# Emergent cooperative goal-satisfaction in large scale automated-agent systems ⋆

Onn Shehory

*The Robotics Institute, Carnegie-Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, USA*

Sarit Kraus

*Dept. of Math and Computer Science, Bar Ilan University, Ramat Gan, 52900 Israel,*
*and*
*Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, USA*

Osher Yadgar

*Dept. of Math and CS, Bar Ilan University, Ramat Gan, 52900 Israel*

**Abstract**

Cooperation among autonomous agents has been discussed in the DAI community for several years. Papers about cooperation [6,45], negotiation [33], distributed planning [5], and coalition formation [28,48], have provided a variety of approaches and several algorithms and solutions to situations wherein cooperation is possible. However, the case of cooperation in large-scale multi-agent systems (MAS) has not been thoroughly examined. Therefore, in this paper we present a framework for cooperative goal-satisfaction in large-scale environments focusing on a low complexity physics-oriented approach. The multi-agent systems with which we deal are modeled by a physics-oriented model. According to the model, MAS inherit physical properties, and therefore the evolution of the computational systems is similar to the evolution of physical systems. To enable implementation of the model, we provide a detailed algorithm to be used by a single agent within the system. The model and the algorithm are appropriate for large-scale, dynamic, Distributed Problem Solver systems, in which agents try to increase the benefits of the whole system. The complexity is very low, and in some specific cases it is proved to be optimal. The analysis and assessment of the algorithm are performed via the well-known behavior and properties of the modeling physical system.

*Keywords:* Multi-agent systems, task allocation.

## 1 Introduction

Multi-agent systems (MAS) have been developed in recent years to address a variety of computational problems of a highly distributed nature. Although a large body of this research is theoretical, one may find an increasing number of simulated and implemented systems of agents (for example [55]). Usually, either simulations or implementations consist of merely dozens of agents. However, during the course of their development, many researchers have realized that, in order to provide solutions to real-world problems, MAS should scale up. Such a scale-up must allow hundreds and thousands of agents to be involved in the execution of highly distributed, dynamically changing large numbers of tasks. Several attempts have been made to allow scalability of this type, and theoretical models which have been developed may be applicable (e.g. [60]), however this is yet to be investigated.

The problems arising in large scale MAS are myriad, and due to their significance should be thoroughly studied. In this paper we cannot address all, hence we concentrate on investigating one facet of this diversity – the issue of task allocation and execution within large-scale cooperative MAS [1]. More specifically, we consider cases in which cooperative autonomous agents allocate themselves to tasks. We provide a model that allows for the dynamic agent-task allocation and is appropriate for large-scale MAS. To support our theoretical claims, we provide a simulation of a dynamic agent system that follows our suggested mechanisms and consists of thousands of agents and tasks. To our best knowledge, up to date, this is the largest simulation of a task allocation and execution in a dynamic, open MAS. The model we propose provides a solution to problems which were not addressed previously in MAS, and may be the basis for future solutions for a larger class of problem domains.

Regardless of their size, MAS are designed to satisfy goals, usually by executing tasks to which these goals may be decomposed. To allow for goal-satisfaction in dynamic systems of multiple agents and goals and, correspondingly, multiple tasks, MAS should be provided with a mechanism for task-agent allocation. Task-allocation methods in DAI (e.g., [14,51,54]) usually require coordination and communication [15,25]. In very large agent-communities, direct rapid con-

[⋆] This material is based upon work supported by NSF grant No. IRI-9423967, ARPA/Rome Labs contract F30602-93-C-0241 and the Army Research Lab contract No. DAAL0197K0135. A preliminary version of this paper appears in the proceedings of ATAL-98.

[1] Cooperative MAS are frequently referred to as a Distributed Problem Solvers (DPS) agent systems. In DPS agent systems as in cooperative MAS, agents attempt to increase the common outcome of the system. A variety of algorithms for agent-cooperation as a DPS system have been presented e.g., in [2,13,37].

nection between all of the agents is usually prohibited, as such connection may clog the communication network [2]. Negotiation processes for establishing cooperation are rather complex. Moreover, coordination-related computations which are based on on-line, bilateral communication among all of the agents may be too complex as well. Therefore, complexities of cooperation methods in MAS become unbearable when the number of agents increases.

To resolve this problem, we apply methods from classical mechanics to model large-scale MAS [50,52]. We adopt methods used by physicists to study interactions among multiple particles. The physics-oriented methods are used to construct a coordinated task-allocation algorithm for cooperative goal-satisfaction. This algorithm is to be used by the single agent within the system, and enable coordination without negotiation and with limited communication [17]. There are many differences between particles and computational systems. Nevertheless, we show that the physics-oriented approach enables low-complexity coordinated task-allocation and execution in very large MAS.

The physics methods, although they may be viewed as restricted to physical domains, allow for a model that is more expressive than other models, e.g., the tileworld model [41], as we discuss in section 8. In addition, unlike most task allocation methods for multiple agents, it consists of an inherent interleaving planning and execution [3]. And, as stated previously, while several models prove to work successfully for systems that consist of few agents (usually less than 20), e.g. [16,17], their computational complexity would prohibit scaling up. Even when the model is based on market equilibrium, as in [39], simulations are limited to less than 20 agents. In contrast, we present a theoretical justification to the ability of our model to scale up. Moreover, we further support these claims by simulations results that consist of thousands of agents and tasks. These results show no increase in computation and communication per task and per agent when the size of the system grows.

Note that since we use a physics metaphor, our model can more easily be applied to problems of physical nature (e.g., transportation, as we demonstrate later). We believe that applying our model to other problem domains is possible as well. We illustrate this possibility by example (section 5.3), however do not prove it in this paper. One may be concerned that using classical mechanics requires continuity of progression of the agents towards goal-satisfaction. This may be simpler to model, however continuity is not a requirement. Ap-

[2] For instance, assume that for each task each agent communicates with all other agents. If $n$ is the number of agents, even $O(n)$ communication operations per agent, which total to $O(n^2)$ in the network are most likely overwhelming when the system consists of thousands of agents.

[3] Note that the weak commitment algorithm [62] does allow for interleaving planning and execution, but requires that inconsistent plans will be abandoned, then starting from scratch.

propriate modeling of the goals in continuous or semi-continuous terms will suffice. We have performed such modeling in the simulations presented later in this paper. Cases where continuity modelling is inadequate will be discussed in future work. A simple example where our model can be applied is a system in which agents have to block holes of various sizes in a planar surface (which has similarities to the tileworld [41]). Each hole to be blocked is a goal, and the filling for blocking holes is the agents' resource. The purpose of the system is to block as much hole-area as possible. Some holes cannot be blocked by a single agent and thus cooperation is necessary [4].

### *1.1 Definitions, assumptions and notations*

We describe the systems with which we deal as a set of agents $\mathtt{N}$ and a set of goals $\mathtt{G}$, both possibly dynamically changing, located in a goal-space $\mathcal{G}$. An $m$-dimensional displacement vector is a vector $D = < d_1, \cdots, d_m >$. The distance between $D^1, D^2$ is defined by $r^{1,2} = \sqrt{\sum_i (d_i^2 - d_i^1)^2}$. Each agent $A \in \mathtt{N}$ and a each goal $g \in \mathtt{G}$ have a displacement vector $D$ which is their location in the $m$-dimensional goal-space $\mathcal{G}$. Since in some domains goals do not have physical properties, the components of a displacement vector $D$ are not necessarily physical distances [5]. We refer to such distances as virtual and denote as virtual the goal-space. We assume that the agents with which we deal have the ability to perceive the virtual displacement in the goal-space, and can perceive the properties of other adjacent agents and goals. This may be done by sensors [6] integrated into the agents. We also assume that each agent knows about the types of resources that other agents may have, but may be uncertain of the particular resource-holdings of any other individual. These two assumptions are necessary since agents who progress within the goal-space need some information regarding properties of other agents and goals. We assume that each agent has a performance capability that can be measured by standard measurement units, which enable quantification of the agents' task execution. In addition, we assume that there is a scaling method which is used to represent the displacement of the agents in the goal-space and to evaluate the mutual distances between goals and agents within this space. This assumption is necessary since virtual distances (or physical distances) are a significant factor in the model we present. We assume that goal-satisfaction can be achieved progressively. That is, a goal may be partially satisfied at one instant, and its remaining non-satisfied part may be completed at another point in time.

[4] More detailed examples will be presented in section 5.

[5] For instance, one can view the number of incorrect letters in a misspelled word as its distance from its correct form.

[6] The interpretation of sensors is extended in this paper to any information reception device.

To present our model, we review several concepts and notations from physics [7]. We start by listing general mathematical notations which we use. A vector is a physical property which has both a direction and a magnitude. Any vector $x$ will be denoted by $\vec{x}$. Physical analysis consists of derivation of functions with respect to time. The first order time-derivative of $x$ is denoted by $\dot{x}$ and the second order time-derivative is denoted by $\ddot{x}$. The gradient operation is denoted by $\vec{\nabla}$. This operator is a vector-derivative and (in Cartesian coordinates) is given by

$$\vec{\nabla} = \left( \frac{\partial}{\partial x}\hat{x}, \frac{\partial}{\partial y}\hat{y}, \frac{\partial}{\partial z}\hat{z} \right) \tag{1}$$

where $\hat{j}$ denotes a unit vector in the direction of coordinate $j$. We continue by listing physical concepts. The displacement of a particle $i$ is denoted by $r_i$. Usually it is referred to as $\vec{r_i}$, the vector of displacement, which is the $(x_i, y_i, z_i)$ coordinates of the particle. $v_i$ denotes the velocity, which is the rate of change of displacement, and $a_i$ denotes the acceleration, which is the rate of change of velocity. The kinetic energy of a particle $i$ is represented by $k_i$, and the potential is represented by $V$. The potential is a spatial function and therefore is sometimes called a field of potential or a potential-well (the latter refers to a specific shape of a potential function). Forces can be derived from the potential. Each particle $i$'s mass is denoted by $m_i$, its displacement is denoted by the displacement vector $\vec{r_i}$, its momentum is denoted by $\vec{p_i}$ and the force that acts on it is denoted by $\vec{F_i}$.

Classical mechanics provides a formal method for calculating the evolution of the displacement and the momentum of particles. Given the initial displacement $\vec{r_i}(0)$ and momentum $\vec{p_i}(0)$ of a particle $i$, its displacement $\vec{r_i}(t)$ and momentum $\vec{p_i}(t)$, at any other time $t$, can be derived via the solution of the equations of motion. These equations are first and second order differential equations. The boundary conditions (that is, the arbitrary constants) of the exact solutions of the equations are the initial displacement and momentum of the particle. For a particle $i$, the equations of motion are:

$$\vec{F_i} = m_i\ddot{\vec{r_i}} = m_i\vec{a_i} \tag{2}$$

$$\vec{p_i} = m_i\dot{\vec{r_i}} = m_i\vec{v_i} \tag{3}$$

[7] The notations and concepts we present here are described in many introductory physics books, e.g., [34].

The nature of the motion of a particle depends on the field of potential in which it moves. This dependency is given by:

$$\vec{F}_i = -m_i \vec{\nabla}_{r_i} V(\vec{r}) \tag{4}$$

For some types of potential $V$, the solutions of the equations, either exact or approximated, are well known or can easily be derived. In our model we employ only such types of potential functions. By relying on the known solutions from physics we may predict the behavior of the agents who follow our model. Simulations (see section 7) are performed to further support the validity of the model.

### *1.3 Adapting physics to DAI*

| DAI | Physics |
|---|---|
| identifying the environments where physics-oriented models are appropriate; matching between particle properties and agents/goals properties | locating particle models and their properties |
| selecting the states of matter that can be used for modeling automated agent-systems | identifying states of matter and the particle properties within |
| developing goal-satisfaction algorithms; adjusting the agent system to the physics system for the validity of the algorithm | using mathematical formulation to predict and describe the properties and evolution of the selected state of matter |
| analysis of the complexity and properties of the algorithms | theoretical and simulation-based analysis of physical particle systems' behavior |

Fig. 1. Steps in applying a physics-oriented model to a Distributed AI, DPS problem domain.

As stated previously, we consider large sets of agents and goals. Each agent has a goal-satisfaction capability and should advance toward satisfying goals. We use a physics-oriented model that consists of particles to represent agents and goals and to develop a distributed cooperative goal-satisfaction mechanism.

| Automated Agents | Physical Model |
|---|---|
| community of agents satisfying goals | non-ionic liquid system |
| agent | dynamic particle |
| goal | static particle |
| agent's capability | particle's mass |
| agent's location in agent-goal space | location of particle |
| goal satisfaction | static-dynamic collision |
| algorithm for goals allocation | formal method for calculating the evolution of displacement |

Fig. 2. The match between the physics model components and the large-scale automated agents' environments.

The first step in applying the physics model to DAI is the match between particles and their properties, agents and their capabilities, and goals and their properties (see Figure 1). The next step is to identify the most appropriate state of matter for modeling an ensemble of agents and goals. The mathematical formulation that is used by physicists, either to describe or to predict the properties and evolution of particles in these states of matter, will serve as the basis for the development of algorithms for the agents' behavior. However, several modifications of the physics-oriented model are necessary to provide an appropriate algorithm for automated agents. In the rest of this paper we shall elaborate on these issues.

The general idea of our model is that entities of the MAS are modeled by particles. Agents and goals are modeled by dynamic particles and static particles, respectively. The properties of a particle $i$, i.e., its mass $m_i$, its displacement and momentum vectors $\vec{r_i}$ and $\vec{p_i}$, its potential $V_i$ and its kinetic energy $k_i$ are abstractions of the properties of the modeled agents and goals as described in Figure 2. The agent's goal-satisfaction capability is represented by the mass (and the potential energy) of its modeling particle. The mass of a static particle represents the size of the goal it models. The displacement of a particle in the physical space models the displacement of the agent in the goal-space.

We model goal-satisfaction by a collision of dynamic particles (that model agents) with static particles (that model goals). However, the properties of particle collisions are different from the properties of goal-satisfaction and several adjustments are needed in order to provide the agents with efficient algorithms. These modifications are described in detail in this paper.

## 2 The physics-agent-system (PAS) model

The model we present entails treating agents, goals and obstacles as if they were particles. That is, each agent will have an initial state and its equations of motion. Note that an agent's equations of motion do not necessarily entail real physical motion; they may represent the progress towards the fulfillment of goals. The potential field in which an agent acts represents the goals, the obstacles and the other agents in the environment. Subject to the potential field, an agent will solve the equations of motion and, according to the results, progress towards the solution of goals and either cooperate or avoid conflicts with other agents. Note that cooperation and conflict-avoidance are emergent properties of our physics-oriented model.

### *2.1 State of matter for PAS*

In order to construct an appropriate potential field to represent the multi-agent environment, we shall examine the properties of physical states and locate the most appropriate one. An appropriate physical state must consist of a potential that, when adapted to the agent-model, will lead the agents to beneficial goal allocation and satisfaction. In the solid state particles are localized, i.e., they are bound strongly to their initial position; this prevents evolution of the system. Thus, when applied to MAS, dynamic goal satisfaction by the agents is prevented. In the gas state, interactions between particles are very weak. In such a case, the system can evolve, but the lack of intensive interaction means that cooperation and conflicts will rarely occur. This may be an interesting issue for future research, but presently we are interested in cases where there are both cooperation and conflict among the agents. The liquid state lies between these two states. As opposed to the solid state, a liquid evolves more rapidly. However, unlike the gas state, a liquid is dense enough to cause interaction among its particles [8]. Therefore, the liquid model is preferred as a more appropriate model for the MAS under consideration.

Among the liquids, there are two main types: ionic and non-ionic liquids. Ionic liquids are such that the mutual potential among each pair of particles is a Culombic potential [9]. This potential is proportional to $1/r$, where $r$ is the distance between the particles. Such a potential diminishes slowly with respect to $r$, and therefore entails a long-range interaction among the particles. This type of interaction means that all of the particles in the system should be considered when calculating the interactions and the evolution of the state of

[8] Note that interactions occur among gas particles too, however the rate of interactions is extremely smaller than this rate in liquids.

[9] Culombic potential is the potential that results from electric charges.

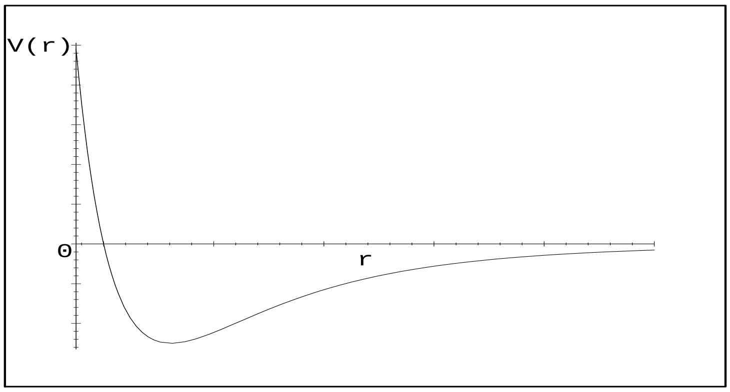



Fig. 3. A typical one-dimensional Lennard-Jones potential-well

each single particle in the system. Typically used to describe the potential of a particle $i$ in a non-ionic liquid, corresponding to its distance $r_{ij}$ from particle $j$ is the Lennard-Jones potential:

$$V(\vec{r})_{LJ_{ij}} = 4\varepsilon \left( \frac{1}{r_{ij}^{12}} - \frac{1}{r_{ij}^{6}} \right) \tag{5}$$

where $\varepsilon$ is a mass-dependent coefficient that scales the potential. Because of the shape of the curve of function 5 (see figure 3), the potential is sometimes called a "potential-well". Such a potential function entails a potential that vanishes after a short distance $r$, due to the high powers of $r$ that are present. The short distance here implies that the interactions between the particles in the system are limited to short distances. That is, a particle interacts only with particles in a limited neighbourhood, and only these are considered for the calculations of the evolution in the state of each particle. The properties of the non-ionic liquid, and in particular the short-range potential, make it appropriate for use as a model for the large-scale MAS with which we deal. In such systems, communication with all of the agents [10] and computations of all possible interactions may be too complex. Note that this implies that the information accessible to an agent regarding its environment is incomplete.

### *2.2 Matching physical properties to agents and goals*

Physical particles may have different masses. As a result of the different masses, particles subject to the same field of potential will have different momentum and kinetic energy. This is because the momentum is expressed by $\vec{p} = m\vec{v}$ and the kinetic energy is expressed by $k = mv^2/2$, and both are products of the mass. In the PAS model, the agent's capability to satisfy

[10] The communication required for each agent is $O(\#agents \times \#tasks)$, which is not considered large. However, in large systems this will most probably be overwhelming (e.g., 1000 agents and 1000 tasks, as we have in our simulation).

goals is represented by the potential energy of the particle that models it. Particles with greater potential energy model agents that can satisfy larger or more difficult goals and sub-goals. This means that a greater mass of a dynamic particle (that models an agent), other properties remaining constant (and thus causing a greater potential energy), entails a larger capability of goal-satisfaction by the agent. The mass of a fixed particle (which models a goal or an obstacle) represents the size of the goal or the obstacle. This means that in order to satisfy a greater goal, which is modeled by a static particle with a greater mass, additional efforts are necessary on the part of the agents.

The displacement vector of a particle $\vec{r_i}$ models the displacement of the agent in the goal-space. This space can be either physical or abstract, since goals are not necessarily (and are usually not) physical.

**Example 2.1** *An example of an abstract goal-space is the space of queries in a multi-database domain. In such a domain, each database is represented by an agent, and the goals that the agents must fulfill are the answering of queries that were directed to their databases. The displacement of an agent in the query-space represents the logical proximity of the information stored in its database, either to the information stored in other databases or to the information necessary for answering a query. Virtual distances in the multi-database domain can be calculated if a pre-defined logical-proximity of a key-words scale is given*[11] *. Note that modeling such a space may be rather difficult and require many adjustments.*

According to the virtual displacement of an agent, one can calculate its distances from other agents, goals and obstacles. These distances are then used to calculate the potential. The momentum vector of particle $i$, $\vec{p_i}$, represents its physical velocity and is used for the calculation of the kinetic energy. In the PAS model, the velocity of a dynamic particle (which models an agent) represents the rate of movement towards the satisfaction of a goal or part of a goal.

**Example 2.2** *In the example in section 3, a system wherein agents have to block holes in a planar surface is presented. The purpose of the system is to block as much hole-area as possible. Some holes cannot be blocked by a single agent. According to PAS, each hole in this system is modeled by a particle, which is represented by a potential-well, and the size of the hole is represented by its mass. A greater mass entails a greater hole, and due to the physical nature of the potential-well, particles surrounding a well with a greater mass will experience stronger attraction to the well. This property of the wells is appropriate for our purposes, because it causes a natural attraction to the holes which is proportional to the size of the holes. The agents in this system are also*

[11] Such proximity scales can be found in various information retrieval systems, e.g., in [46].

*modeled by particles, represented by potential-wells. However, as opposed to the holes which are fixed in their displacements, the agents' potential-wells are free to move. The agents have masses of various magnitudes which represent differences in their abilities to block holes.*

### *2.3 Virtual motion towards goal-satisfaction*

In the physical world, the motion of particles is caused by the mutual attraction (and rejection) between them. In the agents' system, the agents calculate the attraction and move according to the results of these calculations. Since, in the physical world, motion is continuous, the agents' calculations, which result from our PAS model, must resemble this continuity. This can be done by performing the calculations repeatedly with a high frequency.

According to the model, the agents shall calculate the potential, subject to the surrounding goals and agents. This potential is affected by the virtual distance from these neighboring entities. Due to the large size of the systems under consideration, each modeling particle has only a limited effective interaction with the surrounding particles. Consequently, the modeled agents have limited information about the goal-domain. A particle can only have local reactions to the potential field and, in practice, only its near neighbors will affect its potential. The range of interaction among modeling particles has a significant effect on the complexity of the calculations that the modeled agents perform [12]. We denote the radius of interaction among the particles by $r_I$. This radius may sometimes be determined by sensors integrated into the agents, or shall be determined by the designers of the agents. Assuming a random distribution of the particles within the range of the whole system (however not necessarily uniform), the computational complexity that the modeled agents experience grows linearly with respect to the size of the area included within the range of interaction. That is, it increases linearly with respect to $r_I^2$, which is disadvantageous. The increment in the complexity, when $r_I$ is increased results from the corresponding increment in the number of agents and goals within the interaction range. The advantage of the increment of $r_I$ is that within the range of $r_I$, there will be more interacting entities, which may increase the goal-satisfaction and its efficiency. However, due to the sharp reduction in the magnitude of the potential function beyond a short distance from the center of the potential-well, the magnitude of the derived forces is small and the interaction is negligible and diminishes. Simulations that were performed by physicists have shown that when these long-distance interactions are neglected, the results of the simulations still agree with theoretical statistical-mechanics and thermodynamics [42,57]. Therefore, it is common to cut off the range of

[12] The complexity is analyzed in detail in section 4.

interaction by cutting off the potential function after it diminishes to 1% to 10% of its maximal value. Our model follows this cut-off strategy. To illustrate this cutoff within the physical system, we bring forth the following example:

**Example 2.3** *Suppose that the potential function of a particle is formulated by $V = 4\varepsilon(1/r^6 - 1/r^{12})$ (a Lennard-Jones potential). The maximal absolute value of this function can be calculated by setting its derivative to zero. The result of this is the distance wherein the potential function is maximal, that is, $r_{max} = 2^{1/6} = 1.112$. Substituting $r_{max}$ into the potential function, we derive the maximal absolute value of the potential, which is $V = \varepsilon$. Simple calculations show that the reduction of $V$ to 10% of the maximal value will cause a cut-off distance $r_{cutoff} = 1.842$ and reduction to 1% will cause $r_{cutoff} = 2.714$. Both cut-off distances are not significantly far from the particle under investigation. Such cut-off distances are adopted for our MAS.*

The reaction of a particle to the field of potential will yield a change in its coordinates and energies. The change in the state of the particle is a result of the influence of the potential (since $\vec{F_i} = -m\vec{\nabla}V$, i.e., the acting force, is derived from the potential). In our model, each agent will calculate the effect of the potential field upon itself by solving a set of differential equations. According to the results of these calculations, it will move to a new state in the goal-domain, as we describe in detail in the formal protocol in section 2.7. Real motion is not necessary here. In a case of an abstract space, moving to a new state means updating the state parameters. For instant, if a state is represented by a vector of boolean values (as we illustrate in section 5.3), such a state change means a change in some of the boolean values.

The equations that an agent must solve during its virtual motion towards goals are the equations of motion of a particle subject to a potential field. Solving these equations may be complicated, but an approximation by numerical integration (e.g., leap-frog [4]) and Verlet tables (as in [22]) can simplify these calculations while providing results that are, for practical purposes, of the same quality as the accurate results. This numerical integration, which is done with respect to time, must be iterated frequently (as explained previously in this section) and performed with small time-steps $dt$ which will be used as the differential for the numerical integration.

## *2.4 Setting the size of dt*

The size of the time differential $dt$ depends on the properties of the system with which we deal. This dependency is due to the effect that $dt$ has on the number of iterations that must be performed until an agent reaches a goal. A small $dt$ results in a more accurate numerical integration, but the change in the displacement of the modeling particle (and hence of the agent that it models) at each iteration will be very small. This implies that a reduction in the size of $dt$ increases the number of iterations necessary for an agent to reach a goal. This increases the overall time of the goal-satisfaction procedure. A large $dt$ reduces the accuracy of the numerical integration, but reduces the number of iterations necessary for reaching a goal as well. However, large $dt$ has some deficiencies: as $dt$ grows larger, the progress towards a goal at each time-step becomes greater. This leads to situations where a single time-step $dt$ may lead to a large single displacement-translation. Such a translation may move an agent far from the goal towards which it was moving. Moreover, such a behavior contradicts the continuous physical properties that are necessary for the success of our model.

From the deficiencies of either large $dt$ or small $dt$ we conclude that $dt$ should lie somewhere in between. The time-step $dt$ shall be chosen subject to the properties of a specific system. We will use $r_0$ as our unit of measure. Relying on the experience gathered in physics simulations [42], a typical particle in the model will pass a distance of $r_0$ in about ten time-steps $dt$. This requirement implies that the average velocity $\overline{v}$ of a particle (at its initial displacement) directly affects $dt$ by the relation $dt = r_0/\overline{v}$. Therefore, the initial average distance between agents and goals will enable the preliminary setting of $dt$, as we prove later by simulation results.

In the physics model, $dt$ serves mainly as the differential for the numerical integration and represents real movement-time of the particles. In the PAS model, $dt$ serves as a time unit where, in each time step $dt$, agents calculate their parameters and progress according to these calculations. $dt$ in the goal-agent system is different from $dt$ of the corresponding particle system, but both are related to one another by a 1-1 onto function, which is determined by the properties of the corresponding models. To determine $dt$ in the goal-agent system, the computation time and the goal-satisfaction time should be considered. $dt$, which is originally calculated according to the physical properties of the system, will be $dt \geq max(\textit{computation-time}, \textit{movement-time})$ in cases where agents can satisfy goals and perform calculations in parallel, and $dt \geq \textit{computation-time} + \textit{movement-time}$ in cases where agents perform goals and calculations separately.

## *2.5 Collision and goal-satisfaction*

The dynamics of the physical system which models the computational system leads to collisions between particles. In a system that consists of both static and dynamic particles, two types of collisions are possible. One type is a collision between two dynamic particles, which we denote by DDC. The other type of collision is between dynamic and static particles, and we denote it by SDC.

### 2.5.1 *Dynamic-Dynamic Collision (DDC)*

In our model, the DDC represents the interaction between two agents[13]. We would like the agent-agent interaction to prevent situations in which agents collide, either in the physical sense of the word (when relevant) or in its abstract sense[14]. This can be achieved by a mutual repulsion among the particles that model the agents. As a result of the repulsion, agents do not have to negotiate over goals, since the decision on which agents shall perform a specific goal will emerge from the repulsion[15]. This repulsion model is especially necessary in the case of two modeled agents that have a large goal-satisfaction capability. The reason for this necessity is that, in cases where two such agents or more reach a goal, we would prefer the goal-satisfaction to be performed by only one of them. In such cases, one agent can usually perform the goal by itself, and goal-satisfaction by more than one agent will be a waste of efforts. However, a DDC between particles that model agents with a small[16] goal-satisfaction capability shall cause a negligibly small repulsion. This is because we would like such agents to cooperate on the goal-satisfaction.

To satisfy the repulsion requirements, dynamic particles that model agents shall have a potential that consists of a dominant repulsive component. The potential, including its repulsive component, must be proportional to the mass of the particle, since the mass of the particle models the goal-satisfaction capability of the agent. Thus, a greater mass of the modeling particle models a greater goal-satisfaction capability of the modeled agent and, as required, it also leads to a stronger repulsion. In order to satisfy the requirement of strong repulsion, the Lennard-Jones potential of a dynamic particle that models an agent shall be modified from the classical LJ potential[17]. Such a modification

[13] This does not prohibit using a similar model where agent-task interactions are modeled by dynamic-dynamic collisions as well. We do not pursue this direction here.

[14] Agents may collide when they attempt to consume the same resources or to perform the same goal, thus interfering with one another and possibly reducing effectiveness or even prohibiting task execution. For instance, if the agents are robots, then a collision may damage the robots, and therefore should be prevented. In more abstract cases overlapping locations of agents in the model may be allowed.

[15] Note that this takes into account cases where agents do not perform goals equally well, as also shown in section 7.

[16] When we speak of small and large goal-satisfaction capabilities, we do so with respect to the size of the goals towards which the agents are moving.

[17] Note that in molecular dynamics (MD) research, even more significant modifications are applied (e.g., completely omitting parts of the potential function), yet

will be included in our model and will be done by multiplying the repulsive component of the potential function by a pre-defined factor PDF. As a result of this multiplication, the magnitude of the repulsive component will increase as required. For practical use, we should set the magnitude of the PDF so that it will change the repulsive component by an order of magnitude with respect to the attractive component of the potential. The modification of the LJ potential affects the interaction between dynamic and static particles. In order to maintain the magnitude of attraction in this case, the attractive component of the LJ potential of a static particle must be multiplied by the same PDF.

### *2.5.2 Static-Dynamic Collision (SDC)*

The SDC represents interaction between an agent and a goal. In such interactions we would like the static particle that models the goal to attract the dynamic particle that models the agent, in order to lead to goal-satisfaction. The physical motion of dynamic particles towards static particles is continuous, and the attraction into the potential-well of a static particle is a gradual yet continuous process. Therefore, there is no specific point from which the particle starts the collision. However, our model requires such a point to let the agents decide upon the appropriate actions when reaching a goal, in order to satisfy it. Hence, we must decide artificially upon such a point. Adopting physical concepts, we may use the notion of *typical radius* for this purpose. A typical radius of a particle is usually (especially in MD) taken to be the distance from its center to the point wherein the force derived from the potential is equal to zero. That is, $\vec{F} = -m\vec{\nabla}V = 0$. We denote this typical radius by $\sigma$, and a simple calculation – via the derivation of the potential function – yields $\sigma = 4\varepsilon\sqrt[6]{2}$. An SDC occurs when a dynamic particle is in the vicinity of a static particle. Vicinity here means that the distance between them is a few typical radii. Therefore, we arbitrarily decide upon a distance of $3\sigma$ as the point from which the collision starts [18] and denote it by $r_0$. In two and three dimensions, this point becomes a circle and a sphere of radius $r_0$, respectively. In abstract and multi-dimensional spaces, the interpretation of $r_0$ is of logical distance. For instance, in the case of information which is classified by keywords, having reached $r_0$ means that a database agent is, keyword-wise, very close to the information necessary for answering a query, where multi-dimensionality may refer to multiple topics which are relevant to this query.

**Example 2.4** *In the hole-blocking system, the holes are physical entities with a definite size and with accurate boundaries. The model of such a hole must*

---

such modifications do not prohibit a good approximation to the well-known physical behavior. As in MD, the simulations that we have performed support the validity of our model.

[18] This is a common choice in MD as well.

*consist of a specific point from which the modeled hole begins. We use $r_0$ (as described above) to model the hole boundary. In the hole system, the interaction between a hole and an agent starts when the agent physically arrives at the hole. In our model, this will be modeled by the arrival of a dynamic particle that models the agent to a distance $r_0$ from the center of the static particle that models the hole.*

When a dynamic particle reaches a static particle, i.e., it reaches the distance of $r_0$ from its center, a collision occurs. Such a collision can have several results, the two extremes of which are the cases of completely-elastic and completely-inelastic collisions. The first type of collision entails the conservation of all of the kinetic energy of the moving particle, and in our model will represent cases where the resources of the agent are non-expendable. The second type entails the loss of all of the kinetic energy of the dynamic particle, and correspondingly, will model the cases of expendable resources. Between these two extremes, a variety of combinations may be found and can be adjusted to various cases in the agents' system. We shall discuss only the two extremes, since their combination is a simple (but time consuming) process.

### *2.5.3 The behavior of agents during an SDC*

An important issue in our model is the behavior of the particles during the collision and the corresponding behavior of the agents. An agent that reaches a goal may either completely or partially satisfy it. In both cases, the model requires a reduction in the magnitude of the goal. Since the size of the goal is modeled by the mass of its modeling particle, the mass of this particle shall be reduced. It may also require a reduction in the mass of the particle that models the agent in the case of depleting resources[19]. However, the reduction of mass is not a physical property of such a collision. Therefore, introducing such a reduction into the PAS model may deteriorate its implicit advantages. That is, the expected physical evolution of the system may lose its validity. In order to avoid this loss, some modification of the model shall be done. We will allow some non-physical parts into the model, as long as they do not affect the general evolution of the system. This is possible if the model will consist of a scheme for a temporal partition of the evolution of the system. This means that the evolution of the system will be partitioned into several time segments, and in each temporal segment the physical evolution of the system will not depend on the other segments[20]. After the partitioning into

[19] Such reductions are required to conserve correct relations between agents and goals in the system. For instance, an agent that have used most of its resources should not be attracted to a goal as strongly as a fully replenished agent.

[20] Note that the time segment here is different from $dt$ that is used for calculating the change in the coordinates of a particle. A typical time segment will be much longer than $dt$.

time segments, each time segment can be treated as a separate system. The connection between the time-segmented systems will be established via their initial and final states. The initial states of the particles in a new system are the modified final states of the previous system. A detailed method for performing the mass-reduction is presented in section 3.

#### *2.5.4 Time consumption and hindrances*

The nature of the collision in the physics model has several implications for the modeled system. Since the modeled goal-satisfaction process requires time, the modeling collision must also be time-consuming. Fortunately, physical collisions are not instantaneous. Therefore, our model should only adjust the physical collision time to the goal-satisfaction time. However, we would like this adjustment to be an implicit property of the physical behavior. This is possible if the potential-well which models the goal will cause some kind of hindrance to the particle that models the agent, when the particle has entered the region of collision $r_0$. Such a hindrance will emerge if the potential well consists of a central repulsive part. The central repulsive part of the potential-well will cause a gradual relaxation of the particle that will have reached the well, until the particle stops. Time is required for the relaxation process, and this time will model the goal-satisfaction time.

Another issue concerning the SDC is the relation between the mass of the particles and both the goal-size and the agent's ability to satisfy goals. As previously stated, a greater mass models a greater goal in the case of a fixed potential-well and a greater goal-satisfaction capability in the case of a dynamic particle. The kinetic energy of a dynamic particle grows as it gets closer to the potential-well by which it is attracted. Consequently, its velocity grows, and its final magnitude depends on the size of the attracting potential-well and its initial distance from this well. As a result of this dependency, two dynamic particles with the same masses may reach a distance equal to $r_0$ from the center of the same potential-well with different velocities due to their different initial distances from the well. Therefore, these two particles will have different collision time-periods. If the collision time models the goal-satisfaction time, this physical property should be disadvantageous for our model. This is because it means that among two modeled agents with the same goal-satisfaction abilities, the one that was initially "farther away" will perform the goal faster. For clarity of representation, we assume that agents with the same abilities perform a goal in the same amount of time. We thus assume goal-satisfaction time to be longer than collision time. Nevertheless this assumption can be relaxed via a small (though not obvious) modification to the algorithm. This modification is performed by causing another type of hindrance; whenever a dynamic particle reaches a static particle, it must complete the collision within a time-period that is equal to the time that it would have taken for a dynamic particle that started moving towards the static particle from a distance of $r_0$ (we assume that none of the dynamic particles is initially inside the range $r < r_0$) and has the same mass as the colliding particles have. The legitimacy of such a modification to the nature of the physical system can be explained similar to that in the case of mass reduction. Note that this option is not part of the algorithm we present in section 2.7 (however may be added to it). We avoid this addition to keep the algoritm simple.

### *2.6 The potential-wells*

As presented above, we model agents and goals using particles. However, it is common among physicists to represent particles by the graphic shape of their potential function. As can be seen in figure 3, these potential functions have some minimum point, and therefore are called potential-wells. The reason for using the potential-well notion in addition to the particle notion is because of the graphic illustration. This illustration may give a better idea about the attraction and rejection that the potential functions of particles impose. The potential-wells consist of domains of attraction and rejection between particles. The attraction and the rejection are formally represented by the force, which is derived from the potential function. A positive force represents attraction and a negative force represents rejection. The lowest point on the graph is the point where the force nullifies. The force grows negative when moving towards the center of the potential-well (i.e., the central part causes rejection) and positive when moving away from the center. However, as the potential curve becomes asymptotic to a horizontal line, the magnitude of the attractive force diminishes and finally vanishes. In order to illustrate the rejection and attraction within a potential well, the reader may think of a marble that is put in a physical potential-well. It is clear what parts of the well will move the marble towards the center and what parts will reject from the center. It is also clear where, along the curve, the forces will be strong and where they will be weak.

In the agents' system, attraction in the case of DDC models cases in which cooperation among the agents is beneficial, and rejection in these cases models occasions where cooperation is non-beneficial. It is beneficial for two (or more) agents to jointly perform a task when each does not have all of the required capabilities. In such cases the attraction should (and according to our model would) be the more dominant component of the potential function. Cooperation would not be beneficial in other cases, hence rejection is applied. Note that this distinction between beneficial and non-beneficial cooperation does not require different types of potential functions, however different scaling of the terms of these functions may be required. In cases of SDC, attraction models the goal-reaching and goal-satisfaction processes, and rejection models the time consumption during the goal satisfaction, as described in section 2.5.4.

Looking once again at figure 3, we can observe that typical potential-wells (the figure represents such potential-wells) have a limited range of strong attraction and rejection. The other parts of the potential may be attractive but the attraction is rather small. This attraction models regions wherein goal-satisfaction is non-beneficial or brings about very small benefits. The attractive part of the potential is a long-range potential. As can be observed, the magnitude of this potential diminishes after a relatively short distance from the origin. Practically, as simulations that were performed by physicists have proven [32,57], the long-range interaction can be cut off after a reasonable distance (as we discussed in section 2.5.2). Such a cutoff will have only a minor effect on the dynamics of the system of particles.

We describe the likelihood of cooperation by potential functions of types that would fit the properties of the agents. If the benefits of cooperation are functions of more than one variable, then the potential-wells will be multi-dimensional in the space of the variables (usually these variables are the resources of the agents). For simplicity of representation and calculation, we use one-dimensional continuous functions to represent potential-wells. This will limit us to the case of agents that use only one resource [21]. However, there are methods for the expansion of the one-dimensional case to the multi-dimensional case in physics, and these methods can be adopted when our approach is used to analyze systems of computational agents. These methods are appropriate only when the resources are independent. This is because the physics-based methods for the analysis of multi-dimensional functions require such independence. Having $n$ resources $\mathcal{R}^n = \{R_1, \ldots, R_n\}$, we require that $\forall i \in 1 \ldots n$, $R_i$ is not a linear combination of $S \subseteq \mathcal{R} \setminus R_i$, $S \neq \emptyset$.

### *2.7 A protocol for the single agent*

Having described the physical properties of the modeling particles, we may proceed to the protocol according to which the modeled agents shall act. As we have previously proposed, each agent and each goal are modeled by a potential-well. Goals are modeled by wells which have a fixed displacement and agents are modeled by dynamic wells. In order to cause evolution of the system towards goal-satisfaction, each agent uses the information that it can gather by observation (e.g., via sensors) about its neighboring agents and goals and regarding its previous state. According to this information, the agent will construct the local field of potential and solve the equations of motion. The results of the equations of motion will enable the agent to decide what its next step towards goal-satisfaction will be. The exact detailed algorithm for

[21] We discuss the issue of multiple resources and capabilities in sections 6.1, 5.3.

the single agent $i$ is as follows:
Loop and perform the goal-reaching and goal-satisfaction processes until one of the following conditions is satisfied:

- The resources necessary for the completion of the goal-satisfaction have been depleted, or,
- No more goals within the interaction range $r_I$ have been observed for several time-segments [22].

### The goal-reaching process

(i) Advance the time counter $t$ by $dt$.
(ii) Locate all of the agents and goals within the range $r_I$, the predefined interaction distance. Denote the distance to any neighbouring entity $j$ by $r_{ij}$.
(iii) Calculate the mutual Lennard-Jones potential (as described in equation 5) with respect to each of the agents and goals within the range.
(iv) Sum over all of the pairwise potentials $V_{LJ}(r_{ij})$ and calculate the gradient of the sum to derive the force $F_i$ as described in section 7, in equation 4.
(v) Using $F_i$ and the previous state $\vec{r_i}(t-dt), \vec{p_i}(t-dt)$, solve the equations of motion as described in section 7, in equations 2 and 3.
(vi) The results of the equations of motion will be a new pair $\vec{r_i}(t), \vec{p_i}(t)$. Move [23] to the new state that corresponds to the displacement $\vec{r_i}(t)$.
(vii) At each time-step, after moving to a new state, calculate the new kinetic energy $K$ (see section 2.2) and the new potential (see equation 5) according to the new coordinates $\vec{r_i}(t), \vec{p_i}(t)$.
(viii) If, due to the shift to the new displacement, your distance from the center of a particle that models a goal is greater than $r_0$, return to step 1. Otherwise, you have entered the region of strong interaction, i.e., you have reached the goal. Therefore, start the goal-satisfaction process.

### The goal-satisfaction process

After reaching a goal, the agent must satisfy all or at least parts of it. In order to do so according to our model, the following algorithm should be used by the agent [24].

[22] This requirement allows for a high probability of all goals being satisfied, however does not guarantee 100% performance. Nevertheless, greater numbers of agents and tasks in the system bring this probability very close to 1.
[23] We allow agents to move and perform calculations in parallel, if they are capable of doing so.
[24] Note that part of the following algorithm is aimed at adjusting the agent's behavior to the physical model and does not directly contribute to goal-satisfaction.

(i) Move into the potential-well that models the goal according to the physical properties of the entities involved in the process, as described in the goal-reaching process.
(ii) Perform the goal.
(iii) If $m_a$, the mass of the particle that models the agent, is smaller than $m_g$, the mass of the particle that models the goal, subtract $m_a$ from $m_g$. Else, $m_g = 0$. In a case of depleting resources, $m_a$ is reduced in a similar way.
(iv) Return to the goal-reaching process, step 1.

Note that the calculations in the algorithm above shall be performed using the physical properties that result from the physics model. Therefore, properties of the agents and goals shall be transformed to physical properties, then used for physics calculations, and then re-transformed into agent-goal properties. An example of such a transformation is presented in section 5. The iterative method which we propose leads to a gradual reduction in the number and size of the goals to be satisfied, and will lead finally, to completion of the goals. However, the time that such a process consumes may be excessive. We discuss this problem in the next section. In addition, the convergence of the system to a final state wherein either all of the goals have been satisfied, or it is not beneficial to satisfy the unsatisfied goals is proven below.

## 3 Scaling and convergence

Due to the evolution of the system, goals will be satisfied gradually. As suggested in section 2.5.3 and described in section 2.7, after each static-dynamic collision, the mass of the static particle that models the goal diminishes. This implies that the potential-wells which represent the goals will become less attractive, and therefore the rate of goal-satisfaction will decrease. The overall result of such a phenomenon is a gradual decay of the goal-satisfaction process. This means that in a system of the type which we propose, as the goals get closer to full satisfaction, the time for satisfying the rest of the goals decreases logarithmically [25]. That is, the time for completion of all of the goals is diminishing, but the rate of decreasing gradually becomes slower.

In order to solve this problem, we employ a scaling method. The purpose of this method is to amplify the attraction in the system in order to accelerate the goal-satisfaction process, especially when it slows down due to its physical properties. Since the mass of the particles and the potential-wells directly affect the magnitude of the attraction among them, we shall scale the masses of the entities in the system. However, any change in the physical properties

[25] Such a logarithmic decay is well known in the relaxation of multi-particle physical systems.

in the system, including masses, may change the physical behavior. Therefore, the well-known results of similar physical systems do not necessarily hold, and we may be unable to use them to predict the evolution of goal-satisfaction.

However, we shall use the mass-scaling method in instances wherein it preserves the consistency with the physical results. Actually, the distribution and the size of the system disable information about the temporal change in the quantity of goals and sub-goals that have yet to be satisfied. Therefore, there must either be an on-line synchronization method, or mass reduction will have to be performed according to the initial information that each agent has about the system and some pre-defined rules. For reasons of simplicity and low computational complexity, we shall prefer using some pre-defined rules. The decision regarding these rules is subject to the properties of the system, but can be calculated prior to its implementation. To clarify when mass-scaling instances occur, we shall first present the scaling method:

- During the goal-satisfaction process, after each pre-defined number of iterations (we denote this number by $I$), perform the following:
- Multiply the mass of all of the goals by $C$, a pre-defined factor. The result of this multiplication will cause the treatment of each mass of a particle that models a goal as $m_i C^\tau$, where $m_i$ is the mass of goal $i$ without scaling and $\tau$ is the number of times that the mass-scaling has been performed [26].

Now that the scaling method has been presented, we shall show its validity. Each $I$ steps of the goal-satisfaction process that are performed without interruptions are similar to a physical process. The problem arises from the change that may occur after each $I$ steps due to the mass-scaling; such an event is not analogous to physical behavior. However, we can view the evolving system $\mathcal{S}$ as a set of systems $\{S_{[t_0,t_1]}, \ldots, S_{[t_{k-1},t_k]}\}, \mathcal{S} = \bigcup_{i=1\ldots k} S_{[t_{i-1},t_i]}$ of $I$ evolutionary steps each. The state of $\mathcal{S}$ at time $t_i$, immediately before the $i$th mass-scaling, is the final state of $S_{[t_{k-1},t_k]}$, and immediately after the mass-scaling it is the initial state of $S_{[t_k,t_{k+1}]}$. Since we are interested now only in the sub-systems (which are similar to physical systems) the gap between them (which is not similar to physical phenomena) can be ignored.

### *3.1 Local minima*

Many dynamic physical systems converge to local minima. That is, instead of reaching the point of minimum energy, they reach a stable point in which the energy is not minimal. In our case such a phenomenon might cause partial

[26] The mass $m_i$ may also be modified due to partial goal-satisfaction, but the two modifications are independent. In that case, however, the mass of the particle that models the agent may be modified as well.

satisfaction of the goals in the system, even when the agents can, potentially, satisfy all of the goals. The question to be asked is whether such a problem exists in our model. Referring to the results from physics, it is common for a single particle to be captured in a local-minimum point due to a specific configuration of the forces affecting it at this specific point. However, in large particle systems, it is most unlikely to have all of the particles captured in local minima. The probability of the latter increases as the system becomes less energetic. The systems with which we deal are highly energetic, and only while satisfying goals do they gradually lose energy. Therefore, they will reach a low-energy state only when most of the resources have become depleted. This will happen only when most (or all) of the goals are already satisfied.

From the discussion above we conclude that single agents (only a small number of them) may reach local minima. However, the agent system as a whole will normally avoid local minima, and may be exposed to this risk only after satisfying most of its goals. Therefore, the problem of local minima is of lesser importance in our case. This perception was supported by the results of the simulations we have conducted.

### *3.2 Goal performance time*

An important property of our model is its ability to provide task-allocation and execution for a vast majority of the tasks in the system within a bounded time period[27]. This property stems from two facts: first, the average velocity ($v$) of particles in the model is a known constant; second, as we later prove in section 4.3, the length of trajectories ($l$) that particles in the model traverse is, for most of the cases, bound. Since traversal time ($t$) relates to the length of trajectory and to velocity by $t = \frac{l}{v}$, it follows that $t$ is bound as well. This means that one can expect that the time for executing all of the tasks (or at least the vast majority of them) is bound.

## 4 Complexity

The complexity of the model that we provide results from two main factors. One factor is the time necessary for reaching a single goal by a single agent during a single time-segment. The other is the number of time-segments necessary for the completion of the goal-satisfaction procedure, including the time that the agents spend on actual goal-satisfaction. The ratio between the num-

[27] This is subject to having sufficient resources and appropriate capabilities available.

ber of goals and the number of agents within the system has a major effect on the complexity. These factors are analyzed below [28]. We begin our analysis in section 4.1 where we discuss the time consumption during a single time-segment and only for the virtual motion (and not for the goal-satisfaction) of the agents. Then, in section 4.2 we analyze the time consumption of the whole process and prove its convergence to a solution. However, in this section we do not yet incorporate the goal-satisfaction effect on the performance of the process. This analysis is presented in section 4.3.

### *4.1 Time consumption during a single time-segment*

The time that each agent consumes at each time-segment for calculating its progression in the goal-domain is equal to the number of time-steps $dt$ that comprise the time-segment, multiplied by the time necessary for the calculations within the time-step. In order to express the time-consumption we must first present some concepts and notations and formulate the relationships among them. The number of agents and goals within the range of mutual interaction depends on the density of the distribution of agents and goals within the domain of the system. That is, it depends upon the average distances between agents and goals, and not on their total quantities. We denote the number of agents by $N(t)$, the number of goals by $G(t)$, the total area of the goal-domain by $S(t)$, the density by $n(t)$ and the average virtual distance between agents and goals by $d(t)$. We express the above as functions of the time $t$ since they may vary over time. However, in the following analysis we omit $t$ for ease of representation. Using these notations we can state the following [29]

$$n \equiv \frac{N+G}{S} \quad \text{and} \quad S \sim (N+G)d^2 \tag{6}$$

From these basic relations we can conclude that

$$d \sim \sqrt{\frac{1}{n}} \tag{7}$$

Now that we have formally presented the relation between the average distances and the density of the agents and goals in the system, we can formu-

[28] Note that the analysis is constructed from several incremental steps. The reader who is not interested in all of the details of the analysis may skip directly to section 4.3.

[29] We use here the notion of area for reasons of convenience, but we do not restrict ourselves to the two-dimensional case. Rather, we must note that a high dimensionality of the system may reduce the computational complexity. If the dimensionality is denoted by $m$, then the relations above can be converted to $S \sim d^m$ and $d \sim n^{-1/m}$, and this modification may affect the complexity.

late an expression for the time consumption of a single agent depending on the density of the system. The average numbers of agents and goals within the range of interaction are given by:

$$N_I = \frac{N}{N+G} n\pi r_I^2 \quad \text{and} \quad G_I = \frac{G}{N+G} n\pi r_I^2 \tag{8}$$

correspondingly, and the single-agent time-consumption $C_t$ per time-segment is:

$$C_t = ((N_I + G_I)t_V + 2t_{int} + t_T)\mathrm{Itr} \tag{9}$$

where $t_V$ is the time necessary for calculating the mutual potential and deriving the potential function, $t_{int}$ is the time necessary for integrating the force and the velocity to yield the velocity and the displacement respectively, $t_T$ is the time for calculating the translation from the previous displacement to the new displacement, and Itr is the number of $dt$ iterations [30] . The derivation and integration that are necessary for calculating the change in the state of an agent may be complex, but as explained in section 2.5.4, this complexity can be reduced. In addition, this complexity does not depend on the size of the system $S$ or on the number of agents $N$ within it. As a result, the complexity of these calculations becomes a small constant.

An interesting property of our model arises from the relationships above: the number of $dt$ time-steps, that is, Itr, increases slowly with respect to the number of agents or even stays constant, as shown below. In cases where the size $S$ of the goal-domain is fixed, as $G$ and $N$ are increased, the average numbers of agents and goals within the range of interaction $G_I$ and $N_I$ correspondingly increase linearly. However, the number of iterations per agent, i.e., Itr, decreases with respect to the growth in $N_I$ and $G_I$. This reduction in Itr results from the reduction in $d$, the average distance between agents and goals, and depends on some physical properties, too: in large-scale physical systems, the velocity-distribution is a very narrow normal distribution. This means that most of the velocities are close to the average value. We assume that the velocities do not change due to a change in the number of agents in the fixed-size system (which is a reasonable assumption). Under such an assumption, and according to equation 7, when $N$ and $G$ are increased, $d$ decreases proportionally to $\sqrt{N+G}$. A reduction in the distance entails a linear reduction in the number of time-steps necessary for completing the passage of the distance, as can be observed in physics. From this, we conclude that the increment in the numbers of agents and goals leads to a reduction in the number of time-steps necessary for reaching a goal which is inversely-proportional to $\sqrt{N+G}$. Merging this

[30] Note that we alternate between iterations and time-steps, but they have the same meaning.

conclusion with equation 9 yields:

$$C_t \sim (N_I + G_I)\text{Itr} \sim \sqrt{N+G} \tag{10}$$

This increment in $C_t$ holds when the size of the system is fixed. However, such a situation does not necessarily hold. It may be that both the number of the agents and the size of the system have simultaneously been increased. In such cases we shall examine the density of distribution of agents among the system area. This density $n$ is defined in equation 6 as $\frac{N+G}{S}$. In cases that $n$ stays constant, $d$ does not change (see equation 7). Therefore, increasing $N$ leaves $N_I$ unchanged and increasing $G$ leaves $G_I$ unchanged. The fact that $d$ does not change implies that Itr remain constant, too. Thus we derive

$$C_t \sim (N_I + G_I)\text{Itr} \sim \text{const} \tag{11}$$

This is a very important property of the model that we propose, since many systems may preserve their density when they grow larger, and the result seen above promises computational complexity which does not depend on the size of the system.

### *4.2 The general time-consumption for virtual motion*

An important issue for the analysis and the assessment of any algorithm is both its computational complexity and its convergence to a solution. Since the complexity of a single time-segment of the algorithm has been calculated above, we have to complete this assessment by proving that the algorithm converges to a solution and to calculate the complexity of reaching the solution. In this section we do not yet take the goal-satisfaction time-consumption into account. This shall be done in section 4.3.

If we assume the worst case, where at each time-segment only one goal is reached by an agent and only part of this goal is satisfied by the agent, then the number of time-segments for satisfying all of the goals will be $O(G)$. This is because even if each agent completes only a small fragment $f$ of a goal, (and we assume that the size of $f$ does not depend either on the number of agents or on the number of goals), the number of time-segments for the completion of a single goal will be $\frac{1}{f}$ and the number of time segments for completion of all of the goals will be $\frac{G}{f}$. However, the average case is usually better. Since all of the agents are working simultaneously on goal-reaching, Itr – the number of time-steps $dt$ for reaching goals by the agents, decreases. In a case that $N_I < G_I$, this reduction is bounded by $\frac{G}{N}$, (note that $\frac{G}{N} = \frac{G_I}{N_I}$). $\frac{G}{N}$ is the optimal number of time-segments for satisfying all of the goals when $N < G$. The reason for the reduction in Itr is that when an agent has reached a goal, the other agents (except for an average of $\frac{N_I}{G_I}$ that were moving towards the same goal) have also progressed toward goals. Therefore in the next time-segment they will have to move only the remaining distance. We must emphasize that in the case of $G_I > N_I$, this reduction holds only for $O(N_I)$ distances, and not necessarily for all of the distances.

The magnitude of the average distance-reduction during a time-segment is not always known. We have two major assumptions about the average distance-reduction during a time-segment. The first assumption is of a constant reduction from the original distance, which does not depend on the number of time-segments that have already taken place. The second assumption is of a reduction that depends on the number of the time-segments that have taken place. The conclusion from the first assumption is that $d$, the average distance, is reduced to a fraction of its original value. This does not improve the order of complexity of the worst case, although the factor of the complexity is smaller. The second assumption requires that at each time-segment the distance decreases, on average, to $\frac{1}{x}$ of its size in the previous time-segment (where $x > 1$ and constant). The conclusion from this assumption is that the distance that was $d$ at time-segment $t$ will decrease to $x^{-k}d$ at time-segment $t + k$, where $k$ is an arbitrary integer. Since the time of progression toward goals is linear with respect to $d$, the average time will diminish in the same manner as $d$, the average distance, does. Subject to the second assumption, the total of the distances during the whole goal-satisfaction process is the sum over all of the time-segments, and is proportional to

$$\frac{G}{N}\sum_{i=1}^{G}\frac{1}{x^i} \tag{12}$$

or, in the case of the first assumption

$$G\sum_{i=1}^{G}\frac{1}{x^i} \tag{13}$$

The sum in the equations above has a constant upper bound which is equal to $\frac{1}{x-1}$ and does not depend either on $G$ or on $N$. Resulting from this sum are both the convergence of the algorithm to a solution and the expected complexity of the general procedure of goal-reaching. In cases where equation 10 holds, this complexity is $O(G\sqrt{N+G})$ in the worst case and $O(\frac{G}{N}\sqrt{N+G})$ in the average case. In cases where equation 11 holds, the complexity is $O(G)$ in the worst case and $O(\frac{G}{N})$ in the average case. The last result yields from the constant time necessary for each time-segment according to equation 11 and the constant-bounded number of time-segments. In a case that $N > G$, the complexity is reduced either to $O(\sqrt{N+G})$ when equation 10 holds or to $O(1)$ when equation 11 holds.

The last result is disconcerting. In order to resolve this, we must bring forth the rationale for the case of constant-time complexity. When dealing with very large-scale systems of agents, the whole system may be observed as an ensemble of sub-systems, all of them relatively independent, and acting independently in parallel. Thus, the time consumed by a single system for completion of its activities is equal to the time necessary for the whole ensemble. For any system that can be arbitrarily partitioned into many smaller sub-systems such that they are large enough to fit our algorithm requirements, this rationale holds.

One can claim that the partition into sub-systems does not provide an appropriate description of the whole system, because there may be interactions between entities from different sub-systems. We agree that such interactions are possible. However, we can resolve this as follows. If we allow partial overlap of adjacent sub-systems, then the overlapping ranges will represent regions of interaction between entities from adjacent sub-systems. These common interaction regions must be large enough to allow for interaction and small enough to cause only minor effects with respect to the whole system. Since the range of interaction is $r_I$ and the size of the whole system is much greater, we find no difficulty in performing a partition into sub-systems that will satisfy the size requirements of the overlap region. Thus we reject the claim.

### *4.3 Goal-satisfaction and computation time*

We would like to examine the performance of our algorithm with respect to the optimal agent-goal allocation [31] . This allocation depends on the properties of the goal-agent system. Therefore, we shall partition the problem into several sub-problems. We shall deal with cases where $N \leq G$. We discuss hereby the effect of the time necessary for goal-satisfaction on the performance of the algorithm. This discussion expands the analysis in sections 4.2 and 4.1, that considers only the computation time, and completes it. We shall partition the problem subject to the following parameters: size of goals; size of agents; time consumed for goal-reaching (denoted by $t_r$); time consumed for goal-satisfaction (denoted by $t_g$).

#### *4.3.1 Equi-size goals*

The first and the simplest will be the case where all of the goals are of the same size and all of the agents are such that each agent satisfies a goal with the same efforts and time consumption. In such cases, the optimal allocation will cause each agent to perform $\frac{G}{N}$ goals, as shown in section 4.2. We examine the following sub-cases:

[31] This examination is mainly based on the trajectories particles in the model travel, however it has further implications.

(i) $t_g \ll t_r$:
In such a case the goal-satisfaction time is negligibly small. Therefore, the discussion and the results of section 4.2 (in which $t_g$ was ignored) hold.

(ii) $t_g \sim t_r$:
The fact that the time necessary for goal-satisfaction is not negligible any more may have a vast effect on the complexity of the algorithm. However, in the current case, where agents complete goals by themselves, it has a minor effect, and the complexity remains $O(\frac{G}{N})$.

(iii) $t_g \gg t_r$:
In this case, since the main factor becomes $t_g$, but the number of goals that each agent performs is still $O(\frac{G}{N})$, the time consumption will be $\frac{t_g}{t_r}\frac{G}{N}$ per agent.

The proof for the results above is straight forward. The PAS algorithm allocates agents to goals mainly according to their capabilities[32]. That is, it is more probable that agents with more capabilities will be matched to goals that require more capabilities in order to be resolved. In a case that the agents possess of more capabilities than necessary for satisfying the goals (as in the case with which we currently deal), the rejection concept of the PAS model will prevent the allocation of more than one agent to one goal, and therefore all of the agents will be allocated to different goals. Agents that will complete performing a goal will be re-allocated to other goals. Thus, no hindrances may occur and the average allocation ratio will be $O(\frac{G}{N})$ goals per agent, as for the optimal allocation.

### *4.3.2 Goals of various sizes*

The second case should have been the case where the goals are still of the same size, but not all of the agents can complete the satisfaction of a goal by themselves. However, several encounters must occur before all of the goals are satisfied. Therefore, after each encounter some of the goals diminish. As a result, the goals in the system do not remain of the same size. Hence, the case with which we deal will be the case in which both the agents and the goals have a variety of sizes. This may cause agent-goal allocation that will lead to a partial goal satisfaction, and the yielded reduction in the size of the goals will affect the proceeding allocations. We denote the size of goal $g_i$ by $S_{g_i}$ and the size of agent $A_j$ by $S_{a_j}$. The average goal size is denoted by $\overline{S_g}$ and the average agent size is denoted by $\overline{S_a}$. In section 32 the average number of goals per agent was expressed by $\frac{G}{N}$. This average is insufficient for the analysis of goals and agents with various sizes. Here, the average amount of goals-units to

[32] The allocation depends on the distances of the agents from the goals, too, but for agents with similar distances, the capabilities will affect the allocation.

be approached (and satisfied) by an agent, with respect to agent-units amount, shall be considered. This average (denoted by $\mu$) depends on the sizes and not on the number of goals and agents and, for an agent $A_i$, is given by

$$\mu_i = \frac{\sum_j S_{g_j}}{S_{a_i} \sum_k S_{a_k}} \tag{14}$$

We consider three cases, all in which $\overline{S_a} < \overline{S_g}$:

(i) $t_g \ll t_r$:
As before, such a case entails a negligibly small goal-satisfaction time. This implies that the order of the complexity is not modified and is $O(\frac{G}{N})$ in average. However, the sizes of goals and agents affect the factor of the complexity. The time consumed for goal-satisfaction is negligible, but the number of goals performed by an agent $i$ is not the only factor that influences the complexity. An important factor is the total virtual-distance that an agent passes in order to reach the goals that it fulfills. That is, it would be better for the performance of the system that agents will be allocated to tasks in a way that will minimize their virtual trajectories. The system will complete the performance of its goals when the last agent will complete its last goal. Therefore, an optimal goal-agent allocation should seek the minimization of the maximal trajectory. We must check if in the case where $\overline{S_a} < \overline{S_g}$ our algorithm reaches results close to optimal, and provide a method for measuring this proximity.

The PAS model "prefers" the allocation of closer entities over the allocation of distant entities. It also prefers the allocation of greater entities over the allocation of smaller entities. However, the size has only a linear effect on the interaction, where the distance has a high-order polynomial effect. As a result, our algorithm will give priority to the allocation of closer entities. This will be done with respect to the other entities within the range of interaction. We can view this allocation as a Traveling-Salesman Problem (TSP) [33]. Via this observation, we can assess the complexity of our algorithm with the specific settings.

The TSP problem is of finding a tour through nodes in a graph where each edge has a cost, and the tour should visit all of the nodes with a minimal total cost. It is similar to our problem in its property of finding trajectories between nodes with minimal costs. In our problem the agents seek a trajectory with a minimal length (which is similar to cost). There are several differences between TSP and our case as we detail below. These differences result in TSP algorithms being inappropriate for our needs. Yet, the complexity inquired by such algorithms for selecting a trajectory with a minimal cost is useful for comparing with the complexity of our model.

[33] A comprehensive description and discussion of the TSP can be found in [36]

TSP differs over the number of agents dealt with (a single agent, whereas we deal with multiple agents). Multiple agents, in PAS, must avoid conflicts among themselves. Therefore, there are occasions where agents prefer longer paths in order to avoid conflicts. There may also be a decision-problem upon which agent should perform each task, especially when two agents (or more) with the same capabilities are approaching the same task.

We use a TSP approximation algorithms as a means for the assessment of the quality of the trajectories that result from our algorithm. For this purpose we find TSP an appropriate reference. The TSP, which is an NP-complete problem, has several polynomial approximation algorithms [44]. Some of them are proved to reach a solution with a cost which is not greater than twice the optimal cost (i.e., the ratio-bound is 2). This ratio-bound is achieved in cases where the cost function $c$ satisfies the triangle inequality. That is, all of the nodes $x, y, z$ satisfy $c(x,z) \leq c(x,y)+c(y,z)$. In cases where the costs are the distances between the nodes, the Euclidean geometry implies that the triangle inequality holds. The PAS model consists of such distances between particles[34], and therefore a particle's progression can be viewed as a salesman trajectory. The assessment of the results of our algorithm we, using comparison to the results of an approximated TSP solution, is provided below.

Our algorithm implies that a dynamic particle will most probably (but not always[35]) reach the closest static particle. This can be viewed as a greedy algorithm in which the shortest distance (cost) is chosen with a probability $1-\delta$, where $\delta \ll 1$. This probability depends on the ratio between the number of agents $N$ and the number of goals $G$ or on the ratio between the corresponding areas. The reason for $\delta \ll 1$ arises from the physical model. A dynamic particle will usually experience the strongest force from the closest static particle and therefore will move directly towards this particle. This should happen unless $N \gg G$. However, even in the latter case, where many dynamic particles tend to reach the same static particle, if $\sum_i S_{a_i} < \sum_j S_{g_j}$ all of the dynamic particles shall have no preclusions when moving toward the same static particle. Therefore, the probability of reaching the closest static particle is close to 1, and $\delta \ll 1$. In cases that $\sum_i S_{a_i} > \sum_j S_{g_j}$ $\delta$ is not guaranteed to be small, but $\mu$ will be small.

---

[34] When we deal with an abstract space, we define abstract distances. If appropriately defined, these may conform with the triangle inequality. For instance, in a case of "distance" between keywords, having three databases $D_1, D_2, D_3$ and distances measured by the number of different keywords, one can prove that $d(D_i,D_j) + d(D_j,D_k) \geq d(D_i,D_k)$, $i \neq j \neq k \in 1,2,3$ regardless of the order of $i, j, k$.

[35] This is because it may be rejected by another dynamic particle that is moving towards the same static particle.

A greedy TSP approximation algorithm in which the shortest distance (cost) is chosen with a probability 1 was proved[36] to have a ratio-bound of 2. The $1-\delta$ probability we introduce will cause only a small change in the results, and therefore the resulting trajectories of our algorithm will not be far from the ratio-bound of 2. This depends on the number and the sizes of the modeling static particles with which each modeling dynamic particle collides during the iterative goal-satisfaction process. The modified ratio-bound will therefore be $2/(1-\delta)^{\mu}$, where $\mu$ is the number of goal-units that an agent has approached during the goal-satisfaction process[37]. The average $\mu$, that is denoted by $\overline{\mu}$ will be

$$\overline{\mu} = \frac{\sum_j S_{g_j}}{\overline{S_a} \sum_k S_{a_k}} \tag{15}$$

where $\overline{S_a}$ is the average agent size. However, the worst case that is related to a specific agent $A_i$, denote by $\mu_i^w$, may be

$$\mu_i^w = \frac{\sum_j S_{g_j}}{S_{a_i}} \tag{16}$$

where $S_{a_i}$ is the size of agent $A_i$. However, the worst case has a very low probability, because it happens only when all of the goals are performed by one agent, and all of the others perform nothing. According to the analysis above, if $\delta$ and $\mu$ are small, the resulting trajectories of our algorithm are not too far from the optimal TSP trajectories. Note that this property is not affected by the size of the system.

To summarize the case where $t_g \ll t_r$, the order of the time consumption for goal-reaching and goal-satisfaction is the number of goals per agent multiplied by the ratio-bound, i.e., $2\mu/(1-\delta)^{\mu}$. Note that this may be a large number in cases where $\delta$ and $\mu$ are large, but as explained above, in some cases a big $\delta$ implies a small $\mu$.

(ii) $t_g \sim t_r$:
Here, the influence of the goal-satisfaction time comes into account. In the previous case we have presented an analysis of the quality of the trajectories that result from the PAS model. This analysis holds for the current case too. As a result of the time consumption for goal-satisfaction, the complexity from the previous case will be multiplied by a constant, but will remain of the order of $2\mu/(1-\delta)^{\mu}$. The influence of the quality of

[36] The algorithm is based on a greedy minimal spanning tree algorithm. The details and the proof are found in algorithms textbooks such as [7]. This greedy choice is similar to the physical behavior of a particle, which "selects" the direction in which the maximal force is applied. Of course, the reason of this behavior is totally different.

[37] Note that the number of goals per agent here is different from this number in the equi-size goals case, which is $O(\frac{G}{N})$.

the trajectories can be omitted only in the case of $t_g \gg t_r$, as described below.

(iii) $t_g \gg t_r$:

In this case, since the time consumption for goal-satisfaction is much greater than the goal-reaching time, we shall ignore the latter. The goal-satisfaction time of an agent is the sum of the time-periods consumed for the satisfaction of goals by this agent. This sum depends on the goal allocation, which depends on the size of the agent and the sizes of the goals. Therefore, it will be:

$$\frac{t_g}{t_r}\frac{\sum_j S_{g_j}}{S_{a_i}\sum_k S_{a_k}} \tag{17}$$

## 5 Examples

To illustrate the method in which our algorithm may be implemented, we present three examples. In the first (section 5.1) we recall the hole-filling problem and provide a detailed adjustment of the PAS model to this problem. In the second example (section 5.2) we present a transportation system and show that the PAS model can be used for the implementation of cooperative goal-satisfaction in such systems. The third example (section 5.3) illustrates the applicability of our model to real-world MAS with abstract tasks.

### *5.1 Hole-filling robots*

There is a planar surface with holes in it spread in an arbitrary order. The holes have various sizes. On this surface, robots that are designed to fill holes are able to move towards holes and fill them. The robots have various (limited) quantities of filling. The robots are all members of a system which is aimed at filling as much hole-volume as possible. The holes and the filling have the same volume units (e.g., cubic centimeters). The robots have to reach the holes and then fill them. For reaching the holes, the robots are able to move and they have a velocity. For our algorithm illustration, it will be more convenient to assume that the velocities of all of the robots are of the same order of magnitude. This assumption is necessary because otherwise, the assumption that the robots can perceive what happens in their neighbourhood might be violated. This is because fast robots will move at a velocity that will cause them to “appear” too close to other agents before they have been detected.

Hole-filling and moving are both time consuming. The moving time can be

Fig. 4. Holes/robots and particles scaling units

| *Holes/Robots* | *Particles* |
|---|---|
| hole/stuff volume (cubic mm to m) | mass ($\sim 10^{-27}$kg) |
| diameter of the hole (mm to m) | $r_0$ ($\sim 10^{-10}$ m) |
| sensors' effective range (m) | $r_I$ ($\sim 10^{-9}$ m) |
| movement velocity ($\sim$ m/sec) | movement velocity ($\sim$ m/sec) |
| average path time is $\sim 10^1$ sec | average path time $\sim 10^{-8}$ sec |
| $dt_r = 10^{-1}$ sec | $dt_p = 10^{-10}$ sec |
| hole-filling time: $t_{fill} = \frac{V_{hole}}{R}$ | collision time: depends on the distance but limited[38] by $100dt_p$ |
| $t_{fill} = 100dt_r$ | $t_{collision} = 100dt_p$ |

expressed by

$$t_{move} = \frac{distance}{velocity} \tag{18}$$

There is a filling rate which does not depend either on the size of the hole or on the amount of filling that the robot holds. This filling rate depends only on the filling capabilities of the robot. It will be simpler to assume that all of the robots have the same capabilities, and they differ only by the quantities of filling they possess. If we denote the filling rate by $R$, then the time for filling a hole of volume $V_{hole}$ by a robot (that has at least an amount of $V_{hole}$ of filling) is given by

$$t_{fill} = \frac{V_{hole}}{R} \tag{19}$$

### *5.1.1 Fitting the model to the problem*

We present the fitting properties in figure 4. Note that in the table most of the properties on the left column are the given parameters of the MAS, and only $dt_r$ is determined by the designers when fitting the PAS model to the MAS. The designers may also limit the effective range of the sensors or let their agents use only part of the information gathered by these sensors.

[38] This limitation is required in the PAS model as described in section 2.4.

As can be observed from the table, the average robot velocity and particle velocity are of the same order. Due to the large difference of distance-magnitudes, the time-periods are different, too. The mass of particles is much smaller than the volume of holes and filling stuff and has different units. However, this difference should not affect the match of the physics model to the robot implementation. This is because the Lennard-Jones potential, which is employed in our model, is calculated with respect to a single mass-unit, and a particle is assumed to have one mass unit. Therefore, one volume unit in the robot case shall be represented by one mass unit in the physics model. Modification in this requirement will not cause any significant change in the behavior of the system, because any other linear scaling of hole-volume to particle-mass will only cause multiplication of all of the derived results by a constant. The meaning of such a multiplication is no more than zooming.

In the case of holes, which have both radii and volumes, we must take into consideration the relation between them. If a hole's dimension is $d$ and its volume is $V_{hole}$, then the diameter of the hole will be: $dia = \sqrt[d]{V_{hole}}$. This implies that the mass of the particle and its diameter $2r_0$ must satisfy $2r_0 \simeq \sqrt[d]{mass}$. Such a relation does not necessarily exist, and therefore when matching the hole/robot system to the PAS model, the mass of the particles shall be chosen to satisfy this relation. As can be observed from the table (and required by the model), the radius of interaction should be about ten times greater than the radius of the particle. In the robots' system, it may be that the sensors can perceive information from distances that are more than ten times greater than the hole radius. This is not problematic because the robots can either ignore distant entities (which is simple but wasteful), or use this surplus information to make more calculations and elucidate the solution. The latter, of course, requires better computational capabilities, and shall be considered with respect to these. In a case that the designers decide to use the greater sensing capability, the ratio $r_I/r_0$ shall be changed correspondingly.

In order to implement the robot/hole system using the PAS model, designers shall use the matching that we have presented above. That is, during the activity, each robot will perceive the properties of the other holes and robots within the range of its sensors, and will process this information. This processing will include the transformation from the robot/hole units to the particles unit, if necessary, and then the calculations that are required according to PAS. The results of the calculations according to PAS will be reversely transformed to the robot/hole units, and the robots will advance according to these results. The perception-transformation-calculation-retransformation-action procedure will be repeated by each robot every $dt_r$, as required by our model. Since the transformed $dt_r$ is 0.1 second, it will be a sufficient time for any reasonable sensor and processor to perform both the perception and the required calculations.

### *5.2 Applying the PAS model to transportation*

Another example of MAS in which the PAS model can successfully be implemented is a freight transportation system. Such systems have been discussed in the context of DAI, e.g., in [19,20,47,61]. In a case where each carrier is controlled by a computational agent, a coordination mechanism is necessary. If the system is large and communication is limited, agents that try to increase the common benefits of the system can act according to the algorithm that we provide, thus improving the system's overall performance. This improvement stems from the avoidance of the intensive computation of the scheduling problem and the circumvention of bottlenecks that may appear in a centralized mechanism. In order to enable such an implementation, we should adjust the PAS algorithm to the transportation system. This adjustment is described below.

Fig. 5. Carriers/freights and particles

| *carriers and freights* | *particles* |
|---|---|
| carrier | dynamic particle |
| freight | static particle |
| carrier/freight volume | particle mass |
| carrier location | dynamic particle location |
| freight location | static particle location |

The system of freight transportation consists of many carriers. Each carrier has a freight-carrying capability that is given in units of volume and has a given location. The tasks that the carriers must fulfill are freight-transportation tasks. We deal here with freights that should be moved from various locations to other locations. Therefore, each task can be characterized by its original location, final location and volume. The adjustment of the model to the freight system is presented in figure 5. As can be seen from the table, each task will be modeled by a static particle and each agent by a dynamic particle, as required by the PAS model. The volumes will be modeled by particle masses, and the locations by particle locations.

A necessary feature in any agent system and, in particular in the case of transportation, is a communication system. This system must allow for the broadcasting of information concerning task execution and locations of agents and tasks [40]. Failing to handle such information, the ability of the system

[40] Such communication systems exist in transportation companies, and drivers report their location and task execution occasionally. However, although many drivers

to supply transportation services may deteriorate (see section 6.2). However, complete and accurate on-line information may not be assumed. Agents are expected to transmit information concerning their location and goal-satisfaction, but this transmission is usually received only by a subset of the other agents. Even within the target subset of agents there may be some which receive an inaccurate message or do not receive a message at all (we have examined the effect of such message transmition via simulations as described in section 7).

The structure of cities and roadways regulations may prevent movement along the shortest path between two locations, as assumed by the general algorithm. Thus, in the simulation of the transportation example (section 7), the distance between two locations $l_1$ and $l_2$ is calculated as the shortest way that one could drive from $l_1$ to $l_2$. In addition, if the direction for movement[41] $\hat{v}$ calculated by the agent in the goal-reaching process algorithm does not agree with a road direction $\widehat{road}$, then the road with the smallest angel with $\hat{v}$ is selected for movement. This selection is not different from a physical behavior in environments with obstacles, and therefore justified. Another limitation imposed by regulation is a speed limit. We take this to be 50km/hr. A limit on velocities is not part of the physics model, however velocity distribution may allow for values above the limit to be of low probability, thus eliminating them should have a minor effect on the overall behavior of the system, as our simulations show.

Yet, a more detailed adjustment is necessary to enable the implementation of PAS for the transportation case. Therefore, we summarize the above and expand upon it below. We distinguish between two levels of adjustment – the conceptual level and the practical level. The conceptual level consists of the following:

(i) The reception of information concerning a freight-transportation task is modeled by the entrance of a particle into the effective radius of interaction $r_I$. Note that in the transportation case the information is usually gathered via receptors which are not always affected by the distance from the carrier to the freight. This may imply an excess of information. To handle this excess, the information can be filtered to refer only to the most relevant tasks.

(ii) The advancement towards a freight is modeled by the movement of a dynamic particle towards a static particle.

(iii) The execution of a freight-transportation task is modeled by the collision between a static particle, which models the task, and a dynamic particle, which models the agent.

---

receive the transmitted information, usually only the central coordinator uses the information for planning and allocating tasks.

[41] The notation $\hat{v}$ refers to the direction of a vector $\vec{v}$.

In addition to the conceptual level, we present the practical level. As seen previously, this is performed via matching the properties (see figure 6). Most of the properties in the left column of the table are the typical parameters of the specific transportation system with which we deal. Only $dt_c$, which is the time-period[42] of the transportation system, is determined by the designers when fitting the PAS model to the MAS. The radius of interaction $r_I$ in the transportation domain is $\sim 10^3 m$. As can be observed from the table, the average carrier velocity is ten times greater than the particle velocity. This, in addition to the large difference of distance-magnitudes, results in different time-periods as well. That is, $dt_c$ is significantly different from $dt_p$, which is the time-period of the particle system.

Fig. 6. Carriers/freights and particles scaling units

| *Carriers/freights* | *Particles* |
|---|---|
| carrier/freight volume (cubic m) | mass ($\sim 10^{-27}$kg) |
| delivery distance ($\sim 10^4$m) | $r_0$ ($\sim 10^{-10}$m) |
| information range ($\sim 10^3$m) | $r_I$ ($\sim 10^{-9}$m) |
| movement velocity ($\sim$ 10m/sec) | movement velocity ($\sim$ 1m/sec) |
| average reaching time $\sim 10^3$sec | average path time $\sim 10^{-8}$sec |
| average delivery time $\sim 10^3$sec | collision time $\sim 10^{-8}$sec |
| $dt_c = 10$ sec | $dt_p = 10^{-10}$sec |

Note that the magnitude of $dt_c$ is 10 seconds (which is required to conform with the $10^{-11}$ ratio between times in the physical model and the computerised system). This implies that each agent must perform the calculations required according to the PAS model every 10 seconds, and accordingly, move to a new location[43]. Agents must report their location occasionally, to enable other agents to use the information about the location for their calculations. However, it is not required that the agents report their location every 10 seconds. Such a requirement may be impossible if the reports are performed by human agents. It would, however, be possible if an electronic tracking component is installed in every agent. Such a device can broadcast its location and an identification code, thus providing the necessary information with a high frequency and a low cost. Nevertheless, even without the high-frequency reports we have suggested, the system can maintain its stable behavior and continue satisfying goals. The quality of the calculations will deteriorate in such cases, however

[42] The time period is the typical time-step for the iterative calculation as presented in detail in section 2.4.

[43] In the simulations we set it to 1 instead of 10 to speed up the computational process.

reasonable results and goal performance can still be achieved, as explained in section 6.2, where imprecise information is discussed.

The difference between mass of particles and volume of carriers and freights is of the same type as in the hole-filling example. Other adjustments are similar to the adjustments proposed for the hole-filling domain, and may be similarly justified. To avoid redundant repetitions, we refer the reader to section 5.1.

The above adjustment allows computational agents to use the PAS algorithm to coordinate their actions in order to perform the freight-transportation tasks and increase the benefits of the system. This will be done with very low computation and communication efforts, as the simulations (section 7) demonstrate.

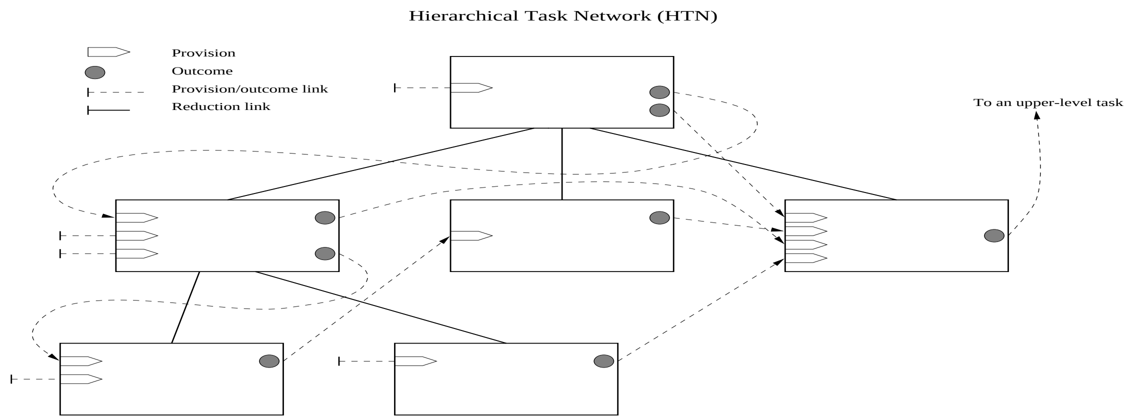
### *5.3 Application to abstract tasks*

Although based on physics, the PAS model may be applied to MAS that execute non-physical tasks. To demostrate this capability, we provide below guidelines to the application of PAS in a real-world multi-agent architecture named RETSINA [55,56]. In brief, a RETSINA MAS consists of three types of agents – information agents, task agents and interface agents. Agents have capabilities to perform tasks and capacities (of these capabilities) which limit the size of the task or the amount of resources being used for its execution. In RETSINA, agents do not know about other agents in advance and may find them via matchmaker- or broker-agents, which are both a specific type of information agents. To date, agents in RETSINA are implemented such that each has a single capability [44]. As such, since for some tasks several capabilities are necessary, the agents form teams on demand to execute these tasks. Note that an agent $A$ in a RETSINA MAS has a limited view of the rest of the agent society. This view includes mainly agents and tasks which are relevant to $A$'s capabilities and tasks. This confined view is similar to, and modeled by, the limitation on interaction range in the PAS model.

Agents within RETSINA satisfy goals by interleaving planning and execution. During this process they gradually de-compose a high-level goal to tasks and subtasks. The lower-level tasks consist of executable code with several pre-conditions. Whenever a lower-level task becomes executable (i.e., all of its pre-conditions are satisfied), it is scheduled for execution and executed when appropriate. The planning mechanism implements an extension of the Heirarchical Task Network (HTN) model [18]. HTNs are heirarchical networks as depicted below. They consist of task-nodes which are connected by two types of edges. Reduction link edges describe the de-composition of a high-level goal

[44] This does not prohibit more complex agents in the future.

to tasks and subtasks (a tree structure). Provision/outcome link edges represent value propagation between task-nodes. Provision/outcome propagation allows one task $T_1$, as it completes execution, to propagate an outcome to another task $T_2$, where this outcome is a provision for the execution of $T_2$. For instance, suppose $T$ is a task of buying a stock. $T$ may de-compose to finding the price ($T_1$) and performing the transaction ($T_2$). The latter ($T_2$) requires that $T_1$ be executed, and therefore $T_1$ should propagate a success outcome to $T_2$ when completed successfully. This outcome is a provision for $T_2$. HTNs allow task-nodes to have multiple provisions and outcomes. A task is executable if it cannot be further reduced and all of its provisions are set (as a result of either outcomes of other tasks or a setting from an outside source). We will use the HTN properties of the RETSINA agents to demonstrate how the PAS model can be used to coordinate task allocation and execution.




The main idea of this modeling is that agents gradually allocate themselves to tasks, sometimes through partial execution of other tasks[45]. Each low-level task (that cannot be further reduced) needs a specific capability for its performance, however usually several pre-conditions must become true (this happens by setting provisions) before a task becomes executable. An agent $A$ would be "attracted" to a task $T$ when it has the required capability and capacity for executing $T$. $A$ would gradually advance toward $T$ by performing the actions which are necessary for satisfying the pre-conditions of $T$. As the number of the latter which are yet false decrease, the agent becomes "closer" to the task. Note that this does not prohibit cases where two (or more) agents simultaneously work on setting true pre-conditions of the same task. Only when these two agents have both the same capability (and excess capacity) will "mutual rejection" among them prohibit simultaneous work on the same task. No direct interaction will emerge among agents who have different capabilities.

To adapt to PAS, we must explain how rejection and attraction, as described

[45] Partial execution is possible when a task is comprised from several sub-tasks, and some of the sub-tasks are executed.

above, are practiced in the RETSINA framework. This would be possible if we can measure and numerically express distances between agents and tasks. For this we define an abstract space – the provision space – which is a space that consists of all of the provisions allowed in RETSINA. Since provisions are represented by boolean values, the displacement of a task $T$ would be represented by a vector of provisions $P_T$, where all of the provisions which are the specific pre-conditions of this task are set to 1 (true) and all of the others are 0 (false). Suppose an agent $A$ has the right capabilities and sufficient capacities and attempts to gradually allocate itself to $T$ and perform it. $A$ would have a provisions' vector $P_A$ where all of the provisions of the approached task $T$ which have already been set are 1, and all other provisions[46] are 0. Note that the provisions of a task may have been set by other agents during partial execution of the task. The distance between $A$ and $T$ is simply the distance between the provision vectors $P_A$ and $P_T$, which is $d_{P_A,P_T} = \sqrt{\sum_i (P_A^i - P_T^i)^2}$.

We define an additional distance measure $D_{P_A,P_T} = \sqrt{\sum_{i|P_T^i=1} (P_A^i - P_T^i)^2}$ (normalized to $T$), where $i|P_T^i = 1$ refers to the $i$'s for which $P_T^i$ is set true. The normalized distance is required for finding an agent-task distance which ignores all other tasks. During the process of setting the provisions true, the distance between $P_A$ and $P_T$ decreases, thus $A$ gets closer to $T$. When $D_{P_A,P_T} = 0$, agent $A$ has completed its allocation to $T$ (in PAS this is expressed by $r_0$). At that point the agent will execute the task (by running its code). Note that $d_{P_A,P_T}$ may be non-zero at this time, indicating that $A$ advanced towards other tasks as well. This also allows the agents to allocate the "closest" one among them, $B$, to a task $T$. Other agents are not as close to $T$ as $B$ is since they were approching other tasks symultaneously. This provides an advantageous behavior, since the latter agents, instead of engaging in task execution (of $T$), will continue advancing towards the other tasks they were already approaching. Thus a selection mechanism, which up to date was not implemented in RETSINA, is introduced. In addition to the definition of distance it is necessary to provide an interpretation of mass in the RETSINA framework. We require that the mass represent the capability of task execution, and its magnitude would be the capacity of this capability. Cases of more than one capability per task (or per agent) are not supported by the PAS model in its current form. Nevertheless an extension of this model to handle such cases is suggested in section 6.1. Moreover, multiple different capabilities may possibly be modeled by several physical charges (e.g., mass and electrical charge). Such an approach is only partially supported by classical mechanics. Extentions may be possible using quantum mechanics, however this is beyond the scope of this paper.

The mechanism presented below discusses the allocation of agents to low-level tasks, however may be used for agent alocation to higher-level tasks as well.

[46] In cases where the agent is working on more that one task some of the other provisions may also be true.

Fig. 7. Agents and tasks in RETSINA vs. particles

| *agents and tasks* | *representation* | *particles* |
|---|---|---|
| agent location | dynamic boolean vector | dynamic location |
| task loaction | static boolean vector | static location |
| task/agent capability | task/agent capacity | particle mass |
| range of interaction | set of relevant tasks/agents | $r_I$ |
| action execution threshold | relevant provisions set true | $r_0$ |
| task performance rate | provisions set true per time | particle velocity |

The executable part of the latter is their de-composition to subtasks. Provisions and outcomes of higher-level tasks are not different from those of low-level tasks. The PAS model requires computation of potential functions and equations of motion. These are used to plan for task allocation and execution. When applied to RETSINA, these computations will be based on the mass and distance as expressed above. The results of these computations are new displacement vectors in the provision space. Upon these an agent should decide what provision to handle (and set true) next, thus advance itself to these new displacements. This will result in agents gradually allocating themselves to tasks by means of provision enablement. A multi-directional advancement is achieved by utilizing and manipulating a provision vector where several tasks (and possibly all of the relevant ones) are represented. Note that the physical movement which is part of the PAS model allows agents not only to get closer to tasks, but move away from them as well. The interpretation of such movements when applied to the provision space is the resetting of provisions (i.e., setting them false). Provisions may become false as a result of, e.g., limited memory of an agent. This may require that it maintain only the most relevant provisions accessible, storing the others externally (in a simillar manner of operating systems handling multiple processes).

For applying the PAS to RETSINA, it is necessary that agents know about relevant tasks and agents and their provision boolean vectors. In RETSINA a task is not a stand-alone entity. Each task $T$ is always in the possession (and responsibility) of some specific agent $A$. This agent is not necessarily the agent that can actually perfrom $T$. In order that other relevant agents know about $T$ and its state, $A$ should add to its advertisement (at a matchmaker agent) the task and its provision vector. Agents who are interested in $T$ may, by this contact information, monitor for updates in $T$'s state. Moreover, members of teams[47] are (which dynamically form in RETSINA to execute a task coop-

[47] The PAS model does not prohibit teams, however does not provide explicit mech-

eratively) should update one another within the team with respect to tasks' and agents' provision-state changes. These mechanisms, which are supported by the RETSINA framework, will provide a "range of interaction" for agents and tasks as required for the PAS implementation.

The PAS modeling of RETSINA is presented in the table above. Note that the modeling presented above only provides guidlines for PAS implementation in RETSINA. Further adjustment and fine-tuning will be necessary to perform this implementation. We intend to persue this direction in future research.

## 6 Extensions of the model

The PAS model, which concentrates on limited range of MAS cases, may be extended to a wider range of situations, relying on its physics properties. We shall demonstrate such extensions of a system with a single type of capability to a multi-capability system, and of a system wherein information is precise (but not complete) to a system with imprecise information.

### *6.1 Multiple types of tasks and capabilities*

The model presented may seem restricted to cases where tasks and capabilities are of a single type. This is not necessarily so, as we discuss below. The capability of an agent to perform tasks and required for a task to be performed by agents is modeled, in the original PAS, by the mass of a particle. This is very restrictive as each particle has only one mass, and therefore it can model only one capability. However, we can extend the model by referring to clusters wherein each cluster is comprised from several particles of different types. Within a cluster, each particle shall model a distinct capability. Such clusters will model the agents and the tasks. Having done this extension, we may analyze the system as a superposition of several systems, each of which consists of one type of particles and thus models one type of capability. The compound system can be analyzed as a multi-dimensional space, for which thermodynamics and statistical mechanics provide several methods of analysis. Hence, the model can be applied for the multi-capability case.

As in the single capability case, the virtual motion of agents will result from the forces that are derived from the potential functions of the modeling particles. However, in the extended case the forces will be calculated in each space separately, and the derived virtual motion will be calculated with respect to the superposition of these forces. Thus, clusters in which all of the particles

---

anisms for team formation.

(or at least most of them) lead to collision will model agents which all of their capabilities (or most of them – in a case that partial goal satisfaction is allowed) fit the required capabilities for the satisfaction of a specific task, and their (multi-dimensional) collision will model the goal satisfaction.

### 6.2 Imprecise information

Since we assume that the information gathered about the adjacent entities is a result of the agents' perception, we must take into consideration the noisy nature of information gathered via sensors [48]. This means that not only is the information incomplete, as we assume in the model (section 10), it is imprecise as well. Therefore, we shall examine the validity of the model behaviour that is subject to such an imprecision. To study this issue, we approach the field of thermodynamics and statistical mechanics [43]. The thermodynamical behaviour of large-scale particle systems, and the corresponding behaviour of smaller simulated systems [49], in the case of bounded (closed) systems [49], is stable. Introducing random low level perturbations into such systems does not significantly affect their overall behaviour (as discussed in, e.g., [27, pages 122–148]). This is because the thermodynamic parameters of the systems do not vary significantly [50]. Accordingly, we conclude that our MAS will be stable, since they are expected to be restricted by time, size, resources, etc. Such stability was evident in the simulations we have performed.

As the noise to signal ratio increases [51], even strictly bounded systems may lose their stability. This loss of stability may occur almost instantaneously, resembling a thermodynamical *phase transition*. However, this may happen only when the ratio is significantly large (greater than $1_{db}$). Relying on the physics properties, we can conclude that our MAS will maintain its stability when exposed to small perturbations, such as that which noisy information may impose.

[48] As previously stated, we extend the meaning of sensor to any information reception device.
[49] Systems with a bounded thermodynamical property are, for instance, systems where the number of particles, the volume, the energy etc. are bounded.
[50] This stability does not always hold for unbounded or loosely bounded systems, where the result of a small perturbation may be chaotic.
[51] The ratio between noise and signal is a common method to measure the quality of systems where information signals are present. The ratio of 10% or one decibel (db) is considered a threshold value.

## 7 Simulation

To examine our model and show its applicability to real problems, and in particular to the transportation case presented in section 5.2, we have performed a set of simulations. Via these we demonstrate effective task allocation and execution in an open, dynamic MAS that consists of thousands of agents and tasks. The problem domain for which the simulations where performed is as follows. We simulate freight deliveries within a metropolitan. Such problems in non-computational environments are commonly solved by having one or a few dispatch centers to which delivery requests are addressed and these each plans and accordingly allocates delivery tasks to delivering agents [52]. This method may face bottlenecks and inefficiency when a large, dynamically changing set of agents and tasks is present. We demonstrate how the PAS model can overcome this limitation.

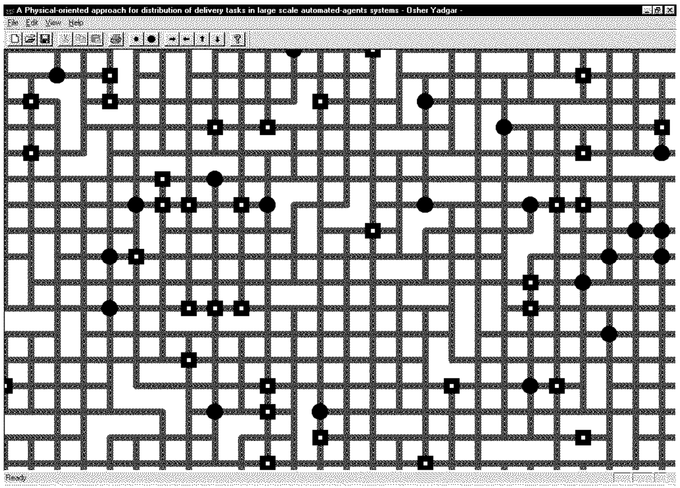


Fig. 8. A fragment of city map

We consider the road-network of a large metropolitan. A snapshot of a part of this network is depicted in figure 8. In this figure squares represent tasks and circles represent messengers (taken from a simulation of 600 messengers and 1200 freights, randomly distributed). The city map is represented by a lattice-like graph. The boundaries of the city are $20,000 \times 30,000$ meters. The lattice includes vertices located 200 meters apart from each other. An edge may exist between each two neighboring vertices. Each vertex represents a junction and each edge represents a road between two junctions. We designate the map "Full Lattice" when each vertex has edges emanating to all of its neighboring vertices. A more realistic map would have some of the edges missing. To obtain such a map we use some probability to determine the existence of each edge. As a result disconnected sub-graphs (designated clusters) may occur. In such cases the largest cluster will be selected to represent the city. We designate the

[52] Note that a similar approach was used also in MAS [47].

map "X% Lattice" when lattice and cluster generation are performed taking the probability of including an edge in the lattice to X%.

In the PAS model, the typical potential of a particle $i$ is the Lennard-Jones potential. When adapting the model to the specific transportation application we experimented with several different potential functions, all of which are sums of derivatives of powers of $r_{ij}$. We checked the effect of modified potential functions on the performance, and after several experiments we finally concentrated on the following:

$$V(\vec{r})_{ij} = \gamma(\alpha \ln r_{ij} + \beta r_{ij}^{-2} + \chi r_{ij}^{-4}) \tag{20}$$

We sought a potential function which is similar in shape and physical properties to the Lennard-Jones potential, and the one above is. It consists of repulsive and attractive components and diminishes after a short distance, however not as short as Lennard-Jones (and by far shorter than a Culombic potential), thus implying that the interaction between the particles in the simulated system is similar to the one in the original PAS.

We have performed several different types of simulations. These varied over the numbers of tasks and agents involved, the homogeneity of agents and tasks, the reliability of communication, the intensity of the lattice map and the distribution of agents and tasks over the city map. The simulations consist of iterations in which new freights dynamically appear at random locations on the map. Messengers (agents) follow our algorithm to perform tasks of reaching freights and delivering them to their destination. Initially, simulations were performed such that agents and tasks are homogeneous in the sense that they have similar capabilities and capacities. We started with these since they are simpler to handle and predict. However it was necessary to examine cases in which agents and tasks are not homogeneous, which are more realistic. In the homogeneous case, masses of particles were set to $1kg$, whereas in the heterogeneous case masses where set randomly out of a given distribution. We have also examined several lattice maps, starting from a full lattice and moving to 90% and 80% lattice maps. Since we have seen no significant difference in the performance between the different maps, we concentrated on the 90% lattice map. We did not test highly disconnected maps, since they represent as class of problems where the solution search space is significantly trimmed, and traditional optimization mechanisms can be exploited instead of the PAS. To learn the effect of unreliable communication on the performance we have experimented with a case in which messages are passed with arrival probability which is smaller than 1. Additional parameters of the simulations are as follows. During the simulation no new messengers appear. Parameter values are $\gamma = 1$, $\alpha = 4000$, $\beta = -15 \times 10^5$, $\chi = 5 \times 10^11$ (in equation 20), $R_0$ is 100 meters, $R_I$ is 2,000 meters.

Throughout the simulations we have examined various numbers of agents.

We allowed agents to appear and disappear dynamically, keeping the average number (in each case separately) constant. In the homogeneous case, we considered five settings of agent and task quantities. From among these, in the first 4 simulation settings the number of agents was 300, 400, 600 and 800, and the initial number of tasks was 1200. In the fifth case, the number of agents was 1200 and the initial number of tasks was 1500. In all 5 homogeneous settings additional tasks where arriving at a rate of 600 tasks per hour. The different quantities of agents in the first four settings allowed us to study the effect of the number of messengers (hence the messengers/freights ratio as well) on the system's performance. The first 4 settings were also experimented with in the heterogeneous case. The fifth setting was aimed mainly at studying the effects of up-scaling, and was not experimented with in the heterogeneous case. The simulations runs were of lengths of more the 70 or more. When calculating averages over a simulation run, we omitted the results of the first 10 hours. Since the system always reaches a stable stable within less than 10 hours, this omission prevents the averages from being affected by the initial conditions. The main results of the simulations are summarized in the following graphs.

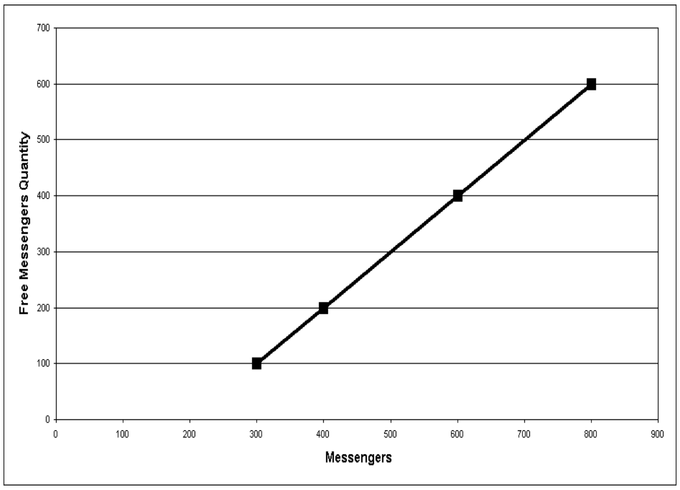



Fig. 9. The number of messengers moving towards freights (y axis) as a function of the number of messengers in the system (x axis). Other agents perform tasks concurrently.

In figure 9 the ratio between the number of messengers in the system and the number of agents that are simultaneously involved in movement towards tasks is presented. The term *Free messenger quantity* is the number of messengers which are currently moving towards freights or searching for them. The other messengers are performing tasks. From the graph one can observe that as the number of messengers involved increases, so does linearly increases the number of those that simultaneously move towards tasks. This result for itself does not seem of merit, however it results in reduction in the time required for task execution (as can be seen in figure 12).

The term *Freight quantity* in figure 10 is the number of freights currently

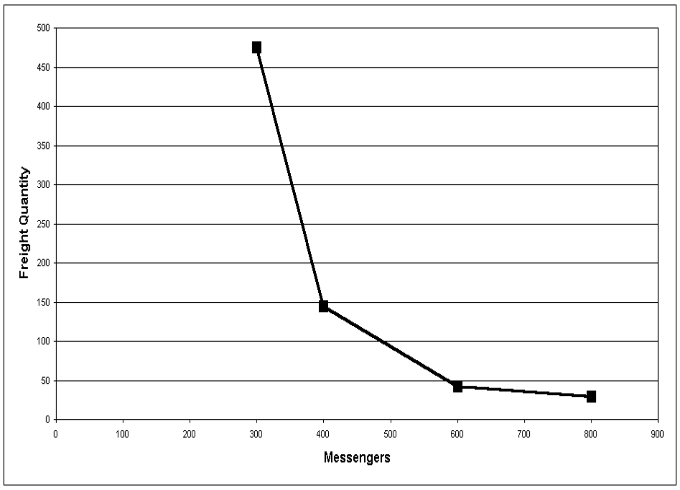



Fig. 10. The number of freights waiting for delivery (y axis) decreases sharply as the number of agents in the system (x axis) increases.

waiting for a messenger to deliver them. We observe that this number drops sharply as the quantity of messengers goes up. The critical point where transition occurs is around 500 messengers. Given that 1200 tasks are present, this means that for significantly lowering the number of freights which are simultaneously waiting to be delivered it is enough to have a ratio of around 0.4 (500/1200) between messengers' and tasks' quantities in the system. Increasing the ratio over 0.5 (600/1200) does not bring about a significant increase in the performance (with respect to the numbers of freights waiting to be delivered).

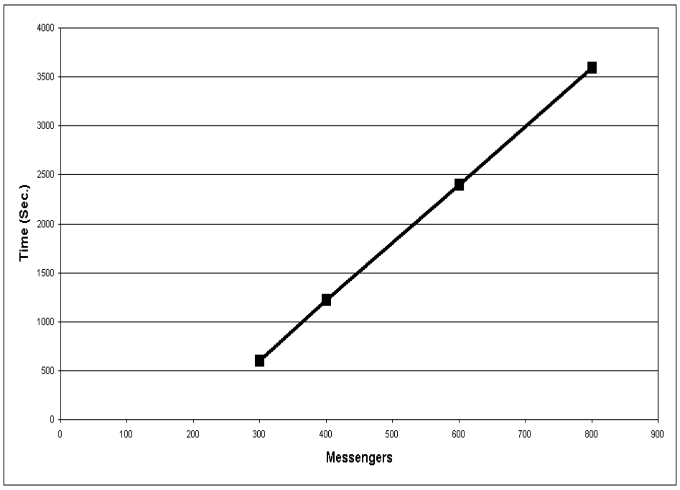



Fig. 11. The time for an agent to reach a task (y axis) increases as the number and density of agents in the system (x axis) increases.

Figure 11 presents the time [53] it takes a messenger, who successfully delivers a freight to its destination, to reach this freight. One can observe that as the

[53] Here and in the following graphs time is measured in seconds.

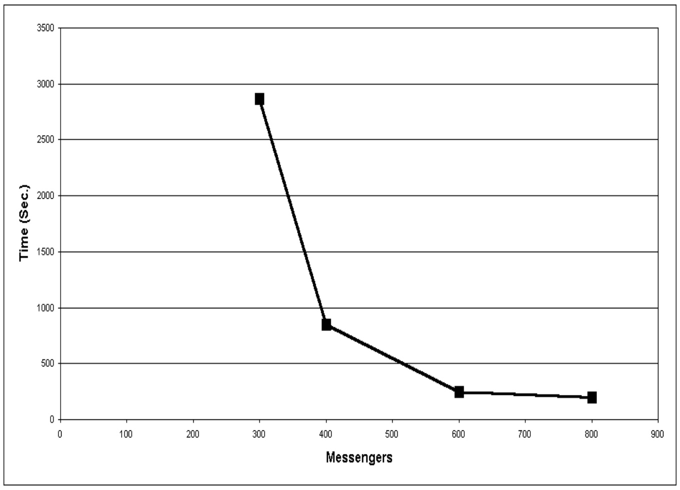



Fig. 12. The average time freights wait for delivery (y axis) decreases sharply as a function of the number of agents in the system (x axis).

quantity of messengers increases (and so does their density), the average time required for a messenger to reach a freight increases as well. This results from more messengers per freight. Given the physics-base behavior of the system, there will be more mutual rejections, so on average a messenger will need more time to reach a freight. This is a disadvantageous property, although it does not mean that increasing the density is all bad. As we have seen before— it significantly reduces the number of freights which simultaneously wait for being delivered. In addition, as shown in figure 12, the average waiting time of the freights decreases as well.

In figure 12 the freight average waiting time, that is, the time that a freight that was successfully delivered to its destination has been waiting before being handled by a messenger is presented. A sharp reduction in the waiting time is observed. We observe phase transition around 500 messengers, similar to the phase transition in the case of *Freight quantity* (figure 10). This further supports the observation that it is not worth while to increase the agent/task ratio to above some ratio which is, in our simulation settings, around 0.4 to 0.5.

Figure 13 presents the average freight fulfillment time, which is the time between the freight initiation and its arrival at its destination. This time subsumes the waiting time and adds to it the execution time. Less steep than in previous graphs, yet clear, is the improvement in the performance reached around 500 messengers. It is important to notice that for 600 messengers and more (and 1200 initial tasks) the overall task execution time is less then 1500 seconds. For a city of the size with which we deal ($20 \times 30$ km) with a speed limit of 50km/hr, this is a desirable fulfillment time.

Figure 14 presents the results of simulations where the probability of message

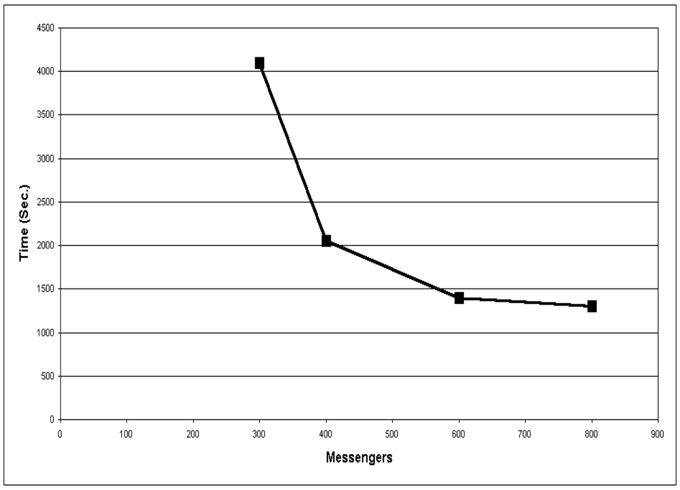



Fig. 13. The average time for task allocation and execution (y axis) decreases as the number of agents in the system (x axis) increases.

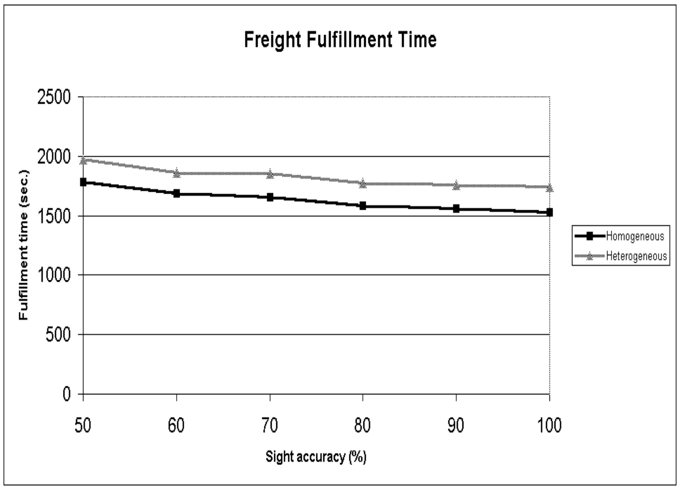



Fig. 14. The average time for task allocation and execution (y axis) decreases as the accuracy of message reception (x axis) increases.

reception varies between 50% and 100%. The simulations were performed for both homogeneous and heterogeneous ensembles of agents and tasks. In these simulations there is a non-negative probability of an agent not receiving information regarding neighboring tasks and agents, although this information was transmitted. The number of agents in both of these sets of simulations was 600 and the initial number of tasks was 1200. The other parameters were as in the previous simulations reported above, except for masses in the heterogeneous case. In that case, the initial masses of tasks were set randomly, out of a uniform distribution, between $1kg$ and $100kg$, while the masses of the agents were set randomly (uniformly distributed) between $80kg$ and $180kg$. If the capacity of an agent was smaller than the size of the task, it delivered only part of the task at a time. The freight fulfillment time in the heterogeneous case is the time it takes a whole freight to arrive at its destination.

From figure 14 we can conclude that the freight fulfillment time increases linearly when the probability of messages arrival decreases. However, even when only 50% of messages arriving, the fulfillment time is better than in the case of 400 messengers with 100% message arrival, as seen in Figure 13 (but, of course, worse than in the case 600 messengers there). Our results indicate that unreliable communication has a limited effect on the time required for task performance. Similar observations apply to other measurements performed with heterogeneity and unreliable communication.

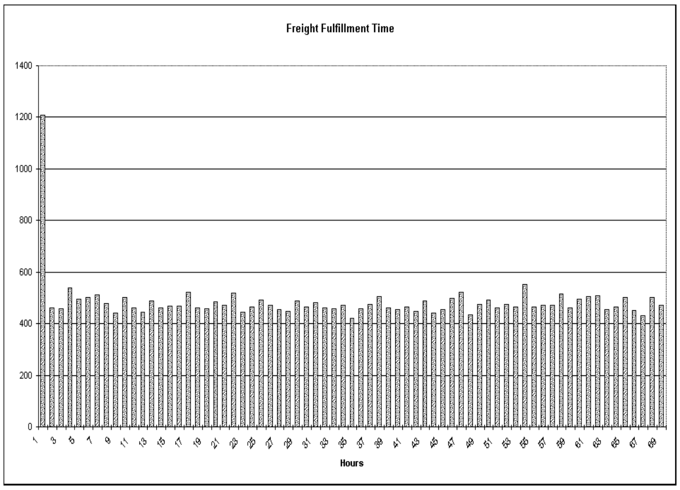



Fig. 15. The average time for task performance (y axis) in the case of agents and tasks *normally* distributed around the center of the city. The execution time decreased significantly compared to an even distribution (except for the first hour of the simulation).

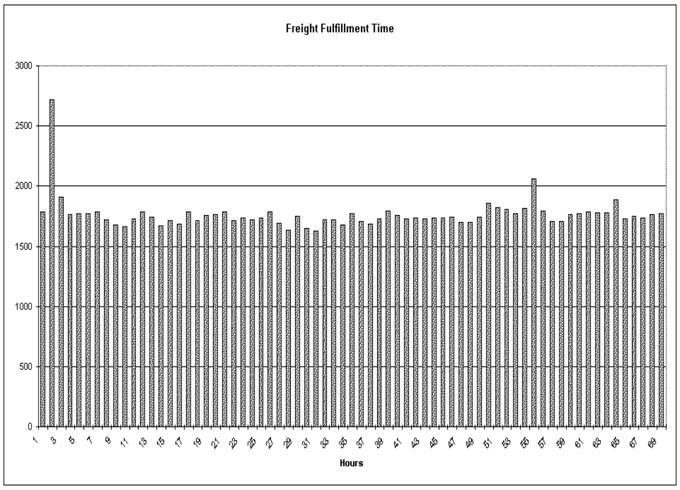



Fig. 16. The average time for task performance (y axis) in the case of agents and tasks *randomly* distributed, with settings similar to the normal distribution case (for comparison).

The previous graphs present results of simulations where tasks and agents are randomly distributed over the city map. In figure 15, where task performance time in seconds ($y$ axis) is measured as a function of the simulation time in hours ($x$ axis), we present the results of a case where the distribution is uneven. Agents and tasks are located according to a two-dimensional normal distribution. The center of the distribution function is the center of the city map (that is, $x = 10000_{mr}$ and $y = 15000_{mr}$), and the standard deviation is $4000_{mr}$. Such distribution means that 2/3 of the tasks and agents are concentrated on 1/4 of the area of the city [54]. The result, as can be observed from the graph, is a very significant improvement in the average task fulfillment time. On the one hand, this should be expected, since the average distance between messengers and freights has decreased too. On the other hand, one may expect localized concentrations of agents and tasks to result in more conflicts, which in turn may reduce performance. For several local densities we examined, this is not the case. Hence, the PAS is applicable to non-uniform distributions as well. The results in figure 15 should be compared to those in figure 16. There, results of simulation with the same settings are presented, except for the distribution, which is random instead of normal. In both figures 15 and 16 variable masses were implemented.

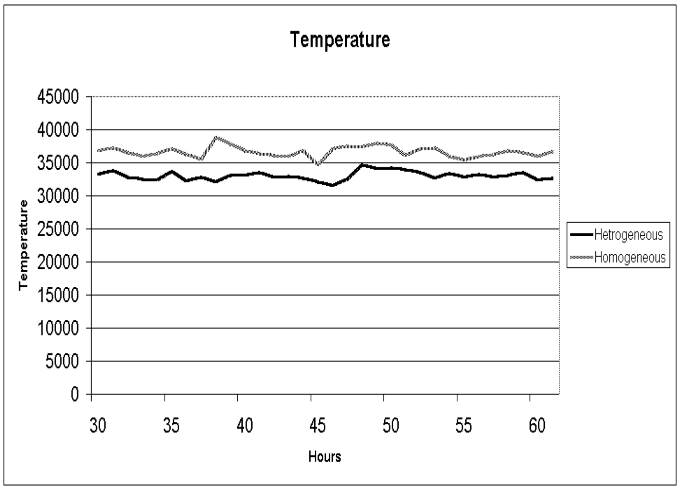



Fig. 17. The temperature (values not calibrated to metric scale) of the system (y axis) as a function of simulation time (x axis) shows to be moderate and stable. This graph demonstrates it for the case of agents and tasks randomly distributed with variable masses. Similar results were reached in all other cases.

In the simulations, we have added a temperature monitoring and control mechanism. Temperature is a parameter that is measured statistically over time, based on velocities of particles. A very low temperature usually results in the system converging to local minima, whereas very high temperatures may result in "hyper-active" agents, which may in turn cause inefficient task allocation and performance. The mechanism we provided periodically computes the temperature and if necessary slightly corrects it. This correction is performed by computing the change required in the average velocity (for reaching

[54] Such distributions are common in cities, where most of the activity may be concentrated around their center.

the desired temperature) and adding it to or subtracting it from the velocities of all agents. We found out that when the parameters were adequately set, corrections were hardly necessary, temperatures in all experiments were moderate and stable, as in figure 17. From these results, and from having all tasks performed, we can conclude that local minima, if reached at all, are scarce.

From the results presented above as well as myriad additional experiments we conclude the following:

- The PAS model can be applied for use in large scale agent systems to solve real problems.
- An increase in the number of agents in the system does not increase the amount of computations per agent. Thus, larger systems do not require more computation time.
- An increase in the number of agents in the system, holding the number of tasks constant, is beneficial only to some extent. Beyond some agents/tasks ratio, no significant improvement in performance is observed. We believe this phenomenon results from redundancy in densely populated agent systems.
- The results observed are similar for different densities of the lattice map used as well as for low probabilities of unreliable communication channels.
- The PAS is applicable for varying, non-uniform distributions of masses and locations of particles (and, respectively, agents and tasks).

## 8 Related work

A large body of DAI research studies coordination among agents for distributed problem solving (for example [13], PGP [16], GPGP [9,17,63]). In [12], Durfee and Lesser study their Partial Global Planning (PGP) approach to coordination by implementing it in the Distributed Vehicle Monitoring Testbed (DVMT). The DVMT is a network of vehicle monitoring nodes. Each node has a planner that plans incrementally. Nodes do not communicate their detailed actions, but do communicate according to a meta-level organization. A PGPlanner modifies local plans as required due to incoming messages. In its incremental planning and restricted communication the PGP model is similar to our model. The DVMT task domain (which was used as a testbed for PGP and GPGP) includes traffic monitoring. This is performed by the agents generating tentative maps for vehicle movements in their areas. Our transportation framework is different: we require that a transportation task be attached to agents that plan for it and perform it. Therefore, our simulated transportation system is significantly different from DVMT.

The majority of the simulations performed by Lesser and his colleagues were in environments that include only few agents (in many cases 4 agents were considered, and the maximal number of agents which they tested was, to our knowledge, 25 agents). This was reasonable for the problems which they have considered, where there is a "natural" geographical distribution of the agents, and the problems of integration of sub problems arise between closely related agents. In dynamic environments where different agents may have more freedom to move (either physically or abstractly), the problem of scaling up is more significant. Nevertheless, Lesser et al also considered problems of scale up and developed methods to reduce the communication needed to reach globally consistent solutions. For example, the GPGP model [8] extended the PGP ideas by allowing more agent heterogeneity, the exchange of more truly partially global information at multiple levels of abstraction and the use of separate scheduling algorithms. However, GPGP provides a meta-level for coordinated planning and is not a coordination algorithm in itself.

Transportation problems were discussed in DAI research previously e.g., in [19,47]. Another example is the DVMT as mentioned above. In [47] self-interested agents are dealt with, whereas we discuss the case of cooperative agents. Another significant difference is the size of the problem domain. Sandholm [47] provides a solution for only few agents, albeit dozens of carriers that work on their behalf and hundreds of deliveries. Thus, the multi-agent problem they solve is of small magnitude whereas the scheduling problems associated with it are rather large. The transportation framework with which they deal is comprised of few dispatch centers which are the agents and dozens of carriers that move freights from these centers, possibly sharing tasks. Task allocation is performed by the dispatch centers and not by the carriers.

Fischer et al [19] assume few dispatch centers as well. However, in their solution each dispatch center has trucks which are each an autonomous agent. Agents are cooperative within their company (the dispatch center) however competitive with regards to other agents. Yet somewhat centralized, all deliveries can only start from dispatch centers, and all of the information with regards to deliveries if forwarded to truck agents from these centers. In our physics-based model no task forwarding is present (or necessary). Note that Fischer et al assume a dynamic system, however admit to have carried out experiments only with respect to a static one. Another difference of their system from ours is the small size of the system. They have simulated up to 3 dispatch center, each having up to 20 truck agents.

Distributed problem solving for large scale problems (where the size is expressed by the number of variables involved) was presented in [62], where Yokoo presents the weak-commitment algorithm for distributed constraint satisfaction. The presented approach is significantly different from ours though it permits scaling-up. The algorithm is not applied to dynamic task allocation and execution among agents as our algorithm does, however this seems to be a possible extension of it. However, although it proves to work efficiently for large scale problems (e.g., 1100 boolean variables), the algorithm has some undesirable properties, such as abandoning partial solutions which prove inconsistent and, (probably resulting from the latter), an exponential complexity of the worst case. Even in the average case the amount of computations increases with the size of the problem solved. In the model we propose the average complexity for large systems is constant. In addition, instead of erasing them and starting from scratch, partial solutions evolve via frequent corrections, thus they do not reach a state of being completely inconsistent with constraints. This may, however, limit the solution we provide from being implemented in some domains. Note that the weak-commitment algorithm concentrates on constraint satisfaction, while our approach discusses task allocation. Nonetheless, the physics-based approach we present may provide solutions to some constraint satisfaction problems, albeit not necessarily to these solved by Yokoo.

The tileworld model [41] was used as a testbed for planning and task allocation and execution in multi-agent systems. The utilization of physics methods allows for a model that is significantly richer than the tileworld model. While the tileworld is a chess-board-like grid with a limited, 4-directional moves, the physics-based model we provide allows, in its very basic physical interpretation, for three dimensional systems with unlimited directional moves. We believe that with some level of abstraction a physics-based model is further more expressive. The tileworld model distinguishes (at least) two different procedures – deliberation and path planning – which are usually performed sequentially, whereas in the physics-based model an inherent property is interleaving planning and execution. And, while the tileworld proves to work successfully for systems of dozens of tasks and agents, (15 agents, 80 tasks in [17]), its computational complexity[55] will probably disable scaling up to thousands of tasks and agents. Such system size is allowed by the physics based model, as simulations prove.

Ephrati, Pollack and Ur [17] suggest the multi-agent filtering strategy as a means for coordination among agents. They have conducted several experiments that show, that for the tile-world, this strategy improves the performance of the agents. This coordination is achieved without explicit negotiation. In our work we do not suggest a strategy, rather we suggest a method for modeling the goal-agent environment. Based upon this model we suggest a detailed algorithm for the single agent for acting efficiently in the environment. We provide an explicit analysis of the quality (with respect to communication and computation consumption) of our algorithm and conditions in which it is most appropriate. In addition, to support the theoretical analysis, we present simulations results. As in Ephrati et al, our model does not require negotia-

[55] As Kinny and Georgeff [31] explicitly say: “to reduce the complexity...we employed a simplified Tileworld with no tiles.”

tion. In addition, the amount of required communication in our model may be significantly smaller than in the filtering methods by Ephrati et al, since in our model agents communicate with a small subset of the whole agent community, whereas there agents presumably communicate with all other agents [56]. This may be of lesser significance when few agents satisfy only few dozens of goals (as in [17]), however is most important for systems of hundreds or thousands of agents and goals, for which we address our solution.

Emergent behavior of computational agents has been discussed in several studies that have been performed in recent years. Glance and Huberman [23] discussed this issue and borrowed methods from statistical thermodynamics in order to study the evolution of social cooperation. In the context of social dilemmas, they used this methodology to study the aggregate behavior of individuals facing social choices. As in our work, they applied results from theoretical physics to study the behavior of a system of individuals. However, unlike us, they used theoretical physics specifically for studying the group properties and not for studying the properties of the single, individual agent. They also did not develop an algorithm for the behavior of such an agent within the group in which it was a member. The main concern of their research was the collective behavior, while we concentrate on the personal behavior, but still discuss the collective behavior which results from this behavior. In another paper [24], Glance and Huberman present a detailed physical formalism of the dynamics of the collective action of a system of individuals. In our work the main issue is the physical behavior of the single agent. We do not use physics in order to analyze existing systems. Rather, we develop an algorithm that is based upon the physical properties, and we rely on the known physical [57] behavior of particles to predict the behavior of the agents that will use the algorithm we have developed.

Shoham and Tennenholtz [53] presented results of simulations that were performed in order to perceive the emergence of conventions in multi-agent systems. They are concerned with the design of multi-agent systems that converge towards common social laws. In our research, though we discuss emergent cooperation, we do not discuss the emergence of the laws according to which the cooperation occurs. Rather, we determine the social laws to be such (physical laws) that they will cause the emergent cooperation of the system when this cooperation is necessary.

The pursuit problem [1] has be widely used as an example problem for multi-agent coordination and cooperation. The problem is of several predator agents

[56] This issue is not explicitly addressed in the paper, but one can conclude it from the information available to an agent about the others.

[57] A variety of computer science problems, in general and AI, in particular have been described and solved by physics-oriented models (e.g., [64]). However, we avoid the description of these studies because we do not find them similar enough to our case.

attempting to cooperatively hunt a moving prey. Its uniqueness is in the necessity of coordination among the predators as well as the dynamic change in the goal location. As Ishida and Korf state [26], off-line algorithms that compute the entire solution will fail to provide an appropriate answer due to the dynamism of the problem. They devise a real-time algorithm, the *moving-target search* algorithm (MTS), to overcome these limitations. They describe the problem by a connected graph where each setting of the agents on the graph is a unique state. The complexity of the algorithm MTS is $O(N^3)$, where $N$ is the number of states. When considering multiple agents on a large graph, this complexity prohibits feasible solutions. Another algorithm, based on Q-learning, was suggested in [40]. There, more agents were involved in the solution however their number was only 4 (hunters) + 1 (prey) agents. The number of trails to reach a solution was reduced by the learning mechanism however is still relatively high. The algorithms used for solving the multi-agent pursuit problem are to complex for problems that consist of thousands of agents and a large scale problem space. Our research is aimed at such large scale systems, and may be found inadequate for too small systems. It must be noted that in our work we do not explicitly discuss cases in which the goals change their locations dynamically [58].

Another approach to describing group behavior is presented by Mataric. In her paper [38], she proposes the definition of a set of basic interactions that will allow the simplification of analysis of group behavior. However, Mataric's approach differs from ours. While she discusses the description and the synthesis of group behavior, she does not provide an explicit set of basic interactions. In our work, we concentrate on the nature of the basic interactions and adopt the physical interactions among particles to model the interactions among agents and goals.

Due to the rapid improvement in the micro-mechanics and microprocessor technologies, a real environment of many simple microscopic autonomous robots is becoming possible. As a result of this progress, as described in the paper of Gage [21], a simple emergent cooperation method among the agents is necessary. In his work, Gage defined a number of specific classes of desired mobility behaviors for use in military scenarios. He also provided a list of topics that should be considered by designers of such systems, and presented results of simulations which he performed to illustrate the behavior of such systems. Gage's research differs from ours in several aspects: while he concentrates on specific classes of motion, we do not restrict our model to either specific classes or exclusively to motion. Gage proves the validity of his methods by performing simulations, whereas we rely on theoretical and experimental physics for

[58] Such dynamics, however, are encapsulated in the physical model, and only marginal modifications will be required to adjust our model the case of moving goals.

proving validity.

The path-planning and robot-navigation (PPRN) research [59] have used potential fields as a means for planning the path, e.g., in [11,29,30,59]. This approach appears to be closely related to our research, as we, too, use potential functions. However, there are several significant differences between the path-planning and robot-navigation problem and our task-allocation and agent-coordination problem, and the techniques that are used for solving these problems. The differences are as follows:

(i) In PPRN research, the main objective is planning an optimal path for the robots to navigate from an initial location to its destination. Our main objective however is to solve a task allocation problem with aspects of agent-coordination in a multi-agent environment.

(ii) While we discuss the multi-agent case, with potentially thousands of agents, the type of planning research that is involved with potential functions usually discusses the single robot case. In cases where the more-than-one robot case is discussed, the number of robots is considered very small as opposed to the MAS we discuss.

(iii) In cases where the PPRN research addresses the multi-robot planning problem, e.g., in [3,21,38,58], the potential field concept is not employed. In such cases the behavior of groups and formations of robots are discussed, given a set of specific strategies according to which the robots act. Among these strategies you may find some in which robots follow a leader or some predefined geometric patterns. In our multi-agent model however the agents' strategy is based on the physical potential-well concept.

(iv) Although both the PPRN and our research use potential fields for problem representation and resolution, the type of potential functions and the way of using them is different, as follows:
   - The potential functions employed for path-planning are artificial. The only expected result from such potential functions is robot-motion according to the potential field gradient. In our research, we employ physics potential functions. This results in a physics-like behavior of the agents that act with respect to these functions. In particular, the use of the potential functions of particles in a non-ionic fluid results in a model that provides the single agent with an algorithm for reaching and performing goals within a large community of agents. In addition, the use of such potential functions enables the prediction of the bulk properties (i.e., the behavior) of the MAS as a whole.
   - In the PPRN research, an attractive potential field (usually quadratic) is employed to lead the robot to the goal, and various shapes of repulsive potential fields are used to cause obstacle-avoidance. In our work, all

[59] A comprehensive overview of these can be found in [35]

of the entities –agents as well as goals– (and obstacles, if present) are modeled by the same type of potential function; i.e., by a physical potential-well.

– The potential-wells in our model may change dynamically due to the fulfillment of goals and the expenditure of resources. This leads to a dynamically alternating potential field which results in a dynamic update of the agents' behavior. Dynamics of the potential field in PPRN research, when present, refer only to the change in the locations of robots and obstacles, and not to a change in the specific function that models a specific entity.

(v) Another important difference of our research, as compared to the path-planning research, is that we do not restrict the model to physical trajectories – the model can be used for abstract motion[60].

(vi) There are cases in which PPRN employ physics-like concepts and analysis methods (e.g., in [11]). Nonetheless, this selection is not based on a model of an existing, large-scale, physical system from which properties can be inferred, as opposed to our our model selection. However, this artificial choice was later proved, with several restrictions, to posses of good properties such as polynomial complexity and near-optimal trajectories[61] [10].

In summary, the PPRN research with artificial potential functions discusses cases where a single robot or a small number of robots must navigate and locate their goals. Our approach is very different: we discuss cases of large-scale MAS, with many agents involved; cases where a specific agent does not have a specific goal towards which it must navigate; cases of cooperative goal satisfaction. These are not the aim of the PPRN research and therefore are not discussed in it.

The issue of allocating agents to goals has widely been discussed among DAI researchers. A well-known model is the Contract Net Protocol [54]. The CNP uses negotiation based on task announcements, bids and contracts for task allocation. The net consists of dynamically alternating *worker* and *manager* nodes, and they exchange information about goals to be performed and subgoals that were already processed. The CNP was not designed based on an existing model, as is our model, therefore the performance of the system is checked only by simulations. In our model, the performance of the system is predicted from its physical properties and the efficiency is formally calculated and compared to the optimal results. The model we present allows (but does not require) minimization of the transmitted information[62] and thus enables

[60] The concept of abstract motion shall be explained in the next section.
[61] Note that the latter is appropriate for a single robot trajectories. Our model is meant for multiple agents task allocation.
[62] Note that this minimization refers to the number of recipients of the information and not necessarily to the amount of information transmitted.

large-scale systems to be efficient.

A study of planning in large-scale MAS has been presented by Wellman [61]. In this research, the general-equilibrium approach from economics serves as the theoretical basis for the planning mechanism. Mechanisms in which competition is applied are used to construct a market-oriented programming environment, which is employed as a means for the construction and analysis of distributed planning systems. In his work, Wellman performed simulations to derive system equilibrium [63], and showed that given the appropriate restrictions, the model reaches near-optimal results (small deviations from the optimal results are described in [61]). As done by Wellman, we also discuss large-scale systems. We, too, apply an analytical model for designing the distributed planning mechanism. However, the use of the physics-oriented approach allows us to predict the resulting behavior of the system and that of its constituent agents based on the known behavior of physical systems. Moreover, while Wellman performed simulations that included 3 to 20 agents [60], (which we view as a comparatively small MAS), we performed simulations with the number of agents exceeding 1000. Another major difference is the type of systems for which the models are appropriate. While Wellman's model is most appropriate for self-interested agents, our model was designed for DPS systems, where the agents try to increase the overall outcome of the system.

## 9 Conclusion

The design and analysis of large-scale agent systems imposes difficulties that are hard to solve even when the proposed solutions are of low-order polynomial complexity. In this paper some aspects of this problem are addressed. Namely, we provide a method for task allocation and execution in several classes of large-scale cooperative MAS. We present a physics-oriented approach that results in a very low complexity on the part of the single agent and may even be of order $O(1)$. Such results are possible since we use a model whose behavior is already known. Therefore, we are not required to perform the numerous explicit calculations that would have otherwise been necessary.

The model we have presented and the algorithm that enables the single agent to act according to the model consist of methods with which the agents allocate themselves to goals in order to satisfy the goals. The agent-goal matching is an emergent result of the physics-oriented behavior of the agents. According to our model, each agent is most strongly attracted to the goal that it will satisfy with the best fit (within a limited range). In addition, in cases where too many agents fit the requirements of the same goal, our model will disenable

[63] The definition is provided there.

some of them from reaching the goal, via the property of mutual rejection. As we have shown, the algorithm that we provide leads to agent-goal allocation, it converges to a solution, the computational complexity is low and communication, if necessary, is of a small amount. Our method does not lead to the optimal goal-agent allocation, but reaching an optimal allocation requires complete on-line information about all of the agents and goals comprising the system and, for a large class of problems, an exponential computation-time.

The physics-oriented approach which we present has several advantages. While common DAI algorithms must be checked for their validity either by a formal proof or by simulations, our model can rely on theoretical and experimental results that are already known from physics. According to these results, we can predict the evolution of the modeled MAS, since it will evolve in the same manner as a corresponding physical system. The local interactions, which enable one to derive the global behavior of the system, assure a low computational complexity of the model. In very large-scale MAS, this approach provides a model that promises emergent cooperative goal-satisfaction activity. As we have shown, these properties proved to hold in a simulated system. In addition, the properties of the system as a whole can be analyzed using concepts from statistical mechanics [64]. The employment of such concepts enables the derivation of the bulk properties of a system via the properties of its components. We leave this analysis for future work.

## References


[1] M. Benda, V. Jagannathan, and R. Dodhiawalla. On optimal cooperation of knowledge sources. Technical Report BCS-G2010-28, Boeing AI Center, 1985.

[2] S. Cammarata, D. McArthur, and R. Steeb. Strategies of cooperation in distributed problem solving. In *Proceedings of IJCAI-83*, pages 767–770, 1983.

[3] Q. Chen and J. Y. S. Luh. Coordination and control of a group of mobile robots. In *Proc. of the 1994 IEEE international conference on Robotics and Automation*, pages 2315–2320, 1994.

[4] R. F. Churchhouse. *Handbook of Appplicable Mathematics – Numerical Methods*. J. Wiley, 1981.

[5] S. E. Conry, R. A. Meyer, and V. R. Lesser. Multistage negotiation in distributed planning. In A. H. Bond and L. Gasser, editors, *Readings*


[64] The analysis of the bulk properties of particle-systems via statistical mechanics methods can be found in many introductory books, e.g., [43]. We demonstrate the usefulness of such methods when measuring the stability of our system in terms of its temperature.

*in Distributed Artificial Intelligence*, pages 367–384. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1988.

[6] R. Conte, M. Miceli, and C. Castelfranchi. Limits and levels of cooperation: Disentangling various types of prosocial interaction. In Y. Demazeau and J. P. Muller, editors, *Decentralized A.I. - 2*, pages 147–157. Elsevier Science Publishers, 1991.

[7] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[8] Keith Decker and Victor Lesser. Generalizing the partial global planning algorithm. *International Journal of Intelligent Cooperative Information Systems*, 1(2):319–346.

[9] Keith Decker and Victor Lesser. Designing a family of coordination algorithms. In Victor Lesser, editor, *Proceedings of the First International Conference on Multi-Agent Systems*, pages 73–80, San Francisco, CA, 1995. MIT Press. Longer version available as UMass CS-TR 94–14.

[10] B. Donald and P. Xavier. Provably good approximation algorithms for optimal kinodynamics planning: Robots with decoupled dynamic bounds. *Algorithmica*, 14(6):443–479, 1995.

[11] B. Donald, P. Xavier, J. Canny, and J. Reif. Kinodynamics motion planning. *Journal of the ACM*, 40(5):1048–1066, November 1993.

[12] E. Durfee and V. Lesser. Predictability vs. responsiveness: Coordinating problem solvers in dynamic domains. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 66–71, St. Paul, MN, August 1988.

[13] E. H. Durfee. *Coordination of Distributed Problem Solvers*. Kluwer Academic Publishers, Boston, 1988.

[14] E. H. Durfee and V. R. Lesser. Negotiating Task Decomposition and Allocation Using Partial Global Planning. In L. Gasser and M. N. Huhns, editors, *Distributed Artificial Intelligence, Volume II*, pages 229–244. Pitman/Morgan Kaufmann, London, 1989.

[15] E. H. Durfee, V.R. Lesser, and D. D. Corkill. Coherent cooperation among communicating problem solvers. *IEEE Transactions on Computers*, 36(C):1275–1291, 1987.

[16] Edmund H. Durfee and Victor R. Lesser. Partial global planning: A coordination framework for distributed hypothesis formation. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5):1167–1183, September 1991. (Special Issue on Distributed Sensor Networks).

[17] E. Ephrati, M. Pollack, and S. Ur. Deriving multi-agent coordination through filtering strategies. In *IJCAI95*, pages 679–685, Montreal, Quebec, Canada, 1995.

[18] K. Erol. *Hierarchical Task Network Planning: Formalization, Analysis and Implementation.* PhD thesis, University of Maryland, 1995.

[19] B. K. Fischer, J. P. Muller, and M. Pischel. A model for cooperative transportation scheduling. In *Proc. of ICMAS-95*, pages 109–116, San Francisco, California, 1995.

[20] K. Fischer and J. P. Muller. A decision-theoretic model for cooperative transportation scheduling. In W. Van de Velde and J. W. Perram, editors, *Lecture Notes in Artificial Intelligence No. 1038, Agents Breaking Away*, pages 177–189. Berlin:Springer-Verlag, 1996.

[21] D. W. Gage. Command control for many-robot systems. *Unmanned systems*, Fall:28–34, 1992.

[22] R. Giles and P. Tamayo. A parallel scalable approach to short-range molecular dynamics on the cm-5. Technical Report TMC-234, Thinking Machines Corporation, 1992.

[23] N. S. Glance and B. A. Huberman. Organizational fluidity and sustainable cooperation. In *Proc. of MAAMAW-93*, Neuchâtel, 1993.

[24] N. S. Glance and B. A. Huberman. The outbreak of cooperation. *Journal of Mathematical Sociology*, 17(4):281–302, 1993.

[25] P. Gmytrasiewicz, E. Durfee, and D. Wehe. The utility of communication in coordinating intelligent agents. In *Proc. of AAAI-91*, pages 166–172, California, 1991.

[26] T. Ishida and R. Korf. Moving target search: A real-time search for changing goals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6):609–619, June 1995.

[27] A. Katz. *Principles of statistical mechanics.* W. H. Freeman and Company, 1967.

[28] S. P. Ketchpel. Forming coalitions in the face of uncertain rewards. In *Proc. of AAAI94*, pages 414–419, Seattle, Washington, 1994.

[29] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98, 1986.

[30] P. Khosla and R. Volpe. Superquadratic artificial potentials for obstacle avoidance and approach. In *Proc. of the 1989 IEEE international conference on Robotics and Automation*, pages 1778—1784, Philadelphia, PA, 1988.

[31] D. N. Kinny and M. P. Georgeff. Commitment and effectiveness of situated agents. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 82–88, Sydney, Australia, August 1991.

[32] J. Koplik, J. R. Banavar, and J. F. Willemsen. Molecular dynamics of Poiseuille flow and moving contact lines. *Physical review letters*, 60(13):1282–1285, 1988.

[33] S. Kraus and J. Wilkenfeld. The function of time in cooperative negotiations. In *Proc. of AAAI-91*, pages 179–184, California, 1991.

[34] L. D. Landau. *Mechanics and electrodynamics.* Pergamon Press, 1972.

[35] J. C. Latombe. *Robot Motion Planning.* Kluwler Academic Publishers, 1991.

[36] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, editors. *The traveling salesman problem.* John Wiley & sons, 1983.

[37] V.R. Lesser. A retrospective view of fa/c distributed problem solving. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1347–1362, 1991.

[38] M. J. Mataric. Kin recognition, similarity, and group behavior. In *Proc. of the 15th Cognitive Science Society Conference*, pages 705–710, Boulder, Colorado, 1993.

[39] T. Mullen and M. Wellman. A simple computational market for network information services. In *Proc. of the First International Conference on Multiagent Systems*, pages 283–289, California, USA, 1995.

[40] N. Ono and K. Fukumoto. Multi-agent reinforcement learning: A modular approach. In *Proc. of ICMAS-96*, pages 252–258, Kyoto, Japan, 1996.

[41] M. E. Pollack and M. Ringuette. Introducing the tileworld: Experimentally evaluating agent architectures. In *Proc. of AAAI90*, pages 183–189, Boston, MA, 1990.

[42] D. C. Rapaport. Large-scale molecular dynamics simulation using vector and parallel computers. *Computer physics reports*, 9(1):1–53, 1988.

[43] F. Reif. *Fundamentals of statistical and thermal physics.* McGraw-Hill Book Company, 1965.

[44] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis. An analysis of several heuristics for traveling salesman problem. *Siam Journal on Computing*, 6:563–581, 1977.

[45] J. Rosenschein. *Rational Interaction: Cooperation Among Intelligent Agents.* PhD thesis, Stanford University, 1986.

[46] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill Book Company, 1983.

[47] T. W. Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proc. of AAAI-93*, pages 256–262, Washington D.C., 1993.

[48] T. W. Sandholm and V. R. Lesser. Coalitions among computationally bounded agents. *Artificial Intelligence*, 94:99–137, 1997.

[49] O. Shehory. Polymer fluid flow simulation using molecular dynamics. MSc thesis, Bar Ilan University, 1992.

[50] O. Shehory and S. Kraus. Cooperation and goal-satisfaction without communication in large-scale agent-systems. In *Proc. of ECAI-96*, pages 544–548, Budapest, Hungary, 1996.

[51] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1-2):165–200, 1998.

[52] O. Shehory, S. Kraus and O. Yadgar. Goal-satisfaction in large-scale agent-systems: a transportation example. In *Proc. of ATAL-98*, Paris, France, 1998.

[53] Y. Shoham and M. Tennenholtz. Emergent conventions in multi-agent systems: initial experimental results and observations. In *Proc. of KR-92*, pages 225–231, 1992.

[54] R. G. Smith. The contract net protocol: high-level communication and control in a distributed problem solver. *IEEE Transaction on Computers*, 29(12):1104–1113, 1980.

[55] K. Sycara, K. Decker, A. Pannu, and M. Williamson. Designing behaviors for information agents. In *Proceeding of Agents-97*, pages 404–412, Los Angeles, 1997.

[56] K. Sycara, K. Decker, A. Pannu, M. Williamson, and D. Zeng. Distributed intelligent agents. *IEEE Expert – Inteligent Systems and Their Applications*, 11(6):36–45, 1996.

[57] C. Trozzi and G. Ciccotti. Stationary nonequilibrium states by molecular dynamics. II. Newton's law. *Physical review A*, 29(2):916–925, 1984.

[58] P. Wang. Navigation strategies for multiple autonomous robots moving in formation. *Journal of Robotic Systems*, 8(2):177–195, 1991.

[59] C. W. Warren. Global path planning using artificial potential fields. In *Proc. of the 1989 IEEE international conference on Robotics and Automation*, pages 316–321, Scottsdale, AZ, 1989.

[60] M. Wellman. Market-oriented programming: Some early lessons. In S. Clearwater, editor, *Market-Based Control: A Paradigm for Distributed Resource Allocation*. 1995.

[61] M. P. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, 1:1–23, 1993.

[62] M. Yokoo. Weak-commitment search fro solving constraint satisfaction problems. In *Proc. of AAAI94*, pages 313–318, Seattle, Washington, 1994.

[63] M. Yokoo, E. Durfee, T. Ishida, and K. Kuwabara. Distributed constraint satisfaction for formalizing distributed problem solving. In *Proceedings of the Twelfth International Conference on Distributed Computing Systems*, pages 614–621, 1992.

[64] A. L. Yuille. Generalized deformable models, statistical physics, and matching problems. *Neural Computation*, 2:1–24, 1990.