Multicriteria optimization in distributed micro-conveying platform

Hakim Mabed, Eugen Dedu, Haithem Skima

Multicriteria Optimization in Distributed Micro-Conveying Platform

Hakim Mabed
Univ. Bourgogne Franche-Comté
1 Cours Leprince Ringuet
25200 Montbéliard
hmabed@femto-st.fr

Eugen Dedu
Univ. Bourgogne Franche-Comté
1 Cours Leprince Ringuet
25200 Montbéliard
eugen.dedu@femto-st.fr

Haithem Skima
Univ. Bourgogne Franche-Comté
24 rue Alain Savary
25000 Besancon
haithem.skima@femto-st.fr

ABSTRACT

In this paper, we study the optimal use of a conveying surface, called smart surface, using distributed microrobotic system that is designed to manipulate fragile micro objects. The smart surface is composed of a 2D array of decentralized micro-modules. Each micro-module is composed of a microactuator, a microsensor, computing and communication units. The cooperation among these micro-modules allows the micro-object to be accurately located and moved on the surface. We discuss in this paper the algorithmic solutions that allow the system to determine the best path to move an object from an initial to a target position in the surface. The optimality of the path is evaluated according to two different and complementary criteria corresponding to short and long term visions. The short term vision optimizes on the speed of conveyance. The long term vision optimizes on the lifespan of the system and maintenance issues. We describe the best way to deal with these two aspects and their impact on the smart surface performance. The results observed confirm the improvement of smart surface performance compared to a naive approach, with a gain in lifespan of up to 160% and good objects transfer times.

Keywords

Multicriteria conveying problem, distributed MEMS system, distributed algorithms, shortest path.

1. INTRODUCTION

A conveyor represents a mechanical equipment that allows the transportation and/or manipulation of various kinds of materials. The use of conveyors is profitable in many operations such as the transportation of imposing objects [17] or tiny and fragile components [7, 4]. They are useful for their quick and efficient transportation with a wide variety of materials, which makes them very popular in the materials handling, assembling and packaging industries. The conveying systems differ according to the used mechanisms, summarized by M. G. Kay [12] in 19 different technologies, including the chain conveyor, wheel conveyor, flat belt conveyor, magnetic belt conveyor, bucket conveyor, vibrating conveyor, pneumatic conveyor, etc. Using conveyors is much safer than using a forklift or other machines, and the absence of human intervention makes them a less strenuous and less expensive alternative.

A smart surface [2] is a distributed surface composed of numerous Micro-Electro-Mechanical System (MEMS) developed for the transportation of microscopic objects over short distances. A grid of MEMS microblocks composed of microsensors, microactuators and control units collaborate to move the objects to target locations[1]. The smart surface conveying system presents two main advantages. First, contact-based technologies are not appropriate for fragile and tiny micro-objects (medicines, micro-electronics parts, etc.), which can be easily damaged, contaminated or even scratched during conveyance. Thus, systems based on air-jet technology [14, 3, 6] such as smart-surface system avoid contact with conveyed objects and provide safer manipulation conditions. Second, a conveyor usually consists of a single monolithic block dedicated to a specific task in a fixed environment. As a consequence, when a failure occurs on a component of the system, the system becomes unable to perform the dedicated task and has to be replaced. The modularity of the smart surface system results in self-reconfiguration features (different paths to convey the object until its destination location) and leads to better flexibility and fault tolerance. The distributed approach compared to a centralized one (e.g. external computational resource) provides scalability and fault tolerance.

In this work, we aim to increase the efficiency of future production lines. The conveying principle consists in sending micro-objects from a start block to a final destination using controlled air flow coming from MEMS valves. To do so, the degradation of all MEMS valves involved in the object transport has to be controlled [15]. MEMS components suffer from various failure mechanisms [22, 18] which impact their performance and reduce their lifespan. This

SAC 2017,April 03-07, 2017, Marrakech, Morocco

http://dx.doi.org/10.1145/3019612.3019704

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

highlights the need to monitor their behavior and estimate their remaining time before failure (RUL, Remaining Useful Life), and take appropriate decisions accordingly, such as reconfiguration and maintenance [20]. These tasks are studied using Prognostics and Health Management (PHM) approaches [10, 9, 16].

This paper does not address the PHM aspect, but only the algorithmic solution allowing to coordinate blocks' actions in order to optimally convey the objects. The next section presents research works related to similar problems and emphasizes the particularity of the conveying problem in smart surface system. Section 3 summarizes the main contributions and major highlights of this work. Section 4 presents the distributed MEMS-based surface. Section 5 describes the scalable distributed algorithm used for optimizing the conveying path. Section 6 presents the studied criteria to manage the tradeoff between system lifespan and object transfer time. Section 7 introduces the test simulator and presents the simulation results. The last section concludes the paper.

2. RELATED WORK

Increasing conveying system lifetime by carefully choosing paths presents some similarities with intelligent transportation problems. The energy-saving question in multi-hop wireless networks [1, 5, 11] is one of these problems. During packet routing, some nodes are used more often than others, and their energy may be exhausted faster the others. To avoid this, packets need to be routed through paths which optimize the energy of nodes in the network. The problem is different from smart surface conveying problem. In the network case, sources and destinations are spread over the whole surface, whereas in conveyors only one or a few nodes (usually on the border of the surface) can act as source or destination, leading to specific usage patterns. In addition the reuse of blocks also leads to longer transfer time due to degradation effects, which is not the case in multi-hop wireless network.

Finding optimal paths in road systems is another similar problem [8]. In both problems, external conditions such as road accidents or dust on the smart surface, affect the system characteristics (transfer time of the MEMS blocks or the traffic speed on the roads). There are differences between these problems. In road networks, many cars circulate at the same time. As a result, the traffic flow propagates in both time and space. This sometimes leads to congestion. Instead, in our conveying problem, objects are introduced in sequence and only one object at a time exists on the surface. Secondly, the road infrastructure lifespan is not immediately impacted by the traffic while the lifespan of the smart surface is partially determined by block use. Finally, the road planner is a centralized computing unit with considerable capabilities, while the path optimization is a distributed process since the MEMS blocks have limited memory and computing capacity.

3. CONTRIBUTIONS

As previously mentioned, even though the principles are the same, each problem has its own features and from this point

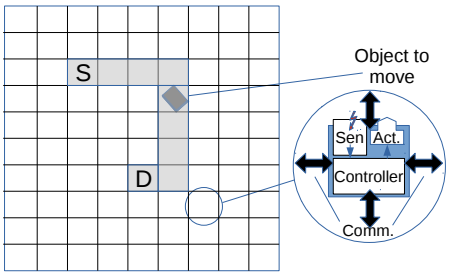

Figure 1: The conveyor and the path taken by an object from S to D.

of view the smart-surface conveying problem is an original one. The distributed nature of the path optimization algorithm is a real challenge. We ensure that the algorithm, run by each block is scalable with a memory and computational complexity less than $O(m)$ (m is the number of blocks). Furthermore, the convergence of the algorithm is ensured with a time complexity $O(m)$. Additionally, to the best of our knowledge, *no article in the literature* deals with increasing conveyor lifespan by avoiding overused blocks. The paper present an original study about how to manage the tradeoff between the short-term performance of the surface (transfer time of an object) and the long-term performance (surface lifespan). The work presented here focuses on sequential conveying system where a new object is introduced one the precedent object is removed from the surface. This assumption allow us to ignore, at this stage, the physical conflicts between objects (collision problem).

4. THE DISTRIBUTED MEMS-BASED CONVEYOR

The conveying surface is composed of a 2D array of decentralized micro-blocks, see Fig. 1. Each block includes a sensor to detect whether an object is above it, an actuator as a MEMS valve, a micro-controller executing code, a power supplier, and a network module allowing communication with its four neighboring blocks. The valves generate air, which allows objects above it to be moved from one point to another.

Let's consider a smart surface composed of m blocks, each one including a MEMS valve. The number of cycles of a block b, $C(b)$, represents the number of objects that the block has already moved to an adjacent position. Every MEMS block is defined by two dynamic values: the RUL and the transfer time. The RUL (remaining useful life) designates the expected remaining number of times that the block could be used before it fails. In real world scenarios, this value is computed in real-time, using the current health state of the block and the formula obtained in the learning step (an offline initial procedure where the behavior of the block is observed); this is studied by Prognostics and Health Monitoring (PHM) field. The transfer time of a given block is the time needed to move an object from the given block to the adjacent one, and depends on valve state, e.g. the more the valve is used, the less it opens to allow the air to push the object. These two values are subject to variations over time

due to a variety of causes. According to the degradation model, criteria such as transfer time of the last object(s), number of uses, frequency of uses and last use date can be considered to forecast the RUL and next transfer time on a given block. Section 7 describes the degradation model used in tests to measure the remaining time before failure of a given block and the transfer time of the block.

In this work, we propose a distributed protocol that allows the surface to determine, in real time, the best itinerary between the current position of an object and its target position. Each time the surface detects an object to move to a given final position, the system computes the best path that reduces the transfer time of the current object while preserving the system lifetime. The distributed conveyance procedure allows each block to determine the neighbor to send the object to if the object crosses it. The conveying procedure called by each block aims to reduce the time to reach the final position and extend the surface lifespan. Whereas the reduction of transfer time is clearly identified as the sum of the transfer time over all the blocks within the path:

$$T(p) = \sum_{b \in p} T(b) \tag{1}$$

the evaluation of the system lifespan on the basis of $RUL(b)$ values is a complex question. Therefore, in this paper we study different measurement functions of the RUL of a path. In the same manner, each block tries to optimize the RUL of the remaining path before reaching the final position. The RUL of a path p starting from block b_{1st} is computed using the recursive function $RUL(p)$ described in equation 2, where F is a function to be defined:

$$RUL(p) = F(RUL(p - \{b_{1st}\}), RUL(b_{1st})) \tag{2}$$

F is an increasing or decreasing monotonic function that should be defined in such manner that it respects the following condition:

$$RUL(p) \left\{ \begin{array}{c} \leq \\ \text{or} \\ \geq \end{array} \right\} F(RUL(p), RUL(b)) \tag{3}$$

where the comparison operator is constant, either $\leq$ or $\geq$. The monotonicity is needed to avoid aberrant situations where adding a loop to a given path may *improve* its evaluation according to the chosen function F.

To convey an object, a block can communicate only with its four neighbors and can send the object to only one of them. When the object reaches its destination, it gets out of the surface, and a new object enters the system.

5. MULTICRITERIA DISTRIBUTED CONVEYING ALGORITHM

As already stated, our problem presents similarities with the problem of finding the shortest path in a graph. Hence, we decide to reuse the well known Dijkstra's algorithm. According to 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.

The distributed Dijkstra approach consists in comparing the best current path with a new path provided by a neighbor, and do this for each node (Algo. 1, line 8). The comparison procedure between two different paths is one of the key aspects in the model discussed in the simulation section. This point defines the selection policy among two potential solutions evaluated according to two numerical criteria, especially when no dominated solution exists.

DEFINITION 1. *Let s_1 and s_2 two potential solutions (paths in our case) 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 or equal than $f_i(s_2)$ and*
- *$\exists f_j, f_j(s_1)$ is better than $f_j(s_2)$*

In our distributed algorithm, each block stores only its own RUL and T, called $myRUL$ and myT, and determines only its *next* block to reach the *dest* position. The memory complexity of the algorithm is reduced to a constant value $O(1)$, making the algorithm scalable. The broadcasting of the state changes of a block is unneeded, hence there is no update message. The complete optimal path is not stored by any block, only the next position is known. Therefore, at the end of the algorithm, every block knows, in advance, where to move the object (*next*) when it is detected above the block.

Our algorithm is presented in Algo. 1, in which the yet to be defined function F is the one given in eq. 2. The different partitions of the path are explored and compared in parallel by blocks to provide the best path, thus reducing the CPU time of the procedure. Each block that receives an $OPTIMIZE$ message from a neighbor compares the path (between the neighbor and the *dest*) with the best ever known. If the new path is better, the considered block sends an $OPTIMIZE$ message to the other neighbors (different from the sender) to inform them about this new best path.

5.1 Termination condition

In the absence of a global knowledge of the system state, the distributed procedure requires a termination condition that ensures that no better path would be found by the neighbors. To this end, two solutions can be used. In the first solution, the worst case is studied to determine the limit of time after which no new path would be found. This worst case happens when the best conveying path, according to F, corresponds to the longest path in terms of number of blocks. Fig. 2 shows three examples of the worst case. These cases occur when the best path includes slightly more than half of the surface blocks. Numerous simulations and bibliographic research showed that the longest path in terms of the number of blocks cannot exceed the value $MaxPathLength$:

$$MaxPathLength = \frac{m}{2} + \max\left(\frac{x}{2} + \tilde{y}, \frac{y}{2} + \tilde{x}\right) \tag{4}$$

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

Proof. Let's consider the worst case described at the right of Fig. 2, where the best found path between the starting

Algorithm 1 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
```

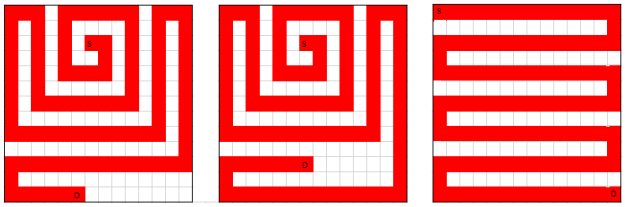

Figure 2: Three examples of conveying algorithm worst case. 'S' represents the starting block and 'D' the destination block.

and the final blocks follows a zigzag (snake) pattern which represents the longest path in the $x \times y$ matrix [13]. In the worst case, the massively distributed algorithm converges when the longest path, which is also the best path, is evaluated. Let's consider a horizontal zigzag pattern. The path is composed of $y/2$ or $y/2+1$ lines (depending on whether the number of lines is even or odd) plus $y/2$ blocks to link the lines. The longest path length is then $(y/2 + y\%2)x + y/2$. On the contrary, if the zigzag pattern is vertical, the longest path length is $(x/2 + x\%2)y + x/2$.

We assume that a message sent during one round of the computing process is available during the next round at the receiver. Therefore the algorithm will be run on each block $MaxPathLength$ rounds. The time complexity of the algorithm is then $O(m^{\frac{1}{2}})$, including potentially many idle rounds that do not consume energy (rounds with no new received message).

The second solution to ensure the termination of the algorithm is to limit the program execution to $MaxRounds \ll MaxPathLength$ rounds. Indeed, long paths, even if they satisfy the optimization criteria, increase the long-term degradation of the surface since many blocks are used. Therefore the importance of optimal resolution of the problem should be put into proper perspective by constraining the conveying path length. The execution time of the algorithm is then improved without really impacting the solution quality. In our tests we set $MaxRounds$ to twice the direct trajectory length between the source and the destination of the object, bringing the time complexity of the algorithm to $O(m)$:

$$MaxRounds = 2 \times d_{Manhattan} \tag{5}$$

6. OPTIMIZATION CRITERIA

The aforementioned algorithm uses an objective function F which computes the quality of a given path in order to select the best one (the maximum or minimum). In this section, we analyze the best way to combine the two criteria (RUL and T) within a single function. The choice of the objective function is a key factor of the efficiency of the proposed solution. It determines the desired trade-off between the short-term (fast object conveyance) and the long-term vision (high system lifespan). The objective function determines also how the lifetime notion measured for each block is extended to measure the global degradation of the system, and especially the impact of a given conveying path on the system durability. In the following paragraphs, we present the different objective functions.

Direct line In this case, the notion of RUL and T are completely ignored and a naive approach is adopted consisting in the direct line linking the source and the destination blocks. Therefore the Dijkstra's algorithm is not used and no optimization criteria is needed.

MaxMin RUL function In MaxMin funtion, the path RUL is equal to the smallest RUL in the path (eq. 6). The objective is to find the path with the highest value of RUL. $RUL(p)$ and $RUL(p')$ are compared to determine which path, p or p', is better. If $RUL(p)$ and $RUL(p')$ are equal then the path with a better transfer time is taken.

$$\max_p RUL(p) = \max_p \min_{b \in p} RUL(b) \tag{6}$$

Powers function The RUL of a path corresponds to the sum of $\beta^{C(b)}$ over the blocks of the path:

$$\min_p RUL(p) = \min_p \sum_{b \in p} \beta^{C(b)} \tag{7}$$

where β is a numerical parameter to be defined. The use of the function power aims to favor the selection of blocks with lower usage by minimizing the value of the function.

Minimum transfer time function The chosen path is the fastest path, i.e. with the minimum transfer time to convey an object from source to destination, no matter the RUL of path's blocks:

$$\min_p T(p) = \min_p \sum_{b \in p} T(b) \tag{8}$$

7. SIMULATION AND ANALYSIS

In this section, we present the simulator and the scenarios used, and compare the different objective functions presented above.

7.1 Scenarios

A scenario is defined by the dimensions of the surface, $n \times n = m$, by the generation rule of the initial and final positions of the objects, and by the degradation model of the system. We simulate surfaces of dimension 6×6, 10×10, 15×15 and 20×20.

We also consider two kinds of rules which select randomly the initial and final positions of objects: in the first rule, source and destination can be any block on the surface; in the second rule, the source is on an edge of the surface and the destination on *another* edge. The first rule refers to manipulation or placement applications while the second rule fits sorting applications.

We use a simple model to estimate the Remaining Useful Life of a given block. In this model, a block is conceived to be usable a given number of times, $C_{\max}$, corresponding to its theoretic capacity (the number of times that the air-jet valve can be activated to move an object before being revised or replaced). Therefore, the remaining time before failure of the surface is determined according to the number of uses of each block. The conceptual capacity $C_{\max}$ of the blocks is known and given by the constructor.

$$RUL(b) = C_{\max} - C(b) \tag{9}$$

We also consider two transfer time degradation models. The first one is a dummy model, also used in [19], where a block transfer time is linearly degraded when the block is used, so each time a block is crossed by an object, the transfer time increases by a constant value α:

$$T(b) = T_0 + \alpha C(b) \tag{10}$$

The second degradation model is a more realistic model, based on experimental results [21]. In our case, we introduce a probabilistic degradation effect modeled by an exponential distribution with mean α. The transfer time of a block b is simulated by a recursive function where after each activation of a block, its transfer time is degraded as follows:

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

where RND is a random value uniformly selected in [0,1].

In the following, we refer to a scenario by the notation $n : GEN : DEG$, where $n \times n$ is the dimension of the surface, $GEN \in \{any, edge\}$ is the generation rule of the objects, and $DEG \in \{lin, exp\}$ is the degradation model used.

7.2 Simulator

We implemented a simulator in VBA Microsoft Excel. It executes a given scenario until a maintenance procedure is required as a result of a block degradation ($\exists b, RUL(b) = 0$). The simulator algorithm is described in Algo. 2.

In all tests, we used the following parameters: $C_{\max} = 100$, $T_0 = 1\,\text{s}$, $\alpha = 10\,\text{s}$, and the parameter in powers functions is $\beta = 1.04$. The initial surface is new, i.e. $C(b) = 0$ for all blocks.

Algorithm 2 Procedure of simulating a scenario.

Require: n, GEN, DEG
1: **while** no need for maintenance **do**
2: a new object enters the surface at initial and final position given by GEN method
3: run the distributed algorithm to convey the object
4: update C, RUL and transfer time values according to DEG method
5: object gets out of the surface
6: **end while**

7.3 Optimization Criteria Comparison

In this section we test and discuss the advantages and disadvantages of the different objective functions presented before. We consider each of the objective functions for 16 scenarios. A comparison is made on the basis of the number of conveyed objects, the average transfer time of conveyed objects, and the average transfer time of only the first 200 conveyed objects. The latter is important since a surface which conveys many 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.

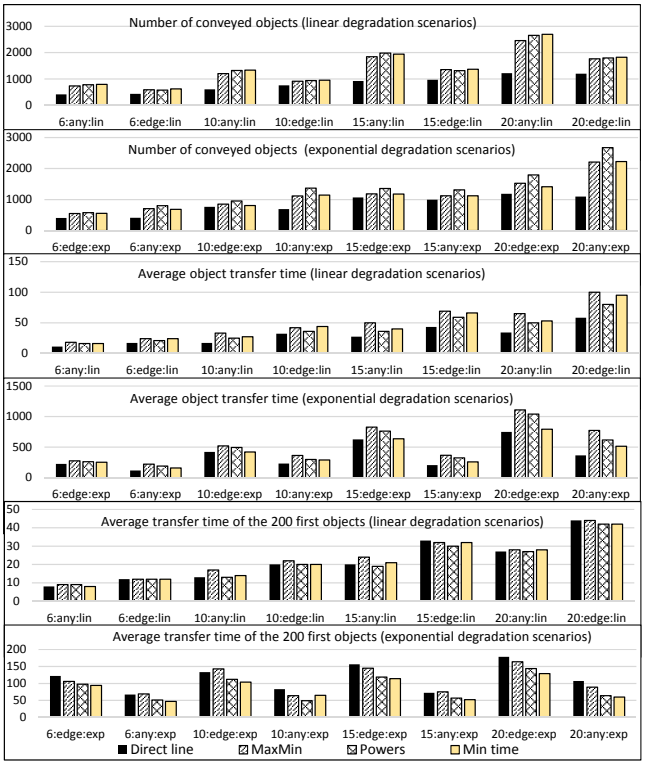

Figure 3: Comparison of different objective functions.

Figure 3 shows the results of this comparison. The functions based on the entire path's blocks such as powers function are better than the functions based on a part of the path, such as MaxMin function. Trying at any cost to avoid overused blocks leads to more crossed blocks with only marginally less degradation (see 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). In some cases, generated paths are longer, leading to many blocks used every time. With the scenarios based on linear degradation, the Min time function performs slightly better than power function. The reason for this is that, in such scenarios, path transfer time is strongly correlated with the path sum of RUL (eq. 9 and 10). However, in exponential degradation scenarios (where RUL and time functions are less correlated), the Min time function provides clearly worse results than the power function.

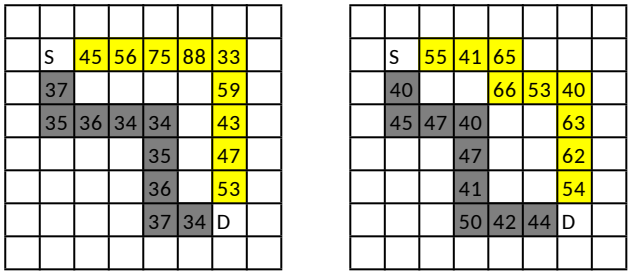

Figure 4: Numerical values refer to $RUL(b)$. In (a) the dark left path is better according to the minimum RUL value (34 against 33), but when using this path many worn blocks become even more worn. In (b) the two paths are equal according to the minimum RUL value (40), but the light right path is better since its blocks have generally higher values.

The good performance of the powers function can be explained mainly by the penalization of the use of overused blocks. The idea is to ensure that the distributed algorithm prefers to use b rather than b' if $C(b) < C(b')$. The Powers function represents, therefore, a good tradeoff between the MaxMin function and a hypothetical average RUL function. An example is given in figure 5. According to MaxMin function, the bottom left path is the best with a minimum RUL equal to 14 (against 12 and 13 for the two other paths). However this path is clearly the worst since it crosses 7 blocks with low RUL. According to the average RUL function, the middle and the top right paths are equal, whereas using the power function, the top right path will be preferred because the use of the block with RUL=12 is more penalized.

The direct line function provides the worst results. Results show that central blocks are much more frequently used than edge blocks, making the system falls sooner. The good results of direct line function according to the average transfer time are due to few conveyed objects, which shows that direct line method leads to a heterogeneous use of the surface.

In terms of average transfer time, the functions emphasizing transfer time (Min time function) provide the best results. However, this criterion alone is not sufficient, since, as stated in the beginning of the section, a function, such as powers, which conveys more objects is penalized compared to a function, such as average time, which conveys fewer objects; indeed, last conveyed objects take more time than first conveyed objects, due to the surface degradation. To complete this comparison we should also consider the average transfer time of the first 200 objects. In this case, power function provides good results. For example in scenario 15:any:exp, power function conveys the first 200 objects in an average time of 57 s, while minimum transfer time function conveys the first 200 objects in an average time of 56 s (just one second of difference).

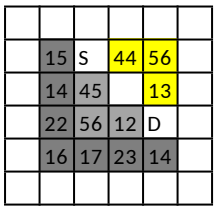

Figure 5: Numerical values refer to $RUL(b)$. The bottom left path is the path selected by MaxMin function even if it is the longest one. The sum of RULs of blocks (path RUL) is identical for the middle and top right paths. In contrast, using powers, the top right path is preferred since crossing the block with RUL=12 is more penalizing.

8. CONCLUSION AND PERSPECTIVES

In this paper we studied the problem of conveying objects on a 2D distributed MEMS surface called smart surface. Despite some similarities with well known problems in transportation and communication fields, the problem remains particular. This particularity is due to the millimetric scale of the system, inducing hardware and software constraints (memory storage, computing capabilities), and to the degradation phenomena observed on the MEMS modules. Degradation effects make that a path to convey an object could be good at a given moment and become bad later. Therefore we proposed a bicriteria model that takes into 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 proposed a scalable (in terms of computational and memory complexity) distributed algorithm to optimize the conveying path and we studied different ways to combine the short-term and long-term criteria within a single objective function used by the optimization algorithm. We proved that methods based on the progressive penalization of the overused blocks provide best results.

The results obtained are encouraging since the lifespan of the system is extended by up to 160% without reducing conveying speed. On the contrary, unbalanced use of the surface blocks provokes the reduction of conveying speed. The relationship between the number of use cycles and the degradation of the transfer time determines the suitable tradeoff

between the RUL and transfer time criteria. We have analyzed two different degradation models defining this relationship. However, other phenomena that could potentially impact the degradation level should be studied, such as the frequency of use, the non-use of a block for long periods, etc.

Two major studies are needed to complete this work. First of all, how to adapt the model and the distributed algorithm to take into account the presence of concurrent objects above the surface. Secondly, we need to introduce the number of orientation changes in the object's followed path as a third criterion to minimize. Indeed, every orientation change implies to stop the object and then start it in the new direction, and both stopping and restarting the movement consume energy and time.

9. REFERENCES

[1] Blough, D. M., and Santi, P. 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, Sept. 2002), pp. 183–192.

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

[3] Dahroug, B., Laurent, G. J., Guelpa, V., Fort-Piat, L., 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.

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

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

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

[7] Helin, P., Calin, M., Sadaune, V., Chaillet, N., Druon, C., and Bourjault, A. Micro-conveying station for assembly of micro-components. In *Intelligent Robots and Systems, 1997. IROS '97., Proceedings of the 1997 IEEE/RSJ International Conference on* (Sep 1997), vol. 3, pp. 1306–1311 vol.3.

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

[9] Javed, K. *A robust and reliable data-driven prognostics approach based on extreme learning machine and fuzzy clustering.* PhD thesis, University of Franche-Comté, Besançon, France, 2014.

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

[11] Kang, I., and Poovendran, R. On lifetime extension and route stabilization of energy-efficient broadcast routing over MANET. In *International Network Conference* (London, UK, July 2002), pp. 81–88.

[12] Kay, M. G. Material handling equipment, 2012.

[13] Keshavarz-Kohjerdi, F., and Bagheri, A. An efficient parallel algorithm for the longest path problem in meshes. *The Journal of Supercomputing 65*, 2 (2013), 723–741.

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

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

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

[17] Pang, Y., and Lodewijks, G. Large-scale conveyor belt system maintenance decision-making by using fuzzy causal modeling. In *Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005.* (Sept 2005), pp. 563–567.

[18] Shea, H. R. Reliability of MEMS for space applications. In *MOEMS-MEMS 2006 micro and nanofabrication* (2006), International Society for Optics and Photonics, pp. 61110A–61110A.

[19] Skima, H., Dedu, E., Bourgeois, J., Varnier, C., and Medjaher, K. Optimal path evolution in a dynamic distributed {MEMS}-based conveyor. In *International Conference on Dependability and Complex Systems DepCoS-RELCOMEX* (Brunow, Poland, June-July 2016), 11, Springer, AISC 470, pp. 395–408.

[20] Skima, H., Medjaher, K., Varnier, C., Dedu, E., and Bourgeois, J. Hybrid prognostic approach for micro-electro-mechanical systems. In *IEEE Aerospace Conference* (Big Sky, Montana, USA, Mar. 2015), 36, pp. 1–8.

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

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