A Parallel Ant Colony Optimization for the Maximum-Weight Clique Problem

Didier El Baz, Mhand Hifi, Lei Wu, Xiaochuan Shi

A Parallel Ant Colony Optimization for the Maximum Weight Clique Problem

Didier El Baz*
*Laboratory LAAS
CNRS, France
elbaz@laas.fr

Mhand Hifi and Lei Wu‡
‡Laboratory EPROAD
Université de Picardie Jules Verne, France
{hifi; lei.wu}@u-picardie.fr

Xiaochuan Shi†
†School of International Software
Wuhan University, China
shixiaochuan@whu.edu.cn

***Abstract*—In this paper, we proposes a parallel ant colony optimization based heuristic for solving the maximum weight clique problem, which is a variation of the classical maximum clique problem. The proposed parallel computing model is based on a cooperation concept with consideration of a multiple ant colonies system. In the cooperation system, we are given a message center and a number of ant colonies. The ant colonies aim at exploring the solution space with their own search strategies. The message center first collects the solution information from the different ant colonies, and then it shares the best solution with them. The performance of the proposed method was evaluated on a set of the standard benchmark instances from literature. The obtained results are compared to those reached by the Cplex solver and the best solutions reported in the literature. As shown from the experimental results, encouraging results have been obtained.**

***Keywords*-Parallel algorithms, weight clique, ant colony, cooperative.**

I. Introduction

In this paper, we present a cooperative parallel optimization model for solving the Maximum Weight Clique Problem (MWCP). The MWCP is a variation of the classical maximum clique problem, which has wide-ranging application in the social network issues. In MWCP, we are given an undirected graph $G = (V, E)$, where $V = \{1, \ldots, n\}$ presents the vertex set and $E \subset V \times V$ presents the edge set. A clique C of G is a subset of the vertex set V, where every two vertices in V are pairwise adjacent, i.e., $\forall\ i, j \in C$, $(i, j) \in E$. For the weighted case, each vertex $i \in V$ is associated with a positive weight, i.e., $\forall\ i \in V,\ w_i > 0$. The objective of MWCP is to determine a clique C^* of maximum weight. Mathematically, MWCP can be written as the following integer linear program:

$$\begin{array}{lll} (P_{MWCP}) & \max \sum_{i \in V} w_i x_i & \\ s.t. & x_i + x_j \leq 1 & \forall\ (i,j) \notin E \\ & x_i \in \{0,1\} & \forall\ i \in V. \end{array} \quad (1)$$

The decision variable x_i ($\forall i \in V$) is equal to 1 if the vertex i is included in the clique; otherwise $x_i = 0$. Equations (1) represents the disjunctive constraints ensuring that the nonadjacent vertices cannot be included in the same clique. Without the loss of generality, we assume that all input data w_i ($\forall i \in V$) are strictly positive integers.

The remainder of the paper is organized as follows. Section 2 reviews some previous works on MWCP and the parallel computing. Section 3 describes the principle of the sequential version of the proposed approach while section 4 states the parallel one. In section 5, the performance of the proposed approach is evaluated on a set of benchmark instances. Finally, section 6 summarizes the contents of the paper.

II. Related works

To the best of our knowledge, few exact methods have been elaborated for the MWCP (see e.g. Babel [1] and Ostergard [15]). In order to solve the large scale data of the MWCP, the exacts methods are always time consuming. Unlike the exact method, which ensures the optimality of the provided solution, the metaheuristic only aims at finding the high quality solution with a reasonable computational effort. For instance, a number of approximate methods have been proposed to solve the large MWCP (see e.g., Pullan [16], Mannino and Stefanutti [14], Busygin [5], Bomze *et al.* [2] and Wu *et al.* [17]).

In recent research, parallel metaheuristics aroused great interest in combinatorial optimization. Most of these approches are based on the execution of a (or a set of) sequential metaheuristic(s) in parallel: 1) with different parameters settings; 2) with different starting solutions and 3) on a series of sub-problems after the application of an exact or an approximate decomposition technique. Among the pioneering approaches, we cite here, a warehouse cooperative parallel meta-heuristic (see Bouthilliera and Crainicb [3]), a cooperative parallel tabu search (see James *et al.* [11]), a parallel multi-start tabu search (see Czapiński [7]), a parallel dynamic programming method for knapsack problems (see Boyer *et al.* [4]), a parallel ant colony optimization on graphics processing units (see Delévacq *et al.* [8]) and a parallel large neighborhood search (see Hifi *et al.* [13]).

Ant Colony optimization (ACO), originally proposed by Colorni *et al.* [6], is a well-known swarm intelligence method, which achieved great success for solving combinatorial optimization problems, even when tackling certain

optimization problems with high complexity. The first parallel ant colony optimization conception was also suggested by Dorigo [10] in the consideration of high-performance parallel computing. A recent survey on parallel ant colony optimization was proposed by Pedemonte *et al.* [9]. They introduced a new taxonomy for classifying the available parallel ACO methods from literature and provided a detailed survey on parallel ACO implementations.

According to results from previous works, the imbedding of the cooperative procedure into the parallel model could contribute to the further reinforcement of ACO. In this work, we are particularly interested in developing a Multiple Ant Colonies System (MACS) (see Pedemonte *et al.* [9]) for solving large scale instances of MWCP. MACS is based on constructing a cooperation system among the different ant colonies, where each colony applies its own strategy to explore the solution space. Once a colony finds a feasible solution, we apply a multi-start tabu search, which was proposed by Wu *et al.* [17], as a random local search to improve the current solution. The proposed MACS model is based on using Message Passing Interface (MPI) which enables us to build a flexible message passing model of parallel programming. More precisely, the proposed parallel program related to MACS consists of gathering data from the different parallel computing units.

III. Ant Colony Optimization

This section discusses the main principle of the sequential ACO, which will be imbedded into MACS (i.e., a parallel model of ACO). The sequential ACO consist of the simulation of the behavior of a single ant colony. We recall that ACO is a simple and efficient swarm intelligence metaheuristic. It has been first introduced in Colorni *et al.* [6] for approximately solving the traveling salesman problem. ACO's principle is based on the observation of real ants which are able to find the shortest path using the pheromone trails deposited by other ants. Generally, an ACO algorithm consists of two phases. In the first phase, one has to determine an auto-updating system which directs the ants towards interesting solutions. As shown in Colorni *et al.* [6], after a sufficient number of iterations, the most of ants find the same solution. In the second phase, one usually applies several local improvement strategies to enhance the solution yielded in the first phase. In this work, we apply the Multi-start Tabu Search (MTS) proposed by Wu *et al.* [17] as a random local improvement strategy. In other words, the second phase can be considered as the fact that one assigns the ants an useful tool in order to increase its capacity.

The first phase of an ACO algorithm is generally composed of two components: a path building strategy and a pheromone updating strategy. The goal of the path building strategy is to determine an available solution in a stepwise way. The pheromone updating strategy contains two keys: the enhancement and the evaporation of pheromone. On the one hand, the enhancement ensures that the more interesting the path is, the more the ants tend to move onto such a path. On the other hand, in order to avoid that the search procedure converges quickly to a local optima, the pheromone trail evaporates with the time passing.

Algorithm 1 : Ant Colony Optimization Heuristic (ACOH).

Input: An instance of P_{WMCP}.
Output: $C^\star$, a local optimum of P_{WMCP}.

1: Set $C^\star$ as a starting feasible solution, where all variables are assigned to 0;
2: **while** the maximum iteration ACOH_{ite} is not reached **do**
3: Apply a random procedure to find a feasible solution of P_{WMCP}, noted by C;
4: Apply MTS to improve C;
5: Update $C^\star$ with the best solution at hand;
6: Update pheromone trails according to $C^\star$ and C;
7: **end while**
8: **return** $S^\star$;

Algorithm 1 summarizes the main steps for performing ACOH. The main loop (step 2-7) of ACOH simulates how the ants belonging to the same colony cooperate to find a feasible clique. Step 3 applies a stepwise procedure to yield a feasible clique, where, at each time, it randomly selects a vertex to insert into the current clique. The random choice respects to the probability distribution characterized by the weights of the vertices and the amount of pheromone deposited for a state transition. The state transition considered in ACOH is simple, let s_{i1} (resp. s_{i0}) be the current state of the vertex i ($\forall\ i \in V$) such that the vertex i is included in (resp. exclude from) the current clique C. Furthermore, let τ_{i1}^t be the amount of pheromone for the state transition such that the vertex i is on the state s_{i1} during the t-th iteration. In order to establish the probability distribution of choosing a vertex from all available choices, the probability of inserting the vertex i into the current clique at the t-th iteration can be defined as follows:

$$p_{i1}^t = \frac{w_i^\alpha + \tau_{i1}^t}{\sum_{j \in V \ and \ j \notin C} (w_j^\alpha + \tau_{j1}^t)}, \quad \forall\ i \in V.$$

The parameter α is used to adjust the impact of the weight of vertices on the computation of the probability. Without the lose of generality, one has $p_{i0}^t = 1 - p_{i1}^t$. The building procedure stops when no more vertex can be inserted into the clique. After using MTS to improve the clique previously determined (cf., step 4), the probability of selecting the vertices is dynamically updated at the end of each iteration. The pheromone updating strategy generally depends on the local optima obtained from previous iterations and the number of iterations already completed. Formally, at the end of the t-th iteration, the amount of pheromone of the vertex

i on the state s_{i1} is modified according to the following equality:

$$\tau_{i1}^{t+1} = (1-\rho)\cdot\tau_{i1}^{t} + \Delta_{i1}^{t}, \quad \forall\ i \in V,$$

where $0 < \rho < 1$ is the evaporation parameter and Δ_{i1}^{t} denotes the reinforcement value for the pheromone of the vertex i on the state s_{i1} at the t-th iteration. Note that the evaporation parameter ρ adjusts the diversity of the search procedure used in ACO (i.e., the step 3) whereas the reinforcement value Δ_{i1}^{t} adjusts its convergence. However, the right choice of both parameters can direct the ant system towards good local optima.

$$\Delta_{i1}^{t} = \begin{cases} f(u^t) & \text{if vertex } i \text{ is on } s_{i1} \text{ at the } t\text{-th iteration} \\ 0 & \text{otherwise,} \end{cases}$$

where u^t (resp. $u^\star$) is the current objective value (resp. the best objective value at hand) and $f(.)$ is a function of u^t such that:

$$f(u^t) = 0.1 \times (\frac{u^t}{u^\star})^{\lambda}. \tag{2}$$

The parameter λ is used with the consideration that the better solution is, the more pheromones are deposited on the corresponding states. Finally, ACOH (cf., Algorithm 1) exits with the best solution when the maximum iteration is reached.

IV. Parallel Ant Colony Optimization

Parallel programming is a technology of computing which carries out the calculation by calling simultaneously computing units. In this section, we introduce a Parallel ACO Heuristic (PACOH) for solving WMCP. The proposed model is designed by using Message Passing Interface (MPI), which addresses primarily the message-passing parallel programming model. In MPI, the data is moved from the address space of one process to that of another process through cooperative operations on each process. The main advantages of MPI is the ease of use, in addition, it enables us to build a portable, efficient, and flexible message passing model of parallel programming.

In this section, we describe the cooperation mechanism used by the proposed parallel algorithm. Indeed, as previously indicated, the core of ACO is composed of two phases: *the auto-updating system* and *local improvement procedure*. As already discussed in section 3, the auto-updating system depends: 1) the path building strategy and 2) the pheromone updating strategy. As shown in Wu *et al.* [17], the application of different stating solutions has a strong impact on the performance of MTS. The main concept of PACOH is based on the simulation a multiple ant colonies system, where PACOH performs simultaneously ACOH (Algorithm 1) in parallel. We assume that the multiple ant colonies system has a message center and a number of ant colonies. In MACS, the message center is responsible for the collection and the distribution of the feedback of different ant colonies. For every ant colony, it first constructs a solution according its individual behavior. Next, it sends its best solution to the message center. Then, the message center choses the best one among the solutions received from all ant colonies. Through the use of the solution sent by the message center, each ant colony updates its own pheromone trails. According the new pheromone trails, each ant colony recalls ACOH to generate new solutions. Finally, the message center stops the global cooperative search procedure when the runtime limit of PACOH is met.

Algorithm 2 : Parallel Ant Colony Optimization Heuristic (PACOH)

Input: A instance of P_{WMCP};
Output: $C^\star$, a local optimum of P_{WMCP};

1: Call $nbp{+}1$ threads and set thread 0 as the root (message center);
2: **while** the time-limit PACOH_{time} is not met **do**
3: For all threads (ant colonies), apply ACOH to find local optima and send them to the root;
4: Update $C^\star$ according to the solutions returned by the threads;
5: The root sends $C^\star$ to all threads;
6: For all threads, update their own pheromone trails according to the solution sent by the root;
7: **end while**
8: **return** $C^\star$.

In algorithm 2, we introduce the main steps of PACOH. The algorithm starts by initializing nbp+1 threads (ant colonies) (cf., step 1), where the thread 0 presents the message center (the root) and the other nbp threads presents the different ant colonies. The main loop (steps 2-7) performs the cooperative search among the different ant colonies. The teamwork consists of the application of ACOH for each ant colony (cf., step 3), the sharing of the best local optima (cf., step 5) and the update of the MACS, i.e., the pheromone trails of each colony (cf., step cp:8). Step 4 stores the best solution in the root. PACOH exits with the best local optimum until the time-limit PACOH_{time} is met.

V. Computational results

In this section, the proposed Parallel Ant Colony Optimization Heuristic (PACOH) was evaluated on the benchmark instances from literature. The test sets for WMCP were generated from the DIMACS benchmark set proposed by Johnson and Trick [12]. The considered test set for WMCP was also used in Pullan [16] and Wu *et al.* [17]. The DIMACS-based benchmark set for WMCP contains either randomly generated graphs or graphs whose maximum clique has been hidden by incorporating low-degree vertices. The number of vertices (resp. edges) of these instances range from 50 to 3300 (resp. 1000 to 5000000) In order to display the performance of PACOH, the obtained results

were compared with those provided by Cplex solver and the best solutions from literature. The proposed parallel algorithm was coded in C++ using MPI library and run on a machine with Intel Xeon, 2×3.06 Ghz with 6-Core.

PACOH is performed with some parameters. According to the results obtained after different trials, the set of values chosen in our experiment ensures the best performance for PACOH on the considered test set. For each thread, the parameters used in ACOH are set as follows: α is randomly generated from 0.8 to 1, ρ takes a random value in $[0.96, 0.98]$, $\lambda = 10$ and $\text{ACOH}_{ite} = 50$. For the root, the parameter used in PACOH are set as follows: $nbp = 10$ and $\text{PACOH}_{time} = 1000$ seconds (i.e., the total CPU runtime consumed by each active thread is limited to 1000 seconds).

Table I reports the best objective values from literature (cf., Pullan [16] and Wu *et al.* [17]), the best objective values provided by the Cplex (version 12.6) in 3600 seconds and the best solution provided by PACOH with the previously discussed parameter setting. Column 1 displays the name of each instance. The best objective values from literature are shown in column 2. Columns 3-4 (resp. columns 5-6) display the best objective values provided by Cplex (resp. PACOH) and their execution time.

As shown in Table I, within the considered runtime limit the Cplex was able to prove the optimality of 44 solutions over 65. For the other 21 hard instances, the Cplex found a new best solution on MANN_a81 (i.e., 111379) in 3600 seconds. However, for the rest of the cases, the Cplex performs worse than the considered algorithms. We can observe that the Cplex even failed to find a feasible solution on p_hat1500-1 in 3600 seconds.

From Table I, we can also observe that, PACOH outperforms the Cplex in 64 cases over 65, where PACOH finds 20 solutions strictly better than the Cplex. When comparing PACOH with the best solutions from literature, PACOH match the best solutions in 63 cases.

VI. Conclusion

In this paper, we proposed a parallel algorithm based upon ant colony optimization for solving the maximum weight clique problem. The proposed approach was designed using message passing interface. Ant colony optimization-based heuristic was introduced in the parallel model in order to provide high quality solutions. The performance of the proposed method was evaluated on the set of the standard benchmark instances of the literature. The obtained results were compared to those provided by the Cplex solver and to those obtained by one of the best methods from literature. As shown in the experimental results, the proposed model was able to provide high quality solutions in most of cases.

References

[1] L. Babel. *A fast algorithm for the maximum weight clique problem.* Computing, Vol. 52, pp. 31-38, 1994.

[2] I.M. Bomze, M. Pelillo and V. Stix. *Approximating the maximum weight clique using replicator dynamics.* IEEE Transactions on Neural Networks, Vol. 11, pp. 1228-1241, 2000.

[3] A. Le Bouthilliera and T.G. Crainicb. *A cooperative parallel meta-heuristic for the vehicle routing problem with time windows.* Computers & Operations Research, Vol. 32, pp. 1685-1708, 2005.

[4] V. Boyer, D. El-Baz and M. Elkihel. *Solving knapsack problems on GPU.* Computers & Operations Research, Vol. 39(1), pp. 42-47, 2012.

[5] S. Busygin. *A new trust region technique for the maximum weight clique problem.* Discrete Applied Mathematics, Vol. 154, pp. 2080-2096, 2006.

[6] A. Colorni and M. Dorigo and V. Maniezzo. *Distributed optimization by ant colonies.* In F. Varlea & P. Bourgine (Eds.), Proceedings of the first european conference on artificial life, pp. 134-142, 1991.

[7] M. Czapiński. *An effective parallel multistart tabu search for quadratic assignment problem on CUDA platform.* Journal of Parallel and Distributed Computing, Vol. 73(11), pp. 1461-1468, 2012.

[8] A. Delévacq, P. Delisle, M. Gravel and M.Krajecki. *Parallel ant colony optimization on graphics processing units.* Journal of Parallel and Distributed Computing, Vol. 73(1), pp. 52-61, 2013.

[9] M. Pedemonte, S. Nesmachnow and H. Cancela. *A survey on parallel ant colony optimization.* Applied Soft Computing, Vol. 11, pp. 5181-5197, 2011.

[10] M. Dorigo. *Optimization, learning and natural algorithms.* Ph.D. thesis, Politecnico di Milano, Italy, 1992.

[11] T. James, C. Rego and F. Glover. *A cooperative parallel tabu search algorithm for the quadratic assignment problem.* European Journal of Operational Research, Vol. 195(3), pp. 810-826, 2009.

[12] D. S. Johnson and M. A. Trick. (1996). *Cliques, coloring, and satisfiability: second DIMACS Implementation Challenge.* DIMACS series in discrete mathematics and theoretical computer science, Vol. 26, 1996.

[13] M. Hifi, S. Negre, T. Saadi, S. Saleh and L. Wu. *A parallel large neighborhood search-based heuristic for the disjunctively constrained knapsack problem.* IEEE 28th International Parallel & Distributed Processing Symposium Workshops, 2014.

[14] C. Mannino and E. Stefanutti. *An augmentation algorithm for the maximum weighted stable set problem.* Computational Optimization and Applications, Vol. 14, pp. 367-381, 1999.

[15] P. R. J. Ostergard. *A new algorithm for the maximum weight clique problem.* Nordic Journal of Computing, Vol. 8, pp. 424-436, 2001.

[16] W. Pullan. *Approximating the maximum vertex/edge weighted clique using local search.* Journal of Heuristics, Vol. 14(2), pp. 117-134, 2008.

[17] Q. Wu, J. Hao and F. Glover. *Multi-neighborhood tabu search for the maximum weight clique problem.* Annals of Operations Research, Vol. 196, pp. 611?634, 2012.

	Best	Cplex		PACOH	
Inst.	OV	OV	CPU	OV_{10}	CPU
brock200_1	2821*	2821	70.8	2821	1000
brock200_2	1428*	1428	32.2	1428	1000
brock200_3	2062*	2062	25.3	2062	1000
brock200_4	2107*	2107	75.3	2107	1000
brock400_1	3422	3164	3600	3422	1000
brock400_2	3350	2991	3600	3350	1000
brock400_3	3471	3300	3600	3471	1000
brock400_4	3626	3223	3600	3626	1000
brock800_1	3121	2713	3600	3121	1000
brock800_2	3043	2464	3600	3043	1000
brock800_3	3076	2462	3600	3076	1000
brock800_4	2971	2695	3600	2971	1000
c-fat200-1	1284*	1284	15.7	1284	1000
c-fat200-2	2411*	2411	10.2	2411	1000
c-fat200-5	5887*	5887	27.1	5887	1000
c-fat500-1	1354*	1354	198.5	1354	1000
c-fat500-2	2628*	2628	184.2	2628	1000
c-fat500-5	5841*	5841	65.7	5841	1000
c-fat500-10	11586*	11586	45.0	11586	1000
hamming6-2	1072*	1072	0.005	1072	1000
hamming6-4	134 *	134	0.14	134	1000
hamming8-2	10976*	10976	0.01	10976	1000
hamming8-4	1472*	1472	3.5	1472	1000
hamming10-2	50512*	50512	0.06	50512	1000
hamming10-4	5129	5029	3600	5129	1000
johnson8-2-4	66*	66	0.004	66	1000
johnson8-4-4	511 *	511	0.01	511	1000
johnson16-2-4	548 *	548	0.01	548	1000
johnson32-2-4	2033*	2033	0.07	2033	1000
keller4	1153*	1153	1.9	1153	1000
keller5	3317*	3317	1781	3317	1000
keller6	8062	6954	3600	8062	1000
MANN_a9	372 *	372	0.029	372	1000
MANN_a27	12283*	12283	3.44	12273	1000
MANN_a45	34265*	34265	126.9	34192	1000
MANN_a81	111128	111379	3600	111116	1000
p_hat300-1	1057*	1057	78.1	1057	1000
p_hat300-2	2487*	2487	81.4	2487	1000
p_hat300-3	3774*	3774	231.1	3774	1000
p_hat500-1	1231*	1231	1856	1231	1000
p_hat500-2	3920*	3920	2353	3920	1000
p_hat500-3	5375	5356	3600	5375	1000
p_hat700-1	1441	1372	3600	1441	1000
p_hat700-2	5290	5290	3600	5290	1000
p_hat700-3	7565	7341	3600	7565	1000
p_hat1000-1	1514	1483	3600	1514	1000
p_hat1000-2	5777	5550	3600	5777	1000
p_hat1000-3	8111	7610	3600	8111	1000
p_hat1500-1	1619	-	3600	1619	1000
p_hat1500-2	7360	5888	3600	7360	1000
p_hat1500-3	10321	9756	3600	10321	1000
san200_0.7_1	3370*	3370	1.9	3370	1000
san200_0.7_2	2422*	2422	1.04	2422	1000
san200_0.9_1	6825*	6825	0.54	6825	1000
san200_0.9_2	6082*	6082	3.77	6082	1000
san200_0.9_3	4748*	4748	19.3	4748	1000
san400_0.5_1	1455*	1455	21.4	1455	1000
san400_0.7_1	3941*	3941	87.4	3941	1000
san400_0.7_2	3110*	3110	560.2	3110	1000
san400_0.7_3	2771*	2771	330.1	2771	1000
san400_0.9_1	9776*	9776	94.7	9776	1000
san1000	1716*	1716	1796	1716	1000
sanr200_0.7	2325*	2325	77.4	2325	1000
sanr200_0.9	5126*	5126	30.7	5126	1000
sanr400_0.9	1835*	1835	2460	1835	1000
sanr400_0.7	2992	2757	3600	2992	1000

Table I
PERFORMANCE OF PACOH ON THE STANDARD BENCHMARK INSTANCES OF THE LITERATURE.