution and optimal amount of mutation operation with respect to the number of ants and length of the paths.

Acknowledgments

Thanks to Science Publishing reviewers for valuable feedback and provided comments which increased the paper quality.

References

[1] P. E. Hart, N. J. Nilsson and B. Raphael, A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics SSC4 4(2), 1968, 100–107

[2] M. Dorigo, G. Caro and L. Gambardella, Ant algorithms for discrete optimization, Artificial Life, 5(2), 1999, 137-172

[3] L. Gambardella and M. Dorigo, Solving symmetric and

asymmetric TSPs by ant colonies, In Proceedings of the IEEE Conference on Evolutionary Computation, ICEC96, IEEE Press, 1996, 622–627

[4] Y. Nakamichi and T. Arita, Diversity control in ant colony optimization, In Abbas HA (ed) Proceedings of the Inaugural Workshop on Artificial Life (AL'01), Adelaide, Australia, Dec 11, 2001, 70-78

[5] R. Kumar M. K. Tiwari and R. Shankar, Scheduling of flexible manufacturing systems: An ant colony optimization approach, proc. Instn. Mech. Engrs Vol. 217 Part B: J. Engineering Manufacture, 2003, 1443–1453

[6] J. H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, University of Michigan Press, 1975

[7] I Sekaj, Evolucne vypocty a ich vyuzitie v praxi, IRIS Press, 2005

[8] M. Becker and H. Szczerbicka, Parameters influencing the performance of ant algorithms applied to optimization of buffer size in manufacturing, IEMS Vol. 4, No. 2, December 2005, 184–191

[9] M. Ciba, ACO algorithm with macro cycles, Proceedings on 14th Conference of Doctorial Students on Elitech'12, Slovak Technical University of Bratislava, May 2012