A Gossip Algorithm for Heterogeneous Multi-Vehicle Routing Problems

Mauro Franceschelli * **Daniele Rosa** * **Carla Seatzu** *
Francesco Bullo **

* *Dep. of Electrical and Electronic Engineering, Univ. of Cagliari, Italy (e-mail: {mauro.franceschelli, daniele.rosa,seatzu@diee.unica.it})*
** *Dep. of Mechanical Engineering, Univ. of Santa Barbara, California, USA (e-mail: bullo@engineering.ucsb.edu)*

Abstract: In this paper we address the heterogeneous multi-vehicle routing problem by proposing a distributed algorithm based on gossip. We consider the case where a set of tasks arbitrarily distributed in a plane, each with a service cost, have to be served by a set of mobile robots, each with a given movement speed and task execution speed. Our goal is to minimize the maximum execution time of robots.

1. INTRODUCTION

The traveling salesman problem (TSP) is a well known topic of research. Interesting surveys can be found in Lawler et al. [1985], Gutin and Punnen [2002], Laporte [1992a]. This problem has received great attention for both its theoretical implications and its several practical applications such as vehicle routing. Interesting examples can be found in Toth and Vigo [2002], Laporte [1992b]. Several extensions to this problem have been proposed by considering at first more than one salesman as in Carlsson et al. [2009], then introducing several additional constraints and objectives to better suit practical applications such as the multi-vehicle routing problem (MVRP) with a variable number of vehicles, finite load capacity, service time windows and several more as in Bektas [2006], Pisinger and Ropke [2007]. Finally, several extensions explore a dynamic setting in which multiple vehicles serve a dynamic number of tasks as discussed in Bullo et al. [2011b].

In this paper we are interested in an instance of the MVRP that we call heterogeneous MVRP (HMVRP) where: the number n of vehicles is given a priori, a set $\mathcal{K}$ is given containing k tasks arbitrarily distributed in a plane, to each task is assigned a servicing cost, each vehicle is characterized by a movement speed and a task execution speed.

It has been shown in Carlsson et al. [2009] that when comparing the length of the optimal tour of one vehicle that visits all tasks locations with the multiple vehicle case, the maximum length of the tours for the multiple vehicle case is proportional to the tour length of the single vehicle case and proportionally inverse to the number of vehicles. Both upper and lower bounds with such scaling were given.

In this paper we extend the result in Carlsson et al. [2009] by considering execution times instead of tour lengths to account for vehicles of different speeds, tasks with arbitrary execution cost and vehicles with different task execution speeds. We provide upper and lower bounds to the optimal solution as function of the single vehicle optimal tour length to put in evidence how the performance is affected by the number of vehicles.

Our objective is to propose a distributed and asynchronous algorithm for the HMVRP based on the iterative optimization of the local task assignment between pairs of vehicles and provide deterministic bounds to its performance.

The proposed approach to the HMVRP not only addresses the issue in a distributed fashion easy to implement in a networked system, but allows to scale the size and exponential complexity of this problem with respect to the ratio k/n between the number of tasks and vehicles instead of k as in the centralized approach.

Summarizing, three are the main contributions of this paper.

- We formalize the centralized problem in terms of a mixed integer linear programming (MILP) problem and extend the bounds in Carlsson et al. [2009] for the multi TSP to the HMVRP.
- We propose a distributed algorithm based on gossip to solve the HMVRP and characterize some of its properties.
- We provide simulations that show that the proposed algorithm attains a constant factor approximation of the optimal solution with respect to the number of vehicles.

2. PROBLEM STATEMENT

Consider a set $\mathcal{N}$ of n mobile robots scattered in a connected region $\mathcal{R}$. Let $\mathcal{K}$ be a set of k tasks scattered in the same region $\mathcal{R}$, that should be assigned to robots to be executed.

Robots move at different speeds and have different execution speeds of tasks. Tasks have different costs. In particular, the following notation is used:

⋆ This work has been partially supported by the European Community's Seventh Framework Programme under project HYCON2 (Grant Agreement n. FP7-ICT-2009-5/N.257462.) and in part by ARO grant W911NF-11-1-0092.

- v_r is the speed of robot R_r,
- w_r is the task execution speed of robot R_r,
- v_{min} (v_{max}) is the minimum (maximum) speed of robots,
- w_{min} (w_{max}) is the minimum (maximum) task execution speed of robots,
- c_i is the cost of the i-th task,
- c_{min} (c_{max}) is the minimum (maximum) cost of tasks.

Moreover, d_{max} is the maximum length of the shortest path between any two points in the region $\mathcal{R}$.

Robots are supposed to first coordinate themselves to improve their task assignment. Once no further improvement can be made they stop the coordination phase and start to serve the tasks autonomously.

To use a notation that is standard in the literature, we assume that robots are initially positioned in depots and should go back to them after the execution of tasks. The set of depots is called $\mathcal{D}$ and the generic r-th depot is D_r.

Now, if $\mathcal{K}_r$ denotes the set of tasks assigned to robot R_r, our goal is that of minimizing the following objective function:

$$J = \max_{r \in \mathcal{N}} J_r = \left(\frac{TSP(\mathcal{K}_r \cup \{D_r\})}{v_r} + \frac{\sum_{i \in \mathcal{K}_r} c_i}{w_r} \right) \quad (1)$$

where TSP $(\mathcal{K}_r \cup \{D_r\})$ is the minimum TSP tour length of robot R_r that, initially positioned in D_r, visits all tasks in $\mathcal{K}_r$ and go back to D_r.

In simple words we want to minimize the maximum execution time of the n robots that have to visit and execute all tasks assigned to them, guaranteeing that each task is executed by exactly one robot.

The above problem can be seen as a generalization of the classical multi-TSP problem. First, because we are also assuming that tasks should not only be visited by the robots, but should be processed by them. Secondly, because the optimization is carried out over an heterogeneous network due to the heterogeneity of the agents and the tasks. Similar problems have been recently addressed in the literature, see e.g. Carlsson et al. [2009], but to the best of our knowledge, never under the assumption of heterogeneous agents and tasks.

Let us conclude this section with the introduction of some notation that will be used in the remaining of the paper. Let $\mathcal{K}_r$ be the set of tasks assigned to robot R_r. We denote as $\tilde{\mathcal{K}}_r$ the ordered set with the same elements of $\mathcal{K}_r$, but whose ordering specifies the order in which tasks in $\mathcal{K}_r$ are visited by robot R_r.

Finally, let $\tilde{\mathcal{K}} = \{\tilde{\mathcal{K}}_1, \ldots, \tilde{\mathcal{K}}_n\}$ be an ordered set of n ordered sets, that summarizes the generic solution of the considered tasks allocation problem. The set $\tilde{\mathcal{K}}$ is called *network state*.

3. OPTIMAL CENTRALIZED SOLUTION

In this section we first discuss a centralized strategy that leads to an optimal solution of the above task assignment problem. Such an approach is based on mixed linear integer programming (MILP). Then we provide a characterization of the optimal solution in terms of an upper and a lower bound on the optimal value of the objective function. This will be useful when evaluating the effectiveness of the decentralized approach proposed in the next section.

To represent all possible directed tours of n robots, let us define a complete directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ where:

- $\mathcal{V} = \mathcal{N} \cup \mathcal{K}$ is the set of $n + k$ nodes;
- $\mathcal{E} = (\mathcal{N} \cup \mathcal{K}) \times (\mathcal{N} \cup \mathcal{K})$ is the set of $(n + k)^2$ edges representing directed paths from the depots in which robots are initially placed to tasks, and viz, and from tasks to tasks [1] .

Moreover, we define the following binary variables that completely identify a task allocation and the order in which tasks are executed by robots. In simple words they completely identify a network state $\tilde{\mathcal{K}}$. Note that, since we want to minimize the total execution times of robots, we always assume that distances among tasks, and among tasks and depots, are covered through straight lines.

- We assign n binary variables x_{ir} to each node $i \in \mathcal{V}$; here $r \in \mathcal{N}$: if $i \in \mathcal{N}$, $x_{ir} = 1$ means that robot R_r starts its tour from node i, while if $i \in \mathcal{K}$, $x_{ir} = 1$ means that task i is executed by robot R_r.
- We assign n binary variables y_{ijr} to each edge $(i, j) \in \mathcal{E}$; here $r \in \mathcal{N}$: $y_{ijr} = 1$ means that robot R_r goes directly from node i to node j in its path.

Moreover, we introduce the following cost coefficients.

- We assign n costs $c_{ir} = c_i / w_r$ to each node $i \in \mathcal{K}$; here $r \in \mathcal{N}$: c_{ir} represents the execution time of task i by robot R_r with an execution speed of w_r.
- We assign n costs $d_{ijr} = l_{ij} / v_r$ to each edge $(i, j) \in \mathcal{E}$; here $r \in \mathcal{N}$: d_{ijr} represents the spent by robot R_r to pass the length l_{ij} of edge (i, j) with speed v_r.

***Proposition** 3.1.* Let us consider the allocation problem formalized in Section 2. An optimal solution can be computed solving the following MILP problem:

$$\begin{cases} J = \min \lambda \\ \text{s.t.} \\ \sum_{i \in \mathcal{K}} x_{ir} c_{ir} + \sum_{(i,j) \in \mathcal{E}} d_{ijr} y_{ijr} < \lambda, & \forall r \in \mathcal{N} & (a) \\ x_{rr} = 1, & \forall r \in \mathcal{N} & (b) \\ \sum_{r \in \mathcal{N}} x_{ir} = 1, & \forall i \in \mathcal{K} & (c) \\ \sum_{j \in \mathcal{V}} y_{jir} = \sum_{j \in \mathcal{V}} y_{ijr} = x_{ir}, & \forall i \in \mathcal{V},\ \forall r \in \mathcal{N} & (d) \\ \sum_{i \notin \mathcal{S}} \sum_{j \in \mathcal{S}} y_{ijr} \geq x_{qr} & \forall \mathcal{S} \subseteq \mathcal{K}, & \\ & \forall q \in \mathcal{S},\ \forall r \in \mathcal{N} & (e) \\ \lambda \in \mathbb{R} & & (f) \\ x_{ir} \in \{0, 1\} & \forall i \in \mathcal{V},\ \forall r \in \mathcal{N} & (g) \\ y_{ijr} \in \{0, 1\} & \forall (i, j) \in \mathcal{E},\ \forall r \in \mathcal{N}. & (h) \end{cases}$$

Proof: The proof is carried out via a detailed explanation of all the constraints and the objective function.

— *Constraints (a)* and *objective function:* The left hand side term of (a) is equal to the total execution time of robot R_r. Thus, given the objective function, constraints (a) aim to minimize the maximum execution time of robots.

[1] In the ets $\mathcal{V}$ and $\mathcal{E}$ the generic r-th depot is identified via the r-th element in $\mathcal{N}$. This has been done for clearity of presentation as it will appear in the following.

— *Constraints (b):* These constraints force each robot to move from its initial position (depot).

— *Constraints (c):* Each task i must be executed by exactly one robot.

— *Constraints (d):* If robot R_r executes task i, it must arrive at node i in some way and at the end of the execution has to leave it. The same holds if node i models a depot, i.e., $i \in \mathcal{N}$.

— *Constraints (e):* Each robot R_r has to make a single connected tour visiting all its tasks, so we have to exclude all the disjoint paths. In words constraint (e) relative to robot R_r, imposes that if robot R_r executes a task $i \in \mathcal{S} \subseteq \mathcal{K}$, there must be an edge passed by R_r to enter in $\mathcal{S}$. These constraints are named *Subtour Elimination Constraints* (SEC) and are typical of vehicle routing problems and TSP models Bektas [2006]. □

The number of unknowns in the MILP (3.1) is equal to

$$N = n(n+k)^2 + n(n+k) + 1 = \mathcal{O}(n^3 + nk^2 + n^2k).$$

The total number of constraints is $\mathcal{O}(n^2k + nk2^k)$. Indeed we have n constraints of type (a), n constraints of type (b), k constraints of type (c), $(n+k)n$ constraints of type (d), and $n\sum_{i=1}^{k} i\frac{k!}{(k-i)!i!} \leq nk2^k$ constraints of type (e).

The following two theorems provide a characterization of the optimal value of the performance index J^*.

***Theorem** 3.2.* The optimal solution J^* of the objective function (1) is upper bounded by

$$J^* \leq C_{up} + D_{up} \tag{2}$$

where

$$C_{up} = \frac{1}{n}\left(\frac{TSP(\mathcal{K})}{v_{min}} + \frac{\sum_{i\in\mathcal{K}} c_i}{w_{min}}\right), \tag{3}$$

$$D_{up} = 2\frac{d_{max}}{v_{min}} + \frac{c_{max}}{w_{min}}. \tag{4}$$

Proof: The proof is based on an heuristics that can be summarized in the following main steps.

- Generate an optimal tour that visits all tasks. Obviously, if an agent with speed v_{min} and execution speed w_{min} follows the tour and executes all tasks, its service time is equal to
$$\hat{J} = \left(\frac{TSP(\mathcal{K})}{v_{min}} + \frac{\sum_{i\in\mathcal{K}} c_i}{w_{min}}\right).$$
- Divide the tour in n consecutive sub-tours using the following rule. Take a robot (e.g. R_1) at random and make it follow the route of the optimal single vehicle tour at the previous item, starting from the position of an arbitrary task. Stop it as soon as its service time $\hat{J}_1$ satisfies the condition $\hat{J}_1 \geq \hat{J}/n$. Now, since the largest cost of tasks is equal to c_{max}, the smallest execution speed of robots is w_{min}, and the time taken to travel between tasks is continuous, it is
$$\hat{J}_1 \leq \frac{\hat{J}}{n} + \frac{c_{max}}{w_{min}}.$$
Select at random a new robot (e.g. R_2) and put it at the end of the route of R_1 and repeat the same strategy, until all robots are considered. If there aren't enough tasks for the robots, simply consider null the service time for the remaining robots.
- Now, if d_{max} is the maximum length of the shortest path between any two points in the region $\mathcal{R}$, the execution time J_r of each robot R_r is such that $J_r \leq \hat{J}_r + 2d_{max}/v_{min}$. Indeed the total service time of each robot corresponds to the time it takes to complete its sub-tour along the route of the optimal single vehicle TSP, plus the time to go from its depot to its first task and go back to the depot. Therefore, it is
$$J_r \leq \frac{\hat{J}}{n} + \frac{c_{max}}{w_{min}} + 2\frac{d_{max}}{v_{min}}, \qquad \forall r \in \mathcal{N}.$$

Since the optimal solution J^* of the objective function (1) can only be smaller or equal than the solution resulting from the above heuristics, for sure it is

$$J^* \leq \max_{r\in\mathcal{N}} J_r \leq \frac{\hat{J}}{n} + \frac{c_{max}}{w_{min}} + 2\frac{d_{max}}{v_{min}} = C_{up} + D_{up}$$

thus proving the correctness of the upper bound. □

***Theorem** 3.3.* The optimal solution J^* of the objective function (1) is lower bounded by

$$J^* \geq C_{lo} - D_{lo} \tag{5}$$

where

$$C_{lo} = \frac{1}{n}\left(\frac{TSP(\mathcal{K})}{v_{max}} + \frac{\sum_{i\in\mathcal{K}} c_i}{w_{max}}\right), \tag{6}$$

$$D_{lo} = \frac{d_{max}}{v_{min}}. \tag{7}$$

Proof: Let $S_{opt} = \sum_{r\in\mathcal{N}} J_r^*$ be the sum of all the service times corresponding to an optimal tasks assignment. Since, by definition $J^* = \max_{r\in\mathcal{N}} J_r^*$, obviously it is

$$J^* \geq \frac{S_{opt}}{n}. \tag{8}$$

Now, let S_{opt}^p be the sum of the contributions to J_r^*, with $r \in \mathcal{N}$, relative to the only time spent moving from one task to another one, or from/toward the depots, without including the time spent to execute tasks.

Obviously, it is

$$S_{opt} \geq S_{opt}^p + \frac{\sum_{i\in\mathcal{K}} c_i}{w_{max}}. \tag{9}$$

Moreover, trivially generalizing the result in Carlsson et al. [2009] to the case of heterogeneous robots, we have that

$$S_{opt}^p + \frac{TSP(\mathcal{D})}{v_{min}} \geq \frac{TSP(\mathcal{D}\cup\mathcal{K})}{v_{max}} \geq \frac{TSP(\mathcal{K})}{v_{max}} \tag{10}$$

or equivalently

$$S_{opt}^p \geq \frac{TSP(\mathcal{K})}{v_{max}} - \frac{TSP(\mathcal{D})}{v_{min}}. \tag{11}$$

By equations (9) and (11) it follows that

$$\begin{aligned} S_{opt} &\geq \frac{TSP(\mathcal{K})}{v_{max}} - \frac{TSP(\mathcal{D})}{v_{min}} + \frac{\sum_{i\in\mathcal{K}} c_i}{w_{max}} \\ &\geq \frac{TSP(\mathcal{K})}{v_{max}} - n\frac{d_{max}}{v_{min}} + \frac{\sum_{i\in\mathcal{K}} c_i}{w_{max}}. \end{aligned} \tag{12}$$

Finally, by equations (8) and (12), it is

$$\begin{aligned} J^* \geq \frac{S_{opt}}{n} &= \frac{1}{n}\left(\frac{TSP(\mathcal{K})}{v_{max}} + \frac{\sum_{i\in\mathcal{K}} c_i}{w_{max}}\right) - \frac{d_{max}}{v_{min}} \\ &= C_{lo} - D_{lo} \end{aligned} \tag{13}$$

thus proving the statement. □

4. DECENTRALIZED SOLUTION

In this section we first propose a decentralized approach to solve the task allocation problem in Section 2 that is based on gossip. Then, a comparison among the computational complexity of the proposed algorithm and the centralized algorithm is provided. Convergence properties of the gossip algorithm are discussed. Finally, some characterizations of the solution obtained via the decentralized approach are proposed.

4.1 MILP Gossip algorithm

The idea of the proposed decentralized algorithm is that robots locally balance their loads according to a gossip interaction rule under the following main assumption:

(A) All robots may interact with all the other robots.

Starting from an initial task assignment, e.g., assuming that robots have the same number of tasks, a couple of robots is selected at random. Selected robots optimally balance their load; a new couple of robots is selected and so on, until no better balancing among robots can be obtained. This can be summarized in the following algorithm where T_{max} denotes a maximum number of steps to be executed that is assumed to be large enough that no further improvement of the objective function can be obtained.

***Algorithm** 1.* (MILP Gossip algorithm).

(1) Tasks are initially assigned to robots so that each robot has either k/n or $k/n+1$ tasks.
(2) Let $t=0$.
(3) While $t \leq T_{max}$
  (a) Choose at random two robots r and q. Let them solve the MILP (3.1) where $\mathcal{N} = \{r, q\}$ and $\mathcal{K} = \mathcal{K}_r \cup \mathcal{K}_q$.
  (b) If the new tasks assignment leads to a smaller total execution time, then update the assignments of robots r and q accordingly,
    else leave them unchanged.
  (c) Let $t = t+1$ and go back to Step 3.
(4) All robots process their own set of tasks following the order specified by the optimal local solution. ■

Note that at Step 3 no specific stopping criterion has been given. Obviously, different solutions can be adopted. One is that of using a central coordinator that keeps track of the fact that no better load balancing can be obtained among robots. Decentralized criteria can also be given that are not formalized in this preliminary paper on this topic. In particular, as future work we plan to exploit the ideas in Franceschelli et al. [2011] based on the fact that by assuming a ring communication topology, it is possible to ensure convergence to an invariant condition in finite time.

4.2 Computational complexity of the local optimization

Let us now discuss the advantages in terms of computational complexity coming from local optimizations using Algorithm 1 with respect to central optimization.

To this aim, let us first present some preliminary results. In particular, the following proposition ensures that when the number of iterations of Algorithm 1 increases, the optimal value of the objective function can never increase. Obviously this does not imply that an optimal solution is obtained, as shown in the following Example 5.1.

***Proposition** 4.1.* Let $J_{gossip}(t)$ be the maximum execution time of robots computed after t iterations of Algorithm 1. For any $t \geq 0$, it is $J_{gossip}(t+1) \leq J_{gossip}(t)$.

Proof: Let R_r and R_q be the two robots selected at time $t+1$. By Algorithm 1 this means that only the tasks allocation of such robots may change, while the load of all the other robots keeps unaltered. Now, since at Step 3.a of Algorithm 1 tasks are assigned to robots R_r and R_q so as to minimize the maximum execution time among them, this implies that the maximum execution time among R_r and R_q either decreases or it keeps unaltered at time $t+1$. Moreover, the maximum execution time among all robots may decrease at time $t+1$ if and only if either R_r or R_q, or both, are the robots to which it corresponds the maximum execution time among all robots at time t. Indeed with no loss of generality, we may assume that R_r is the "critical" robot at time t, i.e., the robot to which it corresponds the maximum execution time among all robots at time t. Three different cases may occur at time $t+1$, after the new tasks allocation. First, R_r may still be the robot with the maximum execution time, but in such a case for sure, its execution time cannot be larger than that at time t. Secondly, robot R_q may be at time $t+1$ the robot with the maximum execution time but for sure its execution time cannot be larger than that of robot R_r at time t. Finally, at time $t+1$, neither to R_r nor to R_q it corresponds the maximum execution time among robots. This implies that a third robot, e.g., R_p, has become the critical one at time $t+1$. In any case for sure its execution time is smaller than that of robot R_r at time t, since by assumption robot R_r was the critical robot at time t. □

Let us now provide an upper bound on the value of the maximum execution time of robots resulting from Algorithm 1 at a generic iteration t. To this aim, we first recall some deterministic upper bounds to the maximum length of the shortest path (SP) between a set $\mathcal{K}$ of k locations in a unit square area, that are due to Few [1955] and Karloff [1989], respectively:

$$SP(\mathcal{K}) \leq \sqrt{2}\sqrt{k} + 7/4, \tag{14}$$

and

$$SP(\mathcal{K}) \leq 0.984\sqrt{2}\sqrt{k} + 11. \tag{15}$$

To the best of our knowledge the above two upper bounds are the best actually proposed in the literature. Moreover, we cannot a priori say which of the above bounds is the most strict one. Indeed the bound in Karloff [1989] has a smaller multiplicative factor with respect to Few [1955], but has a larger additive constant. In the following, we focus on upper bound (14), but obviously similar results can be repeated considering (15).

***Proposition** 4.2.* Let $J_{gossip}(t)$ be the maximum execution time of robots computed after t iterations of Algorithm 1, then $\forall\ t \geq 0$ it is

$$J_{gossip}(t) \leq \left(\sqrt{2}\sqrt{\frac{k}{n}+2}+\frac{7}{4}+\sqrt{2}\right)\frac{d_{max}}{v_{min}}+\left(\frac{k}{n}+1\right)\frac{c_{max}}{w_{min}}.$$

Proof: By Algorithm 1 at time $t = 0$ the maximum number of tasks that can be assigned to a robot is equal to $k/n+1$. Moreover, since each robot starts its path from its depot and has to come back to it, then by equation (14), for any $r \in \mathcal{N}$ it is

$$TSP(\mathcal{K}_r(0) \cup \{D_r\}) \leq \left(\sqrt{2}\sqrt{\frac{k}{n}+2}+\frac{7}{4}+\sqrt{2}\right) d_{max}. \tag{16}$$

Note that the additional term $\sqrt{2}$ between parenthesis comes from the fact that to form a Euclidean TSP tour from a path in a unit square it is sufficient to connect the start and end point to form a cycle, thus increasing the size of the path of at most $\sqrt{2}$ in the unit square. Moreover, d_{max} comes from the fact that in our problem statement depots and robots are not distributed in a square of unitary edge, but in a region $\mathcal{R}$ that is contained in a square of edge d_{max} being by definition d_{max} the maximum length of the shortest path between any two points in $\mathcal{R}$.

Finally, since by assumption $\sum_{i\in\mathcal{K}_r(0)} c_i \leq \left(\frac{k}{n}+1\right) c_{max}$, it follows that

$$J_{gossip}(0) \leq \left(\sqrt{2}\sqrt{\frac{k}{n}+2}+\frac{7}{4}+\sqrt{2}\right)\frac{d_{max}}{v_{min}}+\left(\frac{k}{n}+1\right)\frac{c_{max}}{w_{min}}$$

that proves the statement being by Proposition 4.1 $J_{gossip}(t) \leq J_{gossip}(0)$ for all $t \geq 0$. □

Let us now provide a proposition that characterizes the maximum number of tasks that are assigned to robots at a generic iteration t of Algorithm 1.

***Proposition** 4.3.* Let $K_{max}(t) = \max_{r\in\mathcal{N}} |\mathcal{K}_r(t)|$ be the maximum number of tasks that are assigned to robots at a generic iteration t of Algorithm 1. For any $t \geq 0$ it is:

$$K_{max}(t) \leq \frac{w_{max}}{c_{min}}\left[\left(\sqrt{2}\sqrt{\frac{k}{n}+2}+\frac{7}{4}+\sqrt{2}\right)\frac{d_{max}}{v_{min}} + \left(\frac{k}{n}+1\right)\frac{c_{max}}{w_{min}}\right]. \tag{17}$$

Proof: By Proposition 4.2, for all $t \geq 0$, it holds

$$J_{gossip}(t) \leq \left(\sqrt{2}\sqrt{\frac{k}{n}+2}+\frac{7}{4}+\sqrt{2}\right)\frac{d_{max}}{v_{min}}+\left(\frac{k}{n}+1\right)\frac{c_{max}}{w_{min}}. \tag{18}$$

Now, it is

$$J_{gossip}(t) \geq \frac{K_{max}(t)c_{min}}{w_{max}} \tag{19}$$

since the execution time of $K_{max}(t)$ tasks is greater or equal than that we have if such tasks are at a null distance from the robot that has to process them, all tasks have a cost equal to c_{min} and the robot who process them has an execution speed equal to w_{max}. By equations (18) and (19) the statement of the proposition follows. □

An important remark needs to be done. The above proposition provides an upper bound on the maximum number of tasks that can be assigned to a robot at any iteration. For particular values of the parameters it may happen that the upper bound given by Proposition 4.3 is not significant being it larger than k. However, this only occurs for very particular cases, while for most of the significant and general situations where the number of tasks is sufficiently large, robots and tasks are sufficiently distributed in $\mathcal{R}$ and their costs and speeds are in reasonable ratio, Proposition 4.3 enables us to conclude that

$$K_{max}(t) = \mathcal{O}(k/n).$$

Now, since local optimization considers two robots at a time, the number of tasks that are involved in a local optimization is surely smaller or equal than $2K_{max}(t)$. This means that the number of unknowns of the MILP that should be solved at the generic iteration t of Algorithm 1 is

$$N_{gossip} = \mathcal{O}(k^2/n^2)$$

rather than $N = \mathcal{O}(n^3 + nk^2 + n^2k)$ as in the centralized case. Moreover, the number of constraints is $\mathcal{O}(k2^{k/n}/n)$ rather than $\mathcal{O}(n^2k + nk2^k)$ as in the centralized case.

4.3 Finite time and almost sure convergence

We now introduce two definitions to formalize two important properties of gossip communication schemes, namely *deterministic persistence* and *stochastic persistence*. Similar definitions have been recently proposed in Bullo et al. [2011a] even if in a different context.

As usual in this framework, we assume that the possible interactions among agents are modeled by an undirected graph $G = \{V, E\}$ where agents correspond to vertices, and an edge exists if and only if the interaction among the agents corresponding to the incidence nodes is possible. Obviously, assumption (A) implies that in our case it is $E = V \times V$. At each iteration t of the gossip algorithm a different edge is selected. In the following we denote as $e(t)$ the edge selected at time t, while the set of edges selected in the time interval $[t_1, t_2]$ is denoted as $\bar{e}(t_1, t_2)$, i.e., it is

$$\bar{e}(t_1, t_2) = \bigcup_{t=t_1}^{t_2} e(t).$$

***Definition** 4.4.* (Deterministic persistence). A gossip communication scheme is said to be *deterministically persistent* if $\forall t \geq 0$ there exists a finite $T > 0$ such that

$$\forall e' \in E, \qquad Pr(e' \in \bar{e}(t, t+T)) = 1$$

or equivalently, $\bar{e}(t, t+T) = E$. ■

Deterministic persistence implies that, if we consider a finite but sufficiently large time interval, then for sure all arcs are selected at least once during such an interval.

***Definition** 4.5.* (Stochastic persistence). A gossip communication scheme is said to be *stochastically persistent* if $\forall t \geq 0$ there exists a finite $T > 0$ and a probability $p \in (0, 1)$ such that

$$\forall e' \in E, \qquad Pr(e' \in \bar{e}(t, t+T)) \geq p$$

where $Pr(\cdot)$ denotes a probability. ■

In simple words, stochastic persistence implies that, if we consider a finite but sufficiently large time interval, then

each edge has a probability greater or equal than a finite value p of being selected during such an interval.

***Theorem** 4.6.* Let $\tilde{\mathcal{K}}(t)$ be the network state resulting at time t from the execution of Algorithm 1. If the gossip communication scheme satisfies the *deterministic persistence* property then, for every initial task assignment, there exists a network state $\tilde{\mathcal{K}}^*_{gossip}$ and a finite time $T > 0$ such that $\tilde{\mathcal{K}}(t) = \tilde{\mathcal{K}}^*_{gossip}$, for all $t \geq T$.

Proof: Let us present some preliminary comments.

— First, $\tilde{\mathcal{K}}^*_{gossip}$ is an invariant network state for the state evolution following Algorithm 1. This follows from Step 3.b of Algorithm 1.

— Secondly, if at a given time the network state is updated then the previous network state is no more visited during the algorithm evolution. This also follows from Step 3.b of Algorithm 1 and the monotonicity property expressed by Proposition 4.1.

— Thirdly, the number $N_{n,k}$ of admissible network states is finite since both the number of robots and the number of tasks are finite.

Now, with no loss of generality we assume that at the initial time $t = 0$ it is $\tilde{\mathcal{K}}_r \neq \tilde{\mathcal{K}}^*_{gossip,r}$ for all $r = 1, \ldots, n$, i.e., no robot is in its final assignment. If the communication scheme among agents is deterministically persistent, since the graph modeling the possible interactions among robots is fully connected and the number $N_{n,k}$ of admissible network states is finite, then for sure after some finite time T_0 the robot with the maximum cost in the final assignment reaches its final assignment. Let R_r be such a robot. By Step 3.b of Algorithm 1 this implies that the assignment of R_r is no more changed during the algorithm evolution, i.e., $\tilde{\mathcal{K}}_r(t) = \tilde{\mathcal{K}}^*_{gossip,r}$ for all $t \geq T_0$.

Analogously, after some further finite time T_1 the final assignment is reached by the robot with the second largest cost, and so, until all robots have reached their final assignment. Since all T_i's are finite, this proves that the final network state $\tilde{\mathcal{K}}^*_{gossip}$ is reached in a finite time $T = \sum_{i=1}^{n} T_i$. □

***Theorem** 4.7.* Let $\tilde{\mathcal{K}}(t)$ be the network state resulting at time t from the execution of Algorithm 1. If the gossip communication scheme satisfies the *stochastic persistence* property, then, for every initial task assignment, there exists a network state $\tilde{\mathcal{K}}^*_{gossip}$ and almost surely a finite time $T > 0$ such that $\tilde{\mathcal{K}}(t) = \tilde{\mathcal{K}}^*_{gossip}$ for all $t \geq T$, i.e., the network state converges almost surely in finite time to $\tilde{\mathcal{K}}^*_{gossip}$.

Proof: We prove this theorem following the same arguments an in Carli et al. [2008]. The proof is based on verifying the following three facts:

(i) $\tilde{\mathcal{K}}^*_{gossip}$ is an invariant network state for the state evolution following Algorithm 1;
(ii) $\tilde{\mathcal{K}}(t)$ is a Markov process on a finite number of states;
(iii) starting from any initial network state $\tilde{\mathcal{K}}(0)$, there is a positive probability for the network state to reach $\tilde{\mathcal{K}}^*_{gossip}$ in a finite number of steps.

Let us now check the above three properties in order.

— (i) As already discussed in Theorem 4.6, this follows from Step 3.b of Algorithm 1.

— (ii) As already discussed in the proof of Theorem 4.6, the number of admissible network states $N_{n,k}$ is finite, being finite both the number of robots and the number of tasks. Markovianity immediately follows from the fact that subsequent random selection of edges are independent.

— (iii) This issue can be proved using similar arguments as in Theorem 4.6 with the only difference that now the communication scheme is stochastically persistent, rather than deterministically persistent. This implies that for any initial network state $\tilde{\mathcal{K}}(0)$ there is a finite probability that after some finite time T_0 the robot with the maximum cost in the final assignment reaches its final assignment, that is no more changed during the algorithm evolution. The same holds for the robot with the second largest execution cost in the final assignment, and so, until the invariant network state $\tilde{\mathcal{K}}^*_{gossip}$ is reached. Since the number of possible states is finite, item (iii) holds. □

4.4 Some characterization of the gossip solution

Unfortunately, Algorithm 1 does not guarantee the convergence to an optimal solution. However, some results can be given to characterize its solution at the equilibrium, i.e., after a number of iterations that is sufficiently large so that no better balancing among robots may be obtained. In particular, the following theorem provides a characterization of the maximum distance among the processing times of robots that have locally balanced their loads.

***Theorem** 4.8.* Let $J^*_{gossip,r}$ and $J^*_{gossip,q}$, respectively, be the total execution times of two generic robots R_r and R_q resulting from the application of Algorithm 1. It holds

$$|J^*_{gossip,r} - J^*_{gossip,q}| \leq K_{rq} = 2\frac{d^{rq}_{max}}{v^{rq}_{min}} + \frac{c^{rq}_{max}}{w^{rq}_{min}} \qquad (20)$$

where d^{rq}_{max} is the maximum distance among tasks in $\mathcal{K}_r$ and tasks in $\mathcal{K}_q$, $v^{rq}_{min} = \min\{v_r, v_q\}$, and $w^{rq}_{min} = \min\{w_r, w_q\}$.

Proof: We prove the statement by contradiction, i.e., we assume that $|J^*_{gossip,r} - J^*_{gossip,q}| > K_{rq}$. With no loss of generality, we assume that it is $J^*_{gossip,r} > J^*_{gossip,q}$. Now, let z be the task in $\mathcal{K}_r$ whose distance with respect to tasks in $\mathcal{K}_q$ is minimum. Remove z from $\mathcal{K}_r$ and put it in $\mathcal{K}_q$. Let $\tilde{J}_r$ and $\tilde{J}_q$ be the resulting execution times of robots r and q, respectively. Obviously, it is

$$\tilde{J}_q \leq J^*_{gossip,q} + \frac{c_z}{w_q} + 2\frac{d^{rq}_{max}}{v_q} = J^*_{gossip,q} + K_{rq} \qquad (21)$$

where the inequality follows from the fact that the optimal TSP of robot q is surely smaller than the path obtained by simply adding twice the path from the closest task in $\mathcal{K}_q$ to z. Now, by the contradictory assumption, it is

$$J^*_{gossip,r} > J^*_{gossip,q} + K_{rq} \qquad (22)$$

thus (21) can be rewritten as

$$\tilde{J}_q < J^*_{gossip,r}. \qquad (23)$$

As a consequence

$$\max\{\tilde{J}_q, \tilde{J}_r\} < \max\{J^*_{gossip,q}, J^*_{gossip,r}\}. \qquad (24)$$

However, this contradicts the assumption that $J^*_{gossip,r}$ and $J^*_{gossip,q}$ are the time executions corresponding to an optimal tasks assignment, thus proving the statement. □

***Corollary** 4.9.* Let $J^*_{gossip,r}$ and $J^*_{gossip,q}$, respectively, be the total execution times of two generic robots R_r and R_q resulting from the application of Algorithm 1. It holds

$$|J^*_{gossip,r} - J^*_{gossip,q}| \leq D_{up} \tag{25}$$

where D_{up} is defined as in equation (4).

Let us now provide two theorems that give an upper bound on the maximum execution time resulting from the application of Algorithm 1.

***Theorem** 4.10.* Let J^*_{gossip} be the maximum execution time resulting from the application of Algorithm 1. It is

$$J^*_{gossip} \leq \left(\sqrt{2}\sqrt{\frac{k}{n}+2}+\frac{7}{4}+\sqrt{2}\right)\frac{d_{max}}{v_{min}}+\frac{1}{n}\frac{\sum_{i\in\mathcal{K}} c_i}{w_{min}}+D_{up}. \tag{26}$$

Proof: Let $S_{gossip}(t)$ be the sum of all J_i's at iteration t of Algorithm 1. By Proposition 4.2 it is

$$S_{gossip}(t) \leq nJ_{gossip}(t) \leq n\left(\sqrt{2}\sqrt{\frac{k}{n}+2}+\frac{7}{4}+\sqrt{2}\right)\frac{d_{max}}{v_{min}}+\frac{\sum_{i\in\mathcal{K}} c_i}{w_{min}}. \tag{27}$$

Let $J^*_{gossip,min}$ be the smallest execution time between the vehicles after the execution of Algorithm 1. By Corollary 4.9 it is $J^*_{gossip,min} \geq J^*_{gossip} - D_{up}$. Moreover, $\forall t \geq 0$ it obviously is

$$J_{gossip,min}(t) \leq \frac{1}{n}S_{gossip}(t) \tag{28}$$

thus

$$J^*_{gossip} \leq J^*_{gossip,min} + D_{up} \leq \frac{1}{n}S_{gossip}(t) + D_{up} \leq \left(\sqrt{2}\sqrt{\frac{k}{n}+2}+\frac{7}{4}+\sqrt{2}\right)\frac{d_{max}}{v_{min}}+\frac{1}{n}\frac{\sum_{i\in\mathcal{K}} c_i}{w_{min}}+D_{up}. \tag{29}$$

proving the statement. □

A different upper bound on the value of the objective function computed using Algorithm 1 is given by the following theorem.

***Theorem** 4.11.* Let J^*_{gossip} be the value of the objective function (1) resulting from Algorithm 1. It is

$$J^*_{gossip} \leq \frac{k}{n}\frac{2d_{max}}{v_{min}}+\frac{1}{n}\frac{\sum_{i\in\mathcal{K}} c_i}{w_{min}}+D_{up} \tag{30}$$

where D_{up} is defined as in equation (4).

Proof: Let S_{gossip} be the sum of all J_i's after the execution of Algorithm 1. For sure it is

$$S_{gossip} \leq k\frac{2d_{max}}{v_{min}}+\frac{\sum_{i\in\mathcal{K}} c_i}{w_{min}}. \tag{31}$$

In fact, no task assignment may lead to a maximum execution time that is larger than that obtained assuming that each robot has a speed equal to v_{min}, an execution speed equal to w_{min}, all tasks are at distance d_{max} from depots, and any robot moves from its depot to a task and come back to the depot, and again to a new task, and so on, until all tasks are visited by exactly one robot.

Now, let $J^*_{gossip,min}$ be the smallest execution time of robots after executing Algorithm 1. By equation (25) it follows that:

$$J^*_{gossip} \leq J^*_{gossip,min} + 2\frac{d_{max}}{v_{min}}+\frac{c_{max}}{w_{min}}. \tag{32}$$

Moreover, it obviously is

$$J^*_{gossip,min} \leq \frac{1}{n}S_{gossip} \tag{33}$$

thus

$$J^*_{gossip} \leq \frac{1}{n}S_{gossip} + 2\frac{d_{max}}{v_{min}}+\frac{c_{max}}{w_{min}}. \tag{34}$$

The statement is proved by simply substituting equation (31) in (34). □

Both Theorems 4.10 and 4.11 provide upper bounds on the maximum execution times of robots after the execution of Algorithm 1. Both bounds depend on the ratio k/n: for small values of such ratio, the bound given by equation (30) is smaller, while for large values of k/n the most significant bound is given by equation (26).

5. NUMERICAL SIMULATIONS

In this section we present two series of experimental results. First, we compare the solution computed using Algorithm 1 with the optimal solution computed using the centralized approach (3.1). Secondly, we analyze the value of J^*_{gossip} for different values of k and n, comparing it with the lower and upper bounds given for the centralized optimal solution.

In all the experiments robots and tasks are randomly scattered in a square whose edge is equal to 10 units. Costs of tasks are integer values uniformly randomly generated in the interval $[1, 5]$. Speeds v_i and w_i are real values uniformly randomly generated in $[1, 2]$. Finally, in the MILP gossip algorithm the edge selection is performed in a uniformly random way.

The comparison between centralized and decentralized solution has shown that in most of the cases it is $J^* = J^*_{gossip}$. In particular, in a set of 1000 experiments, in 859 cases $J^* = J^*_{gossip}$ (85.9%). In the remaining 141, we have computed the relative error e_i for each experiment i as $e_i = (J^*_{gossip,i} - J^*_i)/J^*_i$ The mean value $\bar{e}$ of the relative error in our simulations is $\bar{e} = 0.0216$, while the maximum error relative is $e_{max} = 0.196$ (19.6 %). The following example represent a case in which $J^* \neq J^*_{gossip}$.

***Example** 5.1.* Let us consider a system with $n = 3$ robots and $k = 7$ tasks. Robots are initially positioned in the XY plane as summarized in Table 5.1. This table also summarizes the position and costs of tasks and the initial task assignment. Moreover, for each robot R_r it is $v_r = w_r = 1$. Table 5.1 presents the results of the load balancing carried out using both the centralized and the decentralized approach. As it can be seen, the optimal solution of the centralized approach is better than that obtained via gossip. In particular, it is $J^* = 100.5$ and $J^*_{gossip} = 101$. ■

Let us now compare the optimal value of the performance index obtained via the gossip algorithm with the upper and lower bound of the centralized approach given by

	X	Y	Init. Assig.	c_i
Robot 1	0	0	-	-
Robot 2	30	0	-	-
Robot 3	-20	0	-	-
Task 1	20	0	Robot 1	20
Task 2	20	0	Robot 2	3
Task 3	20	0	Robot 1	1.5
Task 4	20	0	Robot 2	78
Task 5	-10	0	Robot 1	1
Task 6	-10	0	Robot 1	17.5
Task 7	-10	0	Robot 3	79.5

Table 1. Example 5.1: initial task assignment.

	Centralized $\mathcal{K}_r$	J_r^*	Gossip $\mathcal{K}_r$	$J_{r,gossip}^*$
Robot 1	$\{1, 2, 6\}$	100.5	$\{1, 3, 5, 6\}$	100
Robot 2	$\{3, 4\}$	99.5	$\{2, 4\}$	101
Robot 3	$\{5, 7\}$	100.5	$\{7\}$	99.5

Table 2. Example 5.1: simulation results.

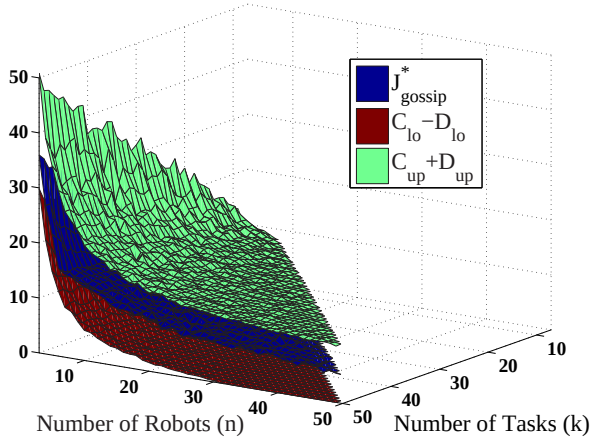

Fig. 1. J_{gossip}^* and the upper and lower bound of the centralized solution given by Theorems 3.2 and 3.3.

Theorems 3.2 and 3.3. The results of such a comparison are reported in Fig. 1 where for each couple (n, k) of n robots and k tasks, J_{gossip}^* and the two bounds are the mean values of 10 experiments. Simulation shows that the maximum service time obtained with the gossip approach remains always between the upper and the lower bound of the centralized approach.

6. CONCLUSIONS AND FUTURE WORK

In this paper we proposed upper and lower bounds for the cost of the optimal solution to the HMVRP which considers vehicles with different movement and task execution speed, and tasks with different servicing costs, extending the bounds for the multi-vehicle routing problem in Carlsson et al. [2009]. Furthermore, we proposed an algorithm based on gossip to solve the HMVRP in a distributed fashion exploiting only pairwise task exchanges between vehicles, thus greatly reducing the computational complexity required to compute a solution. The proposed method scales with exponential complexity with respect to the ratio between the number of tasks and vehicles instead of scaling with respect to the number of tasks. Some characterizations of the gossip solution are also given.

Our conjecture is that the gossip solution always lie within the bounds that characterize the optimal solution. Up to now this conjecture has only been validated via numerical simulation, while a formal proof is missing and constitutes one of our future lines of research in this topic. Such a result, if it holds, is very important because it implies that the MILP gossip algorithm provides a constant factor approximation to the optimal solution. Our future directions of research also involve the investigation of other distributed strategies, not based on MILP, to provide a constant factor approximation of the optimal solution of the HMVRP. Finally, we are investigating the definition of a stopping criterion that is completely distributed.

REFERENCES

Bektas, T. (2006). The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3), 209–219.

Bullo, F., Carli, R., and Frasca, P. (2011a). Gossip coverage control for robotic networks: Dynamical systems on the space of partitions. *SIAM Journal on Control and Optimization*. To appear.

Bullo, F., Frazzoli, E., Pavone, M., Savla, K., and Smith, S.L. (2011b). Dynamic vehicle routing for robotic systems. *Proceedings of the IEEE*, 99(9), 1482–1504.

Carli, R., Fagnani, F., Speranzon, A., and Zampieri, S. (2008). Communication constraints in the average consensus problem. *Automatica*, 44 (3), 671–684.

Carlsson, J., Ge, D., Subramaniam, A., Wu, A., and Ye, Y. (2009). Solving min-max multi-depot vehicle routing problem. *Lectures on global optimization*, 55, 31–46.

Few, L. (1955). The shortest path and the shortest road through n points. *Mathematika*, 2(2), 141–144.

Franceschelli, M., Giua, A., and Seatzu, C. (2011). Quantized consensus in Hamiltonian graphs. *Automatica*, 47(11), 2495–2503.

Gutin, G. and Punnen, A.P. (2002). *The Traveling Saleman Problem and Its Variations*, volume 12 of *Series in Combinatorial Optimization, Springer*. Kluwer Academic Publishers.

Karloff, H.J. (1989). How long can a Euclidean traveling salesman tour be? *SIAM Journal on Discrete Mathematics*, 2(1), 91–99.

Laporte, G. (1992a). The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2), 231–247.

Laporte, G. (1992b). The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3), 345–358.

Lawler, E.L., Lenstra, J.K., Rinnooy-Kan, A.H.G., and Shmoys, D.B. (1985). *The Traveling Saleman Problem: A Guided Tour of Combinatorial Optimization*. Wiley Series in Discrete Mathematics and Optimization. John Wiley and Sons.

Pisinger, D. and Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8), 2403–2435.

Toth, P. and Vigo, D. (2002). *The Vehicle Routing Problem*, volume 9 of *Monographs on Discrete Mathematics and Applications*. SIAM.