An Event-Based Data Fusion Algorithm for Smart Cities

Avinash Kalyanaraman
Department of Computer Science
University of Virginia
Charlottesville, VA 22903, USA
ak3ka@virginia.edu

Kamin Whitehouse
Department of Computer Science
University of Virginia
Charlottesville, VA 22903, USA
whitehouse@virginia.edu

Ubicomp/ISWC'15 Adjunct, September 7-11, 2015, Osaka, Japan

http://dx.doi.org/10.1145/2800835.2801630

Abstract
The last decade has seen a considerable increase in the number of sensors we interact with on a daily basis. However, it is not always possible for a single sensing system to capture the complete story. While statically mounted infrastructure sensors typically capture the *what, where, how much etc* aspects of a detected event , e.g. (*what appliance was used, how much energy did it consume*), they do not always answer the *who* question. On the other hand, the advent of wearables has helped answer the *what and who* aspects - e.g. (*who used the appliance*). Fusing such sensor streams that observe the same event but different attributes of it, opens up many interesting applications. In this paper, we present a globally optimal data fusion algorithm for such pairs of systems, and show why traditional bipartite algorithms do not work. We evaluate our algorithm against two greedy baselines and show that our algorithm has lesser variance in the presence of time skew, false positives and false negatives.

Author Keywords
Sensor Networks; Data Fusion; Bipartite Matching

ACM Classification Keywords
C.3 [Special-Purpose and Application-Based Systems]: Real-time and Embedded Systems; G.2.2 [Graph algorithms]: Graph Theory

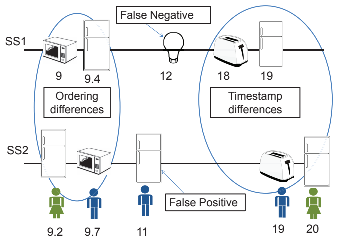

Figure 1: Data Fusion needs to be performed between the static sensing system (SS1) and personal sensing system (SS2) in the presence of ordering differences, false positives, false negatives and timestamp differences

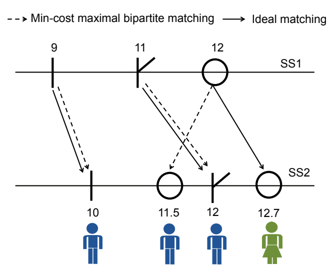

Figure 2: Timelines showing min-cost maximal bipartite match resulting in crossing matches per person during False Positives & Negatives. Each shape represents a unique event type

Introduction

It is reported that nearly 50% of the world's population live in cities, and this value is expected to be 70% by 2050 [15]. With such high concentration of population in cities, there is a growing need to build cities that cater to the needs of the people, improve their quality of life and make optimal use of public resources. On the other hand, the last decade has also seen a proliferation of sensors we interact with on a daily basis. For e.g., in 2013 it was estimated that over 17.2 billion smart home devices alone were sold worldwide. It is believed that such sensing devices will become so commonplace that there will be around 50 billion sensors by 2020 [11].

However it is not always possible for a single type of sensor system to capture the complete story. For instance, statically mounted sensors can capture information on the *"what, how much, where"* aspects, but not answer the personalization question of *who* triggered the event - e.g. what appliance was used, how much water was consumed etc. We refer to such a system as the *static sensing system*. On the other hand, the growth in identity-mounted wearables like Fitbit, smart watches etc, and activity recognition techniques using these wearables [9], have helped answer the *what and who* aspects - e.g. who used the fixture and what appliance was used. We call this system as the *personal sensing system*. Consequently, we have multiple *non time synchronized* heterogeneous sensors with time-stamping errors, each observing the same event (what fixture was used) but having different attributes (who used the fixture v/s how much energy did it consume?). Performing data fusion on the common observed event in such time skewed static-personal sensor streams opens up many useful applications (e.g. Bob consumed 100W when he used the microwave). The challenge in such fusion is the temporal ordering of the observed events in the two streams, the presence of timestamp differences, false positives (FPs) and negatives (FNs). Figure 1 shows an example of how events can show up in the two timelines. We present three diverse motivating smart-city examples warranting such fusion, later in this paper - (a) an energy apportionment application that can provide people with a city-level energy footprint, (b) a meta-research tracking application to evaluate tracking accuracy by comparing against ground-truth and (c) an automatic dietary monitoring application to help obesity control, a growing problem in cities.

The systems under fusion have two main properties which prevent applicability of traditional bipartite matching algorithms - (a) All observed events in the static sensing system have global time ordering (e.g. energy disaggregation techniques which sense the electrical mains), and (b) In the personal sensing system timeline, all events of a given identity are temporally ordered, but there is no event ordering across identities (e.g. If each person carries a smartwatch, all detected appliance usages of a person are in order, but there is no global time-ordering across persons). Any matching between the two systems must adhere to these properties. Traditional bipartite algorithms [8, 3, 16] violate these properties as they have no notion of per-identity ordering and simply maximize the matches at the lowest cost. Consequently, they can cause crossing matches for a given person. An example for such a crossing is shown in Figure 2. A crossing represents an ordering contradiction between the two systems- for e.g. SS1 says Bob used the Fridge(SS1:11) before the Oven(SS1:12), while the crossing match implies that SS2 says he used the Oven(SS2:11.5) before the Fridge(SS2:12).

Consequently, we present a new globally optimal algorithm which performs data fusion on such static-personal sensing streams. We first augment a naive combinatorial approach with an optimal pruning strategy and show that such an approach still consumes unnecessary computational resources. We address this limitation via a Divide and Conquer approach resulting in an asymptotic of O(|P1| * |P2| * ...

|Pn|), where |Pi| is the number of events of the ith identity in a sub-problem. We compare our algorithm against two greedy baselines and show that despite time skews, FPs and FNs, we exhibit much lower accuracy variance.

Motivating examples

We motivate the need for a fusion algorithm between static-personal sensor systems with the following examples:

Our first example is motivated by the work of Ranjan [12]. Homes are one of the major energy consumers in a city, using up nearly 40% of the total US energy budget [1]. Providing personalized energy feedback to residents can potentially reduce 20-50% of a home's energy usage. Building such an application requires answers for - "What fixture was used? How much energy did it consume? Who used it?". Non Intrusive Load Monitoring (NILM) [5] techniques help disaggregate fixture usages by monitoring the mains, thus answering the first two questions - viz *(Oven, 100W, 12:30pm)*. A second wearable-based system that recognizes fixture usages via IMU sensors mounted on the hands of the user (like a smartwatch) answers - "what fixture was used, and who used it?" - viz *(Oven, Bob, 12:31pm)*. We see that the two systems observe the same event (Oven) but different attributes of it (Bob v/s 100W). Performing data fusion on the common Oven usage event results in our desired energy apportionment tuple - (Bob, Oven, 100W) denoting *"Bob consumed 100W by using the Oven"*. Extending such an application to all buildings in a city, can help provide individuals with a detailed city-level energy footprint.

Our next use-case of performing data fusion on the same observed event is a case of meta-research motivated by Hnat [6]. Here the events of a room-level tracking system under test and a ground truth system need to be matched for accuracy evaluation purposes. The tracking system is mounted on every doorway and tracks people based on heights, as they move between rooms. Ground truth for such systems is typically collected by people recording their room transitions on phones. Both systems observe the same event (doorway crossings) but different attributes (height v/s person). Accuracy is evaluated by comparing what tracking thought happened on each event against the corresponding matched phone event. For e.g. if tracking says someone went from Bedroom to Hallway, but the matched ground truth says the opposite (Hallway to Bedroom), then tracking was inaccurate on that event. While we motivate on a tracking use-case in homes, such a study can also be performed for tracking in public places like museums (or) envisioned for futuristic vehicular tracking wherein say, the magnetic field of the car is measured as a biometric in each intersection, and ground truth is collected via GPS mounted on cars.

Our third static-personal data fusion use-case is in the realm of smart-living with automatic dietary monitoring. Obesity, touted by the World Health Organization as the '21st century epidemic', is an important problem for cities to deal with as over 33% of adults living in urban United States are obese [2]. One of the main causes for obesity is the lack of proper quantitative methods to measure energy intake [14], with self-reporting of food consumption suffering from reporting-error and low adherence. Consequently, there has been a growing need for automatic dietary monitoring. Performing data fusion on the works of Kranz [7] and Moncada-Torres [10] does the same. Here, the static sensing system is Kranz's microphone system which uses acoustic data to detect the food item being cooked -e.g. *(Apple, cutting, 12:30pm)*. The personal sensing system of Moncada-Torres consisting of IMUs detects the same cooking event but has an identity attribute to it - e.g. *(Bob, cutting, 12:31pm)*. Performing data fusion on the commonly observed cooking event can help with dietary monitoring by saying (Bob, cutting, Apple). All three applications warranting data fusion have a generic set of properties

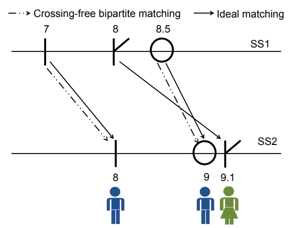

Figure 3: Timelines showing crossing free bipartite match eliminating valid crossings across persons. Both (8, 9.1) and (8.5, 9) can match

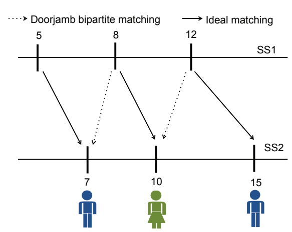

Figure 4: Timelines showing Doorjamb matching minimizing total cost instead of maximizing associations resulting in incorrect (8, 7) and (12, 10) matchings

which we next explain as the *Matching problem*.

Matching problem

The common characteristics between the aforementioned use-cases can be generalized as follows:

1. There are 2 sensing systems (SS) observing the same set of discrete events (e.g. appliance usage or doorway crossing) but having different attributes (e.g. energy consumed vs identity using).
2. Each SS is an infinite time-series that is not in time-sync with each other.
3. One of the SS has notion of identity (e.g. person or car). There is a per-identity time-ordering of events in it (e.g. if each person carries a smartwatch, all detected appliance usages of a person are in order, but there is no global time-ordering across persons).
4. All events are globally ordered in the other SS. (e.g. NILM timeline where fixture usage events are disaggregated by monitoring the mains)
5. Each SS may observe events which the other doesn't see - i.e false positives and false negatives occur.
6. An event in one SS can only be matched with the same event in the other SS. For e.g. a NILM disaggregated Microwave event can match with a gesture recognized Microwave event but not a Fridge event.

Our goal is to find a globally optimal valid matching between the two SS so as to maximize the number of matches, such that the total cost is minimum. While we use the time difference between each association as the mapping cost in this paper, we point out that the cost could be set by the application as well. Ordering constraints enforced by properties (4) and (5) result in the following matching restriction: *"there can be crossing matches across identities, but no crossing matches for a given identity"*. This is because a crossing match denotes an event-ordering contradiction between the two systems. For e.g., in Figure 2, SS1 says Bob used the Fridge(SS1:11) before the Oven(SS1:12), while the crossing implies that SS2 says he used the Oven(SS2:11.5) before the Fridge(SS2:12). It is this matching constraint that prevents direct applicability of traditional bipartite matching algorithms to our problem. The solid lines in Figures 2, 3 and 4 show examples of valid matching assignments.

Related Work

Considerable literature exists on minimum cost maximum bipartite matching [8, 3, 16]. However, as mentioned before such algorithms aren't applicable here as they cause crossing matches for a given identity. At the other end, algorithms for crossing free maximum bipartite matching have been proposed by Fredman [4] and Widmayer [17]. However, these algorithms eliminate valid crossings across persons too. Figure 3 shows an example of valid crossings getting eliminated during timestamp errors. A greedy solution to this problem has been used in Doorjamb[6]. Here, after sorting all matches by weight, each edge from this sorted list is removed in-order and added to a final list if it is a valid match. The criteria to minimize total cost rather than maximize associations makes such an algorithm undesirable. Figure 4 shows an example of incorrect matching during time skews, arising due to total cost minimization.

Approach

In this section, we start with a naive brute-force algorithm and expose its limitations. We then explain an optimal pruning strategy to address them. However, such an approach still maintains a lot of unnecessary matches that consume needless computational resources. We then explain how to address this limitation via a Divide And Conquer technique.

Before we describe the approach, we first define our notations. Let SS_1 and SS_2 be the static and personal sensing systems respectively whose events need to be matched. As mentioned before, there is global event ordering in SS_1 and per-identity event ordering in SS_2. Let SS_{ki} denote the i^{th} event in the k^{th} sensing system SS_k. Finally, let ϵ be the maximum time skew between the two systems -i.e. any event in one system must have it's corresponding match in the other atmost ϵ time units away. Having defined our notations, we describe our algorithm by first explaining the limitations of the *Naive* algorithm.

Naive MHT algorithm:
The *Naive* approach is a variant of Reid's Multiple Hypothesis Tracking (MHT) [13] wherein a list of all possible valid matchings between the two systems are generated. Each such sequence of valid matching is called a *hypothesis*. An (SS_{1i}, SS_{2j}) matching is valid if the events SS_{1i} and SS_{2j} are identical, |timestamp(SS_{1i}) - timestamp(SS_{2j})| $\leq \epsilon$, and the addition of the matching results in no crossing match per person in that hypothesis. Finally, the hypothesis having the most associations at a minimum cost is chosen.

Figure 5 shows an example of the progression of the *Naive* algorithm. In this figure, each level of the tree represents an event being observed, and the path from root node to any node at a given level represents a hypothesis. In this example, event (a) can potentially be matched to events (d), (e), (f) [as they are all within the time window] or (a) could be a false positive (ϕ). Each of these cases represents a hypothesis. Upon observing the next event (b), hypothesis (a→d), progresses as (a→d, b→e), (a→d, b→f) and (a→d, b→ ϕ). Note that (b→d) isn't a valid match, since (d) has already been associated with (a) by this hypothesis. Finally, that hypothesis which has the maximum number of associations with a minimum cost is chosen. It can be seen that such an approach is computationally not tractable as the hypothesis space grows quickly, resulting in the maintenance of an exponential number of hypotheses. This exponential explosion warrants the need for hypotheses pruning.

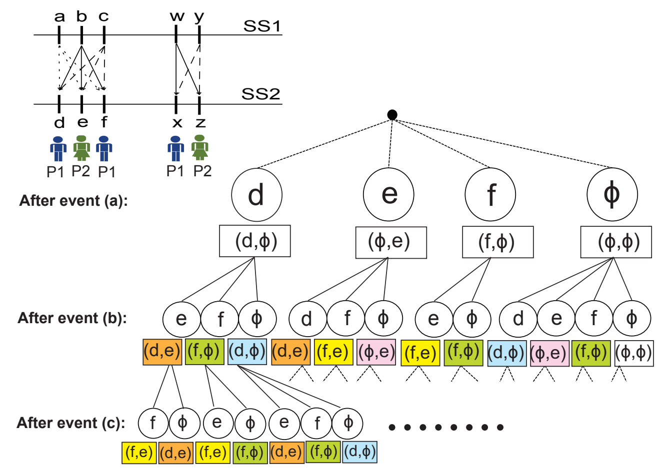

Figure 5: Progression of MHT for first three SS1 events - (a), (b) and (c). Each level represents an SS1 event being observed. The path from root to any node represents a hypothesis. Circles show the SS2 event matched with the observed SS1 event. For e.g. circle (d) in level 1 denotes an a→d hypothesis. ϕ denotes a False Positive. Squares represent the state of that hypothesis. All like-colored squares represent hypotheses in the same state. (Figure best viewed in color)

Naive MHT with Optimal Pruning:
Looking at Figure 5, it is seen that besides storing the associations, each hypothesis also stores its last matched SS_2 timestamp of each person. We refer to this as the *state* of a hypothesis. For example, in Figure 5, hypothesis (a→d, b→e) and hypothesis (a→e, b→d) are in the same state *(d,e)*. We point out that: *Two hypotheses in the same state will behave identically moving forward.* This can be explained as follows : let H_1 and H_2 be two hypotheses in the same

state. Upon a new event SS_{1i} in SS_1, the validity of the (SS_{1i}, SS_{2j}) mapping will be the same in H1 and H2. This is because since both H1 and H2 are in identical state (i.e the last associated timestamp of each person are the same), a crossing match in H1 will cross in H2 too.

Given this intuition that two hypotheses in the same state will progress identically, we define an optimal pruning condition that maintains only one of them - "if two hypotheses are in the same state, then we only retain the 'better' hypotheses." We say H1 is better than H2 if: (1) the number of associations in H1 is greater than H2, and (2) If the number of associations are equal, H1 has a lower cost than H2.

Such a pruning strategy is optimal because if two hypotheses are in the same state, and one of them is better than the other (say H1 is better than H2), then the hypotheses forked out of H1, will always be better than those forked out of H2. In our example, after observation (b), hypothesis (a→d, b→e) and hypothesis (a→e, b→d) have the same state. Moving forward, (a→d, b→e, c→X) will always be better than (a→e, b→d, c→X), for any valid X. This is because (|d-a| + |e-b|) < (|d-b| + |e-a|). Consequently the hypothesis (a→e, b→d) can be pruned after event (b). The MHT algorithm after pruning now maintains O(|P1| * |P2| * ... |Pn|) hypotheses, where |Pi| is the number of events of the i^{th} identity in SS_2.

The Divide And Conquer Approach:
However, such an approach still maintains a lot of unnecessary hypotheses. For e.g., in Figure 5, it can be seen that after event (c), the set of hypotheses which exist will include: *{ [(a→d), (b→e), (c→ ϕ)], [(a→e), (b→d), (c→ ϕ)], [(a→d), (b→f), (c:→ ϕ)] ... }* . These hypotheses will be worse than the [(a→d), (b→e), (c→f)] hypothesis (according to the comparison function defined previously). More importantly, any hypotheses forked from the above list will always be worse than those forked from [(a→d), (b→e), (c→f)]. We can say this with certainty because no event $\geq$ (w) can potentially match with an event $\leq$ (f). Consequently, in this example [SS_1 = {a,b,c} , SS_2={d,e,f}] and [SS_1 = {w,y} and SS_2={x,z}] can be treated as two independent sub-problems.

The question now becomes : "How to partition the given matching problem into sub-problems such that each sub-problem can be solved independent of the rest?". For this, we need to identify a disjoint collection of ({SS_1}, {SS_2}) events-set tuples such that no SS_1 event outside of any given tuple can potentially match itself with an SS_2 event within the same tuple.We refer to this requirement as the *tuple exclusiveness requirement*. Each such tuple is an independent matching problem by itself. This is achieved using a Dynamic Programming technique with the following recurrence equations such that all events between two consecutive 'True' *is_end* become part of the same sub-problem:

For each event *i* in SS1:

$$\begin{aligned} cum\_max(i) &= max(cum\_max(i-1),\ SS1(i).max\_ss2),\ i\ >\ 0 \\ cum\_min(i) &= min(cum\_min(i+1),\ SS1(i).min\_ss2),\ i\ >\ 0 \\ is\_end(i) &= True,\quad if\ cum\_max(i)\ <\ cum\_min(i+1) \\ &= False,\ \ else \quad 0 \leq i < |SS1|-1 \end{aligned}$$

The recurrence is explained as follows, for every event *i* in SS_1: *cum_max(i)* keeps track of the maximum SS_2 event matchable by any SS_1 event preceding *i* (including itself). *cum_min(i)* keeps track of the minimum SS_2 event matchable by any SS_1 event following *i* (including itself). Consequently, *is_end(i)* will be True for the i^{th} SS_1 event only when all events upto *i* can only potentially match with an SS_2 event that occurs strictly before any SS_2 event matchable by all events succeeding *i*. Under such circumstances, events following *i* can be part of a separate sub-problem, as they abide by the *tuple exclusiveness requirement*.

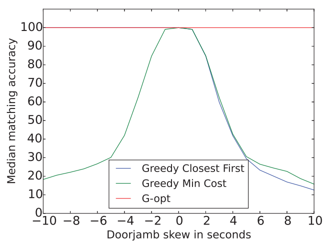

Figure 6: Median accuracy for varying Doorjamb skew with no FPs - Greedy algorithms show high accuracy variance. This is because greater the skew, higher the likelihood finding a local optima

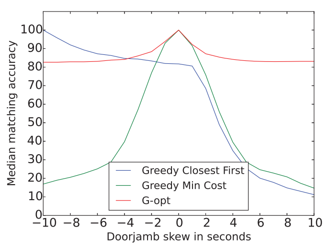

Figure 7: Median accuracy for varying Doorjamb skew with 20% FPs - All algorithms suffer an accuracy drop. G-opt has lower accuracy variance.

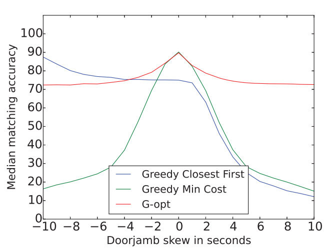

Figure 8: Median accuracy with time skew, 20% FP and 10%FN - All algorithms suffer an accuracy drop as only the phone system observes the event. G-opt still has lower variance

This technique of identifying sub-problems and running MHT + Pruning within each, results in maintenance of O(|P1| * |P2| * ... |Pn|) hypotheses, where |Pi| is the number of events of the ith identity in a sub-problem. This is a significant reduction from the previous *MHT* $+$ *Pruning* algorithm.

Experimental setup

We evaluate our algorithm with a Doorjamb [6] **(DJ)** like setup with 2 persons walking around a home with a smartphone. As they moved between rooms, they clicked on their phone indicating the doorway of transition, and the direction of movement. Each recorded event is of the form *(Timestamp, Doorway, Person, Direction)*. An identical second timeline is created for the DJ system. We empirically study the effect of time skew by offsetting DJ timestamps by -10 to 10 seconds. Consequently, we set ϵ, the maximum time skew between the two systems to be 10 seconds. We vary the false positive (FP) percentage as 0 and 20% of phone events. We generate FPs according to a uniform distribution - i.e an FP is equally likely to occur between any pair of DJ events. We set the FP timestamp to the middle of the two surrounding events. The doorway of the FP is randomly chosen between the doorways of the two surrounding events. We study the effect of False Negatives (FNs) by removing 10% of phone events. FNs are chosen uniformly - i.e. each phone event is equally likely to be a FN in DJ. In all, a total of 100 trials was performed. In this setup, there were 11 doorways in the house, and a total of 438 doorway crossing events.

Our metric of interest is *Matching Accuracy* calculated as the fraction of phone transition events correctly matched with its DJ event. We compare our globally optimal matching algorithm, ***G-opt*** against two baselines which adhere to the aforementioned *Matching problem* properties.

1. Greedy Closest Match: Moving in increasing time order, each DJ event is matched with its closest event for the same doorway in the phone system, such that no crossing occurs for a given person.
2. Greedy Min Cost: We use the greedy algorithm of Hnat [6], which was described in our Related Work.

Evaluation

Figures 6, 7 and 8 show the median matching accuracy of the three matching algorithms in the presence of skew, FP and FN. We see that *G-opt* has far lower accuracy variance than its greedy counterparts despite skew, FPs and FNs. The greedy algorithms suffer a greater accuracy dip with skew because larger skews result in a higher likelihood of finding a closer incorrect match (local optima). We observe that *Greedy Closest Match* has a bias if the skew is opposite to the direction of traversal. We see all algorithms suffer an accuracy drop with FPs. This is because skews can cause the FP to be incorrectly paired with the phone event, as it may be closer in time. The *Greedy Closest Match* has a lower accuracy dip during a negative skew despite FPs because (a) skew has no effect since traversal is in increasing time order, and (b) the injected FP must choose the succeeding event's doorway to cause an accuracy drop. In both *Greedy Min Cost* and *G-opt* choosing either events' doorway can lower accuracy. All algorithms suffer an accuracy drop with FNs as only the phone system observes the event.

G-opt suffers the most slowdown (i.e maintains the most hypothesis) when the same event is performed by multiple people within ϵ, the maximum time skew between the two systems. Consequently, each event in SS1 can potentially match to the events caused by each person in SS2, resulting in an exponential number of hypothesis. Figure 9 shows a sensitivity analysis varying the count of persons causing the same event within ϵ, and seeing the runtime. We see that *G-opt's* runtime is negligible for upto 13 persons. Even for a higher count of 15 identities, the runtime is still around

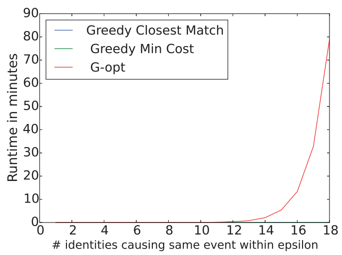

Figure 9: As the number of persons performing the same event within ϵ (the maximum time skew between the two systems) increases, G-opt's runtime remains negligible for as high as 13 persons. The runtime of the greedy algorithms are always negligible.

SP1:SP2 event ratio	% Speedup
100 : 0	110
90 : 10	151
80 : 20	240
70 : 30	398
60 : 40	558
50 : 50	631

Table 1: Speedup of Divide And Conquer approach over just MHT + pruning for a 100 event sample. Smaller the difference in number of events between two sub-problems, greater the speedup because each sub-problem maintains a lesser number of hypothesis.

5minutes only. The runtime of the greedy algorithms are always negligible. However, with such large number of persons within ϵ, greedy algorithms' accuracy will suffer as there is greater likelihood of finding the local optima.

Table 1 shows the benefit of performing *Divide And Conquer* by comparing its runtime over just MHT + pruning. Given a set of 100 events, we varied the number of events within each sub-problem, starting from all events in one and none in the other, and ending with an equal split of events. We noticed that smaller differences in number of events between two sub-problems results in greater speed-up due to the maintenance of lesser number of hypotheses.

Conclusion

In this paper, after motivating on three diverse use-cases, we present a new globally optimal data fusion algorithm for static-personal sensing systems. The algorithm builds over an MHT approach with an optimal pruning strategy. Such a technique still maintains several unnecessary matches that needlessly consume computational resources. We address this limitation via a Divide And Conquer technique which results in the maintenance of O(|P1| * |P2| * ... |Pn|) hypotheses, where |Pi| is the number of events of the ith identity in a sub-problem. Our results also show that our globally optimal algorithm has lesser variance than two greedy counterparts in the presence of time skew, false detections and missed detections. We believe that such a fusion algorithm will become more important with time, as more diverse sensors get deployed for smart city applications.

REFERENCES

1. 2009. Residential Energy Consumption Survey. http://www.eia.gov/consumption/residential/. (2009).
2. Christie A Befort, Niaman Nazir, and Michael G Perri. 2012. Prevalence of obesity among adults from rural and urban areas of the united states: Findings from NHANES (2005-2008). *The Journal of Rural Health* 28, 4 (2012), 392–397.
3. Jack Edmonds and Richard M Karp. 1972. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)* 19, 2 (1972), 248–264.
4. Michael L Fredman. 1975. On computing the length of longest increasing subsequences. *Discrete Mathematics* 11, 1 (1975), 29–35.
5. George William Hart. 1992. Nonintrusive appliance load monitoring. *Proc. IEEE* 80, 12 (1992), 1870–1891.
6. Timothy W Hnat, Erin Griffiths, Ray Dawson, and Kamin Whitehouse. 2012. Doorjamb: unobtrusive room-level tracking of people in homes using doorway sensors. In *Proceedings of Sensys 2012*.
7. Matthias Kranz and others. 2007. Sensing technologies and the player-middleware for context-awareness in kitchen environments. *Networked Sensing Sys 2007* (2007), 179–186.
8. Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly* 2, 1-2 (1955), 83–97.
9. Oscar D Lara and Miguel A Labrador. 2013. A survey on human activity recognition using wearable sensors. *Communications Surveys & Tutorials, IEEE* 15, 3 (2013), 1192–1209.
10. A Moncada-Torres and others. 2014. Activity classification based on inertial and barometric pressure sensors at different anatomical locations. *Physiological measurement* 35, 7 (2014), 1245.
11. Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. 2014. Sensing as a service model for smart cities supported by internet of things. *Transactions on Emerging Telecommunications Technologies* 25, 1 (2014), 81–93.
12. Juhi Ranjan and Kamin Whitehouse. ObjectWatcher: Using Smartwatch Data to Perform Object User Recognition. In *Proceedings of UbiComp 2015*.
13. Donald B Reid. 1979. An algorithm for tracking multiple targets. *Automatic Control, IEEE Transactions on* 24, 6 (1979), 843–854.
14. Edward S Sazonov and Stephanie Schuckers. 2010. The energetics of obesity: A review: Monitoring energy intake and energy expenditure in humans. *Engineering in Medicine and Biology Magazine, IEEE* 29, 1 (2010), 31–35.
15. Agusti Solanas and others. 2014. Smart health: a context-aware health paradigm within smart cities. *IEEE Communications Magazine* 52, 8 (2014), 74–81.
16. N Tomizawa. 1971. On some techniques useful for solution of transportation network problems. *Networks* 1, 2 (1971), 173–194.
17. Peter Widmayer and CK Wong. 1985. An optimal algorithm for the maximum alignment of terminals. *Information processing letters* 20, 2 (1985), 75–82.