Short and Long Term Optimization for micro-object conveying with air-jet modular distributed system

Hakim Mabed[a], Eugen Dedu[a]

[a] *University Bourgogne Franche-Comté FEMTO-ST Institute, CNRS, Montbéliard, France*

Abstract

Smart surface is a new conveying technology composed of a 2D planar surface presenting a matrix of distributed autonomous blocks. Every block contains a micro-electro-mechanical system (MEMS) actuator that controls the transfer of a possible object located above the block to the neighboring blocks, using air-jet forces. The spatial characteristics of the blocks impose some limits on the memory, energy and computation capabilities of the MEMS blocks. In the other hand, the system can reach several thousands of blocks making necessary to propose scalable algorithmic solutions.

This paper studies different distributed algorithms to convey an object from an initial to a target position in the smart surface. The conveying policy emphasizes the long term use of the smart surface and the objects conveying efficiency measured by the time of the transfer. The problem stands as an original case of multi-objective Shortest Path problem (MOSP). Original because the quality of a given path is not evaluated by the sum of the weights of its segments, and because the segment weights change according to the used paths as provided by the algorithm itself. Therefore, the efficiency of a given algorithm is assessed on the basis of its performance during a long period of time.

We describe here the best way to combine these two objectives and we propose a scalable incremental distributed protocol for objects conveying. The path optimality is adjusted according to the required calculation complexity. The performances of the different algorithmic and modeling variations are analyzed in terms of memory, time, computation and exchanged messages complexity. The obtained results prove the scalability of the algorithm, with linear computational, memory and convergence time complexity, and confirm the improvement of smart surface usage compared to a naive approach. The system lifespan increases of up to

130% on 40×40 smart surface, while the transfer cost (time and energy) is reduced. We show also that the computation time of the path with the incremental algorithm can be significantly reduced without significant degradation of the conveying system performance. For example, in a 40×40 smart surface, the number of messages is divided by 4 while the number of conveyed objects is only reduced by a ratio of 4%.
Keywords:

1. Introduction

A conveyor is a mechanical equipment that allows the transportation of objects within the production chain. In this study, we focus on the conveying of sub-millimeter components (1; 2). Using conveyors for micro-object is essential since human intervention is inconceivable. In (3), conveyance systems are classified according to the used transfer forces into 19 different technologies, such as chain conveyor, wheel conveyor, flat belt conveyor, magnetic belt conveyor, bucket conveyor, vibrating conveyor, pneumatic conveyor, etc. Contact-less systems such as air-jet platform present a safer way to convey fragile objects (electrical or chemical products for example).

Smart surface (4) platform, shown in Figure 1, is a robotic system composed of a grid of numerous autonomous elements (MEMS) developed for the transportation of microscopic objects over short distances. Every MEMS block involves microsensors, microactuators and a control unit. The blocks work in a distributed manner (without central unit) in order to transfer the objects to given target locations[1]. On one hand, the distributed architecture of the smart surface platform provides a more robust (fault tolerance) framework than systems based on a single monolithic block where any failure leads to the system shutdown. Indeed, the smart surface presents a modular architecture that leads to self-reconfiguration features, i.e. search for alternative paths to convey an object by avoiding failed blocks. On the other hand, contact-based technologies are not appropriate for fragile and tiny micro-objects (medicines, micro-electronics parts, etc.), which can be easily damaged, contaminated or

[1]Smart Surface project started as a research project (ANR 06 ROBO 0009) supported by French National Research Agency (ANR).

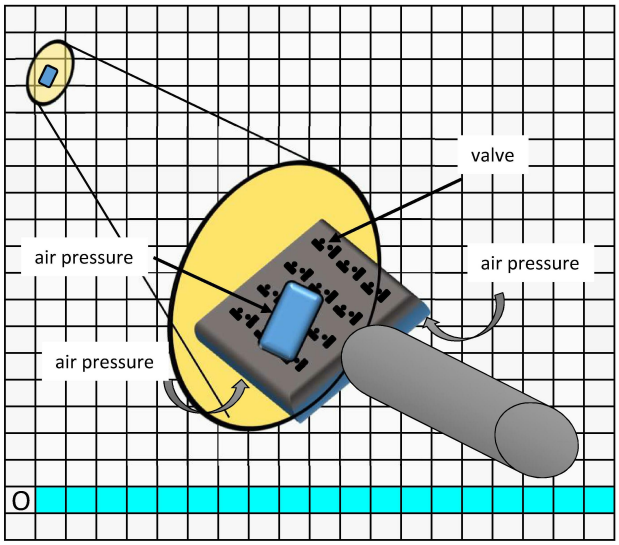

Figure 1: The smart surface architecture. The object is conveyed from its initial position S to its final position D using air pressure forces.

even scratched during conveyance. Thus, systems based on air-jet technology (5; 6; 7) such as smart surface system avoid contact with conveyed objects and provide safer manipulation.

The work presented in this paper, whose contributions will be presented in a dedicated section, is part of the smart surface project which aims to increase the efficiency of future production lines. The conveying principle consists in transporting micro-objects from a starting block to a final destination using controlled airflow coming from MEMS valves of the blocks. To do so, the degradation of all MEMS valves involved in micro-object transport has to be controlled. The reliability of MEMS components is of major concern (8). MEMS components suffer from various failure mechanisms (9; 10) that impact on their performances, their lifespan and the availability of the entire system. This highlights the need to monitor their behavior, assess their health, or, even better, estimate their remaining time before failure (RUL, Remaining Useful Life). According to health estimation, it is essential to take appropriate decisions, such as reconfiguration control and maintenance (11). These tasks can be done by using Prognostics and Health Management (PHM) approaches (12; 13; 14).

For the clarity of the presentation, the PHM aspects are not discussed. We focus on the algorithmic solution that allows thousands of blocks to cooperate in order to optimally convey tiny objects. The work is still relevant no matter the degradation process used by the equip-

ment. Consequently, the main contribution of this work is to propose an original distributed conveying algorithm that manages both the short-term performance of the conveying surface (transfer time of an object) and the long-term performance (material lifespan).

This article is organized as follows. The next section presents an overview of some analogous problems and underlines the specificity of the conveying problem in a smart surface system. Section 3 outlines the contributions of this work. Section 4 introduces the distributed MEMS-based surface. Section 5 presents the proposed algorithms and various ways to evaluate the quality of a given conveying path. Section 6 describes the simulation and the testing environment.

2. Related works

The optimization of transportation system lifetime by the adoption of a smart routing strategy is a common problem in intelligent transportation systems. In multi-hop wireless networks (15; 16; 17), for instance, naive data routing protocols lead to the overuse of some nodes which can rapidly be discharged. Packets routing policy needs to spread the used paths over the network nodes. However, the problem discussed here presents a major novelty. The links features (latency, throughput, capacity, etc.) are generally considered static in telecommunication networks. Smart surface conveying problem assumes that the transfer time of a given block to its neighboring blocks varies over time according to the block use.

In Road Network (18), the traffic optimization aims to avoid congestion. As for the smart-surface conveying problem, external events such as accidents and thunderstorms (in the road network) or clogging and dust risks (in the smart surface) affect the system. Furthermore, human behavior increases the uncertainty in road network unlike the smart surface system where the human intervention is nonexistent. However, in the road network the infrastructure lifespan is not immediately impacted by the traffic while the lifespan of the smart surface is partially determined by the use of blocks.

To conclude, despite some similarities with well-known problems in transportation and communication fields, our problem remains a particular case due to the millimeter scale of the system, inducing hardware and software constraints (memory storage, computing capab-

ilities) and to the addressing of the degradation state of the MEMS modules. Degradation effects make that a path to convey an object could be good at a given moment and become bad later.

Regarding the authors' previous research, an initial study can be found in (19). The work presented here presents a more realistic modeling and a significant extension of the precedent study. The first major original addition of the current article is the introduction of the effect of orientation changes on the object transfer time. We also propose an incremental distributed algorithm that reduces the amount of computational effort and exchanged messages between blocks. The study is enriched with a new optimization criterion (squares sum) that leads to a better exploitation of the smart surface. In addition, a short state of the art on the shortest path problems is given. Also, a related study can be found in (20), which uses a non-scalable algorithm and uses only one function for path quality assessment, both of them being the main points addressed in this article.

3. Contributions

The distributed architecture of the platform and the conveying algorithm aim to guarantee, among others, the modularity and maintainability of the conveying system. The modularity means that no block is compulsory for the system to work. The maintainability concerns the ability of the system to fix some problems by replacing only a subset of the system (some blocks), leading to smaller maintenance costs.

The smart surface conveying problem is an original Multi-Objective Shortest Path (MOSP) problem. The work presented here focuses on a sequential conveying system where a new object is introduced once the previous object is removed from the surface. This assumption allows us to ignore, at this stage of the work, physical collisions between objects. The paper presents an original study about how to manage the trade-off between the short-term performance of the surface (transfer time of an object) and the long-term performance (surface lifespan).

In this context, the distributed optimization of the conveying paths is a complex task. The computational and memory limits of each block make necessary to provide algorithmic solutions with low and constant spatial complexity (memory storage requirement). Proposed

procedures guarantee a computational complexity of less than $O(m)$ on every block, where m is the number of blocks. We are showing that the algorithm convergence is reached with a time complexity of $O(m)$.

4. The distributed MEMS-based conveyor

As already shown in Fig. 1, the conveying surface is described by a 2D grid of decentralized micro-blocks. Each block presents a square surface with a sensor that can detect when an object is above the block, an actuator formed by MEMS valves, a micro-controller executing code, a power supplier, and a network module allowing communication with its four neighboring blocks. The valves allow air to flow and push the object above the block to one of the neighboring blocks. When an object reaches its destination, it gets out of the surface, and a new object enters the system.

There are a lot of challenging issues to be tackled from a technological perspective, such as the fabrication of these micro-devices (sensor, actuator, controller, communication module) and their integration inside the block, and controlling the object above the surface (air levitation is always unstable, the direction depends on object shape and size, so the object can be correctly conveyed only through the cooperation of several blocks). Solving the whole problem requires solving the different parts of it. In the current article we abstract down the problem to a planar conveying problem, presented in the following, and focus on it.

Let's assume a smart surface of m blocks. We notice by $C(b)$, initially 0, the number of objects that crossed the block b. A block is then associated with two dynamic values that describe its state: the RUL and the transfer time.

4.1. Block characteristics

The RUL of a block designates the number of times that the block can again be used before the start of revision procedure. When the RUL of a block reaches 0, the smart surface is stopped and overused blocks are fixed or replaced. We consider that the initial RUL, corresponding to the maximum number of uses of a block, is specified by the manufacturer as part of the device's technical characteristics.

Definition 1. *$RUL(b)$ is the remaining number of times that the block b can be used before the maintenance.*

A block, b, is also associated to $T(b)$ value that refers to the time needed to move an object from the block to an adjacent one. $T(b)$ depends on the state of the valve, i.e. the more the valve is used, the more the valve becomes inefficient. The transfer time is subject to variations over time due to a variety of causes. To forecast the next transfer time on a given block, the degradation model may include criteria such as transfer time of the last object(s), number of uses, frequency of uses and last use date. Section 6 describes the two degradation models we used in tests to simulate the variation of the transfer time of blocks. The transfer time on a given block depends also on the need of object re-orientations, i.e. the precedent and the next blocks are not aligned. Every time the object changes its motion orientation, the smart system stops its progression then pushes it toward the new direction. The re-orientation of an object requires additional time corresponding to the stop-restart procedure.

In the absence of PHM mechanisms, each block estimates its own transfer time as the experienced transfer time during the last use of the block. Consequently, the effective transfer time of the object following the optimal found path could be different from the estimated value (see section 6). After every use of the block b, the transfer time value $T(b)$ is updated. The initial value of transfer time corresponds to the manufacturing value specified by the constructor.

Definition 2. *$T(b)$ is the recorded transfer time during the last use of the block b.*

4.2. Path characteristics

The distributed protocol aims to compute in real time, the best path from the initial position of an object to its final position. When an object is detected and the final position known, the blocks determine the best path that increases the surface lifetime and reduce the transfer time. The transfer time of a given path is calculated as follow:

$$T(p = [b_1..b_n]) = \sum_{b_i \in p} T(b) + To(b_{i-1}, b_i, b_{i+1}) \quad (1)$$

Where $To(b_{i-1}, b_i, b_{i+1})$ refers to the re-orientation time needed to push an object coming from $b_{i-1} \in p$ toward $b_{i+1} \in p$ when b_{i-1}, b_i and b_{i+1} are not aligned.

$$To(b_{i-1}, b_i, b_{i+1}) = \left\{ \begin{array}{ll} 0 & \text{if } b_{i-1}, b_i, b_{i+1} \text{ are aligned} \\ 1 & \text{otherwise} \end{array} \right\} \tag{2}$$

However, whereas the evaluation of a path transfer time is simply computed by the sum of the transfer times of the traversed blocks (eq. 1), the evaluation of the system lifespan on the basis of the $RUL(b)$ values is a complex question. In this paper we study different measurement functions of the RUL of a path. Each block tries to optimize the RUL of the remaining path before reaching the final position. The RUL of a path p starting from the block b_{1st} is computed using the recursive function $RUL(p)$ described in equation 3, where F is a function to be defined:

$$RUL(p) = F(RUL(p - \{b_{1st}\}), RUL(b_{1st})) \tag{3}$$

F is a monotonic function that should be defined in such a manner that it respects the following condition:

$$\left\{ \begin{array}{c} \forall p, \forall b, RUL(p) \leq F(RUL(p), RUL(b)) \\ \text{or} \\ \forall p, \forall b, RUL(p) \geq F(RUL(p), RUL(b)) \end{array} \right\} \tag{4}$$

The monotonicity feature prevents inconsistent functions F where adding loops in a path may *improve* its evaluation.

5. Multi-criteria distributed conveying algorithm

This section presents a brief state of the art on shortest path problems, and afterward the proposed algorithm called incremental massively distributed algorithm. The incremental massively distributed algorithm presents an improvement of the massively distributed algorithm published in (19).

5.1. State of the art on shortest path problems

The shortest path problem is one of the most fundamental network optimization problems, used in various fields such as telecommunication (21), transportation (22), production systems (23). In the graph theory, this problem consists in finding a path between two nodes in a graph by minimizing the total weight of the path. Weights refer to any metric that describes the effort to traverse the path, such as length, time, cost, etc. The literature divides the shortest path problems in three categories:

- Single-source: find a shortest path from a given source node to each of the nodes.
- Single-pair: given two nodes, find a shortest path between them. This is a particular case of single source category.
- All-pairs: find the shortest path for each pair of nodes.

The best known algorithms for solving this kind of problems are: Dijkstra's ((24)), Bellman-Ford ((25)), A* ((26)), Floyd-Warshall ((27)) and dynamic programming algorithms (28), (29), (30, page 185).. The characteristics of these algorithms are given in table 1. Overall, it has to be noted that the algorithms based on computing the best path from multiple sources to multiple destinations (all pairs) are not relevant for our problem: path quality dynamically changes during object conveyance, and we just need the optimal path from the current position of the object to its destination.

5.2. Multi-objective shortest path problems

Real world problems have often a multi-objective nature (31; 32). In shortest path problems, it is not unusual that the paths be evaluated according to several criteria at the same time such as the distance, the cost, the time, the number of hops, etc.

In major works on multi-objective shortest path optimization, the problem is modeled by a graph where each node or edge is labeled with a vector of weights, one per objective (33; 34). The quality of a path according to a given objective is then measured as the sum of the weights of the nodes or edges belonging to the path. Object conveying problem with the smart surface is an original problem. Indeed, the impact on the smart surface lifespan of conveying an object can not be simply evaluated as the sum of the wear (RUL) of the

Table 1: Examples of shortest path algorithms (V is the number of nodes and A is the number of edges in the graph).

Algorithm	Characteristics	Speed
Dijkstra	- single source	- $O(V^2)$
	- positive weight	- fast
	- optimal	
Bellman-Ford	- single source	- $O(\|V\|.\|A\|)$
	- negative weight	- slower than
	- negative cycles	Dijkstra
	- optimal	
Floyd-Warshall	- all pairs	- $O(\|V\|^3)$
	- negative weight	- slower than
	- negative cycles	Dijkstra
	- optimal	
A*	- single pair	- depends on the
	- heuristic, optimality	heuristic
	not guaranteed	- fast

crossed blocks. More subtle formulation of the path impact on the smart surface RUL is to be studied, as it will be shown in section 5.5.

5.3. Standard Dijkstra's algorithm

We decided to reuse the well-known Dijkstra's algorithm by reason of the similarities between the studied problem and the shortest path problem. According to the literature, Dijkstra's algorithm has the smallest complexity among all the algorithms finding the optimal solution. However, Dijkstra's algorithm was conceived for mono-criterion problems. Hence it needs to be adapted to fit our multi-criteria problem.

A first version of the algorithm was presented and analyzed in (20) and used in (35) to discover how the optimal path evolves in time during conveyance. The modified Dijkstra's algorithm, presented in Algorithm 1, returns the path with the higher minimum RUL over the crossed blocks (MaxMin RUL). If several paths have the same MaxMin RUL, the path with minimum transfer time is then returned. To obtain this algorithm, authors started with an original Dijkstra's algorithm that minimizes the transfer time and then modified the initialization part and the relaxation part (lines 4, 8, 16 and 18 have been added, and line 17 has been modified) to maximize the RUL. The obtained algorithm maintains the same complexity as the original Dijkstra's algorithm, and is still optimal.

In this first algorithm, each block computes the complete path toward the target position of the object, and stores it in $next[]$ vector. In addition, each block stores the current estimated lifetime, $RUL(b)$, and transfer time, $T(b)$, of all the blocks. Each time the best path from a neighbor is computed, it is compared with the path provided by the other neighbors (line 17). The "better than" relation between two paths is determining for the optimization strategy and will be discussed in section 6 (simulation results). In other words, the "better than" relation expresses the selection procedure among two possible paths evaluated according to the same criterion.

Definition 3. *Let s_1 and s_2 be two paths with the same source and destination block and let $(f_1, ..., f_n)$ be n evaluation criteria. s_1 is said dominating s_2 if and only if:*

- $\forall f_i,\ f_i(s_1)$ *is better than or equal to* $f_i(s_2)$

Algorithm 1 Modified Dijkstra's algorithm.

Require: RUL[], T[], dest

Ensure: best path toward dest: next[]

```
1: for each block x in the surface do
2:    bestT[x] = ∞
3:    bestRUL[x] = 0
4:    next[x] = undefined
5: end for
6: bestT[dest] = T[dest]
7: bestRUL[dest] = F(RUL[dest])
8: Q = initially contains the source node
9: while Q is not empty do
10:   y = node in Q with best RUL in bestRUL[]
11:   remove y from Q
12:   for each neighbor x of y do
13:      if x ≠ dest then
14:         timeThroughY = bestT[y] + T[x]
15:         RULThroughY = F(bestRUL[y], RUL[x])
16:         if (RULThroughY,timeThroughY) is better than (bestRUL[x],bestT[x]) then
17:            bestRUL[x] = RULThroughY
18:            bestT[x] = timeThroughY
19:            next[x] = y
20:            add x to Q
21:         end if
22:      end if
23:   end for
24: end while
```

- *and* $\exists f_j \mid f_j(s_1)$ *is better than* $f_j(s_2)$.

This first algorithm has some drawbacks:

- Memory: The memory complexity of the algorithm is $O(3 \times m)$, corresponding to the storage of RUL, T and *next* data. The memory limitation of the micro-modules makes this algorithm non scalable when the number of blocks increases.

- Time: Every block contained in the best path will initiate the procedure only when the object is present above it. Therefore, the transfer time includes also the computation of the best path. Note, however, that, with the trade-off of increasing network usage, this point could be ignored if every block sends the entire optimal path to the next block before or during moving the object.

- Network: The storage by each block of the RUL and T data of every block means that the update procedure used to broadcast the changes to each block of the smart surface needs $|p*| \times (m-1)$ exchanged messages, where $|p*|$ represents the length of the best path found. In that case, the algorithm could not be started until the update procedure has finished. In case of numerous blocks, this can slowdown the conveying procedure.

5.4. Massively distributed algorithm

In (19), we proposed a synchronous algorithm which avoids the drawbacks above. Each block stores three scalar values corresponding to its RUL, T and *next*. Where *next* refers to the next block to convey the object until the *dest* position. Therefore, the memory complexity of the algorithm is reduced to a constant value $O(1)$, making the algorithm scalable.

While in Algorithm 1, blocks proceed sequentially, in massively distributed algorithm, optimal sub-paths are computed by the blocks in distributed way. Every time the block receives an $OPTIMIZE$ message, it updates its optimal path by may be changing its *next* block and informs its neighbors by this change.

To ensure that no better path will be found by its neighbors (proof can be found in (19)), each block runs the algorithm 2 for *nbmaxround* rounds, where:

Algorithm 2 Massively distributed Dijkstra-based algorithm.

Require: myRUL, myT, dest

Ensure: best path from me toward dest: next

```
1: bestRUL = worstValue
2: bestT = ∞
3: if self == dest then
4:    send message OPTIMIZE(self, myRUL, myT) to all neighbors
5: end if
6: for each round do
7:    if self receives a message OPTIMIZE(v, pathRUL, pathT) then
8:       if (F(myRUL,pathRUL),myT+pathT) is better than (bestRUL,bestT) then
9:          bestRUL = F(myRUL,pathRUL)
10:         bestT = myT+pathT
11:         next = v
12:         if self is not the source of the object then
13:            send message OPTIMIZE(self, bestRUL, bestT) to all neighbors except v
14:         end if
15:      end if
16:   end if
17: end for
```

$$nbmaxround = \frac{m}{2} + \max\left(\frac{x}{2} + \tilde{y}, \frac{y}{2} + \tilde{x}\right) \quad (5)$$

x, y are the number of columns and lines of the surface ($x \times y = m$) respectively, and $\tilde{N}$ is equal to N if N is odd and 0 otherwise ($\tilde{N} = N \times N\%2$).

5.4.1. Incremental massively distributed algorithm

The objective of the incremental version of the massively distributed algorithm is to reduce the amount of computation, energy consumption and the number of exchanged messages needed to determine the best path. To that end, we take advantage of the results of the previous computations.

Let's take the following example: at instant t, an object located at the block $S1$ needs to be conveyed to the destination block D. After running the massively distributed algorithm, the variable *next* of the block $S2 \neq S1$ designates the best next hop to reach D from $S2$. Now, another object needs to be moved towards D starting from $S2$. If no block within the path between $S2$ and D was crossed since t, then the path P remains valid and does not need to be computed again. This idea is explored in Algorithm 3, where each block stores a vector $next[]$ of m entries and $next[b]$ refers to the best next block to reach the block b (or $NULL$ if the path to the block b is not yet known).

When an object crosses a block b provoking changes of its RUL and T values, all the paths involving the block b are erased by setting to $NULL$ the values $next[]$, as depicted in Algorithm 4. Note that this algorithm is run only in incremental case, after Algorithm 3. More precisely, $\forall n \in Neighbors(b)$ and $\forall b$ with $next_n[b] = b$, $next_n[b]$ is set to $NULL$, where $next_n[]$ refers to the vector *next* stored by the node n. Recursively, $\forall$ modified $next_n[b]$ and $\forall n' \in Neighbors(n)$ with $next_{n'}[b] = n$, $next_{n'}[b]$ is set to $NULL$, and so on. A flag vector $isModified$ is used to report paths which need to be recomputed.

Additionally, we consider a heuristic variant of the algorithm. To further reduce the number of exchanged messages, we propose that blocks do not systematically recalculate the best path to the *dest* block. This relies on the fact that during the conveyance of an object, the variation of the optimal path is often marginal. Therefore, it is counterproductive for a block to re-compute the new path each and every time the current stored path is modified

Algorithm 3 Incremental distributed algorithm.

Require: myRUL, myT, dest, next[]

1: for all d, bestRUL = WorstValue
2: for all d, bestT = ∞
3: for all d, next[d] = $NULL$
4: for all d, isModified[d] = $false$
5: **if self** == dest **then**
6: send message OPTIMIZE(self, myRUL, myT) to all neighbors
7: **end if**
8: **for** each round **do**
9: **if** isModified[dest] AND rand()< *prob* AND self receives a message OPTIMIZE(v, pathRUL, pathT) **then**
10: **if** (F(myRUL,pathRUL),myT+pathT) is better than (bestRUL,bestT) **then**
11: bestRUL = F(myRUL,pathRUL)
12: bestT = T(self)+pathT
13: next[dest] = v
14: **if** self is not the source of the object **then**
15: send message OPTIMIZE(self, bestRUL, bestT) to all neighbors except v
16: **end if**
17: **end if**
18: **end if**
19: **end for**
20: isModified[dest] = false

Algorithm 4 Procedure executed during object conveyance.

Require: myRUL, myT, dest, next[]

```
1: for each round do
2:   if the object is on self then
3:     push the object to next[destination] {move the object to the next position}
4:     send message UPDATE(self, NULL) to all neighbors
5:   else if self receives message UPDATE(v, dest) then
6:     if dest == NULL then
7:       for each destination d do
8:         if next[d] == v then
9:           isModified[d] = true
10:          send message UPDATE(self, d) to all neighbors except v
11:        end if
12:      end for
13:    else
14:      if next[dest] == v then
15:        isModified[dest] = true
16:        send message UPDATE(self, dest) to all neighbors except v
17:      end if
18:    end if
19:  end if
20: end for
```

($next[]$) and to inform neighboring blocks of this change. A re-computing probability $prob$ is used (see algorithm 3, line 9) to trigger the recalculation of the variable $next[dest]$. As such, this variant leads to sub-optimal paths, but reduces the number of exchanged messages.

Compared to the initial massively distributed algorithm, the incremental version increases the memory complexity of the algorithm from $O(1)$ to $O(2 \times m)$. Therefore, the improvement of the computational complexity of the incremental version should be sufficiently high to justify the lack of memory scalability. The termination condition is identical, with the same time complexity. Furthermore, the number of OPTIMIZE messages cannot exceed $3 \times MaxRounds$ in both algorithms, and in the heuristic variant it is even smaller. The efficiency of the incremental mechanism will then be assessed according to the real number of exchanged UPDATE messages that determines the needed computational effort.

5.5. Optimization criteria

The aforementioned algorithms use an objective function F to compute the quality of a given path. Not only it defines what the best path is, but also the desired trade-off between the short-term (fast object conveyance) and the long-term vision (high system lifespan). Consequently, the choice of this function is a key factor for the efficiency of the proposed solution. In this section, we present various ways to combine the two criteria (RUL and T) within a single function F. The global lifetime of the system relies on the lifetime of each block, and especially the impact of a given conveying path on the system durability. In a precedent work (19) we performed a detailed study on the optimization criteria modeling. We retake here only the two following:

MaxMin RUL function The RUL of a path, to maximize, is the smallest RUL among the RUL of path's blocks:

$$\max_{p} RUL(p) = \max_{p} \min(RUL(p - \{b_{1st}\}), RUL(b_{1st})) \quad (6)$$

$$= \max_{p} \min_{b \in p} RUL(b) KaisaMiettinen \quad (7)$$

If two paths have the same RUL, the path with the better transfer time is considered.

Squares function The RUL of a path, to minimize, is the sum of the squares of the

uses number $C(b)$ over the path's blocks:

$$\min_p RUL(p) = \min_p \sum (RUL(p - \{b_{1st}\}), C(b_{1st})^2) \quad (8)$$

$$= \min_p \sum_{b \in p} C(b)^2 \quad (9)$$

If two paths have the same RUL, the path with the better transfer time is considered.

6. Experiments

This section has for objective to prove the relevance of the conveying distributed algorithm in smart surface. We will describe in turn the conveying scenarios and smart surface platform simulator. Then we assess the relevance of the different optimization strategies (optimization criteria) and we study the performance of the incremental distributed algorithm.

6.1. Scenarios

To propose representative set of test scenarios, we conceived 10 scenarios with different smart surface sizes, different rules for generating initial and final positions of the objects and different degradation processes. Simulated smart surface sizes vary from 10x10 to 40x40 (1600 blocks). The initial and final positions of conveyed objects are generated following two different rules. In the "any" rule, the initial and final positions are selected randomly and could be any block in the smart surface. While in "edge" rule, the initial and final positions are located on one of the four sides (first or last line or column) of the smart surface. Each rule corresponds to a class of conveying problems. The "any" rule is suitable for manipulation and placement applications, when "edge" rule fits filtering or classification applications.

At this stage of the work, a real PHM model is not used. So the blocks Remaining Useful Life (RUL) is estimated according to the number of block uses. We assume that blocks are designed for a maximum number of uses, $C_{\max}$. The block RUL can be deduced from the number of uses of the block using equation:

$$RUL(b) = C_{\max} - C(b) \quad (10)$$

Finally, to enlarge the scenarios diversity, we studied two different degradation models. The degradation models describe how the performances (transfer time) of the blocks are

impacted by the successive valves activation. The first degradation model represents a realistic degradation model (36; 20) called exponential model. Every time the block is used, its transfer time increases by a probabilistic value following an exponential distribution with mean α.

$$T(b) = T(b) - \alpha \log(RND) \tag{11}$$

where RND is a randomly selected value in the interval [0,1].

In the second degradation model, called idle-based model, we introduce the period during which a block was not used as a factor of the degradation. The objective is to push the system to regularly select every block in order to conserve the valves health state. This way, the transfer time of a given block varies according to the following expression:

$$T(b) = T(b) - \alpha \log(RND) + \beta \times U(b) \tag{12}$$

$U(b)$ represents the number of cycles during which the block b has not been used.

It should be recalled that the degradation model is not a prediction model used by blocks to forecast their future transfer time. Indeed, blocks refer to their last transfer time (during an object conveying) as the current $T(b)$. The degradation model is used to simulate the gap between the predicted path performance (during the distributed algorithm phase) and the real performance (during the conveying simulation). The simulator functioning scheme is depicted in figure 2.

6.2. Simulator

We implemented a simulator in VBA Microsoft Excel. The simulator executes a given scenario until a maintenance procedure is required as a result of a block end of life ($\exists b, RUL(b) = 0$). At each iteration of the simulator (see figure 2), a new object is generated with its initial and final positions. According to the last experienced T(b) and the current RUL(b), a path is computed to convey the object from its initial to the final position. Then the simulation of the conveying process is started. The object crosses over the blocks of the computed path according to the transfer times simulated using the degradation model (equations 11 or 12).

For all the tests presented hereafter, we used the following parameters: $C_{\max} = 100$, $T_0 = 1$s, $\alpha = 10$s, $\beta = 0.1$s, $To = 0.3$ s. We assume that the smart surface is unused at the beginning of the simulation, i.e. $\forall b, C(b) = 0$.

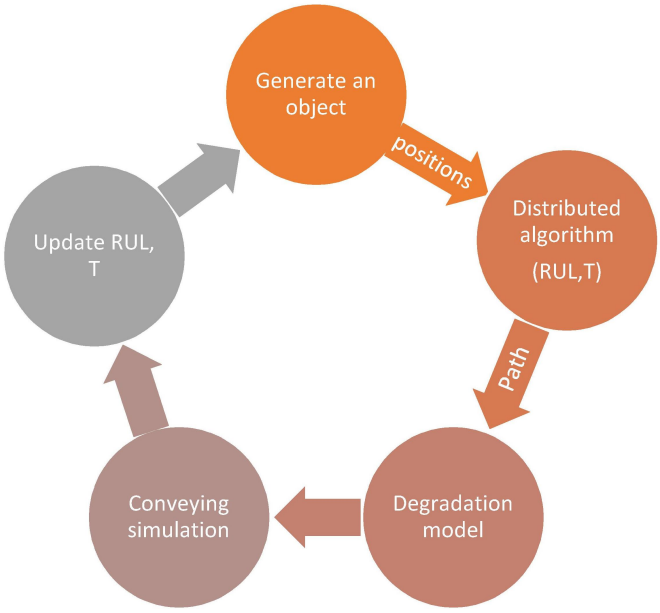

Figure 2: Simulator architecture. At each iteration, the conveying of a new object is simulated (initial and final positions), and the algorithm is run to compute the path that should be followed by the object.

6.3. Optimization criteria comparison

The behavior of each of the two studied criteria (MaxMin RUL and Square Sum objective) is depicted in figure 3, where the the color of each block designates its wear level at the end of the simulation (when the RUL of a given block reach 0). We also show, in figure 3 the wear state of the smart surface at the end of the simulation using two other criteria: the minimum transfer time path and the straight line path. As we can see, the use of the sum of RUL square allows a better exploitation of the smart surface with a good partition of the block usage.

Figures 4, 5 and 6 show the comparison results between the squares sum based optimization and the MaxMin RUL based optimization. The comparison is made, respectively, on the basis of the number of conveyed objects, the average transfer time of conveyed objects and the average transfer time of the first 200 conveyed objects. The latter is important since a surface which conveys more objects will clearly be more degraded at the end than a surface which only conveys a few objects. Also, the average on the first 200 objects allows to assess the conveyance rapidity during the starting period of the system.

Figures 4, 5 and 6 show that the function evaluating the entire path's blocks such as

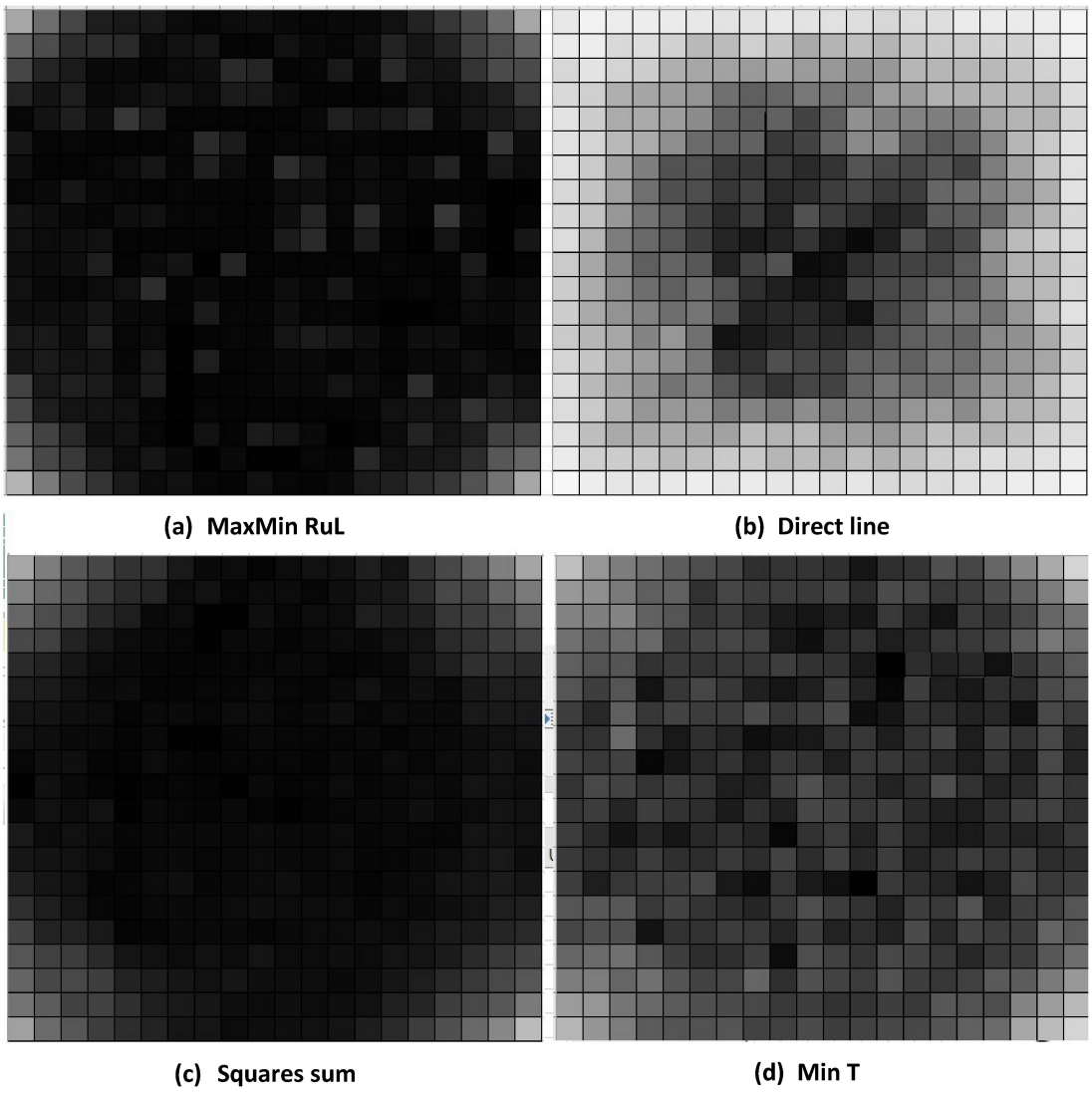

Figure 3: Wear level of a smart surface with 20x20 blocks at the end of 4 simulations with the different optimization criteria: square sum, MinMax RUL, Minimum Transfer time and straight line move. Darker is the block lower is the RUL.

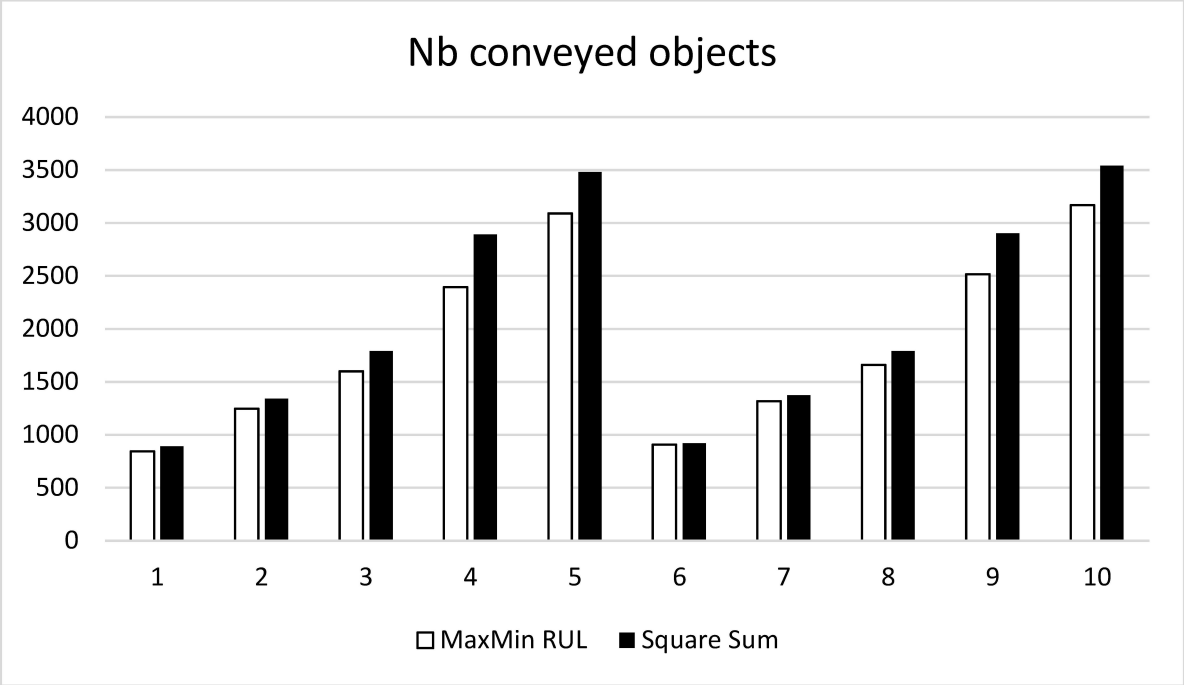

Figure 4: Comparison of the number of conveyed objects on the 10 studied scenarios (x-axis) according to the optimization criteria.

square function performs better than the function that relates only a part of the path, such as MaxMin RUL function. Indeed, trying at any cost to avoid overused blocks leads to more crossed blocks with only marginally less degradation (see the example of Fig. 4.a). Furthermore, the maximum block use does not capture the values of the other blocks in the path. Thus, two paths with equal minimum block RUL can be completely different when considering all the blocks of the path (see fig. 4.b).

We also observe (Figure 5) that Squares sum based optimization provides a better average transfer time even though the average value is computed for a longer period; i.e. more conveyed objects. The transfer time efficiency of Squares sum based optimization is better observed when the comparison is restricted to the 200 first conveyed objects (Figure 6).

6.4. Incremental algorithm performance

Let's recall that the heuristic incremental algorithm reduces the computation time and the number of exchanged messages to the detriment of memory and particularly the accuracy of the results. Computation time and memory have already been presented. Thus, this section

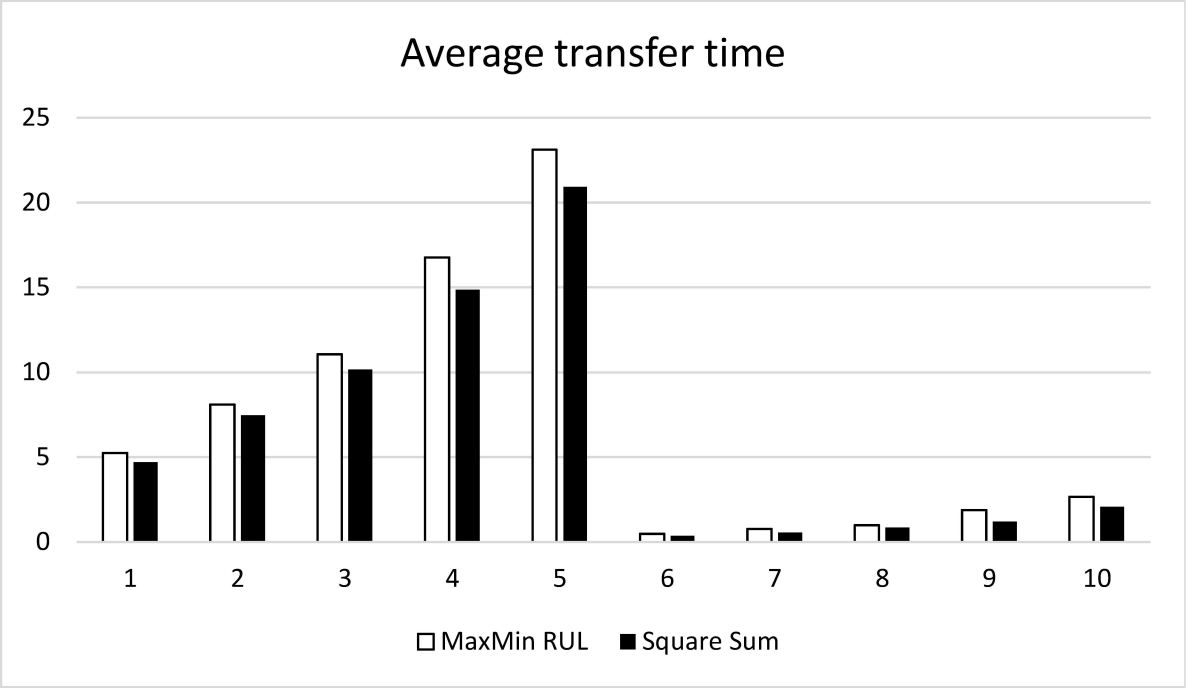

Figure 5: Comparison of the transfer time of conveyed objects on the 10 studied scenarios (x-axis) according to the optimization criteria.

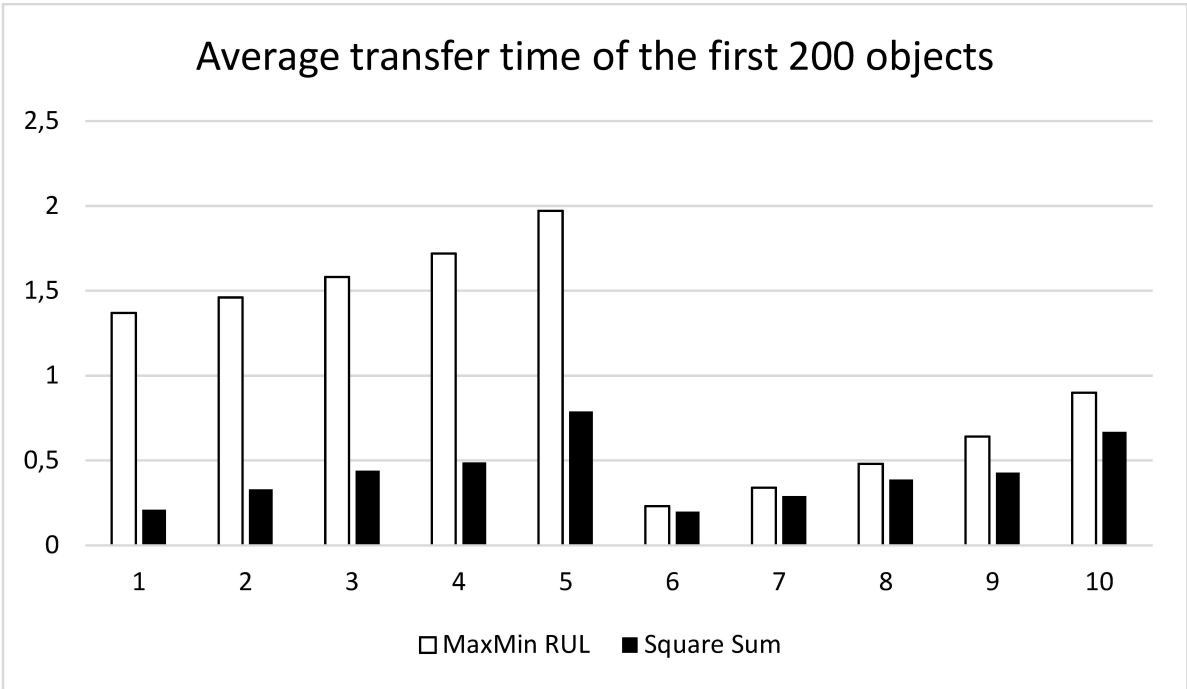

Figure 6: Comparison of the transfer time of the 200 first conveyed objects on the 10 studied scenarios (x-axis) according to the optimization criteria.

considers the other criteria.

Figures 7 and 8 compare incremental and non-incremental algorithms in terms of the number of exchanged messages and the number of conveyed objects. We observe that the variation of the *prob* parameter allows to adjust the desired trade-off between the optimality of the used paths and the number of messages needed to guarantee this optimality. On the one hand, the smaller the value of *prob* parameter, the smaller the number of conveyed objects (due to lower quality paths being used). On the other hand, smaller *prob* values produce fewer exchanged messages and thus faster convergence and less energy consumption. We also observe that the system lifespan (number of conveyed objects) is not significantly degraded when the optimal paths are not systematically updated. For the scenario No 6, compared to the non-incremental algorithm, updating the optimal path only every 20% of time reduces the number of conveyed objects by less than 0.8% (903–895=8 fewer objects) while the number of exchanged messages is reduced by 10%. In practice, it is up to the decision maker to specify the balance between the algorithm lightness and solution optimality considering the effective capacity of blocks and surface dimensions.

Finally, we observe that for relatively big surfaces, the reduction of exchanged messages is less significant. For example, in 20:edge:lin scenario (in bold in the table), the number of messages goes from 16M (non-incremental method) to 10M (using prob=5%). The reason is that the average conveying path length increases with the surface size. The probability that an optimal path needs to be updated depends on the parameter *prob* and the number of hops to the destination block as given by the following formula:

$$\sum_{i=0}^{t} (1 - prob)^i \times prob \tag{13}$$

where t is the length of the stored optimal path between a source and a destination block. We conclude that *prob* parameter needs to be adjusted according to the surface size: the bigger the surface, the smaller the *prob* parameter.

7. Conclusion and perspectives

In this paper, we have studied the optimization of objects conveyance with a modular distributed MEMS surface. We proposed a bi-criteria optimization model that takes into

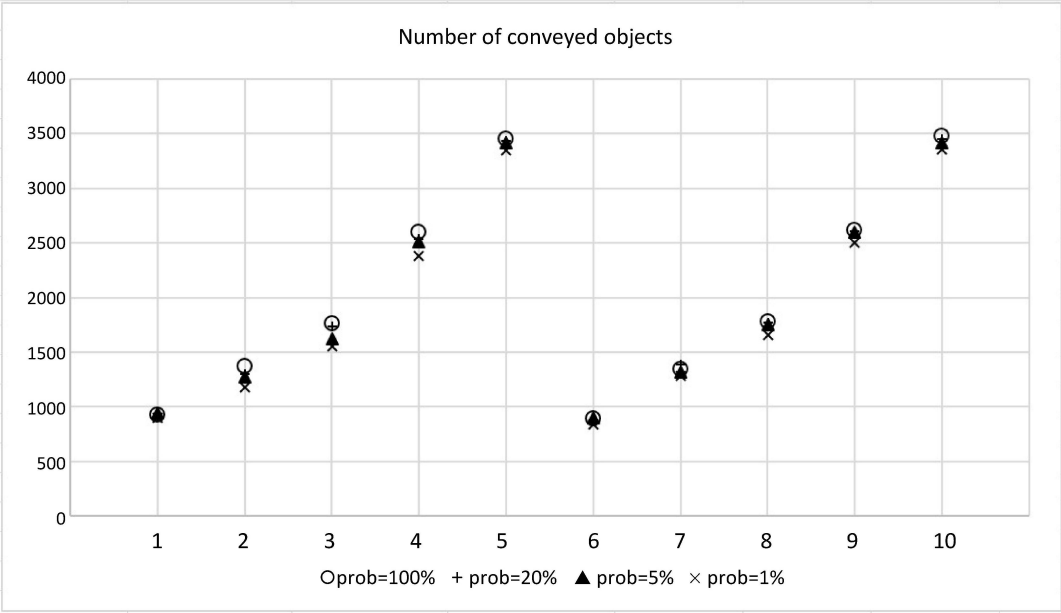

Figure 7: Comparison of the number of conveyed objects on the 10 studied scenarios (x-axis) according to the incremental probability parameter *prob*.

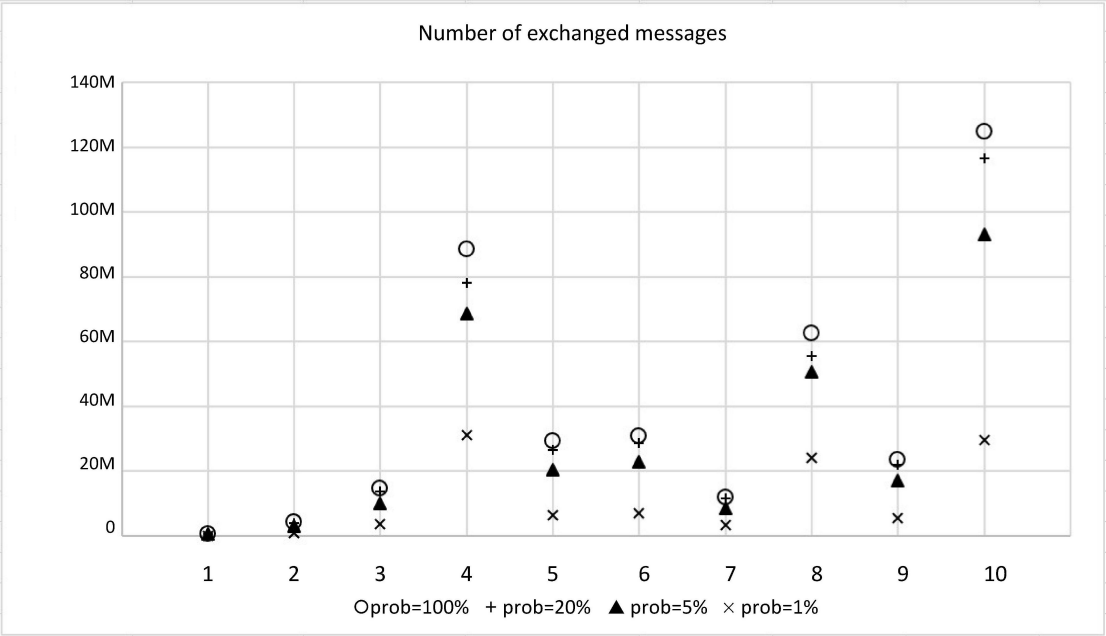

Figure 8: Comparison of the number of exchanged messages on the 10 studied scenarios (x-axis) according to the incremental probability parameter *prob*.

account a short-term objective consisting to convey the current object as soon as possible to its final position, and a long-term objective related to the extension of the system lifespan. We studied different ways to combine these two criteria within a single objective function used by the optimization algorithm. Our results show that methods based on the transformation of the RUL value to distinguish overused blocks from under-used blocks (such as square sum function) provide the best results.

We also implemented an incremental distributed algorithm that allows every block to reuse optimal paths previously found which leads to reduce the number of exchanged messages. The algorithm presents a linear complexity, guaranteeing the scalability of the system with respect to the number of blocks.

The results obtained are encouraging since the lifespan of the system is extended with the use of the square sum criterion by up to 160% compared to naive direct line approach without degrading the conveying speed. Approaches that make the short-term objective a priority, such as direct line or transfer time minimization, lead to an unbalanced (heterogeneous) use of the surface blocks (see figure 3). Later this unbalance provokes the degradation of the conveying speed as well. This is due to the relationship between the blocks RUL and the degradation of the transfer time. Hence, we have analyzed two different degradation models defining this relationship called exponential and idle based degradation model. In both cases, the square sum objective performs better than the other criteria according to the number of conveyed objects and the average transfer time of an object.

The incremental version of the massively distributed algorithm allows adjusting the number of exchanged messages according to the desired degree of optimality. For instance, on the scenario No 10, the use of an updating probability of 1% divides the number of exchanged messages by 4, while the number of conveyed objects has decreased by only 4%.

The main limitation of this work is the sequential introduction of the objects on the smart surface. It is then necessary to adapt the algorithmic solutions in order to take into account the conveying of several objects at the same time. Also, for more accurate modeling of the problem, it will be interesting to associate 4 different transfer times to each block according to the next neighboring block. Finally, we need to consider that a single block is not sufficient to control an object, hence several neighboring blocks need to collaborate to

move the objects.

References

[1] P. Helin, M. Calin, V. Sadaune, N. Chaillet, C. Druon, A. Bourjault, Micro-conveying station for assembly of micro-components, in: Intelligent Robots and Systems. IROS, Proceedings of the IEEE/RSJ International Conference on, Vol. 3, 1997, pp. 1306–1311.

[2] M. Edo, Y. Watanabe, O. Morita, H. Nakazawa, E. Yonezawa, Two-dimensional micro conveyer with integrated electrostatic actuators, in: Micro Electro Mechanical Systems, 1999. MEMS '99. Twelfth IEEE International Conference on, 1999, pp. 43–48.

[3] M. G. Kay, Material handling equipment (2012).
URL `http://www.ise.ncsu.edu/kay/Material_Handling_Equipment.pdf`

[4] K. Boutoustous, G. J. Laurent, E. Dedu, L. Matignon, J. Bourgeois, N. L. Fort-Piat, Distributed control architecture for smart surfaces, in: Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, 2010, pp. 2018–2024.

[5] S. Konishi, H. Fujita, A conveyance system using air flow based on the concept of distributed micro motion systems, Microelectromechanical Systems, Journal of 3 (2) (1994) 54–58.

[6] B. Dahroug, G. J. Laurent, V. Guelpa, L. Fort-Piat, et al., Design, modeling and control of a modular contactless wafer handling system, in: Robotics and Automation (ICRA), IEEE International Conference on, 2015, pp. 976–981.

[7] Y. Fukuta, Y.-A. Chapuis, Y. Mita, H. Fujita, Design, fabrication, and control of mems-based actuator arrays for air-flow distributed micromanipulation, Microelectromechanical Systems, Journal of 15 (4) (2006) 912–926.

[8] K. Medjaher, H. Skima, N. Zerhouni, Condition assessment and fault prognostics of microelectromechanical systems, Microelectronics Reliability 54 (1) (2014) 143–151.

[9] D. M. Tanner, MEMS reliability: Where are we now?, Microelectronics reliability 49 (9) (2009) 937–940.

[10] H. R. Shea, Reliability of MEMS for space applications, in: Reliability, Packaging, Testing, and Characterization of MEMS/MOEMS, Vol. 61110, Sans Jose, CA, USA, 2006, pp. 1–10.

[11] H. Skima, K. Medjaher, C. Varnier, E. Dedu, J. Bourgeois, Hybrid prognostic approach for Micro-Electro-Mechanical Systems, in: IEEE Aerospace Conference, 36, Big Sky, Montana, USA, 2015, pp. 1–8.

[12] L. Jay, W. Fangji, Z. Wenyu, G. Masoud, L. Linxia, S. David, Prognostics and health management design for rotary machinery systems reviews, methodology and applications, Mechanical Systems and Signal Processing 42 (1) (2014) 314–334.

[13] K. Javed, A robust and reliable data-driven prognostics approach based on extreme learning machine and fuzzy clustering., Ph.D. thesis, University of Franche-Comté, Besançon, France (2014).

[14] K. Medjaher, N. Zerhouni, Hybrid prognostic method applied to mechatronic systems, The International Journal of Advanced Manufacturing Technology 69 (1-4) (2013) 823–834.

[15] D. M. Blough, P. Santi, Investigating upper bounds on network lifetime extension for cell-based energy conservation techniques in stationary ad hoc networks, in: International Conference on Mobile Computing and Networking, Atlanta, GA, USA, 2002, pp. 183–192.

[16] P. Floréen, P. Kaski, J. Kohonen, P. Orponen, Lifetime maximization for multicasting in energy-constrained wireless networks, Selected Areas in Communications, IEEE Journal on 23 (1) (2005) 117–126.

[17] I. Kang, R. Poovendran, On lifetime extension and route stabilization of energy-efficient broadcast routing over MANET, in: International Network Conference, London, UK, 2002, pp. 81–88.

[18] H. Huang, S. Gao, Optimal paths in dynamic networks with dependent random link travel times, Transportation Research Part B 46 (5) (2012) 579–598.

[19] H. Mabed, E. Dedu, H. Skima, Multicriteria optimization in distributed micro-conveying platform, in: 32nd ACM Symposium on Applied Computing (SAC), ACM, Marrakesh, Morocco, 2017, pp. 241–248.

[20] H. Skima, C. Varnier, E. Dedu, K. Medjaher, J. Bourgeois, Post-prognostics decision making in distributed MEMS-based systems, Journal of Intelligent Manufacturing 30 (3) (2019) 1125–1136.

[21] D. Medhi, K. Ramasamy, Network routing - algorithms, protocols, and architectures, Morgan Kaufmann, 2007.

[22] Z. Jiang, Y. Jiao, Y. Sheng, X. Chen, A novel model and its algorithms for the shortest path problem of dynamic weight-varying networks in intelligent transportation systems, Journal of Intelligent and Fuzzy Systems 33 (5) (2017) 3095–3102. doi:10.3233/JIFS-169361.
URL https://doi.org/10.3233/JIFS-169361

[23] K. Nip, Z. Wang, W. Xing, Combinations of some shop scheduling problems and the shortest path problem: Complexity and approximation algorithms, in: D. Xu, D. Du, D. Du (Eds.), Computing and Combinatorics - 21st International Conference, COCOON 2015, Beijing, China, August 4-6, 2015, Proceedings, Vol. 9198 of Lecture Notes in Computer Science, Springer, 2015, pp. 97–108. doi:10.1007/978-3-319-21398-9_8.
URL https://doi.org/10.1007/978-3-319-21398-9_8

[24] E. W. Dijkstra, Communication with an Automatic Computer, Ph.D. thesis, University of Amsterdam (1959).
URL http://www.cs.utexas.edu/users/EWD/PhDthesis/PhDthesis.PDF

[25] S. Lewandowski, Shortest paths and negative cycle detection in graphs with negative weights, Tech. rep., University of Stuttgart (2010).

[26] W. Zeng, R. Church, Finding shortest paths on real road networks: the case for A*, International Journal of Geographical Information Science 23 (4) (2009) 531–543.

[27] S. Hougardy, The Floyd-Warshall algorithm on graphs with negative cycles, Information Processing Letters 110 (8) (2010) 279–281.

[28] M. Kröger, Shortest multiple disconnected path for the analysis of entanglements in two and three-dimensional polymeric systems, Computer Physics Communications 168 (3) (2005) 209–232.

[29] Y. Sheng, Y. Gao, Shortest path problem of uncertain random network, Computers & Industrial Engineering 99 (2016) 97–105.

[30] M. G. C. Resende, P. M. Pardalos, Handbook of Optimization in Telecommunications, Springer, 2006.

[31] K. Deb, Multi-objective optimization, Search Methodologies (2013) 403–449.

[32] K. Miettinen, Nonlinear Multiobjective Optimization, Springer Science & Business Media, 2012.

[33] S. Demeyer, Multiple objective shortest path algorithms for transportation problems, 2009, pp. 72–73.

[34] S. Mohideen, B. Balachandran, A comparative analysis of multi objective shortest path problem, International Journal of Engineering Science and Technology 2 (07 2010).

[35] H. Skima, E. Dedu, J. Bourgeois, C. Varnier, K. Medjaher, Optimal path evolution in a dynamic distributed MEMS-based conveyor, in: International Conference on Dependability and Complex Systems DepCoS-RELCOMEX, 11, Springer, AISC 470, Brunów, Poland, 2016, pp. 395–408.

[36] H. Skima, K. Medjaher, C. Varnier, E. Dedu, J. Bourgeois, A hybrid prognostics approach for MEMS: From real measurements to remaining useful life estimation, Microelectronics Reliability 65 (2016) 79–88.