Ant Based Heuristic for OS Service Distribution on Ad Hoc Networks

Tales Heimfarth and Peter Janacik

Heinz Nixdorf Institut, University of Paderborn
Fuerstenallee 11, 33102 Paderborn, Germany
{tales,pjanacik}@upb.de

Abstract. This paper presents a basic and an extended heuristic to distribute operating system (OS) services over mobile ad hoc networks. The heuristics are inspired by the foraging behavior of ants and are used within our NanoOS, an OS for distributed applications. The NanoOS offers an uniform environment of execution and the code of the OS is distributed among nodes.
We propose a basic and an extended swarm optimization based heuristic to control the service migration in order to reduce the communication overhead. In the basic one, each service request leaves pheromone in the nodes on its path to the service provider (like ants leave pheromone when foraging). An optimization step occurs when the service provider migrates to the neighbor node with the higher pheromone concentration. The proposed extension takes into account the position of the node in the network and its energy.
Realized simulations have shown that the basic heuristic performs well. The total communication cost in average is just 40% higher than the global optimum. In addition, both heuristics have a low computational requirement.

1 Introduction

Distributed systems running on MANETs (mobile ad hoc networks) open a new spectrum of applications but also bring new challenges. Many interesting applications in this domain consist of collaborative distributed tasks among geographically dispersed nodes. However, for a good resource utilization and for an adequate development of such distributed applications, the support offered by an operating system (OS) is important. The OS manages the hardware resources and offers a common system call interface in each node simplifying the application development.

The objective of this paper is to introduce a basic and extended heuristic for service distribution used in our OS. NanoOS is a complex, innovative OS for resource constrained embedded devices able to establish an ad hoc network. The code of the OS is distributed among the wireless nodes in order to fit into the small nodes.

Please use the following format when citing this chapter:

Heimfarth, T., Janacik, P., 2006, in IFIP International Federation for Information Processing, Volume 216, Biologically Inspired Cooperative Computing, eds. Pan, Y., Rammig, F., Schmeck, H., Solar, M., (Boston: Springer), pp. 75–84.

As the OS components (services) are distributed, adapting automatically to dynamically changing conditions by changing the distribution of functionality across the ad hoc network is an important issue in our system. This adaptation should also help to reduce the overhead and the energy consumption. For this propose, we develop an distribution and migration algorithm based on swarm intelligence which tries to reduce the communication among different nodes of the system. As processing speed usually is orders of magnitude higher than communication speed, this also affect positively the global performance of the system.

2 Related Work

The academic system called MagnetOS [1] offers a distributed Java virtual machine that provides an automatic migration of elements of the system trying to maximize total application lifetime by utilizing power more efficiently. The migration mechanisms have some similarities with our ant-based migration algorithm but different from our approach, it neither considers the resource availability in the nodes nor the link quality.

Our problem of placement and migration of the OS services to different nodes is very similar to a global scheduler that decides where the processes will be executed in a distributed system.

The static scheduler makes the decisions just with information available at compilation time. There are several theoretical analysis of the task assignment problem. Some approaches consider a graph formed by system nodes together with tasks as vertices and communication costs together with execution costs as edges without considering a multi-hop network topology ([2, 3, 4]). Other research deals with multi-hop networks with a complex topology ([5, 6, 7, 8]).

In our approach, we are using a dynamic distributed non-cooperative scheduling strategy, i.e., the current state of the system is used in order to drive the migration. Moreover, each service is an autonomous agent that decides itself when to migrate and to which node.

In the area of dynamic distributed scheduling algorithms, there are a lot of approaches that try to share the load of networked nodes among them ([9, 10]).

Although the algorithms are distributed, they do not take in account the topology of the network. Moreover, movement of nodes is not considered.

3 Overview of the NanoOS

Our system is composed of three main components: the hardware, the OS and the application running on top of it (see Fig. 1). The hardware platform consists of a set of distributed mobile nodes, each one with small processing unit, limited memory and wireless adapter. Our *NanoOS* runs on top of such an architecture

and provides the adequate set of services to the application. Besides the traditional OS services , the NanoOS provides a set of special services to support the distributed processing, like migration and distributed synchronization.

In our OS, each application is composed by a set of tasks. The OS provides a uniform and remotely available system call environment even with movement of nodes (and connections being broken).

For the purpose of reducing the per node OS footprint and to enable the execution of a rather complex OS in very hardware constrained nodes, the NanoOS distributes the services among the nodes. Each node of the system has just a small part of the services of the complete OS; a group of nodes together form an instance of the OS. At any instant of time one node may connect and use a service residing in another node using a remote method invocation (RMI).

The Figure 1 presents an overview of the system. The tasks from applications use services of the OS. In order to reduce the resource requirement in each node, the services are shared among different application tasks executed in other nodes.

The services and tasks can migrate in order to optimize communications. Moreover, the same migration mechanisms used by the OS services are also offered for application level tasks. Applications' tasks can offer services to others and may also automatically migrate. For sake of simplicity, we will speak from here on simply about migration of services. The main contribution of this paper is the algorithm presented that is responsible to assign dynamically the OS services (or tasks offering services) to nodes trying to minimize the communication overhead.

4 Service Distribution Using Swarm Optimization

After a service discovery phase (not described here), a communication between the node of the application task and the node hosting the service is set up. We now assume a situation where tasks distributed in the system are communicating with services which are distributed as well. A single path routing is responsible for finding a good route between the nodes.

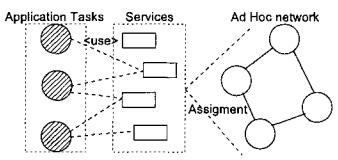

Fig. 1. System overview

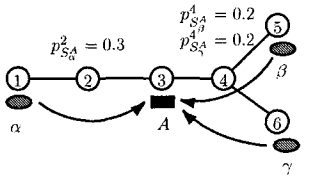

Fig. 2. Pheromone based service distribution

The objective of the service distribution is to change the location where the services (and mobile tasks) are executed (migrating them) during runtime trying to minimize the communication overhead.

There are possible constraints to the movement of the objects: only to a direct neighbor (1 hop); within a neighborhood (k hops); or unbounded (infinity hops). In this paper we are just considering 1 hop movements.

4.1 Load and Communication Models

To support the migration/reconfiguration process, the load of the hardware and the communication pattern are important information and have to be exposed to the OS. For this paper, a simplified load model, just considering the amount of local free memory is used.

In order to model the communication, we create a link metric called *virtual distance* and is represented by $D(u,v)$ (d and v are nodes). It ranges from Γ to $\Gamma + A$:

$$D(u,v) = \Gamma + A \cdot (1 - (s_{(u,v)}^{\alpha} \cdot l_{(u,v)}^{\beta} \cdot d_{(u,v)}^{\gamma} \cdot e_{(u,v)}^{\zeta})^{\frac{1}{\alpha+\beta+\gamma+\zeta}}) \quad (1)$$

The used metrics are the error rate: $e_{(u,v)} \in [0,1]$ (0 means 100% of error rate), the live time: $l_{(u,v)} \in [0,1]$ (this metric varies according the relapsed time of the link, 0 means new link), the delay (correlated with queue): $d_{(u,v)} \in [0,1]$ (0 means maximum delay, 1 means minimum delay) and the RSSI (received power): $s_{u,v)} \in [0,1]$ (0 means no reception signal).

α, β, γ and ζ define the weight of each metric in the geometrical mean.

4.2 Basic Heuristic for Service Distribution

In our approach we are optimizing the position of the services of the system through *migration*, i.e., we try to find the optimal configuration where the communication overhead caused by the remote requests is minimized. In order to solve this online discrete optimization problem, we decide to use an ant inspired algorithm that is described in this section. It is relatively simple and has shown good performance.

The system is represented by the graph $G = (V, E)$ with nodes V and bidirectional links E. The nodes correspond to the physical devices and the links to the wireless connections. The links are weighted with the *virtual distance* metric. Additionally, each service instance $i \in I$ is of a type $p \in P$. A task $a \in A$ has no type. We will use the word service and task to denote a service instance and task instance. Each requester $r \in \{I \cup A\}$ (requester can be services or tasks) of a service $i \in I$ has a service state S_r^i. A node $v \in V$ has a pheromone table $P_v = [p_{S_r^i}^v]_{r \in \{I \cup A\}, i \in I}$. This pheromone level represents the request rate (and traffic) made by the requester r to the service i that are crossing the node v. In our approach, all nodes are responsible for service distribution, since each node's evaluation is based on its *local* view. Moreover, the needed information

is constantly changing, due to frequent pheromone updates so that transferring the decision to just certain nodes would incur an high additional communication overhead without efficiency gains.

Using an analogy with the ant foraging behavior [11], the services in our approach are the equivalent of the food source. The calls made by the requesters are the agents (or ants) and the requesters are the nests. The wireless links form the paths which the ants can use for movement. While the requests are being routed to the destination service, they leave pheromone on the nodes.

The pheromone tables in each node are updated according to the equation $p_{S_r^i}(t+1) = \frac{p_{S_r^i}(t)+\delta p(h)}{1+\delta p(h)}$, where the $\delta p(h)$ is the variation of the pheromone and it is a function of the size of the packet.

In defined time intervals, each service evaluates whether it should migrate to another node in order to improve the communication (reduce the overhead). This neighbor is selected using the following method.

Let $v \in V$ be the local node of the service $i \in I$, $d \in V$ is the destination node of the service and $N_v, v \in V$ is the set of neighbors of v.

$$b^i_{v \to d} = \frac{\sum_{x \in \{I \cup A\}} p^d_{S^i_x}}{\sum_{y \in N_v} \sum_{x \in \{I \cup A\}} p^y_{S^i_x}} \quad (2) \qquad e_i = max(b^i_{v \to d}), d \in N_v \quad (3)$$

$b^i_{v \to d}$ (eq. 2) represents a force between $[0, 1]$ that the service i migrates form node v to node d. The selection of the final destination node $e_i \in V$ of the service i is made simply using the expression 3.

The migration process is initialized when the sum of pheromones of some neighbor exceeds a threshold value Θ.

In Fig. 2, a scenario with six nodes is shown. In this scenario, node 1 has the task α that accesses the service A in node 3. At the same time, tasks β and γ, located in the nodes 5 and 6 respectively, are also using the service A. Let's assume that the pheromone related to the connection $\alpha \to A$ in the node 2 is $p^2_{S^A_\alpha} = 0.3$, the pheromone of the connection $\beta \to A$ in the node 4 is $p^4_{S^A_\beta} = 0.2$ and the pheromone of the connection $\gamma \to A$ is $p^4_{S^A_\gamma} = 0.2$. According to equation 2, the force attracting this service to the node 2 is $b^A_{3 \to 2} = \frac{0.3}{0.3+0.2+0.2} = 0.428$ whereas the force attracting to node 4 is $b^A_{3 \to 4} = \frac{0.2+0.2}{0.3+0.2+0.2} = 0.571$. This means that the service A will migrate to the node with higher total pheromone level, i.e., node 4.

Direction Extension In this section, an identified problem caused by the greedy nature of the presented algorithm is described and a solution is proposed. The problem occurs when more than one nearly located tasks request the same service, but due to the routing algorithm, the requests use different paths. An example of such situation is depicted in Fig. 3. This situation can only occur if there are more than two requesters using the same service. It is more likely to occur when the service is located in a node-dense area of the network.

In Fig. 3, the tasks α, β and γ are accessing the service A in node 3. The total communication cost C can be calculated using the eq. 4, where A is the

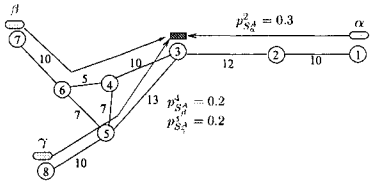

Fig. 3. Instance that results in wrong migration

$$C = \sum_{k=1}^{\#I} \sum_{l=1}^{\#A} B(a_k, i_l) \cdot D(Q(a_k), Q(i_l)) \tag{4}$$

set of application tasks: $A = \{a_1, a_2, ..., a_m\}$ and I the set of service (instances) of the system $I = \{i_1, i_2, ..., i_n\}$.

The function $B(a_m, i_n) \geq 0$ gives the average bandwidth utilization by the requests made by the task a_m to the service i_n. The function $Q : A \cup I \rightarrow V$ maps the tasks and services to the hosting node. The objective function of the service distribution heuristic is to find the assignment function Q that minimizes the communication cost C (and therefore minimizes the energy consumption, assuming that communication is a major energy consuming operation). Our problem is a special instance of the QAP (*Quadratic Assignment Problem*), where instead of Euclidean distance between points, the sum of the virtual distances of the routed path is used ($D(v, w)$, where $v, w \in V$) and the cost is given by the bandwidth utilization ($B(a, i)$, where $a \in A$ and $i \in I$). It is known that the QAP is an NP-hard problem [12].

Returning to our example (Fig. 3), as the average bandwidth utilization is proportional to the pheromone deposited in a node inside the used path, the total communication cost in this case is 16.2 (calculated using eq. 4). As the pheromone in the node 2 is higher than in the nodes 4 and 5, the next step of the basic algorithm would be to migrate the service A to node 2. Here, the communication cost is 17.4. This result shows that the heuristic selects the wrong node to migrate to, increasing the total communication cost. This happens because of the lack of information over not directly connected parts of the network. The main idea of the improvement is to migrate the service not to the neighbor with the biggest amount of requests (requests we call also flow) but to the neighbor whose flow (request traffic) is crossing near to nodes that are in flows from other requests to the same service. If the defined metric (virtual distance) has (geographical) norm properties, this will be equivalent to migrating the service to the geographical *direction* where the highest amount of requests is coming from. Two requests coming from task α and β (see Fig. 3) are transversing neighboring nodes in order to reach A, thus, they should attract the service instead of γ.

In addition, the new migration heuristic is based not just on the pheromone level to drive the migration of the services, but also on a "potential goodness" of each node to receive highly loaded services and the energy level of the nodes. The "potential goodness" η_{vi} measures how appropriate it is for node v to receive service i, i.e., whether the node is central in the network and the service i is

a highly required one. If the complete network would be known by each node, the centrality could be measured by the sum of the distances to every other node. The idea of the potential goodness is that services with high flow are coupled with high probability to locations with good connections to others. Just using this rule, it is possible to obtain good (but not optimal) placement of the services in the network [11].

Definitions and heuristic description:

Like in basic heuristic, each node has just local information, we define η_{vi} where $v \in V$, $i \in I$ and $0 \leq \eta_{vi} \leq 1$ in eq. 5.

$$\eta_{vi} = [1 - \sum_{g \in N_v} \frac{D(v,g)}{(\#N_v)^{\delta} \cdot D_{max}}] \cdot h(i) \quad (5) \qquad b^i_{v \to d} = \frac{[\tau^i_{v \to d}]^{\alpha} \cdot [\eta_{di}]^{\beta} \cdot [E_d]^{\gamma}}{\sum_{x \in N_v} [\tau^i_{v \to x}]^{\alpha} \cdot [\eta_{xi}]^{\beta} \cdot [E_x]^{\gamma}} \quad (6)$$

where $N_v, v \in V$ the set of neighbors of v and $\delta \geq 1$ gives the importance of the number of neighbors, and D_{max} gives the maximum allowed virtual distance. $h(i) : I \to [0,1]$ returns the current request load (how much traffic) that service i is currently serving (where 0 means the service is idle and 1 means full load). The energy of the node is given by E_v and $0 \leq E_v \leq 1$, where 1 means full and 0 empty.

In addition to the already presented pheromone table P_v that stores the rate of requests that are crossing the node v, there is a second table F_v that stores the information about the flows that are occurring in the neighbor nodes. $F_v(S^i_r) : \{I \cup A\} \times I \to \{0,1\}$ return 1, iff some direct neighbor of the node v is routing a request from the requester r to the service i.

The idea is that neighboring communications (like the S^A_β and S^A_γ in the figure) can be recognized as coming from the same network "direction" by the service A.

The table N_v is filled without the necessity of any direct exchange of messages between the node v and the neighbors. Each node just hears the communication originating from neighboring nodes to fill the table. If the node v has a directed connection to the node u where the service i is located, it ignores all the neighboring communication going to to the service i (i.e., for $\forall r \in \{I \cup A\}$, $F_v(S^i_r) = 0$). This avoids the problem that near the sink (service i) all nodes can hear each other, resulting on a false interpretation that all requests are coming from a similar direction.

Each request r to the service i now carries the information collected in the nodes about which requests to the service i are occurring in neighbor nodes (i.e., it collects the N_v information of the nodes when traveling to service i). $F(S^i_{r_1}, S^i_{r_2}) : \{I \cup A\} \times \{I \cup A\} \times I \to \{0,1\}$ return 1 iff r_1 and r_2 are neighboring requests (flows).

In the original heuristic, the "force" attracting the service i from node v to node d ($b^i_{v \to d}$, see eq. 2) does not take into account the requests coming

from near areas of the network. The new $b^i_{v\to d}$ is calculated now in two steps. In the first, the $\tau^i_{v\to d}$ take in account the pheromone (and neighboring flow information) and rate the attractiveness of node d (eq. 7).

$$\tau^i_{v\to d} = \frac{\sum_{x\in\{I\cup A\}} p^d_{S^i_x} + \sum_{x\in\{I\cup A\}}\sum_{z\in\{I\cup A\}}\sum_{g\in N_v-\{d\}} p^g_{S^i_z}\cdot\lceil p^d_{S^i_x}\rceil\cdot F(S^i_z,S^i_x)}{\sum_{y\in N_v}[\sum_{x\in\{I\cup A\}} p^y_{S^i_x} + \sum_{x\in\{I\cup A\}}\sum_{z\in\{I\cup A\}}\sum_{g\in N_v-\{d\}} p^g_{S^i_z}\cdot\lceil p^y_{S^i_x}\rceil\cdot F(S^i_z,S^i_x)]} \quad (7)$$

The first term is the same of the eq. 7, that means, the sum of all requests coming to service i through node d. The second term of the numerator is the sum of the pheromone from flows that are neighbors of the ones traveling through d. As already explained, the F function tests whether S^i_z and S^i_x are neighbor flows, and the ceiling $\lceil p^d_{S^i_x}\rceil$ checks whether the connection S^i_x exists in the node d (i.e. $p^d_{S^i_x} > 0$). The denominator normalizes τ ($0 \leq \tau^i_{v\to d} \leq 1$).

Finally, the eq. 6 returns the new $b^i_{v\to d}$ that is the force between $[0,1]$ that the service i migrates from node v to node d. This combines the pheromone value (with direction concept, eq. 7) with the potential goodness of a location to some service and the available energy. The selection of the destination node of the migration is made, like the basic heuristic, using eq. 3.

Returning to the example shown in Fig. 3, and assuming that all the nodes have the same potential goodness and energy (= 1), for sake of simplicity and that $\alpha,\beta,\gamma = 1$, $\tau^A_{3\to 2} = b^A_{3\to 2} = \frac{0.3}{0.3+0.4+0.4} = 0.27$, $\tau^A_{3\to 4} = b^A_{3\to 4} = \frac{0.2+0.2}{0.3+0.4+0.4} = 0.36$ and $\tau^A_{3\to 5} = b^A_{3\to 5} = \frac{0.2+0.2}{0.3+0.4+0.4} = 0.36$. This result shows that the service A will migrate correctly to the node 4 or 5 instead of 2 when the basic version of the heuristic is used.

5 Results

A simulation environment to evaluate our basic ant-based service distribution heuristic was implemented in C++ using the *Boost* library for graph algorithms support. The routing of network traffic was idealized by using Dijkstra shortest path algorithm.

Instances of the ad hoc network were generated by random selection of nodes' position. Moreover, the task force and the services of the OS including also the usage (dependency) graph were also randomly generated.

The received signal strength (RSSI) was calculated using the free space model for an isotropic point source in an ideal propagation medium (free-space path loss with rx,tx unitary gain: $L_f = \frac{4\pi d^2}{\lambda}$). The limits of the RSSI were determined using two thresholds that have the meaning of maximum signal strength and no signal (unit disk graph). The *RSSI* was the only metric used to produce the *virtual distance* (see Equation 1).

As already said, the objective of the heuristic is to find the assignment function Q that minimizes the communication cost C (and therefore minimizes the energy consumption, assuming that the communication is the main energy consuming operation). We restrict the maximum number of services and tasks to one per node. This simulates a simplistic resource constraint per node.

We run 10.000 different problem instances and the presented basic heuristic is applied in the first assignment (randomly generated). Fig. 4 shows the results. The x-axis shows the number of optimization steps of the algorithm, where the y-axis shows the average communication cost among all the realized simulations. The two straight lines depict the average communication cost of the first random solution to the assignment and the optimal solution (calculated using Branch and Bound over the QAP). Figure 5 shows the cumulative distribution of the testing cases.

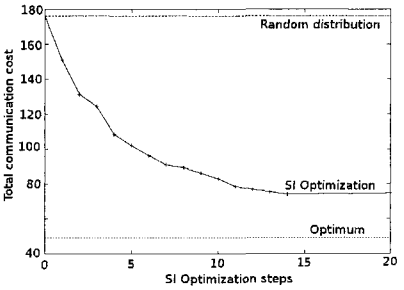

Fig. 4. Results of the simulations

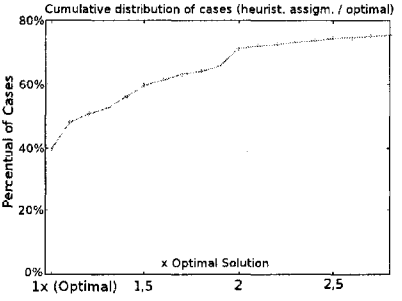

Fig. 5. Cumulative distribution of the cases

6 Conclusions

In this paper we present the concept of the NanoOS for highly distributed applications running on ad hoc wireless networks. This OS allows the migration of application/OS services among nodes. The investigated objective was the minimization of the communication overhead between application tasks and services in an ad hoc network. We proposed ant-based heuristics. The problem was modeled in detail and the quality of our ant-based method was compared with the global optimum using simulations of a large number of problem instances.

The realized simulations of the basic heuristic have shown that it performs well in average (71.97 of cost compared with 50.16 that was the cost of the optimal distribution), i.e., the total communication cost is in average only about 40% higher than the global optimum obtained using Branch and Bound. The initial random distribution of services has an average cost of 176.6. Looking at the cumulative distribution of cases (Fig. 5), we see that for the majority of test cases (70%), the heuristic could find solutions that cost at most 2 times the optimal value. In 40% of the cases, the heuristic could find the optimum.

Moreover, our heuristics have an extremely low computational cost and a small information dependency, where just local communication is necessary for the migration decision. They are also adaptive to changes in the network and can be executed in a distributed manner where each entity tries itself to find a good assignment.

We are planning to simulate the extended version of the heuristic in order to compare it with the actual results. We also want to include movement into the simulation in our future work. Concluding, the results give yet another piece of evidence that principles encountered in the nature (like agents doing just local interactions helping to achieve global results) can be transferred to computers with satisfactory results.

References

1. Emin G Sirer, Rimon Barr, T. W. D Kim, and Lan Y Fung. Automatic code placement alternatives for ad-hoc and sensor networks. Technical report, Cornell University, Ithaca, NY, USA, 2001.
2. H. S. Stone. Multiprocessor scheduling with the aid of network flow algorithms. *IEEE Trans. Software Eng.*, SE-3:85 –93, 1977.
3. V. M. Lo. Heuristic algorithms for task assignment in distributed systems. *IEEE Trans. Comput.*, 37(11):1384–1397, 1988.
4. S. Ramakrishnan, I. H. Cho, and L. Dunning. A close look at task assignment in distributed systems. In *In IEEE INFOCOM 91*, pages 806 – 812, Miami, 1991.
5. Kenjiro Taura and Andrew A. Chien. A heuristic algorithm for mapping communicating tasks on heterogeneous resources. In *Heterogeneous Computing Workshop*, pages 102–115, 2000.
6. Nectarios Koziris, Michael Romesis, Panayiotis Tsanakas, and George Papakonstantinou. An efficient algorithm for the physical mapping of clustered task graphs onto multiprocessor architectures. In *Proc. of 8th Euromicro Workshop on Parallel and Distributed Processing (PDP2000)*, pages 406–413, Rhodes, Greece, 2000. IEEE Press.
7. M. Eshaghian and Y. Wu. Mapping heterogeneous task graphs onto heterogeneous system graphs. In *Proceedings of Heterogeneous Computing Workshop*, 1997.
8. R. F. Freund, M. Gherrity, S. Ambrosius, M. Campbell, M. Halderman, D. Hensgen, E. Keith, T. Kidd, M. Kussow, J. D. Lima, F. Mirabile, L. Moore, B. Rust, and H. J. Siegel. Scheduling resources in multi-user, heterogeneous, computing environments with smartnet. In *In Proceedings of Heterogeneous Computing Workshop*, 1998.
9. Christophe Lang, Michel Trehel, and Pierre Baptiste. A distributed placement algorithm based on process initiative and on a limited travel. In *PDPTA*, pages 2636–2641, 1999.
10. Shyamal Chowdhury. The greedy load sharing algorithm. *J. Parallel Distrib. Comput.*, 9(1):93–99, 1990.
11. Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems.* Oxford University Press, New York, NY, 1999.
12. Sartaj Sahni and Teofilo Gonzalez. P-complete approximation problems. *J. ACM*, 23(3):555–565, 1976.