A Constraint Programming Approach to the Additional Relay Placement Problem in Wireless Sensor Networks

Luis Quesada[1], Kenneth N. Brown[1], Barry O'Sullivan[1], Lanny Sitanayah[2], and Cormac J. Sreenan[2]
[1]*INSIGHT Centre for Data Analytics and* [2]*Mobile and Internet Systems Laboratory*
Department of Computer Science, University College Cork, Ireland
{*l.quesada—k.brown—b.osullivan—l.sitanayah—c.sreenan*}*@cs.ucc.ie*

***Abstract*—A Wireless Sensor Network (WSN) is composed of many sensor nodes which transmit their data wirelessly over a multi-hop network to data sinks. Since WSNs are subject to node failures, the network topology should be robust, so that when a failure does occur, data delivery can continue from all surviving nodes. A WSN is k-robust if an alternate length-constrained route to a sink is available for each surviving node after the failure of up to k-1 nodes. Determining whether a network is k-robust is an NP-complete problem. We develop a Constraint Programming (CP) approach for solving this problem which outperforms a Mixed-Integer Programming (MIP) model on larger problems. A network can be made robust by deploying extra relay nodes, and we extend our CP approach to an optimisation problem by using QuickXplain to search for a minimal set of relays, and compare it to a state-of-the-art local search approach.**

I. Introduction

Rapid improvements in wireless communication and electronics have led to the development of Wireless Sensor Networks (WSNs) for monitoring in many diverse applications, including environmental assessment, fire detection, personal health management, and surveillance. A sensor node is a device with integrated sensing, processing and communication capabilities, and is typically battery powered. A WSN is composed of many nodes, which transmit their data wirelessly over a multi-hop network to data sinks. These networks are prone to failures: the wireless devices are often unreliable, they have limited battery life, transmissions may be blocked by changes in the environment, and the devices may be damaged, e.g. by weather, wildlife or human intervention. For dealing with failures, reliable routing protocols [2], [4], [7], [16] have been proposed, but they rely on a network topology in which alternative routes to a sink are available.

Therefore, one key objective in the planning of a WSN is to ensure some measure of robustness in the topology, so that when failures do occur routing protocols can continue to offer reliable delivery, giving time to the operator to identify and repair the failed nodes. In particular, one standard criterion is to ensure routes to the sink are available for all remaining sensor nodes after the failure of up to $k-1$ other nodes. In addition, since there are sometimes data latency requirements, there may be a limit on the path length from sensor node to sink. Therefore every node in the initial design should have k node-disjoint paths to the sink of length less than the length bound.

To ensure that sensor nodes have sufficient paths, it may be necessary to add relay nodes, which do not sense, but only forward data from other nodes. The possible positions of the relay nodes may be limited, and each position can communicate with only a subset of the other nodes in the network. Finally, installing additional relays comes at a cost thus motivating solutions that minimise the number of additional relays.

In this paper, we present constraint-based solutions to these robust WSN topology design problems. First, we consider the decision problem of whether it is possible, for a given network and a set of candidate relay positions, to find k length-bounded node-disjoint paths to the sink for a single sensor. We classify the problem as being NP-complete, and present Constraint Programming (CP) and Mixed-Integer Programming (MIP) models, and compare to a previously published local search method [14]. We show that the CP model solves the problems in less time than it takes to generate the MIP models. We then consider the extended decision problem, in which we must find qualifying paths for all sensors in the same network. We demonstrate that the CP solution is able to solve these problems in reasonable time, but that the MIP model does not scale. Finally, we consider the optimisation problem, in which the aim is to minimise the number of relays. The MIP model is fastest on small problems but again does not scale up to larger problems. We use QuickXplain [10] to develop an approximate CP solution, which we show is competitive in time with the local search method on the larger problems, although with lower quality solutions.

II. Background and Related Work

A WSN can be modelled as a graph $G = (V, E)$, where V is a set of nodes and E is a set of edges. Each edge connects two nodes that are within transmission range of each other[1], and the two nodes are said to be *adjacent*. A

[1]For simplicity we assume bi-directional links, but this could be easily relaxed by specifying a more complex connectivity graph.

path of length t between two nodes v and w is a sequence of nodes $v = v_0, v_1, \ldots, v_t = w$, such that v_i and v_{i+1} are adjacent for each $0 \leq i < t$. Two nodes are *connected* if there is a path between them. A graph is connected if every pair of nodes is connected.

The problem of deploying relay nodes for increased reliability has long been acknowledged as a significant problem [3], [12], [9], [11]. Greedy Randomised Adaptive Search Procedure for Additional Relay Placement (GRASP-ARP) [14], a recently published local search approach, has been shown to deploy fewer relay nodes with faster runtime compared to the closest known approach [3], [12]. It uses the GRASP stochastic local search method [5], [6], [13] to deploy additional relay nodes for ensuring (k,l)-sink-connectivity, where all sensor nodes have k node-disjoint paths of length $\leq l$ to the sinks. GRASP-ARP works by (i) generating an initial feasible solution by adding relays until sufficient paths are found, (ii) exploring the neighbourhood of the initial solution by adding and removing relays to reduce the number of required relays, and (iii) iterating until a stopping criterion is satisfied [14]. Within each stage, GRASP-ARP uses a CountingPaths algorithm based on the Ford Fulkerson algorithm with node splitting to find k node-disjoint paths. When searching for a new augmenting path, CountingPaths uses a best-first approach, but with a higher priority for nodes already used in a previously obtained path, with the intention of balancing the lengths of the resultant path set. CountingPaths is guaranteed to find k node-disjoint paths if they exist.

If we restrict the Additional Relay Placement decision problem to ensuring robustness for just one of the sensors in the network, it is equivalent to the Bounded Vertex Undirected Disjoint Paths problem (BVUP), i.e., the problem of finding k vertex-disjoint paths in an undirected graph for a given pair of nodes where the paths have bound length, which is known to be NP-Complete [8], although fixed parameter tractable. Additionally, the problem of minimising the number of additional nodes needed to disjointly connect two nodes, given a bound on the length of the paths, is a specific case of the Generalised Constrained Shortest Link-Disjoint Paths Selection, which is also known to be NP-Hard [15].

III. The Bounded Vertex Undirected Disjoint Paths Problem

The main focus of this paper is the decision problem: given a sensor and a sink in a network, do there exist k node-disjoint paths of length no greater than l to the sink? We saw above that this problem is the same problem as the Bounded Vertex Undirected Disjoint Paths problem, which is NP-complete. At first sight, GRASP-ARP offers a solution, since its initialisation phase will continue adding relays until qualifying paths are found, relying on CountingPaths to check the paths. However, CountingPaths is a heuristic algorithm, and does not guarantee to meet the length bound.

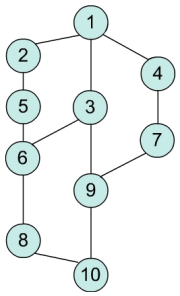


Figure 1. A network for which CountingPaths fails to find two disjoint paths between nodes 1 and 10 of length at most 4.

Consider the problem shown in Figure 1, where the problem is to find two node-disjoint paths of length less than 5 from node 1 to node 10. Counting paths uses a breadth-first search, with nodes ordered by smallest ID. During the first iteration, it finds path A = (1,3,9,10) of length 3. The second search then finds path B = (1,2,5,6,8,10), and the algorithm terminates with the two path lengths of 3 and 5, and thus fails to satisfy the length bound. However, there are two node-disjoint paths of length 4: (1,3,6,8,10) and (1,4,7,9,10). To obtain this solution, the second search would have had to return path C = (1,4,7,9,3,6,8,10), which would produce the correct output when A and C are merged and overlapping segments removed. Path C is not returned, since the breadth-first search reaches node 6 earlier in path B. Similar counter-examples exist for different ordering heuristics.

A. CP approaches

Model. As shown in Figure 2, we model the problem of finding k disjoint paths between a pair of nodes by cloning the source node k times. Finding k disjoint paths is equivalent to finding a tree whose leaf nodes are the clones of the source node, and the root is the target node. Table I shows the constants and variables used in the CP model. Table II shows a CP flow based model for the problem of finding k length-bounded disjoint paths for a sensor s.

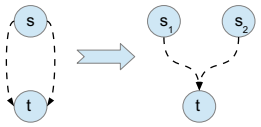


Figure 2. Cloning the source.

As the source node is cloned k times, we use the z variables where z_i denotes the successor of the i-th clone. We post an AllDifferent constraint on the z_i variables to enforce that we end up with k different paths (Constraint

Table I
CONSTANTS AND VARIABLES USED IN THE CP MODELS

Constants	Variables
• V is the set of nodes in the network. • S is the set of sensors. • $R = V - S$ is the set of relays. • k is the lower bound on the connectivity. • s is the sensor (the source). • t is the target. • λ is the upper bound on the length of the paths.	• $y_v \in V$ is an integer variable referring to the successor of node v. • $z_i \in V - \{s\}$ is an integer variable referring to the successor of the i-th clone of s. • $f_{uv} \in \{0, 1\}$ is a Boolean variable referring to the flow on edge $\langle u, v\rangle$. • $l_v \in \{0, \ldots, \lambda\}$ is an integer variable referring to the length of the path from node s to node v.

Table II
CP I: FLOW BASED DECOMPOSITION OF THE TREE CONSTRAINT

ALLDIFFERENT$([z_1, \ldots, z_k])$		(1)
$z_i = v \Rightarrow f_{sv} = 1$	$\forall 1 \leq i \leq k, \forall v \in V$	(2)
$y_u = v \Leftrightarrow f_{uv} = 1$	$\forall u \in V - \{s, t\}, v \in V$ if $u \neq v$	(3)
$\sum_{v \in in(u)} f_{vu} = \sum_{v \in out(u)} f_{uv}$	$\forall u \in V - \{s, t\}$	(4)
$f_{uv} = 1 \Rightarrow l_v \geq l_u + 1$	$\forall \langle u, v\rangle \in E$	(5)

(1)). Constraints (2) and (3) connect the successor variables to the flow variables. We remark that Constraint (2) is an implication and not a double implication since we do not want z_i to be set to v when z_j is set to v (assuming $i \neq j$). That is, having a double implication would be inconsistent with the fact that the z variables need to be set to different values. Notice that Constraint (3) is only posted in those cases where $u \neq v$ thus allowing the successor of an unused node to be bound to itself. In Constraint (4) we use $in(v)$ and $out(v)$ to denote the incoming and outgoing nodes of v. Constraint (4) ensures that the number of used incoming edges of a node is equal to the number of used outgoing edges. In order to ensure that no node is used twice we constrain the incoming and outgoing degrees of each internal node to be at most one. This is implicitly enforced by the y_v variables since a successor variable can only be bound to one value, in conjunction with Constraint (4). Constraint (5) connects the length variables to the flow variables[2]. A solution of the decision problem at hand can be expressed in terms of the y and z variables. The determination of these variables determines the other variables so these are the decision variables in this model.

In what follows, we will call the model of Table II CP I. Alternatively, we can model our decision problem by using one single Tree constraint [1] since both the degree of the nodes and the length of the path in the tree can be constrained through the interface of the constraint. We use CP II to refer to the model obtained using the Tree constraint.

Labelling strategy. During the backtracking search process we construct the paths in a systematic way: we pick one of the clones of the source and decide its successor, then the successor of its successor and we continue this way until we reach the target. Then we pick another clone and do the same until the paths of all clones are found. When all paths have been found, the remaining successor variables are set to their self values. We remark that in constraint programming labelling is interleaved with propagation. During propagation some successor variables may get determined thus avoiding the consideration of those variables during labelling.

In order to implement this dynamic variable ordering, we use the lower bound of the l variables: we pick the y_i variable whose l_i has the highest lower bound. If the highest lower bound is 0, we know that the node is not participating in any path and therefore the value that we pick for y_i is i. Otherwise, the value that we pick for y_i is the one associated with the closest distance to the target aiming at minimising the length of the path[3].

B. MIP approaches

In this section we present our MIP approaches to BVUP. We remark that in MIP we are forced to express all constraints in terms of linear equations, which leads us to models that are more verbose with respect to the CP models. The set of constants and variables of the model is contained in the set of constant and variables of CP. The only difference is that, in MIP, we replicate the set of flow variables per path. More concretely, instead of f_{uv}, we have f^i_{uv}. We also use variables x_v, which are Boolean variables referring to the usage of the nodes.

The model is presented in Table III. Constraints (1) and (2) ensure that flow emanating from the source is 1, for each value of k (i.e., k paths start from the source). Constraints (3) and (4) disable the flow on a given arc if their nodes are not used. Constraint (5) ensures the conservation of the flow. We enforce disjointness by constraining the outgoing flow

[2]We use $\geq$ instead of $=$ because the paths from the source to the target may have different lengths.

[3]The distance matrix is not updated during search so it might happen that the chosen successor is not the closest to the target.

Table III
MIP IV: HANDLING THE UPPER BOUND ON THE LENGTH OF THE PATH VIA REPLICATION OF THE GRAPH

Minimize	obj		
Subject to	$\sum_{v \in in(s)} f^i_{vs} = 0$	$\forall 1 \leq i \leq k$	(1)
	$\sum_{v \in out(s)} f^i_{sv} = 1$	$\forall 1 \leq i \leq k$	(2)
	$f^i_{uv} \leq x_u$	$\forall 1 \leq i \leq k, \forall \langle u, v \rangle \in E$	(3)
	$f^i_{uv} \leq x_v$	$\forall 1 \leq i \leq k, \forall \langle u, v \rangle \in E$	(4)
	$\sum_{v \in in(u)} f^i_{vu} = \sum_{v \in out(u)} f^i_{uv}$	$\forall 1 \leq i \leq k, \forall u \in V - \{s, t\}$	(5)
	$\sum_{v \in out(u), 1 \leq i \leq k} f^i_{uv} \leq 1$	$\forall u \in V - \{s, t\}$	(6)
	$\sum_{\langle u,v \rangle \in E} f^i_{uv} \leq \lambda$	$\forall 1 \leq i \leq k$	(7)

for every internal node (for every value of k) to be at most 1 (Constraint (6)). Because of Constraint (5), Constraint (6) indirectly enforces that the incoming flow for every internal node is at most one (for every value of k). As we are keeping a separate graph per path, the length constraint is enforced by imposing that the used edges in each graph is less than λ (see Equation (7)). As BVUP is a decision problem, the objective (obj) is 1 (i.e., we are just interested in a solution that satisfy the constraints).

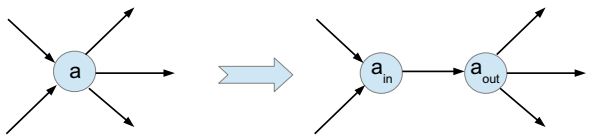


Figure 3. Splitting relay nodes.

Here we are describing the MIP approach that provided us with the best results (MIP IV). However, three other approaches were tried: MIP I, MIP II and MIP III. MIP I is actually an approach to a relaxed version of BVUP where we ignore the length constraint. In MIP II we encode the length constraint by associating an integer variable with each node representing its distance to the source and constraining those variable through conditional constraints. In MIP III we encoded disjointness by splitting the nodes (as shown in Fig 3). That is, instead of constraining the incoming degree and the outgoing degree of the node to be 1, we split each node and set the capacity of the new edge to 1 to enforce that the node only participates in one path (i.e., used once).

IV. EXTENDING TO MULTI-SENSOR BVUP

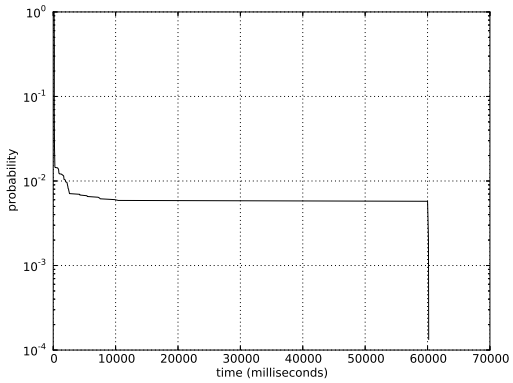
The problem of finding disjoint paths for a selection of sensors to a single target (multi-sensor BVUP) can be decomposed into a set of BVUPs (one per sensor) that can be solved independently. Both the CP and the MIP models can be extended by applying them individually to each sensor.

Results: The instances used in the empirical evaluation are connectivity graphs. To generate them, firstly we generate WSN topologies similar to the technique used in [14]. The two-dimensional network area is divided into grid cells, where one sensor node is placed inside one unit grid square of 8 m $\times$ 8 m and the coordinates are randomly perturbed. This is an approximation of manual deployment of sensor nodes, such as in a building or a city that has regular symmetry. In this simulation, we want the original topologies, i.e. topologies without relays as sparse as possible, because sufficiently dense networks do not need additional relays to guarantee the existence of disjoint paths. In order to get sparse networks (average degree 2–3), we generate more grid points than the number of sensor nodes. For example, we use 6 $\times$ 6 and 11 $\times$ 11 grid squares to randomly deploy 25 and 100 nodes, respectively. Candidate relays are also distributed in a grid area, where a candidate occupies a unit grid square of 6 m $\times$ 6 m. For n25 and n100 topologies, we use 49 and 196 candidate relays, respectively. Both sensor and relay nodes use the same transmission range, i.e. 10 metres. The location of the sink was fixed at the top-left corner of the network. The maximum path length is set to 10 for n25 and 20 for n100 networks.

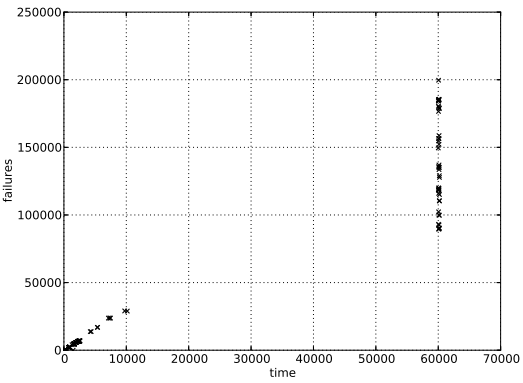
When it comes to the platforms used for the experiments of this section and the next one, the MIP experiments were obtained using CPLEX 12.3 and the version of Choco used in the CP experiments is 2.1.5. The experiments were run on Linux 2.6.25 x64 on a Dual Quad Core Xeon CPU machine with overall 11.76 GB of RAM and processor speed of 2.66 GHz.

The CP approach solves all the n100 instances with average solution time of 511 sec. The MIP models do not scale so well, with the time taken to generate the instances being longer than the solution time for the CP model.

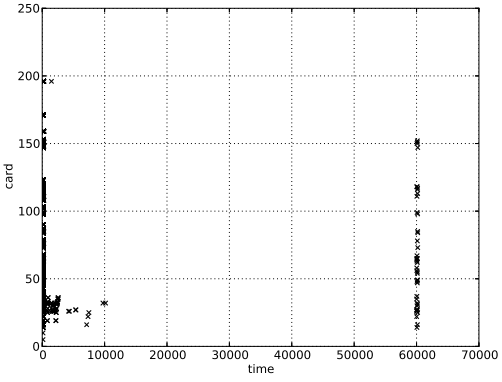
Figure 4 presents the performance of our BVUP approach measured over a set of 7474 instances generated during an execution of our QuickXplan based approach to the Additional Relay Placement problem (explained in the next section), when solving the optimisation problem associated with an instance of 100 sensor nodes and 196 candidate relays. As it can be observed in Figure 4(a), close to 99% of the instances are solved with 10 or less failures. In Figure 4(b) we are comparing the time distribution with the failure distribution. As expected, there is almost a direct correlation between time and failures (ignoring the few cases where we time out). Figure 4(c) relates the time to the



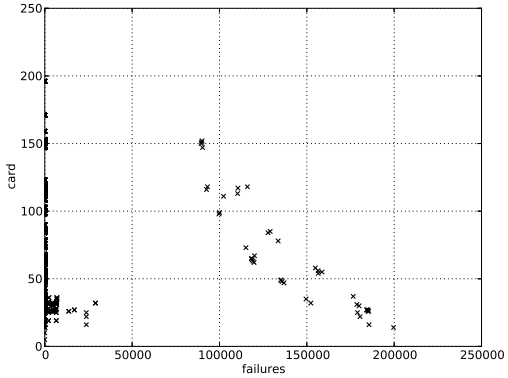
(a) Time distribution.



(b) Time vs Failures.



(c) Time vs Cardinality.



(d) Failures vs Cardinality.

Figure 4. A summary of the performance of our CP approach to BVUP over a selection of 7474 instances generated during an execution of our QuickXplain based approach to the Additional Relay Placement problem when solving an instance of 100 sensor nodes and 196 candidate relays.

cardinality of the set of candidate relays. When focussing on the hard cases, we can observe that there is a high variance with respect to the cardinality thus suggesting that the difficulty of the instances is not correlated to the cardinality of the set. The relation between failures and cardinality (shown in 4(d)) suggests that the speed (i.e., number of failures per unit of time) reduces when we have more relays, which is not surprising since the time spent in propagation increases with the number of relays.

V. Dealing with the Minimisation of Relays

Modelling the minimisation of relays in MIP is straightforward: we just set the objective to the sum of used relays ($\sum_{v \in V-(S \cup \{t\})} x_v$) and replicate the constraints per sensor. In CP we could do something similar: replicate the constraints per sensor, set the objective as done in MIP and use Branch and Bound. However, neither of the two approaches work in practice due to the size complexity of the models. For this reason, we explore another alternative in CP: to compute an approximation to an optimal set of relays by mapping solution of the multi-sensor BVUP problem to conflicts and computing minimal conflicts using QuickXplain.

QuickXplain is an algorithm designed to compute preferred explanations and relaxations for over-constrained problems [10]. In its basic form the algorithm receives a set of inconsistent constraints S and returns a subset C, which corresponds to a conflict of S. C is a minimal set in the sense that C is still inconsistent but the removal of any of its constraints will make it consistent. QuickXplain outperforms related approaches for computing minimal conflicts by applying a divide-and-conquer strategy.

We approximate the minimisation of the number of used relays by mapping solutions of the optimisation problem to conflicts. We say that *a conflict is a set of relays that allows us to find disjoint paths for all the sensors*.

Under this interpretation, we have as many constraints in S as candidates relays. That is, for each relay r we have a constraint in S stating that r is *available* to be used in a path from a sensor to the target. Relays that are not

constrained are *unavailable*. Notice that the more constraints we add to the set the more likely it is to have a conflict. To put it differently, if a subset C' of S is consistent any subset of C' is consistent too thus showing that our model holds monotonicity, which is required to use a minimal conflict approach. Following this interpretation, we have that a minimal conflict corresponds to a minimal set of relays needed to ensure disjointness.

We remark that the solution that we get from our QuickXplain based method is an approximation to the objective of finding a minimum set of relays since the solution is a minimal set and not a minimum set. That is, it could be very well the case that there is another minimal set of relays of smaller cardinality. Moreover, as we are using a timeout for the satisfiability checks, minimality is also approximated since a set of relays might be discarded (i.e., flagged as a non conflict) due to the timeout.

Results: Tables IV and V present the results that we have obtained with the different approaches on the n25 and n100 instances introduced in Section III. For each approach we report the best cost found (the number of chosen relays) and the time (in seconds) spent in finding it. The results of the local search approach (LS), already published in [14], were reproduced using GCC 4.4.4.

Our first observation is that the problem becomes very easy when the length constraint is disregarded. Indeed, BVUP without the length constraint is a mere flow problem. We suspect that the problem is still polynomial when minimising the number of relays (see column MIP I in Table IV) but this still remains to be proved. On the n25 instances [14], MIPs II and III failed almost completely because of the encoding of the length constraint, timing out after 30 minutes. MIP IV solved almost all instances in less than 10 seconds, faster than the local search, but on a few instances was significantly slower. CP I outperformed CP II.

As mentioned before, in CP II we use the Tree constraint [1]. The Tree constraint (as presented in [1]) is in the process of being migrated to the latest version of Choco. The experiments of the CP II approach were carried out using a preliminary version kindly provided by Jean-Guillaume Fages. In this version, the constraints on the degree of the nodes are handled independently using occurrence constraints[4]. There is no pruning taking advantage of the structure of the graph to filter the degree of each node. This is important in our problem since we can early detect failures by discovering articulation points.

We noticed that one issue with our CP approach is the high number of BVUP instances that need to be solved during the execution of QuickXplain. In general, we spend very little time but the number of instances is high. In CP I and CP II we are recreating and reading the models associated with the sensors every time. CP III is an optimisation of CP I where we keep the models in memory to avoid their recreation. In CP IV we enhance CP III by ordering the relays for QuickXplain by increasing distance from the sink, and reducing the timeout for solving each BVUP instance.

In order to appreciate the impact of ordering the relays in CP IV, in Figure 5 we compare this heuristic with its corresponding anti-heuristic on the n25 instances. That is, in the anti-heuristic we order the relays by decreasing distance from the sink. As can be observed in Figure 5, the choice of heuristic makes a significant difference to the performance of the model. The closest-first heuristic reduces the runtime over the anti-heuristic by more than 80% (199 sec to 1072 sec), and yet generates solutions with fewer relays (7.95 to 9.25).

For the n100 instances (Table 4), the MIP model did not scale up, failing to find a feasible solution in 1 hour. The CP approach is competitive in time with the local search method, although with lower quality solutions. It is important to remark that most of the time is spent re-reading models. In Table V we show both: the time spent by Choco solving the models and the total time. We attempted to keep the solvers in memory to avoid the re-reading of the models using the notion of worlds in Choco, but run out of memory for the big instances.

VI. Conclusion and Future Work

Designing wireless sensor network topologies that are robust to failures is an important task for a wide range of different monitoring applications, and is solved in the literature by local search approaches that add additional relay nodes to achieve robustness. We have identified that the underlying decision problem – do there exist k node-disjoint paths of length at most l from a sensor to a sink – is known to be NP-complete. We have developed complete constraint programming and mixed integer programming models for solving the problem. For larger problems, the CP model, implemented in Choco, outperforms the MIP approach, implemented in CPLEX, returning the solutions while the MIP model is still generating the problems. This provides an important tool for network operators, enabling them to evaluate proposed deployments in real time. We then presented the first approach to extending the CP solution to the minimisation problem, where the aim is to minimise the number of additional relays to ensure robustness. We implemented an approach based on QuickXplain, which searches for a minimal set of relays, producing solutions in faster time than the local search, but with higher cost.

Future work will focus on both the decision problem and the optimisation problem. For the decision problem, we will study the performance of the CP model for a wider range of networks. For the optimisation problem, we will implement the following Large Neighbourhood Search (LNS) approach.

A LNS approach to the Additional Relay Placement problem: once we have a solution of multi-sensor BVUP, we

[4]http://www.emn.fr/z-info/choco-solver/

Table IV
RESULTS WITH THE DIFFERENT APPROACHES FOR 25 SENSOR NODES AND 49 CANDIDATE RELAYS

	LS		MIP I		MIP II		MIP III		MIP IV		CP I		CP II	
instance	cost	time	cost	time	cost	time	cost	time	cost	time	cost	time	cost	time
scen-n25-01	6	29.48	4	0.63	4	407	4	608	4	1.56	4	433	6	673
scen-n25-02	7	10.74	4	0.47	-	1800	5	1800	5	3.07	5	106	6	301
scen-n25-03	8	20.29	4	0.58	-	1800	-	1800	5	14.68	8	505	9	651
scen-n25-04	6	18.76	5	0.64	5	49	5	577	5	1.78	9	145	11	666
scen-n25-05	9	11.73	3	0.77	-	1800	-	1800	4	5.82	7	558	8	841
scen-n25-06	10	14.56	5	1.06	-	1800	24	1800	8	359.90	9	298	9	428
scen-n25-07	7	47.25	6	1.02	-	1800	-	1800	7	9.15	7	732	7	320
scen-n25-08	11	19.87	5	0.70	-	1800	30	1800	7	117.36	9	466	9	779
scen-n25-09	9	23.72	6	1.36	-	1800	-	1800	7	33.44	7	229	7	235
scen-n25-10	7	17.02	5	0.53	-	1800	-	1800	6	9.46	6	107	6	199
scen-n25-11	9	12.86	6	1.10	-	1800	-	1800	7	19.04	10	240	11	575
scen-n25-12	3	19.57	3	0.52	-	1800	3	1255	3	3.07	8	432	5	474
scen-n25-13	10	20.81	6	0.94	-	1800	-	1800	8	205.16	10	215	10	634
scen-n25-14	7	18.52	6	0.96	-	1800	-	1800	6	3.08	10	562	11	1172
scen-n25-15	7	1.83	5	0.48	-	1800	6	1640	6	2.02	8	154	6	158
scen-n25-16	6	40.24	5	0.51	-	1800	-	1800	5	2.12	7	173	7	174
scen-n25-17	9	14.79	5	0.92	-	1800	-	1800	6	23.83	9	96	9	373
scen-n25-18	9	10.74	7	0.84	-	1800	-	1800	7	6.76	7	295	7	259
scen-n25-19	8	16.84	5	1.07	-	1800	-	1800	6	4.68	7	537	7	636
scen-n25-20	10	30.08	5	0.65	-	1800	-	1800	7	150.98	10	592	12	1532

Table V
RESULTS WITH THE DIFFERENT APPROACHES FOR 100 SENSOR NODES AND 196 CANDIDATE RELAYS

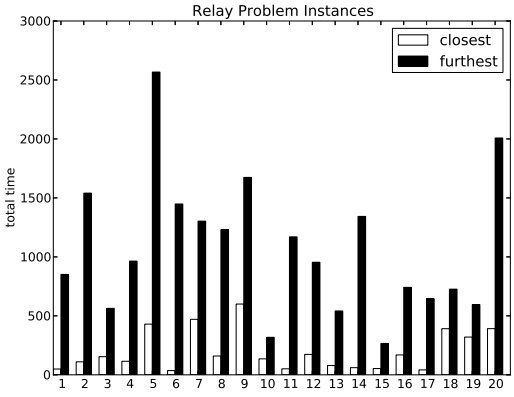
	LS		CP III			CP IV		
instance	cost	time	cost	solve time	total time	cost	solve time	total time
scen-n100-01	9	22108.70	19	4038.98	6723.31	17	1411.00	3845.96
scen-n100-02	11	13144.90	20	4541.52	6787.96	18	1244.68	3486.35
scen-n100-03	5	997.63	19	3357.86	6014.40	15	1447.18	4214.37
scen-n100-04	5	174.95	15	3273.79	6268.68	13	1515.62	4421.60
scen-n100-05	12	15699.70	20	2896.29	5876.92	20	1532.72	4451.92
scen-n100-06	7	4905.40	20	5042.72	8241.95	19	1473.99	4122.45
scen-n100-07	6	1974.81	21	4255.08	7169.11	16	1653.99	4725.40
scen-n100-08	11	14462.00	17	3092.81	5523.46	19	1697.22	4816.48
scen-n100-09	8	1751.50	14	2534.82	5013.07	13	1383.42	4042.70
scen-n100-10	8	17362.70	20	4133.57	7122.75	18	1278.03	3872.07
scen-n100-11	6	12237.10	14	3762.80	6394.02	12	1376.92	4029.24
scen-n100-12	9	2043.61	17	3305.84	6377.01	18	1582.52	4574.32
scen-n100-13	9	1623.28	15	3964.06	6763.79	19	1563.62	4523.04
scen-n100-14	6	14761.90	15	4080.36	7045.69	16	1521.96	4283.17
scen-n100-15	7	15098.70	11	2101.22	4129.03	10	883.64	2909.79
scen-n100-16	5	14958.10	11	1930.22	3884.59	11	871.12	2921.89
scen-n100-17	8	2291.31	21	3290.02	5959.84	16	1524.06	4466.98
scen-n100-18	8	33180.30	21	3828.45	6463.51	14	1164.11	3527.96
scen-n100-19	7	15476.90	16	3129.26	6001.84	14	1365.71	3972.77
scen-n100-20	10	16516.90	14	2985.34	5776.93	15	1287.88	3877.77

can use CP for implementing moves that iteratively improve the quality of the solution. More precisely, suppose that the set of relays used by the current multi-sensor BVUP solution is $R_1 \subseteq R$. We want to find an $R_2 \subseteq R_1$ that can be replaced with a set R_3 (subset of R) such that $(R_1 - R_2) \cup R_3$ is still a solution and $|R_3| < |R_2|$.

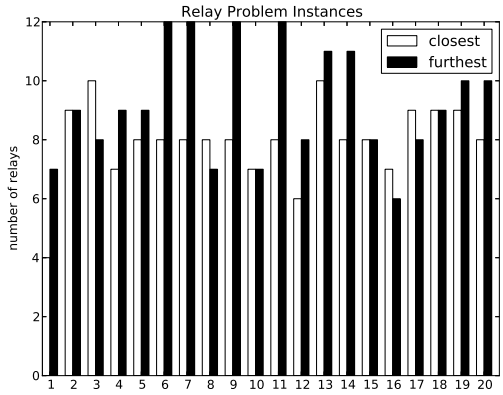
Let us assume that we set the cardinalities of R_2 and R_3 to α and β (which are parameters to be tuned). The task now is to find a set R_2 of cardinality α to be replaced with an R_3 of cardinality β such that $(R_1 - R_2) \cup R_3$ is still a solution. That is, we have $\alpha + \beta$ decisions variables where the domain of those variables in R_2 is R_1 and the domain of those variables in R_3 is $R - R_1$.

Our plan is to use CP for solving this decision problem. It is easy to show that the decision problem to be solved by CP is NP-complete. We prove this by reducing multi-sensor BVUP to this problem. Let $\mathcal{R}$ be the universe of relays for the LNS decision problem. We set $\mathcal{R}$ to $R \cup R_1$ where R_1 (in the reduction) is a set of dummy relays ensuring connectivity such that $|R_1| = |R| + 1$. We constraint $|R_2|$ to be equal to $|R_1|$ and $|R_3|$ to be equal to $|R|$. Finding R_2 and R_3 would be equivalent to solving the given multi-sensor BVUP problem.

The intuition behind the reduction is that checking whether the newly added relays can reconnect the sensors can be complex since, in the worst case, it can be as complex



(a) Comparing time.



(b) Comparing quality of the solution.

Figure 5. Comparing two heuristics for selecting the next relay during the execution of QuickXplain. In *closest* we select the relay that is closest to the target while in *furthest* we do the opposite

as solving a multi-sensor BVUP instance from scratch. Even though the LNS decision problem is NP-complete, the practicality of the approach would rely on the fact that α and β will be set to small numbers.

Acknowledgement: This work was supported by Science Foundation Ireland Grant No. 08/CE/I523.

REFERENCES

[1] N. Beldiceanu, P. Flener, and X. Lorca. Combining Tree Partitioning, Precedence, and Incomparability Constraints. *Constraints*, 13(4):459–489, 2008.

[2] A. Boukerche, R. W. N Pazzi, and R. B. Araujo. Fault-Tolerant Wireless Sensor Network Routing Protocols for the Supervision of Contex-Aware Physical Environments. *Journal of Parallel and Distributed Computing*, 66(4):586–599, Apr. 2006.

[3] J. L. Bredin, E. D. Demaine, M. Hajiaghayi, and D. Rus. Deploying Sensor Networks with Guaranteed Capacity and Fault Tolerance. In *Proc. 6th ACM Int'l Symp. Mobile Ad Hoc Networking and Computing (MobiHoc'05)*, pages 309–319, May 2005.

[4] O. Chipara, Z. He, G. Xing, Q. Chen, X. Wang, C. Lu, J. Stankovic, and T. Abdelzaher. Real-time Power-Aware Routing in Wireless Sensor Networks. In *Proc. 14th IEEE Workshop Quality of Service (IWQoS'06)*, pages 83–92, Jun. 2006.

[5] T. A. Feo and M. G. C. Resende. A Probabilistic Heuristic for a Computationally Difficult Set Covering Problem. *Operations Research Letters*, 8:67–71, 1989.

[6] T. A. Feo and M. G. C. Resende. Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[7] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection Tree Protocol. In *Proc. 7th ACM Conf. Embedded Networked Sensor Systems (SenSys'09)*, Nov. 2009.

[8] P. A. Golovach and D. M. Thilikos. Paths of Bounded Length and Their Cuts: Parameterized Complexity and Algorithms. *Discrete Optimization*, 8(1):72–86, 2011.

[9] X. Han, X. Cao, E. L. Lloyd, and C. C. Shen. Fault-tolerant Relay Node Placement in Heterogeneous Wireless Sensor Networks. *IEEE Transactions on Mobile Computing*, 9(5):643–656, May 2010.

[10] U. Junker. QUICKXPLAIN: Preferred Explanations and Relaxations for Over-Constrained Problems. In D. L. McGuinness and G. Ferguson, editors, *AAAI*, pages 167–172. AAAI Press/The MIT Press, 2004.

[11] S. Misra, S. D. Hong, G. Xue, and J. Tang. Constrained Relay Node Placement in Wireless Sensor Networks to Meet Connectivity and Survivability Requirements. In *Proc. 27th Ann. IEEE Conf. Computer Communications (INFOCOM'08)*, pages 281–285, Apr. 2008.

[12] J. Pu, Z. Xiong, and X. Lu. Fault-Tolerant Deployment with k-connectivity and Partial k-connectivity in Sensor Networks. *Wireless Communications and Mobile Computing*, 9(7):909–919, May 2008.

[13] M. G. C. Resende and C. C. Ribeiro. Greedy Randomized Adaptive Search Procedures. In F. Glover and G. Kochenberger, editors, *State of the Art Handbook in Metaheuristics*, pages 219–249. Kluwer Academic Publishers, 2002.

[14] L. Sitanayah, K. N. Brown, and C. J. Sreenan. Fault-Tolerant Relay Deployment for k Node-Disjoint Paths in Wireless Sensor Networks. In *Proc. 4th Int'l Conf. IFIP Wireless Days (WD'11)*, pages 1–6. IEEE, Oct. 2011.

[15] Y. Xiao, K. Thulasiraman, and G. Xue. Constrained Shortest Link-Disjoint Paths Selection: A Network Programming Based Approach. *IEEE Trans. Circuits and Systems I: Regular Papers*, 53(5):1174–1187, 2006.

[16] Y. Zeng, C. J. Sreenan, L. Sitanayah, N. Xiong, J. H. Park, and G. Zheng. An Emergency-Adaptive Routing Scheme for Wireless Sensor Networks for Building Fire Hazard Monitoring. *Sensors*, 11(3):2899–2919, Mar. 2011.