Computers & Industrial Engineering 64 (2013) 19–30

Contents lists available at SciVerse ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

A distributed multi-agent production planning and scheduling framework for mobile robots ☆

Stefano Giordani [a,*], Marin Lujak [b], Francesco Martinelli [c]

[a] Dipartimento di Ingegneria dell'Impresa, Università di Roma "Tor Vergata", Via del Politecnico 1, 00133 Roma, Italy
[b] University Rey Juan Carlos, CETINIA, Calle Tulipan s/n, 28933 Móstoles, Madrid, Spain
[c] Dipartimento di Informatica, Sistemi e Produzione, Università di Roma "Tor Vergata", Via del Politecnico 1, 00133 Roma, Italy

ARTICLE INFO

Article history:
Received 8 July 2011
Received in revised form 5 March 2012
Accepted 7 September 2012
Available online 26 September 2012

Keywords:
Production planning and scheduling
Dynamic lot sizing
Multi-robot task allocation
Multi-agent system
Optimization algorithm

ABSTRACT

Inspired by the new achievements in mobile robotics having as a result mobile robots able to execute different production tasks, we consider a factory producing a set of distinct products via or with the additional help of mobile robots. This particularly flexible layout requires the definition and the solution of a complex planning and scheduling problem. In order to minimize production costs, dynamic determination of the number of robots for each production task and the individual robot allocation are needed. We propose a solution in terms of a two-level decentralized Multi-Agent System (MAS) framework: at the first, production planning level, agents are tasks which compete for robots (resources at this level); at the second, scheduling level, agents are robots which reallocate themselves among different tasks to satisfy the requests coming from the first level. An iterative auction based negotiation protocol is used at the first level while the second level solves a Multi-Robot Task Allocation (MRTA) problem through a distributed version of the Hungarian Method. A comparison of the results with a centralized approach is presented.

1. Introduction

An external demand of a manufacturing system is generally a fluctuating stochastic process, usually known with a satisfactory accuracy only over a limited time horizon ahead. This introduces, at the strategic level, a high degree of uncertainty in the design of a production system and a supply chain where critical decisions must be taken based on aggregate and approximate information (see, e.g., Mun, 2002).

In traditional shop-floor planning, establishing a production facility requires the selection of static production machines and robot manipulators which would be suitable for long term production plans. With the advances in the development of mobile production resources, now it is possible for many products, once manufactured only by large production machines permanently tied to single locations, to be manufactured with smaller, mobile robots. One of the first such robots was presented in July 2010 by robotic producer Kuka (Bischoff et al., 2010). The shop floor layout with mobile robots makes the strategic decisions less critical with respect to the ones associated to the design of a plant where machines (i.e., production resources) are located on static positions. A dynamic layout represents in fact a less constrained facility where design decisions are postponed to the operative level and become reversible options. Furthermore, it is more effective in responding to a fluctuating external demand and can be considered, for this reason, in the same vein as other solutions adopted through the years in the manufacturing domain for the same purpose, like, among others, Flexible Manufacturing Systems (e.g., Huang and Chen, 1986), Group Technology (see, e.g., Selim et al., 1998), Holonic Manufacturing (see, e.g., Christensen, 1994), and Agile Production Systems (see, e.g., Dugnay et al., 1997).

The high degree of flexibility achieved by the proposed dynamic multi-robot layout leads to a more complex operation management which may render centralized architectures unviable. Centralized architectures in such complex environments are often impractical because of computational and communication bottleneck and the vulnerability of system failure. On the other hand, a bottom-up multi-agent modular architecture distributes computational resources and capabilities among the agents and does not suffer from the "critical point of failure" problem associated with centralized systems (see, e.g., Wooldridge, 2002). Further advantages of a decentralized multi-agent approach are modularity, decentralized knowledge bases, fault-tolerance, redundancy and extendibility, in the sense that new robots can be added to the original system without any change in the system architecture (see, e.g., Lueth and Laengle, 1994).

☆ This manuscript was processed by Area Editor T.C. Edwin Cheng.

* Corresponding author. Tel.: +39 06 72597358; fax: +39 06 72597305.
E-mail addresses: stefano.giordani@uniroma2.it (S. Giordani), lujak@ia.urjc.es (M. Lujak), martinelli@disp.uniroma2.it (F. Martinelli).

0360-8352/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cie.2012.09.004

For all the above reasons and because mobile robots are autonomous entities with limited vision and communication capacities, in this paper we propose a decentralized two-level Multi-Agent System (MAS) framework for the case where the production is executed exclusively or with the additional help of mobile robots, as shown, e.g., by Helms et al. (2002) and Tan et al. (2009). On the first, production planning level, tasks compete for the mobile resources (robots) required for their execution. Assuming that the planning time horizon is subdivided into a finite number of time periods, the objective of the production planning level is the determination of the number of the robots to be assigned in each time period to the tasks. This is done in order to minimize the total production cost for each task (with the products' demand known over all the time periods in the given time horizon). The resulting problem is a multiple decision maker multi-item dynamic lot-sizing problem with limited production capacity (e.g., see Jans and Degraeve, 2008). The problem is NP-hard since it can be shown to generalize the very special case with single-decision maker, single-item, zero inventory holding cost, convex production cost function, unit set-up cost, and no production capacity, that has been proved to be NP-hard by Florian et al. (1980). Since the problem at production planning level is NP-hard this level of the MAS framework is coupled with a heuristic iterative auction based negotiation protocol to coordinate the agents' decisions (see, e.g.: Kutanoglu and Wu, 2006; Roundy et al., 1991; Schneider et al., 2005). The resource prices, needed for the iterative auction based protocol, are updated using a strategy inspired by the subgradient technique used in the Lagrangian relaxation approach (see, e.g., Barahona and Anbil, 2000; Chen et al., 1998).

Given the number of robots assigned to the tasks in each time period according to the decisions made at the first, planning level, on the second, scheduling level, the objective is to minimize the total distance covered by the robots in the reallocation between consecutive periods. Therefore, a Multi-Robot Task Allocation (MRTA) problem is solved for each period. The objective of the MRTA problem is to find the assignment of n robots to a set of n tasks (target positions) based on the optimization of some global objective function (see, e.g., Gerkey & Mataric, 2003). We assume that the decision making environment for this level is decentralized as well, with as many decision makers (agents) as there are the robots in the system. In particular, we assume robots to be collaborative, homogeneous, arranged in regular networks and relying on local communication only between neighboring robots. We use a distributed version of the Hungarian Method for this allocation problem, a distributed combinatorial optimization algorithm which solves the assignment problem in strongly polynomial time (Giordani, Lujak, & Martinelli, 2010).

Note that the problem definition and the proposed modeling framework are general enough so that the production planning and scheduling problem and the solution model can be applied also to other types of mobile manufacturing resources and production operators.

We experiment the proposed model considering a fluctuating demand modeled through an ARMA process. To measure the effectiveness of the approach, the social welfare (see, e.g., Chevaleyre et al., 2006) of the task agents in the decentralized scenario is compared with the performance obtained through a centralized solution. Preliminary results regarding the first level of the proposed framework have been presented by Giordani et al. (2009) where the problem addressed on the second level (the robot movement) was not considered. The integrated solution of the two levels is a viable solution for the incorporation of mobile robots on the shop-floor and provides indeed an interesting insight into the problem. In particular, we show that the decentralized approach of the first level gives comparable results to the centralized one while the required total movement distance of the robots' reallocation is in general inferior.

The remainder of the paper is organized as follows. We introduce related work and review some of the economic models used in MAS negotiation for resource allocation in Section 2. In Section 3, the decentralized production scheduling problem is presented. The two-level solution approach is given in Section 4. In Section 5, we present the simulation results. We close the paper with the conclusions in Section 6.

2. Related work

There is a vast literature on planning and scheduling techniques in manufacturing based on multi-agent systems (see, e.g., Shen et al., 2006, 2007; Wang et al., 2008). For multi-agent interaction and negotiation, there are several applicable economic models which might work well for the production planning level of the presented framework where tasks compete for resources (robots): commodity market, posted price, bargaining, tender/contract-net, and auction model (see, e.g., Buyya et al., 2002; Chevaleyre et al., 2006; Kraus, 2001), which is the approach chosen in this paper.

Regarding the second level, MRTA corresponds to the (linear sum) assignment problem for which the first developed algorithm was the Hungarian Method (Kuhn, 1955). The Hungarian Method or Kuhn–Munkres algorithm is a well known iterative algorithm which maintains dual feasibility during calculation and searches for a primal solution satisfying complementary slackness conditions. If the primal solution is feasible, the solution is optimal. If the primal solution is not feasible, the method performs a modification of the dual feasible solution after which a new iteration starts. Hungarian Method can be implemented using the alternating trees so that its worst case time complexity is limited by $O(n^3)$ (see, e.g., Papadimitriou and Steiglitz, 1982).

There are several main approaches to the assignment problem (see, e.g., Burkard and Çela, 1999). The classical centralized assignment methods find a solution through the iterative improvement of some cost function: in primal simplex methods it is a primal cost, and in Hungarian, dual simplex and relaxation methods it is a dual cost (see, e.g., Bertsekas, 1992).

The Auction algorithms can improve as well as worsen both the primal and the dual cost through the intermediate iterations, although at the end, the optimal assignment is found (Bertsekas, 1991). Bertsekas in this work introduces the auction algorithm in which the agents bid for the tasks in iterative manner, and in each iteration, the bidding increment is always at least equal to ϵ (ϵ-complementary slackness). If $\epsilon < \frac{1}{n}$ the algorithm finds the optimal solution, and it runs in $O(n^3 \cdot \max\{c_{ij}\})$ time, where c_{ij} is the assignment cost of robot i to task j, that is in pseudo-polynomial time. Using ϵ-scaling technique and appropriate data structures a polynomial time version of the auction algorithm running in $O(n^3 \log(n \cdot \max\{c_{ij}\}))$ time is given by Bertsekas and Castanon (1989) and Bertsekas (1992).

Zavlanos et al. (2008) provide a distributed version of the auction algorithm proposed by Bertsekas for the networked systems with the lack of global information due to the limited communication capabilities of the agents. Updated prices, necessary for accurate bidding can be obtained in a multi-hop fashion only by local exchange of information between adjacent agents. No shared memory is available and the agents are required to store locally all the pricing information. This approach calculates the optimal solution in $O(\Delta \cdot n^3 \cdot \max\{c_{ij}\})$ time, with $\Delta \leqslant n-1$ being the maximum network diameter of the communication network.

There are also many parallel algorithms based on the Hungarian Method. For a good survey see, e.g., Burkard and Çela (1999), and Bertsekas et al. (1995). Among the most efficient parallel algorithms for the assignment problem is the one proposed by Orlin

and Stein (1993) that adopting cost scaling technique solves the problem using $\Omega(n^4)$ processors in $O(\log^3 n \cdot \log(\max\{c_{ij}\}))$ time.

In the robotic community, there are many heuristic methods developed for the MRTA problem. Smith and Bullo (2007) describe two algorithms, namely ETSP, and Grid algorithm, for the task assignment in the sparse and dense environments. The agents have a full knowledge of the tasks in the environment and, in the ETSP algorithm, approach the closest ones. If the closest task is occupied, they pre-compute an optimal tour through the n remaining tasks, and search for the first non-occupied task based on certain criteria. Gerkey and Mataric (2003, 2004) describe certain relevant heuristic algorithms for the MRTA problem through the prism of three important factors: computation and communication requirements, and the manner in which tasks are considered for (re)assignment, i.e. whether all or just a part of the tasks are considered for (re)assignment in each iteration. The problem with these approaches, as the authors state, is that there is no characterization of the solution quality that can be expected by the algorithms. Kwok, Driessen, Phillips, and Tovey (2002) apply the classic combinatorial methods, such as Hungarian and Gabow algorithms, to the assignment problem in the context of a set of mobile robots which need to be moved to a desired matrix of grid points to have a complete surveillance of the desired area.

3. Problem formulation

A manufacturing system producing P different types of goods is considered. A task denotes a production process of a particular type of product and all the tasks are assumed different from one another. The tasks are executed by a set of N identical mobile robots, characterized by a hiring cost, defined below.

Given a finite time horizon of T time periods and assuming that the demand for the P products in the T time periods is known, the discrete time production planning and scheduling problem addressed in this paper consists of finding for each time period the number of robots to be assigned to each task and the task production rate, in order to minimize a total cost composed of backlog, inventory, manufacturing, and robot hiring costs.

Once the solution to the above problem has been determined, the robots decide how to move in order to fulfill the assignment requests by covering the minimum total distance.

A two-level hierarchical decentralized production planning and scheduling problem can be then formulated, where, at the first level, the tasks are considered as agents which compete for robots, resources characterized by a hiring price; at the second level, robots are the agents which autonomously decide upon the way to meet the requests of the first level.

3.1. Production planning level

At the first, production planning level, each task agent requests the task coordinator agent for robots for each time period, based on product demands and production costs. The task coordinator agent assigns the robots to the tasks for each unit time period on the basis of their requests.

For all time periods $k = 1, \ldots, T$, the agent representing task i, related to product i, knows product demand rate $d_i(k)$, robot maximum production rate $r_i(k)$ (production capacity), unitary manufacturing cost $c_i(k)$, unitary holding cost $h_i(k)$, unitary backlog cost $b_i(k)$, and unitary robot hiring cost $\rho_i(k)$, all nonnegative. It is reasonable to assume that the robots' displacement time is negligible in respect to the length of the time period, Δt. For each time period k, the decision variables controlled by the agent representing task i are:

- $n_i(k) \geqslant 0$: required number of robots (robot request);
- $u_i(k) \geqslant 0$: production rate.

Production can be anticipated or delayed in respect to the product demands, resulting in holding and backlog costs, respectively. Each task i is associated with a buffer whose content $x_i(k)$ at the end of time period k can be positive (if a stock of completed product items is present in the buffer) or negative (if a backlog of demands for the product realized with task i is in the queue). Using a standard notation, we denote $x_i^+(k) := \max\{x_i(k), 0\}$ as the stock level, and $x_i^-(k) := \max\{-x_i(k), 0\}$ as the backlog level at the end of time period k. Notice that, for each time period k, and task (product) i, only one of $x_i^+(k)$ and $x_i^-(k)$ can be different from 0 (i.e., $x_i^+(k) \cdot x_i^-(k) = 0$ for $k = 1, \ldots, T$ and $i = 1, \ldots, P$).

The local optimization problem addressed by every task agent i includes $x_i^+(k)$ and $x_i^-(k)$ as additional variables, for each time period k. Those variables, together with the production rates $u_i(k)$, are assumed to be continuous according to a fluid approximation. Given the above parameters and variables, the local optimization problem, P_i, addressed by each task agent i can be formulated as follows.

(P_i):

$$\min z_i = \sum_{k=1}^{T} \left[h_i(k) x_i^+(k) + b_i(k) x_i^-(k) + c_i(k) u_i(k) + \rho_i(k) n_i(k) \right] \quad (1)$$

s.t.

$$x_i^+(k) - x_i^-(k) = x_i^+(k-1) - x_i^-(k-1) + \Delta t [u_i(k) - d_i(k)], \quad k = 1, \ldots, T \quad (2)$$

$$u_i(k) \leqslant r_i(k)\, n_i(k), \quad k = 1, \ldots, T; \quad (3)$$

$$u_i(k), x_i^+(k), x_i^-(k), \geqslant 0, \quad k = 1, \ldots, T; \quad (4)$$

$$0 \leqslant n_i(k) \leqslant N \text{ and integer}, \quad k = 1, \ldots, T. \quad (5)$$

The values of $x_i^+(0)$, $x_i^-(0)$, and $n_i(0)$ represent the initial conditions, i.e. the stock level, the backlog level, and the number of preassigned robots to task i respectively, at the beginning of the planning time horizon. For simplicity, but w.l.o.g., these values are assumed to be equal to zero. Constraints (2) are the mass balance constraints among product demand $\Delta t \cdot d_i(k)$, stock level $x_i^+(k)$, backlog level $x_i^-(k)$, and production level $\Delta t \cdot u_i(k)$, for each time period k. W.l.o.g., assuming that, for each time period k and for each task i, the values of $h_i(k)$ and $b_i(k)$ are positive, the constraint $x_i^+(k) \cdot x_i^-(k) = 0$, is implicitly satisfied by the optimal solution, and, hence, omitted in the formulation. Constraints (3) limit the production rate $u_i(k)$ to be not greater than the production capacity $r_i(k) \cdot n_i(k)$. Problem (P_i) belongs to the class of deterministic dynamic lot-sizing problems well known in the inventory management literature (see, e.g., Jans and Degraeve, 2008).

For each task agent i, let $n_i^*(k)$ be the number of required robots at time period k, in the optimal solution of problem P_i, and let z_i^* be the optimal solution of P_i. Each task agent (decision maker) finds such a solution taking into account only its local objective and constraints, but shares with the other task agents limited amount of available robots (resources). Therefore, these local decisions can be implemented only if the following global constraints are satisfied.

$$\sum_{i=1}^{P} n_i^*(k) \leqslant N, \quad k = 1, \ldots, T. \quad (6)$$

In general, this is not the case and, hence, a negotiation process must be implemented among the task agents to come up with a set of local solutions that together satisfy also constraints (6). We assume that this process is supervised by another decision maker, i.e., a coordinator agent, which can be an arbitrary preassigned task agent that assigns the robots to the tasks for each time period on the basis of their requests, guaranteeing the fulfillment of the constraint on the limited robot amount. In Section 4.2, we provide a model for such a negotiation.

3.2. Production scheduling level

On the second, production scheduling level, which represents robots' allocation to task locations, the scope is to assign individual robots to tasks so as to minimize total movement cost (i.e., total distance traveled by robots among task locations) during the planning time horizon, given the number $n_i(k)$ of robots required by task i (with $i = 1, \ldots, P$) in period k (with $k = 1, \ldots, T$) according to the decisions made at the first level.

Let ℓ_i be the location where task i is executed. We assume that at the beginning of the planning time horizon the robots are located at a depot whose location is ℓ_0, and that at the end of the planning time horizon, all the used robots must leave the task locations and go back to the depot. Let $d(\ell_i, \ell_j)$ be the distance that a robot has to travel to go from location ℓ_i to location ℓ_j, with $i, j = 0, 1, \ldots, P$. The distances among locations are assumed to satisfy the triangle inequality, with the assumption, for simplicity and w.l.o.g., that the depot is located at an equidistant point from the task locations.

Therefore, at the beginning of the first time period, $n_i(1)$ robots leave the depot and move to location ℓ_i of task i, for each task $i = 1, \ldots, P$. Successively, for each $k = 2, \ldots, T$, at the beginning of period k, the robots should be reallocated among the task locations with possibly some additional robots leaving the depot and going to the required task locations, such that at the beginning of period k (at least) $n_i(k)$ robots are located at location ℓ_i of task i.

Let $N(k) \leqslant N$ be the maximum number of robots used during the first k periods, i.e., $N(k) = \max_{h=1,\ldots,k}\{\sum_{i=1}^{P} n_i(h)\}$, with $k = 1, \ldots, T$, and let $N(0) = 0$. Since the locations' distances satisfy the triangle inequality, we may assume that only if $N(k) > N(k-1)$, we have that $N(k) - N(k-1)$ additional robots leave the depot and reach some target (task) locations exactly at the beginning of period k. On the contrary, if $N(k) = N(k-1)$ and $\sum_{i=1}^{P} n_i(k) < N(k-1)$, (at least) $N(k) - \sum_{i=1}^{P} n_i(k)$ robots will remain at their current (task) locations also during period k, assuming that at each task location there is sufficient room to park also unused robots. Therefore, let us denote with $n'_i(k) \geqslant n_i(k)$ the number of robots located at location ℓ_i during period k, where $n'_i(k) - n_i(k)$ is the number of robots parked at that location but not used by task i during period k. Clearly $\sum_{i=1}^{P} n'_i(k) = N(k)$. Finally, we may therefore assume that only at the end of the planning time horizon the robots return to the depot.

According to the above considerations, in order to minimize the total distance traveled by the robots during the planning time horizon, we can independently minimize the total distance traveled by the robots for their reallocation to task locations between each two consecutive time periods $k-1$ and k, with $k = 2, \ldots, T$; let us denote with $D(k)$ the minimum total distance covered by the robots during reallocation k. The minimum total distance traveled by the robots is $D = \sum_{k=1}^{T+1} D(k)$, where $D(1)$ is the total distance traveled by $n_i(1)$ robots from the depot location ℓ_0 to the task location ℓ_i, for each task i, at the beginning of time period 1, and $D(T+1)$ is the total distance traveled by the robots to go back to the depot location from their last locations at the end of the planning time horizon. Clearly, $D(1)$ does not depend on the (successive) robot reallocations. The same applies also to $D(T+1)$, since it is assumed that ℓ_0 is located at equidistant position from ℓ_i.

Let us therefore consider the robots' reallocation problem between two consecutive periods $k-1$ and k, with $k = 2, \ldots, T$. We model this problem considering a set $\mathcal{R}$ of $N(k)$ collaborative mobile robots, a set $\mathcal{T}$ of $N(k)$ targets (tasks), and a given $N(k) \times N(k)$ matrix of costs γ_{rt} for allocating robot r to target t. The problem faced by the robots is determining the allocation of robots to targets such that each robot is allocated exactly to one target, each target is allocated to exactly one robot, and the total allocation cost is minimized.

The cost γ_{rt} for allocating robot r to target t depends on the (current) location of robot r and the location of target t.

Since the robots are assumed to be of the same type, without loss of generality, we can re-index them such that robot $r \in \left\{1 + \sum_{h=1}^{i-1} n'_h(k-1), \ldots, \sum_{h=1}^{i} n'_h(k-1)\right\}$, is assumed to be located at location ℓ_i of task i (with $i = 1, \ldots, P$), and robot $r \in \left\{1 + \sum_{h=1}^{P} n'_h(k-1), \ldots, N(k)\right\}$, is a (new) robot that starts to be used for the first time in period k and therefore is assumed to be located at the depot location ℓ_0, and has to go to a certain task location at the beginning of period k. Similarly, the targets are indexed so that target $t \in \left\{1 + \sum_{h=1}^{i-1} n_h(k), \ldots, \sum_{h=1}^{i} n_h(k)\right\}$, is located at location ℓ_i, with $i = 1, \ldots, P$, and target $t \in \left\{1 + \sum_{h=1}^{P} n_h(k), \ldots, N(k)\right\}$, is a dummy (fictitious) target assumed to be located at the dummy (fictitious) location $\bar{\ell}$. Finally, cost γ_{rt} is assumed as the distance between the location of robot r and the location of target t if the latter is non-dummy, otherwise such cost is assumed to be equal to zero since allocating a robot to the dummy target means that this robot will be unused in period k and hence it will remain parked at its previous location during that period.

The mathematical formulation of the problem is

$$\min \sum_{r \in \mathcal{R}} \sum_{t \in \mathcal{T}} \gamma_{rt} \cdot x_{rt} \tag{7}$$

s.t.

$$\sum_{r \in \mathcal{R}} x_{rt} = 1, \quad \forall\, t \in \mathcal{T}, \tag{8}$$

$$\sum_{t \in \mathcal{T}} x_{rt} = 1, \quad \forall\, r \in \mathcal{R}, \tag{9}$$

$$x_{rt} \geqslant 0, \quad \forall\, r \in \mathcal{R}, \quad \forall\, t \in \mathcal{T}, \tag{10}$$

and it corresponds to the well known assignment problem. Although fractional solutions may exist, there always exists an optimal integer solution which corresponds to a real assignment, since no fractional solution is a basic feasible solution of the above linear program (see, e.g., Papadimitriou and Steiglitz, 1982). In particular, $x_{rt} = 1$ if the robot r is assigned to target t; otherwise $x_{rt} = 0$.

Denoting the above problem formulation as the primal problem, it is possible to define the dual problem as follows. Let α_t be the dual variable related to target t, and β_r the same related to robot r. Both dual variables are real and unrestricted in sign. The dual problem is

$$\max \left\{ \sum_{r \in \mathcal{R}} \beta_r - \sum_{t \in \mathcal{T}} \alpha_t \right\} \tag{11}$$

s.t.

$$\beta_r - \alpha_t \leqslant \gamma_{rt}, \quad \forall\, r \in \mathcal{R}, \quad \forall\, t \in \mathcal{T}. \tag{12}$$

It is possible to give an economic interpretation to the dual problem and the dual variables: α_t is the price that each robot r will pay if it gets assigned to target t, while β_r is the robot r's utility for being assigned to a certain target. Constraint (12) of the dual problem states that the utility β_r of robot r cannot be greater than the total cost ($\gamma_{rt} + \alpha_t$) faced by robot r for being assigned to target t. The dual objective function (11) to be maximized is the difference between the sum of the utilities of the robots and the sum of the prices of the targets, i.e., the total net profit of the robots. On the basis of the duality in linear programming, $\sum_r \beta_r - \sum_t \alpha_t \leqslant \sum_r \sum_t \gamma_{rt} \cdot x_{rt}$; therefore, the total net profit of the robots cannot be greater than the total assignment (displacement) cost that the robots have to pay for being assigned to the targets, and only at the optimum, those two are equal.

Obviously, at the optimum, each robot r will be assigned to target t for which the utility of the robot is $\beta_r = \gamma_{rt} + \alpha_t$, i.e., exactly equal to the total cost that the robot r would pay for being assigned to target t.

4. Solution approach

The problems associated with the production planning and scheduling level are solved by two different and decentralized algorithms, the first one consisting of a negotiation among the tasks, and the second one amounting to a distributed version of the Hungarian algorithm, executed collaboratively by the robots.

4.1. Two-level MAS planning and scheduling framework

The two-level multi-agent system framework for planning and scheduling of production with mobile robots is shown in Fig. 1.

At the first, planning level, the negotiation among the tasks is modeled by an iterative auction process controlled by the coordinator agent which is in charge of coordination of the negotiation. The details regarding the auction process are found in Section 4.2. Once when, at the planning level, the number of robots assigned to each task for each time period $k \in \{1 \dots T\}$ is known, the planning level ends and the scheduling level begins its performance with the aim to find for each robot exact movements on the shop-floor through the whole time horizon subdivided in T time periods.

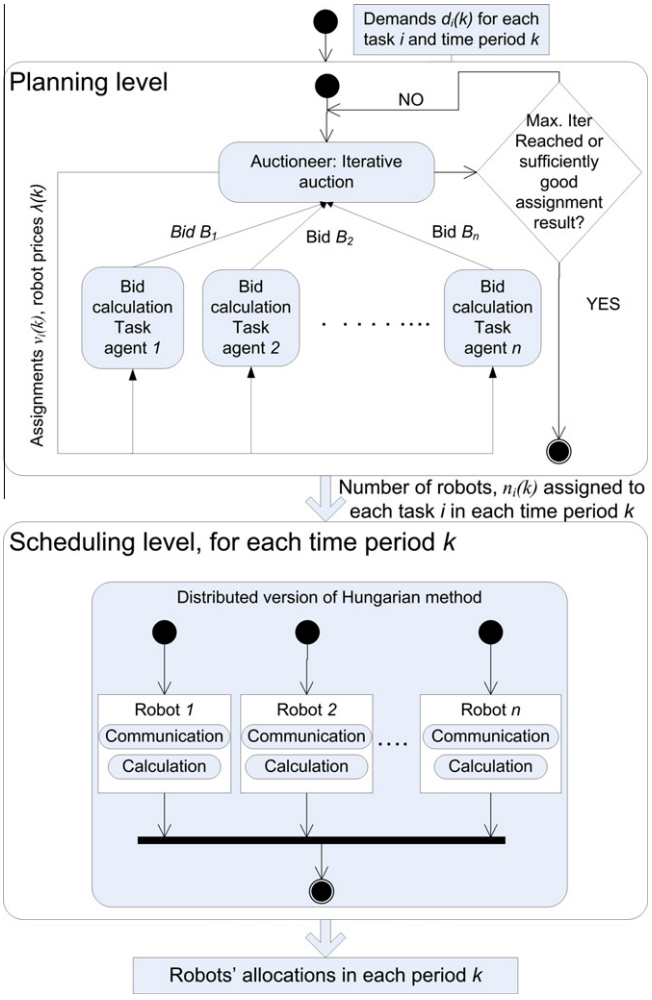

Fig. 1. Two-level framework for multi-agent system production planning and scheduling of mobile robots.

4.2. First level: negotiation process among the tasks

In the iterative auction process among tasks, at each iteration:

- Step 1 The coordinator (*auctioneer*) communicates to all the task agents (*bidders*) current prices of the robots in the T time periods.
- Step 2 Each task agent, based on the local utility function of its local objective, the constraints, and the current robot prices, determines the robot requests (a *bid*) for the T time periods maximizing its utility function and communicates its robot requests to the coordinator.
- Step 3 Based on the bids received from the task agents, the coordinator allocates the robots to the tasks for each time period, maximizing the robots' total utility function.
- Step 4 In order to reduce possible conflicts among tasks generated by their bids (i.e., the violation of constraints (6)), or to stimulate usage of the robots, the coordinator updates the robot prices.

Iteration process repeats until a certain (halting) condition is reached (e.g., a maximum number of iterations have been performed, or the best robot allocation found is sufficiently good from the agents' point of view). At the end, the best robot allocation is retrieved.

Let $B_i = \{n_i(k): k = 1, \dots, T\}$ be the bid of task agent i, and let $\Lambda = \{\lambda(k): k = 1, \dots, T\}$ be the (current) set of the robot prices fixed by the coordinator (each one for each time period). The utility function $U_i(B_i, \Lambda)$ of agent i is the opposite of the total production cost during the planning time horizon:

$$U_i(B_i, \Lambda) = -z_i^*(B_i) - p(B_i, \Lambda). \tag{13}$$

In this expression, $z_i^*(B_i)$ is the (minimum) production cost for task i with fixed numbers of assigned robots $n_i(k)$, with $k = 1, \dots, T$, according to bid B_i, and $p(B_i, \Lambda) = \sum_{k=1}^{T} \lambda(k) n_i(k)$ is the (additional) robot cost. Note that, $z_i^*(B_i)$ is the optimal solution of the local optimization problem P_i with fixed values of $n_i(k)$.

In Step 2, agent i finds the best bid $B_i^* = \{n_i^*(k) : k = 1, \dots, T\}$, i.e., the bid that maximizes its utility function $U_i(B_i, \Lambda)$. Let $U_i^*(\Lambda)$ be its maximum value for a given set Λ of robot prices. Finding B_i^*, and hence determining $U_i^*(\Lambda)$, corresponds to solving problem P_i with $\lambda(k)$ added to the robot hiring cost $\rho_i(k)$ in the objective function of P_i.

In Step 3, the coordinator receives bid B_i^* from each task i, and assigns robots to tasks maximizing resources' (robots') utility function taking into account received bids. Denoting with $v_i(k)$ the number of robots assigned to task i in time period k, robots' utility function $R(v, \Lambda)$ is the total profit obtained from the robot assignment, and, hence,

$$R(v, \Lambda) = \sum_{k=1}^{T} \lambda(k) \sum_{i=1}^{P} v_i(k). \tag{14}$$

Since $v_i(k)$ cannot be greater than robot request (bid) $n_i^*(k)$ of task agent i and $\sum_{i=1}^{P} v_i(k)$ cannot be greater than N, for each $k = 1, \dots, T$, the maximization of $R(v, \Lambda)$ can be easily obtained by (heuristically) assigning robots to tasks on the basis of the values of $n_i^*(k)$, for example adopting one of the following rules and taking the best related solution.

- *Rule 1*. For each time period k, order the tasks according to non-increasing robot requests $n_i^*(k)$, and, following this task order, assign $v_i(k) = \min\{n_i^*(k), N'(k)\}$ robots to task i, where $N'(k)$ (with $0 \leqslant N'(k) \leqslant N$) is the number of non-assigned (yet available) robots in time period k.

- *Rule 2*. For each time period k, assign proportionally robots to tasks according to the robot requests $n_i^*(k)$, that is, denoting with $w(k) = \min\{1, (N/\sum_i n_i^*(k))\}$, assign $v_i(k) = w(k) \cdot n_i^*(k)$ (approximating the value to the closest integer) robots to task i in time period k, and such that $\sum_i v_i(k) \leqslant N$.

Quality of robot assignment is evaluated from the task agents' point of view, measuring the *social welfare* $w(v)$ of the tasks related to a given robot assignment $v = \{v_i(k) | i = 1, \ldots, P, k = 1, \ldots, T\}$ as the (minimum) total production cost of the tasks with that robot assignment, that is,

$$w(v) = \sum_{i=1}^{P} z_i^*(v_i), \tag{15}$$

where $z_i^*(v_i)$ is the optimal solution of P_i with a given number of robots $n_i(k) = v_i(k)$ assigned to task i, for each time period k. Let w^* be the best (minimum) value of the social welfare found so far during iterative auction process. It can be proved that expression

$$w_L(\Lambda) = -\sum_{i=1}^{P} U_i^*(\Lambda) - N\sum_{k=1}^{T} \lambda(k), \tag{16}$$

is a valid lower bound on the best value of the task agents' social welfare, for any vector Λ of non-negative robot prices $\lambda(k)$.

In Step 4, the coordinator updates the robot prices $\lambda(k)$, with $k = 1, \ldots, T$. This is done, considering the *deviation* $dev(k) = \sum_{i=1}^{P} n_i^*(k) - N$ of the total number of requested robots from the number N of available robots, and by increasing the current value of $\lambda(k)$ if $dev(k)$ is positive (i.e., $dev(k)$ is the *excess* of robot requirements), or decreasing it (at most to 0) if $dev(k)$ is negative (in this latter case $-dev(k)$ is the *deficit* of robot requirements). The value of the increase (decrease) of $\lambda(k)$ should be a non-decreasing function of the excess (deficit); moreover, it is a good practice that the auctioneer is more aggressive in the early iterations to get very quickly a good robot assignment, while smaller adjustments can be made in later iterations to refine the quality of the same. A possible choice for the price updating that goes in this direction is that of using an algorithm inspired by the subgradient technique used in Lagrangean relaxation, that experimentally guarantees the convergence of the Lagrangean dual (see, e.g., Held et al., 1974).

At iteration h of the auction process, let $dev^h(k)$ be the deviation of the robot requirements, and Λ^h be the vector of the robot prices $\lambda^h(k)$, with $k = 1, \ldots, T$. The new value of the robot price in time period k at iteration $h + 1$ is:

$$\lambda^{h+1}(k) = \max\left\{0, \quad \lambda^h(k) + \psi^h \frac{w^* - w_L(\Lambda^h)}{\sum_{k=1}^{T} (dev^h(k))^2} dev^h(k)\right\}, \tag{17}$$

where ψ^h is a scalar controlling the aggressiveness of the auctioneer. Following analog practice in Lagrangean relaxation (see, e.g., Held et al., 1974) we start with value $\psi^0 = 2$, and halve its value if the best task social welfare value found so far is not improved within a certain number of iterations.

In Step 1, the amount of information going from the coordinator to task agents is $O(P \cdot T \cdot \log(\lambda_{max}))$ bits, where λ_{max} is the maximum robot price. In Step 2, the amount of information sent by task agents to the robot owner is $O(P \cdot T \cdot \log(N))$ bits. Step 3 is done in $O((PlogP) \cdot T)$ time. Finally, in Step 4, price update is done in $O(P \cdot T)$ time.

4.3. Distributed robot allocation algorithm

This section provides a distributed robot allocation algorithm for the problem of robots' reallocation among task locations, i.e., targets, between two consecutive periods $k-1$ and k of the planning time horizon. As discussed above, this corresponds to the well known assignment problem where n robots have to be assigned to n targets, with $n = N(k)$.

It is assumed that each robot r has the information of the distances (assignment costs) between its current location and the locations of all the targets. The inter-robot communication is performed over a connected dynamic communication network and the solution to the assignment problem is reached without any common coordinator or a shared memory of the system.

To solve the problem in this context, we developed a distributed version of a Hungarian Method for the assignment problem, based on the concept of augmenting paths from the graph theory that we recall briefly in the following.

We refer to the centralized version of the Hungarian Method implemented by means of so-called alternating trees and running in $O(n^3)$ time (see, e.g., Burkard and Çela, 1999). The algorithm is iterative and, maintaining a feasible dual solution, considers the admissible bipartite graph $\overline{G} = (\mathcal{R} \bigcup \mathcal{T}, \overline{E})$ where $\overline{E} = \{(r,t) : \gamma_{rt} + \alpha_t - \beta_r = 0\}$. The algorithm searches for a matching of maximum cardinality in the graph $\overline{G}$. If the matching is perfect, i.e., if every robot is matched with a target, then the matching represents an optimal solution of the robot allocation and the algorithm stops. If the matching is not perfect, the algorithm updates the dual variables so as to increase the dual objective function such that at least one new admissible edge is added to $\overline{G}$, and continues with a new iteration.

Given $\overline{G}$, let $M \subseteq \overline{E}$ be the current maximal matching in $\overline{G}$. The edges belonging to M are called matched edges, and the ones in $\overline{E} \setminus M$ are free edges. Given $\overline{G}$ and the current maximal matching $M \subseteq \overline{E}$, the algorithm iteratively improves the matching along augmenting paths over alternating trees in $\overline{G}$, rooted at free target vertices. An alternating tree is a subgraph of $\overline{G}$ with each target vertex connected to all its adjacent robot vertices through free edges of $\overline{E}$, and with each robot vertex connected to a target vertex through a matched edge. When an alternating tree rooted at free target vertex reaches a target leaf vertex which is adjacent to some free robot vertex, an augmenting path is found, i.e., a path that lets the augmentation of the cardinality of the matching, by exchanging the free and matched edges. When all the possible augmentation steps are performed, and hence the resulting matching in $\overline{G}$ is of maximum cardinality, the dual variables are updated and the new admissible edges are added. Then a new iteration begins searching for the maximum matching on the new admissible graph (see, e.g., Burkard and Çela, 1999).

In our distributed implementation of the Hungarian Method, we maintain the forest F^1 of all the alternating trees rooted at free target vertices. Moreover, we maintain the forest F^2 of the alternating trees in $\overline{G}$ rooted at robot vertices containing all the robot/target vertices not contained in F^1. Clearly, the alternating trees in F^2 are not connected with vertices in F^1.

Initially, all the vertices of $\overline{G}$ are assumed isolated (i.e., $\overline{E} = \emptyset$), and hence forests F^1 and F^2 are set up in the following way. Forest F^1 is composed of n isolated target vertices. Forest F^2, in the similar way, is made of n isolated robot vertices. Dual variables are initially assumed to be $\alpha_t = 0$, for all the targets t, and $\beta_r = \min_t\{\gamma_{rt}\}$, for all the robots r, and are updated as follows. For each robot vertex $r \in F^2$, let $\sigma_r = \min_{t \in F^1}\{\gamma_{rt} + \alpha_t - \beta_r\}$ be the minimum slack value of the dual constraints (12) associated to robot r among target vertices t in F^1, and let t_r be the target corresponding to the arg $-$ min; moreover, let $\delta = \min_{r \in F^2}\{\sigma_r\}$ be the minimum value among the minimum slack values of the robot vertices r in F^2. The new (feasible) values of the dual variables are:

$$\alpha_t := \alpha_t - \delta, \quad \forall \text{ target vertex } t \in F^1$$
$$\beta_r := \beta_r - \delta, \quad \forall \text{ robot vertex } r \in F^1$$

After updating the dual variables, new admissible edges (r^*, t^*) with $r^* \in F^2$ and $t^* \in F^1$ appear. Even though there might be more than one new admissible edge, w.l.o.g., we can add the new edges one at a time to $\overline{G}$, and after adding each (r^*, t^*), a new matching on the augmented admissible graph can be searched for. New edge (r^*, t^*) connects the two forests F^1 and F^2. There are two cases:

- Robot vertex r^* is free (unmatched). In this case there is an augmenting path $(t, \ldots, t^*, r^*)$ starting from a (free) root target vertex t of F^1, connected with t^* through an alternating path of F^1, and ending in r^* through edge (t^*, r^*). Fig. 2 shows this case, where white dots are target vertices, black dots are robot vertices, plain line segments are free edges, double line segments are matched edges, and the bold line segment is the new admissible edge (t^*, r^*); triangles represent subtrees rooted at target vertices. By exchanging the free and matched edges in the augmenting path we get the augmented matching. Since target vertex t is not free any more, the whole tree of F^1 rooted in t moves to F^2 and is connected to root r^* over t^* (see Fig. 3).
- Robot vertex r^* is not free, i.e., it is matched. In this case there is no new augmenting path (see Fig. 4); the two forests of alternating trees are updated by disconnecting robot vertex r^*, along with the whole subtree rooted in r^*, from forest F^2, and connecting this subtree (rooted in robot vertex r^*) to the target vertex t^* in forest F^1 through the new admissible edge (t^*, r^*) (see Fig. 5).

Note that at the end, when we have a perfect matching of $\overline{G}$, all target vertices are matched and therefore forest F^1 is empty.

4.3.1. Implementation details

The robots interactively execute the steps of the algorithm based on the local information and the one received through the messages from their neighbors in a communication graph. The structure of the communication graph is made of the forests F^1 and F^2 restricted to robots. The interconnected robots recalculate the structure of the forests each time the forests change. Therefore, the connection graph is updated each time the update of the forests F^1 and F^2 occurs.

Each robot r keeps in its memory 4 pointers relating to the robots on its neighboring positions in the forests, i.e., the father robot, the oldest son robot, the next older brother robot, and the next younger brother robot, as seen from Fig. 6. The first two pointers are used to move upwards and downwards in the forest while the two latter ones are used to explore the branch.

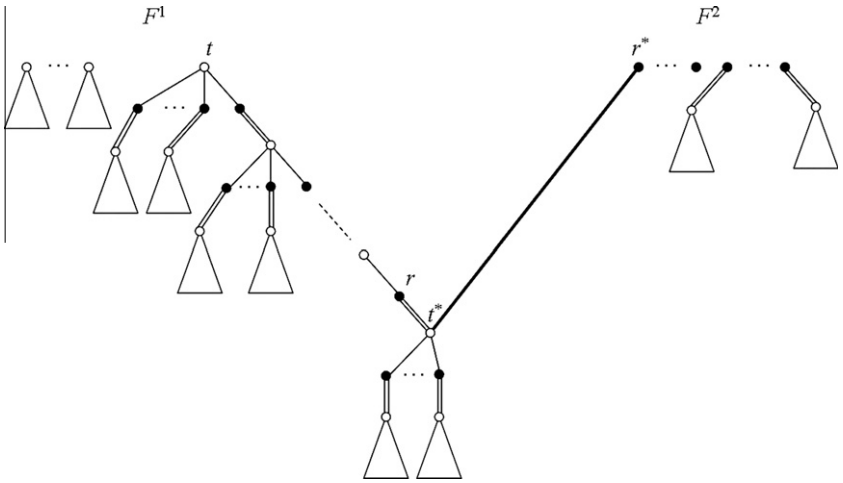

Fig. 2. Case 1: augmenting path $(t, \ldots, t^*, r^*)$.

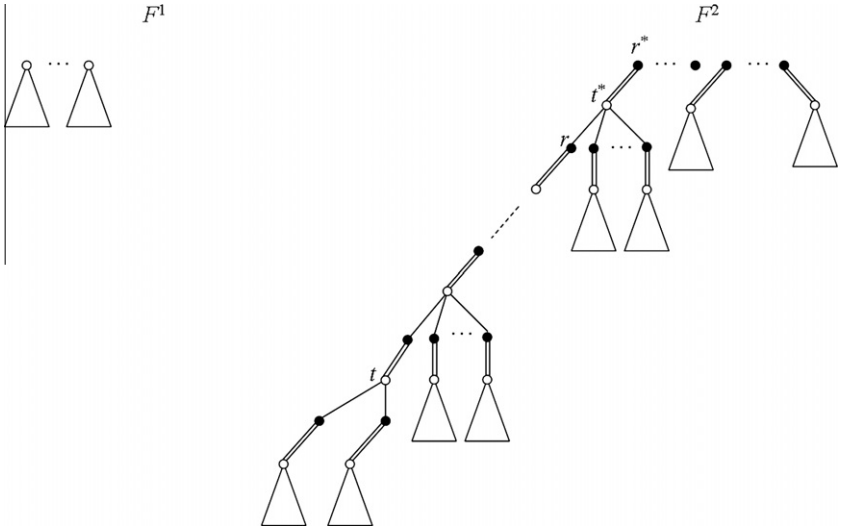

Fig. 3. Case 1: forests' updating.

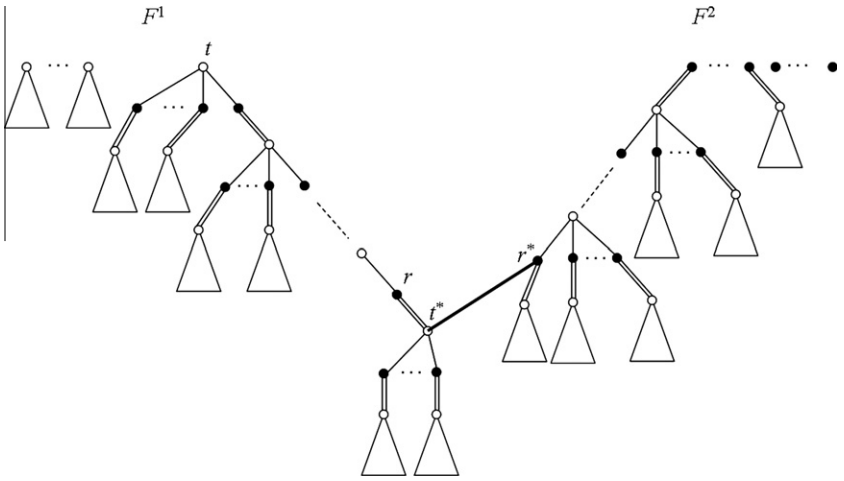

Fig. 4. Case 2: no new augmenting path.

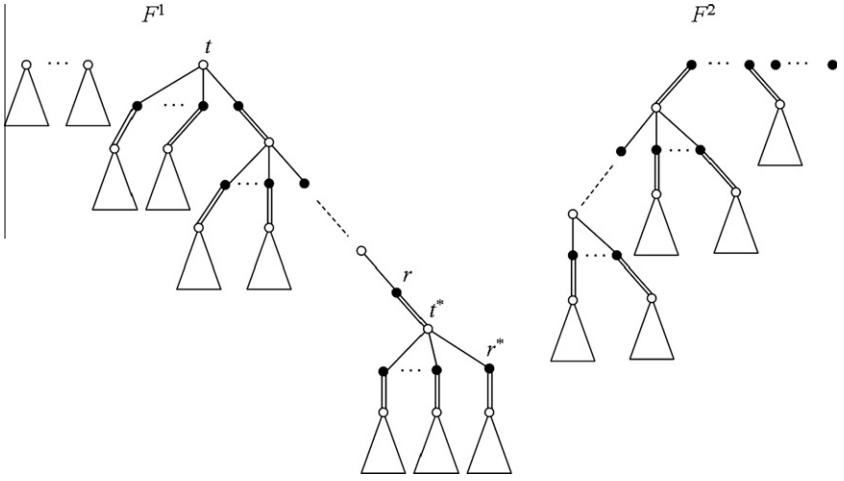

Fig. 5. Case 2: forests' updating.

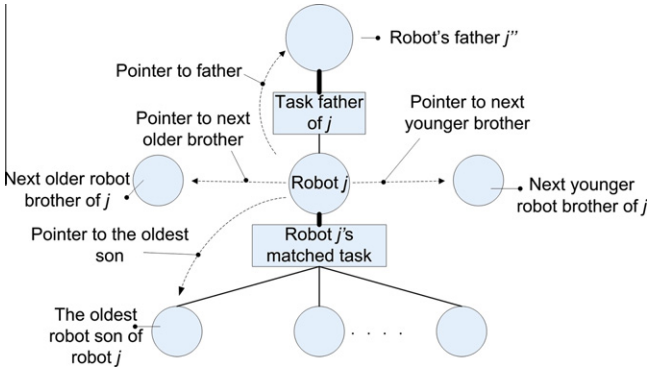

Fig. 6. The pointers to the (robot) neighbors of a robot j.

Each robot r in its memory also keeps the data about its position (whether it is in F^1 or F^2), its matched target m_r (if there is none, i.e., if r is free, $m_r = 0$), and the father target vertex in the forest. Each robot r memorizes also the vector of the distances γ_{rt} from the location of robot r to the targets' locations, the value of the dual variable β_r, and the minimum slack value σ_r along with the target t_r which obtains the σ_r.

Through autonomous calculations and the communication with the (robot) neighbors, robots get and share the information about the position of each target (whether in F^1 or F^2), the values of dual variables α_t of all the targets t, the value of δ for the dual variables' update, the new admissible edge (r^*, t^*) between a robot r^* in F^2 and a target t^* in F^1 due to the dual variables' update, and the root robots $r(F^1)$ and $r(F^2)$ of forests F^1 and F^2 respectively (the first robot vertex in a depth-first search of the forest). All these data are locally stored by each robot. In this way, there is no common coordinator or a shared memory of the robots' system.

In detail, the robots, depending on their positions in the forests, (e.g., whether they are in the augmenting path, in forest F^1 or F^2, etc.) change their roles, and accordingly do some of the steps of the distributed Hungarian algorithm. In the following we describe the robots actions according to their roles.

Initialization (done by each robot r):

- Let $m_r := 0$ (robot r is initially free, i.e., unmatched); let $\alpha_t := 0$ for all the targets t, and $\beta_r := \min_t\{\gamma_{rt}\}$.
- Initialize the pointers in such a way that robot r is connected only with its next older brother robot $(r-1)$ and next younger brother robot $(r+1)$ and has no father and no sons. This means that the forests are initialized as described in Section 4.3.

- Each robot positioned in F^2 determines $[\sigma_r, t_r]$ without exchanging any message with the others.

Root robots $r(F^1)$ and $r(F^2)$ start the message exchange based on depth-first search (DFS) along the forests F^1 and F^2, respectively, asking if there is a free robot. In the contact with the first free robot, the same informs $r(F^2)$ to start a new iteration. In the following we describe the actions of the robots in respect to their roles during each iteration.

- Calculating $[\delta, (r^*, t^*)]$. All the robots $r \in F^2$ participate in calculating $[\delta, (r^*, t^*)]$. The calculation of δ starts from the root robot $r(F^2)$ and follows via the exchange of messages through DFS of F^2. When the values of $[\delta, (r^*, t^*)]$ are found, the information is transmitted to all the robots in F^2 through message exchange based on DFS starting from the root robot of F^2. The same DFS message passing applies to inform the robots in F^1.
- *Dual variables updating*. All the robots r do the following: for each target $t \in F^1$ set $\alpha_t := \alpha_t - \delta$; moreover, if robot r belongs to F^1, set $\beta_r := \beta_r - \delta$. No messages are exchanged in this phase.
- The father robot of t^* (if any), returns the information about himself and about his oldest son to all the robots through the DFS message passing.
- Root robot $r(F^2)$ informs r^* in F^2 to start the next phase of augmenting or nonaugmenting the path.
- *Augmentation*. If r^* is free (see Fig. 2), then it initiates the augmentation of the matching. Over the message passing from r^* and the robots in the augmenting path, robot r^* orders the swapping of the free and the matched edges along the augmenting path. It sends the messages to the robots in F^2 (through DFS message passing in F^2, starting from robot $r(F^2)$) with the information about the new connection with the subtree of target t^* in F^1. The information contains t^*, and all the objects in the tree going backwards in F^1 up to the first target t without robot father along the augmenting path $(t, \ldots, t^*, r^*)$. The robots simultaneously update the matching in respect to the messages received from r^* and update the pointers to the adjacent robots so that the tree of t^* in F^1 rooted in t moves to F^2 and gets connected to root r^* over t^* (see Fig. 3). The robots through the DFS message passing get the updated information regarding the position of the targets in respect to the forests. Each robot r positioned in F^2 calculates the new $[\sigma_r, t_r]$ without exchanging any messages with the others.
- *No_Augmentation*. If r^* is already matched (see Fig. 4), then it does not augment the matching. It sends the message to $r(F^2)$ to inform, through the DFS message passing, all the robots r in F^2 to decrease minimum slack values σ_r by δ; r^* disconnects from forest F^2 attaching itself together with its subtree to F^1 over t^* (see Fig. 5). Each robot r in F^2 updates the new $[\sigma_r, t_r]$ data comparing the old value of minimum slack σ_r with the slack value $(\gamma_{rt} + \alpha_t - \beta_r)$ for each target t in the subtree routed at r^* in F^1, which results in $O(n^2)$ exchanged messages.

The total computational time is $O(n^3)$ as well as the total number of messages exchanged by the robots; nonetheless, the computational time required to perform the local calculation by each robot is $O(n^2)$. Therefore, differently from other known distributed algorithms for the assignment problem recalled in Section 2, the proposed algorithm runs in strongly polynomial time.

Regarding the robustness of the proposed method, if the robot during the execution of the algorithm stops responding, it is considered erroneous and is eliminated from the further calculations. In the case where the robot was unmatched in the forest F^2, the calculation continues without any modifications, ignoring the robot in question. Otherwise, the algorithm starts from the beginning excluding the same.

5. Simulation results

The robotic Multi-Agent System was simulated in the C language using the CPLEX 8.0 callable library for the solution of problems P_i of the upper level and was run on a PC with a 2.2 GHz dual-core duo CPU and 4 GB of RAM.

The production system instances on which the simulation is demonstrated has $P = 50$ tasks (products). The number of time periods of the planning horizon is $T = 100$, with the duration of each period $\Delta t = 1$. The external demand rate of product i is modeled as $d_i(k) = \bar{d}_i \cdot [1 + \delta d_i(k)]$, for $k = 1, \ldots, T$, where $\bar{d}_i$ is the average demand rate of the product produced by task i in the planning horizon and $\delta d_i(k)$ are values generated by an ARMA(2, 2) as in Example 2 of Gaalman (2006). In our numerical example, the average demand is $\bar{d}_i = 5$, for each product i.

The production system is simulated with the production executed by different number of robots, N. In order to cover a wide range of cases (from very highly congested to very low-congested), we consider the number N of robots in the range from 10 to 300. $\overline{N} = 250$ is the number of robots for which the average production rate of the system $\overline{N}\frac{1}{P \cdot T}\sum_{i=1}^{P}\sum_{k=1}^{T} r_i(k)$ (approximately) equals the total average demand rate $\sum_{i=1}^{P} \bar{d}_i$.

The values of the unitary production costs are given for the whole planning horizon. Notice that when the unitary backlog cost $b_i(k)$ is not large enough in respect to the unitary manufacturing cost $c_i(k)$ and the robot hiring cost $\rho_i(k)$, it may happen that, even if the system has enough capacity to clear all the demand, a positive backlog is found at the end of the planning horizon. For this reason, in the numerical case considered here, we assume for each task (product) i the following relation among the parameters of the cost function:

$$h_i(k) \leqslant c_i(k) + \frac{\rho_i(k)}{r_i(k)} \leqslant b_i(k) \tag{18}$$

According to (18), the demand is fulfilled if there are enough production resources available, and, therefore, backlog at the end of the planning horizon can be used as a measurement of congestion. For simplicity we assume that the unitary costs of backlog, $b_i(k)$, inventory, $h_i(k)$, manufacturing, $c_i(k)$, and robot rental, $\rho_i(k)$, are constant and equal for all the tasks in all the time periods. In particular we assume $h_i(k) = 4$, $\rho_i(k) = 4$, $c_i(k) = 1$, and $b_i(k) = 5$ $\forall i$, $\forall k$.

Finally, the robot production capacity $r_i(k)$ is also assumed to be constant and equal for all the tasks and time periods. In particular, we assume $r_i(k) = 1$ for each task i and time period k. In this way, all the considered mixed integer problems are solved within a fraction of 1 s by CPLEX, and, hence, we are able to execute each simulation in less than 2 min; notice, however, that providing a CPU time efficient approach for the considered scheduling problem is out of the scope of this paper.

For the analysis of the results, notice that the stopping criteria of the upper level decentralized algorithm are a maximum number of iterations, in our problem instance equal to 1000, and an error between the best upper and lower bound found, equal to 0.01 (whichever of those two is satisfied first). The performance of the decentralized model is evaluated with respect to that of the centralized optimization model with objective function being the sum of the objective functions (1) of problems P_i, subject to constraints (2)–(5) for each $i \in \{1, \ldots, P\}$, and constraint $\sum_{i=1}^{P} n_i(k) \leqslant N$, for each $k \in \{1, \ldots, T\}$.

Let c_D be the best solution value of the total production cost (social welfare) obtained for the decentralized model by applying the proposed approach, and let c_C be the optimal solution value of the centralized one. Performance of the decentralized model in respect to the centralized one is evaluated measuring the relative *gap* (in percentage) $g = [(c_D - c_C)/c_C] \cdot 100$, that provides an estimation of

the relative extra-cost incurred in the decentralized model in respect to the centralized one for not having at disposal all the system information to optimally reconfigure and distribute the available robots.

In the simulation of the MRTA problem on the lower level of the proposed framework, we assume that the robots move within a convex regular decagon with a circumscribed circle of radius $r_c = 100$ with 10 production centers positioned on the vertices of the same, and with the locations ℓ_i, with $i = 1, \ldots, 50$, of the given 50 tasks equally distributed on the 10 production centers. It is assumed that the robot depot is placed in the center of the decagon, and all the robots are placed in the depot at the beginning, and have to turn back to the depot at the end of the production horizon, all traveling equal distances $r_c = 100$ from the depot to their allocated task positions and vice versa. The time of the robots' rearrangement and their traveling times from one position to another are negligible in respect to the duration Δt of each period. When changing the production position from one task to another, the robots choose the shortest distance path. We do not enter here in the details of collision avoidance since it is not a subject of this paper.

Fig. 7 shows the trend of costs c_D, c_C, and gap g on the upper level, and Fig. 8 presents the solution of the lower level in terms of the total distance traveled by robots in correlation to the varying number of robots in the system.

Since the total mean demand rate in each time period equals 250, the system with up to approximately $N = 230$ robots is very highly congested, and, as a result, by increasing the number of robots N from 0 up to 230, the total production costs linearly decrease since the prevailing are the costs due to the accumulated backlog and are of the same order for the centralized and decentralized model. In the range from 230 to 280 robots, the decentralized model sustains higher costs of production due to the lack of complete system's information for the reconfiguration of the production resources. At $N = 230$, the relative gap between the decentralized and centralized model is 1%, after which it increases and has a peak value at $N = 250$, where the total robot production rate equals the total mean demand rate per period. This gap between decentralized and centralized model decreases with the less congested instances up to $N = 270$, where the gap falls to 3%. By further increasing the number of robots, at $N = 288$ the gap equals zero. Indeed, the maximum number of robots needed for both the centralized and decentralized model is $N = 288$. Increasing N beyond this number does not affect the solution value, as can be seen in Figs. 7 and 8.

In fact, when the total number of robots in the system is significantly lower than $\overline{N}$, the system is highly constrained, and both the decentralized and the centralized approach are characterized by very high and comparable backlog costs. The total costs of the two approaches differ in practice in the values of the other cost components, that are indeed negligible in respect to the backlog cost. Oppositely, when the total number of robots in the system is significantly larger than $\overline{N}$, the local optimal solutions of the task agents' subproblems are likely to satisfy (resource) constraints (6), and then the set of local solutions of the decentralized approach is equivalent to the centralized solution. Moreover, in this case, there is no backlog cost and the total cost of both approaches is low. In the intermediate case, i.e., when N approaches $\overline{N}$, the gap g among the total costs of the decentralized and the centralized approach increases. This is due to the fact that the latter approach can take

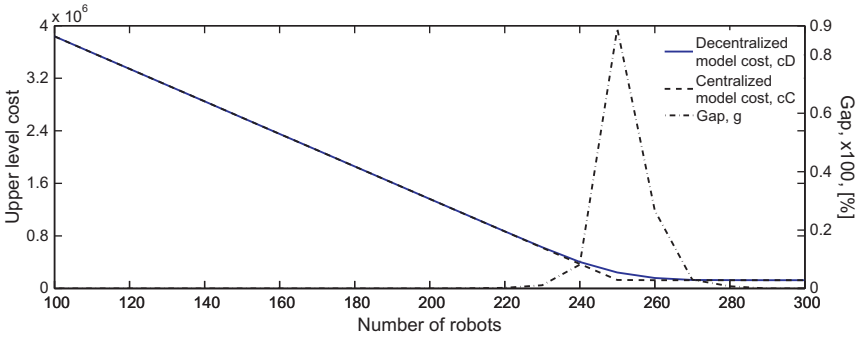

Fig. 7. Decentralized vs. centralized model performance on the upper level.

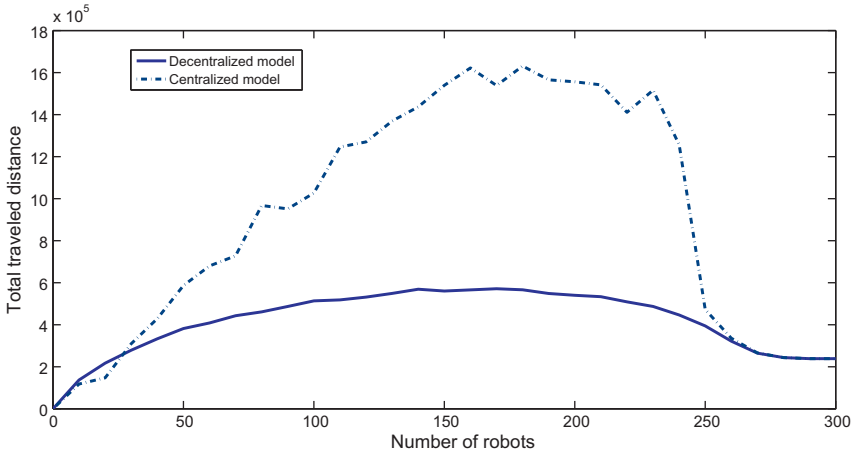

Fig. 8. Distributed vs. centralized MRTA dynamic model, total distance traveled by robots.

full advantage of having at disposal all the information, which allows it to satisfy the resource constraint with a lower backlog cost in respect to the one payed in the decentralized case.

On the second level, total distance traveled by the robots varies with N and follows approximately a bell shaped curve (as can be seen in Fig. 8). Initially, the centralized model has a lower translocation cost for up to $N = 30$, after which the traveled distance of the centralized model increases much more rapidly than the one of the decentralized model. In the congested instances where $230 < N < 280$, robots in the centralized model travel up to 3 times more than the ones in the decentralized case. This is where the centralized model seeks to diminish the total production cost by better reallocating the robots having all the system's information at disposal.

Since, on the upper level, the decentralized model is more costly and gives no better results than the centralized one due to the lack of complete system's information in the decision processes of individual agents, and because of the additional cost λ which each task has to pay in the negotiation for acquiring the robots, the system is not as flexible and the robots do not change production positions as frequently as in the centralized system. On the second level, this lack of flexibility results, in general, in inferior required total movement distance of the robots' reallocation as seen from Fig. 8.

6. Conclusion

In this paper, a dynamic factory layout with a set of mobile production resources (e.g. robots) carrying out the production, has been proposed. Notice, however, that the latter can also be used to increment the production capacity of already existing machines in the traditional factories. In respect to a classical layout where the production locations are placed on static positions, a less constrained facility is obtained, which allows to postpone critical design decisions to the operative level and makes the plant more capable of responding to a fluctuating external demand. The proposed solution can be considered as an extension of traditional factories since every control policy implemented in plants with a static layout can be applied in the dynamic factory proposed in this paper, while the viceversa is not true.

This extension is at the price of a more complex scheduling problem, involving, for each period of the planning time horizon, the determination of the robots' positions in order to minimize all the production costs. Owing to this complexity, and/or also for the inherent decentralized structure of the system, a two-level decentralized multi-agent system production scheduling architecture was proposed: at the first level the agents are the tasks which compete for the robots, and at the second level the agents are the robots which reallocate themselves among different tasks to satisfy the requests coming from the first level. An iterative auction based negotiation protocol was used at the first level while the second level resolves a Multi-Robot Task Allocation problem through a distributed version of the Hungarian Method.

The advantages of the decentralized architecture with respect to the centralized one were discussed in the paper and include robustness and efficiency (notice that the decentralized approach can be in some cases the only viable solution for the structural organization of the system). From the simulation results it can be observed, however, that the lack of information associated with the decentralized solution implies larger production costs, with respect to the centralized approach when the system has a strict number of robots necessary to carry out requested production: in this case, in fact, the centralized approach is characterized by a much higher robot displacement distance with respect to the decentralized solution; the latter tending to under-utilize the production resources and representing a sub-optimal control policy.

Moreover, the proposed dynamic layout (even if sub-optimally controlled) shows a significant superiority with respect to a static plant, in particular if the fluctuation of the demand is characterized by low frequency components or by a drift in the average demand: in these cases, in fact, the strategic static sizing of the plant (i.e. the decision on the number of machines which must be dedicated to a particular product) may result completely inappropriate for long time periods, while a dynamic solution with mobile production robots adequately adapts their roles and positions in the plant to track the perturbations.

References

Barahona, F., & Anbil, R. (2000). The volume algorithm: Producing primal solutions with a subgradient method. *Mathematical Programming, 87*, 385–399.

Bertsekas, D. P. (1991). *Linear network optimization: Algorithms and codes*. Cambridge, USA: MIT Press.

Bertsekas, D. P. (1992). Auction algorithms for network flow problems: A tutorial introduction. *Computational Optimization and Applications, 1*(1), 7–66.

Bertsekas, D. P., & Castanon, D. A. (1989). The auction algorithm for the transportation problem. *Annals of Operations Research, 20*, 67–96.

Bertsekas, D. P., Castanon, D. A., Eckstein, J., & Zenios, S. A. (1995). Parallel computing in network optimization. In M. O. Ball et al. (Eds.). *Network models – handbooks in operations research and management science* (Vol. 7, pp. 330–399). Amsterdam, The Netherlands: Elsevier.

Bischoff, R., Kurth, J., Schreiber, G., Koeppe, R., Albu-Schäffer, A., Beyer, A., et al. (2010). The KUKA-DLR Lightweight Robot arm – a new reference platform for robotics research and manufacturing. In *Proceedings of ISR/ROBOTIK*.

Burkard, R. E., & Çela, E. (1999). Linear assignment problems and extensions. In D. Z. Du & P. M. Pardalos (Eds.). *Handbook of combinatorial optimization* (Vol. 4(1), pp. 221–300). Dordrecht, The Netherlands: Kluwer Academic Publishers.

Buyya, R., Abramson, D., Giddy, J., & Stockinger, H. (2002). Economic models for resource management and scheduling in grid computing. *Concurrency and Computation: Practice and Experience, 14*, 1507–1542.

Chen, H., Chu, C., & Proth, J. M. (1998). An improvement of the Lagrangean relaxation approach for job shop scheduling: A dynamic programming method. *IEEE Transactions on Robotics and Automation, 14*(5), 786–795.

Chevaleyre, Y., Dunne, P. E., Endriss, U., Lang, J., Lemaitre, M., Maudet, N., et al. (2006). Issues in multiagent resource allocation. *Informatica, 30*(1), 3–31.

Christensen, J. H. (1994). Holonic manufacturing systems: Initial architecture and standards directions. *Proceedings 1st Euro workshop on holonic manufacturing systems* (Vol. 14). Hannover, Germany: HMS Consortium.

Dugnay, C. R., Landry, S., & Pasin, F. (1997). From mass production to flexible/agile production. *International Journal of Operations and Production Management, 17*, 1183–1195.

Florian, M., Lenstra, J. K., & Rinnooy Kan, A. H. G. (1980). Deterministic production planning: Algorithms and complexity. *Management Science, 26*(7), 669–679.

Gaalman, G. (2006). Bullwhip reduction for ARMA demand: The proportional order-up-to policy versus the full-state-feedback policy. *Automatica, 42*(8), 1283–1290.

Gerkey, B. P., & Mataric, M. J. (2003). A framework for studying multi-robot task allocation. In A. C. Shultz et al. (Eds.). *Multi-robot systems: from swarms to intelligent automata* (Vol. 2, pp. 15–26). Dordrecht, The Netherlands: Kluwer Academic Publishers..

Gerkey, B. P., & Mataric, M. J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of Robotics Research, 23*(9), 939–954.

Giordani, S., Lujak, M., & Martinelli, F. (2009). A decentralized scheduling policy for a dynamically reconfigurable production system. *Lecture Notes in Artificial Intelligence, 5696*, 102–113.

Giordani, S., Lujak, M., & Martinelli, F. (2010). A distributed algorithm for the multi-robot task allocation problem. *Lecture Notes in Artificial Intelligence, 6096*, 721–730.

Held, M., Wolfe, P., & Crowder, H. D. (1974). Validation of subgradient optimization. *Mathematical Programming, 6*(1), 62–88.

Helms, E., Schraft, R.D., Haegele, M. (2002). Rob@work: Robot assistant in industrial environments. In *Proceedings of the 11th IEEE int workshop on Robot and Human interactive Communication, ROMAN2002* (pp. 399–404). Berlin, Germany.

Huang, P. Y., & Chen, C. S. (1986). Flexible manufacturing systems: An overview and bibliography. *Production and Inventory Management, 27*(3), 80–90.

Jans, R., & Degraeve, Z. (2008). Modeling industrial lot sizing problems: A review. *International Journal of Production Research, 46*(6), 1619–1643.

Kraus, D. S. (2001). *Strategic negotiation in multiagent environments*. Cambridge, MA, USA: MIT Press.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistic Quarterly, 2*, 83–97.

Kutanoglu, E., & Wu, S. D. (2006). Incentive compatible, collaborative production scheduling with simple communication among distributed agents. *International Journal of Production Research, 44*(3), 421–446.

Kwok, K. S., Driessen, B. J., Phillips, C. A., & Tovey, C. A. (2002). Analyzing the multiple-target-multiple-agent scenario using optimal assignment algorithms. *Journal of Intelligent and Robotic Systems, 35*(1), 111–122.

Lueth, T.C., Laengle, T. (1994). Task description, decomposition and allocation in a distributed autonomous multi-agent robot system. In *Proc of the IEEE/RSJ/GI international conference on intl. robots and systems 94': Advanced robotic systems and the real world* (pp. 1516–1523).

Mun, J. (2002). *Real options analysis: Tools and techniques for valuing strategic investments and decisions*. Hoboken, NJ, USA: Wiley.

Orlin, J. B., & Stein, C. (1993). Parallel algorithms for the assignment and minimum cost flow problems. *Operations Research Letters, 14*, 181–186.

Papadimitriou, C. H., & Steiglitz, K. (1982). *Combinatorial optimization: Algorithms and complexity*. Englewood Cliffs, NJ, USA: Prentice-Hall Inc..

Roundy, R. O., Maxwell, W. L., Herer, Y. T., Tayur, S. R., & Getzler, A. W. (1991). A price-directed approach to real-time scheduling of production operations. *IIE Transactions, 23*(2), 149–160.

Schneider, J., Apfelbaum, D., Bagnell, D., Simmons R. (2005). Learning opportunity costs in multi-robot market based planners. In *Proceedings of 2005 IEEE international conference on robotics and automation* (pp. 1151–1156).

Selim, H. M., Askin, R. G., & Vakharia, A. J. (1998). Cell formation in group technology: review, evaluation and directions for future research. *Computers and Industrial Engineering, 34*(1), 3–20.

Shen, W., Hao, Q., Wang, S., Li, Y., & Ghenniwa, H. (2007). An agent-based service-oriented integration architecture for collaborative intelligent manufacturing. *Robotics and Computer-Integrated Manufacturing, 23*(3), 315–325.

Shen, W., Hao, Q., Yoon, H. J., & Norrie, D. H. (2006). Applications of agent-based systems in intelligent manufacturing: An updated review. *Advanced Engineering Informatics, 20*(4), 415–431.

Smith, S.L., Bullo, F. (2007). Target assignment for robotic networks: Asymptotic performance under limited communication. In *Proceedings of American Control Conference 2007 (ACC'07)* (pp. 1155–1160). New York, USA.

Tan, J. T. C., Duan, F., Zhang, Y., Watanabe, K., Kato, R., & Arai, T. (2009). Human-robot collaboration in cellular manufacturing: Design and development. In *Proceedings of the 2009 IEEE/RSJ internat. conf. on intell. robots and systems* (pp. 29–34). IEEE Press.

Wang, C., Ghenniwa, H., & Shen, W. (2008). Real time distributed shop floor scheduling using an agent-based service-oriented architecture. *International Journal of Production Research, 46*(9), 2433–2452.

Wooldridge, M. (2002). *Introduction to multi-agent systems*. Hoboken, NJ, USA: John Wiley and Sons.

Zavlanos, M.M., Spesivtsev, L., Pappas, G.J. (2008). A distributed auction algorithm for the assignment problem. In *Proceedings of 47th IEEE conf. on decision and control* (pp. 1212–1217). Cancun, Mexico.