Ant Colony Optimization based Simultaneous Task Allocation and Path Planning of Autonomous Vehicles

Conference Paper · July 2006

DOI: 10.1109/ICCIS.2006.252349 · Source: IEEE Xplore

4 authors, including:

Asela K Kulatunga
University of Kentucky USA
117 PUBLICATIONS **1,026** CITATIONS

SEE PROFILE

Sarath Siyambalapitiya
University of Peradeniya
11 PUBLICATIONS **59** CITATIONS

SEE PROFILE

Ant Colony Optimization based Simultaneous Task Allocation and Path Planning of Autonomous Vehicles

A. K. Kulatunga, D. K. Liu, G. Dissanayake
ARC Centre of Excellence in Autonomous Systems
Faculty of Engineering, University of Technology, Sydney
akula@eng.uts.edu.au, dkliu@eng.uts.edu.au, gdissa@eng.uts.edu.au

S.B. Siyambalapitiya
Department of Engineering Mathematics, Faculty of Engineering, University of Peradeniya, Sri Lanka
sbs@pdn.ac.lk

***Abstract*— This paper applies a meta-heuristic based Ant Colony Optimization (ACO) technique for simultaneous task allocation and path planning of Automated Guided Vehicles (AGV) in material handling. ACO algorithm allocates tasks to AGVs based on collision free path obtained by a proposed path and motion planning algorithm. The validity of this approach is investigated by applying it to different task and AGV combinations which have different initial settings. For small combinations, i.e. small number of tasks and vehicles, the quality of the ACO solution is compared against the optimal results obtained from exhaustive search mechanism. This approach has shown near optimal results. For larger combinations, ACO solutions are compared with Simulated Annealing algorithm which is another commonly used meta-heuristic approach. The results show that ACO solutions have slightly better performance than that of Simulated Annealing algorithm.**

***Keywords*—Ant colony optimization, task allocation, path planning, autonomous vehicles**

I. INTRODUCTION

Coordination of multiple autonomous vehicles such as automated guided vehicles and straddle carriers in fully or semi automated material handling environments becomes complex due to the combination of Task Allocation (TA) and Path Planning (PP) processes. TA is the process of allocating a task to a vehicle based on an optimization criterion. PP is the process of deciding the shortest or the most suitable path depending on the situation. Owing to their complexity, TA and PP are categorized as NP-hard problems[1]. The coordination problems get even more complex when they have to deal with collision avoidance issues. Collision issues generally occur in path planning, when multiple vehicles operate in congested environments and restrained environments caused by AGVs traveling in opposite directions on single lanes.

Solving a multi-autonomous vehicle coordination problem amounts to making discrete choices such that an optimal solution is found among a finite or a countable number of alternatives. It is impossible to find an optimal solution without the use of an essentially enumerative algorithm, and this increases computational time exponentially with problem size. Branch and bound or dynamic programming algorithms are often used to find solutions of such problem with the help of problem specific information to reduce search space[1]. However, his is only valid to specific type of problems. Many variations of local search algorithms for solving TA problems have been proposed and investigated. Since the quality of solutions obtained by local search algorithms strongly depends on initial conditions, these algorithms have the potential to perform poorly for some set of initial conditions. A heuristic method, which involves trial and error and some contemplated intuition, can produce an approximate solution to a NP hard problem. Since it cannot find the exact optimal solution, there is a clearly trade-off between the computational cost in finding the near optimal solutions within reasonable time and the precision of solutions. Generally, applicable heuristics and evolutionary algorithms are applied for wide range of combinatorial problems, and they have gained much attention in recent years. Ant colony optimization[2, 3] (ACO) is one of the generally applicable techniques used recently to solve NP-hard type problems. After ACO algorithm was applied to well known NP-hard traveling salesman problem[4], it has been used for different NP-hard problems such as vehicle routing[5], quadratic assignment[6], job shop scheduling [7] etc.

On the other hand, different aspects of multiple autonomous vehicle coordination in material handling have been extensively studied in the literature. However, only few of them attempted to combine the TA, PP and collision avoidance due to the complexity of the problem. The container handling method investigated by Meersmans [8] combines the task allocation process with path planning. A heuristic search based Beam Search algorithm is applied. Collision among vehicles are overcome by a loop (uni-directional) based system. Hence, the path planning and collision avoidance problem can be handled easily compared to bi-directional path network environments where a path allows a vehicle to move in two directions along the path. This method results in more empty travels and waiting time, and low efficiency. Bish et al. [9] modeled a transportation problem in container terminals which uses the assumptions of constant vehicle velocity and uni-directional vehicle movement. The congestion in the paths is not taken into account. An algorithm proposed by Koo et al. [10] does fleet sizing and vehicle routing for container

1-4244-0023-6/06/$20.00 ©2006 IEEE

transportation based on Tabu search method in static environment. This algorithm initially starts up with lower bound of fleet size and increases the size till the makespan criteria are satisfied. The vehicle travel time between each segment is fixed and the vehicle speed is assumed to be constant. The deadlock and shortest path search issues are not taken into account in this algorithm. The work done by Qiu et al. [11] gives collision free routing for AGV's in bi-directional path layout. However the path topologies are created to suit a particular system, and scheduling and routing are not done continuously. The vehicle speed is also assumed to be constant in this work. Simultaneous machine and AGV scheduling in flexible manufacturing system is introduced by Ulusoy et al. [12] in order to minimize the makespan, but the routing aspects are not taken into account in this research.

Grunow et al. [13] discussed a dispatching method for multi-load AGV's in a fully automated container terminal. A flexible priority rule based approach is proposed and compared to an alternative mix integer programming (MIP) formulation for several scenarios. Bose et al.[14] compared a vehicle dispatching policy for a seaport container terminal to other different dispatch policies in allocation of straddle carriers to gantry cranes. Genetic algorithm was applied to improve the solution quality of the allocation, and the vehicle traveling speed was assumed as constant.

TA, PP and even collision issues are handled separately or one after another in most of the current research works. This paper is to solve the complex simultaneous TA, PP and collision avoidance problem for multiple vehicles in material handling environments using ACO algorithm in conjunction with a proposed path and motion planning algorithm. The rest of the paper is organized as follows: section II formalizes the problem; section III introduces the ACO based TA and PP approach. Section IV presents simulations and results followed by conclusion and discussion in Section V.

II. MOLDELLING OF THE PROBLEM

A. Simulated environment

When autonomous vehicles are used in logistics (cargo and material handing, etc.), the task allocation, path and motion planning, and collision avoidance of autonomous vehicles are trying to maximize the productivity of the whole systems in a period of time, for example, one day, one week, etc. The productivity is significantly affected by a number of factors such as the environment, vehicle control, vehicle path, collision avoidance strategy, bottleneck area, etc. "Fig.1" shows an example environment in which autonomous vehicles are supposed to deliver and pick up/collect materials from and to any place in the environment.

A network of nodes represents the environment where vehicles can travel. The cross points in the map represent the nodes where the vehicles can reach, and lines are the connections among nodes and the paths the vehicles have to follow. An autonomous vehicle is supposed to move from node to node following the link between nodes. Traffic congestion is unavoidable because of several bottleneck areas in this environment, which significantly affects the productivity.

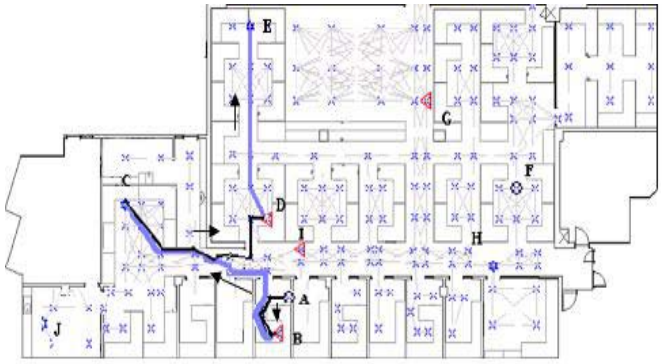

Figure 1. Simulated material handling environment

B. Mathematical formulation

An AGV travels from their start location to the destination on guided paths which contain nodes and links. Each of these paths can be divided into a number of path segments (links). On each of these segments, in order to avoid collissions, AGVs travel on varying speeds set by a SiPaMoP approach [16] (which will be explianed in a later section). Two different maximum speeds are set for empty and loaded AGV's. Task completing time has two components: the travel time from the current location of the vehicle to the start node of the task (trancient time) and the travel time from the task's start node to the destination node (task time).

The methamatical model of the TA problem can de derived as follows:

Number of available tasks: J

Number of available vehicles: R

Available task list: $T = [T_1, T_2, T_3, \ldots\ldots\ldots\ldots T_J]$

Availabe vehicles: $V = [V_1, V_2, V_3, \ldots\ldots\ldots\ldots V_R]$

Maximum empty AGV speed: V_E

Maximum loaded AGV speed: V_L

Loading time of AGV: t_L

Unloading time of AGV: t_U

Total completion time of T_j: TCT_j

Total traveling time of vehicle V_i: TTT_i

Number of tasks allocated to vehicle V_i: N_i

Start time of task T_h: ts_h

Start time of task T_h of j^{th} vehicle: ts_{hj}

Assuming that vehicle V_i is allocated to task T_j and path segments generated by the SiPaMoP algorithm to reach the task T_j's start node is PR_{ij} (e.g., PS_{AB}-path selected to travel from A to B shown in "Figure 1") or from the destination node of the previous task to the start node of the current task (e.g., PS_{CD} – path selected to travel from C to D) and the path selected to travel from task start node to desination is PP_{ij} (e.g., PS_{BC} and PS_{DE} – path selected to travel from B to C and D to E, respectively). PR_{ij} and PP_{ij} contain k_R and k_P path segments respectively.

Therefore, total completion time of the task T_j by vehicle V_i can be calculated as follows;

$$TCT_{ij} = \sum_{k=1}^{k_R} \left(\frac{PR_{ijk}}{V_E}\right) + t_L + \sum_{k=1}^{k_P} \left(\frac{PP_{ijk}}{V_L}\right) + t_U \quad (1)$$

If vehicle V_i completes N_i number of tasks, total traveling time of the respective vehicle will be;

$$TTT_i = \sum_{n=1}^{N_i} [TCT_{in}] \quad (2)$$

Therefore, makespan (MS) of the schedule is;

$$MS = \underset{v=1}{\overset{R}{MAX}} (TTT_v) \quad (3)$$

Since one AGV can perform one task at a time, start time of task j +1 by V_i ($ts_{i(j+1)}$) should be always greater than the start time of task j (ts_{ij}) done by the same AGV.

$$ts_{ij} < ts_{i(j+1)} \quad (4)$$

In addition, start time of the first task by each AGV is assumed to be the same.

III. ACO BASED SIMULTANEOUS TASK ALLOCATION AND PATH PLANNING

A. ACO algorithm

Ant colony algorithms were inspired by the observation of real ant colonies. Ants are social insects that live in colonies and whose behavior is directed more to the survival of the colony as a whole. Social insects have captured the attention of many scientists because of the structural level their colonies can achieve, especially when compared to the relative simplicity of the colony's individuals. An important and interesting behavior of ant colonies is their foraging behavior, and, in particular, how ants can find the shortest paths between food sources and their nest. It was found that real ants are able to communicate information concerning food sources via aromatic essence called pheromone. While they are moving real ants lay down pheromone in a quantity that depends on the quality of the food source discovered. Other ants, observing the pheromone trail, are attracted to follow it. Therefore, path will be reinforced and will attract more ants. This behavioral mechanism can be used to solve combinatorial optimization problems by simulating with artificial ants searching the solution space similar to real ants searching their environment. Additionally, the objective values correspond to quality of the searched food, an adaptive memory is equivalent of the pheromone trails. Furthermore, to guide their search through the set of feasible solutions, the artificial ants are equipped with a local heuristic function.

Ant colony algorithms were first proposed by Dorigo and colleagues [2, 3] as a multi-agent approach to difficult combinatorial optimization problems such as the traveling salesman problem and the quadratic assignment problem. There is currently much ongoing activity in the scientific community to extend and apply ant-based algorithms to many different discrete optimization problems [15]. Recent applications cover problems such as vehicle routing[5], job shop scheduling[7], quadratic assignment problem [6], and so on.

When an ant algorithm is applied to task allocation for multi-autonomous vehicles, an ant represents a vehicle and starts from its start node (depot). The first task of each ant is allocated randomly in our model, then each any selects next task from the available task list until all tasks are selected. For the selection of tasks that are not yet allocated, two aspects are taken into account: how good the choice of that task in previous runs was and how promising the choice of that task in general is. The first information is stored in the pheromone trails τ_{ij} associated with each task-vehicle couple, whereas the second is the local heuristic function mentioned above. This measure of desirability, called visibility, is denoted by η_{ij}.

Each ant's total traveling time is calculated based on its selected tasks, planned routes and travel speeds. The maximum traveling time out of all the ants is considered as the makespan (MS) of overall schedule.

In ant colony optimization algorithm (ACO), each ant selects its next task based on the probability of selection. With feasible task list $T_L = [T_j \in T_L: T_j$ is a task feasible to be allocated], T_j is to be allocated after task T_i:

$$P_{ij} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{u \in T_L}[\tau_{iu}]^{\alpha}[\eta_{iu}]^{\beta}} if \rightarrow Ti \in T_L \\ 0 \rightarrow Otherwise \end{cases} \quad (5)$$

The probability distribution is biased by the parameters α and β that determine the relative influence of the trails and the visibility, respectively. Where τ_{ij} is equal to the amount of pheromone of selecting task 'j' after task 'i'. The value of η_{ij} is defined as the inverse of the complete traveling time which is the sum of the transient time for the vehicle from its current position to the start node of the task, and the task processing time (the traveling time of the vehicle from the task start node to end node). Allocated tasks are removed from the feasible task list.

In order to improve the solutions, the pheromone trails of ants must be updated to reflect the ants' performance and the quality of the solutions found. This update is a key element to the adaptive learning technique of ACO and helps to ensure the improvement of subsequent solutions. The update is conducted

by reducing the amount of pheromone elements in the pheromone table of each task-ant combination of the respective schedule in order to simulate the natural evaporation of pheromone and to ensure that no one task-vehicle combination becomes dominant. This is done by following equation:

$$\tau_{ij} = (1-\lambda)\tau_{ij} + \lambda\tau_0 \qquad (6)$$

where λ is a parameter that controls the speed of evaporation and τ_0 is the initial pheromone value assigned. In our algorithm τ_0 is the inverse of the completed tour cost of (total travel time of an ant for a schedule) each ant. The probability calculations are based on travel times "(2)" instead of travel distances used in TSP and vehicle routing problems. The travel time comes from the collision free path planning by the SiPaMoP[16] algorithm. Each ant calculate the probabilities to selects the next task based on "(5)", the task which gives highest probability is selected as the next task to perform. Then task selection opportunity moves to next ant. Each ant selects one task at a time until all the tasks in the feasible list finishes. After all the tasks are allocated, makespan can be calculated and accordingly the best makespan so far can be updated. This process repeats until fixed number of runs being performed.

The flowchart of the ACO algorithm used in simultaneous TA and PP is given in "Fig. 2". The number of ants and the number of tasks available in the task allocation problem defines the size of the pheromone table. Initially, all the elements of the pheromone table are set to be one. The ACO parameters such as α, β, λ, are set to their respective values based on previously selected optimal values. At the start of each run, the first task for each ant is assigned randomly. From that step onwards, each ant select next task based on the probabilities calculated as in "(5)".

In the proposed system, the probability calculations are based on travel times instead of travel distances used in the TSP and the vehicle routing problems. These travel times comes from the collision free paths generated by a SiPaMoP algorithm. Each ant selects the task which gives highest probability. Each ant selects one task at a time until all the tasks in the feasible list are allocated.

B. SiPaMoP algorithm

SiPaMoP algorithm [16] is a collision free path and motion generating mechanism, which operates in nodes based map environment. It generates collision-free paths and movement speed for all vehicles by dynamically changing the vehicle's path or travel time/speed between nodes of the path. It integrates the path planning, collision avoidance and motion planning into a comprehensive model and optimizes the vehicle path and motion by minimizing the completion time of a set of tasks or maximizing the productivity. In this work, its aim is to minimize the travel time of vehicles from their start nodes to the destination nodes and avoid static and moving obstacles (other vehicles operate in the environment) in a dynamic environment. The shortest path search algorithm of the SiPaMoP is done by the Dijkstra algorithm.

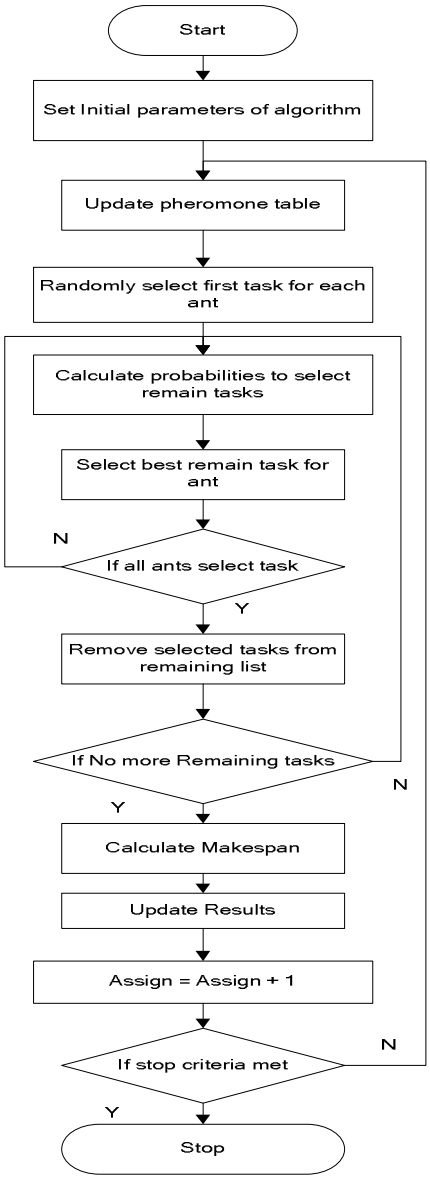

Figure 2. Flow chart of ACO algorithm

By changing the connection weight, the SiPaMoP method is able to vary the coming vehicles' paths and/or speeds in the following four ways:

(a) Keeping the same path as that planned without weight change but changing a vehicle's travel speed. As we assume the paths planned previously have higher priority than the current path being planned, and vehicles are running with their maximum speed when there is no collision on their path, the change in weight will only slow down the coming vehicles.

(b) Waiting until other vehicles passing the link. This happens normally in the bottleneck areas.

(c) Replacing a path by a better one if a shorter path can be found by changing the weight based on current environment conditions;

(d) Combinations of the above three cases.

Clearly, this weight change does not affect the previous planned paths and speeds but will affect the paths and speeds being planed and to be planned. Further information about this approach can be found in [16].

IV. SIMULATIONS AND RESULTS

The ACO algorithm based approach is first applied to small problems where the optimal solution can be obtained within reasonable time using exhaustive search (ES) mechanisms. Each TASK – AGV combination, for example, 8T-4V is a combination of eight tasks and four vehicles, was run with different initial settings, i.e. different tasks, start nodes and tasks destination nodes. An initial setting is called a case in the simulation, for example, 8T-4V-case 1(in table 1). The ACO approach runs a predefined number of times for each case, for example, 6T-2V for 50 runs, 6T-3V for 60 runs and 8T-4V for 75 runs. Following assumptions are considered when performing simulations; (a) Static task allocation problem is considered here, each task has equal priority, (b) All tasks are avialble for allocation at the start of the problem, (c) Tasks loading and unloading times are considerd to be negligible with compared to task times, (d) Speeds of the vehicles when loaded and empty are considered the same, (e) AGV's have capacity to undertake one task at a time. The makespan of each case is compared with optimal value found by exhaustive search mechanism in Table I.

Based on the comparison of ACO approach with optimal values presented in Table I, standard deviation of the ACO solutions is 4.96 while average percentage deviation is 7.89% for 13 simulation cases. In 8T-4V-case1 makespan difference is 14.6 (Mins) and this implies that schedule generated by ACO approach takes 14.6 (Mins) longer than the schedule generated by exhaustive search (ES) method to complete the tasks. Furthermore, each AGV's empty travel time (time taken to reach start nodes of the tasks from their initial nodes) are presented in Table II. This result reveals that ACO approach does more empty travels than the schedule generated by ES by approximately 20(Min).

Next step was to investigate the expandability of the ACO algorithm for larger task-AGV combinations. However, owing to the expensive computational time, ES could not be used for the comparison purposes. Therefore, we use another meta-heuristic approach, simulated annealing (SA), which has been used for many applications in order to compare the ACO approach. Task-AGV combination is expanded up to 40T-10V in the simulations. Makespan, computational time (CPU time) and total empty travel time are evaluated and results are presented in Table III. The results reveal that, makespan obtained by the ACO approach is very similar to the SA algorithm based solution for the larger size problems. In addition, total empty travel time obtained by the ACO and SA approaches comes very close to each other. Although CPU time of the ACO approach is shorter than SA in small size problems, the CPU-time of the ACO approach increases. The Makespan of the ACO approach is better than the SA approach in 10T-5V-S2, 20T-5V-S3. In 30T-10V-S4, ACO approach gives a very close result to the SA approach. In the case of CPU time, the ACO approach taken less computational time than the SA approach in 30T-10V-S4, 10T-5V-S2 and 8T-4V-S1. In 8T-4V-S1 and 30T-10V-S4, the ACO approach has less empty travel time than SA approach. In Table III all the measurements are given in minutes and the runs of the task-AGV combinations from 8T-4V-S1 to 40T-10V-S5 are 75, 100, 150, 200 and 200, respectively.

TABLE I. MAKESPAN COMPARISON OF ACO ALGORITHM WITH OPTIMAL VALUES

Problem	**Makespan (mins)**		
Tasks-AGVs	***Optimal***	***ACO***	***Deviation***
6T-2V-case 1	75.6	80.1	4.5
6T-2V-case 2	55.3	62.8	7.5
6T-2V-case 3	90.9	102.5	11.6
6T-2V-case 4	99.1	109.3	10.2
6T-3V-case 1	54.5	66.6	12.1
6T-3V-case 2	40.7	42.4	1.7
6T-3V-case 3	72.9	72.9	0.0
6T-3V-case 4	68.1	70.8	2.7
8T-4V-case 1	45.3	59.9	14.6
8T-4V-case 2	43.8	49.8	5.9
8T-4V-case 3	60.3	73.0	12.7
8T-4V-case 4	53.4	60.2	6.9
8T-4V-case 5	56.9	64.2	7.2

TABLE II. EMPTY TRAVEL TIME OF 8T-4AGV-CASE1

Method	**AGV 1**	**AGV 2**	**AGV 3**	**AGV 4**	**Total**
ES	19.47	5.48	26.9	22.52	69.17
ACO	30.77	20.9	23.53	14.28	89.48

TABLE III. COMPARISON OF MAKESPAN, CPU TIME AND EMPTY TRAVEL TIMES OF ACO AND SA APPROACHES

Problem	**Makespan**		**CPU_time**		**Empty travel time**	
Tasks-AGV	***ACO***	***SA***	***ACO***	***SA***	***ACO***	***SA***
8T-4V-S1	58.25	55.07	**48.20**	103.59	**87.06**	92.84
10T-5V-S2	**66.93**	73.69	**109.04**	255.28	116.02	113.28
20T-5V-S3	**109.39**	112.57	1031.97	773.22	341.54	311.38
30T-10V-S4	103.20	102.00	**3037.45**	3430.50	**353.05**	442.15
40T-10V-S5	124.90	115.50	6662.63	4908.88	650.07	557.36

The parameters of the ACO and the SA approaches in the simulations were set as: α, β are 3 and 5, respectively, λ (speed of evaporation) is 0.7. Start and stop temperatures of annealing cycle are 100 and 1, respectively, and cooling rate is at 0.9. The hardware and software in the simulations are: Processor Type

of Pentium 4 Hyper threading @ 3.0Ghz (Prescott), and Matlab of V7.0.4.352 (R14) Service Pack 2.

Based on the different cases and simulations investigated, we can say the ACO approach gives good quality results for complex problem of simultaneous TA and PP for multi AGVs. The makespan of the ACO approach for the small size problems falls within 7.89% deviation from the optimal values with considerably less computational time. In the case of large-scale problems, the ACO approach performs as good as the SA approach with better results than SA in some instances. Finally, it has been shown that ACO technique can be used in simultaneous TA and PP of multiple AGVs and it is capable to provide near optimal results within considerably short computational time. Additionally, it is revealed that the ACO approach can be extended to larger size problems.

V. CONCLUSION

Ant colony optimization technique is used for multiple AGV's task allocation and path planning problem in material handling. Comparison of the ACO based approach is done with exhaustive search mechanism to investigate the solution quality for smaller size problems. Simulation results show that the ACO approach can produce near optimal results with much less computation time. Additionally, it is revealed that the ACO approach can be used for lager size of problems and still performs well comparing to other meta-heuristic algorithm such as simulated annealing. Future work intends to improve the solution quality further by using hybrid approach of combining ACO with other techniques.

ACKNOWLEDGMENT

This work is supported in part by the ARC Centre of Excellence programme, funded by the Australian Research Council (ARC) and the New South Wales State Government, Australia. The first author would like to acknowledge Dr. P. Wu of Faculty of Engineering, University of Technology, Sydney for his contributions to this work and the Presidential fund of Sri Lanka for the financial support extended to the first author.

REFERENCES

[1] T. P. Bagchi, "Multiobjective scheduling by Genetic Algorithm by Kluwer Academic Publishers," 1999.

[2] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed Optimization by Ant Colonies," In Proceedings of the First European Conference on Artificial Life (ECAL 91), pp. 134-142, 1991.

[3] M. Dorigo, V. Maniezzo, and A. Colorni, "Positive feedback as a search strategy," Tech. Rep. 91-016 Milan, Italy: Politecnico di Milano, Dipartimento di Elettronica, 1991.

[4] M. Dorigo and L. M. Gambardella, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem," IEEE Transactions on Evolutionary Computation, vol. 1(1), pp. 53-66, 1997.

[5] B. Bullnheimer, R.F. Hartl, and C. Strauss, "Applying the ant system to the vehicle routing problem," in: Metaheuristics: Advances and treands in local search paradgms for optimization, Kluwer, Boston, 1998.

[6] V. Maniezzo, A. Colorni, and M. Dorigo, "The ant system applied to quadratic assignment problem," Technical Report IRIDIA/94-28, Universite Libre de Bruxelles, , 1994.

[7] A. Colorni, M. Dorigo, V. Maniezzo, and M. Trubian, "Ant system for job shop scheduling," Journal of operations research, statistics and computer science, vol. 34(1994), pp. 39, 1994.

[8] P. J. M. Meersmans and A. P. M. Wagelmans, "Dynamic Scheduling of Handling Equipment at Automated Container Terminals " https://ep.eur.nl/handle/1765/137, 2001.

[9] Ebru K. Bish , Frank Y. Chen , Yin Thin Leong, Barry L. Nelson, Jonathan Wing Cheong Ng, and David Simchi-Levi, "Dispatching vehicles in a mega container terminal " OR Spectrum vol. 27, Number 4 pp. 491 - 506 2005.

[10] P. H. Koo, W. S. Lee, and D. W. Jang, "Fleet sizing and vehicle routing for container transportation in a static environment.," OR Spectrum vol. 26(2)to appear, 2004.

[11] L. Qiu, HsuW-J, and "Scheduling of AGVs in a mesh-like path topology :a case study in a container terminal.," Technical Report CAIS-TR-01-35, Nanyang Technological University, School of Computer Engineering, Centre for Advanced Information Systems, 2001.

[12] G. U. Ulusoy, S. F. Sivrikaya, and U. Bilge, "A genetic algorithm approach to the simultaneous scheduling of machines and automated guided vehicles," Computers & Operations Research, vol. 24, pp. 335-351, 1997.

[13] M. Grunow , H. Günther, and M. Lehmann, "Dispatching multi-load AGVs in highly automated seaport container terminals " OR Spectrum vol. 26, Number 2, pp. 211 - 235 2004.

[14] J. Bose, T. Reiners, D. Steenken, S. Voß, and "Vehicle dispatching at seaport container terminals using evolutionary algorithms," In: Sprague R H (ed) Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, DTM-IT, pp 1-10. IEEE, Piscataway, 2000.

[15] M. Dorigo, "Optimization, learning and natural algorithms (in Italian)," Unpublished doctoral dissertation, Politecnico di Milano, Dipartimento di Elettronica, Italy, 1992.

[16] D. K. Liu, X. Wu, A. K. Kulatunga, and G. Dissanayake, "Motion coordination of multiple autonomous vehicles in dynamic and strictly constrained environments", Proceedings of the IEEE International Conference on Cybernetics and Intelligent Systems (CIS 2006), Thailand, 2006