
INDIAN INSTITUTE OF MANAGEMENT KOZHIKODE

Working Paper

**IIMK/WPS/207/QM&OM/2016/19**

October 2016

# Hybridized Ant Colony Algorithm for Convoy Movement Problem

**Alan John Maniamkot** [1]
**P N Ram Kumar** [2]
**R Sridharan** [3]

1 Department of Mechanical Engineering, Indian Institute of Technology Bombay, Mumbai – 400 076, India
2 Associate Professor, Quantitative Methods and Operations Management Area, Indian Institute of Management Kozhikode, IIMK Campus P.O, Kerala – 673570, India
Research Grant/ Project No: SGRP/2014/74
Corresponding author: E-mail: *ram@iimk.ac.in*, Phone: +91- 495 -2809426
3 Department of Mechanical Engineering, National Institute of Technology Calicut, Kozhikode – 673 601, India

# IIMK WORKING PAPER
# Hybridized Ant Colony Algorithm for Convoy Movement Problem

Alan John Maniamkot[1], P N Ram Kumar[2,*], R Sridharan[3]

[1]Department of Mechanical Engineering, Indian Institute of Technology Bombay, Mumbai – 400 076, India
[2]QM & OM Area, Indian Institute of Management Kozhikode, Kozhikode – 673 570, India
[3]Department of Mechanical Engineering, National Institute of Technology Calicut, Kozhikode – 673 601, India
* Corresponding author: E-mail: ram@iimk.ac.in, Phone: +91- 495 -2809426

**Abstract:** Convoy movement problem is the problem of routing and scheduling military convoys across a limited route network while satisfying some strategic constraints. The problem bears lot of similarities with other real-life applications such as scheduling passenger and freight trains along a single line network, scheduling aircraft landings on runways, routing of automated guided vehicles in a FMS environment, handling baggage along a common automated conveyer belt system, to name a few. Being a proven *NP*–complete problem, this problem warrants the usage of meta-heuristics to obtain quick solutions. This work focuses on the development of a hybridized ant colony algorithm that combines local search with ant colony optimization to solve the problem. By testing the methodology on a wide range of hypothetical problem instances, we establish the efficacy and practical relevance of the proposed approach. The importance of using a good seed solution for initializing the trail intensities is analyzed and found that it leads to quicker convergence of the algorithm. The need to hybridize the ant colony algorithm with a local search procedure for obtaining superior results is also demonstrated.

**Keywords:** Military convoys; Ant colony; local search; hybridization; conflicts; metaheuristics.

## 1. Introduction

Convoy movement problem (CMP) is the problem of routing and scheduling military convoys between specific origin and destination pairs across a limited route road/rail network while adhering to some strategic constraints. To better understand the relevance of this problem, we present some background information pertaining to military logistics. In order to undertake missions such as armed conflict, humanitarian relief and peacekeeping, defense establishment would often need to move large number of personnel and equipage from their home bases to the regions of conflict, threat or crisis as swiftly as possible. During the process of military deployment, each unit moves as a *convoy* consisting of fleet of vehicles that must travel nose to tail with a gap of 50–100m between them. Apart from arms and ammunition that are obvious, convoys carry dry rations, fuel, clothing, medicines and personnel involved in support services. Specially designed high capacity transporting vehicles, called as *transporters* are used to carry armored fighting vehicles (AFVs) such as tanks and armored personnel carriers with a view to reduce the physical wear and tear and mechanical failures that might occur to them while moving on the ground.

Any military movement typically happens in one of the two contexts: peacetime and wartime. During peacetime, convoys usually travel in the nights trying to minimize disruptions to civilian traffic while halting *en-route* at pre-decided locations during daytime. However during wartime or crisis situations, convoys from their home bases continue to travel without any halt until they reach their respective destinations (Chardaire *et al*. 2005). Though this problem appears to be the case of simple multiple origin-destination pair shortest path problem, there are two constraints that make the problem significantly different and computationally intractable. They are termed no-crossing and minimum headway constraints. Situations where two convoys cross each other along the same road/rail-link is referred to as *conflict*. Be it peacetime or wartime, conflicts are strictly forbidden as the roads/rail-links used by the convoys may not have the load bearing ability and width to accommodate two convoys at the same time (Lee *et al*. 1996). More importantly, convoys crossing each other are extremely vulnerable to enemy strikes owing to the magnitude of damage that can be inflicted upon. On the similar lines, convoys are not allowed to overtake each other and are expected to maintain *minimum headway* time while traveling along a road/rail link in the same direction. This constraint also helps in preventing accidents and confusion among the personnel. In addition to the no-crossing and minimum headway constraints, convoys are expected to reach their respective destinations on or before their individual due dates and lastly, to prevent convoys from getting exposed to enemy's surveillance for a longer period of time, there are restrictions on the total travel time spent by each convoy while traveling across the network. This problem of routing and scheduling military convoys across a limited route network with an objective to minimize the sum of arrival times of each convoy at its respective destination while satisfying all the aforementioned constraints is known as the convoy movement problem.

With a few modifications, problem instances of CMP can easily give rise to a lot of other practical applications. Scheduling passenger and freight trains along a single line rail network, routing baggage along a common automated conveyer belt system at airports, scheduling aircraft landings on runways, hazardous material transportation, and routing automated guided vehicles (AGVs) in a flexible manufacturing system (FMS) environment are some of the scenarios that share similarities with instances of CMP.

Though smaller instances of CMP can be solved to optimality using commercial solvers, the scope for solving practical and large problem instances using conventional mathematical modeling based approaches is limited owing to its *NP-completeness* (Chardaire *et al*. 2005). Hence, from a practical point of view, it is important to generate routes and schedules for convoys quickly even if it is at the expense of the quality of the solution. This necessitates the development of heuristics/meta-heuristics for solving the CMP. In the operations research literature, there is no dearth of application of heuristics & meta-heuristics for a wide range of optimization problems in the fields of manufacturing, logistics, telecommunications, medicine, power systems, space and defense to name a few (Gonzalez, 2007; El-Ghazali Talbi, 2009). Evolutionary techniques such as Genetic Algorithms (GA), Simulated Annealing (SA), Taboo Search (TS) and Ant Colony Optimization (ACO) are the most widely used search procedures. The key advantage of meta-heuristics over simple heuristics is their adaptability to the problem at hand. With a broad framework, these methodologies are amenable for handling a variety of optimization problems. Though the degree of success varies from case to case, nonetheless, over the years, the techniques have been proven to be quite efficient.

In this study, we propose a hybridized ant colony based metaheuristic approach by combining ant colony algorithm with a local search procedure for solving the convoy movement problem. Ant colony optimization is a class of optimization algorithms based on the pheromone trail laying and foraging behavior of real ants. Dorigo et al. (1996) proposed ACO using the well-known traveling salesman problem. Since then, ACO has been applied to other combinatorial optimization problems such as sequential ordering (Gambardella and Dorigo, 2000), scheduling (Rajendran and Ziegler, 2004), assembly line balancing (Ozbakir *et al.* 2011), vehicle routing problems (M.M.S. Abdulkader *et al*. 2015), quadratic assignment problems (Acan and Unveren, 2015), DNA sequencing (Blum *et al*. 2008), and so on with varying success rates. For a detailed review of ACO applications, interested readers can refer to Dorigo and Blum (2005) and Dorigo and Stutzle (2009). We intend to investigate the suitability of the proposed Hybridized Ant Colony (HAC) algorithm for CMP, in terms of computational time and solution quality, by testing it on a wide range of problem instances. We use both CPLEX based optimal methodology and GA based meta-heuristic procedure reported in the literature to compare and comment on the quality of the results obtained. Also, we analyze the effect of choosing a heuristic seed solution for initializing the pheromone trail intensities and hybridizing the ant colony algorithm with a local search procedure on the overall performance.

The rest of the paper is organized as follows. We review the literature pertaining to CMP and highlight the research gap in Section 2. The proposed hybridized ant colony algorithm with a detailed flowchart is explained in Section 3. The characteristics of the generated test problem instances is presented in Section 4. The computational results are summarized in Section 5 followed by conclusions and scope for further work in Section 6.

## 2. Literature Review

In this section, we review the literature pertaining to the convoy movement problem with specific focus on the solution methodology adopted for solving the problem. To the best of our knowledge, Bovet *et al*. (1991) were the first to introduce the convoy movement problem. They consider the problem of scheduling a collection of military convoys along one single road-link with a pre-specified time-window for every convoy's departure. They explicitly consider forbidding conflicts. The authors propose a mixed integer programming model and a heuristic based on Taboo search procedure. Schank *et al*. (1991) and McKinzie and Barnes (2004) in their review work observe that majority of the strategic military mobility models in vogue use either cumbersome and ineffective classical optimization algorithms or simplistic and ineffective greedy approaches. They advocate the use of advanced computer models to improve the fidelity and reliability of the results generated. Lee *et al*. (1996) propose a branch-and-bound algorithm for solving a basic version of the CMP with delays and a hybrid approach based on Genetic Algorithms (GA) and branch-and-bound (GA to compute the delays and B&B algorithm to compute the paths). An important limitation of this work is the generation of complex and circuitous paths that convoys, occasionally, have to take to reach their destinations. Montana *et al*. (1999) investigate the problem of routing and scheduling military convoys between a single origin and destination pair using genetic algorithms while considering multiple objectives. They divide the problem into two parts (a) selecting a fixed set of routes and (b) convoy formation of trucks and assignment of routes and departure times for individual convoys. Harrison and Rayward-Smith (1999) consider the problem of finding minimal cost linkages in graphs and discuss its relevance to the convoy movement problem. Harrison (2000) presents a formal specification of the convoy movement problem and presents a model in terms of a time-space network. A Lagrangean relaxation based heuristic technique is proposed and evaluated on realistic scenarios based on the UK MoD's Scenario Advisory Group (SAG) settings.

Chardaire *et al*. (2005), in their seminal work, establish that the CMP is *NP*-complete by establishing that the disjoint connecting path problem can be polynomially reducible to the decision version of the CMP. They introduce an integer programming model based on the concept of a time-space network for a simplified version of the model. Tuson and Harrison (2005) propose a simple heuristic based on delay search and demonstrate that the *NP*-

hardness is only a worst case measure of the problem's time complexity and real world problems need not necessarily be hard to solve. Robinson and Leiss (2006) propose a methodology combining genetic algorithms with discrete event simulation for convoy scheduling. They show that their approach automatically removes conflicts from a convoy schedule iteratively and generate quick results for realistic problem instances. Ram Kumar and Narendran (2008, 2009a) propose a robust mixed integer programming (IP) model for solving the convoy movement problem. They demonstrate its suitability for small to moderate size problem instances (up to 50 cities and 10 convoys) and propose simple heuristics based on Dijkstra's algorithm for larger problem instances. The amenability of their IP model for lagrangean relaxation by relaxing no-crossing and minimum headway constraints is presented in detail in Ram Kumar and Narendran (2011). Ram Kumar *et al*. (2009b) propose simulated annealing based meta-heuristic procedure for bi-criteria CMP considering total travel time and travel span as objectives.

Gopalan and Narayanaswamy (2009) consider a dynamic version of the CMP where convoy demands arise over time. By proving that the 3-satisfiability problem polynomially reduces to the restricted version of CMP that they consider, they propose three approximation algorithms. Goldstein *et al*. (2010) propose a genetic algorithm for CMP that allows convoys to cross at vertices on a directed route network. They empirically analyze the model via discrete event simulation on a single instance of the problem. Lau *et al.* (2010) propose a hybrid methodology combining Dijkstra's algorithm with constraint programming techniques for routing of convoys in an urban city. Gopalan (2015) characterizes the computational complexity of several restricted versions of CMP. The author proposes a polynomial time algorithm for the specific case of single objective two convoy problem while presenting an approximation algorithm for larger number of convoys. Recently, Sadeghnejad-Barkousaraie *et al*. (2016) study the peacetime CMP from a civilian perspective with an objective to minimize civilian traffic disruptions. They develop an exact hybrid algorithm combining *k*-shortest path algorithm with minimum weighted *k*-clique in a *k*-partite graph.

To summarize, though there has been a concerted effort over the past 20 years to develop solution methodologies based on exact methodologies such as branch & bound (B&B) and integer programming (IP) for solving problem instances of CMP, the efforts are focused on solving smaller and restricted versions of CMP. Heuristics based on Dijkstra's algorithm and meta-heuristics which have been applied for CMP include Taboo search (Bovet et al. 1991), Genetic Algorithms (Lee *et al*. 1996; Montana *et al*. 1999; Goldstein *et al*. 2010), and Simulated Annealing (Ram Kumar *et al*. 2009b).

In order to validate the results of this work, especially with respect to computational time, we choose the work of Goldstein *et al*. (2010) for the following reasons:

- First and foremost, it is a well-known fact that meta-heuristics are very quick compared to the conventional optimization procedures. Hence, it is only justifiable if the present work is compared with other meta-heuristics reported in the literature.
- Genetic Algorithms proposed in Lee *et al*. (1996) and Montana *et al*. (1999) suffer from inherent limitations. For example, Lee *et al*. (1996) use GA just to compute the initial delays associated with convoys by pre-fixing their routes. Similarly, the work of Montana *et al*. (1999) restricts movement of convoys between single origin and destination pair.
- Taboo search reported in Bovet *et al*. (1990) is applicable for situations when convoys are to be routed and scheduled along only single road/rail link.
- Simulated Annealing based procedure reported in Ram Kumar *et al*. (2009b) is quite specific to a multi-objective problem drawing ideas from goal programming approach.
- The heuristics reported for CMP are essentially extensions of Dijkstra's algorithm aimed at restoring feasibility of routes and schedules through repairing mechanisms. The focus is more on obtaining feasible solutions rather than near-optimal solutions.
- The version of GA presented in Goldstein *et al*. (2010), being the latest, considers most of the features of CMP unlike older studies.

Given the success rate with which ant colony optimization has been used for solving a variety of routing & scheduling optimization problems and considering the fact that there is no published research work, at least to the best of our knowledge, investigating the suitability of ant colony algorithms for CMP, motivates us to bridge the research gap by developing a hybridized ant colony algorithm for a generic version of the CMP.

## 3. Hybridized ant colony algorithm

### *3.1 Notations*

The list of notations used to describe the proposed HAC is as follows:

$m$ - Number of ants (also military convoys)

$\eta_{ij}$ - Inverse of the distance between nodes $i$ and $j$

$\tau_{ij}^{k}$ - Pheromone trail intensity of $k^{th}$ ant on arc *(ij)*

$\rho$ - Evaporation rate constant

$Q$ - Constant used to update trail intensities

$P_{ij}^{k}$ - Probability of ant 'k' traversing edge *(ij)*

$N_{eligible}$ - Set of nodes eligible for further traversal from the current node

$L_{best}$ - Objective function value of the best solution found so far

### *3.2 Pseudo code*

Ant colony optimization is a meta-heuristic for obtaining near-optimal solutions for hard combinatorial optimization problems in very less computational time. The ants deposit a chemical called pheromone while walking on the ground to mark the path from the nest to food sources and back. The amount of pheromone influences the probability of ants choosing one path over the other. Over a period of time, the process culminates in the ants determining the shortest path between the nest and source of food. In this work, artificial ants representing military convoys construct conflict-free routes between corresponding origin-destination pairs. The route construction mechanism coupled with taboo lists always ensures that only feasible solutions are generated. A local search procedure is then employed for further improvement in the solution quality. This process completes one iteration of the algorithm. Before the start of next iteration, the taboo lists are emptied and the whole procedure is repeated until the termination criterion is met. The HAC algorithm is summarized as follows. The detailed description pertaining to the generation of seed solution, route construction mechanism and local search procedures is presented in the following subsections aided with a flowchart illustrated in Fig.1.

Step 1: Generate a seed solution and initialize the trail intensities across the network

Step 2: Repeat the following procedure until the termination criteria is satisfied:

- Assign the first node in the path of all the ants to the corresponding source nodes of the convoys.
- Construct routes between origin-destination pairs for ***m*** number of ants.
- Employ the local search procedure to improve the quality of the solution.
- Update the information about the best solution found so far.
- Update the trail intensities across all the arcs of the network.

Step 3: Terminate the algorithm and report the best solution.

### *3.3 Seed solution and initialization of trail intensities*

The pheromone trail intensity across the arcs of the network can be initialized using either a small positive constant as suggested in Dorigo *et al.* (1996) or a heuristic solution. In this work, we use the "initial delay" based heuristic proposed by Lee *et al.* (1996) to generate a seed solution for the problem and use it for initializing trail intensities. The heuristic works by routing convoys along the shortest paths between corresponding *O-D* pairs while delaying their start time at the source nodes in such a way that the minimum headway and no-crossing constraints are satisfied. Using computational experiments, described later in Section 5, we also evaluate the effect of choosing a heuristic over a small constant on the overall performance of the algorithm. For edges that are part of the seed solution, the pheromone intensity is assigned a value of $Q/L$ with $L$ being the sum of arrival times of all the convoys at their respective destinations. For the rest of the edges, to ensure diversification of search, the pheromone intensity is set at 50% of the value of $Q/L$.

### *3.4 Construction of routes*

Each ant represents a military convoy waiting to traverse from its source node to the destination node. The first node in the path of all the ants is set to the corresponding source node of the convoy. During route construction, every ant performs the following functions until it reaches the destination node:

a. From any given node, it prepares a list of nodes eligible for traversal by checking the following three criteria:
   - Presence of no other convoy (another ant) traversing that particular edge in the opposite direction;
   - If a convoy is detected traveling in the same direction as that of the ant on the edge, then minimum headway time should be satisfied; and
   - The edge has not been traversed earlier (taboo list)

   Only if all the above three criteria are satisfied, the node will be added to the eligible list.

b. From the list of eligible nodes, it chooses the next node based on a probability function (Equation 1) of the distance between the nodes and the amount of pheromone deposit on the edge connecting the nodes. However, it is important to mention here that the pheromone deposits by multiple ants along a particular edge are considered independent. This implies the probability of an ant choosing a particular edge depends on the visibility and the quantum of pheromone deposits made only by those ants that traversed between that particular *O-D* pair in the earlier iterations. This step forms the crux of the algorithm.

$$P_{ij}^{k} = \begin{cases} \dfrac{(\tau_{ij}^{k})^{\alpha}(\eta_{ij})^{\beta}}{\sum\limits_{l \in N_{eligible}} (\tau_{il}^{k})^{\alpha}(\eta_{il})^{\beta}}, \; if \;\; j \in N_{eligible} \\ 0, \qquad\qquad Otherwise \end{cases} \tag{1}$$

### *3.5 Local search procedure*

After the construction of routes by the ants, for possible improvement in the quality of the solution, a local search procedure is employed. The purpose is to explore the neighborhood of the best solution obtained so far for further improvement. Here, we use insertion based local search. On the paths generated by each ant, a node that has direct connectivity with the destination node is randomly chosen and both of them are joined bypassing the rest of the route. If the resulting solution is devoid of conflicts with the routes and schedules of other convoys, the new solution is updated as the current best solution. This procedure is repeated 10 times on the routes of every ant. The best solution obtained at the end of local search is used to update the pheromone trails for the beginning of subsequent iteration.

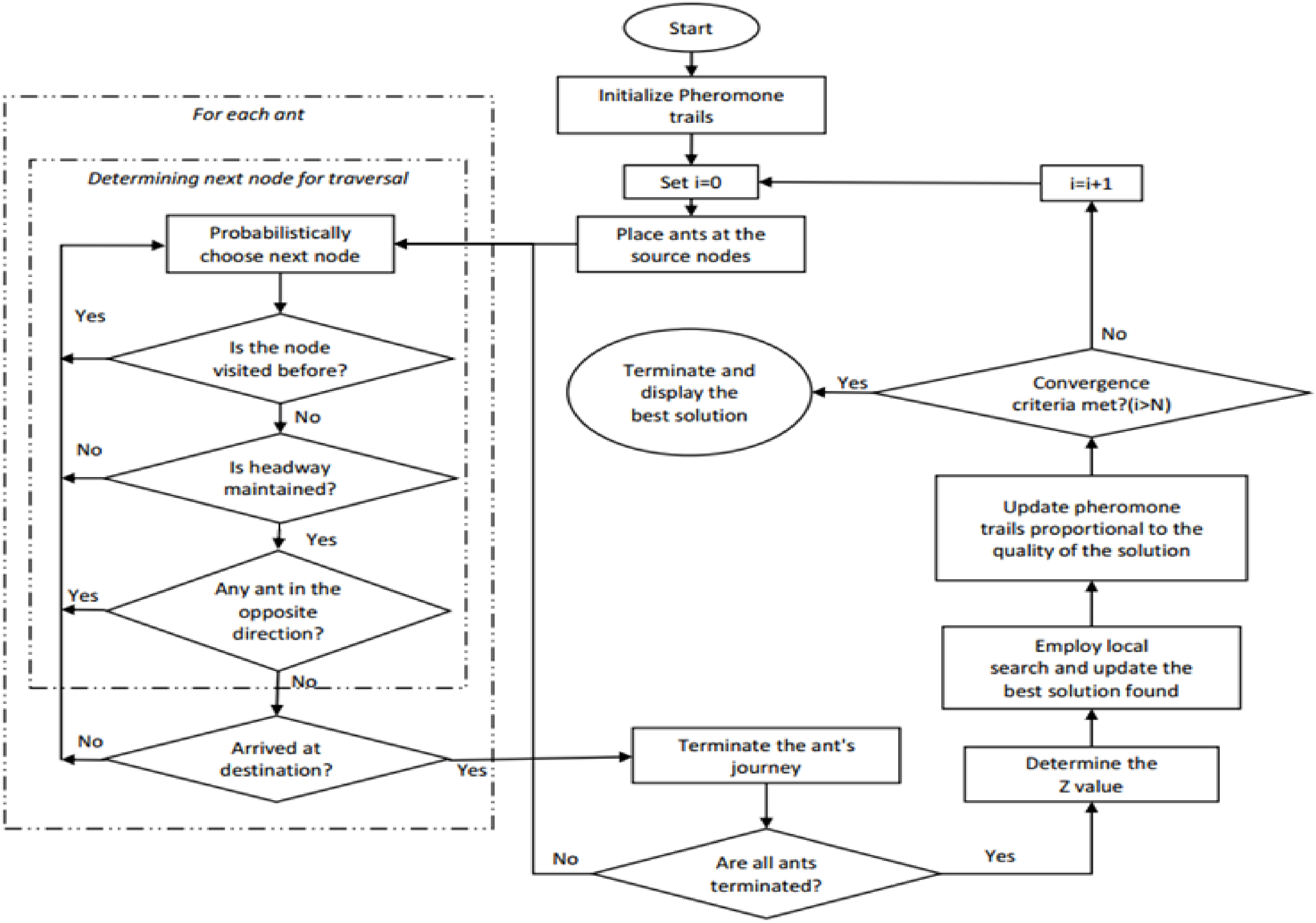


**Figure 1:** Flowchart of HAC

### *3.6 Updating pheromone trail intensities*

After performing local search on the best solution obtained at the end of every iteration, the pheromone trail intensities are updated on all the arcs. Pheromone is a chemical that evaporates over time. As originally proposed by Dorigo *et al.* (1996), the evaporation rate is controlled by a parameter called trail evaporation constant $\rho$ whose value lies between 0 and 1. As shown in Equation (2), the trail intensity on the arcs that are part of the best solution is augmented whereas for the rest of the arcs, it is lowered.

$$\tau_{ij}^{k} = \begin{cases} \rho * \tau_{ij}^{k} + 500 / L_{best}, \ if \ \ edge \ ij \in best \ \ solution \\ \rho * \tau_{ij}^{k} \qquad \text{Otherwise} \end{cases} \tag{2}$$

## 4. Computational Experiments

In this section, we explain in detail how the problem instances were generated, their characteristics and the values chosen for different parameters of the algorithm. We also briefly describe the working of GA for CMP as presented in Goldstein *et al.* (2010).

### *4.1 Problem datasets*

The unavailability of benchmark problem instances and the inaccessibility of the DSTL (Defence Science and Technology Laboratory, UK) datasets used in Chardaire *et al.* (2005) prompted us to generate our own hypothetical problem instances based on the network characteristics reported in the literature. We characterize each problem instance of CMP with four parameters: Number of nodes (N); Number of convoys (C); Arc-density factor (μ); and Identical destination factor ($\theta$). We consider 8 different sized networks by varying the number of nodes from 10 to 100 as summarized in Table 1. The number of convoys is dependent on the number of nodes and is controlled by a parameter called node-convoy ratio whose value varies between 2, 3 and 4. The arc density factor, which is a measure of sparseness of the network, can be loosely defined as the ratio of number of available arcs to maximum possible number of arcs. The arc density factor is varied between 0.25, 0.50, and 0.75 for all the networks with arc lengths (measured in time units) ranging from 100 to 1000. Identical destination factor, as the name suggests, indicates the percentage of convoys heading towards the same destination. The value of $\theta$ varies between 0.2, 0.4 and 0.6. For the sake of simplicity, it is assumed that all the convoys are ready at their origin nodes at time zero and travel across the network at the same speed. The due date imposed for every convoy to reach its destination is four times the shortest time between the corresponding *O-D* pair. To avoid input bias, for every combination of input parameters, 10 problem instances are generated by changing the origin and destination pairs. A total of 2160 problem instances are generated and solved. The network data has been found to be consistent and representative of real-life instances by serving Indian army personnel, who wish to remain anonymous.

**Table 1**: Network Characteristics

| Network ID | Number of Nodes | Number of Convoys | | |
|---|---|---|---|---|
| N1 | 10 | 3 | 4 | 5 |
| N2 | 15 | 4 | 5 | 8 |
| N3 | 25 | 7 | 9 | 13 |
| N4 | 35 | 9 | 13 | 18 |
| N5 | 50 | 13 | 17 | 25 |
| N6 | 75 | 18 | 27 | 35 |
| N7 | 85 | 22 | 29 | 43 |
| N8 | 100 | 25 | 34 | 50 |

*4.2 Parameter setting*

For a given problem instance, the number of ants in the HAC is always equal to the number of convoys to be routed across the network. The value of $Q$ for initializing the trail intensity across all the arcs is chosen as 2500. After conducting initial trial of experiments, this value is arrived at as a trade-off between computational time and extent of exploration of search space. Too big a value for $Q$ results in saving computational time but poor exploration and similarly a small value allows to examine the search space thoroughly but takes longer time for convergence. We found that the combination of $\propto = 1, \beta = 2.5 \text{ } and \text{ } \rho = 0.8$ results in the best performance of the algorithm.

*4.3 GA based search procedure*

Goldstein *et al*. (2010) propose a form of genetic algorithm, christened as "local beam search", to generate quick solutions to CMP. Essentially, the procedure starts with finding the shortest paths for all the convoys between respective *O-D* pairs using Dijkstra's algorithm. The initial population consists of 10 copies of this solution. For each solution, 8 new solutions are generated by resolving the conflicts either by slowing down the speed of the convoys or by re-routing the convoys. This results in generating 80 new solutions. The best 10 out of the total 90 solutions are selected for next generation. The search process is terminated when there is no improvement from one generation to another. There is no mention about any crossover or mutation mechanisms to improve the solution quality.

## 5. Computational results

The proposed HAC algorithm and the GA version reported in Goldstein et al. (2010) were coded in Visual C++ and solved on a desktop computer run on dual core (3.2 GHz) processor with 4GB of RAM on a Windows 7 operating system. The programs were allowed to run for a maximum of 15 minutes (900 seconds) or when there is no improvement in the final solution for successive 100 iterations, whichever is earlier. For solving the problem instances to optimality, the integer programming (IP) mathematical model reported in Ram Kumar et al. (2010) was used and solved using CPLEX optimization studio version 12.5 with a computational time limit of 8 hours. Problem instances for which optimal solutions were found within the set time, the quality of heuristic solutions is assessed using Equation (3). For larger problem instances, where CPLEX fails to converge to optimality even after 8 hours, the heuristic solutions are compared with lower bounds (the problem being a minimization objective) as shown in Equation (4). Obviously, lesser the % gap, better the quality of the heuristics. We use the Lagrangean Relaxation (LR) procedure reported in Ram Kumar and Narendran (2011) to generate lower bounds. In their work, the authors propose absorbing the no-crossing and minimum headway constraints into the objective function of their IP model by attaching Lagrangean multipliers and solving the rest of the problem as a multiple origin-destination pair shortest path problem.

Quality of Heuristic solution w.r.t optimal solution = $\frac{Z_{Heuristic} - Z_{optimal}}{Z_{optimal}} x100\%$ (3)

Quality of Heuristic solution w.r.t lower bound = $\frac{Z_{Heuristic} - Z_{Lagrangean}}{Z_{Lagrangean}} x100\%$ (4)

The results are tabulated, network wise, from Tables 2 to 9. The following are the inferences drawn from the results.

- From the results, it is clear that, as the problem size increases, the computational time also increases. In the case of CPLEX, the rate of increase appears to be exponential whereas it looks linear in the case of HAC and GA.
- While the HAC could solve problem instances up to 85 nodes and 43 convoys within the set computational time of 900 seconds, the GA could only solve up to 35 nodes. One possible reason for the poor performance of GA may be because of the extensive computational effort required for the generation of feasible solutions itself. As mentioned in Section 4.3, the GA procedure reported is only an advanced form of local search rather than a global search procedure. Moreover, the absence of crossover and mutation mechanisms may also be contributing to the poor performance.
- Within the set time of 8 hours, CPLEX could generate optimal solutions only up to a problem size of 50 nodes. Beyond network N5, lower bounds were generated and used to benchmark the quality of heuristics. The fact that CPLEX takes more than 8 hours to converge to optimality for moderate sized problem instances makes this research work all the more relevant and practical.
- Across all the networks, consistently, HAC outperforms GA both in terms of computational time and computational quality. Though the superiority of HAC over GA is expected, the poor performance of GA can again be attributed to the fact that the information available in the public domain about the finer details of the working of GA is limited which directly affects the GA's performance.
- For problem instances up to 50 cities, the average quality of HAC is less than 5%. Beyond this size, the average % gap almost gets doubled to 10%. This is partially because of the fact that the lower bounds generated using Lagrangean relaxation represent only proxy optimal solutions. The quality of lower bounds also influences the final % gap reported.
- From Fig. 2, it can be observed that, for all the networks, the identical destination factor appears to be least influencing of the computational time. Upon thorough examination of the routes and schedules of the convoys, it is learnt that though multiple convoys are headed towards an identical destination, there were no conflicts among them to share the same edge at the same time. Hence there was no "competition" as such to occupy an edge and, by default, the minimum headway constraint was satisfied. So, the identical destination factor did not make any difference to the computational time.

- From Fig. 3, it can be inferred that the node-convoy ratio plays an important role in determining the computational time for convergence. It can be noted that problem instances with higher number of convoys (lower node-convoy ratio) took relatively longer time for convergence. This is not at all surprising because of the way the algorithm has been designed and executed.
- Similarly, from Fig. 4, it is evident that higher the arc-density factor value, the longer it took the algorithm to converge to a near-optimal solution. Again, this could be because the pheromone intensities are to be updated at the end of every iteration on the large number of edges present in the network. Moreover, with higher connectivity, the calculation of transition probabilities, the updating of taboo lists, etc. consume relatively more computational effort.
- From Fig. 5, it is interesting to note that the quality of heuristic solutions is better for networks with higher arc-densities and relatively poor for networks with lower arc-densities. This is a counter intuitive result. Upon detailed examination of the results, we understood that for networks with low arc-densities, the number of feasible solutions itself is very less. It is because of this reason, the conflicts among the convoy schedules were large in number and seldom there was any improvement in the quality of solutions over hundreds of iterations. This problem led to the algorithm getting quickly converged to a local optima rather than a global optima. Even the local search procedure proved to be futile in these cases.

**Figure 2: Identical destination factor Vs Computational time**

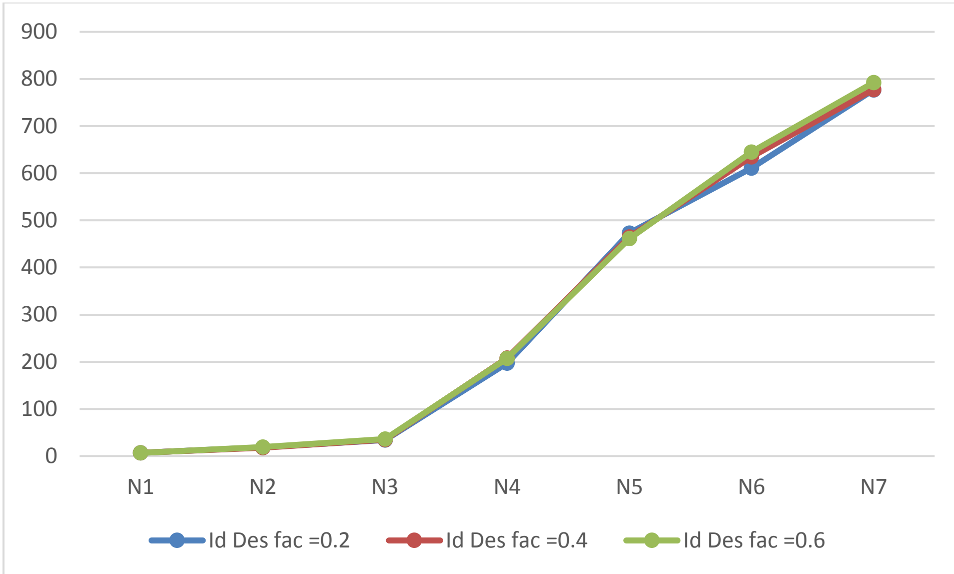


**Figure 3: Node-convoy ratio Vs Computational time**

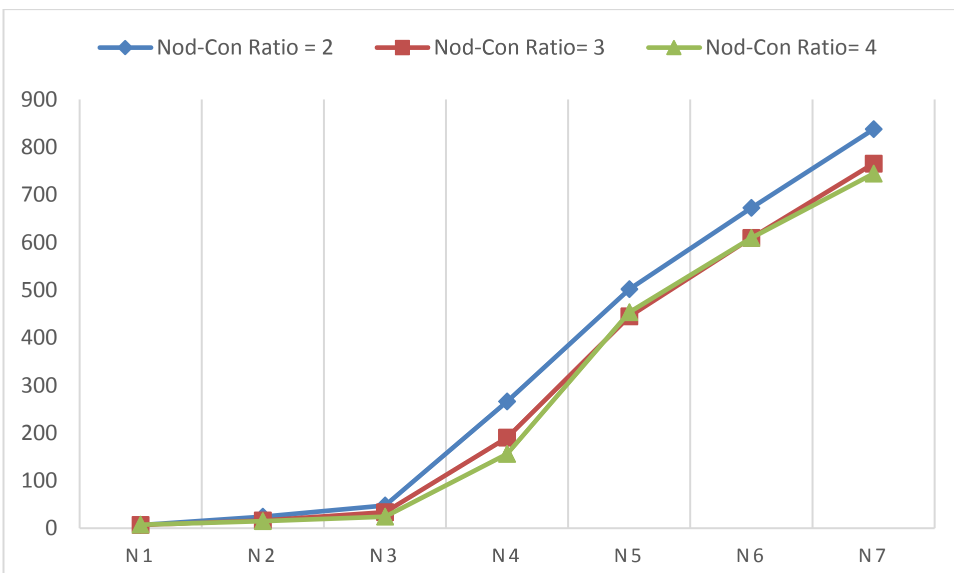


**Figure 4: Arc-density factor Vs Computational time**

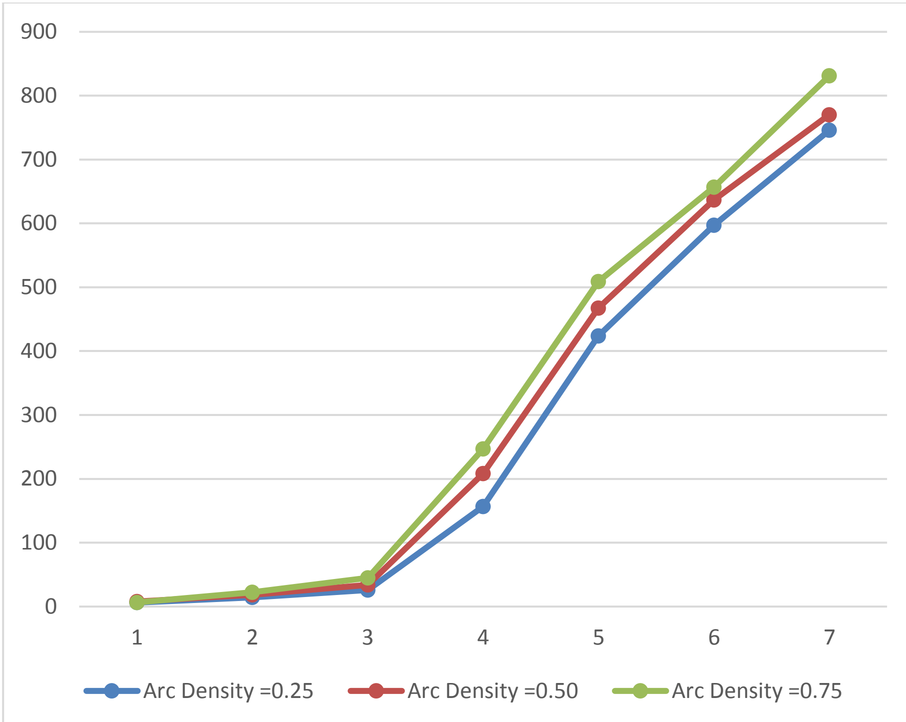


**Figure 5: Arc-density factor Vs Computational quality**

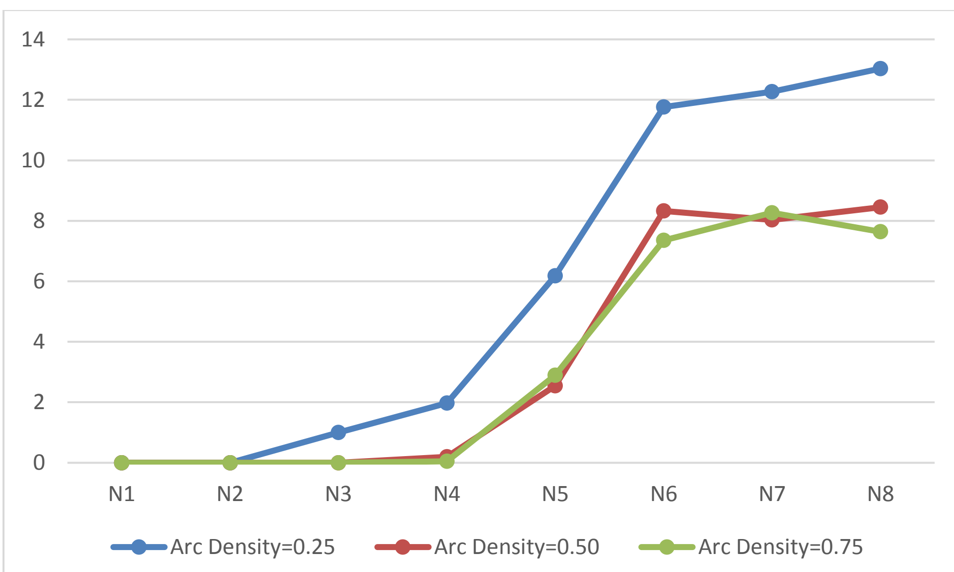


**Table 2**: Results of Network N1

| Network ID | Number of Nodes | Number of Convoys (Node-convoy ratio) | Arc-density factor | Identical Destination Factor | Average Computational time (seconds) | | | Average Computational quality | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | CPLEX | HAC | GA | HAC* | GA@ |
| N1 | 10 | 5 (2) | 0.25 | 0.2 | 0.95 | 8.07 | 11.76 | 0 | 0 |
| | | | | 0.4 | 0.71 | 7.83 | 13.69 | 0 | 0 |
| | | | | 0.6 | 0.97 | 6.62 | 13.4 | 0 | 0 |
| | | | 0.50 | 0.2 | 0.59 | 7.41 | 14.75 | 0 | 0 |
| | | | | 0.4 | 0.97 | 5.02 | 10.81 | 0 | 0 |
| | | | | 0.6 | 0.28 | 7.71 | 9.74 | 0 | 0 |
| | | | 0.75 | 0.2 | 0.62 | 4.62 | 9.28 | 0 | 0 |
| | | | | 0.4 | 0.85 | 7.54 | 13.88 | 0 | 0 |
| | | | | 0.6 | 1.05 | 4.96 | 11.18 | 0 | 0 |
| | | 4 (3) | 0.25 | 0.2 | 0.72 | 6.15 | 14.96 | 0 | 0.03 |
| | | | | 0.4 | 0.92 | 4.54 | 14.74 | 0 | 0 |
| | | | | 0.6 | 0.73 | 4.55 | 13.23 | 0 | 0 |
| | | | 0.50 | 0.2 | 0.93 | 7.53 | 9.69 | 0 | 0 |
| | | | | 0.4 | 0.5 | 8.61 | 11.78 | 0 | 0 |
| | | | | 0.6 | 0.4 | 6.82 | 9.33 | 0 | 0 |
| | | | 0.75 | 0.2 | 0.72 | 7.81 | 9.96 | 0 | 0 |
| | | | | 0.4 | 0.25 | 7.69 | 9.27 | 0 | 0 |
| | | | | 0.6 | 0.98 | 5.75 | 10.08 | 0 | 0 |
| | | 3 (4) | 0.25 | 0.2 | 0.28 | 4.28 | 9.46 | 0 | 0.01 |
| | | | | 0.4 | 0.82 | 6.94 | 12.04 | 0 | 0 |
| | | | | 0.6 | 0.26 | 7.45 | 9.58 | 0 | 0 |
| | | | 0.50 | 0.2 | 0.65 | 8.81 | 14.65 | 0 | 0 |
| | | | | 0.4 | 0.38 | 10.98 | 14.46 | 0 | 0 |
| | | | | 0.6 | 0.41 | 7.50 | 10.92 | 0 | 0 |
| | | | 0.75 | 0.2 | 0.92 | 7.46 | 9.41 | 0 | 0 |
| | | | | 0.4 | 0.27 | 4.11 | 14.98 | 0 | 0 |
| | | | | 0.6 | 0.43 | 8.16 | 12.85 | 0 | 0 |

*** -** $((Z_{HAC} – Z_{CPLEX})/ Z_{CPLEX}) * 100\%$

**@ -** $((Z_{GA} – Z_{CPLEX})/ Z_{CPLEX}) * 100\%$

– Failed to converge within the set time

**Table 3**: Results of Network N2

| Network ID | Number of Nodes | Number of Convoys (Node-convoy ratio) | Arc-density factor | Identical Destination Factor | Average Computational time (seconds) | | | Average Computational quality | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | CPLEX | HAC | GA | HAC* | GA@ |
| N2 | 15 | 8 (2) | 0.25 | 0.2 | 1.45 | 18.13 | 34.21 | 0 | 0.13 |
| | | | | 0.4 | 1.47 | 15.75 | 20.47 | 0 | 0 |
| | | | | 0.6 | 2.11 | 19.24 | 25.53 | 0 | 0 |
| | | | 0.50 | 0.2 | 1.95 | 24.31 | 15.02 | 0 | 0.08 |
| | | | | 0.4 | 2.38 | 19.28 | 15.38 | 0 | 0 |
| | | | | 0.6 | 2.01 | 27.89 | 20.14 | 0 | 0 |
| | | | 0.75 | 0.2 | 2.49 | 29.34 | 33.08 | 0 | 0 |
| | | | | 0.4 | 2.28 | 31.15 | 33.06 | 0 | 0 |
| | | | | 0.6 | 1.33 | 30.98 | 24.77 | 0 | 0 |
| | | 5 (3) | 0.25 | 0.2 | 1.01 | 14.18 | 12.39 | 0 | 0.21 |
| | | | | 0.4 | 0.94 | 12.12 | 20.96 | 0 | 0 |
| | | | | 0.6 | 1.30 | 12.19 | 22.48 | 0 | 0 |
| | | | 0.50 | 0.2 | 1.57 | 15.72 | 11.23 | 0 | 0 |
| | | | | 0.4 | 1.28 | 17.02 | 30.52 | 0 | 0.07 |
| | | | | 0.6 | 1.42 | 17.33 | 26.11 | 0 | 0.18 |
| | | | 0.75 | 0.2 | 1.52 | 20.09 | 24.71 | 0 | 0 |
| | | | | 0.4 | 1.85 | 20.78 | 30.83 | 0 | 0 |
| | | | | 0.6 | 2.85 | 18.78 | 29.35 | 0 | 0 |
| | | 4 (4) | 0.25 | 0.2 | 1.33 | 12.91 | 18.26 | 0 | 0.10 |
| | | | | 0.4 | 2.64 | 14.14 | 26.45 | 0 | 0 |
| | | | | 0.6 | 1.31 | 12.09 | 32.07 | 0 | 0 |
| | | | 0.50 | 0.2 | 2.21 | 14.77 | 29.03 | 0 | 0 |
| | | | | 0.4 | 1.79 | 15.25 | 32.60 | 0 | 0.34 |
| | | | | 0.6 | 2.12 | 15.88 | 30.63 | 0 | 0 |
| | | | 0.75 | 0.2 | 1.88 | 18.78 | 13.75 | 0 | 0 |
| | | | | 0.4 | 1.67 | 13.12 | 23.29 | 0 | 0 |
| | | | | 0.6 | 1.75 | 18.1 | 32.29 | 0 | 0 |

*** -** $((Z_{HAC} – Z_{CPLEX})/ Z_{CPLEX}) * 100\%$

**@ -** $((Z_{GA} – Z_{CPLEX})/ Z_{CPLEX}) * 100\%$

– Failed to converge within the set time

**Table 4**: Results of Network N3

| Network ID | Number of Nodes | Number of Convoys (Node-convoy ratio) | Arc-density factor | Identical Destination Factor | Average Computational time (seconds) | | | Average Computational quality | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | CPLEX | HAC | GA | HAC* | GA@ |
| N3 | 25 | 13 (2) | 0.25 | 0.2 | 49.65 | 34.08 | 125.31 | 1.94 | 4.96 |
| | | | | 0.4 | 53.14 | 29.62 | 81.12 | 2.38 | 7.27 |
| | | | | 0.6 | 50.99 | 35.47 | 129.16 | 2.25 | 5.94 |
| | | | 0.50 | 0.2 | 62.24 | 41.30 | 81.14 | 0 | 3.55 |
| | | | | 0.4 | 55.71 | 48.26 | 116.7 | 0 | 6.77 |
| | | | | 0.6 | 51.19 | 45.92 | 103.28 | 0 | 6.76 |
| | | | 0.75 | 0.2 | 58.37 | 62.03 | 104.62 | 0 | 3.08 |
| | | | | 0.4 | 54.26 | 67.19 | 93.77 | 0 | 3.5 |
| | | | | 0.6 | 54.38 | 63.44 | 115.19 | 0 | 5.38 |
| | | 9 (3) | 0.25 | 0.2 | 33.45 | 23.17 | 92.18 | 0.77 | 3.38 |
| | | | | 0.4 | 38.92 | 19.50 | 97.94 | 0.03 | 7.29 |
| | | | | 0.6 | 38.07 | 30.04 | 100.36 | 0.41 | 6.8 |
| | | | 0.50 | 0.2 | 44.21 | 32.20 | 97.35 | 0 | 5.93 |
| | | | | 0.4 | 37.60 | 35.06 | 88.69 | 0 | 7.73 |
| | | | | 0.6 | 48.29 | 34.88 | 90.85 | 0 | 4.82 |
| | | | 0.75 | 0.2 | 51.15 | 42.72 | 106.3 | 0 | 5.68 |
| | | | | 0.4 | 41.29 | 40.06 | 97.12 | 0 | 3.69 |
| | | | | 0.6 | 47.76 | 42.39 | 96.4 | 0 | 4.7 |
| | | 7 (4) | 0.25 | 0.2 | 36.79 | 20.61 | 104.55 | 0.29 | 7.39 |
| | | | | 0.4 | 30.91 | 19.47 | 82.83 | 0.42 | 3.78 |
| | | | | 0.6 | 35.48 | 19.12 | 100.53 | 0.48 | 5.54 |
| | | | 0.50 | 0.2 | 38.22 | 20.48 | 109.48 | 0 | 6.62 |
| | | | | 0.4 | 39.01 | 23.50 | 128.57 | 0 | 3.7 |
| | | | | 0.6 | 41.25 | 23.69 | 107.71 | 0 | 6.38 |
| | | | 0.75 | 0.2 | 40.00 | 27.11 | 116.88 | 0 | 3.93 |
| | | | | 0.4 | 43.82 | 31.86 | 94.08 | 0 | 3.97 |
| | | | | 0.6 | 43.16 | 29.45 | 86.55 | 0 | 5.17 |

*** -** $((Z_{HAC} - Z_{CPLEX}) / Z_{CPLEX}) * 100\%$

**@ -** $((Z_{GA} - Z_{CPLEX}) / Z_{CPLEX}) * 100\%$

– Failed to converge within the set time

**Table 5**: Results of Network N4

| Network ID | Number of Nodes | Number of Convoys (Node-convoy ratio) | Arc-density factor | Identical Destination Factor | Average Computational time (seconds) | | | Average Computational quality | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | CPLEX | HAC | GA | HAC* | GA@ |
| N4 | 35 | 18 (2) | 0.25 | 0.2 | 722.08 | 203.11 | 589.5 | 3.17 | 11.1 |
| | | | | 0.4 | 784.25 | 194.66 | 548.22 | 4.58 | 9.04 |
| | | | | 0.6 | 800.12 | 175.50 | 525.33 | 3.99 | 12.02 |
| | | | 0.50 | 0.2 | 922.23 | 255.83 | 641.85 | 0.64 | 10.04 |
| | | | | 0.4 | 928.36 | 284.61 | 531.96 | 0.82 | 12.91 |
| | | | | 0.6 | 875.29 | 288.93 | 461.5 | 0.30 | 9.05 |
| | | | 0.75 | 0.2 | 1134.2 | 308.26 | 452.4 | 0.12 | 9.15 |
| | | | | 0.4 | 1189.5 | 344.10 | 649.41 | 0.08 | 14.83 |
| | | | | 0.6 | 1331.6 | 340.58 | 524.27 | 0.24 | 9.02 |
| | | 13 (3) | 0.25 | 0.2 | 704.21 | 142.44 | 659.73 | 2.33 | 10.23 |
| | | | | 0.4 | 755.32 | 158.65 | 618.51 | 1.20 | 16.91 |
| | | | | 0.6 | 739.48 | 158.88 | 616.53 | 1.48 | 7.13 |
| | | | 0.50 | 0.2 | 824.26 | 191.36 | 494.63 | 0 | 13.4 |
| | | | | 0.4 | 779.12 | 185.65 | 550.56 | 0 | 13.67 |
| | | | | 0.6 | 844.70 | 182.94 | 572.37 | 0 | 10.48 |
| | | | 0.75 | 0.2 | 902.53 | 215.18 | 645.3 | 0 | 12.24 |
| | | | | 0.4 | 900.77 | 233.49 | 483.43 | 0 | 11.24 |
| | | | | 0.6 | 991.13 | 240.07 | 604.3 | 0 | 13.2 |
| | | 9 (4) | 0.25 | 0.2 | 558.29 | 103.55 | 656.54 | 1.03 | 18.51 |
| | | | | 0.4 | 621.34 | 134.72 | 502.2 | 0 | 18.94 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | 0.6 | 603.58 | 139.00 | 511.75 | 0 | 14.05 |
| | | | 0.50 | 0.2 | 589.37 | 167.84 | 582.21 | 0 | 16.03 |
| | | | | 0.4 | 611.08 | 158.42 | 481.98 | 0 | 8.11 |
| | | | | 0.6 | 676.03 | 160.08 | 450.85 | 0 | 17.92 |
| | | | 0.75 | 0.2 | 653.23 | 184.25 | 538.31 | 0 | 15.91 |
| | | | | 0.4 | 661.28 | 177.31 | 411.71 | 0 | 9.08 |
| | | | | 0.6 | 665.90 | 180.22 | 682.51 | 0 | 17.5 |

*** -** $((Z_{HAC} – Z_{CPLEX})/ Z_{CPLEX}) * 100\%$

**@ -** $((Z_{GA} – Z_{CPLEX})/ Z_{CPLEX}) * 100\%$

– Failed to converge within the set time

**Table 6**: Results of Network N5

| Network ID | Number of Nodes | Number of Convoys (Node-convoy ratio) | Arc-density factor | Identical Destination Factor | Average Computational time (seconds) | | | Average Computational quality | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | CPLEX | HAC | GA | HAC | GA |
| N5 | 50 | 25 (2) | 0.25 | 0.2 | 20677.15 | 446.83 | 900$^{\$}$ | 7.64 | 16.69 |
| | | | | 0.4 | 21115.23 | 480.68 | 900$^{\$}$ | 6.41 | 27.38 |
| | | | | 0.6 | 23917.92 | 409.34 | 900$^{\$}$ | 6.04 | 20.05 |
| | | | 0.50 | 0.2 | 20571.97 | 524.46 | 900$^{\$}$ | 3.51 | 29.1 |
| | | | | 0.4 | 22577.27 | 500.39 | 900$^{\$}$ | 2.76 | 22.19 |
| | | | | 0.6 | 22945.03 | 502.95 | 900$^{\$}$ | 2.66 | 16.47 |
| | | | 0.75 | 0.2 | 23374.18 | 586.18 | 900$^{\$}$ | 4.94 | 16.04 |
| | | | | 0.4 | 21094.35 | 519.68 | 900$^{\$}$ | 2.71 | 19.31 |
| | | | | 0.6 | 22371.91 | 547.26 | 900$^{\$}$ | 4.46 | 28.49 |
| | | 17 (3) | 0.25 | 0.2 | 16445.57 | 437.72 | 900$^{\$}$ | 5.81 | 15.87 |
| | | | | 0.4 | 19518.63 | 395.36 | 900$^{\$}$ | 6.64 | 15.07 |
| | | | | 0.6 | 13095.67 | 395.13 | 900$^{\$}$ | 5.71 | 20.18 |
| | | | 0.50 | 0.2 | 20337.13 | 455.55 | 900$^{\$}$ | 3.77 | 19.84 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | 0.4 | 18272.67 | 413.7 | 900$ | 2.38 | 18.43 |
| | | | | 0.6 | 19935.75 | 425.77 | 900$ | 1.84 | 26.68 |
| | | | 0.75 | 0.2 | 21605.44 | 471.77 | 900$ | 1.31 | 28.31 |
| | | | | 0.4 | 17534.92 | 489.53 | 900$ | 1.61 | 27.21 |
| | | | | 0.6 | 15679.83 | 516.07 | 900$ | 2.79 | 28.82 |
| | | 13 (4) | 0.25 | 0.2 | 11024.69 | 396.28 | 900$ | 6.38 | 25.75 |
| | | | | 0.4 | 11504.41 | 444.27 | 900$ | 5.66 | 17.81 |
| | | | | 0.6 | 14189.49 | 406.77 | 900$ | 5.29 | 23.1 |
| | | | 0.50 | 0.2 | 13991.56 | 509.28 | 900$ | 2.28 | 15.94 |
| | | | | 0.4 | 14560.15 | 434.25 | 900$ | 2.42 | 17.7 |
| | | | | 0.6 | 15907.69 | 438.97 | 900$ | 1.27 | 25.01 |
| | | | 0.75 | 0.2 | 17747.93 | 429.85 | 900$ | 3.75 | 22.32 |
| | | | | 0.4 | 18556.11 | 508.73 | 900$ | 2.70 | 25.14 |
| | | | | 0.6 | 18155.88 | 510.17 | 900$ | 1.79 | 22.37 |

***** - $((Z_{HAC} – Z_{CPLEX})/ Z_{CPLEX}) * 100\%$

**@** - $((Z_{GA} – Z_{CPLEX})/ Z_{CPLEX}) * 100\%$

$ - Forced termination after 900 seconds

– Failed to converge within the set time

**Table 7**: Results of Network N6

| Network ID | Number of Nodes | Number of Convoys (Node-convoy ratio) | Arc-density factor | Identical Destination Factor | Average Computational time (seconds) | | | Average Computational quality | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | CPLEX | HAC | GA | HAC** | GA@@ |
| N6 | 70 | 35 (2) | 0.25 | 0.2 | - | 639.75 | 900$ | 11.49 | 25.25 |
| | | | | 0.4 | - | 652.31 | 900$ | 13.22 | 21.83 |
| | | | | 0.6 | - | 634.44 | 900$ | 13.07 | 25.18 |
| | | | 0.50 | 0.2 | - | 639.22 | 900$ | 9.65 | 21.96 |
| | | | | 0.4 | - | 683.27 | 900$ | 7.35 | 20.46 |
| | | | | 0.6 | - | 726.97 | 900$ | 8.57 | 25.33 |
| | | | 0.75 | 0.2 | - | 684.95 | 900$ | 7.42 | 26.72 |
| | | | | 0.4 | - | 716.74 | 900$ | 8.09 | 20.56 |
| | | | | 0.6 | - | 672.84 | 900$ | 7.37 | 34.31 |
| | | 27 (3) | 0.25 | 0.2 | - | 583.03 | 900$ | 10.96 | 21.58 |
| | | | | 0.4 | - | 602.77 | 900$ | 10.62 | 27.23 |
| | | | | 0.6 | - | 609.05 | 900$ | 12.51 | 27.18 |
| | | | 0.50 | 0.2 | - | 591.71 | 900$ | 7.04 | 31.06 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | 0.4 | - | 594.04 | 900$ | 9.15 | 20.09 |
| | | | | 0.6 | - | 602.63 | 900$ | 8.34 | 27.83 |
| | | | 0.75 | 0.2 | - | 608.16 | 900$ | 6.57 | 31.74 |
| | | | | 0.4 | - | 609.51 | 900$ | 7.52 | 26.03 |
| | | | | 0.6 | - | 684.53 | 900$ | 6.03 | 31.49 |
| | | 18 (4) | 0.25 | 0.2 | - | 510.24 | 900$ | 10.48 | 29.93 |
| | | | | 0.4 | - | 570.23 | 900$ | 12.81 | 20.73 |
| | | | | 0.6 | - | 571.12 | 900$ | 10.67 | 22.48 |
| | | | 0.50 | 0.2 | - | 606.46 | 900$ | 9.52 | 28.75 |
| | | | | 0.4 | - | 679.58 | 900$ | 8.94 | 24.88 |
| | | | | 0.6 | - | 608.33 | 900$ | 6.41 | 30.93 |
| | | | 0.75 | 0.2 | - | 636.43 | 900$ | 7.98 | 25.38 |
| | | | | 0.4 | - | 604.51 | 900$ | 8.33 | 29.03 |
| | | | | 0.6 | - | 693.89 | 900$ | 6.88 | 34.56 |

**** -** $((Z_{HAC} - Z_{LAGRANGEAN}) / Z_{LAGRANGEAN}) * 100\%$

**@@ -** $((Z_{GA} - Z_{LAGRANGEAN}) / Z_{LAGRANGEAN}) * 100\%$

$ - Forced termination after 900 seconds

– Failed to converge within the set time

**Table 8:** Results of Network N7

| Network ID | Number of Nodes | Number of Convoys (Node-convoy ratio) | Arc-density factor | Identical Destination Factor | Average Computational time (seconds) | | | Average Computational quality | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | CPLEX | HAC | GA | HAC** | GA@@ |
| N7 | 85 | 43 (2) | 0.25 | 0.2 | - | 794.45 | 900$ | 13.32 | 31.4 |
| | | | | 0.4 | - | 818.59 | 900$ | 12.83 | 24.75 |
| | | | | 0.6 | - | 807.12 | 900$ | 12.56 | 32.35 |
| | | | 0.50 | 0.2 | - | 863.63 | 900$ | 7.42 | 20.93 |
| | | | | 0.4 | - | 787.03 | 900$ | 8.63 | 24.93 |
| | | | | 0.6 | - | 815.63 | 900$ | 7.19 | 18.89 |
| | | | 0.75 | 0.2 | - | 892.08 | 900$ | 8.25 | 22.62 |
| | | | | 0.4 | - | 869.15 | 900$ | 8.49 | 31.07 |
| | | | | 0.6 | - | 889.32 | 900$ | 8.80 | 24.14 |
| | | 29 (3) | 0.25 | 0.2 | - | 703.52 | 900$ | 13.31 | 21.08 |
| | | | | 0.4 | - | 747.95 | 900$ | 10.62 | 28.35 |
| | | | | 0.6 | - | 721.01 | 900$ | 11.79 | 23.47 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | 0.50 | 0.2 | - | 740.67 | 900$ | 8.44 | 31.57 |
| | | | | 0.4 | - | 744.63 | 900$ | 6.30 | 22.15 |
| | | | | 0.6 | - | 813.77 | 900$ | 9.78 | 23.03 |
| | | | 0.75 | 0.2 | - | 783.29 | 900$ | 8.17 | 15.18 |
| | | | | 0.4 | - | 802.93 | 900$ | 9.92 | 23.02 |
| | | | | 0.6 | - | 830.79 | 900$ | 7.20 | 24.74 |
| | | 22 (4) | 0.25 | 0.2 | - | 724.78 | 900$ | 14.72 | 23.51 |
| | | | | 0.4 | - | 689.16 | 900$ | 10.04 | 33.08 |
| | | | | 0.6 | - | 705.77 | 900$ | 11.24 | 25.07 |
| | | | 0.50 | 0.2 | - | 699.85 | 900$ | 7.74 | 21.18 |
| | | | | 0.4 | - | 731.03 | 900$ | 9.16 | 28.09 |
| | | | | 0.6 | - | 736.32 | 900$ | 7.57 | 32.96 |
| | | | 0.75 | 0.2 | - | 791.23 | 900$ | 6.19 | 22.73 |
| | | | | 0.4 | - | 809.51 | 900$ | 8.39 | 25.33 |
| | | | | 0.6 | - | 811.44 | 900$ | 8.99 | 27.61 |

**** -** $((Z_{HAC} – Z_{LAGRANGEAN})/ Z_{LAGRANGEAN}) * 100\%$

**@@ -** $((Z_{GA} – Z_{LAGRANGEAN})/ Z_{LAGRANGEAN}) * 100\%$

$ - Forced termination after 900 seconds

– Failed to converge within the set time

**Table 9**: Results of Network N8

| Network ID | Number of Nodes | Number of Convoys (Node-convoy ratio) | Arc-density factor | Identical Destination Factor | Average Computational time (seconds) | | | Average Computational quality | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | CPLEX | HAC | GA | HAC** | GA@@ |
| N8 | 100 | 50 (2) | 0.25 | 0.2 | - | 900$ | 900$ | 14.34 | 29.66 |
| | | | | 0.4 | - | 900$ | 900$ | 13.28 | 32.83 |
| | | | | 0.6 | - | 900$ | 900$ | 11.87 | 21.88 |
| | | | 0.50 | 0.2 | - | 900$ | 900$ | 8.96 | 21.53 |
| | | | | 0.4 | - | 900$ | 900$ | 7.91 | 20.76 |
| | | | | 0.6 | - | 900$ | 900$ | 8.36 | 24.62 |
| | | | 0.75 | 0.2 | - | 900$ | 900$ | 6.58 | 28.53 |
| | | | | 0.4 | - | 900$ | 900$ | 7.49 | 33.78 |
| | | | | 0.6 | - | 900$ | 900$ | 7.41 | 25.67 |
| | | 34 (3) | 0.25 | 0.2 | - | 900$ | 900$ | 13.94 | 20.05 |
| | | | | 0.4 | - | 900$ | 900$ | 10.66 | 23.01 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | 0.6 | - | 900[$] | 900[$] | 13.68 | 25.17 |
| | | | 0.50 | 0.2 | - | 900[$] | 900[$] | 7.63 | 29.01 |
| | | | | 0.4 | - | 900[$] | 900[$] | 8.16 | 34.02 |
| | | | | 0.6 | - | 900[$] | 900[$] | 8.76 | 22.18 |
| | | | 0.75 | 0.2 | - | 900[$] | 900[$] | 6.57 | 23.61 |
| | | | | 0.4 | - | 900[$] | 900[$] | 8.99 | 20.82 |
| | | | | 0.6 | - | 900[$] | 900[$] | 7.01 | 20.85 |
| | | 25 (4) | 0.25 | 0.2 | - | 900[$] | 900[$] | 14.09 | 21.73 |
| | | | | 0.4 | - | 900[$] | 900[$] | 13.65 | 25.89 |
| | | | | 0.6 | - | 900[$] | 900[$] | 11.79 | 24.34 |
| | | | 0.50 | 0.2 | - | 900[$] | 900[$] | 8.01 | 20.54 |
| | | | | 0.4 | - | 900[$] | 900[$] | 8.33 | 25.67 |
| | | | | 0.6 | - | 900[$] | 900[$] | 9.92 | 37.53 |
| | | | 0.75 | 0.2 | - | 900[$] | 900[$] | 8.80 | 27.42 |
| | | | | 0.4 | - | 900[$] | 900[$] | 7.47 | 29.68 |
| | | | | 0.6 | - | 900[$] | 900[$] | 8.43 | 35.78 |

**** -** (($Z_{HAC}$ – $Z_{LAGRANGEAN}$)/ $Z_{LAGRANGEAN}$) * 100%

**@@ -** (($Z_{GA}$ – $Z_{LAGRANGEAN}$)/ $Z_{LAGRANGEAN}$) * 100%

$ - Forced termination after 900 seconds

– Failed to converge within the set time

*5.1 Effect of seed solution on the performance of HAC*

Rather than initiating the ant colony algorithm with the conventional approach of assigning a small positive trail intensity for all the edges of the network, we used a heuristic seed solution instead. With a view to study the effect of choosing a seed solution on the computational time for convergence, we choose Networks N4 and N5. In our view, these are neither small nor big networks and adequately represent realistic size problem scenarios. The purpose of this exercise is to investigate whether choosing a seed solution over a random solution or small positive trail intensities results in saving of computational time or not. For this, the algorithm is run with both Scenarios A and B, described below, until the solution quality is same as that of the HAC.

*Scenario A*: All the edges of the network are assigned a small positive pheromone trail intensity of 0.25 units.

*Scenario B*: A random feasible solution is used as a seed solution. As discussed earlier, edges that are part of the feasible solution are assigned a trail intensity of 2500/L whereas the rest of the edges are assigned 50% of 2500/L, with *L* representing the objective function value of the random feasible solution.

For the sake of brevity, we report only the average of all the results obtained pertaining to both the networks.

**Table 10:** Effect of seed solution on computational time

| Network | Average Computational Time (Seconds) | | |
|---|---|---|---|
| | **HAC** | **Scenario A** | **Scenario B** |
| N4 | 204.06 | 311.74 | 298.33 |
| N5 | 466.55 | 611.29 | 545.41 |

From Table 10, it is evident that initiating the HAC with a good heuristic solution always results in saving of computational effort to the extent of 50%. In the absence of a heuristic solution, even choosing a random feasible solution to initialize the trail intensities appears to be a good strategy for quicker convergence of the algorithm.

*5.2 Effect of hybridization on the performance of HAC*

In the present work, the ant colony algorithm is hybridized with a local search procedure for improvement in the solution quality. To investigate whether hybridization helps in arriving at a better final solution and to check whether the extra computational effort required for performing local search is justifiable, we perform the following analysis. By choosing networks N4 and N5, for the same reasons mentioned earlier, we run the HAC code with and without the local search procedure by following the termination criteria of 900 seconds. The results are as follows.

**Table 11:** Effect of hybridization on computational quality

| Network | Average computational quality w.r.t optimal solution | | Average computational time for convergence | |
|---|---|---|---|---|
| | **HAC with Local search** | **HAC without Local Search** | **HAC with Local search** | **HAC without Local Search** |

| | | | | |
|---|---|---|---|---|
| N4 | 0.74% | 7.32% | 204.06 | 192.38 |
| N5 | 3.87% | 12.08% | 466.55 | 438.21 |

From Table 11, it is obvious that hybridization has a significant effect on the computational quality of the HAC solutions. Further, it is observed that both with and without local search, the algorithm converged much before the time limit of 900 seconds and most importantly took approximately the same computational time. This implies that the local search procedure does not require much computational effort. To summarize, hybridizing the ACO procedure appears to be a good idea for substantial improvement in the quality of solutions with negligible additional computational effort.

## 6. Conclusions and Scope for further work

We proposed a hybridized ant colony algorithm with local search procedure to solve the convoy movement problem. We generated hypothetical test problem instances to evaluate the efficacy of the proposed approach. The computational experiments indicate that the proposed ant colony algorithm produces promising results in less computational time. The results also suggest that the use of a good heuristic solution as a seed solution for initializing the trail intensities aids in quicker convergence of the algorithm. Lastly, the results strongly point to the need to hybridize the ant colony algorithm with local search procedures for superior performance.

Though the HAC performed well for most of the problem instances, there is one exception. Particularly for networks with lower arc densities, the performance was relatively inferior. A thorough analysis and re-designing the algorithm to improve the performance in these cases is the logical next step. No real-life problem is complete without the consideration of multiple and conflicting objectives. Hence, development of a robust ACO framework for a multi-objective CMP is another good extension of this work.

## 7. References


1. Acan, A., and Unveren, A. (2015). A great deluge and tabu search hybrid with two-stage memory support for quadratic assignment problem. Applied Soft Computing, 36, 185 – 203.
2. Blum, C., Yabar, M., and Blesa, M.J. (2008). An ant colony optimization algorithm for DNA sequencing by hybridization. Computers & Operations Research, 35(11), 3620‑3635.

3. Bovet, J., Constantin, C., & de Werra, D. (1991). A convoy scheduling problem. Discrete Applied Mathematics, 30, 1 - 14.
4. Chardaire, P., G.P. McKeown, S.A. Harrison, and S.B. Richardson (2005). Solving a Time-space network formulation for the Convoy Movement Problem. Operations Research, 53(2), 219-230.
5. Dorigo, M., and Blum, C. (2005). Ant colony optimization theory: A survey. Theoretical Computer Science, 344(2-3):243–278.
6. Dorigo, M., and Stutzle, T. (2009). Ant colony optimization: Overview and Recent Advances. Technical Report, Universit´e Libre de Bruxelles.
7. Dorigo, M., Maniezzo, V., and Colorni, A. (1996). Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics – Part B, 26(1), 29–41.
8. El-Ghazali Talbi (2009). Metaheuristics: From design to implementation. New York: Wiley
9. Gambardella, L.M., and Dorigo, M. (2000). Ant Colony System hybridized with a new local search for the sequential ordering problem. INFORMS Journal on Computing, 12(3), 237–255.
10. Goldstein, D., Shehab, T., Casse, J., & Lin, H.-C. (2010). On the formulation and solution of the convoy routing problem. Transportation Research Part E: Logistics and Transportation Review, 46, 520 - 533.
11. Gonzalez, T.F. (2007). Handbook of Approximation algorithms and Metaheuristics. Florida: Chapman and Hall/CRC Press.
12. Gopalan, R. (2015). Computational complexity of convoy movement planning problems. Mathematical Methods of Operational Research, 82(1), 31 - 60.
13. Gopalan, R., and Narayanaswamy, N. (2009). Analysis of algorithms for an online version of the convoy movement problem. Journal of the Operational Research Society, 60, 1230 - 1236.
14. Harrison, S.A. (2000). Convoy planning in a digitized battlespace. RTO IST Symposium on “New information processing techniques for military systems”, Istanbul, Turkey.
15. Harrison, S.A., and Rayward-Smith, V.J. (1999). Minimal cost linkages in graphs. Annals of Operations Research, 86, 295 – 319.

16. Lau, H. C., Agussurja, L., and Thangarajoo, R. (2008). Real-time supply chain control via multi-agent adjustable autonomy. Computers and Operations Research, 35(11), 3452 – 3464.
17. Lee, Y., McKeown, G., & Rayward-Smith, V. (1996). The convoy movement problem with initial delays. Modern Heuristic Search Methods, (pp. 215 - 236).
18. M.M.S. Abdulkader, Gajpal, Y., and ElMekkawy, T.Y. (2015). Hybridized ant colony algorithm for the multi compartment vehicle routing problem. Applied Soft Computing, 37, 196 – 203.
19. McKinzie, K., & Barnes, J. W. (2004). A review of strategic mobility models supporting the defense transportation system. Mathematical and computer modelling, 39, 839 - 868.
20. Montana, D., Bidwell, G., Vidaver, G., & Herrero, J. (1999). Scheduling and route selection for military land moves using genetic algorithms. Proceedings of the Congress on Evolutionary Computation.
21. Ozbakir, L., Baykasoglu, A., Gorkemli, B., and Gorkemli, L. (2011). Multiple-colony ant algorithm for parallel assembly line balancing problem. Applied Soft Computing, 11(3), 3186 – 3198.
22. Rajendran, C., and Ziegler, H. (2004). Ant colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. European Journal of Operational Research, 155, 426-438.
23. Ram Kumar, P.N., and Narendran, T.T. (2008). Integer programming formulation for convoy movement problem. International Journal of Intelligent Defence Support Systems, 1, 177 - 188.
24. Ram Kumar, P.N., and Narendran, T.T. (2009a). A mathematical approach for variable speed convoy movement problem. Defense & Security Analysis, 25, 137 - 155.
25. Ram Kumar, P.N., and Narendran, T.T. (2011). On the usage of lagrangean relaxation for the convoy movement problem. Journal of the Operational Research Society, 62, 722 - 728.
26. Ram Kumar, P.N., Narendran, T.T., and Sivakumar, A.I. (2009b). Bi-criteria convoy movement problem. Journal of Defense modeling and Simulation: Applications, Methodology, Technology, 6(3), 151 - 164.
27. Robinson, E.M., and Leiss, E.L. (2006). Applying Genetic algorithms to Convoy scheduling, in IFIP International Federation for Information Processing, Volume 217, Artificial Intelligence in Theory and Practice, ed. M. Bramer, (Boston: Springer), pp. 315-323.

28. Sadeghnejad-Barkousaraie, A., Batta, R., & Sudit, M. (2016). Convoy Movement Problem: A Civilian Perspective. Technical Report, State University of New York.
29. Schank, J., Mattock, M., Sumner, G., Greenberg, I., Rothenberg, J., & Stucker, J. P. (1991). A review of strategic mobility models and analysis. Technical Report DTIC Document.
30. Tuson, A.L., and Harrison, S.A. (2005). Problem difficulty of real instances of convoy planning. Journal of the Operational Research Society, 56, 763 – 775.

Research Office

Indian Institute of Management Kozhikode

IIMK Campus P. O.,

Kozhikode, Kerala, India,

PIN - 673 570

Phone: +91-495-2809238

Email: research@iimk.ac.in

Web: https://iimk.ac.in/faculty/publicationmenu.php