Autonomous Robots manuscript No.
(will be inserted by the editor)

Top–Down vs Bottom–up Methodologies in Multi–Agent System Design

Valentino Crespi[1], **Aram Galstyan**[2], **Kristina Lerman**[2]

[1] Department of Computer Science, California State University, Los Angeles

[2] USC Information Sciences Institute

Received: date / Revised version: date

Abstract Traditionally, two alternative design design approaches have been available to engineers: top-down and bottom-up. In the top-down approach, the design process starts with specifying the global system state and assuming that each component has global knowledge of the system, as in a centralized approach. The solution is then decentralized by replacing global knowledge with communication. In the bottom-up approach, on the other hand, the design starts with specifying requirements and capabilities of individual components, and the global behavior is said to emerge out of interactions among constituent components and between components and the environment. In this paper we present a comparative study of both approaches with particular emphasis on applications to multi–agent system engineering and robotics. We outline the generic characteristics of either

approach from the MAS perspective, and identify three elements that we believe should serve as criteria on how and when to apply either of the approaches. We demonstrate our analysis on a specific example of load balancing problems in robotics. We also show that under certain assumptions on the communication and the external environment, both bottom–up and top–down methodologies produce very similar solutions.

1 Introduction

Traditionally, two alternative design methodologies, called top-down and bottom-up, have been used in building complex systems. In the top-down methodology, the design starts from the top with the assumption that resources are globally accessible by each subcomponent of the system, as in the centralized case. The specification is then defined in terms of the global systems state and implies that each individual component should be able to retrieve or estimate, with sufficient accuracy and within a reasonable time delay, resources that are local to other agents of the system. Under these conditions the properties of a classical centralized solution to the global specification are expected to hold, up to some tolerable performance degradation, also in a decentralized environment. In the bottom-up methodology, on the other hand, the rules of agent interactions are designed typically in an *ad hoc* manner, although recent work has attempted to formalize the design process for some applications [1]. In systems designed starting from the bottom, the global state of all the components is assumed to be impos-

sible to obtain, and the desired collective behavior is said to emerge from interactions among individual agents and between the agents and the environment. In summary, in the top-down design the final distributed solution is obtained as a process of relaxation of the constraints that require instant access to remote resources with infinite precision. The bottom-up design starts with a rigorously pre–decided set of rules for the individual behaviors and local interactions and then proceeds with the inference of the global emergent behavior.

While the question of which design is appropriate for a given system extends over the most diverse areas in computer science and computer engineering [2–6], we intend to conduct a comparative study of the two approaches in a typical domain of multi-agent systems engineering. The main questions we are interested in understanding are the limitations and advantages of either approach, and in establishing criteria for their applicability.

In this paper we focus on demonstrating how to apply the two methodologies separately to the same case study. We would like, in both cases, to address questions such as: what are the analytical and design challenges? What are the mental processes that lead a designer to the final solution? What kind of assumptions are crucial in order to expect a high performance solution? And do the two approaches produce the same solution?

In Section 3 we provide a detailed description of the two methodologies in comparison. We then investigate nontrivial case studies, in the robotics domain, that emphasize commonalities and differences between the two ap-

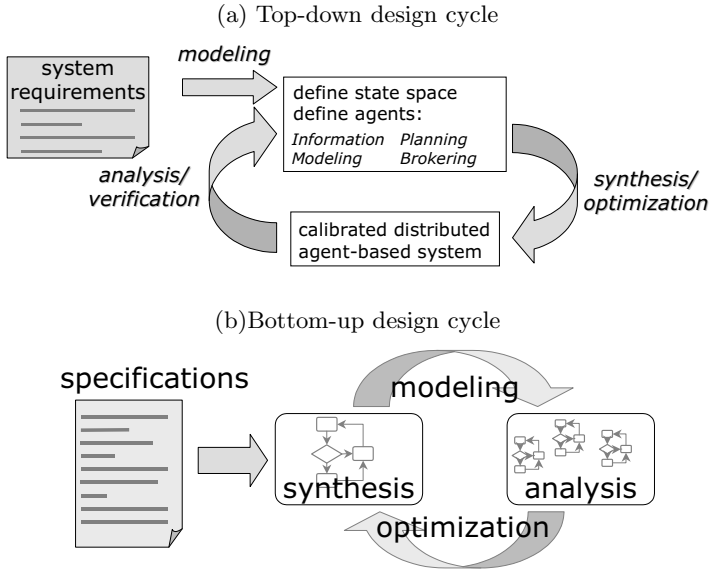

Fig. 1 Conceptual representation of the two design methodologies

proaches (Section 4). Finally we conclude this work with a discussion on the results and future developments (Section 5).

2 Previous Research

Evolutionary methods have a long track record in robot controller synthesis [7,6]. Researchers have begun to formalize the design of different classes of agents. Jones and Mataric [1,8] have proposed a methodology for formally specifying the task domain and automatically synthesizing the controllers which satisfy different spatio-temporal coordination requirements. Their approach is appropriate to the construction domain, which requires a specific sequence of actions (placing bricks) to be performed. Harper and Winfield [9] present a methodology for designing behavior-based controllers

using Lyapunov stability theory. They illustrate the methodology by constructing a position control system for a subsumption architecture-based agent (an airplane). In another contribution Winfield [10] uses temporal logic to design verifiable controllers for an MRS. The emphasis in that work is on the formal definition of agent and group behaviors rather than controller synthesis. McNew and Klavins [11] use graph grammars to synthesize controllers for certain tasks where the network topology is important.

For some classes of multi-agent systems an agent controller can be viewed as an automaton, or a grammar, which can be automatically learned from samples of sentences of the language generated by that grammar. A robot's task description, for example, is a sequence of symbols describing the rules the robot follows while executing a task. A task description can be viewed as a set of sentences generated by the grammar. Ott and Lerman [12] used grammar induction to learn the controller from a relatively small subset of all possible strings (task descriptions) of the grammar.

This paper borrows techniques from the mathematical analysis of the collective behavior of multi-agent systems. Our earlier work proposed a formal framework for creating mathematical models of the collective behavior in groups of multi-agent systems [13]. This framework was successfully applied to study the collective behavior in groups of robots [14–16].

3 Foundations of the Design

In this section we describe the fundamental components of each design approach.

3.1 Top Down

Employment of Agent-based technology in the design of distributed systems has been notoriously limited by the lack of a general unified model for performance estimation. The difficulty in this matter concerns both the qualification of candidate solutions and the quantification of their performance.

In response to this need, a quantitative top-down design methodology that combines ideas from Agent Based computing and Classical Control Theory has been proposed [17,18]. This methodology is based on the hypothesis that agent systems can be decomposed, by analogy to what happens in classical systems theory, into three ingredients: **observation, state estimation and control**, with the ultimate goal of being able to derive reliable performance predictions and guarantees [19]. According to this approach the design process follows three main phases as outlined in the subsequent subsections. The relationship between the phases is illustrated in Fig. 1(a).

3.1.1 Modeling The purpose of this phase is twofold: a) agents in the system are identified and categorized based on the taxonomy described below; and b) a description of the knowable global state of the system is produced:

- **information agents** that gather information about the environment and allow dissemination of it in a distributed manner (these would be "sensors" in classical systems theory for example);
- **modeling agents** that collect data from many information agents and update internal knowledge, i.e., produce new estimates of "real world" state (these would be state estimators, like Kalman filters, in classical systems theory);
- **planning agents** that use the current world state estimates, the viable action or control options and the current goals or intentions to plan new actions to carry out. These agents may need additional information for their planning operation, and so they may task **brokering** agents to report on available resources such as additional state and action information. This last category of agents have no counterpart in classical control theory where resources are hardwired to components and so there is no need for brokering.

3.1.2 Synthesis Agent controllers are designed following a three-stage top-down process. At first, it is assumed that each agent can access remote resources local to other agents instantly and with infinite precision. So an initial centralized solution is designed. Next, limitations of the distributed environment are applied and so the visibility of each agent is gradually reduced. Consequently inter-agent communication issues arise as now each agent needs to replace global resources with local resources. As a result a fully decentralized solution is produced. Finally, and this is a pre-optimization

stage, the fully distributed solution must be calibrated in order to determine a minimum level of agent visibility necessary for the system to perform reasonably well compared to the ideal non-distributed solution attained at stage 1 of the process. More specifically:

a) Centralization The first stage of the design is characterized by the assumption that each autonomous subcomponent of the system has full knowledge of the global state through remote access to all the resources and events that have been acquired or recorded by any other subcomponent of the system. This produces a centralized solution that however runs locally on each subcomponent.

b) Distribution The second step is to remove this assumption and replace global resources with local resources compatibly with the constraints of the given distributed environment. This introduces communication issues due to the fact that now agents must acquire information through interactions with other *locally reachable* (the notion of locally reachable is context dependent) components in order to infer the most likely global state of the system. The result of this stage is an algorithm that runs locally and uses local information. It records all the events that occur within its range and maintains a representation (hypothesis) of the global state. When two agents come into contact, an exchange of knowledge and experience takes place on a peer-to-peer basis.

c) Calibration The final stage consists of calibrating the obtained distributed solution via parameter tuning. This in turn is achieved through adaptation and learning.

3.1.3 Analysis/Optimization The inter-agent communication must be optimized in order for the distributed system to perform as predicted at the beginning of the synthesis phase. The **Analysis** conducted in this phase may lead to a review (feedback) of the original Modeling of the agent system thus creating a *cycle* as depicted in 1.

The top-down synthesis of agent controllers relies heavily on the ability of agents to approximate well information and resources that are collectively known by any other agent in the system. In other words, the problem is initially solved neglecting communication delays (between agents) and bandwidth limitations and then successively refined by introducing those constraints and applying approximation techniques from distributed computing and Control Theory. This is the main point: being able to apply powerful tools and techniques from well-established disciplines such as Distributed Computing, Optimization Theory and Control Theory in order to obtain performance guarantees on the final distributed algorithm.

While stage 3 of the synthesis aims at optimizing various parameters of the distributed algorithm (communication ranges, bandwidth, etc.), the focus of the Optimization phase is on the characterization of what relevant information should be maintained locally and communicated to peers during local interactions. This is important since during the decentralization the

centralized solution the representation of the global state is replicated into many locally maintained approximations of it. One of the crucial problems that arises is how to minimize redundancies (determine relevant information) and characterize the minimum amount of information that needs to be exchanged in local interactions (reduce communication complexity).

In summary this methodology (instances can be found in [20, 21]) is the summa of an Agent Taxonomy derived from classical Control Theory and inspired by an Agent Based vision of the world and a Top-Down design process. Its purpose is to be able to apply analytical and rigorous tools from Classical Control Theory to Distributed Control Systems.

3.2 Bottom–Up

The bottom-up design methodology is very popular for producing autonomous, scalable and adaptable systems often requiring minimal (or no) communication. It has been used to control robotic systems (e.g., [22–24]), embedded systems, sensor networks [25], and information agents [26] among others. Within this design paradigm, a researcher specifies an agent's behavior, its interactions with other agents and the environment. The behaviors are specified by an algorithm called a controller. Useful collective behavior, be it load balancing in Grid computing, or clearing an arena from pucks in a robotics application, is said to *emerge* out of interactions among agents and between agents and the environment.

In the applications listed above, the agents themselves are rather simple and do not rely on abstract representation, planning, or higher order reasoning functions. These agents usually use reactive control, although increasingly, they are being endowed with a capacity to learn. Generally speaking, the simple agents described above can be represented as stochastic Markov processes or a probabilistic finite state automaton. Within this framework, the design process consists of three steps: Synthesis, Modeling and Analysis, and Optimization, as illustrated in 1(b). Below we describe these steps in more details.

3.2.1 Synthesis In the Synthesis phase one has to define the agent controller which can be described by an automaton that is the behavioral representation of an agent. In the case of a reactive agent (i.e., one that makes a decision about what action to take based on its current state and input from its sensors) the controller can be characterized by a finite state automaton (FSA). Each state of the automaton represents the action or a behavior the agent is executing, with transitions coupling it to other states. Consequently, the behavioral dynamics of a reactive agent can be considered as an ordinary Markov process. Consider, for example, a foraging task, where the goal is for robots to collect pucks scattered about an arena and deposit them at a pre-specified home location. A single robot engaged in a foraging task will have to execute the following behaviors: (i) *searching* for pucks by wandering around the arena, (ii) puck *pickup* and (iii) *homing* or bringing the puck to a pre-specified home location. Transition from *searching*

to *pickup* is triggered by a puck being sensed in the arena, from *pickup* to *homing* by the gripper closing around the puck, and transition from *homing* to *searching* is caused by the the robot reaching the home destination. A task description is a sequence of symbols specifying the rules robots follow while executing a task: e.g., how observations trigger transitions from one state to another.

Given that FSAs are equivalent to regular grammars, Ott and Lerman [12] showed that the automatic controller synthesis can be treated as a *grammar induction* problem.[1] In other words, given a task description — or sentences generated by some grammar — one can learn the grammar, and equivalently, the automaton, that produced these sentences. Moreover, Ott and Lerman showed that the method was able to generate correct grammars from relatively short task descriptions by learning to generalize properly.

3.2.2 Modeling and Analysis Once a controllers for individual agents have been constructed, one needs to develop a mathematical model of the collective behavior. Remarkably, the finite automaton of a single agent in many cases can be used for adequately describing the macroscopic or collective behavior of a large-scale system composed of many such controllers. In particular, Lerman et. al. have developed models based on Stochastic Master Equation and its first moment, Rate Equation, to describe the average collective behavior from the details of the agent automaton. The model consists

[1] That work treats a more general problem of context-free grammar induction, but can be easily extended to regular grammar induction.

of coupled differential equations describing how the average group behavior changes in time. This modeling approach is based on the theory of stochastic processes. Many types of computer processes (including robots, sensor nodes, Grid agents) can be represented as stochastic processes, because they are subject to unpredictable influences, including environmental noise, errors in sensors and actuators, forces that cannot be known in advance, interactions with other agents following complex trajectories, etc. One does not assume knowledge of agents' exact trajectories; instead, we model each agent as a stochastic process and derive a probabilistic model of the aggregate, or average, behavior. Such probabilistic models often have very simple, intuitive form, and can be easily written down by examining details of the individual agent control diagram [27,14,28].

3.2.3 Optimization Mathematical models can be used not only to validate the controller, but also to estimate individual parameters that optimize group-level performance. Using mathematical analysis one can finally answer a number of design questions. Controller synthesis may produce a range of values for internal parameters that result in a valid controller, but these different controllers might result in different group–level efficiency. For example, some parameter values may lead to faster convergence to the desired steady state, while others will lead to smaller deviations from it. Moreover, in a case when an agent's controller is represented as a Finite State Automata (FSA), analysis can be used not only for estimating the parameters of the controller, but also suggesting appropriate structure and

transition probabilities for the FSA, so that the desired global behavior will be achieved on average.

4 A Case Study

In the preceding sections we described the main characteristics of two design methodologies from the multi–agent system perspective. In this section we further elaborate on the features of the two approaches on an example multi–robot load balancing problem. This application, while simple, is non-trivial and illustrative of the challenges faced in designing a generic multi-agent system.

Here is the scenario, introduced by [29] and analyzed by [16]: In a closed arena a known number of $M = R + G$ (G are green and R are red) pucks have been disseminated in unknown positions. The numbers of either type of puck, R and G, are unknown and can even change in time. We deploy N robots equipped with a red lamp and a green lamp to collect the pucks. Each robot can be foraging for one type of puck at any given time and its foraging state will be displayed by lamp color to other robots. We assume that robots have a memory buffer of a certain length where they can store their recent observations of pucks and other robots. The goal of the application is to have, on average, the same proportion of red to green robots as the proportion of red to green pucks in the arena. The task is to define color–selection rules based on robots' memory and interaction with other robots and/or environment.

4.1 Top-down solutions

4.1.1 Distributed Potential and Gradient Descent How should we approach this problem using the top-down methodology? We assume that each component, in this case robot, is capable of accessing resources that are local to other components, as happens typically in any centralized control system.

The general idea is to start with this assumption in order to apply well-known and tested methods from classical control theory (i.e., machine learning, parameter estimation, gradient descent, etc). Then, at a second stage, we study the effects of the decentralization and we determine conditions under which the solution still performs satisfactorily.

In the following we define an objective function to be minimized using a gradient descent method that will lead to a distributed algorithm. This function should contain as constants all the quantities that are known to the robots, i.e., all the observations. The variables instead become the descriptors of the changing state of the robots. As in dynamical systems theory robots will try to optimize the objective function by following an appropriate "trajectory" on the search space towards an optimal point.

Let c_i be the optimal probability that robot i displays green. So, a potential objective might be the following:

$$V(c_1, c_2, \ldots, c_n) = \left(\sum_{i=1}^{n} \frac{c_i}{n} - \frac{\sum_{i=1}^{n} g_i}{\sum_{i=1}^{n} (g_i + r_i)} \right)^2 , \qquad (1)$$

where, n is the total number of robots and g_i (r_i) is the number of green (red) pucks observed by robot i. This function states literally that the aver-

age proportion of green robots *a priori* should be the same as the globally observed proportion of green pucks. In fact we can observe that $V = 0$ is the global optimum and is attained at

$$\sum_{i}^{n} \frac{c_i}{n} = \frac{\sum_{i=1}^{n} g_i}{\sum_{i=1}^{n} (g_i + r_i)} \ . \tag{2}$$

Since each c_i represents the probability that the robot should be green in order to reflect the correct rate of observed green pucks then each robot can guess the color to display by sampling (periodically) from a Bernoulli distribution with probability c_i. So, on average, the rate of green robots should reflect the estimated probabilities. Values c_i are local to robots and are updated in order to attain the global minimum of V. A classical way to achieve this is through a gradient descent optimization method:

$$c_i(t+1) = c_i(t) - \gamma_t \nabla_{c_i} V(\mathbf{c}(t)) \ . \tag{3}$$

Here, γ_t is a sequence chosen according to well-know sufficient conditions that depend on properties of the gradient (see for example [30]). This is to ensure mathematical convergence to a stationary point.

If we now limit the communication range of the robotic agents they will be forced to interact only with other agents at close range. As a consequence each agent will need to approximate its gradient whose exact computation depends on global quantities, i.e., vector $\mathbf{c}$. The result is the following distributed algorithm:

1. Each robot maintains locally a history of observations within a bounded time interval. In particular it keeps a local statistics of the pucks: g_i, r_i. It also maintains a local estimate of the optimal color probability c_i.

2. At fixed intervals of time each robot updates its local c_i variable by applying the following rule that depends only on local quantities: $c_i(t+1) =$

$$c_i(t) - \frac{2\gamma_t}{|N(i)|}\left(\sum_{j\in N(i)}\frac{c_j(t)}{|N(i)|} - \frac{\sum_{j\in N(i)} g_j}{\sum_{j\in N(i)}(g_j+r_j)}\right) \quad (4)$$

where $N(i) = \{j \mid d(i,j) < \rho\}$ with ρ being the communication range of the robots and d the Euclidean distance (in the experimental verification we set γ_t to 0.1).

3. Each robot i decides its own color by sampling from a Bernoulli distribution with mean $c_i(t+1)$.

The analysis of this solution can be conducted using classical tools in Nonlinear Optimization and Distributed Computing [30,21]. First, it is necessary to discuss the possible presence of local minima for V (in this special case there are none but in general we may expect a nontrivial issue). Then we need to establish bounds to γ_t that ensure convergence of the method to a stationary point of V (also we may introduce barrier functions to force $c_i(t)$ within range $(0,1)$). Finally study how the performance degrades as the communication range reduces to zero. Figure 2 (a) provides experimental evidence of the validity of this solution.

4.1.2 Parameter Estimation Another top-down way to look at the same problem is that the various robots need to *estimate*, as the time varies, the

proportion of green vs red pucks and then sample their own color accordingly.

If we assume that the parameters of the population of pucks are constant (g and r do not vary overtime) we can approach the estimation problem using a Least Square (LS) method. Define $p = g/(r+g)$. Each robot i gathers an observation of the environment in the form of a ratio $p_i = g_i/n_i$, where g_i and n_i are respectively the number of green pucks and the total number of pucks observed by robot i.

We may see the combined observations of all the robots as n independent measurements of the form

$$p_i = p + w_i \tag{5}$$

where w_i are n normal random variables with zero mean and common variance σ. Random variables w_i's represent a notion of noise corresponding to what the robot does not know because of lack of experience. We can realistically model w_i as a Gaussian RV under the assumption that the pucks have been disseminated on a grid with a certain degree of spacial uniformity and that the robot explores the area without reversing course over the observation window. The (global) LS Estimator (LSE) of p is then [31]

$$\hat{p} = \arg\min_p \sum_{i=1}^{n} (p - p_i)^2 = \frac{1}{n} \sum_{i=1}^{n} p_i \ . \tag{6}$$

The assumptions on the densities of w_i's imply that $\hat{p}$ is also the Maximum Likelihood Estimator (MLE). In particular it is unbiased and, as the number of observations increases to infinity, it converges to the right value of the parameter.

Finally let us apply the restrictions imposed by the decentralized environment: in this case a limited communication range of the robots that reduces the size of the samples each robot is able to average at any instant of time. These restrictions translate into the following distributed algorithm:

- Each robot maintains locally a history of observations within a bounded time interval. In particular it keeps a local statistics of the pucks: $p_i = g_i/(g_i + r_i)$.
- At fixed intervals of time each robot tries to compute the LS Estimator of the unknown parameter p exchanging values p_i's with all the other robots that are within range:

$$\hat{p}_i = \frac{1}{|N(i)|} \sum_{j \in N(i)} p_j \tag{7}$$

  where $N(i) = \{j \mid d(i,j) < \rho\}$ and ρ is the communication range of the robots.
- Each robot i decides its own color by sampling from a Bernoulli distribution with mean $\hat{p}_i$.

The performance analysis of this solution is conducted by applying well known instruments from the theory of parameter estimation to establish, for example, how the variance of the distributed estimators varies in relation to the communication range and other quantities. This gives us the means to study with precision the time of convergence and the adaptation capabilities of the system to sudden changes of the environment (e.g., changes in the distribution of pucks).

4.2 Bottom–up solutions

While the top-down design cycle started with defining a global potential function Eq. 1 to be minimized, the design process in the bottom-up approach starts with specifying an individual robot controller without any regard to global system properties. Again, let r_i and g_i be the number of red and green pucks respectively in i^{th} agent's observation window. Our task is to construct a controller that prescribes an action for each realization of g_i and r_i. For the present problem two possible actions are choosing red or green foraging state. Hence, the agent's controller can be characterized by a finite states machine with two states, *Red* and *Green*, and transitions between the state with probabilities $f_{R\to G}(r_i, g_i)$ and $f_{G\to R}(r_i, g_i)$. During a sufficiently short time interval, the robot can be considered to belong to a *Green* or *Red* foraging state. This is a very high level, coarse-grained description. In reality, each state is composed of several robot actions and behaviors, such as wandering the arena, detecting pucks, avoiding obstacles, *etc.* However, since we want the model to capture how the fraction of robots in each foraging state evolves in time, it is a sufficient level of abstraction to consider only these states.

The transition probabilities between the states depend on the robot's observations, specifically, on the number of pucks of each type, r_i and g_i that the i^{th} robot has encountered during a certain time interval. Generally speaking, r_i and g_i are random variables that depend on a variety of factors such as a robot's speed, view angle, local density of pucks, and so on. To

make the model amenable to analysis, however, we make a simplifying assumption that the process of encountering a puck is a Poisson process with rate $\lambda = \alpha M_0$ where α is a constant characterizing the physical parameters of the robot such as its speed, view angles, etc., and M_0 is the number of pucks in the arena. Thus, we assume that a robot's observation is independent of the robot's actual physical trajectory, but is instead governed by probabilities determined by simple geometric considerations[2]. Let $R(t)$ and $G(t)$, $R(t) + G(t) = M_0$, be the actual number of red and green pucks respectively (unknown to robots), that generally can be time dependent. The probability that in the time interval $[t-\tau, t]$ the robot has encountered exactly r and g pucks is the product of two Poisson distributions:

$$P(r, g) = \frac{\lambda_R^r \lambda_G^g}{r!g!} e^{-\lambda_R - \lambda_G} \tag{8}$$

where $\lambda_R = \alpha \int_{t-\tau}^{t} dt' R(t')$ and $\lambda_G = \alpha \int_{t-\tau}^{t} dt' G(t')$ are the means the of respective distributions. In the case when the puck distribution does not change in time, $R(t) \equiv R = const$, $G(t) \equiv G = const$, one has $\lambda_R = \alpha R \tau$, $\lambda_G = \alpha G \tau$.

Once we have specified the individual controller and the observation model, the next step is to derive the resulting global behavior. Recall that the goal of the system designer is to achieve a balance in the number of *Red* and *Green* robots that reflect the distribution of red and green pucks. Hence, it is natural to choose the number of robots in each state as variables

[2] This approximation has been shown to produce remarkably good agreements with experiments [32,33].

describing the global state. Let $N_r(t)$ and $N_g(t)$ be the average (or expected) number of *Red* and *Green* robots at time t, $N_r(t) + N_g(t) = N$. During a small time interval $[t, t + \Delta t]$ the i–th robot will change their color with probabilities specified by the controller, i.e., probabilities $f_{R\to G}(r_i, g_i)\Delta t$ and $f_{G\to R}(r_i, g_i)\Delta t$. Recall that $P(r, g)$ is the probability that a robot has observed r red and g green pucks in the observation window $[t - \tau, t]$. Since the robots' observations are statistically independent, $P(r, g)$ is also the probability that a randomly chosen robot (of either color) has observed r *Red* and g *Green* pucks. Thus, the expected change in the number of *Red* robots during the time interval Δt can be represented as

$$N_r(t + \Delta t) - N_r(t) = -N_r(t) \sum_{r,g=0}^{\infty} f_{R\to G}(r, g)P(r, g)\Delta t + (N - N_r(t)) \sum_{r,g=0}^{\infty} f_{G\to R}(r_j, g_j)P(r, g)\Delta t \quad (9)$$

The first (second) term in Eq. 9 describes the number of *Red* (*Green*) robots changing their state to *Green* (*Red*). Furthermore, defining the fraction of *Red* robots as $n_r(t) = N_r(t)/N$, and taking the limit $\Delta t \to 0$ we arrive at

$$\frac{dn_r(t)}{dt} = -n_r(t) \sum_{r,g=0}^{\infty} f_{R\to G}(r, g)P(r, g) + (1 - n_r(t)) \sum_{r,g=0}^{\infty} f_{G\to R}(r, g)P(r, g) \quad (10)$$

Clearly, the properties of Eq. 10, such as steady state value $n_r(t \to \infty)$, are determined by the transition probabilities $f_{R\to G}$ and $f_{G\to R}$. As a designer, we want to choose these functions in such a way as to ensure the desired global behavior. In our case, this amounts to having fractions of

robots of either color that reflect the puck distribution in the steady state., e.g., $n_r^s \equiv n_r(t \to \infty) \approx R/G$. As we will see below, this condition can be met by choosing the following transition rates:

$$f_{G\to R}(r,g) = \varepsilon \frac{r}{r+g} \equiv \varepsilon\gamma(r,g) \tag{11}$$

$$f_{R\to G}(r,g) = \varepsilon \frac{g}{r+g} \equiv \varepsilon(1-\gamma(r,g)) \tag{12}$$

where ε is a constant characterizing how often robots have to make a decision whether to switch states. Eq.10 then reads

$$\frac{dn_r}{dt} = \varepsilon\overline{\gamma}(1-n_r) - \varepsilon(1-\overline{\gamma})n_r \tag{13}$$

where $\overline{\gamma}$ is give by

$$\overline{\gamma} = \sum_{r,g=0}^{\infty} P(r,g)\frac{r}{r+g} \tag{14}$$

Note that if the pucks distribution changes in time then $\overline{\gamma}$ is time–dependent, $\overline{\gamma} = \overline{\gamma}(t)$. The solution of Eq. 10 subject to the initial condition $n_r(t=0) = n_0$ is readily obtained:

$$n_r(t) = n_0 e^{-\varepsilon t} + \varepsilon \int_0^t dt' \bar{\gamma}(t-t') e^{-\varepsilon t'} \tag{15}$$

To proceed further, we need to calculate $\overline{\gamma}(t)$ (e.g., the average of γ over the Poisson distribution). Instead of going through lengthy but straightforward calculations, we present the final result which reads

$$\overline{\gamma}(t) = \frac{1}{\tau}\int_{t-\tau}^{t} dt' \mu_r(t') + e^{-\alpha\tau M_0}\left(\frac{1}{2} - \frac{1}{\tau}\int_{t-\tau}^{t} dt' \mu_r(t')\right) \tag{16}$$

where $\mu_r(t) = R(t)/M_0$ is the fraction of red pucks. Eq. 15 and 16 fully determine the evolution of the dynamics of the system.

To analyze its properties, let us first consider the case when the puck distribution does not change with time, $\mu_r(t) = \mu_0$. Then the we have

$$n_r(t) = \overline{\gamma} + (n_0 - \overline{\gamma})e^{-\varepsilon t} \tag{17}$$

$$\overline{\gamma} = \mu_0 + e^{-\alpha\tau M_0}(1/2 - \mu_0) \tag{18}$$

Hence, the probability distribution approaches its steady state value $n_r^s = \overline{\gamma}$ exponentially. Note that for large enough $\alpha\tau M_0$ the second term in the expression for $\overline{\gamma}$ can be neglected so that the steady state attains the desired value $n_r^s \approx \mu_0$. For small values of $\alpha\tau M_0$ (i.e., small density of pucks or short history window), however, the desired steady state is not reached, and in the limit of very small $\alpha\tau M_0$ it attains the value $1/2$ regardless of the actual puck distribution.

In Fig 2 we present the results of application of top–down and bottom–up algorithms described above. There are 20 red and 80 green puck scattered in an arena of size 600. Initially, all the robots are at state *Red*. Both algorithms (top–down with gradient descent and bottom–up) converge to the correct value after some transient. Starting from an initial fraction of red red pucks $n_r(t = 0) = 1$ In the top–down case one can clearly see the effect of the communication on the convergence rate. Remarkably, in the case where there is no communication, the top–down gradient–descent method produces the very similar results with the bottom–up algorithm. In fact the homologue of $c_i(t)$ in the top-down method is $n_r(t)$ in the bottom-up approach. If we discretize in the same way the differential equation defining

(a) Top-down approach (b) Bottom-up approach

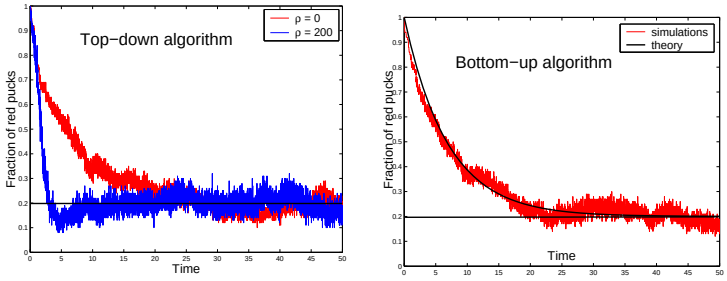

Fig. 2 Convergence to correct puck distribution for (a) top–down and (b) bottom–up approaches. Two curves for the top down approach are for $\rho = 0$ (no communication) and $\rho = 200$ in a 100×100 square grid of side length 600 (along each direction points were separated by 6 units of length).

$n_r(t)$ (13) we obtain

$$n_t(t+1) = (1 - \delta_t \epsilon) n_r(t) + \delta_t \epsilon \overline{\gamma}$$

which has the same form as (4) with $\gamma_t \sim \delta_t \epsilon$. This explains well the experimental curves. This is not surprising as both γ_t and ϵ somehow control how new observations (g_i, r_i in the top-down approach) or new estimates of averages ($\overline{\gamma}$ in the bottom-up approach) affect current estimates.

A quite different result is obtained by analysing the second top-down approach based on the LS estimation (see Fig. 3). Here we can observe that the value of ρ affects essentially the variance of the estimator. The intuition here is that as ρ increases then the number of samples that concur to the calculation of the mean also increases thus reducing the variance consistently with the well-known convergence properties of the sample mean estimator.

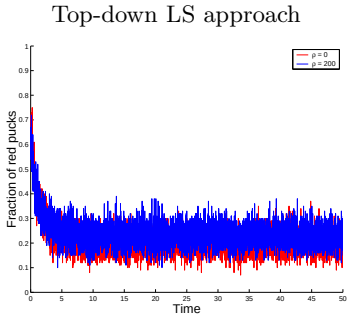

Fig. 3 Convergence to correct puck distribution for the top–down LS approach.

5 Conclusion

In conclusion, we have presented a comparative study of top-down and bottom-up design methodologies from the perspective of multi-agent system engineering. We suggest that the difference between two approaches is primarily manifested in the requirements about the availability of local resources to the rest of the system. This requirement is very important for the top-down approach, while not so imperative for the bottom-up. Note also that the statistical properties of the external environments in which agents operate affect their ability to generalize their local experience. So, under certain conditions both approaches not only become viable but may even produce the same solution. An example is represented by the Puck Allocation problem when approached using parameter estimation techniques in both the methodologies.

More generally, we have identified the following three main elements in our comparison:

Specifications: Bottom-up approach starts with the specification of the individual agent behavior through a set of agent capabilities or rules of engagement which delimit the set of obtainable group–level behaviors. The top-down approach starts with global requirements as in a centralized control system and translates those into necessary agent capabilities. Note that the last step assumes implicitly that the global system requirements can be delegated to individual components. For some tasks this might not be straightforward.

Communication and Noise: Communication is important in both the approaches but its impact is completely different if not opposite. In the top-down case a form of explicit communication is a requirement implied by the necessity for individual components to access remote resources according to the global design. In the bottom-up case, communication is optional in so far as the impact of the propagation of the information throughout the system on the emergent behavior is more like a positive side effect of the design rather than an expected feature required in the specification. As a consequence the performance of top-down systems, although optimal in ideal conditions, is expected to be very sensitive to communication noise and latency. It is then necessary to analyze the parameter space that defines how accurately individual components access or estimate remote resources. In extreme situations caused typically by intolerable levels of noise, inadequate communication range or insufficient propagation of information, the bottom-up approach seems to represent the only viable solution.

Analysis and performance guarantees: Both approaches are similar in their reliance on simulations to analyze global system properties. Whenever mathematical analysis is possible, however, the two approaches differ in their choice of mathematical tools. The top-down approach usually utilizes mathematically rigorous tools such as Classical and Stochastic Control Theory, Operations Research, Optimization Theory, Machine Learning and Parameter Estimation, whereas the bottom–up design relies heavily on mean–field methods such as statistical mechanics and the Master Equation, mean–field theory, dynamical systems theory, etc. Hence, in the top-down approach it is possible to establish stringent bounds on the system behavior and make performance guarantees within its range of applicability as it relies on well established classical methods. The analysis tools in the bottom-up approach, on the other hand, can be extremely efficient in describing the average system behavior but as a rule they do not allow for worst-case analysis.

While we do not claim this list to be complete, we strongly believe that these features will be helpful in determining the applicability of either methodology for a given multi-agent system problem and resources for its solution.

ACKNOWLEDGMENTS

This work results from research programs at the Department of Computer Science at the California State University, Los Angeles, and at the Informa-

tion Sciences Institute, USC, supported by the AFOSR Grant FA9550-07-1-0421 and by the NSF Grant Award No. 0413321.

References

1. Chris V. Jones and Maja J Matarić. From local to global behavior in intelligent self-assembly. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA'03), Taipei, Taiwan, Sep 2003*, pages 721–726, 2003.
2. M. Pizka and A. Bauer. A brief top-down and bottom-up philosophy on software evolution. In *Principles of Software Evolution, 7th International Workshop on (IWPSE'04)*, September 2004.
3. Niklaus Wirth. Program development by stepwise refinement. *Communications of the ACM*, 14:221–227, 1971.
4. G. McFarland. The benefits of bottom-up design. *ACM SIGSOFT Software Engineering Notes*, 11:43 – 51, 1986.
5. Tomas Isakowitz, Arnold Kamis, and Marios Koufaris. Reconciling top-down and bottom-up design approaches in rmm. *DATA BASE*, 29(4):58–67, 1998.
6. S. Kornienko, O. Kornienko, and P. Levi. Generation of desired emergent behavior in swarm of micro-robots. In R. Lopez de Mantaras and L. Saitta, editors, *European Conference on Artificial Intelligence (ECAI-04)*. IOS Press, 2004.
7. K. Sims. Evolving 3d morphology and behavior by competition. In R. Brooks and P. Maes, editors, *Proceedings of Artificial Life IV*, pages 28–39, 1994.
8. C.V. Jones. *A Formal Design Methodology for Coordinated Multi-Robot Systems*. PhD thesis, University of Southern California, 2005.

9. C. Harper and A.FT Winfield. A methodology for provably stable behavior-based intelligent control. *Robotics and Autonomous Systems*, 54:52–73, 2006.
10. A.F.T. Winfield, J. Sa, M.C. Fernandez-Gago, C. Dixon, and M. Fisher. On formal specification of emergent behaviors in swarm robotic systems. In *Advanced Robotic Systems*, 2005.
11. J.-M. McNew and E. Klavins. A grammatical approach to cooperative control. In *CCOGraphGrammars*, 2005.
12. M. Ott and K. Lerman. Using grammar induction to synthesize robot controllers for dynamic task allocation. In *submitted to IROS-07*, 2007.
13. K. Lerman, A. Martinoli, and A. Galstyan. A review of probabilistic macroscopic models for swarm robotic systems. In Sahin E. and Spears W., editors, *Swarm Robotics Workshop: State-of-the-art Survey*, number 3342 in LNCS, pages 143–152. Springer-Verlag, Berlin Heidelberg, 2005.
14. K. Lerman and A. Galstyan. Mathematical model of foraging in a group of robots: Effect of interference. *Autonomous Robots*, 13(2):127–141, 2002.
15. A. Martinoli, K. Easton, and W. Agassounon. Modeling of swarm robotic systems: A case study in collaborative distributed manipulation. *Int. Journal of Robotics Research*, 23(4):415–436, 2004.
16. K. Lerman, Chris V. Jones, A. Galstyan, and Maja J. Matarić. Analysis of dynamic task allocation in multi-robot systems. *International Journal of Robotics Research*, 25(3):225–242, 2006.
17. George Cybenko. Agent-Based Systems Engineering. *Darpa Task Program Research Proposal. http://actcomm.thayer.dartmouth.edu/task/*, Oct 2000.
18. Valentino Crespi and George Cybenko. Agent-based Systems Engineering and Intelligent Vehicles and Road Systems. *Darpa Task Program white paper. http://actcomm.thayer.dartmouth.edu/task/*, April 2001.

19. D. Kotz, G. Jiang, R. Gray, G. Cybenko, and R. Peterson. Performance analysis of mobile agents for filtering data streams on wireless networks.

20. Valentino Crespi, George Cybenko, Daniela Rus, and Massimo Santini. Decentralized Control for Coordinated flow of Multiagent Systems. In *Proceedings of the 2002 World Congress on Computational Intelligence. Honolulu, Hawaii*, May 2002.

21. Valentino Crespi and George Cybenko. Decentralized Algorithms for Sensor Registration. In *Proceedinds of the 2003 International Joint Conference on Neural Networks (IJCNN2003), Portland, Oregon*, July 2003.

22. C. Kube and H. Zhang. The use of perceptual cues in multi-robot box-pushing. In *IEEE International Conference on Robotics and Automation*, pages 2085–2090, Minneapolis, Minnesota, 1996.

23. R. Arkin and T. Balch. Cooperative multiagent robotic systems. In D. Kortenkamp, R. P. Bonasso, and R. Murphy, editors, *Artificial Intelligence and Mobile Robots*. MIT/AAAI Press, Cambridge, MA, 1998.

24. O. Holland and C. Melhuish. Stigmergy, self-organization, and sorting in collective robotics. *Artificial Life*, 5:173–202, 2000.

25. C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *ACM/IEEE International Conference on Mobile Computing and Networks (MobiCom 2000)*, Boston, Massachusetts, August 2000.

26. Hans Chalupsky et al. Electric elves: Applying agent technology to support human organizations. In *Proceedings of the Thirteenth Annual Conference on Innovative Applications of Artificial Intelligence (IAAI-2001), Seattle, WA*, 2001.

27. K. Lerman, A. Galstyan, A. Martinoli, and A. Ijspeert. A macroscopic analytical model of collaboration in distributed robotic systems. *Artificial Life Journal*, 7(4):375–393, 2001.

28. K. Lerman and A. Galstyan. Macroscopic Analysis of Adaptive Task Allocation in Robots. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS-2003), Las Vegas, NV*, pages 1951–1956, oct 2003.

29. Chris V. Jones and Maja J Matarić. Adaptive task allocation in large-scale multi-robot systems. In *Proc. of the IEEE International Conference on Intelligent Robots and Systems (IROS'03), Las Vegas, NV*, pages 1969–1974, Oct 2003.

30. D.P. Bertsekas and J.N. Tsitsiklis. Gradient Convergence in Gradient Methods. *SIAM J. on Optimization*, 10:627–642, 2000.

31. Yaakov Bar-Shalom, X. Rong Li, and Thiagalingam Kirubarajan. *Estimation with Application to Tracking and Navigation*. Wiley Interscience, 2001.

32. A. Martinoli, A. J. Ijspeert, and L. M. Gambardella. A probabilistic model for understanding and comparing collective aggregation mechanisms. In Dario Floreano, Jean-Daniel Nicoud, and Francesco Mondada, editors, *Proc. of the 5th European Conference on Advances in Artificial Life (ECAL-99)*, volume 1674 of *LNAI*, pages 575–584, Berlin, September 13–17 1999. Springer.

33. A. J. Ijspeert, A. Martinoli, A. Billard, and L. M. Gambardella. Collaboration through the exploitation of local interactions in autonomous collective robotics: The stick pulling experiment. *Autonomous Robots*, 11(2):149–171, 2001.