Available online at www.sciencedirect.com
ScienceDirect
Progress in Natural Science 18 (2008) 1173–1178
Progress in Natural Science
www.elsevier.com/locate/pnsc

Short communication

An ant colony algorithm for solving Max-cut problem

Lin Gao [a,*], Yan Zeng [b], Anguo Dong [a,c]

[a] School of Computer Science and Technology, Xidian University, Xi'an 710071, China
[b] Computer Science Department, Xi'an Institute of Post & Telecommunications, Xi'an 710061, China
[c] School of Science, Chang'an University, Xi'an 710064, China

Received 4 March 2008; received in revised form 31 March 2008; accepted 1 April 2008

Abstract

Max-cut problem is an NP-complete and classical combinatorial optimization problem that has a wide range of applications in different domains, such as bioinformatics, network optimization, statistical physics, and very large scale integration design. In this paper we investigate the capabilities of the ant colony optimization (ACO) heuristic for solving the Max-cut problem and present an AntCut algorithm. A large number of simulation experiments show that the algorithm can solve the Max-cut problem more efficiently and effectively.

Keywords: Ant colony optimization; Max-cut problem; Algorithm

1. Introduction

Ant colony optimization algorithm (ACO) presented by the Italian scholar Dorigo et al. [1] in the early 1990s is a population-based heuristic search algorithm. It simulates an ant colony's foraging behavior to solve the given problem. Ants can often find the shortest path between food source and their nest. ACO has been applied successfully to a large number of difficult combinatorial optimization problems, such as traveling salesman problem (TSP) [2], quadratic assignment problem [3], job-shop problem [4], and knapsack problem [5]. Max-cut problem is an NP-complete and classical combinatorial optimization problem that has a wide range of applications in different domains, such as bioinformatics, network optimization, statistical physics, and very large scale integration design. Currently, most methods used in solving the Max-cut problem are based on semi-definite programming relaxations (SDP relaxations) algorithm. The idea that the Max-cut problem can be naturally relaxed to a semi-definite programming problem was first observed by Lovasz [6] and Shor [7]. Goemans and Williamson [8] proposed a randomized algorithm that uses semi-definite programming to achieve a performance guarantee of 0.87856. Some researchers used other methods to solve the Max-cut problem. In 2001, Wu et al. [9] used a discrete Hopfield-type neural network to resolve the Max-cut problem. In 2003, Ono and Hirata [10] presented the spectrum method for the Max-cut problem. However, each method has its own shortcomings. Therefore, people are still seeking the efficient and effective solution to the Max-cut problem. In this study, we investigate the capabilities of the ant colony optimization (ACO) heuristic for solving the Max-cut problem and provide new insight into this problem. We call the ACO algorithm for the Max-cut problem "AntCut".

2. Max-cut problem

Here, we give the mathematical description of the Max-cut problem [11].

* Corresponding author. Tel.: +86 29 88202696; fax: +86 29 88201531.
E-mail address: lgao@mail.xidian.edu.cn (L. Gao).

1002-0071/$ - see front matter
doi:10.1016/j.pnsc.2008.04.006

Definition 1 (*Max-cut problem*). Given an undirected graph $G=(V,E)$, where $V=\{1,2,\ldots n\}$ is the set of vertices, $E\subseteq V\times V$ is the set of edges, w_{ij} is the weight of the edge e_{ij} which connects vertices i and j, and a cut S of graph G is any subset of V. For each S, the weight of the cut S is defined as:

$$\mathrm{Cut}(S)=\sum_{e_{ij}\in\delta(S)} w_{ij} \tag{1}$$

where $\delta(S)$ is the set of edges that one endpoint belongs to S while the other endpoints belong to $V-S$. Max-cut problem is to find an S, making the biggest $\mathrm{Cut}(S)$.

Definition 2 (*Cut vector*). A cut vector $X\in\{-1,1\}^n$ is defined as:

$$x_i=\begin{cases}1 & i\in S\\ -1 & i\in V-S\end{cases} \tag{2}$$

Then the Max-cut problem is equivalent to the following problem:

$$\max_{x\in\{-1,1\}^n}\sum_{i<j} w_{ij}\frac{1-x_ix_j}{2} \tag{3}$$

3. ACO for Max-cut problem

Max-cut problem is to find $(S,V-S)$, which is a cut of vertex set V, making the maximum weight of edges crossing the cut (one endpoint in S, the other endpoint in $V-S$). AntCut uses the ACO heuristic to choose, at each step, a vertex to change the set that the vertex belongs to, and gradually approaching to the max cut. The choice of the probability of each vertex should be in proportion to the increasing weight of the cut by changing the set that the chosen vertex belongs to.

3.1. Pheromone strategy

AntCut needs to satisfy the following conditions. The bigger the weight of the cut is, the higher the probability of the edges being cut is, and the greater the pheromone on the edge is.

The amount of pheromone on edge e_{ij} is represented by τ_{ij}, which represents the learned desirability for the edge e_{ij} crossing the cut. The upper bound $\tau_{\max}$ and lower bound $\tau_{\min}$ are given, and the pheromone value of each edge is initialized to be $\tau_{\max}$. Then the global pheromone is updated by

$$\tau_{ij}(t+1)=(1-\rho)\tau_{ij}(t)+\Delta\tau_{ij}(t) \tag{4}$$

where ρ is an evaporate factor as well as a constant, and t is the number of iterations.

The gain of pheromone on edge e_{ij} is represented by $\Delta\tau_{ij}$ as

$$\Delta\tau_{ij}=\frac{1}{Q+\mathrm{Cut}(S_{\mathrm{best}})-\mathrm{Cut}(S_{\mathrm{better}})} \tag{5}$$
$$\text{if } e_{ij}\in\delta(B_{\mathrm{best}})\cup\delta(B_{\mathrm{better}}),\ \text{else } \Delta\tau_{ij}=0,$$

where $\mathrm{Cut}(S_{\mathrm{best}})$ is the biggest cut that the ants find in this cycle, $\mathrm{Cut}(S_{\mathrm{better}})$ is the bigger one, and Q is a constant, which controls the size of $\Delta\tau_{ij}$.

3.2. Selection of the probability

The choice of the probability $p(v_i)$ denotes the probability of changing the set that the vertex v_i belongs to. If v_i belongs to S, then $p(v_i)$ is the probability of shifting v_i into $V-S$. Otherwise, if v_i belongs to $V-S$, then $p(v_i)$ is the probability of shifting v_i into S

$$p(v_i)=\frac{\Delta f_i}{\sum_{j\in\mathrm{Candidate}}\Delta f_j} \tag{6}$$

where Δf_i is the increasing value of the cut by changing the set that v_i belongs to and is decided by the pheromone and the weight of all the edges connected by the vertex v_i

$$\Delta f_i=\sum_{e_{ij}\in E} x_ix_j\tau_{ij}^{\alpha}w_{ij}^{\beta},\quad i=1,2,\ldots,n \tag{7}$$

where x_i is cut, τ_{ij} is the amount of the pheromone on the edge e_{ij}, w_{ij} is the weight of the edge e_{ij}, and as the local heuristic information, α and β are the weight factors of pheromone and local heuristic information, respectively.

When all the edges e_{ij} belong to the same set (that is, e_{ij} is not cut), x_i and x_j have the same symbol, and $x_ix_j\tau_{ij}^{\alpha}w_{ij}^{\beta}$ is positive. Otherwise, when the edges e_{ij} belong to the different sets (that is, e_{ij} is cut), x_i and x_j are opposite, and $x_ix_j\tau_{ij}^{\alpha}w_{ij}^{\beta}$ is negative. Eq. (7) shows that Δf_i equals the sum of the uncut edge minus the sum of the weight of the cut edges. Δf_i is the amount of expected cut increase when the set that vertex v_i belongs to is changed. The higher Δf_i is, the greater the expected increase of weight of the cut will be, and the greater the probability of the selected vertices v_i is.

Δf_i is gradually obtained in the following iterative process:

(1) Because the initial value of cut vector $\boldsymbol{x}$ is $\{1\}^n$, the initial value of Δf_i is

$$\Delta f_i=\sum_{e_{ij}\in E}\tau_{ij}^{\alpha}w_{ij}^{\beta},\quad i=1,2,\ldots,n \tag{8}$$

(2) When choosing a vertex v_i to change the set it belongs to (that is, to change the value of the cutting vector), Δf_i should be adjusted based on the following steps: Firstly, adjust the value of Δf_i using

$$\Delta f_i=-\Delta f_i \tag{9}$$

Next, for each v_j connected with v_i, we adjust Δf_i as

$$\Delta f_j=\Delta f_j+2x_ix_j\tau_{ij}^{\alpha}w_{ij}^{\beta} \tag{10}$$

This would ensure that Δf_i is equivalent to the expected increase of the weight of the cut after the set that vertices belong to is changed.

In Eq. (6), the candidate is the set of the candidate components, and the elements are all the vertices with the Δf_i value larger than zero. We change the value of Δf_i according to the symbol of Δf_i, and also adjust the candidate set.

Note that here $p(v_i)$ is defined as the probability of changing the set that vertex v_i belongs to, rather than the probability of picking v_i to join the S. If always choosing vertices from set (V-S) to join S, it could lead to that, for some vertices v_k in S, many vertices connected with v_k will successively join S. If we remove v_k from the set S, the weight of the cut may increase. Fig. 1 illustrates the choice of vertex, in which we use i, j and x to represent the number of vertices belonging to different sets.

3.3. Local search strategy

After the cut is constructed by each cut, calculating all the vertices based on

$$\Delta l_i = \sum_{e_{if} \in E} x_i x_j w_{ij}, \quad i = 1, 2, \ldots, n \tag{11}$$

If we can find a Δl_i greater than 0, then we change the set that v_i belongs to. If there are more Δl_i with the value greater than 0, then we choose the vertex according to the greed strategy, and change the set of the vertex. Furthermore, we update Δl_i based on the same strategy with Δf_is, that is, $\alpha = 0$, $\beta = 1$, until there is no vertex whose Δl_i is larger than 0.

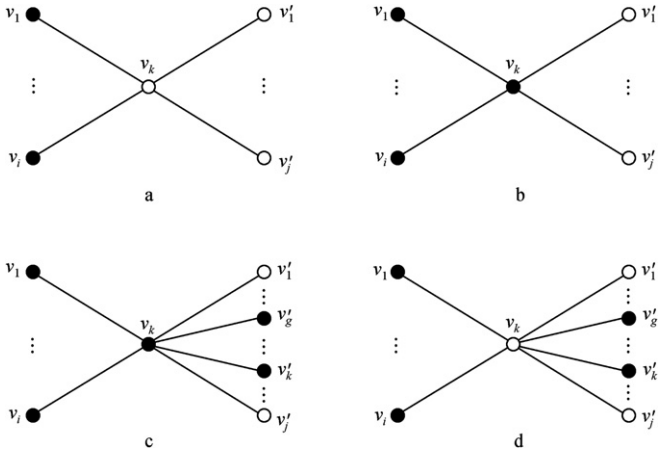

Fig. 1. Four kinds of choices of vertex. (a) The vertices $v_1, \ldots v_i \in S$ connected with the vertices $v_1 \ldots v'_j \in V - S$ through the vertex v_k ($i < j$); (b) v_k selected then added to S, and the cut amount increased by $j - i$; (c) $v'_g, \ldots v'_h \in V - S$ gradually added to S; (d) v_k removed from S, and the amount of cut edge increased by $i - j + 2 * (h - g)$.

3.4. Algorithm framework

Pseudo-code description of the algorithm.

```
Procedure AntCut
  Initialization of pheromone trails to τmax
    While not satisfied with the end conditions
      For each ant of the group
        Initiate cut vector Δfi and candidate set;
        While candidate set is not empty do
          Choose a vertex vi from candidate set with
p(vi)
          Update cut vector Δfi and candidate set;
      Local search
    Global pheromone-update
  End of procedure
```

The end condition here in the third line is generally confirmed by reaching the largest number of cycles and finding the optimal solution. If any one of these two conditions is met, the whole repetition can be ended and the so-called optimal solution is found.

4. Parameter selection and discussion

There are many parameters in the ant colony algorithm, and these parameters always have been selected based on experience. To improve the algorithm's computational performance, we first make some experiments with the parameters in order to get experience for the selection of parameters. We focus on the evaluation of parameters α, β and ρ. Then we consider the other parameters, such as the number of ants, and the maximum number of iterations. In the case of increasing these parameters, the quality of the solution will be improved. These parameters are determined based on the specific scale of the problem. We do not discuss it in detail here.

4.1. Selection of heuristic factors α and β

We performed experiments with graph $G1$, $G11$ in G-set, the test problems in G-set were generated by Helmberg and Rendl using a graph generator written by Rinaldi. Many algorithms used to solve the Max-cut problem are tested with this group of test problems. The number of vertices in $G1$ is 800, the number of edges in $G1$ is 19176, the edge density is 6.12%, and the weight of all the edges is 1; while the number of vertices in $G11$ is 800, the number of edges in $G11$ is 1600, the edge density is 0.63%, and the weight of the edges is -1 or 1.

The following parameter groups were taken in experiment: the number of ants is 3, the largest number of iterations is 1000, and $\rho = 0.005$. We take different numerical combinations of α and β. The operational results are obtained by comparing the optimal solution (average value) over 10 runs as shown in Table 1.

Table 1
Influence of heuristic factors α and β on the performance of AntCut algorithm

Graph	α	β			
		0	1	2	5
G1	0	11498(11384.2)	11509(11489.3)	11488(11385.0)	11429(11395.5)
	1	11506(11485.8)	11608(11590.6)	11508(11484.1)	11485(11453.4)
	2	11506(11485.2)	11606(11590.9)	11505(11489.5)	11495(11485.6)
	3	11502(11469.6)	11604(11588.1)	11503(11472.9)	11479(11435.8)
	5	11510(11465.5)	11609(11585.0)	11504(11438.7)	11484(11391.7)
G11	0	406(397.8)	488(455.6)	406(399.8)	402(395.4)
	1	400(395.4)	508(505.6)	418(405.2)	407(399.8)
	2	408(402.4)	508(506.00)	426(416.0)	410(400.2)
	3	418(406.0)	512(505.2)	426(422.8)	411(402.4)
	5	424(417.20)	512(506.2)	436(428.4)	425(412.8)

First, from the experimental data we see that the value of β can only be 1. When the value of β is not equal to 1, the amount of the maximum cut will drop substantially. The reason is that we can see from Eq. (7) that β acts on w_{ij} (the weight of each edge). If $\beta = 0$, it means that the weight of edges in the ants searching process is completely ignored. If $\beta > 1$, the weight of edges is over emphasized. Therefore, the value of β can only be 1.

Secondly, when $\beta = 1$, if $\alpha = 0$, the pheromone is ineffective, and ants search is completely under the guidance weight of the edges. The solution they find is clearly inferior to the value when α is not 0, and so introduction of the pheromone is advantageous for the ants in finding the optimal solution. When α is relatively larger ($\alpha = 5$), the best solution over 10 runs may be improved. However, the average of the solution drops. It means when increasing α, pheromone plays a more important role in guiding the ant. There might be some better solutions, but it may run into the local search more easily, and algorithm stability is reduced.

Through these experiments and analyses above, we can see that the choice of $\alpha = 2$ or $\alpha = 3$, $\beta = 1$ can help us achieve satisfactory results.

4.2. Adjustment of evaporate factor

When doing the adjustment of the evaporate factor, we consider not only its relation with the pheromone heuristic factor α, but also its relation with the increase of pheromone $\Delta\tau$.

The definition of $\Delta\tau$ is in Eq. (5), in which the definition of constant Q is given by the experience, and the value range of $\Delta\tau$ is $(0, 1/Q]$. The more the solution ants find in this iterative cycle, the closer the value of $\Delta\tau$ is to $1/Q$. The pheromone trail widens the gap between levels in the joint action of $\Delta\tau$ and ρ. The analyses of the relation between $\Delta\tau$ and ρ are as follows:

① $\Delta\tau$ is too small, and ρ is too large: The pheromone's accumulation of all the edges is too small and it volatilizes too fast, and so it tends to $\tau_{\min}$ rapidly. This search process is similar to random search, which is very difficult to find good solutions.

② $\Delta\tau$ is too small, and ρ is too small: The pheromone trail volatilizes too slowly and has a small increase. During the initial iteration, the difference in pheromone trails is small, which is similar to random search. But after a certain number of iterations, pheromone trails will open a reasonable gap. Then some more outstanding solution would be found. Because they are too small, a large number of iterations are required.

③ $\Delta\tau$ is too large, and ρ is too large: After a certain number of iterations, pheromone trails reach two extreme conditions: (i) as they volatilize too much, the pheromone of part edges is close to $\tau_{\min}$; (ii) as $\Delta\tau$ is too large, the pheromone of part edges is close to $\tau_{\max}$.

④ $\Delta\tau$ is too large, and ρ is too small: Pheromone trails are close to $\tau_{\max}$, which cannot open the gap.

In summary, the ρ should not be too large. Therefore, the following experiment will limit ρ within (0.002–0.150), while the corresponding Q is 100. The other parameters are: the number of ants is 3, $\alpha = 3$, $\beta = 1$, and the maximum of iterations is 1000. The results in Table 2 are the optimal (average) result over 10 runs.

From the above results we can see that when $\rho = 0.005$, the cut that the algorithm finds is the best.

Through the above experiment, we optimized parameters in the AntCut algorithm: $\alpha = 2$ or $\alpha = 3$, $\beta = 1$, $\rho = 0.005$, which are used in our simulation experiment with the graphs in *G*-set by AntCut algorithm.

Table 2
The influence of evaporate factor ρ on the performance of the algorithm

ρ	Sample graph G1		Sample graph G11	
	Optimal	Average	Optimal	Average
0.15	11510	11498.8	496	468.5
0.10	11553	11528.5	494	467.8
0.08	11556	11539.1	498	475.2
0.05	11559	11534.9	500	482.6
0.03	11563	11543.1	505	490.2
0.02	11576	11553.4	510	498.3
0.01	11595	11575.4	508	497.6
0.005	11608	11582.4	514	504.5
0.002	11603	11580.6	512	502.6

Table 3
The operation results of AntCut

No.	Graph	|V|	Edge density (%)	AntCut	SDP bound	0.879 SDP bound
1	G1	800	6.12	11611 (11586.5)	12087	10612
	G2			11605 (11581.0)	12084	10617
	G3			11634 (11609.4)	12077	10611
2	G11	800	0.63	519 (509.8)	627	551
	G12			518 (507.7)	621	546
	G13			529 (518.2)	645	567
3	G14	800	1.58	3042 (3030.8)	3187	2800
	G15			3015 (3008.5)	3169	2785
	G16			3017 (3008.2)	3172	2782
4	G22	2000	1.05	13244 (13224.5)	14123	12408
	G23			13218 (13198.3)	14129	12414
	G24			13156 13137.1)	14131	12415
5	G32	2000	0.25	1262 (1249.8)	1560	1371
	G33			1238 (1221.9)	1537	1351
	G34			1198 (1169.1)	1541	1354
6	G35	2000	0.64	7546 (7521.7)	8000	7029
	G36			7531 (7510.2)	7996	7025
	G37			7570 (7546.6)	8009	7037
7	G43	1000	2.10	6651 (6631.5)	7027	6174
	G44			6640 (6613.9)	7022	6170
	G45			6640 (6618.6)	7020	6168
8	G48	3000	0.17	6000 (5989.7)	6000	5272
	G49			6000 (5991.6)	6000	5272
	G50			5880 (5876.2)	5988	5261

5. Experimental results and conclusions

The *G*-set consists of graphs of various sparsity and sizes. These graphs vary in size from 800 to 3000 nodes and in density from 0.17% to 6.12%, which were used to validate the AntCut algorithm.

Table 3 gives the optimal values over 10 runs. Parameters in Table 3: the number of ants is 3, $\alpha = 2$, $\beta = 1$, $\rho = 0.005$, and the maximum of iterations is 1500.

From Table 3 we may see that the AntCut algorithm can solve most of the Max-cut problems successfully. The majority of results in 18 examples among the 24 cases are better than 0.879 SDP bound, particularly for the samples G48, G49, G50. However, for those samples like G11, G12, G13, G32, G33 and G34, the results are relatively poor. Through the observation, we know that after a certain period of time (approximately 600 iterations), the algorithm performance began to go down. This is similar to the deception and bias [12] in evolutionary algorithms.

In recent years, many studies have found a similar phenomenon. For example, Blum and Sampels [13] studied the application of the ACO algorithm to shop scheduling problems. They discovered that the performance of the ACO algorithm may decrease over time, depending on the pheromone model and problem instance. This behavior is clearly undesirable, because in general it worsens the probability of finding better solution over time. Montgomery et al. [14] are trying to extend the work of Blum and Sampels [13] to assignment problems, and to attribute search bias to different algorithm components. Blum and Dorigo [15] have studied the second order deception encountered by them when they were seeking the combinatorial optimization problems with the ACO algorithm. However, the reasons for the decrease in algorithm performance remain unknown, and further studies are needed.

Acknowledgements

This work was supported by National Natural Science Foundation of China (Grant No. 60574039), Xi'an Applied Materials Innovation Fund (Grant No. XA-AM-200605) and SRF for ROCS.

References

[1] Dorigo M, Maniezzo V, Colorni A. The Ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern B 1996;26(1):29–41.
[2] Dorigo M, Gambardella LM. Ant colonies for the traveling salesman problem. Biosystems 1997;43(2):73–81.
[3] Talbi EG, Roux O, Fonlupt C, et al. Parallel ant colonies for the quadratic assignment problem. Future Gen Comput Syst 2001;17(4):441–9.
[4] Xing WX, Xie JX. Modern optimization methods. Beijing: Tsinghua University Press; 2001.
[5] Dorigo M, Stützle T. The ant colony optimization meta-heuristic: algorithms, applications, and advances. Technical Report IRIDIA-2000-32, Universit'e Libre de Bruxelles, 2000.
[6] Lovasz L. On the Shannon capacity of a graph. IEEE Trans Inform Theory 1979;25:1–7.
[7] Shor NZ. Quadratic optimization problems. Sov J Comput Syst Sci 1987;25:1–11.
[8] Goemans MX, Williams DP. Improved approximation algorithms for Max-Cut and satisfiability problems using semidefinite programming. J ACM 1995;42:1115–45.

[9] Wu LY, Zhang XS, Zhang JL. Application of discrete Hopfield-type neural network for max-cut problem. In: Proceedings of ICONIP, 2001. p. 1439–44.
[10] Ono T, Hirata T. Spectrum method for the MaxCut problem. In: MIC2003, the fifth metaheuristics international conference, Kyoto, Japan, 2003. p. 25–8.
[11] Wang CR. Graph theory. 3rd ed. Beijing: Beijing Institute of Technology Press; 2004.
[12] Goldberg DE. Simple genetic algorithms and the minimal deception problem. In: Genetic algorithms and simulated annealing. London: Pitman; 1987. p. 74–88.
[13] Blum C, Sampels M. Ant colony optimization for FOP shop scheduling: a case study on different pheromone representations. In: Proc Congr Evol Comput (CEC), vol. 2. Los Alamitos, CA, 2002. p. 1558–63.
[14] Montgomery J, Randall M, Hendtlass T. Search bias in constructive metaheuristics and implication for ant colony optimization. In: Lecture notes in computer science. Proceedings of the fourth international workshop, Ant Colony Opt Swarm Intell (ANTS), 2004. p. 391–8.
[15] Blum C, Dorigo M. Search bias in ant colony optimization: on the role of competition-balanced systems. IEEE Trans Evolut Comput 2005;9(2):159–74.