On the Admission of Dependent Flows in Powerful Sensor Networks

Reuven Cohen Ilia Nudelman Gleb Polevoy
Department of Computer Science
Technion – Israel Institute of Technology
Haifa 32000
Israel

***Abstract*—In this paper we define and study a new problem, referred to as the Dependent Unsplittable Flow Problem (D-UFP). We present and discuss this problem in the context of large-scale powerful (radar/camera) sensor networks, but we believe it has important applications on the admission of large flows in other networks as well. In order to optimize the selection of flows transmitted to the gateway, D-UFP takes into account possible dependencies between flows. We show that D-UFP is more difficult than NP-hard problems for which no good approximation is known. Then, we address two special cases of this problem: the case where all the sensors have a shared channel and the case where the sensors form a mesh and route to the gateway over a spanning tree.**

I. Introduction

In wireless sensor networks, sensors probe the surrounding environment and generate reports of the collected readings. Using wireless communication, these reports are sent to a control center, usually through a gateway deployed in the physical proximity of the sensors. While much of the focus of the sensor network community has been on the design of miniature low-power wireless sensor networks, an important networking revolution has been taking place for powerful sensors such as radars and cameras [10], [11], [19], [21], [28]. Such sensor networks are used in a variety of civilian and military applications such as earthquake sensing, weather monitoring, road traffic monitoring, and Network Centric Operations (NCO)[9], [21].

As indicated by [11], these emerging systems raise a number of new research challenges that do not exist in mote-class wireless sensor networks. Although these systems are not limited by energy considerations, the data they generate exceeds, by several orders of magnitude, the bandwidth capacity of the wireless networks that connect them to their gateway. Since delivering all this data is not possible, a decision must be made as to which flow to deliver and which to drop. This decision should be based on the bandwidth of each flow and its profit to the whole system.

This brings to mind the well-known NP-hard Unsplittable Flow Problem (UFP) [7], [12]. However, UFP does not capture an important property of the considered radar/camera sensors: the dependency between different flows. For example, consider two Doppler radars that scan partially overlapping areas. Some of their detected events are likely to be similar. Thus, the profit of delivering both flows should be smaller than the sum of their individual profits. In other cases, however, the combined profit of two different flows might be greater than the sum of their individual ones. For instance, in the context of adaptive sensing of the atmosphere, data from multiple radars allows for more accurate estimation of wind velocity vectors [19]. When some of the dependencies between flows are positive and others are negative, the dependency set is said to be mixed and the resulting optimization problem is harder.

In this paper we define and study a new problem, referred to as *the Dependent Unsplittable Flow Problem (D-UFP)*. We present and discuss this problem in the context of powerful radar/camera wireless sensor networks (WSNs), but we believe it has important applications in the admission of large flows in other networks as well. While UFP's goal is to maximize the profit gained by accommodating independent flows, our generalization takes into account the dependency between flows. Thus, the profit from delivering two flows is not necessarily equal to the sum of their profits. The algorithms proposed in this paper allow the sensor network to determine which flows to deliver and which to drop. In some applications, such as NCO [21], the network may need to run the algorithm very often, sometimes even once a minute, in order to adapt itself to the changing reality.

The rest of the paper is organized as follows. In Section II we discuss related work. In Section III we introduce our framework and define D-UFP. In Section IV we describe a new algorithm for solving D-UFP in the case where all the sensors transmit to the gateway over one common channel. In Section V we describe a new algorithm for solving D-UFP in the more general case where multi-hop communication is needed in order to reach the gateway. Section VI presents a simulation study of the proposed algorithms. Section VII generalizes D-UFP(tree) to the case where dependencies exist not only between pairs. Finally, Section VIII concludes the paper.

II. Related Work

The problem addressed in this paper is one of the most fundamental networking problems: how to accommodate traffic in a network when the demand exceeds the supply. While this problem is relevant to many network applications, including optical networks, cellular networks, and standard IP networks, it is especially relevant to powerful sensor networks, where the amount of data generated by the nodes is greater by several orders of magnitude than the bandwidth of the wireless links connecting the sensor nodes to the gateway.

One approach to the bandwidth scarcity problem in sensor networks is data fusion: instead of sending the raw data to the network gateway, the sensors perform local computations and transmit only the required and partially processed data [1]. In [13] the authors address specifically the case where close sensors cover overlapping areas. Such sensors negotiate with each other before transmitting data to ensure that only useful information will be transferred.

However, data fusion is often not possible in powerful sensor networks. First, in many of these networks, human operators play a key role in deciding how to process the data [9]. Second, while data fusion can help to reduce traffic when different flows contain similar information, as in [13], it cannot capture "positive dependency," i.e., the case where the joint profit of two flows is greater than the sum of their individual profits (a typical case when the senors are radars [19]). Third, data fusion is effective when the dependent flows originate at close sensors, but not when they originate at remote sensors that cover remote areas, as is usually the case for positive-dependent flows.

In this paper, we use a profit/utility function to determine which flows will be routed and which will not. This function indicates the "profit to the system" from each flow, and the positive or negative conditional profit for every pair of flows. In [19], the authors also use a utility function to address the issue of profit assignment in the context of radar networks. Their utility function is more related to the various possible radar configurations. However, similar considerations can be used in order to prioritize the flows created by each radar.

UFP has been extensively studied in the past and was shown to be not only NP-Complete [7] but also APX-Complete [12], which means that it has no PTAS[1]. Moreover, there exists no $\log^{\frac{1}{3}-\epsilon}(|E|)$-approximation and no $\log^{\frac{1}{2}-\epsilon}(|N|)$-approximation for UFP for any $\epsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\text{polylog}(n)})$ [2], [3], [4].

A greedy $\sqrt{|E|}$-approximation for UFP is presented in [5], [17], and approximations for some special cases are provided in [8]. When the graph is a tree, UFP can be approximated with a factor of 2 [12].

Another relevant problem is the Quadratic Knapsack Problem (QKP) [6], because we show that it is equivalent to D-UFP under some constraints. QKP is NP-hard in the strong sense, as proved in [6]. Hence, we cannot expect to find an FPTAS for it. A branch-and-bound algorithm for QKP is presented. The algorithm first finds tight lower and upper bounds using an efficient heuristic. Then, suboptimal Lagrangian multipliers are found and a variable reduction procedure is performed. Finally, the algorithm invokes a recursive branch-and-bound procedure. During each iteration, an upper bound for QKP is calculated using Lagrangian relaxation, and is solved through a number of continuous Knapsack problems.

For positive only dependencies and instances with up to $n = 400$ items, the algorithm achieves a high quality upper bound, within 1% of the optimum. However, when dependencies are both positive and negative, the quality of the upper bound drastically decreases, which is translated into a substantial increase in the running time.

Another technique for solving mixed dependencies is a branch-and-bound method based on an upper bound derived by semidefinite programming [14], [16], [15]. In [15], semidefinite programming is extended to 0-1 QKP and three types of semidefinite relaxations are obtained and analyzed. In [16], the authors show how these relaxations could be strengthened by a polyhedral cutting plane method, and a tight bound is derived. Using a limited computational experiment, the authors also report that the lower and upper bounds are of good quality but computationally slow.

III. The Dependent Unsplittable Flow Problem (D-UFP) in Powerful WSNs

A. Dependent Flows

The Unsplittable Flow Problem (UFP) is a well-known optimization problem with many networking applications. The input of UFP is a network graph with a capacity for each link, and a set F of flows. Every flow $f_i \in F$ is defined by a quadruple flow descriptor (s_i, d_i, b_i, p_i), where s_i is the source, d_i is the destination, b_i is the bandwidth demand, and p_i is the profit/utility gained

[1]A PTAS (Polynomial Time Approximation Scheme) is an approximation scheme whose running time is polynomial in the size of the input.

example	p_{11}	p_{22}	p_{12}
1	a	b	$-c/2$
2	0	0	$c/2$
3	a	b	$c/2$
4	$N_1 \cdot a + N_2 \cdot c$	$N_2 \cdot b$	$-N_2 \cdot c/2$
5	a	0	$c/2$

TABLE I
EXAMPLES OF DEPENDENT PROFIT ASSIGNMENT

from routing this flow. The goal is to determine which flows should be accommodated and what route each one should use in order to maximize the total profit. The fact that the flows are unsplittable means that each one is either fully accommodated along a single route or not accommodated at all. We now generalize UFP by modeling possible dependencies between the profits of different flows.

Definition 1. Consider two flows f_1 and f_2. Suppose that the profit from accommodating only f_1 in the network is p_1 and the profit from accommodating only f_2 is p_2. These two flows are said to be dependent if the profit of accommodating both of them is either smaller or greater than $p_1 + p_2$.

Definition 2. Let F be a set of flows. Let every flow be represented by a node in a dependency graph $G(F, E)$. An edge exists between two nodes f_1 and f_2 if flows f_1 and f_2 are dependent. Let $G_1(F_1, E_1), G_2(F_2, E_2) \ldots$ be the connected subgraphs of G such that $\bigcup F_i = F$ and there is no edge in G between G_i and G_j for $i \neq j$. Then,

(a) The ***maximum dependency rank*** of F is defined as MAX_i $(|F_i|)$ and is denoted by $\mathcal{R}(F)$.
(b) The ***dependency order*** of F is the cardinality of the maximum clique in G.
(c) The ***dependency degree*** of $f_i \in F$ is the degree of this node in G, i.e., the number of flows it depends on, and is denoted by $deg(f_i)$. The average dependency degree of F is defined as $\sum_{i,f_i \in F}$ $(deg(f_i))/|F|$ and is denoted by $\mathcal{D}(F)$. ■

B. Profit Assignment

Profit assignment to dependent flows is a big challenge. In what follows we present examples that cover the most common dependency combinations. Table I summarizes these examples.

The first example is of a Doppler radar. The individual profit of a flow generated by each radar can be set to be proportional to the size of the covered area. However, when the areas of two radars partially overlap, we assign them a joint negative dependent profit. Thus, if the area covered by R_1 is a, by R_2 is b, and by both is c, we have $p_{11} = a,\ p_{22} = b$ and $p_{12} = p_{21} = -c/2$. If the same area is covered by more than two radars, we have a dependency order higher than 2, which is addressed only in Section VII.

The second example is of directionality-based location discovery sensors. Consider GPS-like sensors, where each sensor knows its own location and can only tell the exact direction (azimuth) of a detected event. One can extract the location of such an event by getting information from two such sensors [26], [22]. Thus, if c is the profit from detecting the location of the event, then $p_{11} = p_{22} = 0$ and $p_{12} = p_{21} = c/2$. If more than two sensors can detect the same event, we have a dependency order higher that 2.

The third example has to do with the importance of getting a full picture from the battlefield in order to make some crucial decision. Suppose that some region of interest in the battlefield is covered by two sensors/radars. Each device scans a mutually exclusive area and gains some individual profit. However, acquiring the data from both flows gives a "full picture" of the region and is therefore associated with an extra profit of c. In this case we have $p_{11} = a,\ p_{22} = b$ and $p_{12} = p_{21} = c/2$.

In the fourth example we consider a camera that produces two video flows, one high and one low resolution. Suppose that we have two types of terminals: type-1 is a PDA with a small screen that can benefit only from the low resolution flow, and type-2 is a high resolution screen that can benefit from both flows but benefits more from the high resolution flow. The profit of type-1 from the high resolution flow is 0 and from the low resolution flow is a. The profit of type-2 from the high resolution flow is b and from the low resolution flow is $c < b$. If there are N_1 type-1 terminals and N_2 type-2 terminals, then we have $p_{11} = N_1 \cdot a + N_2 \cdot c$, $p_{22} = N_2 \cdot b$ and $p_{12} = p_{21} = -N_2 \cdot c/2$.

In the last example we consider two flows such that one is not useful without the other but the other has some merit without the first. A video flow and the corresponding audio flow is one example. In this case, $p_{11} = a$ holds for the video flow, $p_{22} = 0$ holds for the audio flow, and $p_{12} = p_{21} = c/2$ holds for both flows.

C. Problem Formulation and Classification

We are now ready to define the new Dependent Unsplittable Flow Problem (D-UFP):
Problem 1 (D-UFP):

Instance: A network defined by a graph $G = (V, E)$ with a capacity function $c : E \to \mathbb{Z}^+$. A set F of n flows $f_1, f_2, \ldots, f_n$, each defined by a flow descriptor (s_i, d_i, b_i, p_i). A profit/utility function $\mathcal{P}$ from which one can compute the aggregated profit of each subset of flows.

Objective: Find a subset $F' \subseteq F$ of flows that has a feasible routing and a maximum profit.

When the dependency order is limited to 2, the profit can be represented using a symmetric profit matrix. In this case, a diagonal element of $\mathcal{P}$, i.e., p_{ii} for every i, indicates the profit gained from accommodating the flow f_i. A non-diagonal element, p_{ij} for every $i \neq j$, is half of the extra (negative or positive) profit gained by accommodating both f_i and f_j. Thus, the profit gained by accommodating two flows f_i and f_j is $p_{ii} + p_{jj} + p_{ij} + p_{ji}$, where $p_{ij} = p_{ji}$. The extra profit $p_{ij} + p_{ji}$ might be negative for some pairs and positive for others.

D-UFP can be classified according to the following characteristics:

1) Network type: The topology of the network has a critical impact on the problem. In this paper we address two cases: the case where all the sensor are connected to a single broadcast channel through which they send their flows to the gateway and the case where they are connected in a multi-hop tree whose root is the gateway.
2) Dependency type: We distinguish between the case where the joint profit of every two dependent flows f_i and f_j is always larger (or, equivalently, always smaller) than $p_i + p_j$, and the general case where for some pairs f_i and f_j the joint profit is larger than $p_i + p_j$ while for others it is smaller.
3) Maximum dependency rank: We distinguish between the case where the dependency rank is low, e.g., $\mathcal{R}(F) \leq 20$ and the case where it is high.
4) Average dependency degree: We distinguish between the cases where $\mathcal{D}(F)$ is relatively small, e.g., $\mathcal{D}(F) = \mathcal{R}(F)/10$ and the cases where it is large.

IV. D-UFP in a Single Shared Channel

A. D-UFP(SC)

In this section we focus on a version of D-UFP that fulfills the following requirements:

(R1) There is only one broadcast channel, used by all sensors to send their flows to the network gateway.

(R2) Mixed dependencies, i.e., the profit of any two dependent flows f_i and f_j, might be either smaller or larger than $p_i + p_j$.

(R3) A low dependency rank ($2 \leq \mathcal{R} \leq 20$) and/or a low dependency degree ($\mathcal{D}(F) \ll \mathcal{R}(F)$).

(R4) A large number of flows ($n > 100$).

(R5) A dependency order of 2. As previously stated, in such a case the profit function $\mathcal{P}$ can be represented by a symmetric matrix.

This version is referred to as D-UFP(SC), where SC stands for Single Channel. D-UFP(SC) is very relevant to many state-of-the-art satellite-based powerful sensor networks. The justification for (R3) is that very often the sensors cover disconnected areas, such that each area is covered by a different set of sensors and radars. Thus, although n can be in the order of 100, $\mathcal{R}(F)$ is usually not larger than 20. Moreover, even if $\mathcal{R}(F)$ is relatively large, $\mathcal{D}(F)$ can still be assumed to be considerably smaller.

Due to (R1) and (R5), D-UFP(SC) is equivalent to the Quadratic Knapsack Problem (QKP) defined below. Requirements (R2)-(R3) enable us to consider specific QKP instances, for which we propose efficient algorithms even though the number of flows might be large (R4).

The Quadratic Knapsack Problem (QKP): The instance of this problem is a set S of n items $s_1, s_2, \ldots, s_n$ and a capacity c. Each item s_j has a weight w_j and a profit defined by the profit matrix $\mathcal{P} = (p_{ij})$, described earlier. The objective is to find a subset $S' \subseteq S$ of items that has a feasible packing, namely, $\sum_{s_j \in S'} w_j \leq c$, and a maximum profit $\sum_{s_i, s_j \in S'} p_{ij}$.

QKP is NP-hard in the strong sense [6]. Moreover, [25] shows that when every p_{ij} might be either positive or negative, as in our case, no polynomial time algorithm with fixed approximation ratio exists unless $\mathsf{P} = \mathsf{NP}$.

B. An MCKP-based Algorithm for D-UFP(SC)

We now propose a new algorithm for solving D-UFP(SC). While this algorithm scales very well in the number of flows, it strongly depends on the maximum dependency rank $\mathcal{R}(F)$. Thus, it fits our D-UFP(SC) (R1)-(R5) requirements. This algorithm transforms a D-UFP(SC) instance with a given dependency rank to an instance of the Multiple Choice Knapsack Problem (MCKP), defined as follows.

The Multiple Choice Knapsack Problem (MCKP): The instance is a set of n items, belonging to m disjoint classes of items $C_1, \ldots, C_m$, and a capacity c; each item $i \in C_j$ has a profit $p(i)$ and a weight $w(i)$. The objective is to find a subset C' of items, with exactly one item from each class, whose packing is feasible, i.e., $\sum_{i \in C'} w(i) \leq c$, such that the aggregated profit $\sum_{i \in C'} p(i)$ is maximized.

MCKP can be optimally solved in pseudo-polynomial time of $O(nc)$ [24]. It can also be approximated with a factor of 2 in linear time [29]. The best FPTAS for MCKP is given in [20], with running time of $O(mn/\epsilon)$.

We now show how to transform a D-UFP(SC) instance to an MCKP one. Consider a set F of items, divided into m mutually disjoint subsets $F_1, \ldots, F_m$. We construct m MCKP classes such that each class C_i consists of $2^{|F_i|}$ binary vectors. Each vector $v \in C_i$ represents one option for choosing items in F_i. For example, if $|F_i| = 4$, the vector 1101 indicates that the first, second, and fourth items from F_i are chosen. Each such vector is also associated with a profit and a weight. The profit is the sum of the profit of the chosen items, while taking into account the dependencies. The weight is the sum of the weight of the chosen items. The MCKP knapsack size is the same as the D-UFP(SC) one. We can now apply an MCKP algorithm for the transformed instance. MCKP will choose one item (vector) from each class C_i, which is translated into a selection of a subset of dependent items from each class. Algorithm 1 summarizes this idea.

Algorithm 1. *(solving D-UFP(SC) using a reduction to MCKP)*

1) *Create independent item sets from the D-UFP(SC) instance, $\{F_i\}_{i=1}^{m}$.*
2) *Transform the independent D-UFP(SC) sets into an MCKP instance in the following way:*
   a) *transform each item set $F_i,\ i = 1, \ldots, m$ to an MCKP class C_i, with $2^{|F_i|}$ items (vectors);*
   b) *let $F \subseteq F_i$ be the chosen items indicated by the vector $v \in C_i$; the profit and weight assigned to v are $p(v) = \sum_{k,l \in F'} p_{kl}$ and $w(v) = \sum_{k \in F'} w_k$.*
3) *Set the MCKP capacity to be equal to the bandwidth of the D-UFP(SC) shared channel.*
4) *Solve MCKP.* ■

C. Reducing the Maximum Dependency Rank of the D-UFP(SC) Instance

The time complexity of reducing a D-UFP(SC) to MCKP is $O(2^{\mathcal{R}(F)})$. Hence, even a polynomial time approximation for MCKP cannot solve instances with $\mathcal{R}(F) > 20$ in reasonable time. In what follows we propose an algorithm for reducing the maximum dependency rank of the input set of flows F. This algorithm takes advantage of requirement (R3), which states that if the dependency rank is higher than 20, the maximum dependency degree is low. Still, such a reduction is likely to decrease the profit because MCKP has an FPTAS while D-UFP(SC) with mixed dependencies does not (unless $\mathcal{P} = \mathcal{NP}$). In the following we reduce the D-UFP(SC) dependency rank by translating this problem into the Simple Graph Partitioning Problem (SGPP) [27].

The Simple Graph Partitioning Problem (SGPP): The instance is a simple graph $G = (V, E)$, an edge weight function $c : E \rightarrow \mathbb{R}^{+}$, and a positive constant $\mathcal{R}'$. The objective is to divide G into disconnected segments by removing some of the edges such that each segment will have at most $\mathcal{R}'$ nodes and the total weight of the pruned edges is minimized.

In order to translate our problem to SGPP, we represent the items (flows) by the graph nodes and assign to every edge a value that indicates the correlation between two dependent items. While several such correlation functions can be considered, in the following we use $c(i, j) = 2|p_{ij}|$. This represents the correlation as the total dependent profit the algorithm will have to ignore if the edge is removed.

SGPP is NP-Complete. All previously proposed algorithms solve small instances, with up to $n = 50$ nodes, while we are interested in larger ones. We now propose an efficient algorithm that is based on the Kruskal method [18] and whose running time complexity is $O(|E| \cdot log|E|)$. The algorithm starts with a graph whose edge set is empty. It then goes through a list that contains all the edges sorted in a decreasing order of their weight and adds the next edge if this edge does not create a connected segment bigger than $\mathcal{R}'$.

Procedure 1. *(Kruskal method for SGPP) For each connected component $G_i = (V_i, E_i)$ of G for which $|V_i| > \mathcal{R}'$ holds do*

1) *Start with a graph $G'_i = (V_i, E'_i)$, where $E'_i = \emptyset$.*
2) *Sort the edges of E_i in decreasing order of their weight.*
3) *Go through the sorted list and add an edge $(u, v) \in E_i$ to E'_i if u and v are already in the same connected segment or if they create a new connected segment with no more than $\mathcal{R}'$ nodes.* ■

In section VI we show the effectiveness of executing this procedure prior to Algorithm 1.

V. D-UFP IN MULTI-HOP POWERFUL WSNS

A. D-UFP(tree)

The single channel case addressed in Section IV covers many applications of powerful WSNs. However, there are also important applications for which it is neither economical nor possible to equip each powerful sensor with a satellite transceiver. In such applications, the sensors use some variant of WiFi communication in order to transmit their flows, and multi-hop routing is needed in order to reach the gateway. In this section we study the new D-UFP problem in such networks, under the conventional assumption where the collection of multi-hop paths from the sensors to the gateway form a spanning tree rooted at the gateway. Figure 1 shows an example of the routing tree connecting 9 sensors to the gateway. We also assume that each collection of wireless links connecting a set of sensors to their parent (e.g., the collection $\{s4 \rightarrow s1,\ s5 \rightarrow s1\ \}$ or the collection $\{s6 \rightarrow s2,\ s7 \rightarrow s2\ \}$) can be viewed as a single shared channel.

If the capacity of the level-1 shared channel connecting the level-1 nodes to the gateway is smaller than or equal to the capacity of every other tree channel, then the problem is reduced to the single channel case. In such a case we can ignore the multi-hop routing and solve the problem using the algorithms proposed in Section IV. Therefore, in what follows we address the general case, where a collection of flows can be admitted into the root channel, but cannot reach this channel due to some other channel's lack of bandwidth. The new problem, called

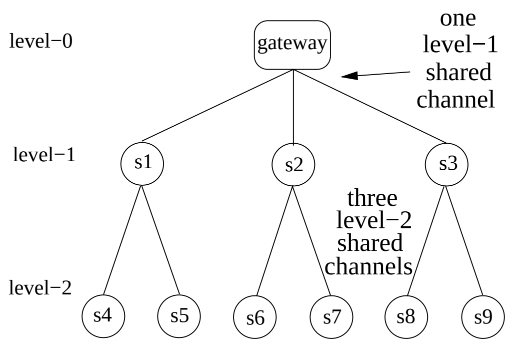


Fig. 1. The multi-hop routes to the gateway form a tree

D-UFP(tree), has the same (R2)-(R5) requirements as D-UFP(SC), but a new (R1) requirement as follows:

(R1) The sensors have any mesh topology. Routing is performed over a (shortest-path or any other) tree rooted at the gateway.

B. An MMKP-based Algorithm for D-UFP(tree)

In Section IV we solved D-UFP(SC) by transforming it into an MCKP instance, with a knapsack size equal to the channel bandwidth. To use a similar approach for D-UFP(tree), we should consider each shared channel as a different dimension of a knapsack and each flow as being a multidimensional item. Each dimension of the knapsack indicates the available bandwidth on one tree channel. Each dimension of an item indicates the bandwidth requirement of the corresponding flow from each tree channel. That is, a 100Mb/s flow originating at sensor $s4$ in Figure 1 requires 100Mb/s on the channel to $s1$ and on the channel to the gateway, but no bandwidth on the other channels.

The extension of MCKP to a multidimensional knapsack is called MMKP (Multiple Dimensional Multiple Choice Knapsack Problem), and is defined as follows.

The Multiple Dimensional Multiple Choice Knapsack Problem (MMKP): The instance is a set of n items, belonging to m disjoint classes of items $C_1, \dots, C_m$, and a vector of capacities $c_1, \dots, c_D$. Each item $i \in C_j$ has a profit $p(i)$ and a weight vector $\vec{w(i)} \in \mathbb{R}^D$. The objective is to find a subset C' of items, with exactly one item from each class, whose packing is feasible (i.e., $\sum_{i \in C'} \vec{w(i)} \leq \vec{c}$), such that the aggregated profit $\sum_{i \in C'} p(i)$ is maximized.

Algorithm 2. *(Solving D-UFP(tree) using a reduction to MMKP)*

1) *Create independent item sets from the D-UFP(tree) instance, $\{F_i\}_{i=1}^m$.*
2) *Transform the independent D-UFP(tree) sets into an MMKP instance in the following way:*
   a) *transform each item set $F_i,\ i = 1, \dots, m$ to an MMKP class C_i, with $2^{|F_i|}$ items (vectors);*
   b) *let $F \subseteq F_i$ be the chosen items indicated by the vector $v \in C_i$; the profit and weight assigned to v are $p(v) = \sum_{k,l \in F'} p_{kl}$ and $w(v) = \sum_{k \in F'} w_k$, respectively.*
3) *Set each capacity c_i of MMKP to the bandwidth of one channel in the tree, such that the number D of capacities in the vector $c_1, \dots, c_D$ is equal to the number of channels.*
4) *Solve MMKP.* ■

Algorithm 2 works as long as each flow has a unique path, even if the collection of paths does not form a tree. However, the time complexity of MMKP depends on the number of dimensions (channels) of the knapsack. Later on we shall use the tree assumption to substantially reduce this complexity.

MMKP generalizes the Multidimensional Knapsack Problem (MKP). Thus, it is not only NP-hard, but is also unlikely to have an EPTAS. In [23], a polynomial-time approximation scheme is proposed for a similar problem called the Multiple-Choice Multidimensional Knapsack problem (MMK). Due to the similarity of MMK to MMKP, the algorithm proposed by [23] can also be used for solving MMKP. The time complexity of this algorithm is $O((nm)^{\lceil D/\epsilon \rceil})$, where $\frac{1}{1+\epsilon}$ is the approximation ratio. Thus, we cannot afford large D/ϵ values. Since D represents the number of tree channels in a D-UFP(tree) instance, which in general can be large, we must reduce its value somehow.

C. An Efficient Probabilistic Algorithm

We now propose an efficient probabilistic algorithm to solve D-UFP(tree) using algorithm 2 while bounding the value of D. The main idea behind the proposed scheme is as follows. Assuming that the capacities of the tree's channels do not vary much, and because all the flows should reach the gateway, the low level tree channels, i.e., those that are close to the root, are likely to be the network bottleneck. For example, if every node in Figure 1 generates exactly one flow, then the load on each of the level-2 channels is 2 flows whereas the load on the level-1 channel channel is 6 flows. The following algorithm takes advantage of this property.

Algorithm 3. *(A probabilistic algorithm for solving D-UFP(tree) efficiently)*

1) *Set $R \leftarrow 1$.*
2) *Repeat:*
   a) *run Algorithm 2 while taking into account, in step (3), only the channels up to level R of the tree;*
   b) *check if the feasible solution found by Algorithm 2 for the channels in level $1 \cdots R$ is also feasible for every channel $i > R$;*
   c) *if the solution is infeasible, set $R \leftarrow R + 1$.*

3) *Until a feasible solution for the whole tree is found or R is too big for step 4 of Algorithm 2.*

To quantitatively estimate Algorithm 3, we define a distribution of channel capacities and a set of not-necessarily-feasible flows, and evaluate the value of $\Pr(A|A_R)$, where

$$A \overset{\text{def}}{=} \left\{\begin{array}{l}\text{the solution satisfies the capacity con-}\\ \text{straints in the whole tree}\end{array}\right\} \text{and}$$

$$A_R \overset{\text{def}}{=} \left\{\begin{array}{l}\text{the solution satisfies the capacity con-}\\ \text{straints until (including) the depth } R\end{array}\right\}.$$

To find a lower bound for $\Pr(A|A_R) = \frac{\Pr(A)}{\Pr(A_R)}$, we need to find a lower bound for $\Pr(A)$ and an upper bound for $\Pr(A_R)$, or equivalently, an upper bound for $\Pr(\overline{A})$ and a lower bound for $\Pr(\overline{A_R})$. The first bound is the best of:

$$\Pr(\overline{A}) \leq \sum_{i=1}^{L} \Pr(\overline{A(i)})$$

and

$$\begin{aligned}\Pr(A) &= \Pr(\cap_{i=1}^{L} A(i))\\ &= \prod_{i=1}^{L} \Pr(A(i)| \cap_{j=1}^{i-1} A(j)) \geq \prod_{i=1}^{L} \Pr(A(i));\end{aligned}$$

the second bound is:

$$\Pr(\overline{A_R}) = \Pr(\cup_{i=1}^{R} \overline{A(i)}) \geq \max_{i=1,\ldots,R} (\Pr(\overline{A(i)})),$$

where

$$A(i) \overset{\text{def}}{=} \left\{\begin{array}{l}\text{the solution satisfies the capacity con-}\\ \text{straints of all the level-}i\text{ channels}\end{array}\right\}.$$

To find $A(i)$, we define for every node v:

$$E(v) \overset{\text{def}}{=} \left\{\begin{array}{l}\text{the solution satisfies the capacity con-}\\ \text{straint at the shared channel to } v\end{array}\right\}.$$

From the independence of the $E(v)$'s on the same level,

$$\Pr(A(i)) = \prod_{v \in \text{level } i-1} \Pr(E(v)). \quad (1)$$

Let $F(v)$ be a random variable that indicates the total flow through v. Thus,

$$\begin{aligned}\Pr(E(v)) = \Pr(\{F(v) \leq C(e)\}) &=\\ \Pr(\{F(v) - C(e) \leq 0\}),& \quad (2)\end{aligned}$$

where e is the channel to v and $C(e)$ is its capacity.

Suppose that we have a complete l-ary tree of height L. Let the channel capacities be independent and identically distributed random variables, with $C(e) \sim \mathbf{Bin}(\lambda\rho, \mathbf{p})$. Suppose that every non-root tree node seeks to transmit $X \sim \mathbf{Pois}(\mathbf{d})$ flows to the root and that the bandwidth demand of each flow is $Y \sim \mathbf{Bin}(\rho, \mathbf{p})$. To use Eq. 2 in order to solve Eq. 1, we can approximate $F(v) - C(e)$ by a Gaussian distribution.

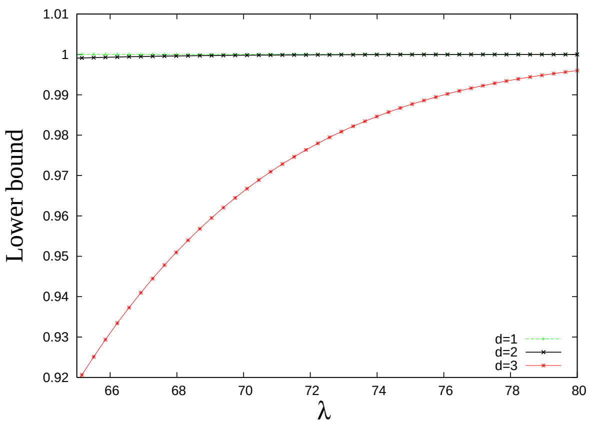


Fig. 2. The computed lower bound on $\Pr(A|A_R)$

From the above discussion it is clear that $\Pr(A|A_R)$ depends mainly on the average number of flows originating from each node (d), and on the ratio between the capacity of each channel and the bandwidth demand of each flow (λ). In Figure 2 we show the results of the above analysis as a function of λ and d for $l = 2$, $\rho = 10$ and $p = 0.1$. In this figure $R = 1$ and $L = 4$, which means that instead of checking the constraints for $\sum_{i=1}^{4} 2^{i-1} = 15$ channels, we check them only for the root channel.

Recall that the results shown in the figure are only lower bounds. Simulation results indicate that the actual value of $\Pr(A|A_R)$ is much bigger and can be obtained for larger values of d and/or for smaller values of λ.

VI. Simulation Study

We study the performance of our algorithms using Monte Carlo simulations on instances constructed as follows. Let $\{F_i\ ,\ i = 1,\ldots,m\}$ be m disjoint subsets of dependent flows. Each of these subsets is represented by a diagonal block in the profit matrix $\mathcal{P}$:

$$\begin{pmatrix} (\mathcal{P}_1)_{|F_1|\times|F_1|} & 0 & \ldots & 0\\ 0 & (\mathcal{P}_2)_{|F_2|\times|F_2|} & \ldots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \ldots & (\mathcal{P}_m)_{|F_m|\times|F_m|} \end{pmatrix}.$$

Let $\mathcal{R}(F)$ and $\mathcal{D}(F)$ be the maximum dependency rank and the average dependency degree of the considered instance. The size of each diagonal block is uniformly distributed in $[1, \mathcal{R}(F)]$, and in each of the block's rows there are, on the average, $\mathcal{D}(F)$ non-zero entries. Each non-zero entry value $p_{ij} = p_{ji}$ is uniformly distributed in $[-r/2, r/2]$, where r is a constant for the whole matrix. The weight w_j of each item j is uniformly distributed in $[1, r/2]$. Finally, the capacity of the channel is uniformly distributed in $[1, \sum_{j=1}^{n} w_j]$.

We start with the results for the single channel case. We first execute Algorithm 1 while solving MCKP (in step 4) using the optimal algorithm described in [24]. Figure 3 shows the running time of Algorithm 1 as a function of the dependency rank and as a function of the number of flows (when the dependency rank is maximal, i.e., $\mathcal{D}(F) = \mathcal{R}(F) - 1$). It is evident that the execution time grows exponentially with $\mathcal{R}(F)$ for every $\mathcal{D}(F)$ value, and only polynomially (or even linearly) with the number of flows.

When the instance rank is high ($\mathcal{R}(F) > 20$), the running time of Algorithm 1 renders it impractical. In such cases we use Procedure 1 in order to reduce the rank of the instance before Algorithm 1 is invoked. Rank reduction might have a negative effect on the performance of Algorithm 1. In order to study this effect, we use as a benchmark instances with only positive dependencies. With such dependencies, the algorithm from [6] is shown to be optimal. Figure 4 shows the performance of Algorithm 1 with Procedure 1 for this benchmark. The x-axis is the dependency degree, whereas the y-axis is the profit ratio between the performance of Algorithm 1 with Procedure 1 and the performance of the optimum solution. As expected, as we increase the dependency rank and the degree, the profit ratio monotonically decreases. However, when we consider our requirements that a high dependency rank dictates a low dependency degree, we get solutions that are not far from the optimum.

Next, we study scenarios with multiple Doppler radars and 100 flows. We assume that each radar has the same bandwidth demand, but a different profit. The profit depends on the importance and size of the covered area. For both tactical and practical considerations, some areas are covered by more than a single radar. The purpose of this study is to show the importance of taking such dependencies into account. We thus compare the results of our model with the results of a model that ignores the dependencies and make a decision based on the largest profit/weight ratio (a simple Knapsack algorithm).

We study two profit scenarios. In Scenario A, the individual profits are all equal, whereas in Scenario B different profits are assigned to different flows. Each scenario is tested in two layouts, with different dependency ranks and different overlapping areas between dependent radars. The results are presented in Figure 5. The x-axis is the channel's normalized capacity, i.e., $c/\sum_{i\in N} w_i$, and the y-axis is the percentage of additional profit gained by taking the dependencies into account. In all of the graphs, as the average channel capacity and the overlapping areas grow, the benefit from taking dependencies into account increases. We can see a similar behavior when we increase the dependency rank from $\mathcal{R}(F) = 30$ to $\mathcal{R}(F) = 100$. We can also see that for the same $\mathcal{R}(F)$, the profit in Scenario B is larger that in Scenario A. The reason is that in Scenario B the dependency has a larger variance, and therefore a larger impact on the total profit.

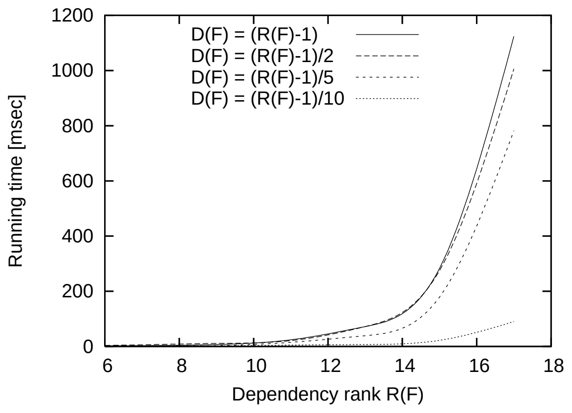
We can summarize that for the Doppler radar scenarios, when the dependencies between flows are not ignored, the performance of the radar sensor network substantially increases.

We now present simulation results for the performance of our algorithms in a mesh network over a routing tree. In these simulations, we run Algorithm 3 for $R = 1$. The output of this algorithm for R=1 is either feasible for the whole tree or infeasible for some of the nodes. If it is feasible, then the solution is also optimal. Otherwise, the algorithm is executed for higher value of R. It is easy to see that infeasible solutions are possible only if the capacity of the root is smaller than the capacity of some lower node. We are therefore interested in the probability that a solution of Algorithm 3 for $R = 1$ is feasible for the whole tree when the ratio α between the capacity of a level-$(i-1)$ tree node and the capacity of a level-i tree node, for every i, is smaller than 1.

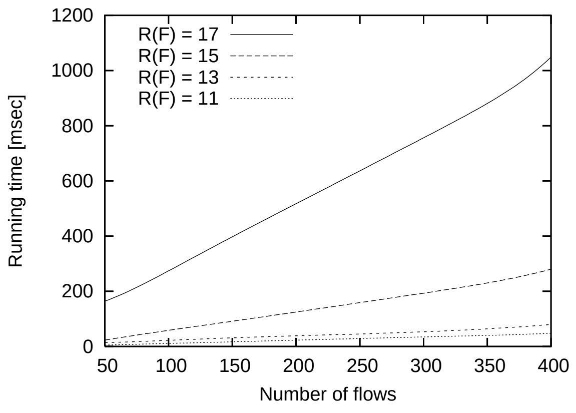
In Figure 6 we show the probability described above as a function of α. For this set of simulations, we generated flows from random trees in the following way. A routing tree has 3 levels. Every tree node has d children, where d ranges between 3 and 7, and there are 10 flows originating from every node. $\mathcal{R}(F) = 15$ and $\mathcal{D}(F) = 5$ are the maximum dependency rank and the average dependency degree of the considered instance. As before, the size of each diagonal block in the profit matrix $\mathcal{P}$ is uniformly distributed in $[1, \mathcal{R}(F)]$, and in each of the block's rows there are, on the average, $\mathcal{D}(F)$ non-zero entries. Each non-zero entry value $p_{ij} = p_{ji}$ is uniformly distributed in $[-r/2, r/2]$, where r is a constant for the whole matrix. The weight w_j of each item j is uniformly distributed in $[1, r/2]$. Finally, the capacity of the channel is uniformly distributed in $[1, \sum_{j=1}^{n} w_j]$.

As expected, when α gets closer to 1, the probability that a solution for the root is feasible for the whole tree gets also closer to 1. But we can see that for practical trees it is enough to have $\alpha \approx 0.5$. We can also see that when the number of children per node increases, the probability that a solution for the root is feasible for the rest of the tree increases as well. For instance, when $\alpha = 0.4$ we see that the probability is 1 if there are 7 children and 0.8 if there are only 3. The reason for this is that when there are more tree nodes, the flows originating in every level-i tree are assigned bandwidth from more level-i channels. Thus, it is less likely that a solution feasible for the root will impose intolerable load on a lower channel.

Figure 7 shows the feasibility of the solution as a function of the load for the case where each node has



(a) $n = 400$



(b) $\mathcal{D}(F) = \mathcal{R}(F) - 1$

Fig. 3. Execution time (in msec.) of Algorithm 1

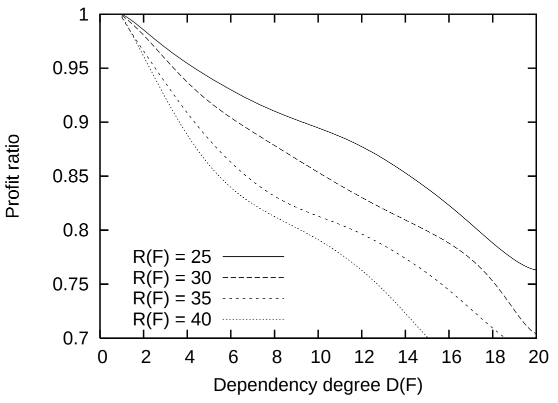


Fig. 4. The performance of Algorithm 1 with Procedure 1 divided by the performance of the optimal solution

4 or 5 children. Each graphs is for the same number of children, but for a different number of flows originating at every node. We can see in both graphs that when the number of flows increases, the probability that a solution feasible for the root is also feasible for the rest of the tree increases as well. The reason for this is that when there are more flows, Algorithm 3 is more likely to choose flows from different nodes, in which case the load imposed on a given non-root channel is smaller.

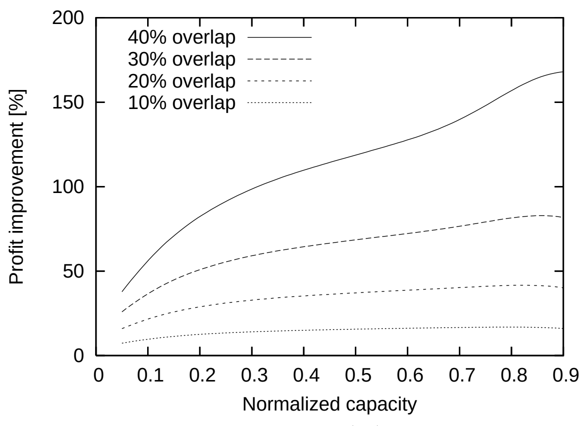
VII. Extension to Higher Order of Dependencies

We now discuss flows with higher dependency orders. Consider the three Doppler radars in Figure 8. Let S_A, S_B and S_C be the areas covered by each radar. Similarly, let S_{AB} be the area covered by both A and B, S_{ABC} the area covered by A, B and C, and so on. Table II shows the profit for every combination of flows. This profit is assumed to be proportional to the scanned areas.

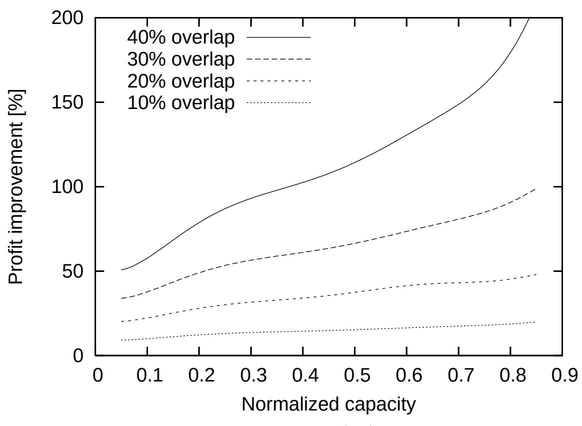
When the dependency order is higher than 2, the profit function $\mathcal{P}$ can no longer be represented by a matrix, D-UFP is no longer equivalent to QKP, and Algorithm 1 cannot be used. The profit to be gained from a given combination of flows must be now calculated in an entirely different manner. Let this combination be represented by a binary vector v, as in Algorithm 1, and let the profit gained from each combination be $\mathcal{P}(v)$. Then we can adapt Algorithm 1, as follows, to handle dependencies of higher orders.

An MCKP instance is constructed by defining m MCKP classes, where m is the number of mutually disjoint subsets (see Definition 2). Each class C_i consists of $2^{|F_i|}$ binary vectors, and vector $v \in C_i$ represents one combination items chosen from F_i. The profit for each combination is determined by $\mathcal{P}(v)$. The time complexity for this construction is similar to that discussed in Section IV-B, where only dependencies between pairs were considered. In addition we need to revise Procedure 1. A D-UFP instance can no longer be represented as a simple graph but by a hyper-graph. In this hyper-graph, each node indicates a single flow, and each edge, connecting any number of nodes, represents the extra profit gained by accommodating all the flows represented by the connected nodes. Procedure 1 should go through the list of edges of the relevant hyper-graph, sorted by their weight. Any edge that exceeds the segment size criterion is removed from the graph.

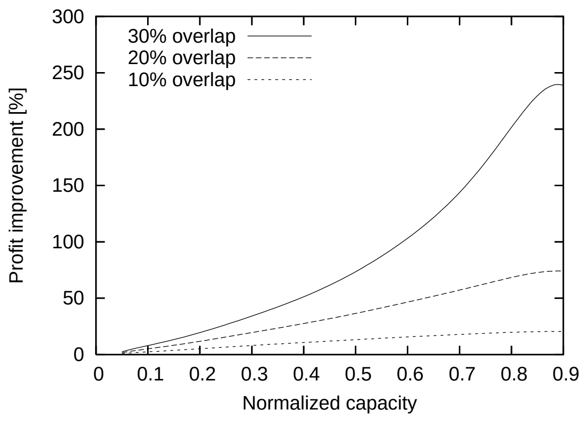
Next we compare the profit gained when dependency orders of both 2 and 3 are taken into account to the profit gained when only a dependency order of 2 is considered. As in Scenario B in Section VI, the profit assigned to each flow is proportional to the scanned areas (see Table II). The results of the comparison are presented in the 6 graphs in Figure 9. For each graph, the x-axis is the channel's normalized capacity, i.e., $c/\sum_{i \in N} w_i$, and the y-axis is the percentage of extra profit gained by not



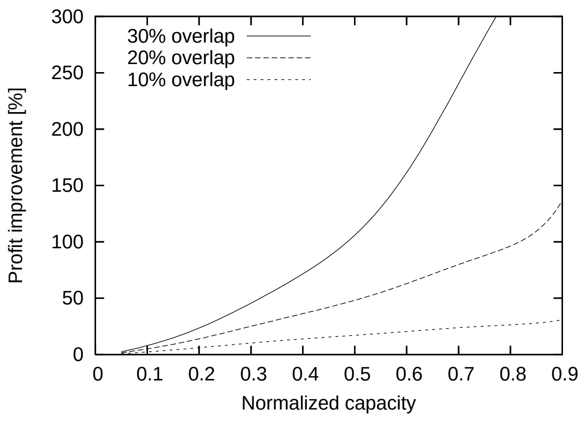
(a) Scenario A. $\mathcal{R}(F) = 30$



(b) Scenario A. $\mathcal{R}(F) = 100$



(c) Scenario B. $\mathcal{R}(F) = 30$



(d) Scenario B. $\mathcal{R}(F) = 100$

Fig. 5. Profit improvement due to considered dependency

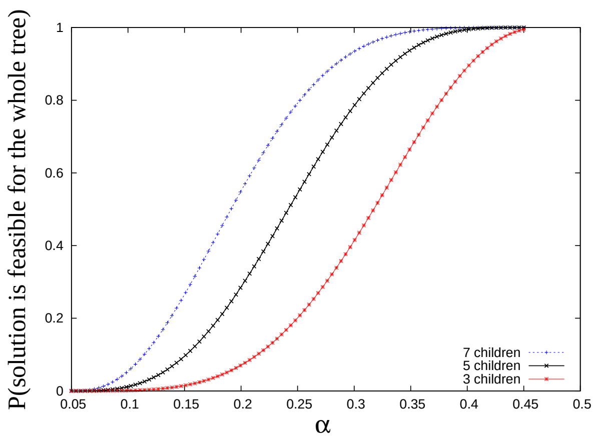


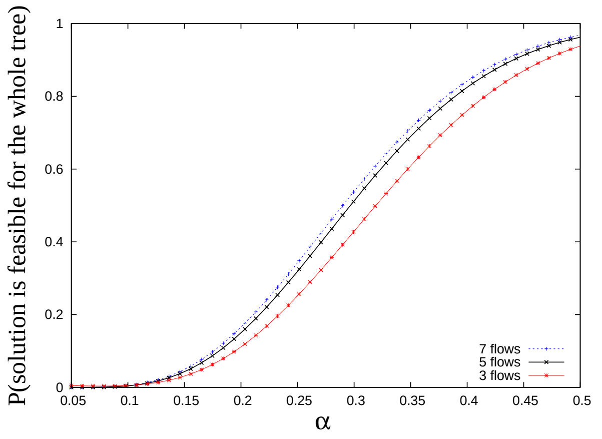
Fig. 6. Feasibility as a function of α and number of children

radars	assigned profit
A	S_A
B	S_B
C	S_C
A, B	$S_A + S_B - S_{AB}$
A, C	$S_A + S_C - S_{AC}$
B, C	$S_B + S_C - S_{BC}$
A, B, C	$S_A + S_B + S_C - S_{AB} - S_{AC} - S_{BC} + S_{ABC}$

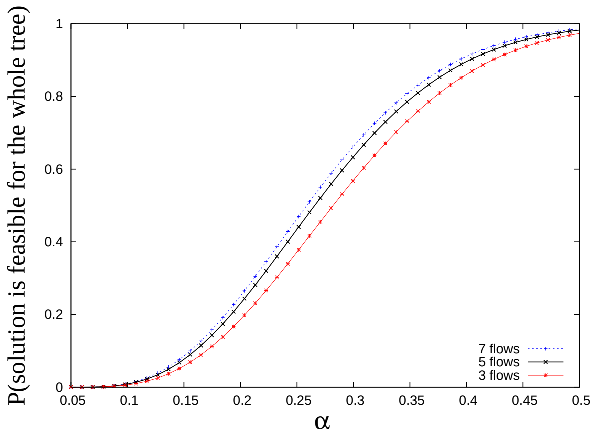
TABLE II
PROFIT ASSIGNMENT BASED ON THE AREA COVERED BY EVERY RADAR

ignoring a dependency order of 3. Each graph depicts two curves. The first shows the extra profit gained by taking a dependency order of 2 into account. The second shows the extra profit gained by taking a dependency order of 3 into account as well. We consider two $\mathcal{R}(F)$ values: 30 and 100. We also consider three values of area overlap: $10\%, 20\%$ and 30%.

As expected, Figure 9 indicates that the performance gain is larger when the percentage of overlap and the dependency rank are larger. For example, consider Figure 9(a), where the overlapping area is 30% and $\mathcal{R}(F)$ is 30. When the channel's capacity is 70% of the total bandwidth demand, performance improves by more that 50% over the case, shown in Figure 9(c), where the



(a) 4 children



(b) 5 children

Fig. 7. Feasibility as a function of α and load

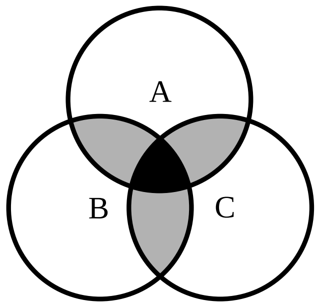


Fig. 8. Three 360° radars with $\mathcal{O} = 3$

overlap is only 20%, and it improves by more than 100% over the case, shown in Figure 9(e), where the overlap is only 10%. The improvement results for a larger dependency rank, shown in Figure 9 (b), (d) and (f), where it is equal to 100 are even stronger.

Our key conclusion, however, is that taking into account a dependency order of 3 rather than of 2 alone yields much less benefit than taking into account a dependency order of 2 rather than of 1 alone. The main reason for this is that the area mutually scanned by any three radars is always smaller than the area scanned by any two radars of the same group, i.e., $S_{ABC} \leq S_{AB}, S_{BC}, S_{AC}$.

Although we considered here only one example for which taking into account a dependency order of 3 is not considerably more beneficial than taking in account a dependency order of 2 alone, we believe that the same conclusion holds for most of the practical scenarios.

VIII. Conclusions

We defined and studied a new problem, referred to as the Dependent Unsplittable Flow Problem (D-UFP), in the context of large-scale powerful (radar/camera) sensor networks. In order to optimize the selection of flows transmitted to the gateway, D-UFP takes into account possible dependencies between flows. D-UFP is a very difficult problem, even under many constraints. When the network consists of one shared channel, D-UFP was shown to be equivalent to QKP, which is NP-hard in the strong sense and has no good approximation. We presented an efficient algorithm for this case of D-UFP, under several constraints, and for the case where the sensors form a spanning tree. Among other things, our simulation results revealed that taking the dependencies into account can increase the performance of a typical Doppler radar system by $10 - 100\%$.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E.Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4), 2002.

[2] M. Andrews, J. Chuzhoy, S. Khanna, and L. Zhang. Hardness of the undirected edge-disjoint paths problem with congestion. In *Foundations of Computer Science (FOCS)*.

[3] M. Andrews and L. Zhang. Hardness of the undirected edge-disjoint paths problem. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing (STOC)*, 2005.

[4] M. Andrews and L. Zhang. Logarithmic hardness of the undirected edge-disjoint paths problem. *Journal of ACM*, Sept. 2006.

[5] Y. Azar and O. Regev. Combinatorial algorithms for the unsplittable flow problem. *Algorithmica*, 44, 2006.

[6] A. Caprara, D. Pisinger, and P. Toth. Exact solution of the quadratic knapsack problem. *Informs Journal on Computing*, 11:125–137, 1998.

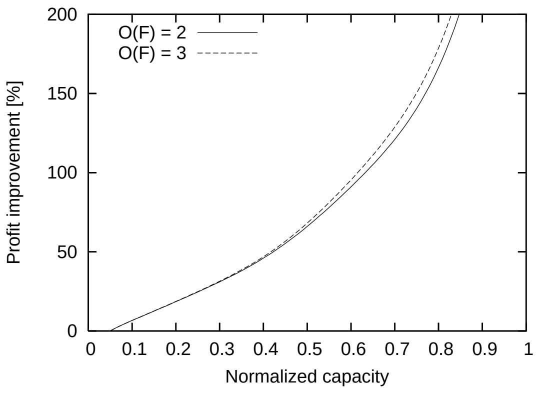
[7] A. Chakrabarti, C. Chekuri, A. Gupta, and A. Kumar. Approximation algorithms for the unsplittable flow problem. In *APPROX*, pages 51–66, 2002.

[8] A. Chakrabarti, C. Chekuri, A. Gupta, and A. Kumar. Approximation algorithms for the unsplittable flow problem. *Algorithmica*, 47, 2007.

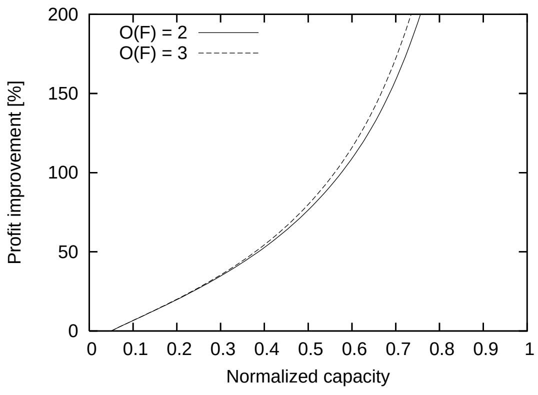
[9] C. Chong and S. Kumar. Sensor networks: evolution, opportunities, and challenges. *Proc. of the IEEE*, 9(8).

[10] B. Donovan, D. J. McLaughlin, and J. Kurose. Principles and design considerations for short-range energy balanced radar networks. In *IGARSS*, 2005.

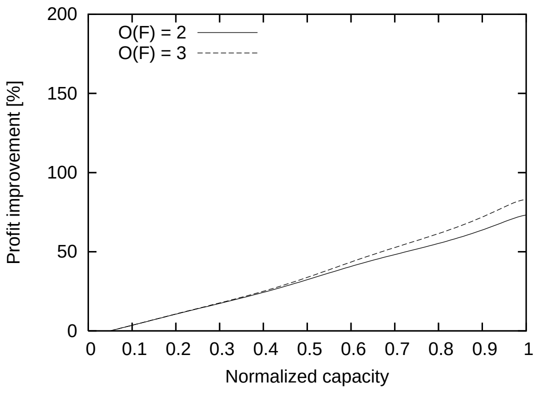
[11] P. K. Dutta, A. K. Arora, and S. B. Bibyk. Towards radar enabled sensor networks. In *IPSN*, 2006.



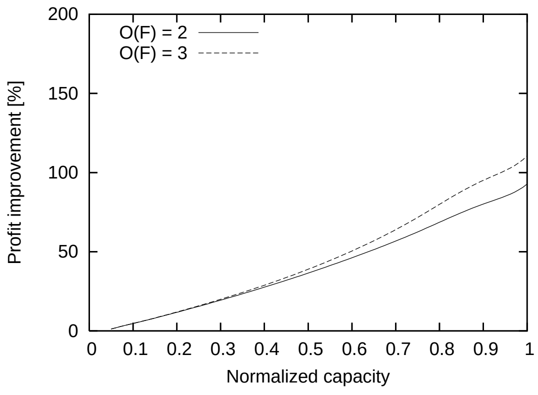
(a) 30% overlap $\mathcal{R}(F) = 30$



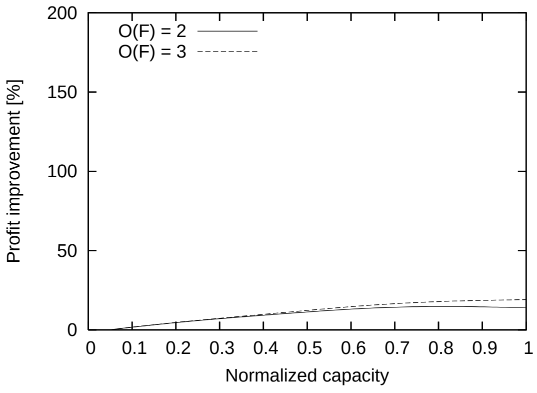
(b) 30% overlap $\mathcal{R}(F) = 100$



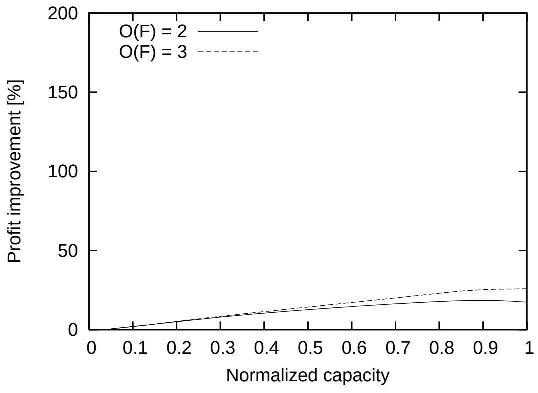
(c) 20% overlap $\mathcal{R}(F) = 30$



(d) 20% overlap $\mathcal{R}(F) = 100$



(e) 10% overlap $\mathcal{R}(F) = 30$



(f) 10% overlap $\mathcal{R}(F) = 100$

Fig. 9. Profit improvement due to third order dependency

[12] N. Garg, V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, 1997.

[13] W. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *MobiCom*, 1999.

[14] C. Helmberg. Semidefinite programming for combinatorial optimization. Technical report, Konrad-Zuse-Zentrum, Berlin, Oct. 2000. ZIP-Report ZR-00-34, Habilitationsschrift.

[15] C. Helmberg, F. Rendl, and R. Weismantel. Quadratic knapsack relaxations using cutting planes. In *Proceedings of the 5th International IPCO Conference on Integer Programming and Combinatorial Optimization*, pages 175–189, London, 1996. Springer-Verlag.

[16] C. Helmberg, F. Rendl, and R. Weismantel. A semidefinite programming approach to the quadratic knapsack problem. *Journal of Combinatorial Optimization*, 4:197–215, 2000.

[17] P. Kolman and C. Scheideler. Improved bounds for the unsplittable flow problem. In *SODA*.

[18] B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, Feb 1956.

[19] J. Kurose, E. Lyons, D. McLaughlin, D. Pepyne, B. Philips, D. Westbrook, and M. Zink. An end-user-responsive sensor

network architecture for hazardous weather detection, prediction and response. In *AINTEC*, 2006.

[20] E. L. Lawler. Fast approximation algorithms for knapsack problems. *Symposium on Foundations of Computer Science*, 0:206–213, 1977.

[21] M. Li, T. Yan, D. Ganesan, E. Lyons, P. Shenoy, A. Venkataramani, and M. Zink. Multi-user data sharing in radar sensor networks. In *SenSys*, 2007.

[22] A. Nasipuri and K. Li. A directionality based location discovery scheme for wireless sensor networks. In *ACM WSNA'02*.

[23] B. Patt-Shamir and D. Rawitz. Vector bin packing with multiple-choice. In *SWAT*, 2010.

[24] D. Pisinger. A minimal algorithm for the multiple-choice knapsack problem. *European Journal of Operational Research*, 83:394–410, 1994.

[25] D. J. Rader and G. J. Woeginger. The quadratic 0-1 knapsack problem with series-parallel support. *Operations Research Letters*, 30(3):159–166, 2002.

[26] A. Savvides and M. B. Srivastava. Location discovery. *Mobile Ad Hoc Networking*, 2003.

[27] M. M. Sorensen. *A Polyhedral Approach to Graph Partitioning*. PhD thesis, The Aarhus School of Business, Denmark, 1995.

[28] S. Soro and W. Heinzelman. A survey of visual sensor networks. *Advances in Multimedia*, 2009.

[29] Y. Zhou. Improved multi-unit auction clearing algorithms with interval (multiple-choice) knapsack problems. In *ISAAC'2006*.