Ad Hoc Sensor Networks, Constraint Programming and Distributed Agreement

Christophe Guettier
IC-Parc, Imperial College
London SW7 2AZ, UK

Gérard Le Lann
INRIA, Rocquencourt B.P. 105
F-78153 Le Chesnay Cedex, France

Jean-François Hermant
Independent Consultant
www.jfhermant.com

***Abstract*— Numerous applications rest on sensor networks. Managing such networks in an efficient manner involves solving complex distributed computing problems, that problems are magnified in ad hoc sensor networks, where a limited knowledge of the topology is available. The two major issues that must be addressed simultaneously are, on the one hand, optimal sensor allocation and, on the other hand, tolerance to sensor and communication failures, which leads to solving dynamic assignment/reconfiguration problems. This paper describes a class of solutions that combines the problem solving capabilities of constraint programming techniques with the properties of distributed real-time fault-tolerant agreement. With such solutions, those safety, timeliness and dependability properties that define the management problems under consideration. The efficiency of these solutions are illustrated with real world examples.**

I. Introduction

Over the last fifty years, significant advances have led to the development of high-performance radars, sonars and other detection systems. Finding many essential applications in both defense and civilian domains, these sensor systems are based on high-performance parallel architectures coupled with phased array antennae of large dimensions. However, most often, these systems rest on complex and rigid designs, which results into little reconfiguration capabilities.

Opposed to these concepts, a new generation of networked detection systems is trading off high-performance with availability, flexibility, affordability and survivability. Although these sensor networks are inheriting classical Digital Signal Processing (DSP) functions (Beam Forming, Fast Fourier Transform or Doppler Filtering), they must be rapidly deployed in an ad hoc manner into a non-cooperative (not to say aggressive) environment, for which an incomplete or uncertain knowledge is available. Subject to sensor failures and message losses, these detection systems must constantly *adapt* their own organisation to the current environment. In fact, these ad hoc sensor networks must constantly optimize their internal organisation according to the environment stress by deciding which sensor router should be part of the active sensor subnet. This necessitates dealing at the same time with network connectivity, communication coverage, information gain and faulty sensors. Furthermore, reconfiguration and assignment decisions have to be consistent system-wide. Finally, they must meet performance and feasibility constraints – referred to as Quality-of-Service (QoS).

These issues can be split into two classes:

- Application level issues (CP and Search algorithms),
- System level issues (Fault-Tolerant Distributed Agreement).

Our approach combines Uniform ConsensuS and Uniform CoordinatioN (UCS/UCN) with Constraint Programming (CP) techniques for solving the sensor subnet management problem in terms of information gain or service availability. This approach also enables the satisfaction of sensor resource, communication coverage and interconnection constraints. In fact, because of its combinatorial structure, subnet management problem is equivalent to NP-hard problems (multi-knapsack, transshipment, perfect coupling, ...). Constraint Programming has been shown to be an efficient approach to model and solve such combinatorial problems, even for large scale instances.

II. Managing ad hoc sensor networks

A. Background

Applications of ad hoc sensor networks will range from defense and medicine to agriculture, and encompass various sensing techniques like passive, active methods, directive antenna, etc. These applications lie between the "extremes" of possible sensor networks. At one extreme are high performance sonars, radars and other detection devices, which rely on parallel processing and high-performance communication devices (crossbar) to execute DSP algorithms on huge dataflows [Gue97]. These applications cannot be distributed on a wide area network. At the other extreme are loosely coupled sensor networks, or *smart matter systems* [FCGS03]. Berkeley Motes, for example, require very low power, and their processing and interconnection capabilities are very limited.

As for traditional sensing systems, ad hoc sensor networks will have to cope with important data-flows resulting from multi-dimensional signal processing techniques. In addition to ever increasing complexity, ad hoc sensor networks cover large sensing areas using multiple redundant sensor routers. Therefore, network management, which impacts directly the efficiency of a sensor network, must be ensured with limited human supervision. Issues that arise from the need for "autonomous management" are further complicated with the following requirements: (1) at all times, an active subnet is maintained, by selecting incoming and outgoing sensor routers (or member) according to a changing environmental stress, satisfying connectivity constraints as well as optimizing the efficiency of the global sensing task, (2) reconfiguration and management problem solving tasks must be achieved in the presence of partial failures.

As a running example, we consider the monitoring of multiple tracks using several adirectional wireless sensor routers of homogeneous resolution and communication power.

B. Network model

A network consists of a set of N sensors, i.e. elements capable of sensing, processing, receiving and sending data concurrently. Sensors are not mobile. Sensors in charge of routing functions are referred to as sensor routers. A subnet is a connected topology of n sensor routers ($n < N$), a subset of the whole sensor network (fig. 1).

This paper is concerned with subnet management. We make the following assumptions:

1) Sensor routers: Sensor routers are not mobile, but can be switched on and off at any stage. They may fail by crashing. Up to $f_{max} < n$ sensor routers may fail during some given time interval.[1] Sensor routers that do not fail are called correct routers.

2) Network links: Fully connected, the netwrok allows every pair of sensor router to communicate using some wireless technology if their relative distance is below a given coverage [MKPS01]. It is also assumed that new links can be created between any new sensor router and all its neighbors within a given coverage.

3) Computational model: We assume a synchronous model of computation, i.e. any message sent on a link by a correct router at time t is delivered to any other correct router before $t + \gamma$.[2]

These assumptions are made for the sake of simplicity. In fact, it is possible to solve UCS and UCN problems under more general (hence, more realistic) assumptions, such as asynchronous computational models and message omission failures (lossy links and sensors).[LS03].

C. Distributed sensing with ad hoc sensor networks

A representative example of a function served by a ad-hoc sensor network is target detection or tracking. Within the network, a subnet of "available" sensor routers cooperate to exchange sensor data to track a single or multiple targets. The task is to deliver information sensed by any subnet sensor router to every other subnet sensor router (for example, plane signatures for Air Traffic Control systems). The aim is that all sensor routers in the subnet should acquire the same information about the targets within a specified end-to-end latency.

When the environment is changing, the subnet must decide which new sensor routers shall join the subnet and which existing subnet members shall leave, in order to maximize information retrieved from the targets. The set of sensor routers composing the new subnet must be choosen so that it is fully connected, and it satisfies the different criteria relevant to the QoS:

- The level of fault-tolerance is acceptable;
- End-to-end sensing latency must be lower than a specified bound;
- Message / processing loads are met.

On the assumption that target sensing is the only input to the network, the subnet must continuously deliver the required information in spite of two kinds of environment changes:

1) Targets change status (start or stop emitting / reflecting signals). Target status are broadcasted through the subnet with other tracking information.
2) Sensor routers fail or appear in the environment. When a sensor router appears in the environment, its localization is broadcasted to all the subnet. All potential link that a new sensor router may establish are then deduced from the coverage.

Testing each sensor router one by one is not a satisfactory approach, because the different criteria involve both members of the existing subnet and the set of new sensor routers. Such "generate and test" approach has to evaluate simultaneously all the criteria for any subset of the total set of available sensor routers (including existing subnet members and new sensor routers), which corresponds to 2^N combinations. This combinatorial structure can be assimilated to the one of a knapsack problem, known to be NP-complete. In order to find a feasible solution and to potentially maximize the amount of information retrieved by the sensor network, a "generate and test" approach can not be really applied and it is necessary to solve globally the subnet management problem, considering the different criteria as hard constraints.

This constrained optimization problem can be illustrated with figure 1. Supposing that active subnet members $\{S1, S2, S3, S4\}$ do not provide enough quality information about targets $T2$ and $T3$, any subset of $\{N1, N2, N3, N4\}$ would improve sensing performance. Supposing, furthermore, that admitting $\{N3, N2\}$ would provide more information, too much communication would be generated (since both sensors would communicate data about $T3$ and $T4$, the new subnet might be saturated by the new traffic). Admitting $\{N1, N3\}$ or symmetrically $\{N4, N2\}$, on the other hand, would increase significantly information quality about $T2$ and $T3$ and would generate an acceptable level of communication, provided $S1$ and $S2$ remain in the subnet with their existing links. Also, $S3$ and $S4$ cannot leave the network as they maintain connectivity for relaying informations on $\{T1, T2, T3\}$. The question is, how is this solution derived, given such constraints as broadcast latency and resilience (fault-tolerance)? This is a constrained optimisation problem crucial to providing QoS.

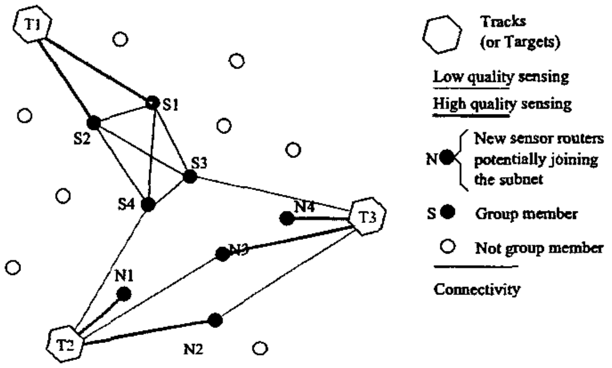

Fig. 1. Distributed tracking with ad-hoc sensor networks

This problem can be solved by a single subnet administrator or leader. However, the sensor network can be lost if the leader fails while making decisions. Therefore, it is necessary to replicate the administrator throughout the network, so that a solution to the subnet management problem can be proposed by any alive sensor router that is part of the subnet, whenever it is required. We discuss later how to reach such a common agreement, based on solutions proposed by each subnet sensor router, even when some of these sensor routers are failing.

D. Decision making criteria

One of the criteria required for admiting a set of sensor routers within the subnet is the measurement of information gain. It models the quality of sensor data and can be directly related to the antenna gain. Other criteria, later considered as hard constraints for admiting a set of sensor routers within the subnet, are essential to provide QoS:

1) Fault-tolerance: The subnet must continue to operate in spite of a maximal number of sensor router immediate stops (later denominated as f_{max}) within a given period of time.

[1]Typically, the time it takes to run a consensus algorithm.

[2]Bound γ includes delays due to retransmissions (for recovering from transmission errors).

2) Latency of reliable broadcasts: The sensing latency (the time after which sensing data are received by all the subnet members) is a very important criterion, which depends on the delays of reliable broadcasts. Overheads due to connectivity maintenance and reliability mechanisms must also be included when modeling the latency criteria.

3) Network and processor utilization: Message loads (sensors and the network) and processing loads must not exceed specified bounds, in order to avoid congestions or thrashing.

III. OUR SOLUTIONS

Constraint Programming has been shown to be an efficient framework for designing solving algorithms that can cope with large scale combinatorial problems. However, the decision making capability that such algorithms provide has to be distributed throughout the network in order to be fault-tolerant. During the decision making process, losses of partial computations arising due to partial failures must be masked. Existing distributed constraint satisfaction algorithms, like Asynchronous Weak Commitment Search [Yok95] do not address fault-tolerance, but rather data privacy. Essentially, the novelty of our approach lies with combining Constraint Programming and Distributed Fault-Tolerant Agreement.

In our approach, any sensor router proposes a solution to the subnet management problem, using a constraint solving algorithm. Solutions obtained by the different sensor routers can be different for many reasons:

- constraint solving can be interupted at different times (see section § III-B.3), providing solutions of different qualities;
- choices concerning the ad-hoc sensor architecture can generate different instances of the subnet management problem (depending for example on data sensed locally and data collected from other sensor routers);

Hence, a global agreement on the set of proposed solution must be reached, so that a global decision can be applied to the whole subnet, in spite of sensor router failures. In the distributed algorithms community, a lot of work has been conducted over the last ten years in the area of distributed and fault-tolerant agreement [Lyn96]. It turns out that many solutions devised for distributed fault-tolerant agreement are solutions to our problems. Of a particular interest for ad hoc sensor networks, reliable broadcast, atomic broadcast, uniform consensus and uniform coordination belongs to this class of problems. Furthermore, their complexity is vastly smaller, and their speed much higher, than that of solutions strictly based upon application-level semantics.

A. Distributed fault-tolerant agreement

For our purposes, we need to consider two kinds of agreement only. One is known under the name of Consensus, the other under the name of Coordination.

The generic specification of Consensus is as follows. Initially, every processor (sensor router) has some solution to the subnet management problem, which is sent as a Proposition to every other processor (via a broadcast protocol). When "enough" Propositions have been received, a processor chooses one of these Propositions, referred to as a Decision. A Decision must be (1) unique—the same for every correct processor, (2) some initial Proposition. This problem is far from being trivial, when one assumes failures (e.g., a broadcast may be incomplete, hence any two processor "views" may differ by up to f_{max} Propositions).

Uniform ConsensuS (UCS) is Consensus with the additional constraint that the Unicity property (Decision) applies also to processors that are about to fail. In fact, the only variants of agreement problems that make sense in reality are the uniform versions. Until recently, it was believed that the worst-case lower bound for UCS is $(f_{max} + 1)\, D$, where D is the upper bound for interprocessor message passing delays. In [HL02], one shows that the worst-case lower bound can be smaller, and one gives a UCS algorithm that is worst-case time optimal. Basically, for most common cases, i.e. failure-free runs, UCS is achieved in D, which is the absolute worst-case lower bound. The worst-case upper bound for UCS is $d.f_{max} + D$, where d is the maximum delay for detecting sensor router failure (and in practice, $d << D$). These results also apply to reliable broadcast and atomic broadcast [ALT02]. In section § III-C, we show how these constraints can be part of the subnet management problem model.

The generic specification of Uniform CoordinatioN (UCN) is as follows. Initially, every processor has some solution (the result of a constraint solving algorithm, in our case), which is sent as a Contribution to every other processor (via a broadcast protocol). When "enough" Contributions have been received, a processor performs some computation out of these Contributions (an "aggregation" of partial computations), which results into a Proposition. Then, UCS is to be solved, initial solutions being the computed Propositions. Thanks to the Unicity property, only one "aggregated" (global) computation will be decided system-wide. For most common cases, using optimal UCS, UCN is achieved in $2\, D$, which is the absolute worst-case lower bound.

Clearly, UCS solves our dynamic subnet reconfiguration or subnet membership problems (Propositions are names of sensors "seen" as good new sensor routers, or sensor routers to be excluded from the subnet).

UCS and UCN rest on constructs known under the name of Unreliable Failure Detectors [CT96], denoted FDs. One may implement FDs by having every processor broadcast "I am alive" messages more or less periodically. Whenever a processor p has not been "heard of" in time by processor q, p is added to the list of "suspects" maintained by q. Lists of "suspects" are inconsistent, usually. Various FD semantics have been defined, the most popular being the Strong FD and the Perfect FD. Surprisingly (maybe), even though lists of suspects may be different for any two processors, it is possible to solve agreement problems with FDs.

Fast FDs [HL02], UCS and UCN algorithms are currently being implemented by a satellite manufacturer, to be delivered to ESTEC (European Space Agency) summer 2003, and made available to the European space industry.

B. Using Constraint Programming

Informally, the combinatorial structure of the subnet configuration problem can be compared to constrained multi-knapsack, transhipment or perfect coupling problems. Extensions of Constraint Programming (CP) that include Operation Research methods have been demonstrated to be powerful frameworks for formulating and solving such difficult multi-knapsacks problems. By combining the modeling capabilities of CP, arc-consistency techniques, branch and bound and efficient dynamic solving methods, it is possible to cope effectively with difficult problem instances.

1) Constraint-based modeling: Model-based constraint solving (MBS) exploits different representations of a problem structure. The modelling process starts by capturing the problem invariants, and refines this using several approximations, sufficient conditions and abstractions. A model consists of constraints and mathematical variables that represent partially or totally the problem to be solved. By logical composition of multiple models, partially overlapping, a global formulation of the whole problem emerges. When considered separately, each model can be viewed as a relaxed formulation of the global problem. An MBS approach is required to formulate and compose the different dimensions of QoS. A cost function can be part of the modelling, generally relying on branch and bound algorithms. Models and constraints defining the subnet management problems are described in section (§ III-C).

2) Constraint solving: In constraint programming, inference involves propagation algorithms, such as Arc-Consistency (AC). These algorithms perform domain filtering by maintaining feasible assignments (AC4), or by reducing variable domains (AC5), both of which can be performed in polynomial time. However, to solve the subnet management problem, such methods must be supplemented with other techniques, e.g. column generation or dynamic programming using graph structures. Arc-consistency algorithms are incomplete since they stop at a fixed point when the set of variables becomes arc-consistent, which does not guarantee that a solution exists in the resulting variables domains. To filter consistent variable assignments, hybrid algorithms are developed.

3) Hybrid search algorithms: Therefore, search algorithms have to be designed for completing constraint solving, by exploring consistent or inconsistent assignment of variables. Using a CP framework, solving methods consist in filtering consistent assignments of the set of variables, by exploring search trees. Optimization schema, such as branch and bound or iterative deepening can also be combined with the solving method.

In addition, these search algorithms must operate in a dynamic environment, where part of the solution already exists and has to be adapted to a new input. This is required when adapting the subnet to environment changes and traffic demand. Anytime search methods can provide a solution of increasing quality. Examples of such methods are:

- Probe Backtracking (PB)[SW00], which repairs a tentative assignment to the set of variables. Such assignments are calculated by relaxing some of the model constraints. PB has been used for solving traffic placement problems in wide area networks.
- Repair Algorithms (RA) [RR00] [JRR94], which approximate incrementally feasible variable assignments by minimizing the degree of inconsistency. A repair algorithm typically starts from an existing, but inconsistent solution. Repair methods are widely used in the planning and scheduling communities.
- Valued Constraint Satisfaction Problems (VCSP), which associate a priority to each constraint model. Constraints are satisfied in an order compliant with priorities. VCSP is used to solve aerospace planning problems (considering highly dynamic environment). VCSP belongs more generally to the class of Soft Constraints, that finds many applications in the industry.

Such algorithms proceed incrementally and can be interrupted at any time, providing a solution of a given quality. This approach is likely to provide the requisite flexibility for subnet management to be executed in a dynamic environment. Section (§ III-C.5) describes such algorithms.

C. Overview of the solving method

Both problem model and search algorithm are developed and experimented in ECLiPSe [WNS97] that provides a constraint solving library based on AC4 and AC5 as well as facilities to design incremental search techniques.

Various options can be envisaged to model the subnet management problem. For illustrating our approach, we give the main constraints that model subnet problem, with the network sensor gain as a cost function.

Decision variables $X_p \in \{0,1\}$, $0 \leq p < N$, model the set of sensor routers that may be admitted in the group. When X_p is positive, sensor router p is considered active subnet member. Additional decision variables $Y_t^p \in \{0,1\}$, $0 \leq p < N$, $0 \leq t < T$ model targets allocated to processors. Constant T is the number of targets detected when solving the management problem. When Y_t^p is positive, target t is allocated to sensor router p.

1) Allocation constraints: Allocation constraints guarantee the consistency of target allocation in the following terms:

Each target is allocated to an active subnet member of the subnet:

$$\forall t, p\ Y_t^p \leq X_p$$

Each subnet active member can follow a limited number of targets $T_{max} \in \mathbb{N}$ (which is a constant) at the same time:

$$\forall p,\ \sum_{t=0}^{T-1} Y_t^p < T_{max}$$

2) Connectivity constraints: For simplicity, only connectivity and delay constraints are modeled, but bandwidth capacity constraints can also be given in similar ways: [3]

To assume a fully connected topology, the distance between every pair of active subnet member does not exceed constant $C \in \mathbb{R}$ (where euclidean distances $d(p,p') \in \mathbb{R}$ between each pair of sensor routers (p,p') are considered constant and known at problem solving time):

$$\forall p, p',\ min(X_p, X_{p'}).d(p,p') < C$$

3) Duration and fault-tolerance constraints: Duration D_p for broadcasting target informations from active member p must be lower than a specified amout of time D_{max}. Communication time depends on data volume (e.g. related to the number of tracks and an arbitrary constant $K \in \mathbb{R}$) and the maximal distance between two active subnet members. This leads to the following non-linear constraint:

$$\forall p,\ D_p < D_{max},$$
$$D_p = max_{p,p'}(min(X_p, X_{p'}).d(p,p')).K.\sum_{t=0}^{T-1} Y_p^t$$

In case of reliable broadcast, worst case duration of reliable broadcast B is insured using failure detection delay d. Reliable broadcast must not exceed a specified latency B_{max} (again, constant $K' \in \mathbb{R}$ is arbitrary given, and $K' << K$).

$$\forall p,\ B_p < B_{max},\ B_p = D_p + f_{max}.d,$$
$$d = max_{p,p'}(min(X_p, X_{p'}).d(p,p')).K'$$

Similar constraints can also be stated to consider the response-time of UCS or UCN.

4) Maximizing ad hoc sensor gain: Finally, the global gain G of the network is maximized using the following linear function:

$$G = \sum_{p=0}^{N-1} \sum_{t=0}^{T-1} g.\frac{Y_p^t}{d(p,t)^2}$$

where $d(p,t) \in \mathbb{R}$ is the euclidean distance between the sensor router and the target, actually considered known and constant at problem solving time. Constant g is an elementary gain.

[3]Although we give a high-level description of the constraints, they might differ significantly according to low-level systems and implementation.

5) Constraint solving method: The constraint solving algorithm must instantiate variables X_p and Y_t^p in order to maximize the network sensor gain G. These variables constitute a Proposition to be processed by UCS or UCN. We assume that the same solving algorithm is replicated on each sensor router. The algorithm uses a combination of branch and bound, RA and PB. Probing values are derived from the solution that has been consistent before the subnet environment changed. This provides an anytime behavior to the solving technique.

Inconsistent assignments are repaired using a depth-first search algorithm. Figure 2 illustrates the trade-offs between search response time and completness on a specific problem instance (5 tracks and 6 sensor routers). An optimal solution is found after 5 seconds using a complete repair search, but its rate of improvement is lower than a non-complete repair search.

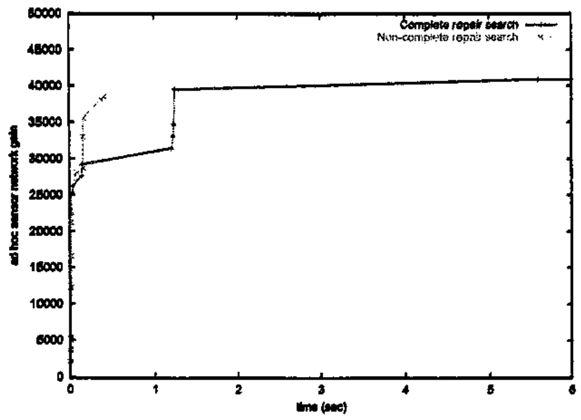

Fig. 2. Solving the subnet management problem with repair algorithms

Figure 3 represents the gain of an ad hoc sensor network subject to a set of failures. These results are obtained by simulation. Each event represents a solving step followed by a distributed agreement. Both algorithms keep the network alive, but the complete repair search provides better network optimization throughout the execution.

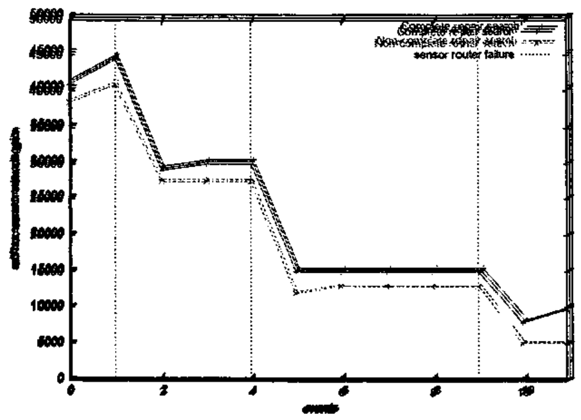

Fig. 3. Impact of subnet management on sensor network gain in presence of sensor router failures (14 targets, 20 sensor routers)

IV. Conclusions

We have proposed an approach to ad hoc sensor network management which combines constraint solving techniques and fault-tolerant distributed agreement algorithms. To provide dynamic subnet management decisions in spite of sensor router failures, we have introduced a repair-based solving method that is distributed over the network using UCS or UCN. Lastly, in order to provide Quality of Service, we have also illustrated how to model connectivity and latency constraints resulting from sensor data broadcasting, UCN and UCS. Our approach provides multiple methods for designing ad hoc sensor network architectures as well as new ways to compromise fault-tolerance and performances.

Acknowledgement

We thank Prof. Barry Richards for his constructive comments and his strong support writing this paper.

References

[ALT02] M. Aguilera, G. Le Lann, and S. Toueg. On the impact of fast failure detectors on real-time fault-tolerant systems. In *Proc. of DISC'02*, 2002.

[CJL+01] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum, and L. Viennot. Optimized link state routing protocol. In *Proc. of IEEE INMIC*, Pakistan, 2001.

[CT96] T. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, March 1996.

[FCGS03] M. Fromherz, L. Crawford, C. Guettier, and Y. Shang. Distributed adaptive constrained optimization for smart matter systems. *AAAI Spring Symposium on Intelligent and Distributed Systems*, March 2003.

[Gue97] C. Guettier. *Optimisation globale du placement d'applications de traitement du signal sur architectures parallèles utilisant la programmation logique avec contraintes*. PhD thesis, École des Mines de Paris, 1997.

[HL02] J-F Hermant and G. Le Lann. Fast asynchronous uniform consensus in real-time distributed systems. *IEEE Transactions on Computers*, 51(8):931–944, August 2002. Special Section on Asynchronous Real-Time Distributed Systems.

[JRR94] J. Jiang, T. Richards, and B. Richards. No-good backmarking with min-conflict repair in constraint satisfaction and optimisation. In *Proceedings of Principles and Practice of Constraint Programming 94*, pages 36–48, Seattle, 1994.

[LS03] G. Le Lann and U. Schmid. How to maximize computing systems coverage. submitted for publication, April 2003.

[Lyn96] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann Pub, 1996.

[MKPS01] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *INFOCOM*, pages 1380–1387, 2001.

[RR00] T. Richards and B. Richards. Nonsystematic search and no-good learning. *Journal of Automated Reasoning*, 24:483–533, 2000.

[SW00] H. El Sakkout and M. Wallace. Probe backtrack search for minimal perturbation in dynamic scheduling. *Constraints, Special Issue on Industrial Constraint-Directed Scheduling*, 5(4), 2000.

[WNS97] M. Wallace, S. Novello, and J. Schimpf. Eclipse : A platform for constraint logic programming. *ICL Systems Journal*, 12, 1997.

[Yok95] M. Yokoo. Asynchronous weak-commitment search for solving distributed constraint satisfaction problems. In U. Montanari and F. Rossi, editors, *Principles and Practice of Constraint Programming - CP'95*, pages 88–102. Springer, Berlin,, 1995.