A Formal Model of Multi-Agent Computations

Maciej Smołka[1]

Institute of Computer Science, Jagiellonian University, Kraków, Poland
`smolka@ii.uj.edu.pl`

Abstract. The paper contains an extension of a formal model of multi-agent computing system developed in previous publications towards considering a more general system state. We provide also some deeper details of the model in the case of a homogeneous hardware environment. The model provides us with a precise definition of the optimal task scheduling together with results on the existence and characterization of optimal scheduling strategies.

1 Introduction

The application of the multi-agent paradigm together with local, diffusion-based agent scheduling strategies provide us with a relatively simple decentralized management of large-scale distributed computations. Multi-agent systems based on the idea of task diffusion (cf. [1]) dedicated to large-scale computations have been developed for the last several years [2]. A formal model for such systems was introduced in [3] and has been further developed in [4–6]. It provides us with a precise definition of the optimal task scheduling as well as results on the existence of optimal scheduling strategies and optimality conditions. In this paper the model is extended to consider a more general system state. We provide also more detailed modelling of a simple but realistic case of a homogeneous network of identical machines. Finally results on the existence and characterization of optimal scheduling strategies are presented.

2 Architecture of a computing MAS

The principles of the MAS architecture which form the basis for our mathematical model were defined in [2, 7] and has been further developed in [8, 9]. The full description as well as the details of realization and test results can be found therein. Here let us recall only some crucial points.
The suggested architecture of the system is composed of *a computational environment* (MAS platform) and *a computing application* being a collection of mobile agents called *Smart Solid Agents* (SSA). The computational environment is a triple $(\mathbf{N}, B_H, perf)$, where:

$\mathbf{N} = \{P_1, \ldots, P_N\}$, where P_i is *a Virtual Computational Node* (VCN). Each VCN can maintain a number of agents.

B_H is the connection topology $B_H = \{\mathcal{N}_1, \ldots, \mathcal{N}_N\}, \mathcal{N}_i \subset \mathbf{N}$ is an immediate neighborhood of P_i (including P_i as well).

$perf = \{perf_1, \ldots, perf_N\}, perf_i : \mathbb{R}_+ \to \mathbb{R}_+$ is a family of functions where $perf_i$ describes relative performance of VCN P_i with respect to the total memory request M^i_{total} of all agents allocated at the node.

The MAS platform is responsible for maintaining the basic functionalities of the computing agents. Namely it delivers the information about the local load concentration $\mathbf{L}_j$ and Q_j (see (2) and (3) below), it performs agent destruction, partitioning and migration between neighboring VCN's and finally it supports the transparent communication among agents.

We shall denote an SSA by A_i where index i stands for an unambiguous agent identifier. Each A_i contains its computational task and all data necessary for its computations. Every agent is also equipped with a shell which is responsible for the agent logic. At any time A_i is able to denominate the pair (E_i, M_i) where E_i is the estimated remaining computation time measured in units common for all agents of an application and M_i is the agent's RAM requirement in bytes. An agent may undertake autonomously one of the following actions: *continue* executing its internal task, *migrate* to a neighboring VCN or decide to be *partitioned*, which results in creating two child agents $\{A_{i_j} = (T_{i_j}, S_{i_j})\}, j = 1, 2$. We assume that in the case of the agent partitioning the following conditions hold: $E_i > E_{i_j}$, $M_i > M_{i_j}$, $j = 1, 2$. The parent SSA disappears after such a partition.

A computing application may be characterized by the triple $(\mathbf{A}_t, G_t, Sch_t)$, $t \in [0, +\infty)$ where $\mathbf{A}_t$ is the set of application agents active at the time t, G_t is the tree representing the history of agents' partitioning until t. All agents active till t constitute the set of its nodes $\bigcup_{s=0}^{t} \mathbf{A}_t$, while the edges link parent agents to their children. All information on how to rebuild G_t is spread among all agents such that each of them knows only its neighbors in the tree. $\{Sch_t\}_{t \in [0,+\infty)}$ is the family of functions such that $Sch_t : \mathbf{A}_t \to \mathbf{N}$ is the current schedule of application agents among the MAS platform servers. The function is defined by the sets ω_j containing indices of agents allocated on each $P_j \in \mathbf{N}$. Every ω_j is locally stored and managed by P_j. Each server $P_j \in \mathbf{N}$ asks periodically all local agents (allocated on P_j) for their requirements and computes the local load concentration

$$L_j = \frac{E^j_{total}}{perf_j(M^j_{total})} \text{ where } E^j_{total} = \sum_{i \in \omega_j} E_i \text{ and } M^j_{total} = \sum_{i \in \omega_j} M_i \tag{1}$$

Then P_j communicates with neighboring servers and establishes

$$\mathbf{L}_j = \left\{(L_k, E^k_{total}, M^k_{total}, perf_k) : \ P_k \in N_j\right\} \tag{2}$$

as well as the set of node indices

$$Q_j = \{k \neq j : \ P_k \in N_j, \ L_j - L_k > 0\}\,. \tag{3}$$

The mentioned papers as well as [4] describe migration and partitioning strategies which make use of the above-defined quantities. These strategies are related to the physical phenomenon of the molecular diffusion in crystals.

3 System state revisited

Next let us recall the features of a mathematical model of multi-agent computations, which is based on the presented architectural principles. It has already been presented in [3–6]. Here we shall propose a generalization of the model, namely we shall consider a more general form of the system state.

Let us recall that we have considered the set of all possible application's agents $\mathbf{A}$. Because of problems with the determination of such a set the crucial idea is to avoid considering the evolution of a single agent and to study the dynamics of a whole computing application instead. We observe the system state in discrete time moments. Let us recall the notion of the *vector weight of an agent*, which is the mapping $w : \mathbb{N} \times \mathbf{A} \longrightarrow \mathbb{R}_+^M$. Note that otherwise than in our previous papers we allow M to be greater than 2. We assume that the dependency of the total weight of child agents after partition upon their parent's weight before partition is well-known and linear, i.e. there is a matrix $\mathbf{P} \in \mathbb{R}_+^{M\times M}$ such that in the case of partition $A \to \{A_1, A_2\}$ we have

$$w_{t+1}(A_1) + w_{t+1}(A_2) = \mathbf{P}w_t(A). \qquad (4)$$

This time we relax the assumption of componentwise dependency. If this is the case $\mathbf{P}$ is diagonal.
Next let us recall the notion of the *total weight of all agents allocated on a virtual node P at any time t*, i.e.

$$W_t(P) = \textstyle\sum_{Sch_t(A)=P} w_t(A)$$

This notion is crucial in our search for a global description of the system dynamics since it is a global quantity describing the state of the system in such a way that is appropriate for our purposes.
As for the components of w it is straightforward to choose E_i and M_i defined in Sec. 2 as we did in previous papers. Then obviously E^i_{total} and M^i_{total} are the components of W. In general it may be convenient to find other *state variables* (see Sec. 5). In any case both E^i_{total} and M^i_{total} should remain *observables* of our system.
In the sequel we shall assume that the number of virtual nodes $\sharp\mathbf{N} = N$ is *fixed*.
Let us introduce the notation

$$W_t^j = W_t(P_j)$$

for $j = 1, \ldots, N$. Then W_t may be interpreted as a vector from $\mathbb{R}_+^{MN}$ or, if it is more convenient, as a nonnegative $M \times N$ matrix.
The equality $W_t = 0$ means that at the time t there is no computational activity. Thus 0 is the target of the application's evolution: once our computing MAS reaches this state we want it to stay there forever.

4 Equations of evolution

As said before we shall consider W_t as a state of the computing application. Now we shall formulate the equations of evolution of W_t. Since they contain a stochastic term (see below), W_t turns out to be *a discrete stochastic process.* The following state equations are an adaptation of the equations presented in [3] or [5] to the generalized state.
First consider three simple cases.
'Established' evolution (when there are neither partitions nor migrations). Then we assume that the state equation has the form

$$W_{t+1} = F(W_t, \xi_t) \tag{5}$$

where F is a given mapping and $(\xi_t)_{t=0,1,\dots}$ is a given sequence of random variables representing the background load influence. We assume that ξ_t are mutually independent, identically distributed and have a common finite set of values Ξ, which is justified in many natural situations [6]. Since we want W_t to stay at 0 (i.e. we want 0 to be *an absorbing state* of W_t), we need to assume that for every t

$$F(0, \xi_t) = 0 \tag{6}$$

with probability 1.
Partition at node j. Then we have

$$\begin{cases} W^j_{t+1} &= \Big(I - \operatorname{diag}(u^{jj}_t(W_t))\Big)\, W^j_t + \ \mathbf{P}\operatorname{diag}(u^{jj}_t(W_t))\ W^j_t \\ W^i_{t+1} &= W^i_t \qquad \text{for } i \neq j \end{cases} \tag{7}$$

where components of $u^{jj}_t : \mathbb{R}^{MN}_+ \to [0,1]^M$ are the proportions of the weight components of splitting agents to the corresponding components of the total weight of all agents at node j at time t. By $\operatorname{diag}(v)$ we denote the square matrix obtained by putting the elements of the vector v on the diagonal and 0's outside the diagonal.
Migration from j to k. In this case the state equations have the form

$$\begin{cases} W^j_{t+1} &= \Big(I - \operatorname{diag}(u^{jk}_t(W_t))\Big)\, W^j_t \\ W^k_{t+1} &= W^k_t + \operatorname{diag}(u^{jk}_t(W_t)) W^j_t \\ W^i_{t+1} &= W^i_t \qquad \text{for } i \notin \{j,k\} \end{cases} \tag{8}$$

with u^{jk}_t analogous to u^{jj}_t.
In reality all three cases usually appear simultaneously. Therefore the final state equations shall be a combination of them which will reduce to a particular 'simple' case when there are no activities in the system related to the other cases. Here we present such a combination, namely we propose the following form of the state equations

$$\begin{cases} W^i_{t+1} &= g^i\left(F^i(\widetilde{W}_t, \xi_t),\ \mathbf{P}\operatorname{diag}(u^{ii}_t(W_t))\ W^i_t,\ \sum_{j\neq i} \operatorname{diag}(u^{ji}_t(W_t))\ W^j_t\right) \\ W^i_0 &= \widehat{W}^i \end{cases} \tag{9}$$

for $i = 1, \dots, N$, where $\widetilde{W}_t^i = \left(I - \sum_{k=1}^N \operatorname{diag}(u_t^{ik}(W_t))\right) W_t^i$ and $\widehat{W}$ is a given initial state.

Remark 1. In our previous papers we considered only $g(s, p, m) = s + p + m$. A more general assumption on g could be $g(s, 0, 0) = s$, $g(0, p, 0) = p$, $g(0, 0, m) = m$.

It follows that W_t is a *controlled stochastic process* with a *control strategy*

$$\pi = (u_t)_{t \in \mathbb{N}}, \quad u_t : \mathbb{R}_+^{MN} \longrightarrow U. \tag{10}$$

The control set U contains such elements α from $[0,1]^{M(N \times N)}$ that satisfy at least the following conditions for $m = 1, \dots, M$.

$$\alpha_m^{ij} \cdot \alpha_m^{ji} = 0 \text{ for } i \neq j, \quad \alpha_m^{i1} + \dots + \alpha_m^{iN} \leq 1 \text{ for } i = 1, \dots, N. \tag{11}$$

In fact quite often the conditions imposed on U shall be more restrictive (see next section).
The first equation in (11) can be interpreted in the following way: *at a given time migrations between two nodes may happen in only one direction*. The second equality says that *the number of agents leaving a node must not exceed the number of agents present at the node just before the migration*.

Remark 2. It is easy to see that the control set U defined by the conditions (11) is compact (and so are of course its closed subsets).

In the most general case one might want to take the whole $\mathbb{R}_+^{MN}$ for the state space of the stochastic process W_t. But, on the other hand, in the most common situation its components represent some resources which are naturally bounded and quantized (see [5, Sec. 2]) for an analysis of a special case). Therefore we shall assume that the state space is a finite subset of $\mathbb{N}^{MN}$ containing 0. Let us call the elements of this finite set s_i, i.e. $S = \{s_0 = 0, s_1, \dots, s_K\}$. This analysis of the state space has some consequences. First of all we have to assume that F and g have values in S. Likewise, the condition (11) is not sufficient for the equations (9) to make sense. Namely we have to assume that for any $t \in \mathbb{N}$ and $W \in S$

$$u_t(W) \in U_W = \{\alpha \in U : \ G(W, \alpha, \xi) \in S \text{ for } \xi \in \varXi\}$$

with

$$G^i(W, \alpha, \xi) = g^i(F^i((I - \textstyle\sum_{k=1}^N \operatorname{diag}(\alpha^{ik}))W^i, \xi), \ \mathbf{P} \operatorname{diag}(\alpha^{ii})\ W^i, \\ \textstyle\sum_{j \neq i} \operatorname{diag}(\alpha^{ji})\ W^j) \tag{12}$$

denoting the right hand side of (9). The above equality implies that $G(W, 0, \xi) = F(W, \xi) \in S$ for any W and ξ, which means that $0 \in U_W$, therefore U_W is nonempty for every $W \in S$. Another consequence of (12) is that $G(0, \alpha, \xi) = F(0, \xi) = 0$, i.e. 0 remains an absorbing state of W_t even if we apply some agent operations.

Remark 3. Given (9) it is easy to see that W_t is a *controlled Markov chain* with transition probabilities $p_{ij}(\alpha) = \Pr(G(s_i, \alpha, \xi_0) = s_j)$ for $i, j = 0, \dots, K$, $\alpha \in U_{s_i}$. The transition matrix for the control u is $P(u) = [p_{ij}(u(s_i))]_{i,j=0,\dots,K}$.

5 Analysis of a special case

In this section we shall present more details on the state variables and the state equations in a simple but nontrivial special case. Let us assume that we use a homogeneous computational environment, i.e. VCN's are deployed on different identical physical nodes. Furthermore assume that the evolution of E_i for each agent when there are no migrations, partitions or delays related to a higher background load is like on Fig. 1, i.e. linearly decreasing from the time of the agent's creation t_s until the end of computations. For each agent the maximal value $\bar{E}$ of E_i may be different. Assume also that M_i is equal to a positive constant throughout the agent's life and before its creation and after its destruction M_i is equal to 0 (Fig. 1). Of course not all computational agents behave in the

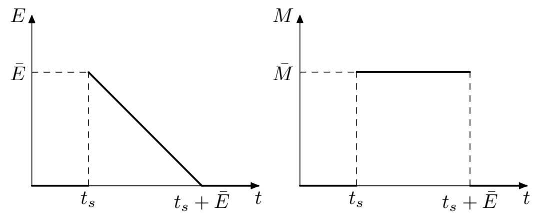

Fig. 1. An agent's remaining time of computations (in common units, left picture) and memory requirement (in bytes, right picture)

presented way (e.g. HGS agents [9] do not), but there is an important class of agents related to CAE computations (e.g. SBS linear solver agents [7]) whose evolution can be modeled as above.

In order to find appropriate state variables we divide the agents into *generations* according to their place in the partition history, i.e. the generation 0 consists of agents never split and the last generation (with number G) contains agents too small to be partitioned. We assume that within each generation all agents have the same memory requirement. Then the memory requirement becomes a *strictly decreasing* function of a generation $M : \{0, \ldots, G\} \longrightarrow \mathbb{N}$. Denote by $N^j_{k,t}$ the number of agents in the generation k at the node j at the time t and by $E^j_{k,t}$ the total remaining time (cf. (1)) of all agents in the generation k at the node j at the time t. Using these notations we define the state variable

$$W^j_t = \left[E^j_{0,t}, \ldots, E^j_{G,t}, N^j_{0,t}, \ldots, N^j_{G,t}\right]^T$$

(W^T denotes the *transposed matrix for* W). Note that both $E^j_{total} = \sum_{k=0}^{G} E^j_k$ and $M^j_{total} = \sum_{k=0}^{G} M(k) N^j_k$ are observables of our system, however they are not state variables.

Next let us consider the evolution of W_t. To this end let us assume that during the partition an agent is split into two equal children in such a way that both get a half of the parent's remaining time of computations. Then the equations of the partition for one agent belonging to the generation k ($k < G$) have the form

$$\begin{cases} E^j_{k,t+1} = E^j_{k,t} - e & N^j_{k,t+1} = N^j_{k,t} - 1 \\ E^j_{k+1,t+1} = E^j_{k+1,t} + e & N^j_{k+1,t+1} = N^j_{k+1,t} + 2 \end{cases}$$

where e is the splitting agent's remaining time. Thus the partition matrix has the following form

$$\mathbf{P} = \begin{bmatrix} B & 0 \\ 0 & 2B \end{bmatrix}$$

where B is the matrix which has 1's right under the diagonal and 0's elsewhere. It means that the system evolution follows (7). Similarly it is easy to see that (8) describes the dynamics of migration in this situation as well. Therefore let us concentrate on the 'established' case. Then the evolution equation of variables E is the following

$$E^j_{k,t+1} = E^j_{k,t} - N^j_{k,t} + D^j_{k,t}$$

where $D^j_{k,t} \in \{0, \ldots, N^j_{k,t}\}$ is a random number of agents delayed due to the high background load. In order to drop the dependency of the set of values of D on the current state we shall rewrite it in the following way $D^j_{k,t} = [N^j_{k,t}\xi^j_{k,t}]$ where $[x]$ stands for the whole part of x and $\xi^j_{k,t}$ is a sequence of random variables with values in $[0, 1]$. The equations of evolution of variables N are slightly more complicated, namely we have

$$N^j_{k,t+1} = \begin{cases} N^j_{k,t} - \widetilde{D}^j_{k,t} & \text{if } E^j_{k,t} \geq 2N^j_{k,t} \\ E^j_{k,t} - N^j_{k,t} - \widetilde{D}^j_{k,t} & \text{if } N^j_{k,t} < E^j_{k,t} < 2N^j_{k,t} \\ E^j_{k,t+1} & \text{if } E^j_{k,t} = N^j_{k,t} \end{cases} \tag{13}$$

where $\widetilde{D}^j_{k,t} \in \{0, \ldots, N^j_{k,t} - D^j_{k,t} - 1\}$ is another random variable. It represents the uncertainty about whether agents within a generation are at the same point of computations: the smaller the value of $\widetilde{D}^j_{k,t}$, the better balanced the generation. The equality $\widetilde{D}^j_{k,t} = 0$ means that all agents within the generation j are exactly at the same point. The main reason of a probable poor balancing are agents-wanderers, which do not perform any computations and migrate over the network instead for some time, and finally end up at the original node in a very early stage of computations in comparison to agents which have stayed 'home' and worked hard. $\widetilde{D}^j_{k,t}$ enables us to consider such situations in a probabilistic way. As before we put $\widetilde{D}^j_{k,t} = [(N^j_{k,t} - D^j_{k,t})\zeta^j_{k,t}]$ where $\zeta^j_{k,t}$ is a sequence of random variables with values in $[0, 1)$. Thus the interpretation of the equations (13) in the case of the perfect balancing of computations within generations is the following. The first means that when there is sufficiently much work to do the number of agents is constant. The second and third mean that at the end of computations we have a number of agents and each of them is to finish its task in a time unit. To sum up, in the presented case we have obtained the state equations, which have the form (5). This shows what F may look like. In general we need to consider migrations and partitions as well, ending up with equations like (9).

6 Optimal scheduling problem

Now let us recall the definition of the optimal scheduling for a computing MAS. We shall follow the steps of [3] and put our problem into the stochastic optimal control framework.
The general form of the cost functional for controlled Markov chains of our type is (cf. [10])

$$V(\pi; s) = E[\textstyle\sum_{t=0}^{\infty} k(W_t, u_t(W_t))] \tag{14}$$

where π is a control strategy (10) and s is the initial state of W_t, i.e. $W_0 = s$. Since 0 is an absorbing state we shall always assume that remaining at 0 has no cost, i.e. $k(0, \cdot) = 0$. This condition guarantees that the overall cost can be finite.
Let us define the following set of admissible control strategies $\mathbf{U} = \big\{\pi : \ u_t(W) \in U_W, \ t \in \mathbb{N}\big\}$. Now we are in position to formulate the *optimal scheduling problem.* Namely given an initial configuration $\widehat{W}$ we look for a control strategy $\pi^* \in \mathbf{U}$ such that

$$V(\pi^*; \widehat{W}) \quad = \quad \min\big\{V(\pi; \widehat{W}) \quad : \quad \pi \ \in \ \mathbf{U}, \ \ W_t \text{ is a solution of (9)}\big\}. \tag{15}$$

Consequently, an *optimal scheduling* for W_t is a control strategy π^* realizing the minimum in (15). Changing the cost functional we obtain various criteria for the optimality of the scheduling.
Let us present here some cost functionals of type (14) which are appropriate for multi-agent computations. The first one is the expected total time of computations

$$V_T(\pi; s) = E\big[\inf\{t \geq 0 : W_t = 0\} - 1\big]. \tag{16}$$

In other words, in this case a scheduling is optimal if it is expected to finish the computations in the shortest time.
The second functional promotes the good mean load balancing over time. It has the following form

$$V_L(\pi; s) = E\left[\textstyle\sum_{t=0}^{\infty} \sum_{i=1}^{N} (L_t^i - \overline{L}_t)^2\right], \quad \overline{L}_t = \frac{1}{N} \textstyle\sum_{i=1}^{N} L_t^i \tag{17}$$

where L_t^i is the load concentration (1) at the i-th VCN at the time t. These quantities are well defined because we have assumed that E_{total}^i and M_{total}^i are observables of our system.
Both the above examples do not contain an explicit dependency on the control. Generalizing it a little allows us to penalize migrations. Namely take $\varphi : S \to \mathbb{R}_+$, $a \geq 0$ and $\mu_m^{ij} : [0, 1] \to \mathbb{R}_+$ nondecreasing and such that $\mu_{ij}(0) = 0$, and put

$$V_M(\pi; s) = E\left[\textstyle\sum_{t=0}^{\infty} \left(\varphi(W_t) + a \sum_{m=1}^{M} \sum_{i \neq j} \mu_m^{ij}(u_{m,t}^{ij}(W_t))\right)\right]. \tag{18}$$

In the above expression φ can have the form of the term under the estimated value in (16) or (17), μ^{ij} allows us to penalize the *distance* between i-th and j-th VCN and a is a tuning factor (the greater is the value of a, the greater is the influence of the 'migration' term).

7 Existence and characterization of optimal strategies

Let us consider first the existence of solutions for problem (15). To this end let us denote by $R(u) = [p_{ij}(u(s_i))]_{i,j=1,\ldots,K}$ the 'probably not absorbing' part of the transition matrix for a control u. The following proposition is the main existence result.

Proposition 4. *Assume that*
(R1) $R^K(u)$ is a contraction for every u such that $u(s) \in U_s$ or
(R2) $R^n(u)$ is a contraction for some $n \geq 1$ and u as above but additionally $k(s_j, \alpha) \geq \varepsilon > 0$ for $j \neq 0$, $\alpha \in U_{s_j}$.
Then there exists the unique optimal solution of (15).

Proof. It is sufficient to notice that U_s are finite so they are compact and $k(s_j, \cdot)$ is obviously continuous. It means that the assumptions (A1)–(A3) from [10, Chap. 4] hold. Thus the thesis is a straightforward consequence of [10, Theorem 4.2].

Corollary 5. *Consider our example cost functionals V_T, V_L and V_M.*

1. *Problem* (15) *for V_T has the unique solution provided (R2) holds.*
2. *Problem* (15) *for V_L has the unique solution provided (R1) holds.*
3. *Existence of the solution for* (15) *for V_M depends on the assumption on φ. If the latter is separated from* 0 *we need (R2) otherwise (R1).*

Now we shall present some Bellman-type optimality conditions for (15), which are another consequence of [10, Theorem 4.2] and its proof. We shall formulate them in the following proposition.

Proposition 6. *Assume (R1) or (R2). Then the optimal solution of* (15) *is a stationary strategy $\pi^* = u^\infty = (u, u, \ldots)$ and it is the unique solution of the equation*

$$V(\pi^*; s) = \min_{\alpha \in U_s} \left[\sum_{j=1}^{K} p_{ij}(\alpha) V(\pi^*; s_j) + k(s, \alpha) \right]. \tag{19}$$

The solution of (19) *exists and is the optimal solution of* (15).

The simple but important consequence of this proposition is that in order to find the optimal scheduling we need to consider only stationary strategies. Note that diffusion strategies defined in [8, 4] are stationary, thus we can try to verify their optimality (or quasi-optimality) by means of a variant of the equation (19). On the other hand this equation also allows us to compute the optimal strategy by means of some iterative procedures like Gauss-Seidel [10].

8 Conclusions

The presented MAS architecture accompanied by diffusion-based agent scheduling strategies make a convenient and efficient environment for large-scale distributed computations. The presented mathematical model based on the stochastic optimal control theory provides us with a new precise definition of optimal scheduling in such an environment. It also enables us to obtain some results on the existence of optimal scheduling strategies (Proposition 4 and Corollary 5) as well as the optimality conditions (Proposition 6). These results show in particular that any optimal scheduling must belong to the class of stationary strategies, which have been utilized during tests [8, 9]. In this paper the formal model has been extended to allow more general system state. This in turn has enabled us to provide a more detailed description of the generalized model in an important special case (Sec. 5) which form the basis for experiments designed to verify the model. Such experiments are being undertaken with results soon to come.

References

1. Luque, E., Ripoll, A., Cortés, A., Margalef, T.: A distributed diffusion method for dynamic load balancing on parallel computers. In: Proceedings of EUROMICRO Workshop on Parallel and Distributed Processing, San Remo, Italy, IEEE Computer Society Press (1995) 43–50
2. Uhruski, P., Grochowski, M., Schaefer, R.: Multi-agent computing system in a heterogeneous network. In: Proceedings of the International Conference on Parallel Computing in Electrical Engineering (PARELEC 2002), Warsaw, Poland, IEEE Computer Society Press (2002) 233–238
3. Smołka, M., Grochowski, M., Uhruski, P., Schaefer, R.: The dynamics of computing agent systems. Lecture Notes in Computer Science **3516** (2005) 727–734
4. Grochowski, M., Smołka, M., Schaefer, R.: Architectural principles and scheduling strategies for computing agent systems. Fundamenta Informaticae **71**(1) (2006) 15–26
5. Smołka, M.: Optimal scheduling problem for computing agent systems. Inteligencia Artificial **9**(28) (2005) 101–106
6. Smołka, M., Schaefer, R.: Computing MAS dynamics considering the background load. Lecture Notes in Computer Science **3993** (2006) 799–806
7. Grochowski, M., Schaefer, R., Uhruski, P.: An agent-based approach to a hard computing system — Smart Solid. In: Proceedings of the International Conference on Parallel Computing in Electrical Engineering (PARELEC 2002), Warsaw, Poland, IEEE Computer Society Press (2002) 253–258
8. Grochowski, M., Schaefer, R., Uhruski, P.: Diffusion based scheduling in the agent-oriented computing systems. Lecture Notes in Computer Science **3019** (2004) 97–104
9. Momot, J., Kosacki, K., Grochowski, M., Uhruski, P., Schaefer, R.: Multi-agent system for irregular parallel genetic computations. Lecture Notes in Computer Science **3038** (2004) 623–630
10. Kushner, H.: Introduction to Stochastic Control. Holt, Rinehart and Winston (1971)