Energy-Efficient Task-Mapping for Data-Driven Sensor Network Macroprogramming Using Constraint Programming

Farshid Hassani Bijarbooneh, Pierre Flener, Edith Ngai, and Justin Pearson

Department of Information Technology
Uppsala University, Box 337, SE – 751 05 Uppsala, Sweden
{farshid.hassani, pierre.flener, edith.ngai, justin.pearson}@it.uu.se

Abstract Using constraint programming (CP), we address the task-mapping problem in data-driven macroprogramming for wireless sensor networks (WSNs). A task graph representing the flow of data among tasks assists the application developer to specify the features of a WSN at a high level of abstraction. A problem that arises in this context is how to map the tasks to nodes in the target network before the deployment of sensors, in order to achieve an energy-efficient WSN. We take a published formulation of the task-mapping problem solved by mixed integer programming (MIP) solvers, and rewrite it as a constraint program. We minimise the maximum energy spent by each node for real-life instances of the problem, and show that our CP model results in significantly better runtimes than the MIP model.

1 Introduction

Wireless sensor networks (WSNs) operate as distributed systems where sensors cooperatively monitor or control a condition in an environment, such as temperature, speed, or pressure [1]. Nodes in a WSN consist of a processor, a radio transmitter, and a battery. It is of great concern to reduce the energy consumption at each node before deploying the network to the environment. The lifetime of a WSN depends critically on the energy consumption of the nodes, especially in cases where the battery cannot be charged once it is drained [4].

In a WSN, a *node* is assigned to perform several tasks. *Tasks* are pieces of code implementing applications of the network. The entire network repeats the same behaviour over a time period called a *round*. Nodes have an initial energy level, which drops as the they communicate and process the assigned tasks in each round of the network [9].

Tasks are grouped in three categories: sensing tasks, operative tasks, and actuator tasks. *Sensing tasks* call a sensor to collect data at each round. For example, invoking a sensor to measure temperature in a room is performed by a sensing task. *Operative tasks* perform operations on data that has been gathered by the sensing tasks. Taking the temperature from differently positioned sensors in a room and computing the average is an example of such a task. Finally,

actuator tasks perform an action to affect the environment, which is based on data processed by operative tasks. For example, consider a task that turns on a heater to warm the room. Sensing tasks and actuator tasks can only run on nodes with the appropriate sensors or actuators, while operative tasks are free to run on any node with sufficient computational resources. Programming a task for each sensor individually is a very time consuming process. We favour a methodology that allows defining and deploying tasks regardless of WSN architecture. In the WSN context, *data-driven macroprogramming* refers to an approach that facilitates sensor network programming by specifying the features of a WSN as a task graph representation [2,6]. Using this method, we create a task graph based on data flow that is independent of the network topology.

Data-driven macroprogramming poses many challenges including how to map the task graph efficiently onto the nodes in a WSN. A task mapping can be made more efficient by reducing the energy consumption. Note that task mapping is not only initiated at the deployment of the macroprogram, but also at any time during the operation of the WSN if the energy level of a node drops under a certain level.

In this paper, we optimise the task-mapping process in a multi-hop [1] heterogeneous [8] WSN to achieve overall minimum energy consumption by the sensors. A *multi-hop* WSN allows several nodes to forward data toward their destination, so nodes have to consider routing information. A *heterogeneous* WSN is a network that is able to provide several wireless services simultaneously.

For the mapping process, we consider task computation costs, data rates, node routing costs, and placement constraints with regard to a directed acyclic graph representing a data-driven task graph. Our kind of task graph and the general modelling framework are the same as those in [2,11]. We take the framework and instance data related to two real-life applications of the problem: the monitoring of heating, ventilation, and air conditioning [3] and highway traffic management [7]. We implement our model as a constrained optimisation problem. Our approach follows the non-linear mathematical formulation of [11] and rewrites it into a constraint programming (CP) model. We show that our CP model achieves at least an order of magnitude speedup compared to the MIP model.

2 The Problem and Some Applications

In a task mapping problem, we need to consider the cost of routing a message between any two nodes via a particular node, which means that on a WSN with n nodes we will have a cost routing table of size n^3. The combinatorial problem is to map optimally the tasks with regard to such three-dimensional cost routing matrix, as well as considering the cost of sensing and processing tasks. A current method for optimal task mapping involves modelling and solving it uses an MIP solver [11], which shows poor performance on moderate-size real-life instances of the problem (up to 124 nodes and 80 tasks).

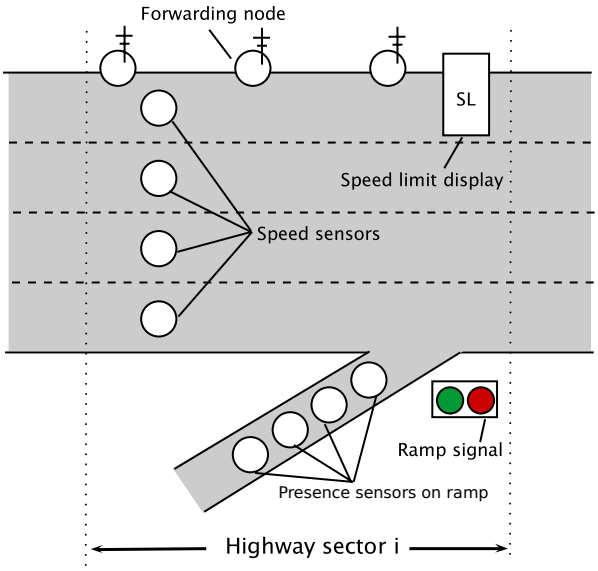

Figure 1. Highway traffic management system

We investigate two real-life applications. In both examples we are given a data-driven task graph and a set of nodes of different types. The data-driven task graph is then preprocessed by the *Srijan macroprogramming toolkit* [11] to provide a set of constraints for each node on which tasks can be run and what are the possible data flows. The goal is then to find a mapping that satisfies these constraints and minimises the energy consumption of the network.

The first application is a highway traffic management system where the aim is to reduce the congestion of vehicles on a highway by controlling speed limits and vehicle access to the highway. Figure 1 represents one sector of a highway. There are uniformly distributed forwarding nodes on the edge of the highway to let the sectors communicate. Speed sensors are randomly distributed on the four lanes so that each is in the range of another speed sensor or a forwarding node. The speed sensor on each lane senses the speed of passing vehicles, a presence sensor indicates the presence of a vehicle on the ramp, and a red/green signal on the ramp controls vehicle entry to the highway. Figure 2 depicts the specification of data-driven task graph for one sector of a highway. Nodes in this graph represent a task and the arcs indicate flow of data between the tasks. The tasks `SpeedSampler` and `RampSampler` call a sensor to capture the speed or presence of a vehicle respectively. The sensing tasks send the sensed data to the operational tasks. There are four operational tasks: `AvgQueueLength` takes the average number of vehicles waiting in a queue per round, `AvgSpeed` takes the average speed of vehicles on the lane, `RampSignal` takes the average queue length from both neighbour sectors and decides whether it is necessary to use the ramp signal or not, and `SpeedLimit` takes data from the current and neighbouring sectors and

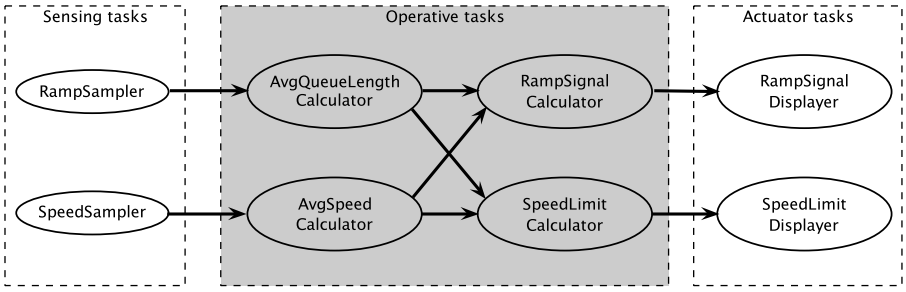

Figure 2. Specification of data-driven task graph for one sector of a highway in the traffic problem

calculates the speed limit, which is shown on the display. We experiment on two instances of this problem: the first instance includes 74 nodes and 36 tasks, and the second instance includes 124 nodes and 60 tasks.

The second application is building environment management for monitoring of heating, ventilation, and air conditioning (HVAC). This is similar to the highway traffic problem, but here the sensing tasks are humidity sampling and temperature sampling, which are fixed to some nodes, and the actuator tasks are fixed to selected nodes to control humidity and temperature. The operative tasks collect the sampled data from sensors, compute averages, and respond with a proper action for the actuator nodes. We experiment on two instances of this problem: the first instance includes 7 nodes and 6 tasks, and the second instance includes 106 nodes and 80 tasks.

The operative tasks defined for the two applications are free to be mapped to any nodes. We minimise the energy consumed by each node, based on the communication cost for gathering the collected data from all the other nodes, processing the operations, and sending the results to the actuators. Note that, once tasks are mapped to nodes, we can compute the energy spent by each node in one round. The energy for a node is equal to the number of times the node runs a task, multiplied by the size of the data being sent by the node, multiplied by the cost of routing one unit of data by a node.

3 Model

We present a constraint model of the task mapping problem by formulating it as the non-linear system of [11], which was linearised in [11] so that it could be solved by MIP solvers. In our CP model, we model the non-linear equations with binary expressions and logical operators, and solve our model using *Gecode* [5], an open-source modular object-oriented programming platform for constraint-based global search.

3.1 The Decision Variables

Let t and n be the number of tasks and nodes respectively, and let $v_1, v_2, \ldots, v_t$ be integer decision variables in the node domain $[1, \ldots, n]$, such that v_i denotes the node that is associated with task i. Let $x = \{x_{ip} \mid 1 \leq i \leq t, 1 \leq p \leq n\}$ be a two-dimensional array of redundant 0/1 decision variables, such that the value of x_{ip} is 1 if and only if node p is associated with task i (that is, $v_i = p$). In the MIP model, the solver directly operates on the x variables, whereas in the CP model, we branch on the v variables and maintain the relationship between the v and x variables with channelling constraints. Throughout this paper, the variables always have lowercase identifiers, the constants always have uppercase identifiers, the indices i, j always refer to a task, and the indices p, q, r always refer to a node.

As explained in Section 2, a sensing task or an actuator task may be fixed to some nodes, and therefore it cannot be mapped to any other nodes. This limit is a placement constraint that is reflected in the model by dropping value p from the domain of variable v_i for every task i that cannot be mapped to node p (in other words, constraining x_{ip} to be 0 for all tasks i that cannot be mapped to node p).

Let e_p be an array of integer decision variables in the domain $[0, \ldots, \infty]$, such that e_p is the total energy spent by node p according to the tasks mapped to it by the v_i variables.

3.2 The Problem Formulation

The total energy e_p spent by node p in one round consists of two terms: the communication cost and the computation cost. We can ignore the computation cost, since it has a far smaller value in our problems than the communication cost. The communication cost e_p for each node p is the sum of the costs of routing messages between every pair of nodes that are communicating according to the data-driven task graph at each round in the system:

$$e_p = \sum_{(i,j) \in A} F_i \cdot S_{ij} \cdot R_{v_i v_j p} \quad \forall p \ (1 \leq p \leq n) \tag{1}$$

where T is the set of tasks taken from the specification of data-driven task graph, and $A \subseteq T \times T$ is the set of arcs in the instantiated data-driven task graph, representing the flow of data from task i to task j. Task i is fired F_i times at each round, and S_{ij} is the size of the data from task i to task j. The energy consumed by node p while routing one unit of data from node q to node r is denoted by R_{qrp}. The indices v_i and v_j in (1) on the R matrix are decision variables mapping arbitrary task i and task j to node q and node r, respectively. Intuitively, e_p is the cost for a particular mapping given by v and it can also be calculated by considering *all* possible nodes q and r for v_i and v_j in (1) and multiplying by $x_{iq} \cdot x_{jr}$ to indicate whether there is a message routed from node

q to which task i is assigned to node r to which task j is assigned via node p:

$$e_p = \sum_{(i,j)\in A} \sum_{q=1}^{n} \sum_{r=1}^{n} F_i \cdot S_{ij} \cdot R_{qrp} \cdot x_{iq} \cdot x_{jr} \quad \forall p \ (1 \leq p \leq n) \tag{2}$$

The optimisation metric is defined by minimising the maximum fraction of the initial energy spent by any node in the system at a round. This metric is defined regarding the fact that, in task-mapping for WSNs, we aim at maximising the time to reconfiguration (TTR), which is the amount of time after which the energy level of some node goes below a fraction of its initial energy, denoted by E_p. By minimising the maximum fraction of energy spent by all nodes, the value of TTR will be maximised. Thus, the objective function to be minimised is the following:

$$\max_{1\leq p\leq n} \ \frac{1}{E_p} \cdot e_p \tag{3}$$

3.3 The Constraints and Objective Function

For a MIP solver, constraint (2) is non-linear and needs to be replaced by an equivalent linear constraint. As shown in [12], an extra 4D array of binary variables y_{ijqr} can be introduced to replace $x_{iq} \cdot x_{jr}$ and channelling constraints (not shown here) must then be added to keep the relationship between the x variables and the y variables.

Our CP model changes the non-linear part of (2) into a binary expression as follows:

$$e_p = \sum_{(i,j)\in A} \sum_{q=1}^{n} \sum_{r=1}^{n} F_i \cdot S_{ij} \cdot R_{qrp} \cdot (x_{iq} \wedge x_{jr}) \quad \forall p \ (1 \leq p \leq n) \tag{4}$$

The conjunction constraint for binary operator $\wedge$ (logical and) is here more efficient than the constraint *mult* in *Gecode*. This model does not require explicitly defining a variable to replace $x_{iq} \wedge x_{jr}$. We do not need to either implement constraint (4) by a direct use of the *linear* constraint, since we would then also need to linearise the non-linear form by creating a 4D array of redundant variables to represent $x_{iq} \wedge x_{jr}$ (or $x_{iq} \cdot x_{jr}$).

We let the cost c ($0 \leq c \leq 1$) be a floating-point decision variable that holds the maximum fraction of the initial energy E_p spent by every node p in the network at each round. To perform the minimisation of (3), we constrain the total energy e_p of each node p to be at most equal to the cost c, subject to minimising c. This contributes to better propagation by relating the cost c to a single decision variable e_p for each node p. The cost c takes the maximum value for the energy spent by any node, so by minimising c we actually minimise the objective function (3). As a result, the following constraints are posted:

$$\frac{1}{E_p} \cdot e_p \leq c \quad \forall p \ (1 \leq p \leq n) \tag{5}$$

We assume that the initial energy E_p for all nodes p is the same. Therefore, E_p does not affect the resulting cost c in the model, so we can avoid the fraction in (5) and revise the domain of c to $[0, \ldots, E_p]$, which is an integer domain:

$$e_p \leq c \quad \forall p \ (1 \leq p \leq n) \tag{6}$$

A constraint is necessary to enforce that each task is mapped to exactly one node:

$$\sum_{p=1}^{n} x_{ip} = 1 \quad \forall i \ (1 \leq i \leq t) \tag{7}$$

Finally, we maintain the relationship between the node decision variables v_i and the binary variables x_{ip} by channelling between them as follows:

$$v_i = p \iff x_{ip} = 1 \quad \forall i, p \ (1 \leq i \leq t, 1 \leq p \leq n) \tag{8}$$

We can implement the channelling constraint (8) by either the *element* constraint, reification, or the *channel* constraint of *Gecode*. In the reification implementation, the constraints (7) are implied and we can ignore them, as they slow down propagation, whereas for the other cases we need to keep (7), since further propagation is required to prune the domain of x_{ip} to 0 if and only if $v_i \neq p$. In our CP model, reification both with and without implied constraints (7) performs faster than the *element* constraint, but reification performs slower than the *channel* constraint. In this paper, we only present the results based on the channelling implementation of (8).

The search procedure branches on the v_i variables only. It uses the first-fail strategy for variable selection, and selects values greater than the mean of smallest and largest values in the domain of the selected variable.

4 Experiments

We experimented with two representative instances (chosen from [11]) each for the two applications mentioned in Section 2, the HVAC and highway traffic management problems. We chose the smallest size (HVAC $\langle 7, 6 \rangle$), medium size (highway traffic $\langle 74, 36 \rangle$), and largest size (HVAC $\langle 106, 80 \rangle$ and highway traffic $\langle 124, 60 \rangle$). Our CP model is implemented in *Gecode* (revision 3.2.2) and runs under Mac OS X 10.6.3 64 bit on an Intel Core 2 Duo 2.53 GHz with 3MB L2 cache and 4GB RAM. We set a timeout of one hour for each instance, recording the best minimum cost achieved and the time at which it has been reached, if solving times out.

We also solved our instances using the MIP solvers *Gurobi* (revision 2.0.2),[1] *SCIP* (revision 1.2.0),[2] and *lpsolve* (revision 5.5)[3], taking the linearised formulation of (2) to solve under the same system configuration and experimental

[1] available from `http://gurobi.com`
[2] available from `http://scip.zib.de`
[3] available from `http://lpsolve.sourceforge.net`

	HVAC				Highway Traffic			
⟨nodes, tasks⟩	⟨7, 6⟩		⟨106, 80⟩		⟨74, 36⟩		⟨124, 60⟩	
time and cost	time	cost	time	cost	time	cost	time	cost
Gecode	0.01	10.00	0.73	12960	1.74	**300**	13.35	**300**
Gurobi	0.29	10.00	0.01	12960	44.00	**300**	212.00	**300**
SCIP	0.14	10.00	3.22	12960	≤3600.00	**320**	≤3600.00	**300**
lpsolve	0.04	10.00	≤3600.00	**12960**	≤3600.00	**320**	≤3600.00	**360**

Table 1. Optimisation results of different solvers for four instances of the HVAC and highway traffic management task mapping problems. The time unit is seconds. The boldface values indicate best costs achieved before a one-hour timeout.

set-up. We chose *lpsolve* only since it is the solver used in [11]. We also chose to experiment with *Gurobi* and *SCIP*, as these two solvers are among the fastest commercial and non-commercial MIP solvers respectively (according to the *SCIP* home page).

The instance data is stored both in an internal format used by our CP model and in the MPS file format [10]. MPS is a standard column-oriented format for storing linear programming models for MIP solvers.

In Table 1, we present the results of our CP model compared to the model solved by the three MIP solvers. The problem was optimally solved for the two instances of the HVAC problem (except for HVAC ⟨106, 80⟩ using *lpsolve*), and the runtimes indicate the total time to optimally solve the two instances of the HVAC problem. For the highway traffic management instances, the boldface values indicate a best cost achieved before timeout, and the runtime to arrive at this cost. Note that *lpsolve* and *SCIP* only report the best cost *after* timing out, and we do not know when this cost was first found.

In the second HVAC instance, an MIP solver, namely *Gurobi*, is faster than *Gecode*; that is because *Gurobi* managed to solve the problem in one simplex iteration. We observe that, in the rest of the instances, *Gecode* can reach the same solution about 90% faster than the fastest MIP solver we used, *Gurobi*. The CP model is also always faster than the MIP approach in [11]. These results show that constraint programming can efficiently solve the problems introduced in task mapping for WSNs. It is also worth mentioning that MIP solvers use a pre-solve phase to eliminate as many variables and constraints as possible before solving the actual problem. For example, pre-solve in the hardest instance (highway traffic ⟨124, 60⟩) using *Gurobi* is taking 22 seconds, which is included in the runtime of 212 seconds in Table 1, and this is already longer than the total time spent by *Gecode* to find the same solution.

Table 2 presents the complexity of the MIP model in terms of the numbers of variables and constraints according to *Gurobi*. The second row shows the same numbers after the pre-solve phase.

	HVAC				Highway Traffic			
⟨nodes, tasks⟩	⟨7, 6⟩		⟨106, 80⟩		⟨74, 36⟩		⟨124, 60⟩	
	#vars	#cons	#vars	#cons	#vars	#cons	#vars	#cons
MIP size	78	118	16855	25308	21449	56462	50737	130072
presolved size	43	2	8481	88	20857	54662	49745	127056

Table 2. Complexity of the MIP model in terms of the numbers of variables and constraints according to *Gurobi*

	HVAC		Highway Traffic	
⟨nodes, tasks⟩	⟨7, 6⟩	⟨106, 80⟩	⟨74, 36⟩	⟨124, 60⟩
number of propagators	64	62395	294952	1164526
memory (KB)	19	17610	89734	641871
runtime	0.01	0.73	1.74	13.35

Table 3. Complexity of the CP model in terms of the number of propagators as well as the memory (in KB) and runtime (in seconds) taken by *Gecode*

Table 3 presents the complexity of the CP model in terms of the numbers of propagators and the memory allocated by *Gecode* (in KB). It also includes the runtimes according to Table 1, indicating that an increase in the number of propagators is proportional to the runtime increase of the solver.

5 Conclusion

Macroprogramming for WSNs is an evolving area, where efficiency of a WSN can benefit considerably by task mapping in a configuration phase, as well as in reconfiguration when the energy level of a node drops below a certain amount.

5.1 Summary

We presented a CP model for the task mapping problem in a multi-hop heterogeneous WSN. Our model involves placement constraints and the cost of routing. The optimisation is done by minimising the maximum energy spent by each node. We model the non-linear formulation of [11] effectively using binary expressions and show that it is competitive with a published MIP model.

5.2 Related Work

To the best of our knowledge, there has been no work addressing a CP approach for a general case of task mapping in a multi-hop WSN achieving energy optimisation under routing costs.

The main related work to ours is [11], where the problem specification and formulation are taken from; we took the real-life instances from that work to compare with MIP solvers. However, we use CP to capture directly the non-linear constraints. In [11], two objectives are used: minimising the maximum energy spent by all nodes, namely *energy balance*, and minimising the sum of the energy spent by all nodes. Energy balance is a better metric, since it maximises the time to reconfiguration. Minimising the total energy spent is the classical metric.

In [13], the authors only investigate *single*-hop *homogeneous* WSNs. We consider *multi*-hop *heterogeneous* networks including the routing costs. That work, unlike ours, also considers the minimum schedule length subject to energy consumption constraints.

5.3 Future work

We believe it is possible to model the problem using a pure CP approach and without considering the MIP formulation.

We expected that a simple and fast heuristic such as first-fail would give a better result in a shorter runtime. On the other hand, a more advanced branching that takes the task graph and the target network features into account may result in a faster solution or a better cost under the one hour timeout.

The computation cost of tasks was ignored both in our CP model and in experiments with the MIP model of [11], since the computation cost for invoking a task by a node, in our data set, is far smaller than the communication cost of routing the message. However, it might be of interest to investigate the impact of computational cost presented in the model.

Acknowledgements. This research is sponsored by the Swedish Foundation for Strategic Research (SSF) under research grant RIT08-0065 for the project *ProFuN: A Programming Platform for Future Wireless Sensor Networks*. We thank Animesh Pathak of INRIA Paris-Rocquencourt (France) for useful discussions and the datasets.

References

1. I. F. Akyıldız, W. Su, Y. Sankarasubramaniam, and E. Cayırcı. Wireless sensor networks: A survey. *Computer Networks*, 38(4):393–422, 2002.
2. A. Bakshi, V. K. Prasanna, J. Reich, and D. Larner. The abstract task graph: A methodology for architecture-independent programming of networked sensor systems. In *EESR'05: Proceedings of the 2005 workshop on End-to-End, Sense-and-Respond systems, applications and services*, pages 19–24, Berkeley, CA, USA, 2005. USENIX Association.
3. M. Demirbaş. Wireless sensor networks for monitoring of large public buildings. *Computer Networks*, 46:605–634, 2005.
4. S. C. Ergen and P. Varaiya. Energy efficient routing with delay guarantee for sensor networks. *Wireless Network*, 13(5):679–690, 2007.

5. Gecode Team. Gecode: Generic constraint development environment, 2006. Available from `http://www.gecode.org`.
6. R. Gummadi, O. Gnawali, and R. Govindan. Macro-programming wireless sensor networks using KAIROS. In V. K. Prasanna, S. S. Iyengar, P. G. Spirakis, and M. Welsh, editors, *Distributed Computing in Sensor Systems (DCOSS), First IEEE International Conference*, volume 3560 of *Lecture Notes in Computer Science*, pages 126–140. Springer Berlin, 2005.
7. T. T. Hsieh. Using sensor networks for highway and traffic applications. *Potentials, IEEE*, 23(2):13–16, April–May 2004.
8. D. Kumar, T. C. Aseri, and R. Patel. Energy efficient heterogeneous clustered scheme for wireless sensor networks. *Computer Communications*, 32(4):662–667, 2009.
9. A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *WSNA'02: Proceedings of the 1st ACM international workshop on Wireless Sensor Networks and Applications*, pages 88–97, New York, NY, USA, 2002. ACM Press.
10. J. L. Nazareth. *Computer Solution of Linear Programs*. Oxford University Press, 1987.
11. A. Pathak and V. K. Prasanna. Energy-efficient task mapping for data-driven sensor network macroprogramming. *IEEE Transactions on Computers*, 59:955–968, July 2010.
12. A. H. G. Rinnooy Kan, G. L. Nemhauser, and M. J. Todd. *Optimization*. North-Holland, Elsevier, 1989.
13. Y. Tian, E. Ekici, and F. Özgüner. Energy-constrained task mapping and scheduling in wireless sensor networks. In *IEEE International Conference on Mobile Ad hoc and Sensor Systems*, pages 8–218. IEEE Computer Society Press, 2005.