Reducing Roadmap Size for Network Transmission in Support of Cloud Automation

Kostas Bekris, Rahul Shome, Athanasios Krontiris, Andrew Dobson
Department of Computer Science - Rutgers, the State University of New Jersey
Piscataway, 08854, NJ, USA
Email: kostas.bekris@cs.rutgers.edu

This work aims to highlight the benefits of Cloud Automation for industrial adopters and some of the research challenges that must be addressed in this process. The focus is on the use of cloud computing for efficiently planning the motion of new robot manipulators designed for flexible manufacturing floors. In particular, different ways that a robot can interact with a computing cloud are considered, where an architecture that splits computation between the remote cloud and the robot appears advantageous. Given this synergistic robot-cloud architecture, this work describes how solutions from the recent literature can be employed on the cloud during a periodically updated preprocessing phase to efficiently answer manipulation queries on the robot given changes in the workspace. In this setup, interesting tradeoffs arise between path quality and computational efficiency, which are evaluated in simulation. These tradeoffs motivate further research on how motion planning should be executed given access to a computing cloud.

I. Emerging Opportunities in Cloud Automation

Traditional automation involves single-task mechanisms carefully separated from people operating in highly-structured environments. These systems execute repetitive tasks with high efficiency and accuracy without the need for significant computation. Such solutions are cost effective only for large-scale industrial settings where similar products are assembled for long periods of time. A transformation is taking place, however, fueled by financial drivers, as well as technological and foundational developments. The future of automation lies in flexible factory floors and quick burst manufacturing processes, which can provide complex, short life-cycle products fast. This will allow product types and supplies to be updated frequently, where the factory floor is reconfigured on-demand through interaction with human operators. This requires a new level of adaptability of the automation infrastructure, effectiveness even in less-structured setups and safe co-existence with people.

Part of the solution lies in the increasing availability and affordability of rich sensors (e.g., 3D point cloud devices) and new compliant, light-weight arms that are easily programmable (e.g., Baxter robot, Universal arms, etc.). Such solutions can be adopted by smaller-scale businesses, such as in food processing, which have largely missed out on the benefits of automation. But they also require dealing with higher-level and computationally demanding cognitive processes, such as perception, modeling, planning, learning and coordination. Thus, future, smaller-scale, flexible automation facilities will depend upon effective computational solutions, both at a systems and an algorithmic level.

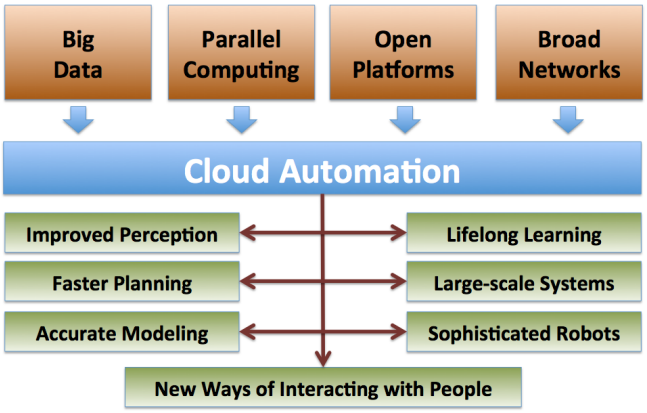

Fig. 1. Aspects of cloud automation and the different ways it can alter the automation landscape.

At the same time, there is a revolution that is taking place in computing and which can influence the transformation of automation. Technological advances make it easier to access computing as a service over a network, where shared resources, software, and information are available to individual machines and devices [1]. This development can alter the automation landscape in many different ways, giving rise to the new field of Cloud Automation [10], [13], [16]. In particular, the use of cloud-based computation can provide effective access to big data, which can include global knowledge-bases and libraries of object models, meta-data or environmental maps. Similarly, the cloud can leverage parallel computation on demand to quickly address the needs of computationally demanding robot operations, such as motion planning. The cloud can even act as the intermediary between the automation facility and the human operators, which can in this way easily access a global view of industrial operations. It can also accommodate the interaction between multiple robots, which can now share knowledge and learn collectively from experience over time.

Through these mechanisms, cloud computing can be applied across a variety of industries to significantly advance the efficiency, quality, productivity and reliability of the automation process. Some of the areas that will be beneficially affected are the following (see Figure 1):

Improved Perception and World Knowledge: Feeding data from rich sensors to the cloud can help to continuously

monitor the location of products, tools, people and equipment in a flexible, potentially less structured factory floor [16]. Access to large knowledge-bases and databases can provide quick identification of not only the type of perceived objects but also access to meta-data that allow to reason about how robotic equipment can handle and manipulate them [7].

Fast Resolution of Planning Challenges: Some of the most computationally demanding processes in robotics and automation relate to planning, manipulation and grasping [3], [7], [11]. They require searching high-dimensional, continuous configuration spaces and model the different ways a robotic equipment can interact with a physical object. Massively parallel computation can significantly help by speeding up the operation of sampling-based processes for planning [9], [14] and evaluating multiple hypothesis about the state of the world given uncertainty [11].

More Accurate Models: Frequently, robotic algorithms make concessions by simplifying the underlying model to achieve reasonable solution times. Access to significant computational power allows the adaptation of higher-fidelity models for planning purposes, including physics-based simulation [2]. This is a computationally expensive alternative but can reveal physical interactions, which are difficult otherwise to reason about, including for novel objects, and in applications, such as one-time and on-demand manufacturing.

Opportunities for Life-long Learning: Cloud computing can be used to access and process examples and demonstrations of past physical interactions with objects. Given this knowledge, it is possible to employ life-long learning for improving a robot's interaction with its surroundings over time [17]. It also allows to collect data regarding the behavior of people, their interaction with the same objects and learning from these observations. This can take place in a collective manner, where information can be shared among different facilities and improved over time [23].

Highly Networked Large-scale Systems: Networked autonomous systems can coordinate their motion via the cloud to improve overall efficiency. An example is Kiva System's automated warehouse management system, which achieves scalability to thousands of robots [25]. In the context of distributed automation, each robot can now have access to global information by using the cloud. This simplifies coordination challenges relative to decentralized alternatives.

Computing for Sophisticated Robots: Cloud-based factories will be able to easily incorporate more sophisticated robots, such as humanoids [13], and new types of sensors or interaction devices, which increasingly depend upon computation to operate effectively.

New Ways of Interaction with People: Cloud computing provides a global view of an automation facility to a remote human operator through sophisticated interfaces [4]. This can include 3D displays where a remote operator observes the operation from different viewpoints in the factory floor. It also allows people to provide feedback online. Recommendations can be incorporated directly in the online knowledge-base, as well as in the planning and modeling processes taking place in parallel on the cloud [22].

II. Schemes for Cloud-based Planning

A key automation operation that can greatly benefit from the use of cloud computing involves motion planning and manipulation. For instance, consider a manipulator, such as Baxter, which can be used in a factory floor to package products. Given a more flexible, less structured environment, the products to be manipulated are not guaranteed to be always in the same configuration. Sensing is used to detect and keep track of their location. Moreover, the environment is generally dynamic and can involve multiple people, robots, tools, containers and products, which continuously move in the facility. The robot is directly controlled by a local workstation, which can communicate with a computing cloud.

The question is how the local workstation can best interact with the larger-scale computing cloud so that the manipulation task is accomplished efficiently. There are two extremes regarding the interaction of the robot with the cloud (Fig. 2):

1) **Local-only Computation***:* The traditional setup involves a planner running locally on the robot given the latest sensing input available to it without any external interaction. In this case, the robot continuously updates the internal representation of the world given its sensing input and then calls a motion planner (e.g., an RRT-based sampling-based planner for dealing with individual queries [14]), to compute a manipulation path on the fly for reaching the target object, transferring it and then releasing it.

2) **Cloud-only Computation***:* The alternative corresponds to the robot outsourcing this computation to the cloud. In particular, the robot's workstation can continuously forward the sensing data to the cloud, which maintains an updated representation of the environment. The computation of the manipulation path is computed by a motion planning algorithm that resides exclusively on the cloud. Upon the computation of a path, the cloud transmits this solution to the robot, which follows the cloud computed path. Thus, the communication loop in this scenario involves the robot transmitting sensing data and receiving solution trajectories.

The last alternative describes a potentially viable way of incorporating cloud computing to robotic operations but also raises some concerns. In particular, it requires frequent transmission of large amounts of data to the cloud, including the frequent communication of 3D point clouds. Furthermore, in this scenario the robot is fully dependent on the cloud and the reliability of this communication. Given a communication failure, which can occur even in a relatively reliable industrial environment, or any other issues that may arise in interacting with the cloud, the robot is not able to plan and react to changes in its environment.

3) **Synergistic Robot-Cloud Computation***:* The above issues motivate a hybrid solution, where some computation takes place on the local workstation but the cloud is responsible for the more computationally heavy but not time-critical operations. Such a solution should result in a) reduced communication load relative to cloud-only computation and b) increased responsiveness to environmental changes relative to the local-only computation, where the robot is limited by the computational capabilities of an individual workstation.

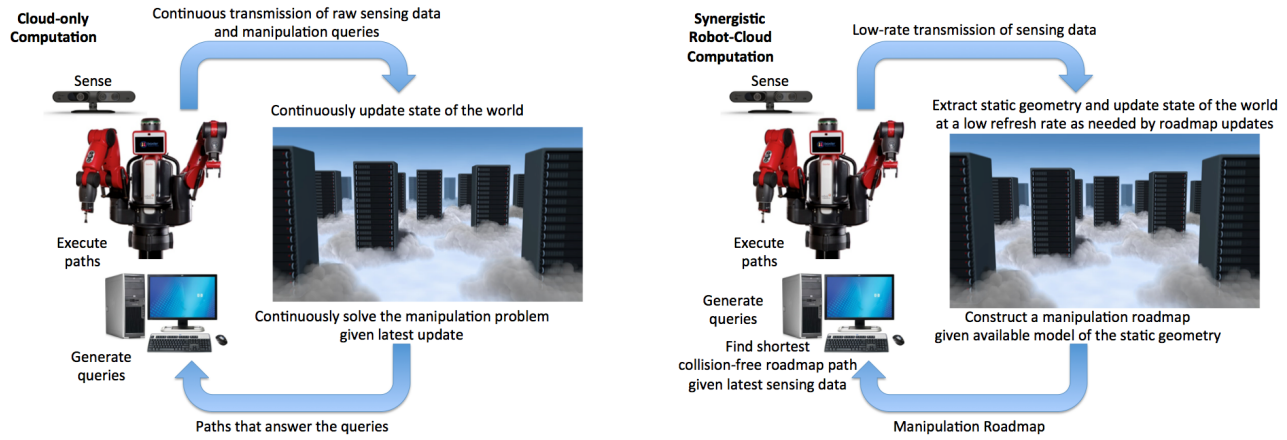

Fig. 2. Different ways of interaction between a manipulator (e.g., Baxter) and the cloud. (left) Cloud-only computation. (right) This work argues the benefits of a synergistic cloud-robot computation model that is centered around the exchange of proper motion planning data structures.

In the synergistic mode of operation, the robot only occasionally transmits sensing data to the cloud at a low rate. The computing cloud is then responsible to identify the static obstacles and preprocess the collision-free configuration space of the robot. The result is a motion planning data structure, i.e., a roadmap, which contains collision-free paths given the static geometry, and which can be used for manipulation purposes. When computed, the roadmap is transmitted from the cloud to the robot's workstation where it is used to quickly answer queries.

The local workstation has access to the most updated sensing information, i.e., the latest 3D point cloud, which is more recent than the data available for the roadmap's construction on the cloud. The local workstation does not just blindly follow the shortest path on the roadmap. Instead, it finds the shortest path that does not result in collisions with the latest available sensing data. Thus, the language of communication between the robot and the cloud becomes the motion planning data structure, i.e., the roadmap, which is updated periodically on the cloud given sensing updates and transmitted back to the robot at a low frame rate.

An effective implementation of the above model should aim to achieve the following:

- *Computational Efficiency:* Minimize the time to compute solutions for motion planning queries on the local workstation relative to local-only computation. This requires taking advantage of the cloud-based preprocessing.
- *Communication Overhead:* It must reduce the communication overhead of the robot-cloud interaction relative to the cloud-only computation. This should help to reduce the sensitivity of the approach to bandwidth, latency and throughput challenges.
- *Safety and Reactivity to Changes:* It should return safe solutions, where the robot is able to safely avoid any obstacles that are detected by its sensors but which where not known to the cloud when the roadmap was constructed.
- *Path Quality:* It should compute high-quality solutions, where the manipulator moves in an efficient manner without executing redundant motions that will delay the execution of the task.

III. Planning Using Roadmap Precomputation

The following discussion focuses on motion planning primitives from the related literature that can be used to achieve a proper balance among the above four objectives in the context of a synergistic robot-cloud computation.

A. Computing a Manipulation Roadmap

The planner executed on the cloud can explore the task's state space by generating a manipulation roadmap $\mathcal{R}(\mathcal{V}, \mathcal{E})$, given the robot and an object that needs to be manipulated [21]. The geometry of the object is assumed known but the configurations that it will appear in front of the robot are not. The method operates over collision-free states of the form $x = ((q, p), t)$, where $q \in \mathcal{Q}$ is a configuration for the manipulator (e.g., a 7-dim. configuration for a Baxter arm), p is the configuration of the single object (a 6-dim. rigid body configuration) and t specifies the type of the manipulation state, i.e., a transfer or a transit. The roadmap is a graph with vertices that correspond to individual manipulation states and edges that correspond to local paths connecting such states.

A transfer state corresponds to one where the manipulator is grasping the object. A transit state corresponds to one where the object is in a stable configuration without any forces exerted by the manipulator. In a transit state, the manipulator may not be grasping the object. A transition state corresponds to stable grasps, i.e., these states lie at the intersection of transfer and transit states (e.g., see Fig. 3). Thus, a multi-modal roadmap is needed for manipulation, which contains two types of nodes, those that are transfer states and those that are transit states. Edges between nodes of the first type correspond to the robot transferring an object, while edges

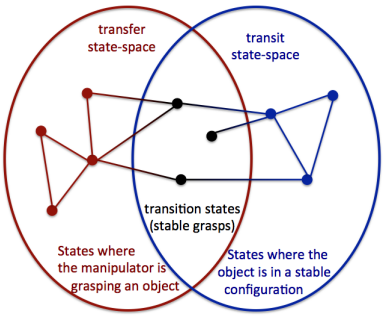

Fig. 3. The constructed roadmap needs to span the space of manipulation planning.

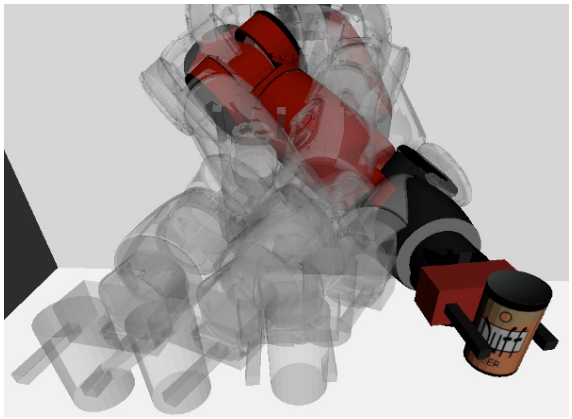

Fig. 4. Transition states for a Baxter robot and an object corresponding to stable grasping configurations. Given the object placement and a grasp, an IK solution can be used to compute the grasping arm configuration.

between nodes of the second type correspond to the robot moving in the environment while the object is placed in a stable configuration. An edge reaching a transition state leads to the robot grasping an object in a stable configuration.

A popular approach for generating such roadmaps is following a sampling-based process, such as in the Probabilistic Roadmap Method (PRM) [9]. The nodes/states of the roadmap are sampled and an attempt is made to connect pairs of neighboring nodes according to a metric as long as the edge/local path between the two nodes in the pair is collision-free. In the context of manipulation, the first step in the construction of the manipulation roadmap $\mathcal{R}$ is to sample transition states. This can be achieved by first sampling a stable, collision-free object configuration p, and then using inverse kinematics to define the corresponding manipulator's grasping configuration q (Fig. 4). If the inverse kinematics function returns a solution and the resulting configuration (q, p) is collision-free, then two states are generated with the same configuration: one of the transit type and one of the transfer type. Both of these states are inserted in $\mathcal{R}$ and an edge is added between them.

Given the set of transition states and edges, the method needs to proceed to separately explore the transit and the transfer subset of the state space in a PRM fashion. For the transit component, additional grasping configurations for the manipulator are sampled, where the object is no longer required to be in a stable configuration. Then neighboring transit states according to a configuration space metric are considered for connection by interpolating between the two configurations. If the interpolated path is collision-free, the edge is added to the roadmap. The same process is repeated for transfer states. The objective is to achieve a roadmap that has the minimum number of connected components, so that the manipulator is able to transfer objects among the sampled transition states.

B. Using a Precomputed Roadmap

To effectively utilize the roadmap $\mathcal{R}$, which is computed on the cloud, on the local workstation, a lazy evaluation process for computing the shortest collision-free path is helpful.

In particular, Algorithm 1 provides an outline of the operation on the local workstation. The algorithm assumes the availability of the manipulation roadmap $\mathcal{R}(\mathcal{V}, \mathcal{E})$ that has been computed and transmitted by the cloud. The algorithm also has access to the latest local sensing data $\mathcal{S}$. The sensing data can be in the form of an unfiltered point cloud, which can be used by appropriate software packages [18] to perform collision checking given a manipulation state x for the robot and the object. This means that no expensive processes for modeling the world need to be executed on the local workstation but raw data can be used directly.

Algorithm 1: $LazyEval(\mathcal{R}(\mathcal{V}, \mathcal{E}), \mathcal{S}, x_s, x_g)$

```
1 π ← ShortestPath(R(V,E), x_s, x_g);
2 while π ≠ ∅ and ColFree(π, S) ≠ True do
3     for e ∈ E ∩ π so that ColFree(e, S) = False do
4         E = E \ e;
5     π ← ShortestPath(R(V,E), x_s, x_g);
6 return π;
```

The algorithm starts by searching the roadmap for a solution path π from the start manipulation state x_s to the goal one x_g (Line 1). This typically entails a heuristic search process on the graph. The shortest path π on the roadmap is then tested for collisions given the latest sensing data $\mathcal{S}$ (Line 2). If the path is collision-free, then it can be returned (Lines 2 and 6). If not, then all the edges $e \in \mathcal{E}$ of $\mathcal{R}$ that were used by the path π are removed (Lines 3-4). Then a new solution path is computed on the updated roadmap (Line 5). If no solution path is found after the removal of edges, then a failure to find a solution is reported by returning an empty path (Lines 2 and 6).

Every time that the process of Algorithm 1 is completed, any edges that have been removed from the roadmap are reinserted as they may be useful in consecutive queries and for updated sensing date $\mathcal{S}$. In the case of a failure, the robot can revert to a similar behavior to that of local-only computation and search the configuration space of the robot. This can involve either starting a new motion planning process from scratch or by extending the available roadmap.

C. Effects of Roadmap Properties to Cloud-based Schemes

The above process returns the shortest collision-free path, if one exists on the available roadmap. The success of the overall operation in terms of the metrics, i.e., computational efficiency, safety and path quality, will depend on the characteristics of the roadmap computed on the cloud. This is where recent theoretical progress in understanding the properties of motion planning data structures plays an important role. These properties influence the type of roadmaps that should be computed by the cloud given a dynamic scene.

In particular, the PRM approach is probabilistically-complete, i.e., it will eventually provide a roadmap that contains solutions to every motion planning query that is solvable [9]. Recent results also identify the requirement for a sampling-based roadmap to provide good quality paths [8]. A roadmap method is asymptotically optimal if any new node

added to the roadmap is tested for collision-free connections to at least $logn$ neighbors, where n is the number of roadmap nodes. The property of asymptotic optimality means that as the number of nodes goes to infinity, the probability of returning the optimal path goes to 1. The asymptotically optimal version of PRM is called PRM* [8].

Based on these properties, it becomes apparent that large, dense roadmaps are beneficial for providing good quality solutions with high probability. Given a computing cloud's access to vast computing resources, it may seem desirable to build an as heavy data structure as possible in the context of cloud automation. Nevertheless, this data structure needs to be communicated to the robot's workstation, where it will also need to be stored and queried locally. The workstation may not even be able to store a huge data structure managed distributedly on the cloud. The size of the communicated data structure should be such so that it can still fit in the memory of an individual machine. Furthermore, a very large and dense roadmap will result in slower query resolution relative to smaller, sparser roadmaps because the cost of the heuristic search process directly depends on the graph size. Additionally, a large, dense roadmap will take more time to construct relative to a smaller alternative, even if a computing cloud is employed. Thus, the size of the data structure will affect the refresh rate of the roadmap given the latest available sensing data transmitted from the robot.

In this way, an important tradeoff arises between the query resolution time and path quality, which is influenced by the size and density of the computed roadmap. Recent research efforts have focused exactly on this tradeoff and aim towards compact motion planning representations with desirable near-optimality guarantees in terms of path quality [5], [15], [20], [24]. Note that naively pruning the PRM* roadmap by removing edges and nodes so as to achieve a compact representation results in a data structure that (a) with high probability fails to answer queries, which can be answered by PRM*, and (b) for the cases it succeeds, it results in significant degradation in path quality. This is why this work does not consider such solutions.

A useful tool towards the definition of compact representations is the notion of a roadmap spanner [5], [15], which extends the idea of graph spanners [19] in general configuration spaces. A t-spanner is a subgraph of a graph, where the paths between all pairs of vertices are dilated by a factor of t relative to paths on the original graph, where t is the stretch factor of the spanner. This means that a graph spanner is a more compact representation of the original graph with path quality degradation guarantees.

An extension of this notion in the context of the PRM* algorithm resulted in the Incremental Roadmap Spanner approach (IRS), which provides the same property in configuration spaces [15]. IRS returns a roadmap that has the same set of vertices like PRM* but only a subset of its edges. IRS is guaranteed to return a solution path to a motion planning query that is at most t times longer than the solution returned by PRM* with the same set of vertices, where the stretch factor t is an input parameter. Existing work has showed that IRS provides orders of magnitude sparser data structures [15], which has a direct relation to online query resolution time and in practice the degradation in path quality is significantly smaller than the theoretical guarantee provided by the stretch factor. Furthermore, a recent variant of the approach can be constructed significantly faster than PRM* [6], which in the context of a robot-cloud synergistic computation would allow a faster refresh rate of the roadmap given sensing updates.

The notion of a spanner can be pushed even further in continuous spaces and include the rejection of roadmap vertices. In graph theory, every node of the original graph is maintained in the spanner. But this does not need to be the case for continuous spaces. An individual configuration can be used to represent its local region and no neighboring samples need to be added as new nodes in the roadmap if they are not necessary for completeness or path quality purposes. This realization has led to Sparse Roadmap Spanner (SPARS) approaches [5]. This framework can achieve additional improvements in terms of roadmap size and online query resolution time compared to IRS. It does not make significant sacrifices in path quality or any sacrifice in terms of the number of queries that the method can answer relative to PRM* for the same number of iterations.

The existing evaluation of compact roadmap approaches has focused on scenarios where the roadmaps are used in the same space where they have been constructed. They have also not been tested in manipulation challenges. In the context of dynamic and flexible manufacturing floors, however, changes may occur in the environment between the time that the roadmap was constructed on the cloud and when it is queried by the robot. Thus, it is interesting to evaluate the extent to which the benefits of compact motion planning representations apply to the case of a changing environment.

IV. Evaluation of Methods and Tradeoffs

This section evaluates PRM*, IRS and SPARSin terms of computational efficiency, path quality and success ratio for a changing scene. The setup shown in Figure 5 involves a 7 DOF Baxter arm that must transfer three rigid bodies. The experiments were performed with the help of a simulation software [12]. In a situation similar to the one in Kiva Systems' warehouses, mobile robots may bring a shelf containing products in an area so that they will be packed. While today this task is performed by people, it is envisioned that in the future, robot manipulators will be able to address this challenge. Such robots will have access to significant computing power and precomputation can be employed to improve their performance. But neither the shelf, nor the objects will be always accurately placed and locally the robot needs to adapt to the current state of the scene.

Inspired by this scenario, there are three variations of the problem considered in the experiments performed here: an easy, a medium-level and a hard difficulty instance. The easy challenge involves the objects placed on a tabletop surface. This is the environment that is provided to the roadmap during the precomputation phase. The medium difficulty problem has the objects placed inside a shelf, which is not

Fig. 5. The three versions of the environment for evaluating the methods, inspired by the Amazon Picking Challenge. From left to right: easy, medium, hard. The offline roadmaps are computed given the easy environment. This problem involves a Baxter arm grasping an object from the tabletop in a vertical configuration and placing it inside a box in a horizontal configuration. The medium challenge has the objects placed inside a shelf that was not known during the construction of the roadmap and which limits the motion of the robot. The hard challenge has multiple shelves, the objects placed deeper inside these shelves, which further limits the capability of the arm of grasping these items.

available during roadmap construction. The hard instance has the objects placed deep inside multiple shelves, close to obstacle geometry in a way that complicates their grasp. The objects are initially placed on a horizontal surface in a vertical configuration and must be placed in a horizontal configuration inside a box. For a challenge to be considered solved, the robot must manage to transfer all three objects.

The pick-and-place subtask for each object is solved by querying a multi-modal manipulation roadmap, computed as described in the previous section in three different ways (PRM*, IRS and SPARS), which give rise to graphs of different density and size. Each pick-and-place manipulation problem is split into three sub-problems: a) a transit plan from an initial configuration of the arm to a grasping configuration of the movable object on the horizontal surface, b) a transfer plan between the reachable grasping configuration of the initial object location to a grasping configuration at the goal, c) a transit plan from the grasping configuration at the goal to the safe initial arm configuration.

The higher-level task planner attempts three different randomly sampled grasps at the initial and goal object positions and evaluates all nine combinations to find the best solution given the available roadmap. This means that for a single pick-and-place task, the transit roadmap is queried six times (three ways to reach the object at the initial location and three ways for the arm to be retracted from the goal placement). The transfer roadmap is queried nine times, i.e., for all combinations between grasps in the initial and goal placement of the object. The reported online query resolution time is the time taken to transfer all three objects, which corresponds to eighteen queries of the transit roadmap and twenty-seven queries of the transfer roadmap.

The memory footprint of the roadmap increases with the number of iterations and is the aggregate of all vertices and edges in the roadmap. The PRM* roadmap is the biggest and most dense. The IRS roadmaps have the same vertices as PRM* but fewer edges, while SPARS also removes nodes. The statistics are recorded for ten different random seeds and different roadmap sizes varying from 1,000 to 10,000 and 30,000 iterations for all three algorithms. Given the multi-modal nature of the manipulation roadmap, each iteration corresponds to an attempt to add both a transit and a transfer node to the roadmap. Thus, a PRM* roadmap after 10K iterations has 20K nodes and about 800K edges.

a) Query Resolution Figure 6 compares the methods in terms of running time. The cost of the lazy evaluation approach is affected by the presence of novel obstacles. The query resolution time is also affected by the size of the roadmap, which directly relates to the number of A* expansions as well as the probability of finding collision-free solutions. In most cases, SPARS outperforms IRS and IRS outperforms PRM* due to sparsity and reduced size. The relative performance of PRM* tends to degrade as bigger roadmaps are computed because the A* calls become increasingly more expensive. As the number of obstacles increases, these results are affected by a lower success rate.

An incremental A^* approach was also tested instead of the lazy method, where the heuristic search directly explores the roadmap for the shortest collision-free path. This approach results in significantly slower online query resolution times and the corresponding data are not reported here.

b) Solution length: The average solution length, shown in Figure 7, gives a measure of achieved path quality. PRM* generates roadmaps of increased connectivity and, as expected, achieves the best path quality. Despite having a lot fewer vertices and edges, SPARS outperforms IRS in many cases. Overall, the path degradation is smaller than the stretch factor of the methods used ($t = 2$) across the various environments. As the number of novel obstacles increases, the near-optimality guarantees are not valid but the path degradation does not exceed the stretch factor.

c) Success ratio: The success ratio, shown in Fig. 8, is lower for smaller roadmaps for scenes where there are novel objects. The easy environment is solved by all three methods. The success ratio improves, however, as the amount of precomputation increases, motivating the need for investigating appropriate roadmap precomputation schemes. The PRM* and IRS methods exhibit similar success ratio across

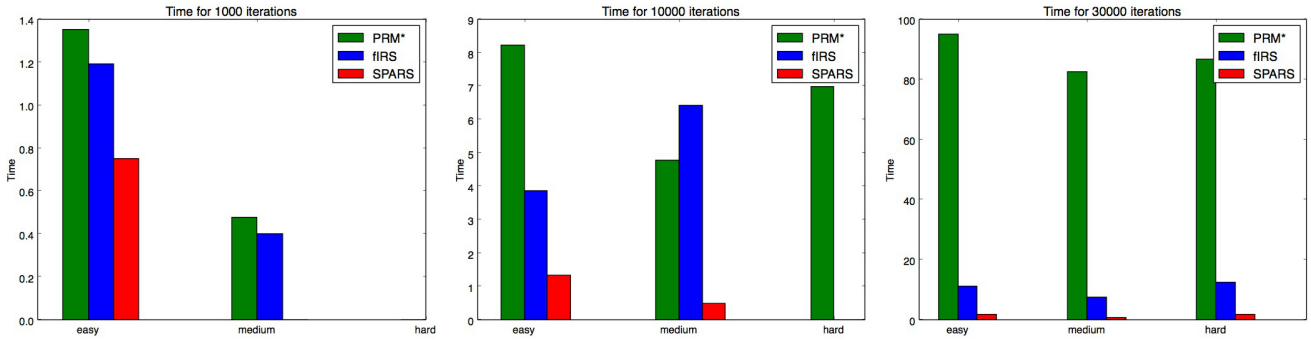

Fig. 6. Online query resolution time for returned paths given a precomputed roadmap after (left) 1,000, (medium) 10,000 and (right) 30,000 iterations. There is a small difference between the running time of the methods when the input roadmaps are small. But when the roadmaps become larger, i.e., when enough precomputation is available so that the hard problems are also solved, the benefits of the sparse representations become significant.

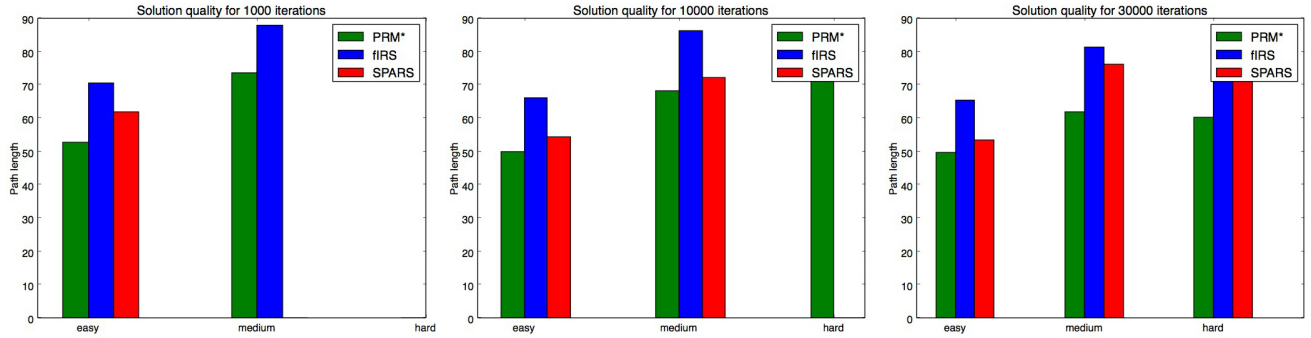

Fig. 7. Path quality of returned paths given a precomputed roadmap after (left) 1,000, (medium) 10,000 and (right) 30,000 iterations. There is some small degradation in path quality for the paths returned by the sparse roadmaps but the path degradation is significantly lower than the theoretical guarantees.

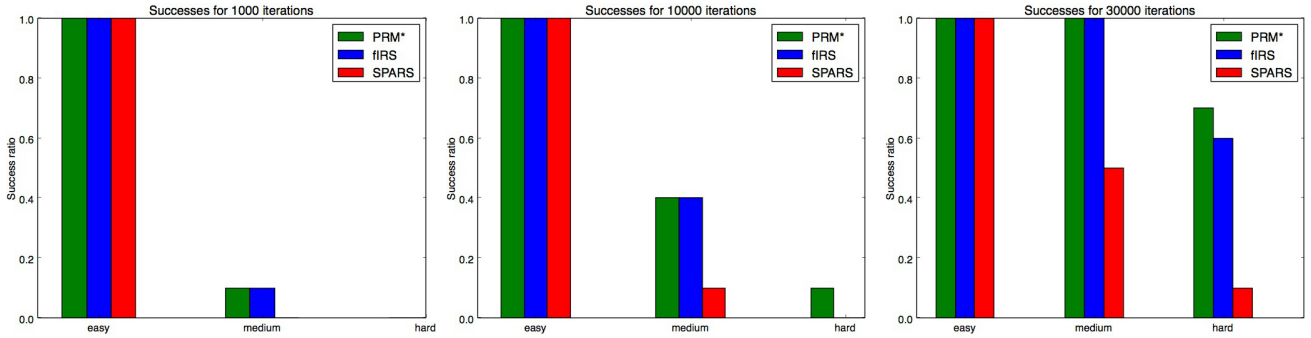

Fig. 8. Success ratio of online query resolution given a precomputed roadmap after (left) 1,000, (medium) 10,000 and (right) 30,000 iterations. Given small roadmaps, only the easy environment can be solved, but for larger precomputed roadmaps the success ratio increases for the harder challenges. IRS has equivalent success ratio with PRM* but the success ratio of SPARS suffers in harder instances.

the different environments. The success ratio of the SPARS method, however, suffers in the harder environments. This is reasonable as this is the sparsest roadmap returned.

d) Construction time The average construction time depends upon both the number of iterations and the amount of computation performed in every iteration. A fast IRS iteration is typically faster than a PRM* one, and this results in a reduced construction time for IRS. SPARS introduces a small computational overhead for reasoning about which edges and vertices to retain but it is still in the order of PRM* as it benefits from the sparsity of the computed data structure. The construction is a process that can also be parallelized on a computing cloud so as to increase the frequency with which the roadmap can be updated.

e) Communication overhead Once the roadmap is computed, it needs to be transmitted to the local workstation. This involves serializing the data structure, communicating it using individual ROS messages and then deserializing it. Figure 9 provides the time it takes to perform these operations for roadmaps of different sizes computed by the varying algorithms. These times are dominated by the serialization and deserialization process. The actual communication costs are significantly lower. The data suggest a major benefit in computing light-weight data structures in terms of effectively communicating them to a robot from a remote machine. For PRM* roadmaps it was not even possible to transmit them in individual ROS messages above a certain size (more than $5K$ iterations) and a more complex communication process is needed. The roadmap edges are storing information, i.e., a sequence of configurations, which is useful during the online resolution but increase the storage and communication overhead of dense roadmap solutions.

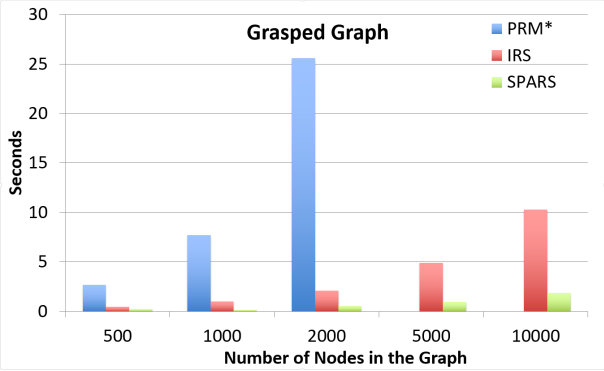

Fig. 9. Cost of transmitting the data structure between two remote machines for roadmaps of different sizes computed by different approaches.

V. Discussion

This work emphasizes the emerging opportunities in industrial automation due to the availability of computing as a service with the advent of cloud computing. The objective is to motivate efforts at the intersection of motion planning and cloud computing, especially in the context of flexible, adaptive factory floors that employ modern, compliant manipulators.

This work proposes a synergistic cloud-robot process for addressing manipulation problems, where the computational load is split between the local workstation and the remote cloud facility. In this process, standard motion planning roadmaps [8], [9] can be seen as the main form of interaction between the cloud and individual robots. The capability of roadmaps to effectively answer path planning queries depends on their size and density. Recently proposed algorithms provide compact alternatives with near-optimality guarantees [5], [15], [20], [24]. The evaluation of these methods presented here shows the benefits of investing computational power to compute roadmap data structures on the cloud, which are then used to answer motion planning queries on the robot's local workstation given an updated model of its surroundings. In this context, it was shown that compact, near-optimal roadmaps decrease the cost of communication and result in faster query resolution. They also result in some path quality degradation, which is typically lower than the theoretical guarantees they provide. For the sparsest alternative, which results in the fastest query resolution, there is also degradation in success ratio. This motivates the development of methods that balance sparsity and adaptability to scene changes.

It is interesting to further investigate the best form of interaction between a cloud and manipulators. Direct comparisons with the cloud-only computation mode will be interesting so as to quantify the benefits of the proposed synergistic approach. Furthermore, the existing results motivate the development of hierarchical query resolution schemes. This involves sparse roadmaps available in the robot's local workstation, which can quickly answer queries when successful. When the sparse roadmaps fail, then the robot's workstation communicates with the cloud to update the cloud's representation of the world and query the denser data structure stored there. Thus, cloud-based methods that continue the exploration of the robot's configuration space given the latest scene update and fix the roadmap on the fly will be useful. Real-world, long-duration experiments that integrate a physical manipulation platform with a computing cloud will demonstrate the feasibility of the proposed synergistic computation model and highlight its efficiency in practice. This will open the door for integrating the above methods with approaches for dealing with uncertainty, as well as long-term learning through cloud-based processes.

References

[1] M. Armbrust, I. Stoica, M. Zaharia, A. Fox, R. Grith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, and A. Rabkin. A View of Cloud Computing. *Communications of the ACM*, 53(4), April 2010.

[2] S. Berard, J. Trinkle, B. Nguyen, B. Roghani, J. Fink, and V. Kumar. daVinci code: A Multi-model Simulation and Analysis Tool for Multibody Systems. In *IEEE IROS*, 2007.

[3] D. Berenson, P. Abbeel, and K. Goldberg. A Robot Path Planning Framework that Learns from Experience. In *IEEE ICRA*, 2012.

[4] Y.-Y. Chen, J.-F. Wang, P.-C. Lin, P.-Y. Shih, H.-C. Tsai, and D.-Y. Kwan. Human-Robot Interaction based on Cloud Computing Infrastructure for Senior Companion. In *IEEE TENCON*, 2011.

[5] A. Dobson and K. E. Bekris. Sparse Roadmap Spanners for Asymptotically Near-Optimal Motion Planning. *IJRR*, 33(1), 2014.

[6] A. Dobson, A. Krontiris, and K. E. Bekris. Sparse Roadmap Spanners. In *WAFR*, 2012.

[7] C. Goldfeder, M. Ciocarlie, and P. K. Allen. The Columbia Grasp Database. In *IEEE ICRA*, pages 1710–1716, May 2009.

[8] S. Karaman and E. Frazzoli. Sampling-based Algorithms for Optimal Motion Planning. *IJRR*, 30(7):846–894, 2011.

[9] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. "probabilistic roadmaps for path planning in high dimensional configuration spaces". *IEEE TRA*, 12(4):566–580, 1996.

[10] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg. A Survey of Research on Cloud Robotics and Automation. *IEEE T-ASE*, (conditional) 2014.

[11] B. Kehoe, D. Warrier, S. Patil, and K. Goldberg. Cloud-Based Grasp Analysis and Planning for Toleranced Parts Using Parallelized Monte Carlo Sampling. *IEEE T-ASE*, (conditional) 2014.

[12] A. Kimmel, A. Dobson, Z. Littlefield, A. Krontiris, J. Marble, and K. E. Bekris. Pracsys: An extensible architecture for composing motion controllers and planners. In *SIMPAR*, 2012.

[13] J. Kuffner. Cloud-Enabled Robots. In *IEEE-RAS Int. Conf. on Humanoid Robots*, 2010.

[14] S. M. LaValle and J. Kuffner. Randomized Kinodynamic Planning. *IJRR*, 20(5):378–400, May 2001.

[15] J. Marble and K. E. Bekris. Asymptotically Near-Optimal Planning with Probabilistic Roadmap Spanners. *IEEE TRO*, 29(2), 2013.

[16] G. Mohanarajah, D. Hunziker, M. Waibel, and R. D'Andrea. Rapyuta: A Cloud Robotics Platform. *IEEE T-ASE*, accepted 2014.

[17] T. Niemueller, S. Schiffer, G. Lakemeyer, and S. Rezapour-Lakani. Life-long Learning Perception using Cloud Database Technology. In *IROS 2013 - Cloud Robotics Workshop*, 2013.

[18] J. Pan, S. Chitta, and D. Manocha. FCL: A General Purpose Library for Collision and Proximity Queries. In *IEEE ICRA*, 2013.

[19] D. Peleg and A. Schaffer. Graph Spanners. *Journal of Graph Theory*, 13(1):99–116, 1989.

[20] D. Shaharabani, O. Salzman, P. K. Agarwal, and D. Halperin. Sparsification of Motion-Planning Roadmaps by Edge Contraction. In *IEEE ICRA*, 2013.

[21] T. Simeon, J.-P. Laumond, J. Cortes, and A. Sahbani. Manipulation Planning with Probabilistic Roadmaps. *IJRR*, 23(7-8):729–746, 2004.

[22] A. Sorokin, D. Berenson, S. S. Srinivasa, and M. Hebert. People helping Robots helping People: Crowd-sourcing for Grasping Novel Objects. In *IEEE/RSJ IROS*, pages 2117–2122, October 2010.

[23] M. Waibel. et al. RoboEarth. *IEEE Robotics & Automation Magazine*, 18(2):69–82, June 2011.

[24] W. Wang, D. J. Balkcom, and A. Chakrabarti. A Fast Streaming Spanner Algorithm for Incrementally Constructing Sparse Roadmaps. In *IEEE IROS*, pages 1257–1263, 2013.

[25] P. R. Wurman, R. D'Andrea, and M. Mountz. Coordinating Hundreds of Autonomous Vehicles in Warehouses. *AI Magazine*, 2008.